THEORY OF COMPUTATION

1 FA & RE.

* TOC: 4 Models (Syllabus)

VI
- → FA & RE.
- → Push down automata & CFG.
- → RL, CFLS, Pumping Lemma
- → Turing Machines & Undecidability.

about 60%

direct Qs

* Theory of Computation:

- designing some math models.
- Study of Some math models/ m/c.

FA ⟶ PDA ⟶ TM ⟶ LBA.

- Machines & Languages

* Language

- start from alphabet.

1] Alphabet : symbols (finite no. of symbols)
→ finite, non-empty set of symbols (limited).

- (Σ)

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

$\Sigma = \{a\}$

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

2] Power of an alphabet : → Kleen closure → Positive closure.

$\Sigma^0, \Sigma^1, \Sigma^2 \cdots \Sigma^n$.

→ length of an alphabet.

$a^0, a^1, a^2, a^3 \cdots a^n$

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

(Λ) ε , a, aa, aaa .....
(empty string/ empty char)

- len of a is 0. → if power is 0.

3] Σ^* → Kleen closure/Kleen star.
→ 0/more occurrences.

- $\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cdots \cup \Sigma^n$.

$\Sigma^* = \{\varepsilon, a, aa, aaa \cdots\}$ if $\Sigma = \{a\}$

4] Positive closure of alphabet

Σ^+ (1/ more occurrences).

- $\Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \cdots \cup \Sigma^n$.

if $\Sigma = \{a\}$

$\Sigma^+ = \{a, aa, aaa \cdots\}$.

∴ $\boxed{\Sigma^+ = \Sigma^* - \{\varepsilon\}}$ → VI.

5] collection of alphabets

String/ Word (w)

- collection of characters/alphabets.

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

abc → invalid word (W)
ab → valid string/ word (W)
abba → valid word

6] Operations on strings:

→ length of a string
→ prefix of a string.
→ suffix of a string.
→ proper prefix/ suffix of a string.
→ Substring of a string
→ Reverse of a string
→ palindrome of a string.
→ union of 2 strings.
→ concatenation of 2 Strings

(i) Length of String

– no. of char in the string.

- $w = abc$. $\Sigma = \{a, b, c\}$

$L = |W| = 3$

- $w = \varepsilon$

$L = |W| = 0$

(ii) Prefix of a String → leading characters of every string

- $w = abc$

→ ε is prefix of every string.

$\Rightarrow \{\varepsilon, a, ab, abc\}$

- $W = abcd$.

$\Rightarrow \{\varepsilon, a, ab, abc, abcd\}$

(iii) Suffix of a String → trailing char.

$\Rightarrow$ ε is suffix of every string.

Scanned with CamScanner

non-empty proper = n−1

- w = abc

⇒ {ε, c, bc, abc}

- w = abcd

⇒ {ε, d, cd, bcd, abcd}

Q If L = n, how many no. of possible prefix/suffix?

→ n+1

(iv) Proper prefix/suffix set

→ ignore the word from set

- w = abc

⇒ {ε, a, ab} (Prefix)

- w = abc

⇒ {ε, c, bc} (Suffix)

Q If L = n, how many proper prefix/suffix?

→ n

(v) Substring of a String

→ take any prefix/suffix set

- w = abc

Prefix = {ε, a, ab, abc}

{ε, a, b, ab, c, bc, abc}

→ Substring set

- w = abc

Suffix = {ε, c, bc, abc}

⇓ Prefix

⇒ {ε, c, b, bc, a, ab, abc}

→ Substring Set

- w = abcd

Prefix = {ε, a, b, c, d, ab, bc, cd, abc, bcd, abcd}

Proper Substrings $= \frac{n(n+1)}{2}$

Non-empty $= \frac{n(n+1)}{2}$

Proper Non-empty $= \frac{n(n+1)}{2} - 1$

- $W = abcd$

Prefix $= \{\varepsilon, a, ab, abc, abcd\}$

Suffix.

$\Rightarrow \{\varepsilon, a, b, ab, c, bc, abc, d, cd, bcd, abcd\}$

Q. If the string length is n, how many strings? (Substrings)

$$\boxed{\frac{n(n+1)}{2} + 1}$$

Derive?

- $L=1 \longrightarrow 4$
- $L=2 \longrightarrow 3$
- $L=3 \longrightarrow 2$
- $L=4 \longrightarrow 1$

$\frac{n(n+1)}{2} + 1$ ↳ ε

$(4+3+2+1)+1$.

Q. Non-zero substrings

$\frac{n(n+1)}{2}$ ← VI.

(vi) Reverse of a String (w^R)

$w = abc \longrightarrow cba$

$w = \varepsilon \longrightarrow \varepsilon$

if $w = abc$, $w^R = cba$

(vii) Palindrome of a String

if, $\boxed{w == w^R}$

eg: $aba = w$, $w^R = aba$ ✓

$w = aabb$, $w^R = bbaa$ X.

(viii) Union of String (+, ∪, |)
3 symbols.

eg: $w_1 = ab$, $w_2 = cd$

$w_1 + w_2 = \{ab, cd\}$ or $\{cd, ab\}$.

(ix) **Concatenation (·)**

eg: $w_1 = ab$, $w_2 = cd$

$w_1 \cdot w_2 = \{abcd\}$.

$w_2 \cdot w_1 = \{cdab\}$

~~* Set of Str~~

Language (L)

· $\boxed{L \subseteq \Sigma^*}$ if $\Sigma = \{0\}$.

$\Sigma^* = \{\varepsilon, 0, 00, 000 \ldots$

Subset

- collection of strings which are derived from the alphabet Σ.

· Based on no. of strings.

Finite language

Infinite language

eg: $L = \{W \mid |W| \leq 3, w \in 0^*\}$

$\Rightarrow \{\varepsilon, 0, 00, 000\}$ ∴ 4 Strings

eg: $L = \{W \mid |w| \geq 0, w \in 0^*\}$.

$\Rightarrow \{\varepsilon, 0, 00, 000 \ldots\}$ ∴ ∞ strings.

Finite Automata

- Automata = self automatic.

· 5-tuples.

- finite no. of states / i/p symbols.

(FSM)

finite/ non-empty set of states

- $M = \{Q, \Sigma, q_0, \delta, F\}$

transition function

eg: lift

- States = Floors (Q)
- Σ = buttons (input symbols)
- q_0 = initial floor (init. state) (string proc. starts from here)
- δ = transition/ mapping function

$F\text{-}1 \xrightarrow[(2)]{} F\text{-}2$

$$\boxed{\delta = Q \times \Sigma \rightarrow Q}$$

$F1 \times 2 \longrightarrow F2$.
$F3 \times 4 \longrightarrow F4$

- F (Final/ Acceptance State) — must in FA without output

we can decide if the string is valid/ not.

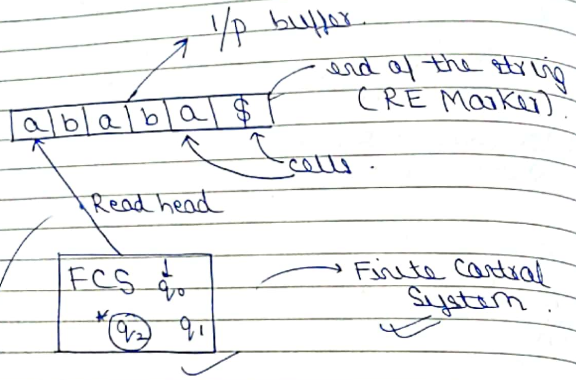
VI

13

** Finite Automata is classified into: (based on o/p concept)

- FA without output (Language acceptor/ language recogniser)
  - DFA
  - NFA
    - NFA with ε transition (ε-NFA)
- FA with output (output generators/ transducers)
  - convert from 1 energy to another so this name.
  - (No need of final states here.)
    - Millay
    - Moore

* Finite Automata Model

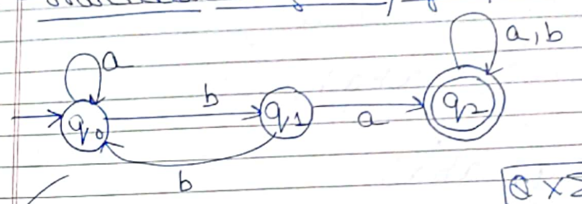


VI
- 1 char can be read at a time
- (Left → Right only)
- unidirectional.
- $ → end of the string (Now check)

(If last state → final ∴ string ✓)
 if not ∴ string ✗)

* FA → Transition diagram/graph.
 → Transition table

Transition diagram/graph



$\boxed{Q \times \Sigma \rightarrow Q}$

- edges = input symbols (Σ)
- vertex = Q

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

$\Rightarrow (\{q_0, q_1, q_2\}, \{a, b\}, q_0, q_2)$

$\delta : Q \times \Sigma \rightarrow Q$ → DFA

$\delta(q_0, a) = q_0$
$\delta(q_0, b) = q_1$
$\delta(q_1, a) = q_2$
$\delta(q_1, b) = q_0$
$\delta(q_2, a) = q_2$
$\delta(q_2, b) = q_2$

buffer value (read from left to right)

$\delta(q_0, ababa)$

$\downarrow$

$\delta(q_0, baba)$

$\downarrow$

$\delta(q_1, aba)$

$\downarrow$

$\delta(q_2, ba)$

$\downarrow$

$\delta(q_2, a) = q_2 \cdot (F)$

$\downarrow$

$\delta(q_2)$ ✓

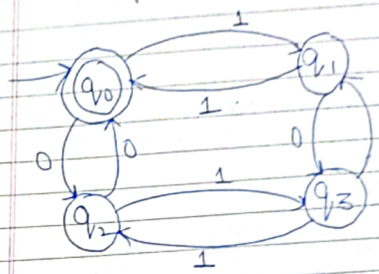
∴ if $\boxed{\delta(q_0, W) \in F}$ ∴ valid String.

∴ abb → invalid.

* Transition Table

δ	a	b
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_0
$*q_2$	q_2	q_2

Q Verify whether the following strings are accepted by this DFA or not.



(i) 10101 X

(ii) 010101 X

(iii) 1010110 X.

(iv) 110011 ✓

* DFA (Deterministic Finite Automata)

- we can identify by using transitions
- from each state, we should mention the path of each i/p symbol.
- For 1 i/p symbol, to a distinct state.

Appln of FA : Designing of [Lexical Analyser]

- If no path / more than 1 path → NFA.

language accepter / recogniser.

- DFA/NFA → both are desc. by 5 tuples only

DFA : $\delta : Q \times \Sigma \rightarrow Q$

(Q = all states, Σ = i/p symbols, Q = next states)

1 final state is req. for checking validity of string.

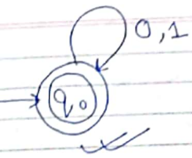
F → set of final states.

$F \subseteq Q$

Q. To construct a DFA which accept the strings any no. of 0s and any no. of 1s where : $\Sigma = \{0,1\}$

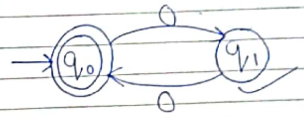
$\Sigma = \{0,1\}$

$L = \{\varepsilon, 0, 1, 01, 10, 00, 11, 010 \dots\}$



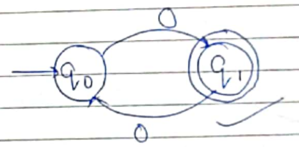
Q. which accepts even no. of 0s where, $\Sigma = \{0\}$

$L = \{\varepsilon, 00, 0000 \dots\}$

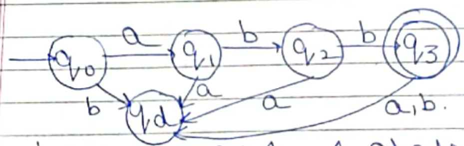


Q. odd no. of 0s . $L = \{0, 000, 00000 \dots\}$

$L = 1 \; (\Sigma 1)$
∴ 2 states

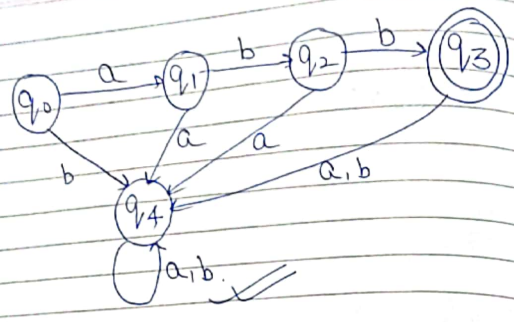


Q. Construct DFA to accept the string abb only $\Sigma = \{a,b\}$



$L = 3$, ∴ $n+1 = 4$ states

qd → dead/trap state.

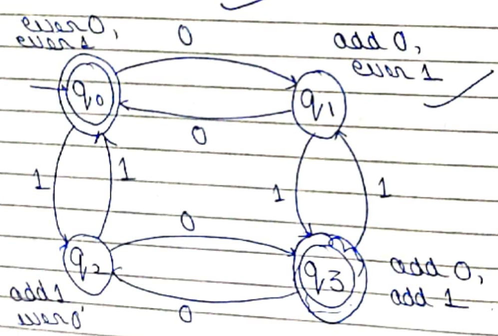


• if only 1 string : n+2 states
eg: abb = 5 states

Q Construct a DFA which accept all strings of even no. of 0s and even no. of 1s.

(ii) even 0s, odd 1s.
(iii) odd 0s, even 1s
(iv) odd 0s, odd 1s.

(i)



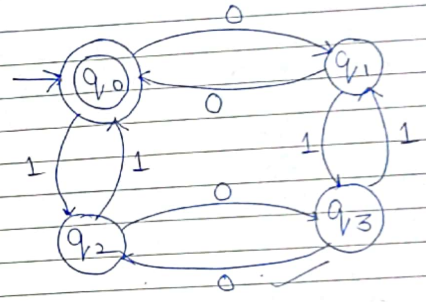
(iii) $L_{EE} = \{\varepsilon, 00, 11, 0101 \ldots\ldots\}$.

$L_{EO} = \{1, 001, 0101 \ldots\ldots\}$.

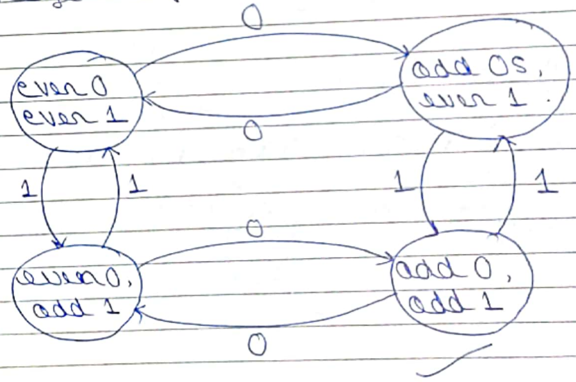
$L_{OE} = \{0, 01001, \ldots\ldots\}$.

$L_{OO} = \{01, 10, 010101 \ldots\}$.

(ii)



Change the final state for (ii), (iii), (iv).

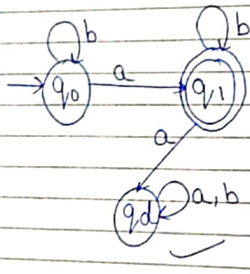


Q Construct a DFA which accept all strings of a s and b s having:

(i) exactly one a
(ii) atleast one a
(iii) not more than 3 a s

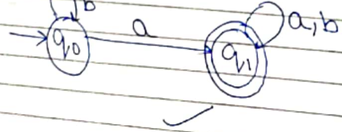
(i) $\Sigma = \{a, b\}$

(i) $L = \{a, bab, bbab, babb \cdots ba\}$



(ii) atleast one a.

$L = \{a, aa, baab, abb, aab \cdots\}$



(iii) not more than 3 a

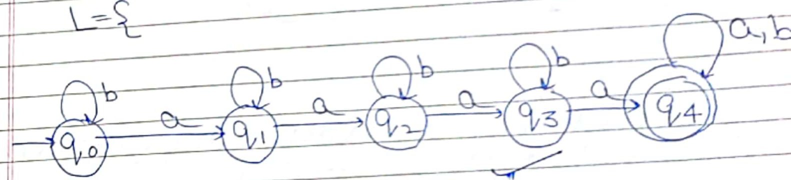
$L = \{\varepsilon, a, aab, baaabb, abb \cdots\}$

ababab a X.

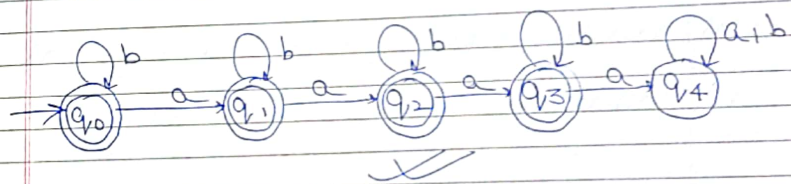
not

→ construct more than 3 as. $\Sigma = \{a, b\}$

$L = \{$

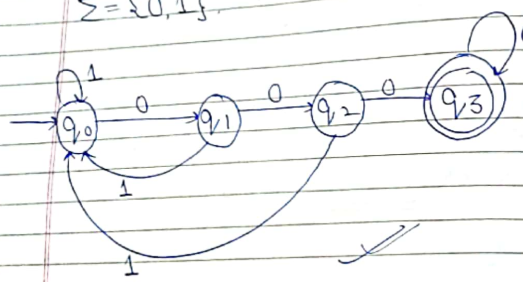


. not more than 3 a. s



Q. Construct a DFA which accept all strings of 0s and 1s, having the substring 000.

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



Q. ~~has~~ ~~having~~ the last two symbols same in the string. — (i)

~~(ii)~~ last two symbols are different

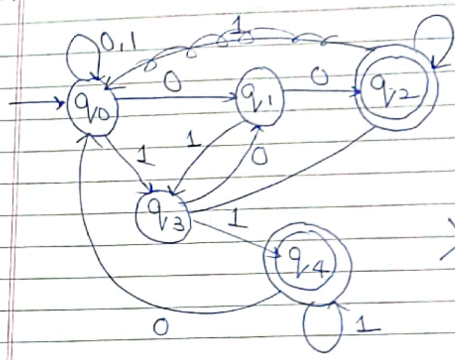
(iii) starting & ending with same symbol.

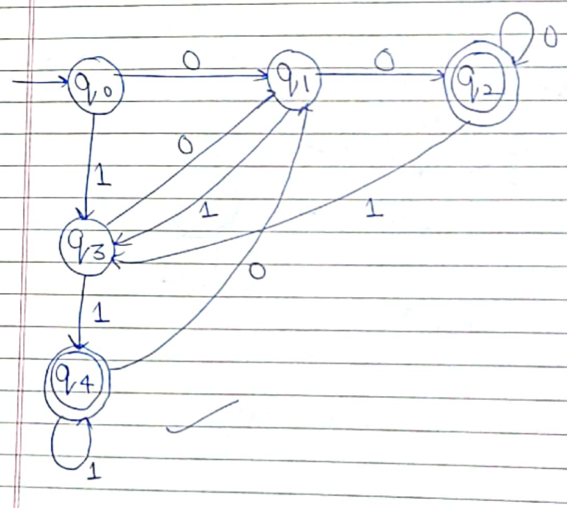
(iv) starting & ending with diff symbol.

0011 ✗

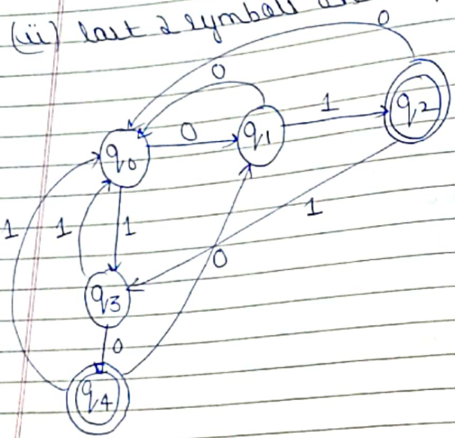
Try:

(ii) $L = \{00, 11, 011, 100, \dots\}$

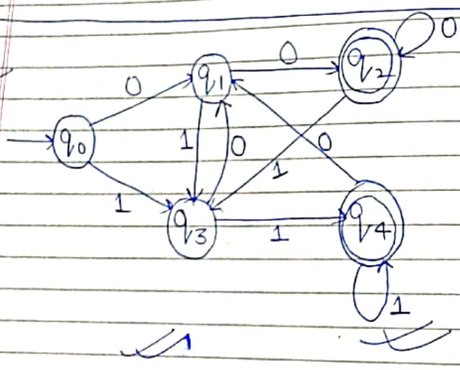




(iii) last 2 symbols are diff.

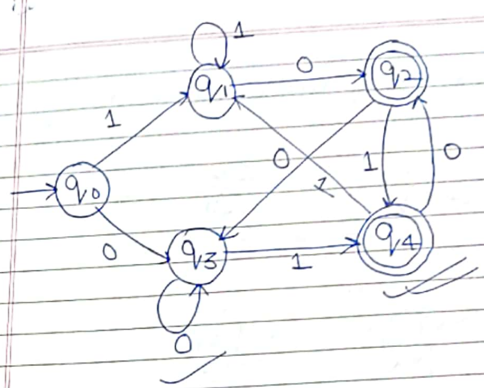


(ii)



[last 2 symb are same]

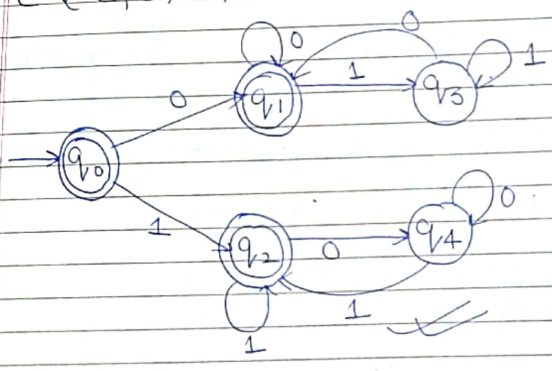
(ii) last 2 symbols are different.



(iii) starting & ending with same symbol

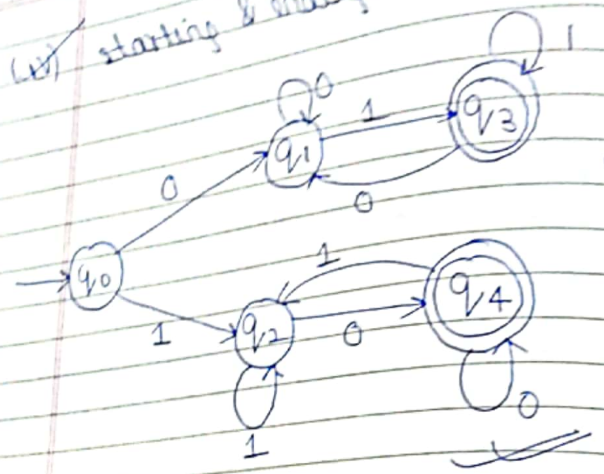
L = { ε, a, b, aba, bab, abbaba ... }

L = { ε, 0, 1, 010, 101, 0110100 ... }



L = { ε, 0, 1, 00, 11, 010, 101, 0110 ... }

(iv) starting & ending with different sym



**

Q. which of the following is recognized by DFA?

(a) The Set $\{1, 2, 4, 8 \ldots 2^n\}$ are wri... in binary

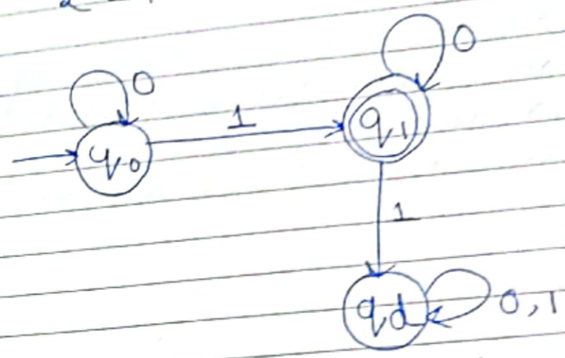
(b) $\{1, 2, 4, 8 \ldots 2^n$ written in unary

(c) $\{0, 101, 11011, 1110111 \ldots\}$

(d) The Set has equal no. of a's and equal no. of b's

FA → No memory to ... [biggest drawback]

(a) $\{1, 10, 100, 1000, 10000 \ldots\}$



∴ Valid DFA

1, 2, 4, 8, 16

(b) ⇒ $\{1, 11, 1111, 11111111 \ldots\}$

(if finite long. → DFA ...

· infinite language

· common diff. dne

1, 2, 4, 8, 16 ....

(c) $\{0, 101, 11011, 1110111 \ldots\}$ → $a^n 0 a^n$

there is no bidirectional possib. in FA

∴ same no. of 1 & 0

(d) cant check. (no extra memory)

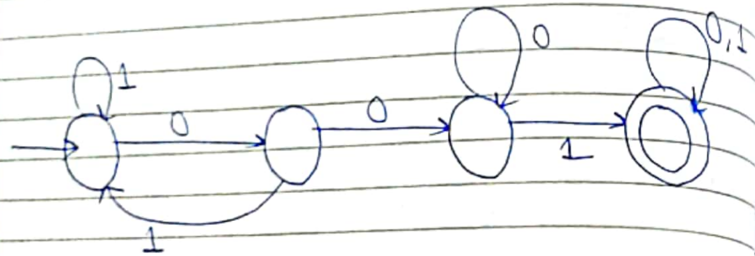
$\boxed{\because \text{unidirectional memory}}$

Q. Construct the following DFA: M

→ Let S denote the 7 bit binary strings in which first, fourth & last bits are 1.
The no. of strings in S that are accepted by M is:

(a) 1
(b) 5
(c) 7
(d) 8

001



1001001 (1)

1101001 (2).

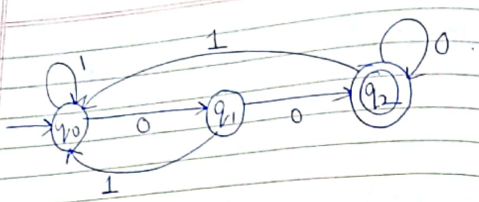
4 rem positions

$\Rightarrow 2^4 = 16$ possible strings.

1 _ _ 1 _ _ 1 → 16 combinations

1 0 0 1 0 0 1 ✓
1 0 0 1 0 1 1 ✓
1 0 0 1 1 0 1 ✓
1 0 0 1 1 1 1 ✓
1 0 1 1 0 0 1 ✓
1 0 1 1 0 1 1
1 0 1 1 1 0 1
1 0 1 1 1 1 1
1 1 0 1 0 0 1 ✓
1 1 0 1 0 1 1
1 1 0 1 1 0 1
1 1 0 1 1 1 1
1 1 1 1 0 0 1 ✓
1 1 1 1 0 1 1
1 1 1 1 1 0 1
1 1 1 1 1 1 1

= 7

Q 

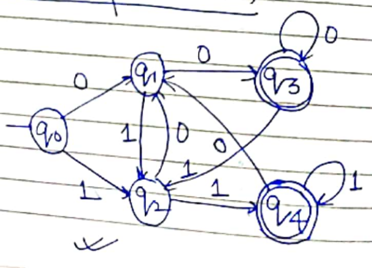
The following DFA accept the set of all strings over {0, 1}

(a) begins either 0 or 1.
(b) ends with 0.
(c) ends with 00.
(d) contains the substring 00. (001X)

Q Let L be the set of all binary strings whose last two symbols are the same. The no. of states in the minimal state DFA accepting the language L is:

(a) 2
(b) 5
(c) 8
(d) 3.

(given problem)



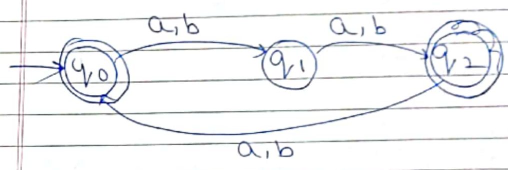
multiples of 3

Q Construct DFA for the following languages.

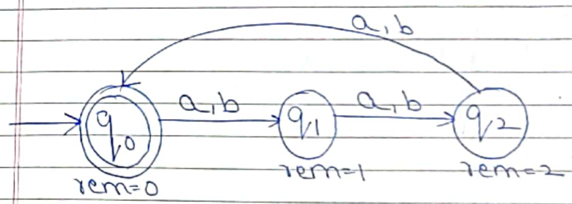
$L = \{W \mid |W| \bmod 3 = 0,\ \omega \in (a,b)^*\}$

$|W| \% 3 = 0$

{ε, aaa, bbb, abb, aab, aba ..}



• mod n ∴ n states

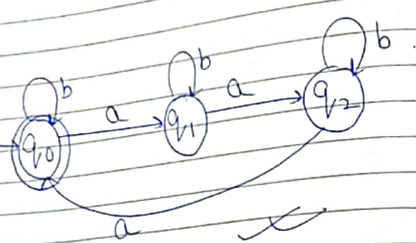


• How many min no. of states? = 3

if $|W| \bmod 3 = 1 \rightarrow q_1 = F$
$|W| \bmod 3 = 2 \rightarrow q_2 = F$.

no. of a

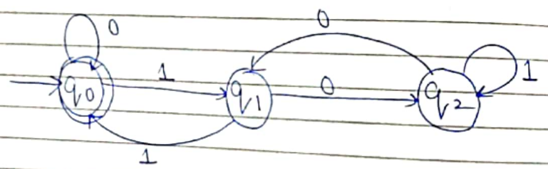
Q $L = \{w \mid n_a(w) \bmod 3 = 0,\ w \in (a,b)^*\}$



Q Construct a DFA which accepts all binary strings that are divisible by 3.

∵ rem = 3

	0	1
*q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2



8 4 2 1
1001
10101

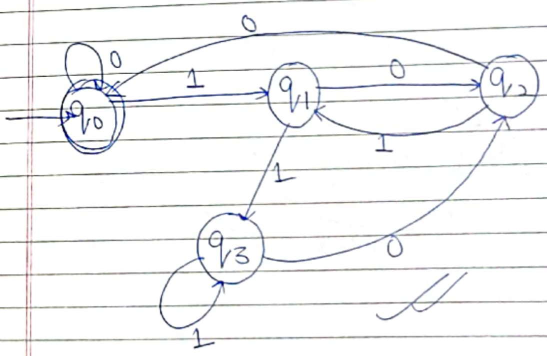
01

010
011
8 4 2 1
0100 = 1.
0101 = 2.

Q mod 4 (Construct a DFA which accepts all binary strings that are divisible by 4)

	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3

← transition table



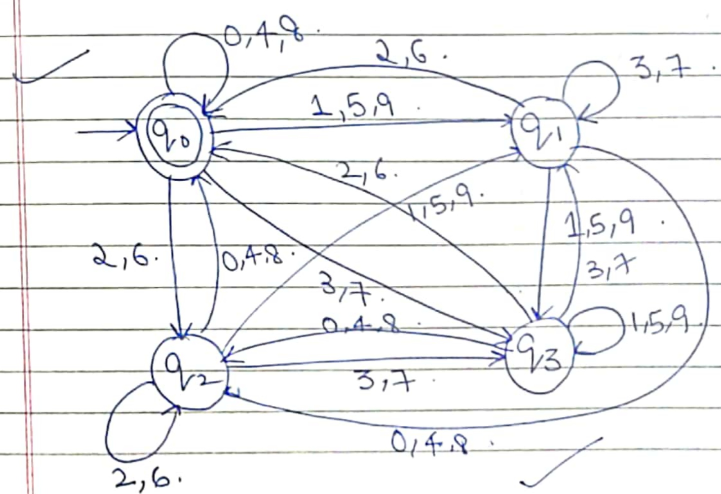
$0 \rightarrow \{0,3,6,9\}$.
$1 \rightarrow \{1,4,7\}$
$2 \rightarrow \{2,5,8\}$.
→ Groups.

$0\%3=0$.
$1\%3=1$.
$2\%3=2$
$3\%3=0$
$4\%3=1$
$5\%3=2$
$6\%3=0$
$7\%3=1$
$8\%3=2$
$9\%3=0$.

- Language is infinite.

Q. if mod 4 : q_0, q_1, q_2, q_3
(decimal nos divisible by 4)

	0	1	2	3	4	5	6	7	8	9
$\rightarrow * q_0$	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1
q_1	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3
q_2	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1
q_3	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3.



$0 \rightarrow \{0, 4, 8\}$
$1 \rightarrow \{1, 5, 9\}$
$2 \rightarrow \{2, 6\}$
$3 \rightarrow \{3, 7\}$

[if mod n ⟶ n states are required]

Q. Construct a DFA which accepts all binary strings which are divisible by 2 or divisible by 3.

[div. by m or div. by n = m×n states]

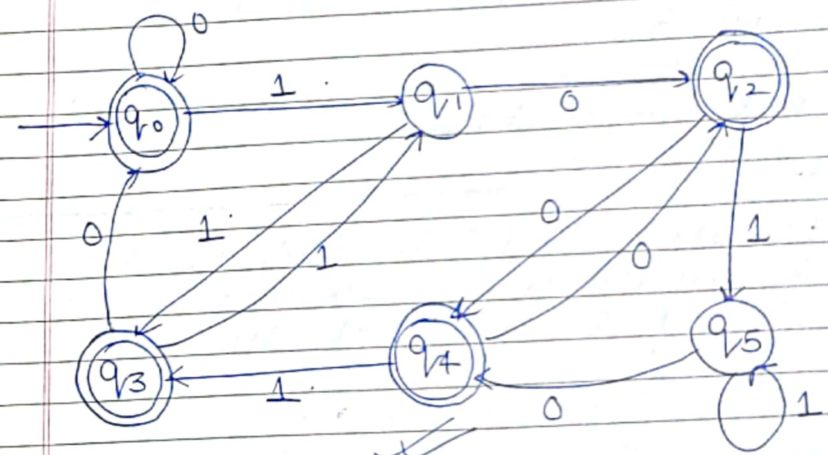
	0	1
$\rightarrow q_0^*$	q_0	q_1
q_1	q_2	q_3
q_2^*	q_4	q_5
q_3^*	q_0	q_1
q_4^*	q_2	q_3
q_5	q_4	q_5

VI

$L_2 = \{0, 2, 4, 6 \ldots\}$ (div. by 2)

$L_3 = \{0, 3, 6, 9 \ldots\}$. (div. by 3).

• $L_2 \cup L_3 = \{0, 2, 3, 4, 6, 8, 9 \ldots\}$



eg: 9 (1001) (3)
(1010) = 10 (2)
(1011) = 11 ∴ rejected

Q. The smallest finite automata which accept the language:

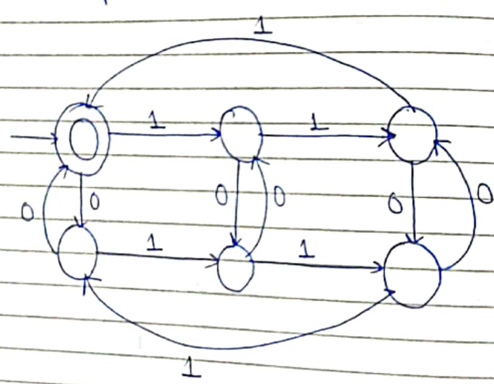
$L = \{ x \mid |x| \text{ is divisible by } 3 \}$

has:

(i) 2 states
(ii) 3
(iii) 4
(iv) 5.

mod n = n states.

Q. The following FSM accept all those binary strings in which the no. of 1s & no. of 0s are respectively:



1 0

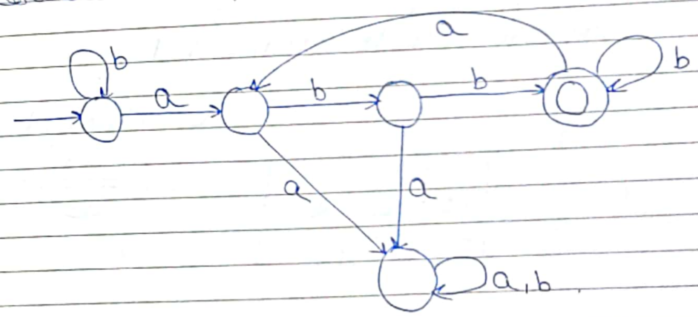
(a) divisible by 3, divisible by 2
(b) odd & even
(c) even & odd
(d) divisible by 2 & 3

Soln: 1s are divisible by 3;
0s are divisible by 2.

eg: 10101

w = { abbbb, abbabbb }

Q. Consider the following FA (M).



The language accepted by M is:

(a) { w ∈ (a,b)*, every a in w is followed by exactly 2 bs }

(b) { w ∈ (a,b)*, every a in w is followed by atleast 2 bs }
(& also end with b).

(c) $\{w \in (a,b)^*$, w contain the substring as abb $\}$

(d) $\{w \in (a,b)^*$, w doesnot contain aa as substring $\}$

Soln.

(c) abba X. (not accepted)

(d) a, ε (not final state) X.

Q. Consider the following 2 FAs : (Product of 2 FAs)

Y:

	a	b
→1	1	2
*2	2	1

Z:

	a	b
→1	2	2
*2	1	1

∴ 4 states in (Z×Y)

Which of the following represents the product of automata : i.e. Z×Y

(a)

	a	b
→P	S	R
Q	R	S
R*	Q	P
S	P	Q

(b)

	a	b
P	S	Q
Q	R	S
R*	Q	P
S	P	Q

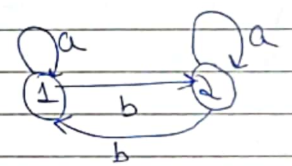
(c)

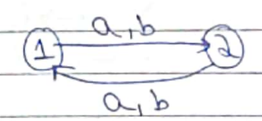
	a	b
→P	Q	S
Q	R	S
R*	Q	P
S	Q	P

(d)

	a	b
→P	S	Q
Q	S	R
R*	Q	P
S	Q	P

Y:





Solⁿ. $Z \times Y$.

eg: $(1,2) \times (2,2)$

$= (1,1), (1,2), (2,1), (2,2)$

$\boxed{Z \times Y}$ (Cross Product)

	a	b
→(1,1)	(2,1)	(2,2)
(1,2)	(2,2)	(2,1)
(2,1)	(1,1)	(1,2)
*(2,2)	(1,2)	(1,1)

new states

∴ (1 is start state; 2 is final state)

P → (1,1)
R → (2,2)
Q → (1,2)
S → (2,1)

(By order)

	a	b
→ P	S	R
Q	R	S
S	P	Q
R*	Q	P

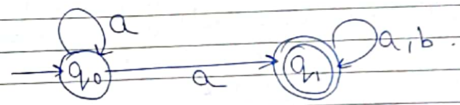
∴ (a)

Q. Non-Deterministic Finite Automata (NFA).

- 5 tuple notation

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

- around more than 1 path on same symbol



• $q_0(b) \to$ no path (zero transcend)
$q_0(a) \to q_0 / q_1$ (more than 1 transition)

(NFA is easier to construct) — less efficient than DFA.

* $M = (\underbrace{Q, \Sigma}_{\text{finite \& non empty}}, q_0, \delta, F)$

$\boxed{\delta : Q \times \Sigma \rightarrow Q}$

[Power Set]

$\boxed{\delta : Q \times \Sigma \rightarrow 2^Q}$ → NFA

max. limit on any i/p symbol : 2^Q

$\delta(q_0, a) \rightarrow \{\phi, q_0, q_1, q_0q_1\}$

$\delta(q_0, b) \rightarrow \{\phi\}$

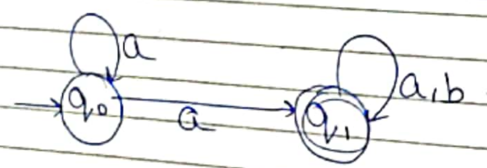
$\therefore \{\phi, q_0, q_1, q_0q_1\}$ → Max. limit

$2^q = 4$

* F → Set of final states

$\boxed{F \subseteq Q}$

* Strings accepted by NFA



a : accepted.

∴ [Verify all paths, if atleast 1 path goes to final state, then string will be accepted]

→ NFA acceptance.

→ verify all possibilities [Backtracking]

• DFA is more efficient than NFA (less time).

→ verify all possibilities till we get.

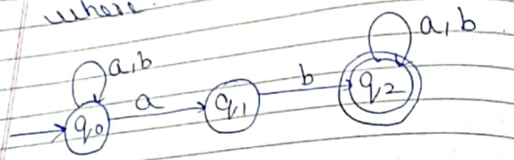
— For every NFA, we can construct eq. DFA

(NFA $\cong$ DFA)

For every NFA : eq. DFA $\boxed{\text{VI}}$

every DFA is NFA ; but every NFA is not DFA ; but can be converted to DFA

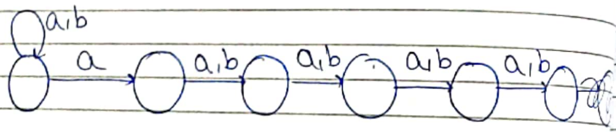
Q. Construct a NFA which accepts all strings, contain the substring as ab where $\Sigma = \{a, b\}$



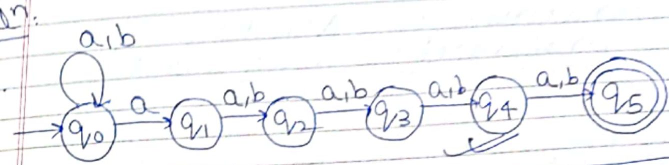
eg: aba = q_2

Q. Construct NFA that accepts all strings of as and bs where fifth symbol is a, while reading the string from the right end.

_ a _ _ _ _

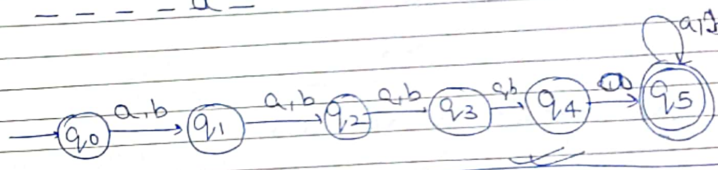


Soln:

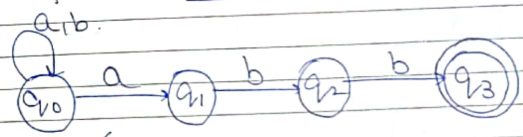


Q. left to right.

_ _ _ _ a _



Q. ending with abb.



NFA

* Conversion from NFA to DFA or Equivalence of NFA and DFA

· for every NFA, there is an equivalent DFA.

– If a procedure is available, (NFA → DFA)
definite algorithm
→ decidable problems

eg: hack your password. → Subset Construction method

* Conversion from NFA to DFA

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

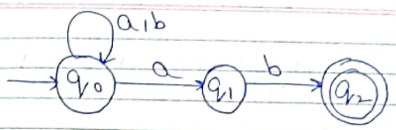
$\Downarrow$ $\boxed{VI}$

$M' = (Q', \Sigma, q_0, \delta', F') \longrightarrow$ DFA.

(Σ is fixed for NFA to DFA)
q_0 also remains same).

∵ init state & i/p symbol is same

Q/ strings ending with ab.

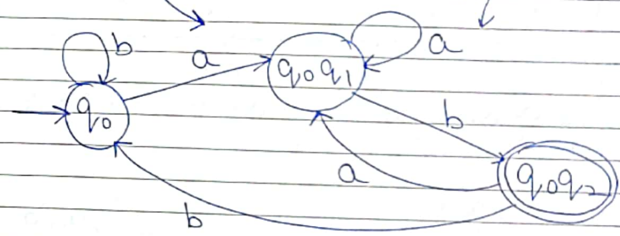


δ'	a	b
$\to q_0$	$\{q_0q_1\}$	q_0
q_0q_1	$\{q_0q_1\}$	$\{q_0, q_2\}$
$*q_0q_2$	$\{q_0q_1\}$	q_0

$\delta'(q_0q_1), a = \delta(q_0, a) \cup \delta(q_1, a)$

Subsets

DFA



$\delta'([q_0q_1], b) = q_0 \cup q_2 = q_0q_2$

GATE

Q. If NFA has n states, how many possible states in the DFA?

$\leq 2^n$. [Power Set].

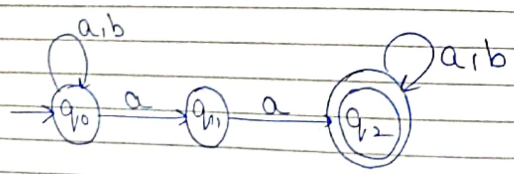
Proof:

if q_0, q_1, q_2

$\{\phi, q_0, q_1, q_2, q_0q_1, q_0q_2, q_1q_2, q_0q_1q_2\} \Rightarrow 2^3 = 8$

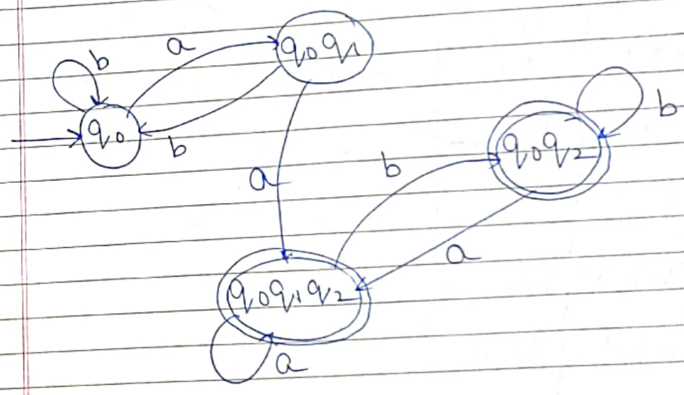
Max. no of states $\longrightarrow 2^n$.

Q. Construct an equivalent DFA to the NFA:



Subset Construction Method.

	a	b
$\rightarrow q_0$	q_0q_1	q_0
q_0q_1	$q_0q_1q_2$	q_0
$*q_0q_1q_2$	$q_0q_1q_2$	q_0q_2
$*q_0q_2$	$q_0q_1q_2$	q_0q_2



	a	b
$\rightarrow$ A	B	A
B	C	A
* C	C	D
* D	C	D

	a	b
A	B	A
B	CD	A
CD	CD	CD

{AB} {CD}

[A] [B] [CD]

54

Q

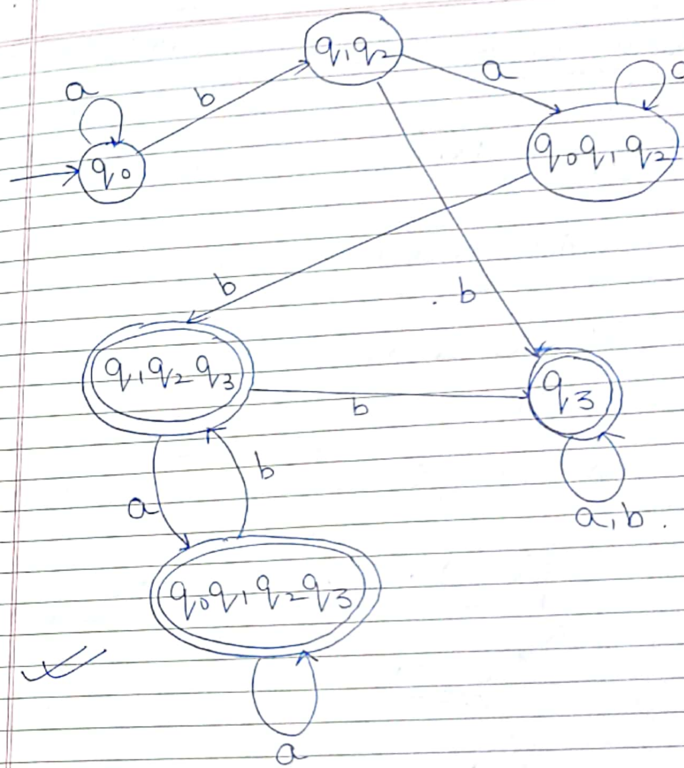
NFA

	a	b
$\rightarrow q_0$	q_0	$q_1 q_2$
q_1	$q_0 q_1$	ϕ
q_2	q_2	q_3
$*q_3$	q_3	q_3

DFA

	a	b
$\rightarrow q_0$	q_0	$q_1 q_2$
$q_1 q_2$	$q_0 q_1 q_2$	q_3
$q_0 q_1 q_2$	$q_0 q_1 q_2$	$q_1 q_2 q_3$
$*q_1 q_2 q_3$	$q_0 q_1 q_2 q_3$	q_3
$*q_0 q_1 q_2 q_3$	$q_0 q_1 q_2 q_3$	$q_1 q_2 q_3$
$*q_3$	q_3	q_3

55



∴ 6 States

DFA : $Q \times \Sigma$
NFA : $Q \times \Sigma$
ε-NFA : $Q \times (\Sigma \cup \{\varepsilon\})$

* NFA with ε Transitions :

ε-NFA

- In DFA → ε transitions not allowed.

* 5 tuple notation

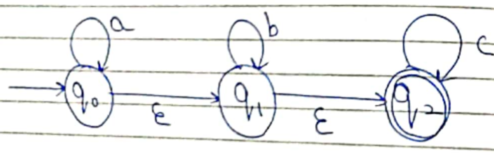
$M = (Q, \Sigma, q_0, \delta, F)$

- By reading ε, we can change state.

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

→ ε-NFA

eg.



$a \cdot \varepsilon = a$
$b \cdot \varepsilon = b$
$a \varepsilon b = ab$

$\delta(q_0, a) = \delta(q_0, a \cdot \varepsilon \cdot \varepsilon)$

$\downarrow$

$\delta(q_0, \varepsilon \cdot \varepsilon)$

$\downarrow$

$\delta(q_1, \varepsilon)$

$\downarrow$

$\delta d \Rightarrow q_2 \in F$

∴ a, b, c is also accepted.

∴ $(a^* b^* c^*)$ is RL → Regular Language

∴ $\boxed{RL \rightarrow a^* b^* c^*}$

* Use of ε-transitions

1. Without reading i/p symbol, we change state. → Advantage

Disadv.
→ unnecessary increasing no. of states.

Decidable.

* Procedure for ε-NFA ⟶ NFA

Step 1: To find out ε^* (ε-closure) of all states in the given diagram.

Step 2: Calculate extended transition functn ($\hat{\delta}$) for every state & every i/p symb.

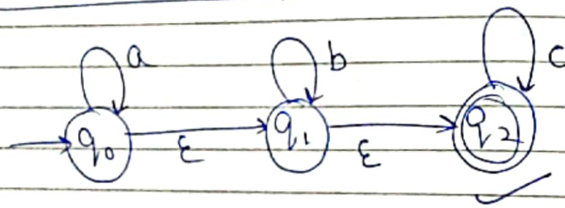
$$\hat{\delta}(q,a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q,\varepsilon),a\right)\right)$$

$$\hat{\delta}(q,\varepsilon) = \varepsilon\text{-closure}(q)$$

$\boxed{V\text{-}I}$

Step 3: Identify the final States.

* ε-closure of q_0:



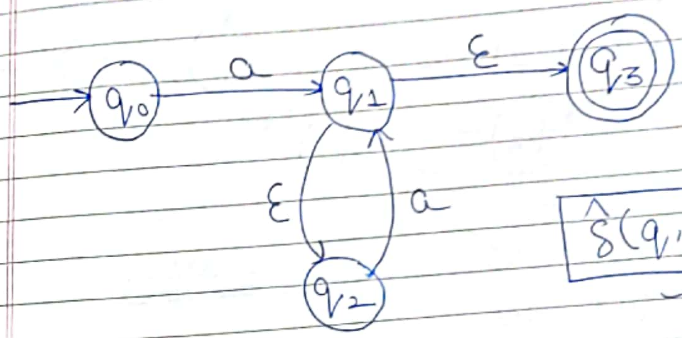
ε-closure$(q_0) = \{q_0, q_1, q_2\}$

ε-closure$(q_1) = \{q_1, q_2\}$

ε-closure$(q_2) = \{q_2\}$

$\boxed{\varepsilon\text{-closure}\left[\delta\left(\hat{\delta}(q,\varepsilon),a\right)\right]}$

Q Find out ε closure of all states for the following diagram.



$\boxed{\hat{\delta}(q,\varepsilon) = \varepsilon\text{ closure}}$

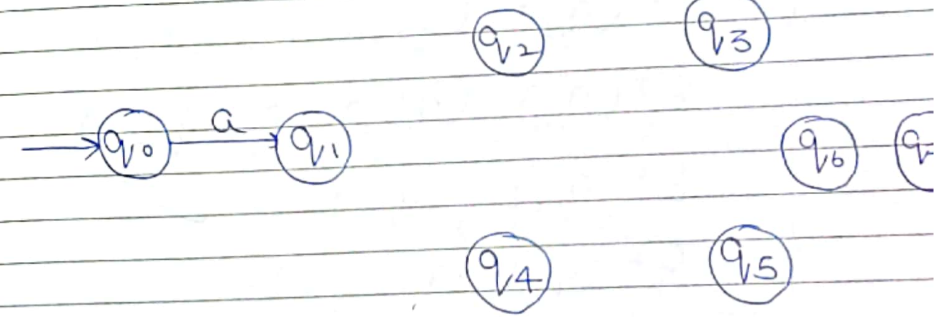
$\varepsilon^*(q_0) = \{q_0\}$

$\varepsilon^*(q_1) = \{q_1, q_2, q_3\}$

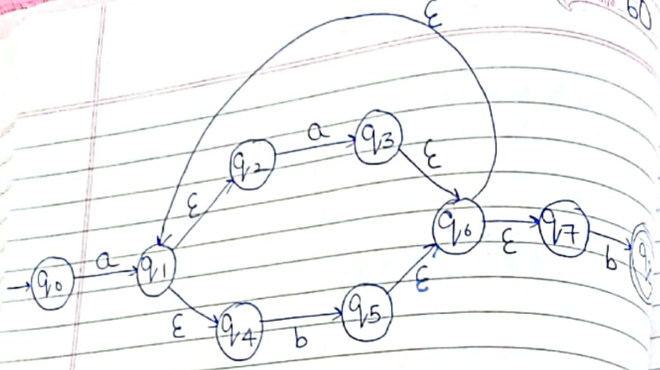
$\varepsilon^*(q_2) = \{q_2\}$

$\varepsilon^*(q_3) = \{q_3\}$

Q



PTO



ε-closures

$\varepsilon^*(q_0) = q_0$

$\varepsilon^*(q_1) = (q_1, q_2, q_4)$

$\varepsilon^*(q_2) = q_2$

$\varepsilon^*(q_4) = q_4$

$\varepsilon^*(q_3) = (q_3, q_6, q_1, q_2, q_4, q_7)$

$\varepsilon^*(q_5) = (q_5, q_6, q_7, q_1, q_2, q_4)$

$\varepsilon^*(q_6) = (q_6, q_1, q_2, q_4, q_7)$

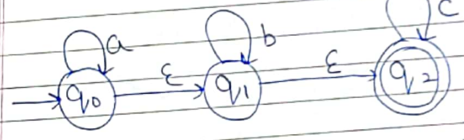
$\varepsilon^*(q_7) = (q_7)$

$\varepsilon^*(q_8) = q_8$.

q_0

Q Eliminate ε transitions from the following diagram:

ENFA $\rightarrow$ NFA.



S1: $\varepsilon^*(q_0) = \{q_0, q_1, q_2\}$

$\varepsilon^*(q_1) = \{q_1, q_2\}$

$\varepsilon^*(q_2) = \{q_2\}$

	a	b	c
q_0	q_0	ϕ	ϕ
q_1	ϕ	q_1	ϕ
q_2	ϕ	ϕ	q_2

S2: $\hat{\delta}(q_0, a) = \varepsilon^*(\delta(\hat{\delta}(q_0, \varepsilon), a))$

$= \varepsilon^*(\delta(q_0, q_1, q_2), a)$.

$\Rightarrow \varepsilon^*(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a))$

$\Rightarrow \varepsilon^*(q_0) = \{q_0, q_1, q_2\}$

$\hat{\delta}(q_0, b) = \varepsilon^*(\delta(q_0, q_1, q_2, b)$

$\Rightarrow \varepsilon^*(q_1) = \{q_1, q_2\}$

Page 62

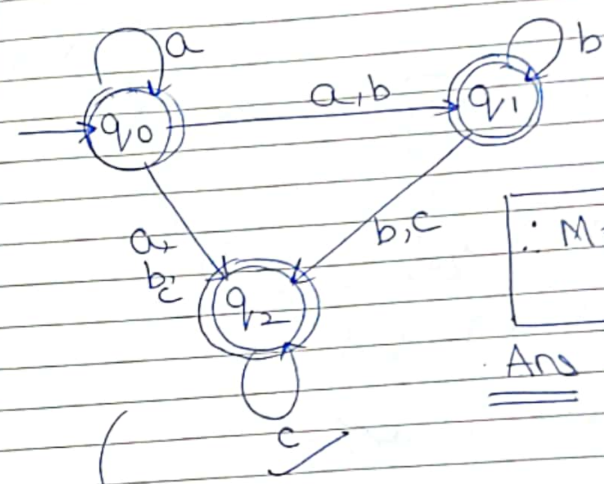
- $\hat{\delta}(q_1, a) = (q_1, q_2) \to a$
  $\Rightarrow \phi$
- $\hat{\delta}(q_0, c) = q_2$
- $\hat{\delta}(q_1, b) = (q_1, q_2) \to b$
  $\Rightarrow \{q_1, q_2\}$
- $\hat{\delta}(q_1, c) = (q_1, q_2) \to c$
  $\Rightarrow q_2$
- $\hat{\delta}(q_2, a) = (q_2) \to a$
  $= \phi$
- $\hat{\delta}(q_2, b) = (q_2) \to b$
  $\Rightarrow \phi$
- $\hat{\delta}(q_2, c) = (q_2) \to c$
  $= q_2$

Page 63

$M = \{Q, \Sigma, q_0, \delta, F'\}$
↳ ε-NFA

[Final states change]

∴ Same no. of states



$\therefore M = \{Q, \Sigma \cup (\varepsilon), q_0, \delta', F'\}$

Ans → ε-NFA

→ NFA without ε transitions

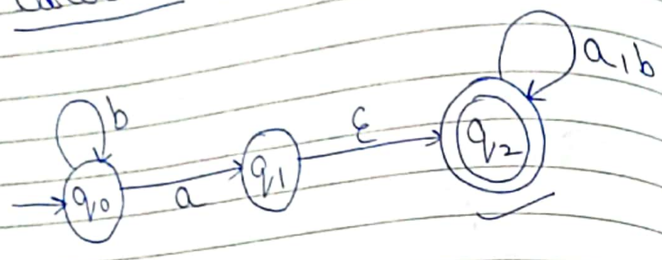
- $\varepsilon^*(q_0)$ contains F ∴ $q_0 \to F$
  $\varepsilon^*(q_1)$ contains $F(q_2)$ ∴ $q_1 \to F$

$\boxed{RE = a^* b^* c^*}$

eg: ba $b \to q_1, q_2$
$\underline{a}$ → we dont have path

∴ not accepted

Q. Convert the following ε-NFA to NFA.



$\varepsilon^*(q_0) = \{q_0\}$

$\varepsilon^*(q_1) = \{q_1, q_2\}$

$\varepsilon^*(q_2) = q_2$

$\hat{\delta}(q_0, a) = q_0, a = \{q_1, q_2\}$

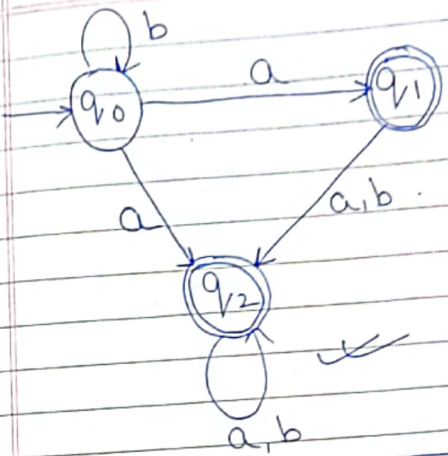
$\hat{\delta}(q_0, b) = q_0, b = \{q_0\}$

$\hat{\delta}(q_1, a) = \{q_1, q_2\}\ a = \{q_2\} = q_2$

$\hat{\delta}(q_1, b) = (q_1, q_2), b = q_2$

$\hat{\delta}(q_2, a) = q_2\ a = q_2$

$= q_2$



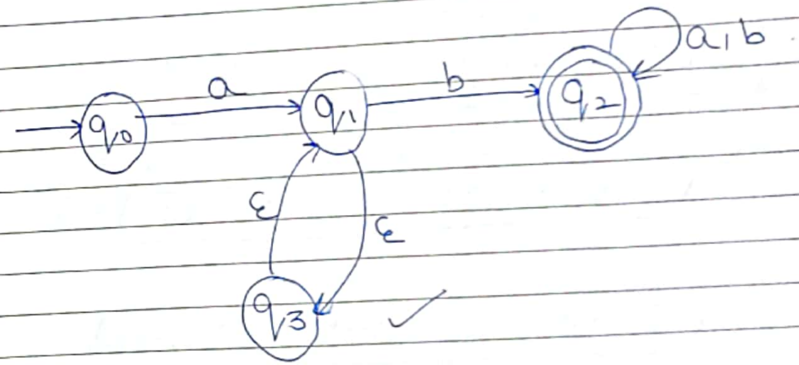
[NFA]

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

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

[ε-NFA]

Q.



$\varepsilon^*(q_0) = q_0$

$\varepsilon^*(q_1) = \{q_1, q_3\}$

$\varepsilon^*(q_2) = q_2$

$\varepsilon^*(q_3) = \{q_3, q_1\}$

$\hat{\delta}(q_0, a) = q_0, a = q_1 = \{q_1 q_3\}$

$\hat{\delta}(q_0, b) = q_0, b = \phi$

$\hat{\delta}(q_1, a) = \phi\ (q_1 q_3), a = \phi$

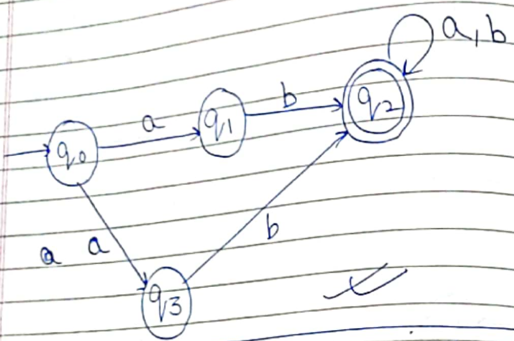
$\hat{\delta}(q_1, b) = (q_1 q_3), b = q_2 \Rightarrow q_2$

$\hat{\delta}(q_2, a) = q_2, a = q_2$

$\hat{\delta}(q_2, b) = q_2, b = q_2$

$\hat{\delta}(q_3, a) = (q_3 q_1)\, a = \phi$

$\hat{\delta}(q_3, b) = (q_3 q_1)\, b = q_2$



[Direct Conversion]

* Conversion of : ε-NFA to DFA :

By Subset Construction Method :

ε-NFA : $M = (Q, \Sigma \cup \{\varepsilon\}, q_0, \delta, F)$

$\Downarrow$

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

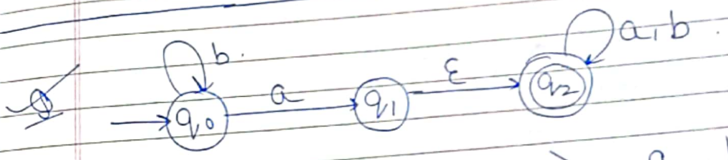
$\boxed{\text{VI}}$

$\varepsilon^*(q_0)$ [ε-closure of q_0]

*

· $\delta'(q, a) = \varepsilon\,\text{closure}(\delta(q, a))$

· q_2 / combination of $q_2 \rightarrow F$



	a	b
q_0	q_1	q_0
q_1	ϕ	ϕ
q_2	q_2	q_2

$\Downarrow$

δ'	a	b
q_0	$\{q_1, q_2\}$	q_0
$*q_1, q_2$	$\phi\ q_2$	q_2
$*q_2$	q_2	q_2

Subset constructⁿ method : DFA

$\varepsilon^*(q_0) = q_0$

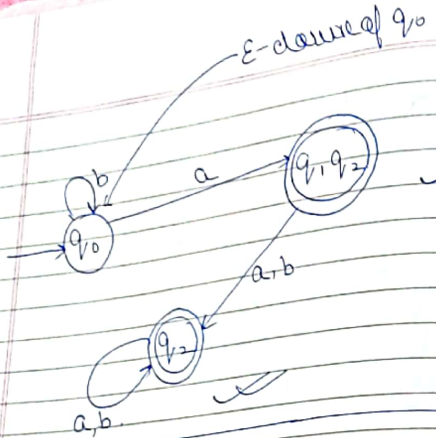
$\varepsilon^*(q_1) = q_1 q_2$

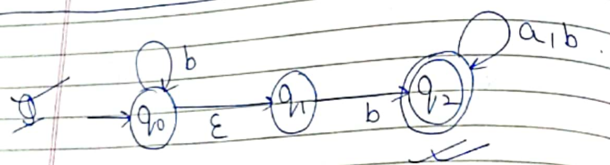
$\varepsilon^*(q_2) = q_2$

$\delta'((q_1, q_2), a) = (q_1, q_2), a = q_2$

$\delta'((q_1, q_2), b) = (q_1, q_2), b = q_2$

$\delta'(q_2, a) = q_2$

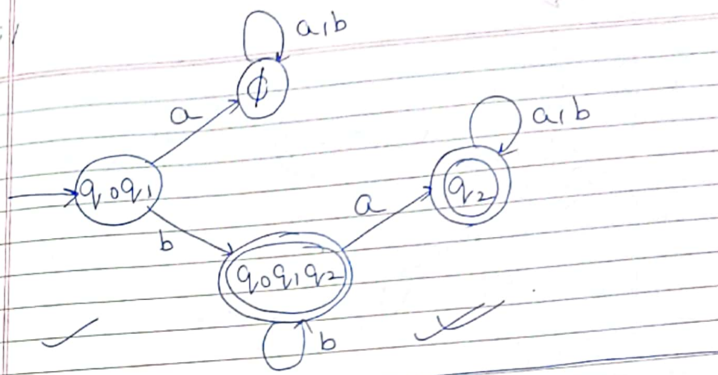


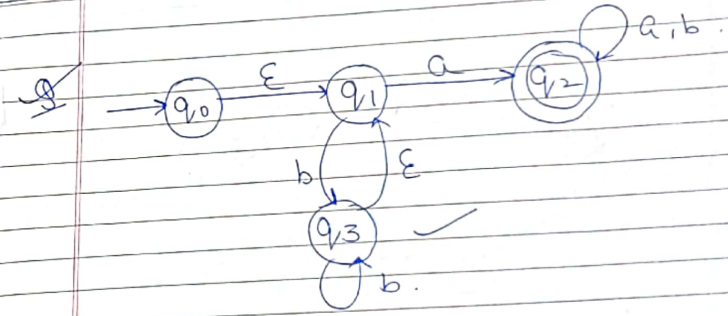


δ'	a	b
q_0q_1	ϕ	$q_0q_1q_2$
q_2	q_2	q_2
$q_0q_1q_2$	q_2	$q_0q_1q_2$

Revised

$\varepsilon^*(q_0) = q_0 q_1$
$\varepsilon^*(q_1) = q_1$
$\varepsilon^*(q_2) = q_2$

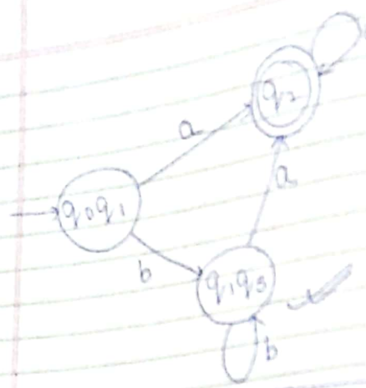


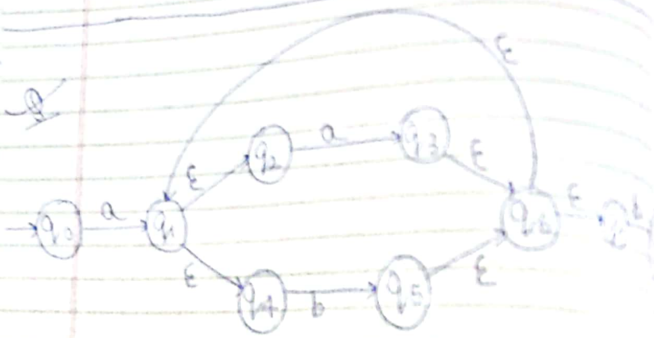


$\varepsilon^*(q_0) = q_0 q_1$

	a	b
q_0q_1	q_2	q_3q_1
q_3q_1	q_2	q_3q_1
q_2	q_2	q_2

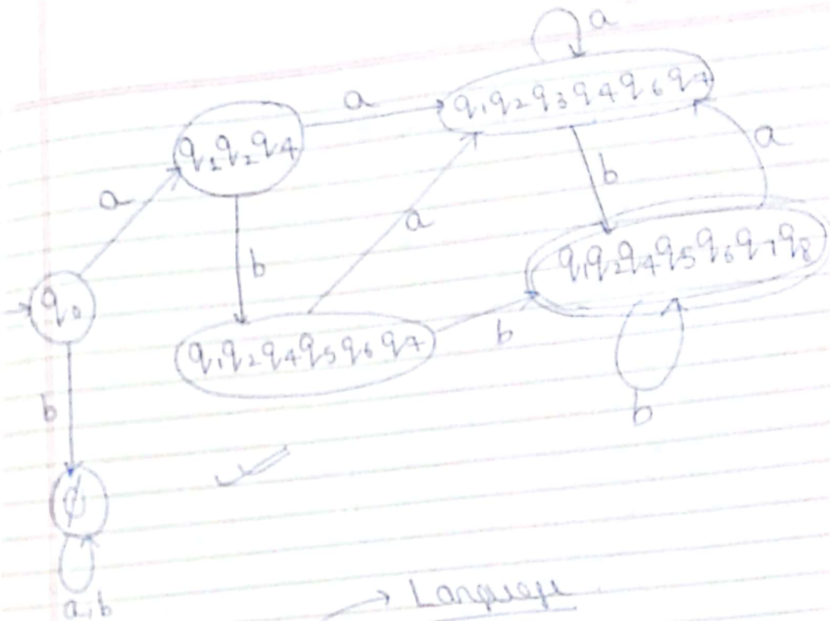
$\varepsilon^*(q_0) = q_0q_1$
$\varepsilon^*(q_1) = q_1$
$\varepsilon^*(q_2) = q_2$
$\varepsilon^*(q_3) = q_3q_1$





$\varepsilon^*(q_0) =$

δ'	a	b
q_0	$(q_1q_2q_4)$	ϕ
$q_1q_2q_4$	$(q_1q_2q_3q_4q_6q_7)$	$(q_1q_2\ldots q_7)$
$q_1q_2q_3q_4q_6q_7$	$(q_1q_2q_3q_4q_6q_7)$	$(q_1q_2\ldots q_7)$



→ Language

a ______ b

→ atleast 1 a/b

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

$\boxed{RE \Rightarrow a\cdot(a+b)^+ \cdot b}$

* Equivalence of 2 DFAS

- equivalence of 2 DFAS

Procedure

q_0 = init state of M_1

$M_1 \to q_0$, $M_2 \to P_0$

$q_0 \xrightarrow{a} (X,Y) \xleftarrow{a} P_0$

X: state of M_1
Y: State of M_2

i) $X \to F$, $Y \to NF$ } stop process. (not eq.)

ii) $X \to F$, $Y \to F$ / $X \to NF$, $Y \to NF$ } then continue

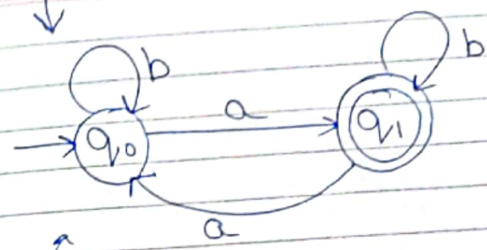
$q_0 \xrightarrow{b} (X',Y') \xleftarrow{b} P_0$

(continue, until no more new pairs are added).

Decidable Problem

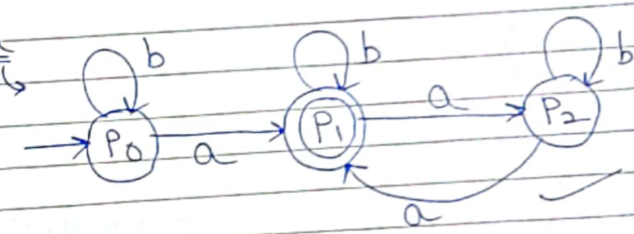
(instead of infinite comp. of strings)

eg:



M_1

M_2

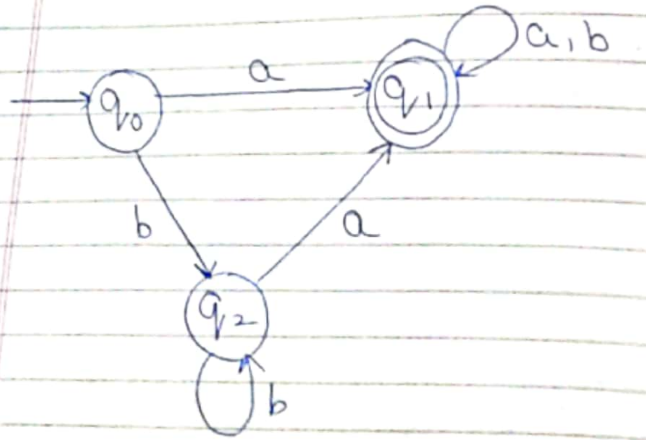


	a	b
$\to (q_0, P_0)$	(q_1^F, P_1^F)	(q_0^{NF}, P_0^{NF})
(q_1, P_1)	(q_0^{NF}, P_2^{NF})	(q_1^F, P_1^F)
(q_0, P_2)	q_1^F, P_1^F	q_0^{NF}, P_2^{NF}

All pairs are over.

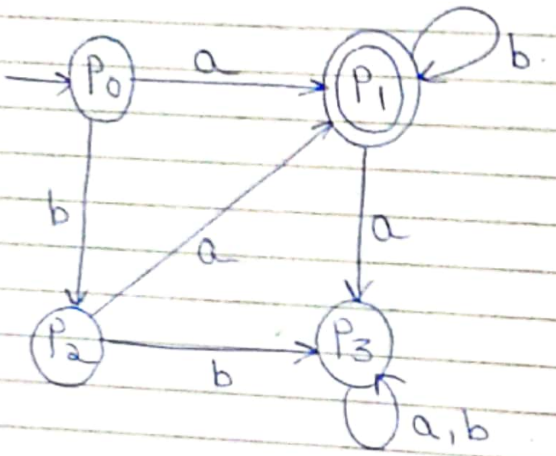
$\therefore$ $\boxed{M_1 == M_2}$. Ans

Q $\underline{\underline{M_1}}$



$\underline{\underline{M_2}}$

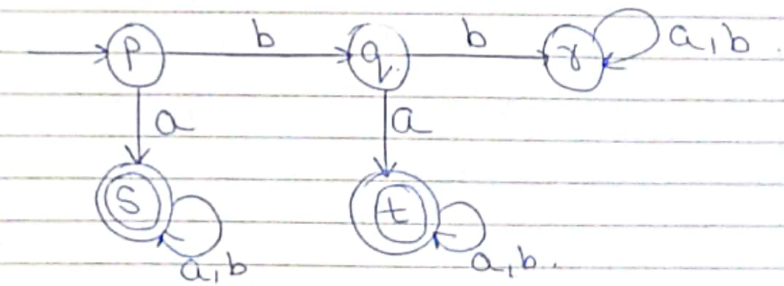
aa rejected here



	a	b
(q_0, P_0)	q_1, P_1	q_2, P_2
(q_1, P_1)	q_1, P_3	

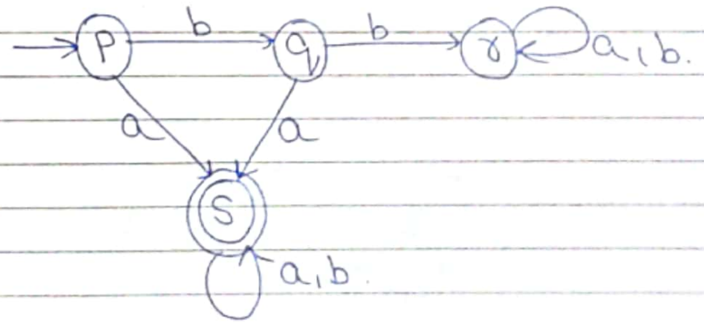
not equal

Q Consider the following DFA (D) has the $\Sigma = \{a, b\}$

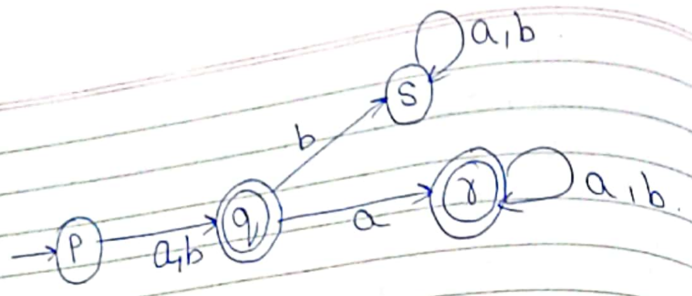


which of the following FSM is equivalent to the given DFA · D?

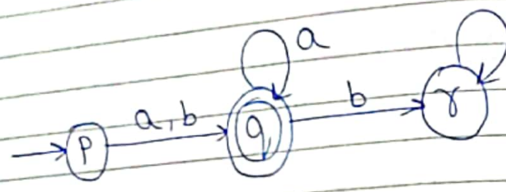
(a)



(b)

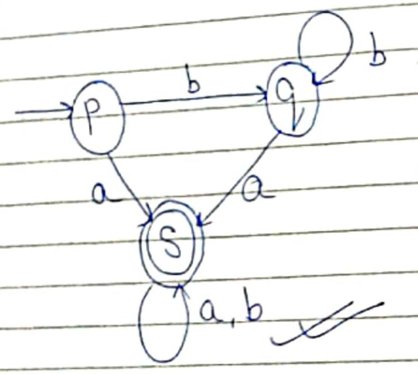


(c)



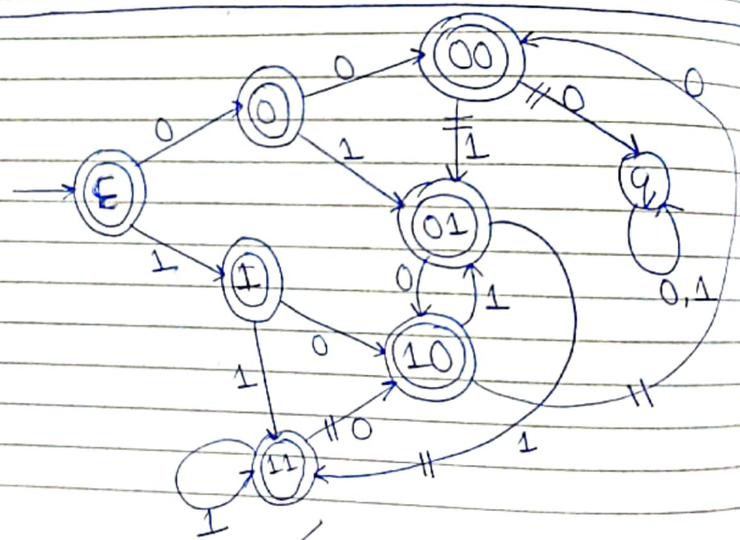
b ✓
eg: b X

(d)



bba ✓
but eg: bba X

Q



Q Consider the set of strings on Σ{0,1} in which every substring of 3 symbols has atmost 2 zeros.

eg: 001110 , 011001

- All strings of length less than 3 are also in the language.

A partially complete DFA that accepts this language is given below:

- The missing edges in the DFA are:

(a)

	00	01	10	11	q
00	1	0			
01				1	
10	0				
11			0		

(b)

	00	01	10	11	q
00		0			1
01		1			
10				0	
11		0			

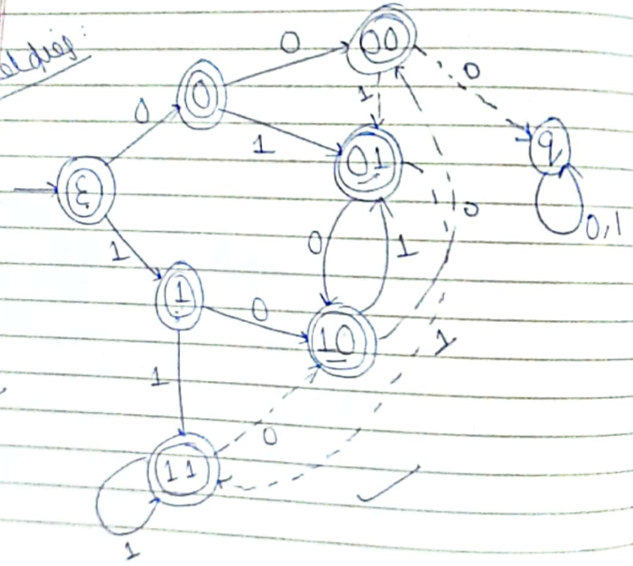
(c)

	00	01	10	11	q
00		1			0
01		1			
10			0		
11		0			

(d)

	00	01	10	11	q
00		1			0
01				1	
10	0				
11			0		

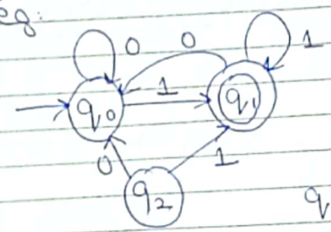
Initial diag:



* Minimization of Finite Automata

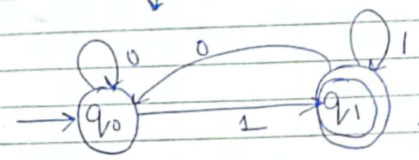
- reducing only the no. of states
  - → equivalent
  - → unreachable

eg:



$q_2 \rightarrow$ unreachable

↓



· (for every lang. → 1 minimal DFA)

VI

* 2 algorithms for finding equivalence :

(i) Π- Equivalence Method

(ii) Table Filling (Myhill-Nerode Algo.)

(i) Π- Equivalence / Partition Method

$\Pi_0 = \left(\{ Q_1^0, Q_2^0 \} \right.$ → set of non-final states; Q_1^0 → set of final states

(q_0, q_1)

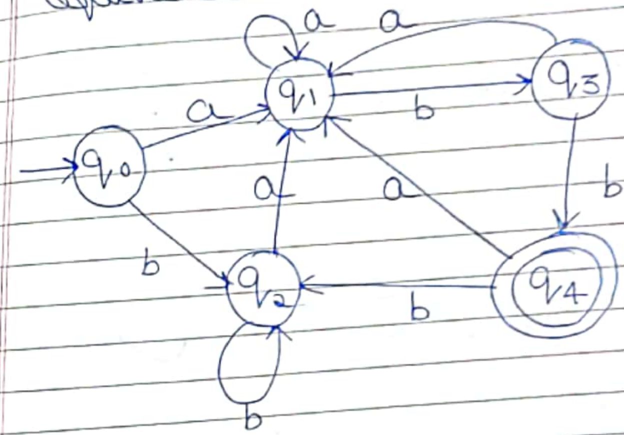
a → (p,q)

b → (X,Y).

i) p & q → same group $\therefore q_0 = q_1$

ii) X & Y → same group

Q Minimize the following FA using Π- equivalence method.



q_0, q_2 (by seeing)

Soln:

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4^*
$^*q_4$	q_1	q_2

$\Pi_0 : [\{q_4\}, \{q_0, q_1, q_2, q_3\}]$

$\Pi_1 : [\{q_4\}, [q_0 q_1 q_2] [q_3]]$

$\Pi_2 : [q_4] [q_3] [q_0 q_2] [q_1] \rightarrow 4.$

States → $q_4^*, q_3, \vec{q_{02}}, q_1$

* (q_0, q_1) — a: q_1, q_1; b: q_2, q_3 — $q_0 = q_2$

(q_0, q_2) — q_1, q_1; q_2, q_2 — $q_0 = q_2$

(q_0, q_3) — q_1, q_1; q_2, q_4 — $q_0 \neq q_3$

- $\Pi_1 = [q_4][q_0 q_1 q_2][q_3]$

$\Pi_2 =$

$(q_0 q_1)$ — q_1, q_1; q_2, q_3 — $q_0 \neq q_1$

$(q_0 q_2)$ — q_1, q_1; q_2, q_2 — $q_0 = q_2$

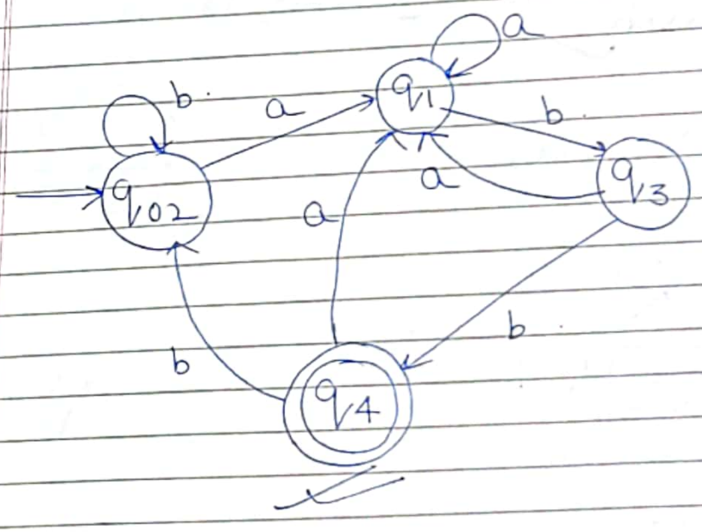
∴ • $\Pi_2 = [q_4][q_0 q_2][q_1][q_3]$

algorithm continues till ← stopping condition

$\Pi K = \Pi K+1$

$\boxed{VI}$

	a	b
$\rightarrow q_{02}$	q_1	q_{02}
q_1	q_1	q_3
q_3	q_1	q_4^*
$^* q_4$	q_1	q_{02}



Q.

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
$*q_4$	q_1	q_2

bit lengthy process.

* Table Filling Algorithm for the same problem.

- No. of states = 5

$5 \rightarrow 10$ cells
$6 \rightarrow 15$
$7 \rightarrow 21$.

q_1	X			
q_2	$q_2=q_0$	X		
q_3	X	X	X	
q_4	X	X	X	X
	q_0	q_1	q_2	q_3

relation from q_4-q_3 / q_3-q_4

q_n to q_1

till q_{n-1}

$(q_4, q_0) = (q_0, q_4)$ ... mirror img

$(q_1, q_1) \rightarrow$ not available.

Step 1 : $(Q-F) \times F$ (Mark those)

$(q_0, q_1, q_2, q_3) \times (q_4) =$

$\rightarrow (q_0, q_4), (q_1, q_4), (q_2, q_4), (q_3, q_4)$

- (q_0, q_3)

$q_1 \quad q_2$
$q_1 \quad q_4$ $\therefore$ 1 is already marked.

$\therefore (q_0 q_3)$ marked.

- (q_1, q_3)

$q_1 \quad q_3$
$q_1 \quad q_4$.

- (q_0, q_2)

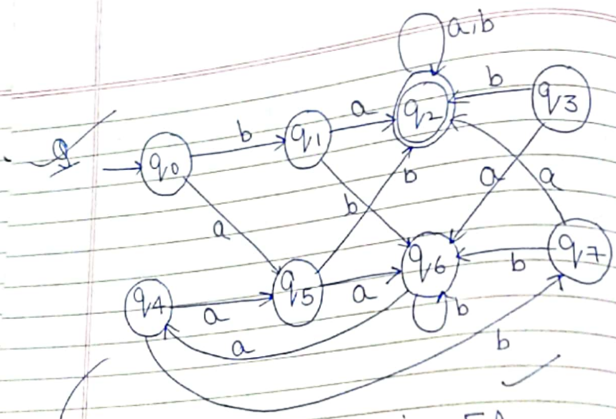
$q_1 \quad q_2$
$q_1 \quad q_2$

- (q_2, q_3)

$q_1 \quad q_2$
$q_1 \quad q_4$.

- (q_1, q_2)

$q_1 \quad q_3$
$q_1 \quad q_2$



Minimize the following FA

Tran^n Table

	a	b
$\rightarrow q_0$	q_5	q_1
q_1	q_2^*	q_6
$*q_2$	q_2^*	q_2
q_4	q_5	q_7
q_5	q_6	q_2^*
q_6	q_4	q_6
q_7	q_2^*	q_6

q_{17}
q_{04}

q_{04}
q_{17}
q_2
q_3
q_5
q_6

$q_1 = q_7$
$q_0 = q_4$.

Π - Equivalence method

Π_0 : $\{q_0, q_1, q_4, q_5, q_6, q_7\}$, $[q_2]$

Π_1 : $[q_0 q_4 q_6]$ $[q_1 q_7]$ $[q_5]$ $[q_2]$

Π_2 : $[q_0 q_4]$ $[q_6]$ $[q_1 q_7]$ $[q_5]$ $[q_2]$

$\therefore \Pi_3 = \Pi_1$

$\therefore$ 5 states in the

Π - Equivalence / Partitioning method

Sol^n

Π_0 : $[q_2]$ $[q_0 q_1 q_4 q_5 q_6 q_7]$

Π_1 : $[q_2]$ $[q_0 q_4 q_6]$ $[q_1 q_5 q_7]$

Π_2 : $[q_2]$ $[q_0 q_4]$ $[q_6]$ $[q_1 q_7]$ $[q_5]$

Π_3 : $[q_2]$ $[q_0 q_4]$ $[q_6]$ $[q_1 q_7]$ $[q_5]$

	a	b
$\rightarrow q_{04}$	q_5	q_{17}
q_{17}	q_2	q_6
$* q_2$	q_2	q_2
q_5	q_6	q_2
q_6	q_{04}	q_6

* FA with output

- Final State is not required.
  generate corresponding o/p only
- o/p generators / transducers.

[λ: output function

∴ FA ⟶ 6 tuple notation

* FA with output
  - ⟶ Meley
  - ⟶ Moore

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

λ ⟶ output function

Δ ⟶ finite set of output symbols

δ ⟶ $Q \times \Sigma \longrightarrow Q$.
(Same as DFA).
[Transition functn]

Σ ⟶ finite set of input symbols.

∴ Diff b/w Meley & Moore. $\boxed{\text{VI}}$

→ λ (O/p function).
- rest tuples are same.

* Meley m/c :

- o/p is dependent on transition.
- current state & i/p symbol.

λ: $\boxed{Q \times \Sigma \longrightarrow \Delta}$

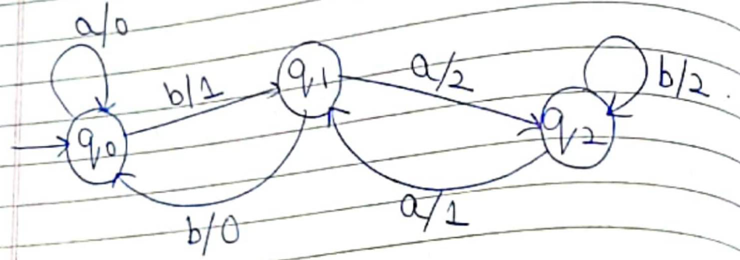
* Moore m/c :

∴ o/p symbol dependent on state
(current state.

∴ $\boxed{\lambda : Q \rightarrow \Delta}$

* Mealy m/c

$\delta : Q \times \Sigma \rightarrow Q$
$\lambda : Q \times \Sigma \rightarrow \Delta$



$M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0,$

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

$\Delta : \{0, 1, 2\}$,

$\lambda : \begin{bmatrix} \lambda(q_0, a) = 0 \\ \lambda(q_0, b) = 1 \\ \lambda(q_1, a) = 2 \\ \lambda(q_1, b) = 0 \\ \lambda(q_2, a) = 1 \\ \lambda(q_2, b) = 2 \end{bmatrix})$

VI

9

eg: I/p: abb.

010.

$\rightarrow q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_0$.

0 1 0

o/p: 010. Ans.

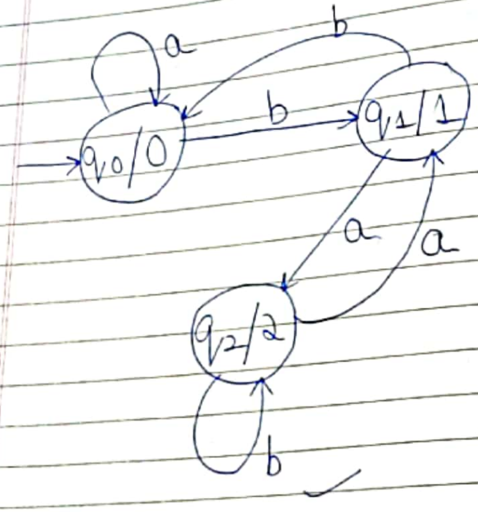
** If the i/p string length is n, the o/p string length is n in Mealy m/c.

	a	b
PS	NS, o/p	NS, o/p
q_0	q_0, 0	q_1, 1
q_1	q_2, 2	q_0, 0
q_2	q_1, 1	q_2, 2.

$\delta : Q \times \Sigma \rightarrow Q$
$\lambda : Q \rightarrow \Delta$

* Moore m/c

- Output Symbol associated with state



$\lambda(q_0) = 0$
$\lambda(q_1) = 1$
$\lambda(q_2) = 2$

eg: $w = abb$

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_0$.

0 0 1 0 .

o/p : 0010.

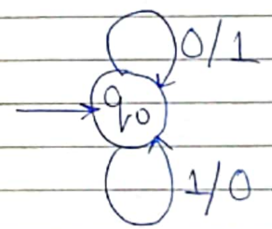
** If the i/p string length is n, the o/p string length is $n+1$ in Moore m/c

∴ without reading any symbol, o/p is generated.

	FA		o/p
	I/p = a	I/p = b	o/p
$\rightarrow q_0$	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

Q. Construct a Mealy machine to generate 1s complement of a given binary number.

Soln.



0 1 1 0
↓ ↓ ↓ ↓
1 0 0 1

· drawback of FA :
(TM resolves this)

Q To generate 2s complement of a given binary number — Mealy m/c.

0110
↓
1001
+ 1
1010

eg: 1010
↓
0101
+ 1
0110

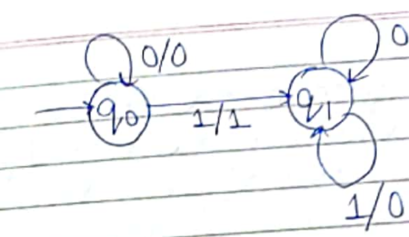
read string from R to L : [LSB to MSB]

0110

eg: 1100
↓ 2s comp.
0100

0101 1001

1101 0101



$\boxed{VI}$

Ans

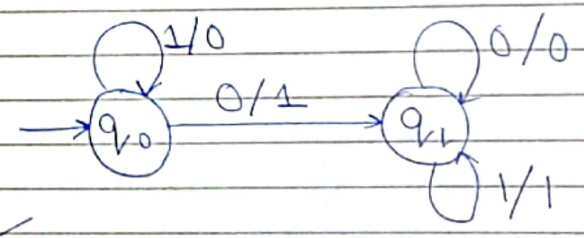
Check : 1001 = 0111

(reads from LSB to MSB) $\boxed{VI}$

Q construct a Mealy m/c to produce o/p as increment by 1 for the given binary number.

eg: 1001
+ 1
1010 →
↓
1011 → 1100 → 1101 → 1110

(until the first zero, everything is reversed)



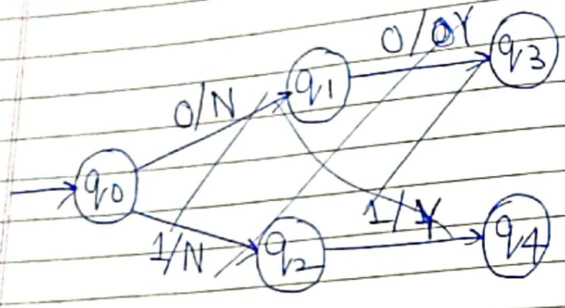
1011
↓
0111

1011
+ 1
1100

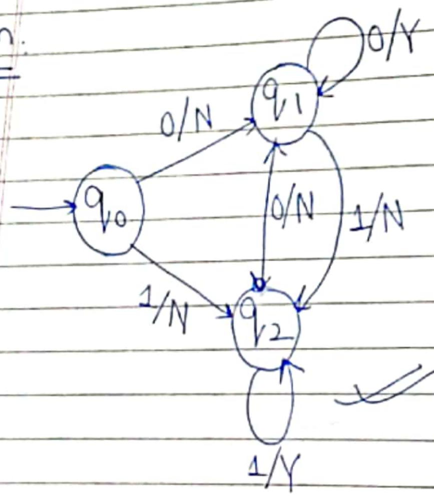
01 → N
10 → N
11 → Y

Q. Construct a Mealy m/c to produce output as Y whenever current i/p symbol and previous i/p symbol are the same otherwise o/p as N.

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



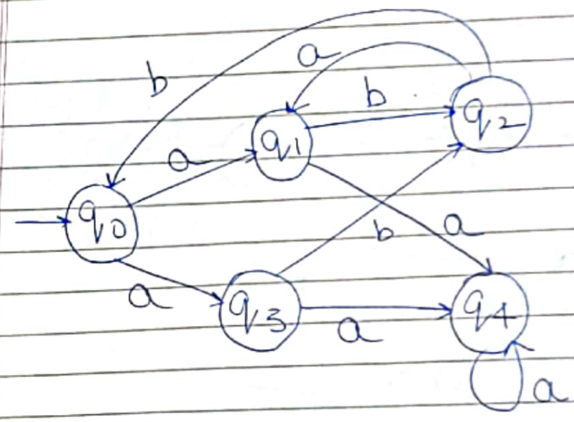
Soln:



VI

– Very similar like DFA → Moore

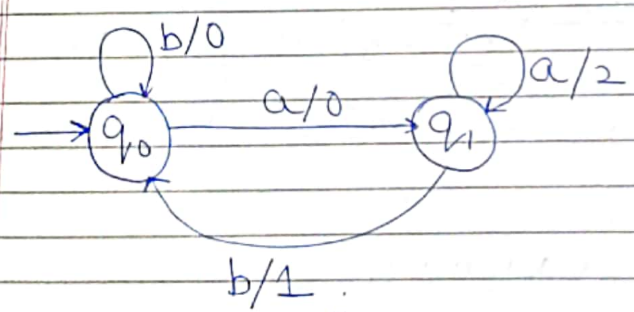
Q. Construct a mealy m/c that takes strings of a's & b's as i/p and o/p is 1 if i/p ends with ab, o/p is 2 if i/p ends with aa & otherwise o/p is 0.



(dont solve it like a string acceptor)

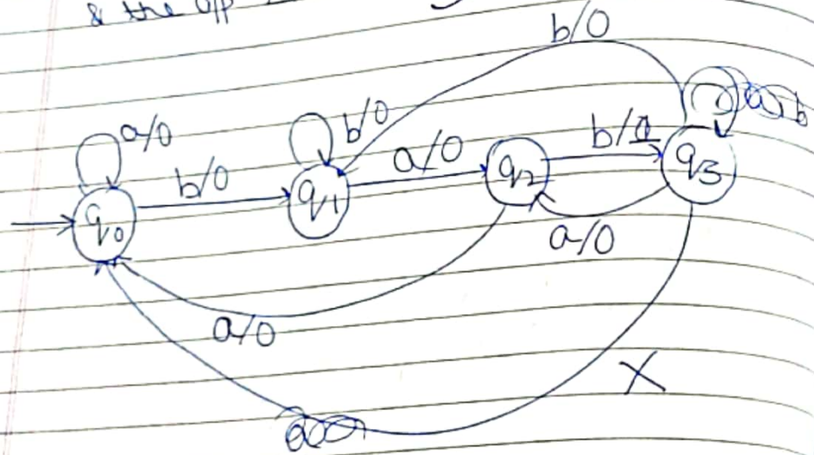
Soln:

1 → ab   ba/bb → 0   abb ⇒ 0
2 → aa   aab = 1

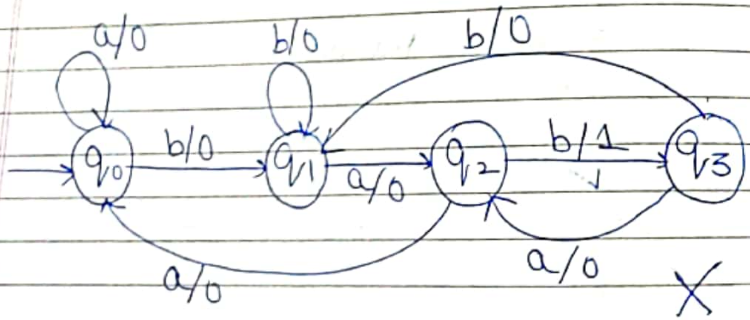


Page 100

Q Construct a Melay m/c to compute no. of substrings of the form bab that occurs in an input string over $\Sigma = \{a, b\}$ & the o/p $\Delta = \{0, 1\}$



Soln



babbab

1 1 = count = 2

Page 101

[its not end its counting no. of substrings]

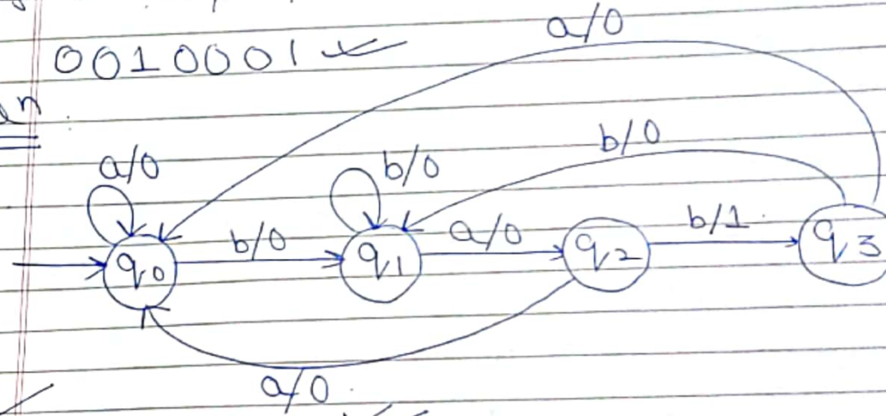
bababab

00101

eg bababab

0010001

Soln



[Draw the FA]

Q A FSM with the following state table has a single i/p X & a single o/p Z.

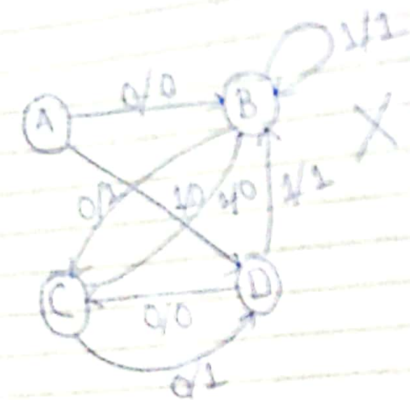
	X=1	X=0
PS	NS, Z	NS, Z
A	D, 0	B, 0
B	B, 1	C, 1
C	B, 0	D, 1
D	B, 1	C, 0

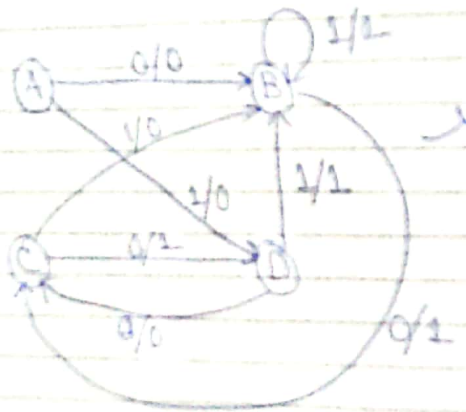
→ Melay m/c

$\delta(A, 1) = D$
$\lambda(A, 1) = 0$

If the initial state is unknown, then the shortest input sequence to reach the state C is

(a) 01
(b) 10
(c) 101
(d) 110

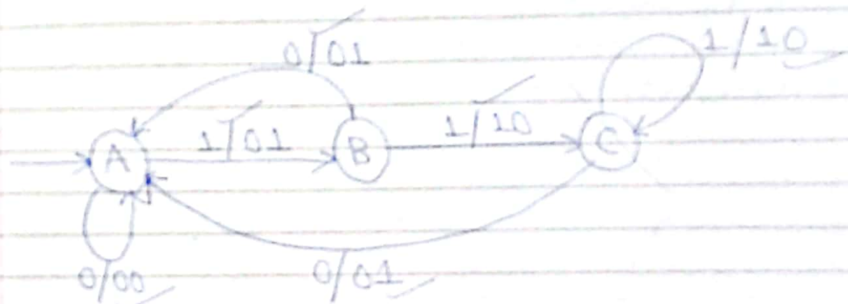




A → 00/10
B → 0/10
D → 0

Q. A FSM described by the following state diagram with A as initial state, where an edge is labelled by X/Y where
X stands for 1 bit i/p
Y stands for 2 bit o/p

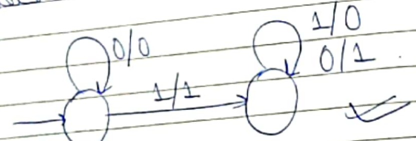
(a) o/p is the sum of the present & previous bit of the i/p
(b) o/p {0, 1} whenever the input seq contains 11
(c) o/p {0, 0} whenever the input seq contains 10
(d) None of the above



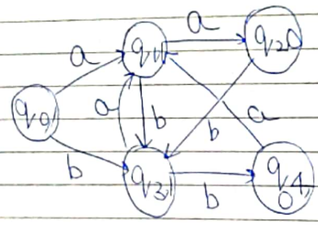
Page 104

Q. The following diagram represents a FSM which takes as input, a binary no from the LSB, which one of the following is true?

(a) It computes 1s complement of the i/p no
~~(b)~~ It computes 2s comp. of i/p no
(c) increments the i/p no
(d) decrements the i/p no



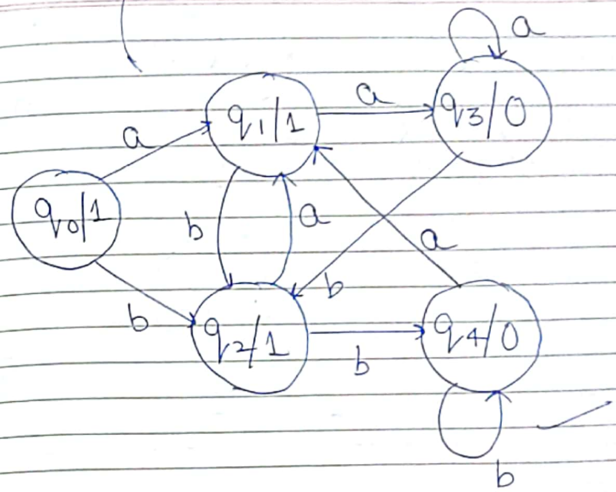
Q. Construct a Moore M/c that takes a & b as input & output it 0, if i/p ends with aa & bb, otherwise output is 1



Page 105

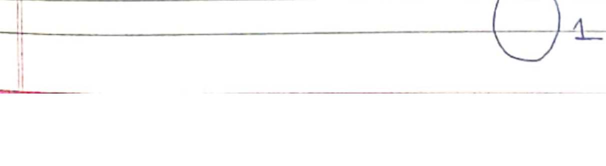
Soln:

Similar to DFA.

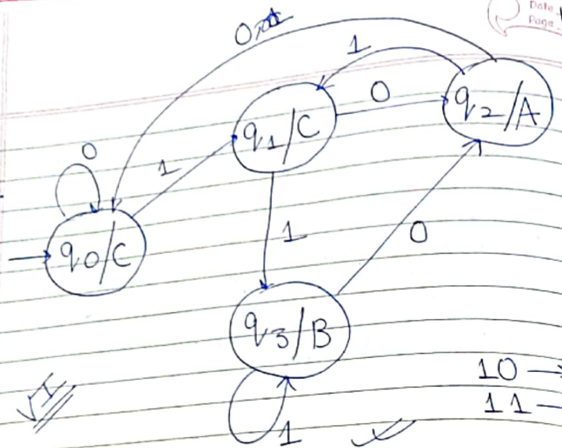


Q. Construct a Moore M/c $\Sigma = \{0,1\}$, as input; & produces A as o/p, if input ends with 10 and produces B as o/p, if i/p ends with 11; otherwise produces C as o/p.

10 → A
11 → B.



Soln.

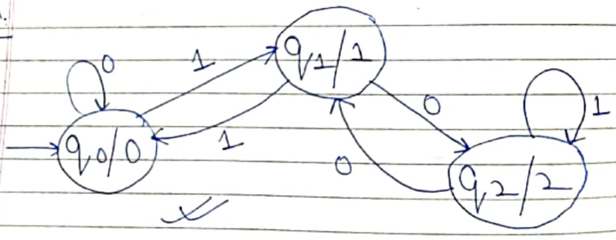


Q. Construct a Moore M/C that takes all binary no. as i/p & produces residue mod 3 (rem value) as o/p.

	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

→ DFA.

Soln.



* Conversion from Moore to Melay M/C:

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

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

~~Melay → Moore~~

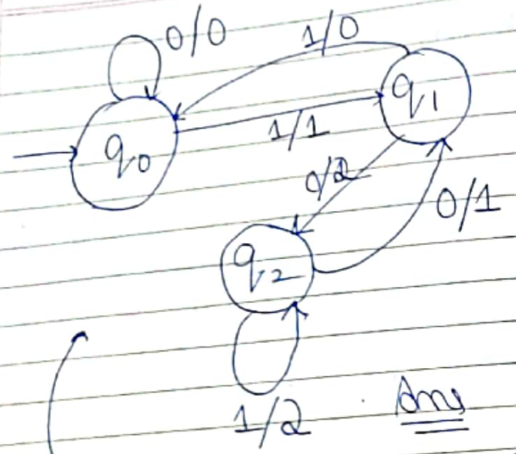
$\lambda'(q, a) = \underbrace{\lambda(\delta(q, a))}_{\text{Moore}}$ [go to the next state, call the o/p]

	0 NS/o/p	1 NS, o/p
→ q_0	$q_0, 0$	$q_1, 1$
q_1	$q_2, 2$	$q_0, 0$
q_2	$q_1, 1$	$q_2, 2$

← Melay

$\rightarrow \lambda'(q_0, 1) = \lambda(\delta(q_0, 1))$

$\lambda(q_1) = 1$.



Ans

resultant Melay M/C.

→ ignore the first bit. (∵ Moore p).

eg: 011

0010. o/p of Moore

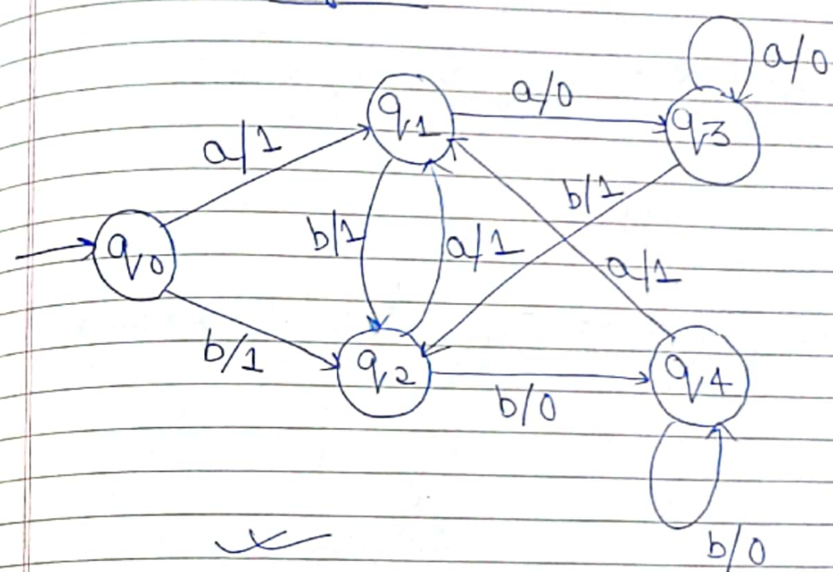
011.

010 : o/p of Meloy.

∴ ignore the first bit :

Q construct Meloy m/c

Pg 105.



* Conversion from Meloy m/c to Moore m/c

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

$\Downarrow$

$M' = (Q', \Sigma, [q_0 b], \delta', \Delta, \lambda') \longrightarrow$ Moore

same remaining

output of q_0

Since, the no. of states change, δ also changes.

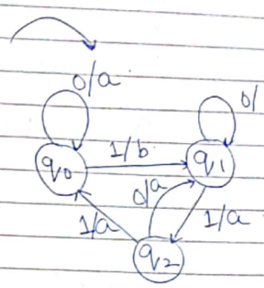
* $\delta([q_0 b], a) = [\delta(q_0, a)$
$\lambda(q_0, a)]$

Transition function of Moore

$\boxed{\lambda'([q_0 b]) = b}$

	I/p = 0		I/p = 1	
	NS	O/p	NS	O/p
q_0	q_0	a	q_1	b
q_1	q_1	b	q_2	a
q_2	q_1	a	q_0	a

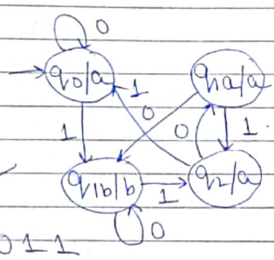
Meloy

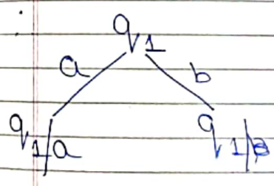


$\Downarrow$

	I/p = 0	I/p = 1	
	NS	NS	O/P
$\rightarrow q_0$	q_0	q_{1b}	a
q_{1a}	q_{1b}	q_2	a
q_{1b}	q_{1b}	q_2	b
q_2	q_{1a}	q_0	a

Moore





0 1 1

a a b a

aaba

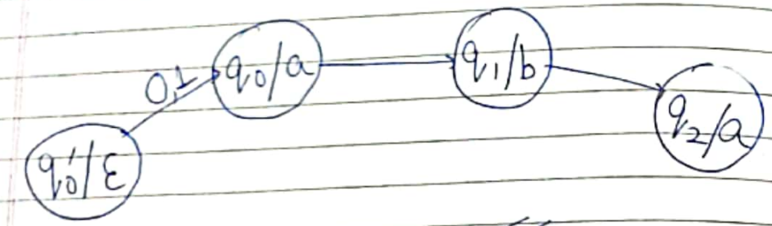
$q_0/a \xrightarrow{0} q_0/a \xrightarrow{1} q_1/b \xrightarrow{1} q_2/b$

~~a~~aba

to remove the extra letter in the beginning.

q_0'

	i/p=0	i/p=1	o/p
q_0'	q_0	$q_1 b$	ϵ

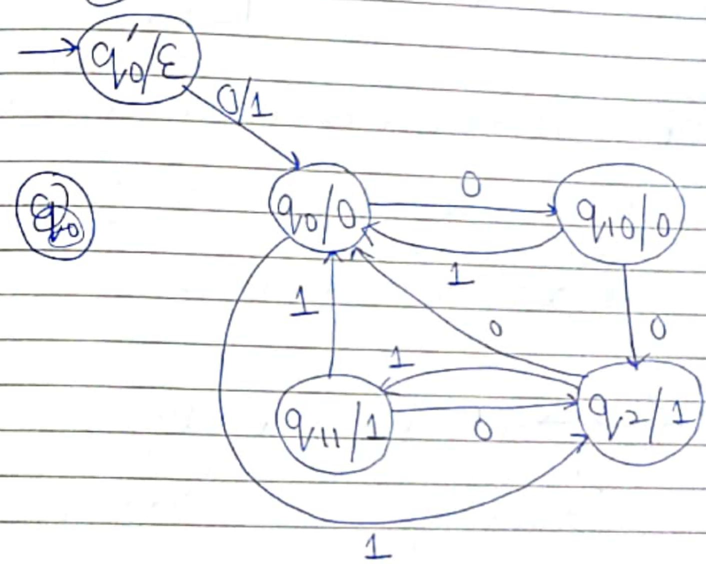


aba. Ans

	a (0)	λ	b (1)	λ
$\rightarrow q_0$	q_1	0	q_2	1
q_1	q_2	1	q_0	0
q_2	q_0	0	q_1	1

12/1/19
Saturday

	I/P = 0 NS	I/P = 1 NS	
$\rightarrow q_0$	q_{10}	q_2	
q_{10}	q_2	q_0	0
q_{11}	q_2	q_0	1
q_2	q_0	q_{11}	1



eg: 1001
↓
1.

Lecture 3

* Regular Expressions

- RE is an algebraic notation → FL
  - i/p symbols + operators (3 oper.)
  - $\Sigma = \{a, b\}$
    - → star (*) [Kleen]
    - → +
    - → ·

→ * : 0/more occurrences (unary op.)

eg: $a^* = \{\varepsilon, a, aa, \dots a^n\}$

$\Rightarrow a^0 \cup a^1 \cup a^2 \cup a^3 \dots \cup a^n$

→ + → unary → a^+ (positive closure) → 1/more occurrences → $\{a, aa, aaa \dots a^n\}$

→ binary → a+b [or(1)/union]

a/b
a∪b
a+b

$\Rightarrow \{a, b\}$

→ · : concatenation (binary).

eg: $a \cdot b \Rightarrow ab$ — a followed by b.
$b \cdot a \Rightarrow ba$.

$\boxed{a \cdot b \neq b \cdot a}$

* Use of RE (purpose)

- define / describe language rules
- define identifier rules / number rules
- not possible to define all strings.

→ just define the rules / compiler rules.

eg: create password for banking.

Application: define language rules.

eg: $l \rightarrow [A-Z]$
$d \rightarrow [0-9]$

$id \rightarrow l(l+d)^*$

[identifier pattern]

- $(a+b)^*$ → any comb. of a & b. $\{\varepsilon, a, b, ab, ba \dots\}$

1a ✗
a ✓

eg: $Num \rightarrow d^+$ | (integers).

Num $\rightarrow d^*.d^+$ (float)

.2, .3, .4 ...
1.2, 1.1 ...

**Examples of RL

Q. Write a RE for producing any no. of a s.

$RE = a^*$

Q. any no. of a s without ε

$RE = a^+$

Q. any no. of a s & b s

$RE = (a+b)^*$

Q. any no. of a s followed by any no. of b s followed by any no. of c s

$RE = a^* b^* c^*$

Q. except 0 length strings

$RE = a^+ b^+ c^+$.

every alphabet exists.

Q. Write a RE for even length strings

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

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

$RE = (aa + ab + ba + bb)^*$.

Q. ~~RE~~ odd length strings

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

Q. even no. of 0s over $\Sigma = \{0\}$.

$RE = (0\cdot 0)^*$.

$L = \{\varepsilon, 00, 0000, 000000 \ldots\}$

Q add no. of 0 s.

RE = $(00)^*.0$ or $0.(00)^*$

Q starting & ending with same symbol over $\Sigma = \{a, b\}$

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

Q starting & ending with different symbol

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

Min. symbol = ab / ba → $\boxed{VI}$

Min symbol = ϵ / a / b

Q atleast one pair of a s over $\Sigma = \{a, b\}$

R) L = aa

[continuous as]

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

Q starting with aa

$aa(a+b)^*$

Q ending with aa

$(a+b)^*aa$

Q1 Write RE for generating exactly length 2.

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

Q2 atleast 2

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

Q3 atmost 2

RE = $(a+b+\epsilon).(a+b+\epsilon)$

(1) $L = \{aa / ab / ba / bb\}$ → exactly $L[2]$

$\Rightarrow (a+b) \cdot (a+b)$

(2) $L = \{aa, ab, ba, bb \cdots\}$ → atleast $L(2)$.

$\Rightarrow (a+b) \cdot (a+b) \cdot (a+b)^*$.

(3) $L = \{(a+b+\epsilon) \cdot (a+b+\epsilon)$ → atmost $L(2)$

$\Rightarrow aa, ab, ba, bb, a, b, b, \wedge$

$\Rightarrow aa + ab + a + ba + bb + b + a + b + \epsilon$ → $L = 0$.

$\therefore L = \{0, 1, 2\}$ $L \to$ length.

Q. no. of a s are exactly 2 over $\Sigma = \{a, b\}$.

(ii) no. of as are atleast 2.

(iii) no. of as are atmost 2.

Soln: (Try)

(i) $RE = b^* . a . b^* . a . b^*$.

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

(iii) $L = \{$ $RE = b^* . (a+\epsilon) . b^* . (a+\epsilon) . b^*$.

(i) no. of as exactly 2

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

(ii) atleast 2 a. _ a _ a _

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

(iii) atmost 2 as — a/ϵ — a/ϵ —

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

$\epsilon + a + aa$ (0L/1L/2L a)

Q. Which one of the RE describe the L = {0, 1} consisting of strings that contain exactly 2 1s.

(a) $(0+1)^* . 11 . (0+1)^*$ → atleast a pair of 1s

(b) $0^* . 11 . 0^*$ → exactly 2 cont. 1s

(c) $0^* . 1 . 0^* . 1 . 0^*$

(d) $(0+1)^* . 1 . (0+1)^* . 1 . (0+1)^*$ → atleast 2 ones

(b) consecutive 2 1s

(a) atleast 1 pair of 1s.

Q. Which one of the following RE over $\Sigma = \{0, 1\}$ denote the set of all strings not containing 100 as substring? → L = {ϵ, 0, 1 ...}

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

(b) $0^* 1010^*$

(c) $0^* 1^* 01^*$

(d) $0^* (10+1)^*$

VI

(a) $(1+0)^*$ → 100

(b) 10^* → 100

(c) & (d) both contain go in rev. order.

(c) ms = 0

(d) ms = ϵ. (ϵ doesnt contain 100)

Q Which 2 of the following RE are equivalent?

(a) (i) & (ii)
(b) (ii) & (iii)
(c) (i) & (iii)
(d) (iii) & (iv)

(i) $(00)^* . (\varepsilon + 0) \longrightarrow$ odd & even
(ii) $(00)^* \longrightarrow$ even
(iii) $0^* \longrightarrow$ odd & even
(iv) $0 . (00)^* \longrightarrow$ odd

Q Consider the language $L = \{ab, aa, baa\}$ which of the following strings are in L^*.

(a) abaabaaabaa
(b) aaaabaaaa
(c) baaaaabaaaab
(d) baaaaabaa

(ans) (1) a, b, c
(2) b, c, d
(3) a, b, d $\longrightarrow$ a, b, d
(4) none of the above.

* Identity rules / Algebraic Closure for RE

- identical : both generate same set of strings. → decidable

ϕ:
- if for any E, FA can be made
∴ E = RE

$\boxed{VI}$.

* →

ϕ:

$\phi^* = \varepsilon$
$\phi + r = r + \phi = r$
$\phi . r = r . \phi = \phi$

$\boxed{r.r^* = r^+}$

ε:

$\varepsilon^* = \varepsilon$
$\varepsilon . r = r . \varepsilon = r$
$\varepsilon + r . r^* = \varepsilon + r^* . r = r^*$
$\varepsilon + r^* = r^*$
$(\varepsilon + r)^* = r^*$

Scanned with CamScanner

γ :

$\gamma + \gamma = \gamma$

$\gamma^* . \gamma^* = \gamma^*$

$(\gamma^*)^* = \gamma^*$

p, q :

VI $\leftarrow$ $\boxed{(p+q)^* = (p^* + q^*)^* = (p^* . q^*)^*}$

$(pq)^* p = p.(qp)^*$ pqpqp

p, q, γ :

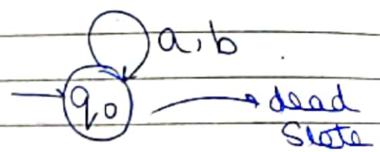
$(p+q).\gamma = p\gamma + q\gamma$.

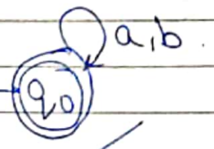
$\gamma.(p+q) = \gamma p + \gamma q$.

accepts nothing

$\phi \rightarrow \{\}$

$\varepsilon \rightarrow \{\varepsilon\}$





Q $R = \varepsilon + a^* . (abb)^* . (a^* . (abb)^*)^*$.

Try: $R = \varepsilon + a^* . (abb)^* . (a + abb)^*$.

$\Rightarrow \varepsilon + a^* . (abb)^* . [a(\varepsilon + bb)]^*$

Soln.

$\underline{\underline{\varepsilon + RR^* = R^*}}$

$R = \varepsilon + \gamma . \gamma^*$.

$\Rightarrow \gamma^*$.

$R \Rightarrow [\underset{p}{a^*} . \underset{q}{(abb)^*}]^*$. $= (p^* q^*)^*$.

$\boxed{R = (a + abb)^*}$

Q $(1 + 00^*1) + (1 + 00^*1)(0 + 10^*1)^*$

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

Soln

$\gamma =$

$\Rightarrow 1 + 00^*1 [\varepsilon + (0 + 10^*1)^* (0 + 10^*1)]$

$. \ \varepsilon + \gamma\gamma^* = \gamma^*$.

$\Rightarrow (1+00^*1)\cdot(0+10^*1)^*$

$\Rightarrow 1\cdot(\cancel{\varepsilon}+00^*)\cdot(0+10^*1)$

$\varepsilon + rr^* = r^*$ [Taking common to right side]

$\therefore$ ~~1000~~ ~~100~~*

$\Rightarrow (\varepsilon+00^*)\cdot1\cdot(0+10^*1)^*$

$\Rightarrow \underline{0^*1\cdot(0+10^*1)^*}$

Hence proved.

Q. If the regular Set (A) is represented by $A=(01+1)^*$.

& the RS (B) is represented by:

$B=((01)^*\cdot1^*)^*$.

which of the following is true?

(a) $A \subset B$
(b) $B \subset A$
(c) A & B are incomparable
(d) $A = B$

$A=(p+q)^*$.
$B=(p^*\cdot q^*)^*$.

Q. which of the following RE identities are true?

(a) $r(^*) = r^*$
(b) $(r^*s^*)^* = (r+s)^*$
(c) $(r+s)^* = r^*+s^*$
(d) $r^*s^* = r^*+s^*$.

(c) r & s cant be gen. together
(d) $r^*s^* \Rightarrow sr$ X

$r^*s^* \neq r^*+s^*$

rs $\quad$ rs X

$r^*+s^* \Rightarrow \{\varepsilon, r, s, rr, ss, rrr \ldots\}$

Q. Let S & t be a Language over $\Sigma = \{a, b\}$ represented by the RE

$(a+b^*)^*$ & $(a+b)^*$ respectively.

which of the following is true?

(a) $S \subset t$
(b) $t \subset S$
(c) $S = t$
(d) $S \cap t = \phi$

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

$L = \{ a^*, (a+b)^*, (a+bb)^* \dots \}$

$\therefore S = t$.

Q. Consider the language S : $L_1 = \phi$ and $L_2 = a$.
which one of the following represents $L_1 \cdot L_2^* \cup L_1^*$.

(a) $\{\varepsilon\}$
(b) ϕ
(c) a^*
(d) $\{\varepsilon, a\}$

$\phi \cdot a^* \cup a^*$.

$\Rightarrow \phi \cup a^* = \phi + a^* = a^*$.

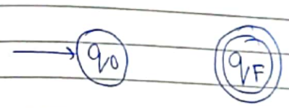
$\Rightarrow \phi \cdot a^* \cup \phi^*$.

$\Rightarrow \phi + \varepsilon = \varepsilon$.

---

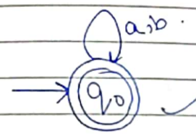
* RE to FA → DFA, NFA, ε-NFA.

(i) ϕ : RE

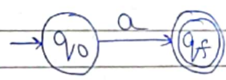


NFA

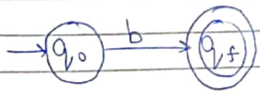
(ii) ε



(iii) a



(iv) b



Operation on RE = RE

· (a+b)

→ RE → FA

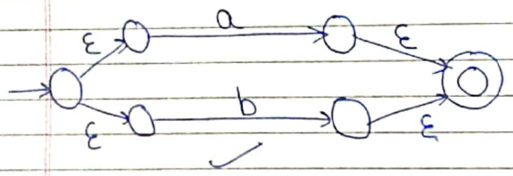
Decidable Problem

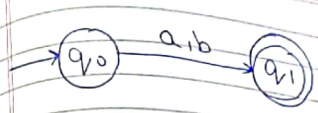
(ā)

Procedure

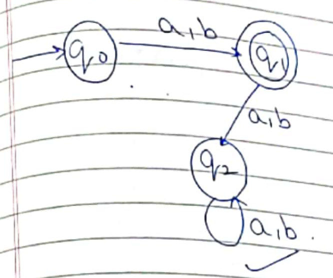
RE → ε-NFA → NFA → DFA

(v) a+b



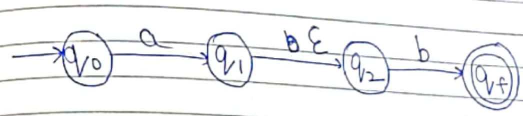


: NFA

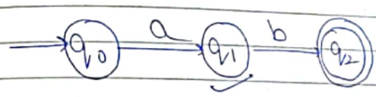


: DFA (minimal DFA)

(vi) a·b

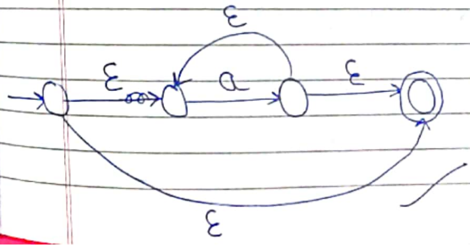


ENFA



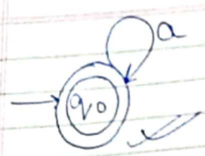
NFA

(vii) $a^* = \{ \varepsilon, a^1, a^2, a^3 \dots \}$



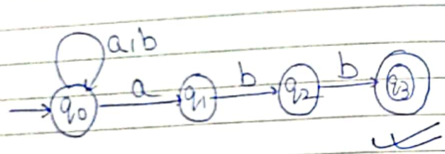
: ε-NFA

RE = equivalent FA

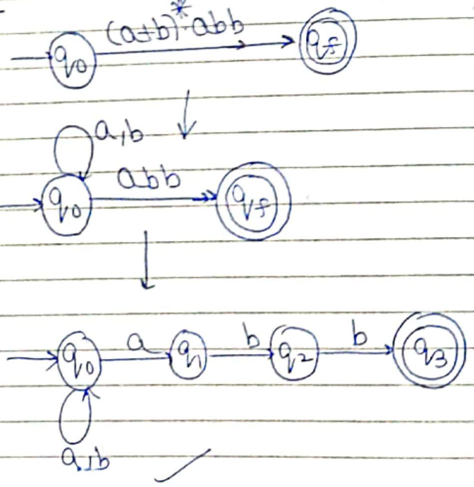


Q. Construct NFA for the following RE:

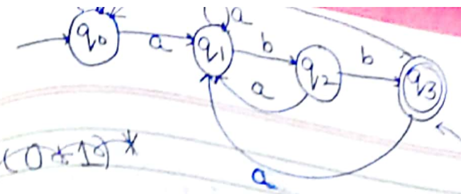
(ii) $(a+b)^* . abb$



Soln:



Steps

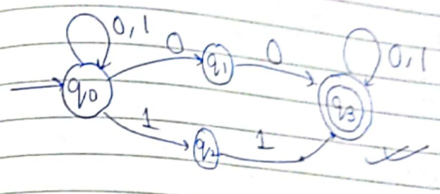


DFA

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

Q. Construct NFA for the following RE:

$(0+1)^*(00+11)(0+1)^*$



contains substring 00/11

Q. Construct a DFA for the following RE:

$(111+11111)^*$

How many no. of states are required to construct its DFA?

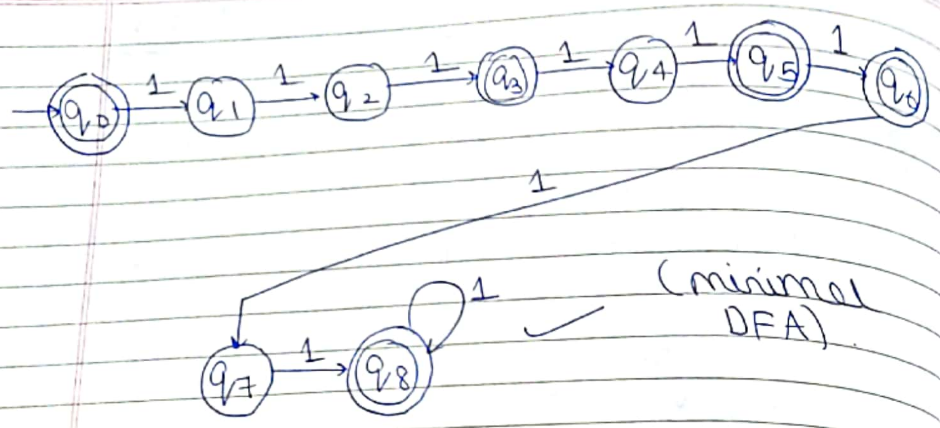
$L = \{\varepsilon, 111, 11111, 11111111$

$= \{\varepsilon, 3, 5, 6, 8, 9, 10, 11, 12, 13 \ldots$

same

$\therefore$ for 8 1s : $l = 8$ $\therefore$ 9 states

$(n+1)$



(minimal DFA)

9 states. Ans

Q The length of the shortest string not in the language over $\Sigma = \{a, b\}$ of the following RE:

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

$L = \{\varepsilon, a, b, ab, ba, aa, bb, aba, abbaaa$

- L0 ✓ → ε
- L1 ✓ → a, b
- L2 ✓ → ab, ba, aa, bb.
- L3 → aaa, aab, aba, abb, baa, bab, aba, bbb.

Length = 3 (bab)

Q Let L be the language represented by the RE.

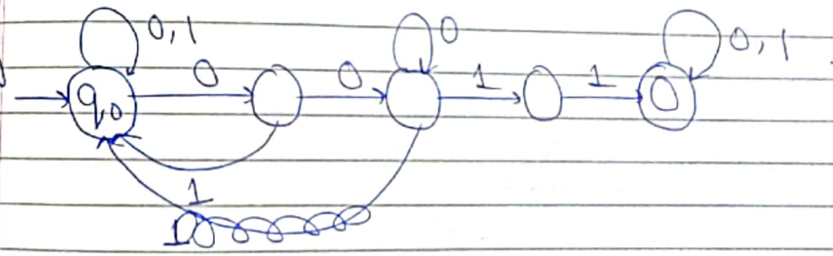
$\Sigma^* 0011 \Sigma^*$
where $\Sigma = \{0, 1\}$

What is the min. no. of states in a DFA that recognizes the $\bar{L}$? (complement of L)

Soln.

(a) 4
(b) 5
(c) 6
(d) 8

Try



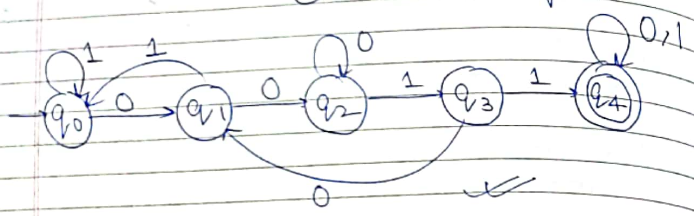
138

$\Sigma^* 0011 \Sigma^*$

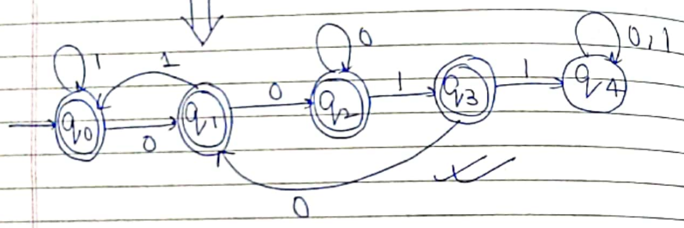
$(0+1)^* 0011 (0+1)^*$ → VI

min = 0011 (minimum string)

∴ states = n+1 = 5 (min. 5 states are req.)



⇓ I



5 states. Ans.

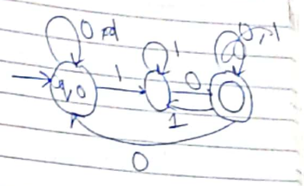
139

Q. Find the min. no. of states for constructing the DFA for the following regular expression is :

→ $(0+1)^* 10$

L = { 010 / 010 }

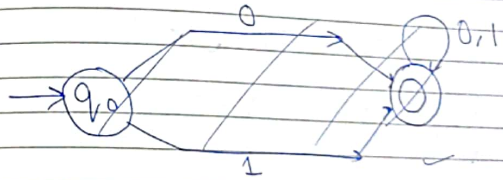
⇒ 3 states



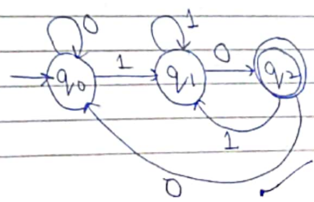
→ $\underbrace{(0+1)^*}_{\text{None}} (0+1) (0+1)^*$

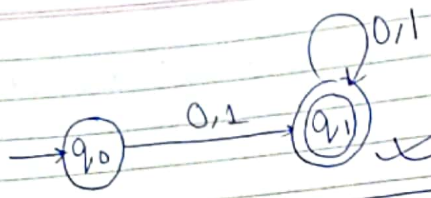
min. String = 0 / 1

∴ 2 states



L = { 0 , 1 , ... }





* FA to RE

→ Formula: Arden's Lemma / theorem → Decidable

$R = Q + RP$

Soln: $R = QP^*$ derive

$R = Q + (Q + RP)P$ (Sub. R)

$R \Rightarrow Q + QP + RPP$

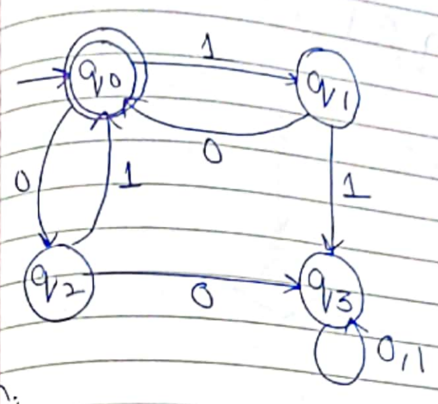
$R \Rightarrow Q + QP + (Q + RP)PP$

$\Rightarrow Q + QP + QPP + RPPP$

$\therefore R = Q(\varepsilon + P + PP + PPP \quad)$

$\therefore \boxed{R = QP^*}$

Q. Write the RE for the following FAs



$\Rightarrow (01 + 10)^*$

Soln: RE for final state : q_0

if more than 1 F : union REs

$q_0 = q_1 0 + q_2 1 + \varepsilon$ —— (i)

$q_1 = q_0 1$ —— (ii)

$q_2 = q_0 0$ —— (iii)

$q_3 = q_1 \cdot 1 + q_2 \cdot 0 + q_3 \cdot (0 + 1)$ —— (iv)

replace q_1 & q_2 form q_0 eqn.

$q_0 = q_1 0 + q_2 1 + \epsilon$

$q_0 \Rightarrow q_0 10 + q_0 01 + \epsilon$ (using q_1 & q_2)

$q_0 = q_0 (01 + 10) + \epsilon$

$R = Q + RP$

$R = QP^*$

$R = \epsilon (01 + 10)^*$

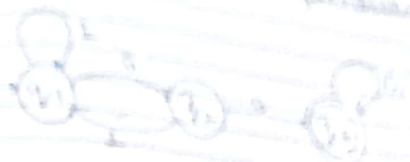
$\boxed{R = (01 + 10)^*}$ Ans

q_0 is not dependent on q_3 — no need of q_3

$R = Q + RP$

$R = QP^*$

Find the R.E. from the following FA



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

$q_3 = q_2 0 + q_3 0 + q_3 1$

$q_1 = \epsilon + q_1 1 + q_2 1$

$q_2 = q_1 0$

$\cancel{q_3 = q_1 0 + q_2 0 + q_3 1}$

$\cancel{\Rightarrow (\epsilon + q_1 1 + q_1 01)}$

$q_1 = \epsilon + q_1 1 + q_1 01$

$q_1 \Rightarrow q_1 (1 + 01) + \epsilon$

$\boxed{q_1 = (1 + 01)^*}$ $\boxed{q_2 = (1 + 01)^* 0}$

$q_3 = (q_1 00 + q_3 0 + q_3 1)$

$q_3 =$

$q_3 = q_3(0+1) + q_2 0$

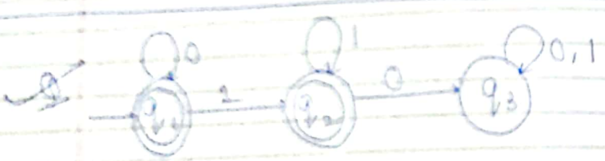
$q_3 \Rightarrow q_3(0+1) + (1+01)^* 00$

R R P Q

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

$$\boxed{R = (1+01)^* 00 (0+1)^*}$$

Q



R Q P → $R = Q + RP$

$q_1 = \varepsilon + q_1 0$

$q_2 = q_1 1 + q_2 1$

$q_3 = q_2 0 + q_3(0+1)$

$\boxed{q_1 = 0^*}$ $\quad q_2 = 0^* 1 + q_2 1$

$$\boxed{\Rightarrow q_2 = 0^* 1 1^*}$$

$q_1 \cup q_2$

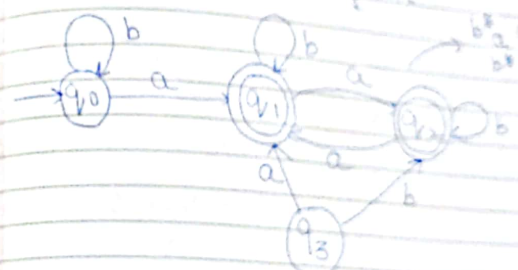
$$\boxed{R = 0^* + 0^* 1 1^*}$$

$0^*(\varepsilon + 1 1^*)$

$\boxed{R = 0^* 1^*}$ Ans

$\boxed{\varepsilon + a a^* = a^*}$

Q Consider the following FA



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

(1) The language accepted by this FA is given by the RE

(a) $b^* a b^* a b^* a b^*$ → 3a

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

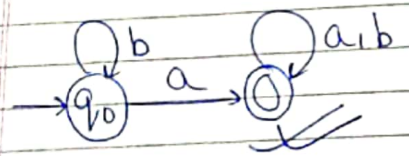
(c) $b^* a \cdot (a+b)^*$

(d) $b^* a \cdot b^* a \cdot b^*$ → 2a

(Q) The minimal state FA equivalent to the above FSM has the following no. of states:

(a) 1
(b) 2 ✓
(c) 3
(d) 4.

Try: $b^* \cdot a \cdot (a+b)^*$



Equivalence of 2 States.

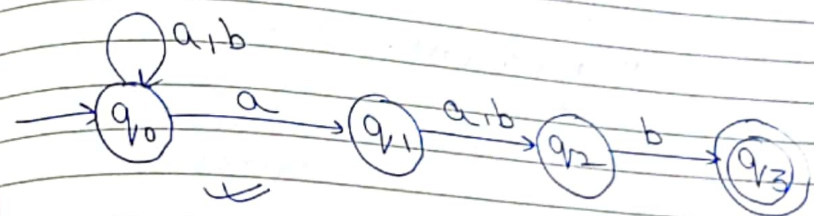
$q_1 b = q_1, \; q_2, b = q_2$
$\underbrace{\qquad}_{F}$

$q_1 a = q_2, \; q_2 a = q_1$
$\underbrace{\qquad}_{F}$.

$\therefore \underline{\underline{q_1 = q_2}}$.

Q. Which RE best describes the language accepted by the following NFA

(a) $(a+b)^* \, a \cdot (a+b) \cdot b$ ✓
(b) $(abb)^*$. $\rightarrow \varepsilon$
(c) $(a+b)^* \cdot a \cdot (a+b)^* \cdot b(a+b)$ $\rightarrow$ abax.
(d) $(a+b)^*$



II Context Free Grammar & Push Down Automata

* CFG

$G \rightarrow$ 4 tuple def. G = Grammar.

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

- $V \rightarrow$ Variables / Non-terminals
- $T \rightarrow$ Terminal Symbols.
- $P \rightarrow$ Production rules / Grammar rules of the form:

$$\alpha \longrightarrow \beta .$$

where, α = exactly 1 NT.

$\therefore \alpha \Rightarrow V$

$\beta = (V \cup T)^*$.

$\alpha \in V$
$\beta \in (V \cup T)^*$

$\alpha \longrightarrow \beta$

$\alpha \downarrow V$, $\beta \downarrow (V \cup T)^*$

$\underbrace{(V \cup T)^*}$ any no. of terminals / Non-terminals including ε.

$S \rightarrow$ Start Symbol of G.

- starts from S (derivation)

S = Non-terminal ($S \in V$)

* Based on PR, we can decide CFG / RG

CSG: $|\beta| \geq |\alpha|$.

$\alpha \longrightarrow \beta$
$\alpha \in (V \cup T)^*$
$\beta \in (V \cup T)^*$

$Aa \longrightarrow Ba$.
L=2 L=2.

CFG: $A \rightarrow \varepsilon$ → CFG.
L=1 L=0. X

$\therefore$ ε not allowed in CSG

CFG ⟶ CFL $\xrightarrow{\text{accepted by}}$ PDA

eg: $E \longrightarrow E+T \mid T$
$T \longrightarrow T*F \mid F$
$F \longrightarrow (E) \mid id$.

$V \rightarrow$ Capitals.
$T \rightarrow$ leaf nodes of parse tree.

$G = (\underbrace{\{E, T, F\}}_{V}, \underbrace{\{+, *, id, (,)\}}_{T}$

- $P = \left\{ \begin{array}{l} E \rightarrow E+T \\ E \rightarrow T \\ T \rightarrow T*F \\ T \rightarrow F \\ F \rightarrow (E) \\ F \rightarrow id \end{array} \right\}$
- $S = \{E\}$ → comes first in the PR.

eg: $S \rightarrow (a) \mid S$

$G = \left(\{S\}, \{(, a,)\}, \begin{array}{l} S \rightarrow (a), \\ S \rightarrow S \end{array} S \right)$

- Grammers help to derive strings → collection of T.

∴ (collection of terminals symbols = Strings)

↓

VI

eg: $S \rightarrow (S) \mid a$

$S \rightarrow a;$ | $S \rightarrow (S)$ [$S \rightarrow a$] $S \rightarrow (a)$ | $S \rightarrow (S)$ $S \rightarrow ((S))$ $S \rightarrow ((a))$

- $L = \{a, (a), ((a)), (((a))), ((((a)))) \ldots\}$

$S \rightarrow (S)$ [$S \rightarrow (S)$]
$S \rightarrow ((S))$ [$S \rightarrow (S)$]
$S \rightarrow (((S)))$ [$S \rightarrow [S]$]
$S \rightarrow ((((S))))$ [$S \rightarrow a$]
$S \rightarrow ((((a))))$.

→ collection of Strings : Language (CFL)

eg: $L = \{ (^n a)^n \mid n \geq 0 \}$ → From Pattern.

* generate set of Strings, & from pattern get the lang. definition.

→ language definition.

terminals → leaf nodes in Parse
Non-terminals → branch node

* from S → string
  → Process: Derivation

· [Derivation is a process to get string from S, using production rules].

Top Down — Bottom up

Top Down → discussed in TOC (root → leaf).

Q
$E \rightarrow E+T \mid T$
$T \rightarrow T*F \mid F$
$F \rightarrow (E) \mid id$

w: id + id * id from the given rules

Soln.

Top Down Derivation

Parsing (LMD)

· $E \rightarrow E+T$ [$E \rightarrow T$]
$\rightarrow T+T$ [$T \rightarrow F$]
$\rightarrow F+T$ [$F \rightarrow id$]
$\rightarrow id+T$ [$T \rightarrow T*F$]
$\rightarrow id+T*F$ [$T \rightarrow F$]
$\rightarrow id+F*F$ [$F \rightarrow id$]
$\rightarrow id+id*F$ [$F \rightarrow id$]
$\rightarrow id+id*id$. $= w$

Hence Derived.

* Derivations are classified into 2 types based on replacement of the Non-terminal.

(i) left most derivation.
(ii) right most derivation.

(i) $E \rightarrow E+T$ → (select the left most NT first for deriv.)
- start scanning from left.

(Above is done by LMD).

(ii) (If we select the right most NT first for derivation)

if in step 1 ⟶ Left
Step 2 ⟶ Right
∴ Mixing.

General Derivation

(neither Left most / Right most)

eg:

$$E \rightarrow E+T \;[T \rightarrow T*F]$$
$$\rightarrow E+T*F \;[F \rightarrow id]$$
$$\rightarrow E+T*id \;[T \rightarrow id]$$
$$\rightarrow E+id*id \;[E \rightarrow T]$$
$$\rightarrow T+id*id \;[T \rightarrow F]$$
$$\rightarrow F+id*id \;[F \rightarrow id]$$
$$\rightarrow id+id*id.$$

} RMD

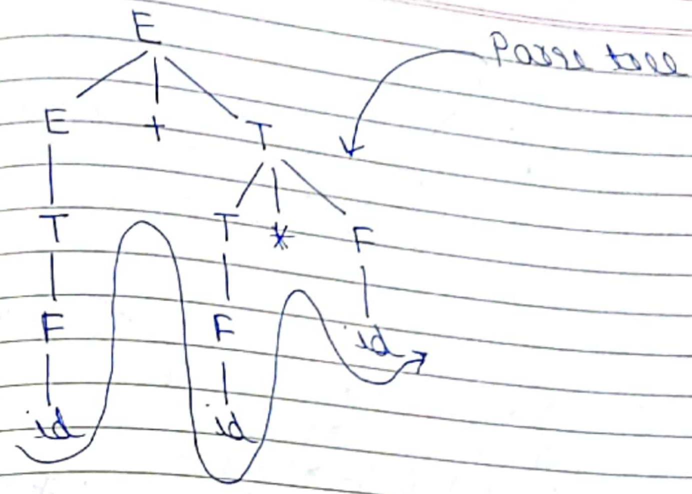
Hence, derived using RMD.

---

* The pictorial representation of derivation is known as Derivation tree / Parse tree.

Properties:

- Start (S) = root node.
- Terminal (T) / ε = leaf nodes
- Non-terminal (V) = internal nodes.



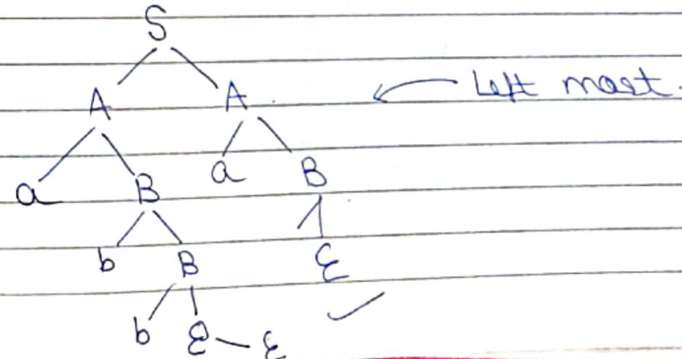
∴ id + id * id : String

Q. Consider the following G.

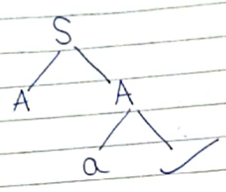
$S \longrightarrow AA$
$A \longrightarrow aB$
$B \longrightarrow bB \mid \varepsilon$

To find Left Most Derivation, Right most derivation tree for the string abba.

Soln:



abba.



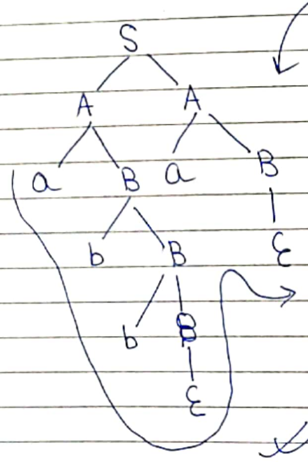
Soln:

LMD

$S \rightarrow AA$ [$A \rightarrow aB$]
$S \rightarrow aBA$ [$B \rightarrow bB$]
$S \rightarrow abBA$ [$B \rightarrow bB$]
$S \rightarrow abbBA$ [$B \rightarrow \epsilon$]
$S \rightarrow abbA$ [$A \rightarrow aB$]
$S \rightarrow abbaB$ [$B \rightarrow \epsilon$]
$S \rightarrow abba$

∴ 7 steps

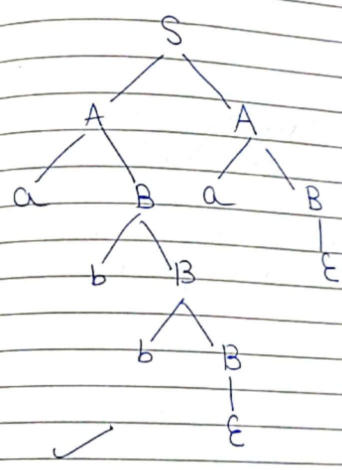
LMD tree



∴ abba

RMD

$S \rightarrow AA$ [$A \rightarrow aB$]
$S \rightarrow AaB$ [$B \rightarrow \epsilon$]
$S \rightarrow Aa$ [$A \rightarrow aB$]
$S \rightarrow aBa$ [$B \rightarrow bB$]
$S \rightarrow abBa$ [$B \rightarrow bB$]
$S \rightarrow abbBa$ [$B \rightarrow \epsilon$]
$S \rightarrow abba$.



RMD

Q. Consider the following G

$S \longrightarrow XB \mid YA$
$A \longrightarrow X \mid XS \mid YAA$
$B \longrightarrow Y \mid YS \mid YBB$

Consider the following strings:

(i) XXYYX ×
(ii) XXYYXY ×
(iii) XYXY ✓
(iv) YXXY ✓
(v) YXX
(vi) XYX

But YXXY ✓
XYXY ✓

Which of the above strings are generated by the given grammar?

(a) 1, 2, 3
(b) 2, 5, 6
(c) 3, 4
(d) 1, 3, 4

(3) XYXY $\quad S \rightarrow XB\ [B \rightarrow YS]$
$\rightarrow XYS\ [S \rightarrow XB]$
$XYXB\ [B \rightarrow Y]$
$XYXY$ ✓

YXXY $\quad S \rightarrow YA\ [A \rightarrow XS]$
$YXS\ [S \rightarrow XB]$
$YXXB\ [B \rightarrow Y]$
$YXXY$ ✓

Q. Find out the language for the following grammar

(i) $S \longrightarrow aSa \mid \varepsilon$

$L = \{ \varepsilon, aa, aaaa, a^6, a^8 \ldots \}$

$S \rightarrow aSa\ [S \rightarrow aSa]$
$\rightarrow aaSaa\ [S \rightarrow \varepsilon]$
$\rightarrow aaaa$

$\therefore L = \{ a^{2n} \mid n \geq 0 \}$ ← Lang. definition

(ii) $S \rightarrow aSa \mid a$

$L = \{ aaa, aaaaa, a \ldots \}$

$L = \{ a^{2n+1} \mid n \geq 0 \}$

(iii) $S \rightarrow aSb \mid \varepsilon$

$L = \{\varepsilon, ab, aabb, aaabbb \ldots\}$

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

(iv) $S \rightarrow aB \mid bA$
$A \rightarrow a \mid aS \mid bAA$
$B \rightarrow b \mid bS \mid aBB$ $\boxed{VI}$

Solⁿ

$L = \{ab, ba, abab,$ $S \rightarrow aB\ [B \rightarrow bS]$
$S \rightarrow abS\ [S \rightarrow aB]$
$abba, \ldots\}$ $S \rightarrow abaB\ [B \rightarrow b]$

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

abba

$S \rightarrow aB\ [B \rightarrow bS]$
$\rightarrow abS\ [S \rightarrow bA]$
$\rightarrow abbA\ [A \rightarrow a]$

(v) $S \rightarrow 0A$
$A \rightarrow 0AB \mid 0A \mid 1$
$B \rightarrow 1$

Solⁿ

$L = \{01, 001, 0011, 00011$

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

(vi) $S \rightarrow aSa \mid bSb \mid \varepsilon$ $\longrightarrow \boxed{VI}$

$L = \{\varepsilon, aa, bb, aaaa, bbbb,$
abba, baab $\ldots\}$
→ even length palindrome.

$\therefore L = \{w \cdot w^R \mid w \in (a,b)^*\}$

(vii) $S \rightarrow aCa$
$C \rightarrow aCa \mid b$

$L = \{aba, aabaa \ldots\}$

$L = \{a^n b a^n \mid n > 0\}$

ab_b
ab b_b b

(viii) $S \rightarrow aAB$
$A \rightarrow bBb$
$B \rightarrow A \mid \epsilon$

VI

Soln

$L = \{ab, abbbb, abb,$

$L = \{$ ~~abb~~

$L = \{abb, abbbb, abbbbbb \ldots\}$

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

Q Consider:

$S \rightarrow Xa \mid Ya$
$X \rightarrow Za$
$Z \rightarrow Sa \mid \epsilon$
$Y \rightarrow Wa$
$W \rightarrow Sa$

where S = Start Symbol
a = Terminal.
$V = \{S, X, Z, Y, W\}$.

Soln $L = \{$

Construct a DFA to recognise the same language.
What is the min. no. of states required for the DFA?

(a) 2
(b) 3
(c) 4
(d) 5.

Soln

$L = \{$ aa, aaaaa,
aaaaa,
aa,
aaaaa,

$L = \{aa, aaaaa$

$S \rightarrow Xa\ [X \rightarrow Za]$
$\rightarrow Zaa\ [Z \rightarrow Sa]$
$Saaa\ [S \rightarrow Xa]$
$Xaaaa\ [X \rightarrow Za]$
$Zaaaaa\ [Z \rightarrow \epsilon]$

$S \rightarrow Ya\ [Y \rightarrow Wa]$
$Waa\ [W \rightarrow Sa]$
$Saaa\ [S \rightarrow Xa]$
$Xaaaa\ [X \rightarrow Za]$
$Zaaaaa\ [Z \rightarrow \epsilon]$
aaaaa.

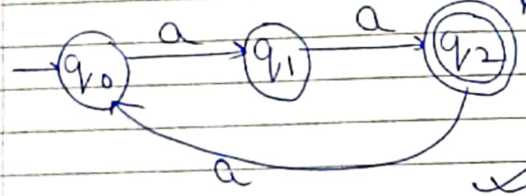
$L = \{aa, aaaaa, aaaaaaaa, \dots\}$

$a^2, a^5, a^8, a^{11} \dots$ $\xrightarrow{3}$ $\xrightarrow{3}$ $\xrightarrow{3}$ $\boxed{VI}$

$2 \% 3 = 2$
$5 \% 3 = 2$
$8 \% 3 = 2$
$11 \% 3 = 2$

Min=3 States

DFA



Q Construct the following 2 Grammars over $\Sigma = \{X, Y, Z\}$.

G_1:

$S \rightarrow X | Z | XS | ZS | YB$
$B \rightarrow Y | Z | YB | ZB$

G_2:

$S \rightarrow Y | Z | YS | ZS | XB$
$B \rightarrow Y | YS$

Which one of the following choices describes the property satisfied by the strings of these languages?

(a) G_1: No Y appears before any X.
G_2: Every X is followed by atleast 1 Y.

(b) G_1: No Y appears before any X.
G_2: No X appears before any Y.

(c) G_1: No Y appears after any X.
G_2: Every X is followed by atleast 1 Y.

(d) G_1: No Y appears after any X.
G_2: Every Y is followed by atleast 1 X.

G_1: $S \rightarrow XS$ $[S \rightarrow YB]$
$\rightarrow XYB$ $[B \rightarrow Y]$
$\rightarrow XYY$

G_2: $L = \{XY,$ ~~XY~~ $XYYZXY$

Q. Which of the following L is generated by foll: Grammar:

$$S \rightarrow aS \mid bS \mid \varepsilon$$

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

(b) $\{w \in (a,b)^* \mid$ w has equal no. of as & bs$\}$

(c) $\{a^n \mid n \geq 0\} \cup \{b^n \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$

✓(d) $\{a,b\}^*$

Soln: $L = \{a, b, ab, baba \ldots\}$

$(a+b)^*$.

(C) ba X.

(b) a, b, aba all available.

Lecture 3 [TOC]

Q A CFG-G is given with the following productions, where S : Start Symbol
A : Non-terminals
a/b : terminals.

$S \longrightarrow aS \mid A$ a^* +

$A \longrightarrow aAb \mid bAa \mid \varepsilon$. $L = \{\varepsilon, abab, ab, ba, baba$.

(i) Which of the following string is generated by the above grammar?

(a) aabbaba.
(b) aabaaba
(c) abababb
~~(d)~~ aabbaab.

Soln:

(ii) For the correct answer in above q, how many steps are required to derive the string and how many parse trees are there?

~~(a)~~ 6 and 1
(b) 6 and 2
(c) 7 and 2
(d) 4 and 2.

Soln.

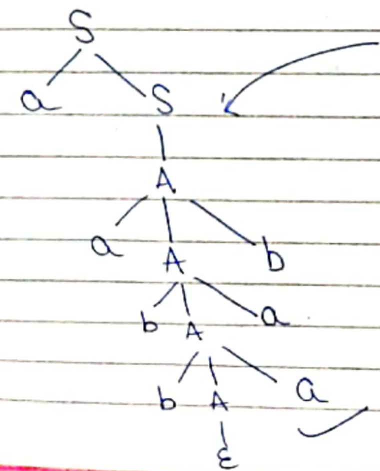
aab a

$S \rightarrow aS\ [S \rightarrow A]$
$\rightarrow aA\ [A \rightarrow aAB]$
$\rightarrow aaAB\ [A \rightarrow bAa]$
$\rightarrow aabAa\ [A \rightarrow bAa]$
$aabbAa\ [A \rightarrow aAb$

(d) aabbaab ✓

$S \rightarrow aS\ [S \rightarrow A]$.
$\rightarrow aA\ [A \rightarrow aAb]$.
$aaAb\ [A \rightarrow bAa]$
$aabAab[A \rightarrow bAa]$.
$aabbAaab[A \rightarrow \varepsilon]$
aabbaab.

6 steps.



Parse tree.

Q. Consider the following grammar

$S \rightarrow aSa \mid bSb \mid a \mid b$

The language generated by above grammar is:

(a) all palindromes
(b) all odd length palindromes
(c) strings that begins & ends with the same symbol.
(d) all even length palindromes

(c) $\rightarrow \varepsilon$, aa, bb, X.

* The CFG can be classified into (based on the no. of derivations)
  - → Ambiguous Grammar
  - → Un Ambiguous Gramm.

Ambiguous Gramm

- More than 1 parse tree.
- if atleast 1 string, more than 1 RM/LM/G parse tree.

(if more than 1 LMD/RMD,
G → ambiguous).

Language → n no. of strings

- verify all strings → not Possible

$\boxed{\text{if more than 1 PT : ambiguous}}$

∴ undecidable Problem

(finding whether Grammar is ambiguous/not?

eg. $E \to E+E \mid E*E \mid (E) \mid id$

· id + id * id

$$\left.\begin{array}{ll} \cdot\ E \to E+E & [E \to id] \\ id + E & [E \to E*E] \\ id + E*E & [E \to id] \\ id + id * E & [E \to id] \\ id + id * id & \end{array}\right\} \text{1st LMD}$$

$$\left.\begin{array}{ll} \cdot\ E \to E*E & [E \to E+E] \\ E+E*E & [E \to id] \\ id + id * id & \\ id + id * id & \end{array}\right\} \text{2nd LMD}$$

∴ Ambiguous

Q. Check whether are ambiguous/not?

~~$S \to SS \mid a \mid b \mid \varepsilon$~~

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

(a) $E \to E+T \mid T$
$T \to T*F \mid F$
$F \to (E) \mid id$

Try: id + (id * id)

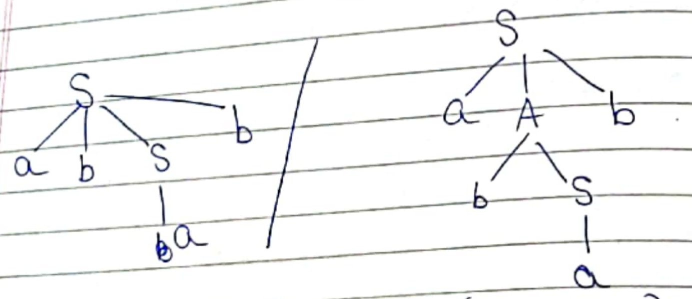
⇒ unambiguous

(b) $S \to SS \mid a \mid b \mid \varepsilon$ $\boxed{\text{VI}}$

$S \to \varepsilon$ | $S \to SS$ [S
εS
ε

∴ ambiguous

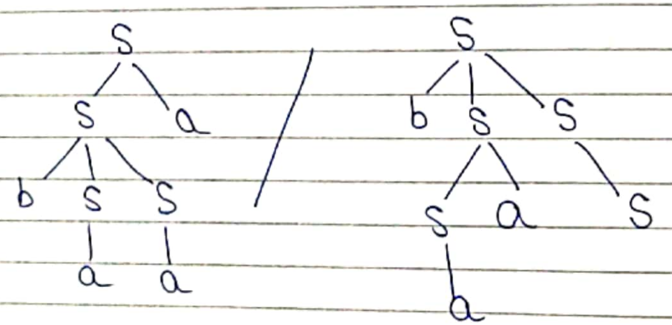
(c) $S \rightarrow a \mid aAb \mid abSb$
$A \rightarrow aAAb \mid bS$.



for deriving abab. (ambig.)

(d) $S \rightarrow a \mid Sa \mid bSS \mid SSb \mid SbS$.

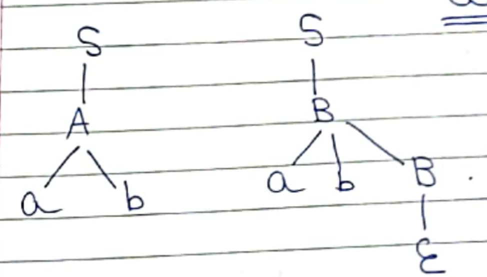
baaa



ambiguous

(e) $S \rightarrow A \mid B$
$A \rightarrow aAb \mid ab$
$B \rightarrow abB \mid \epsilon$.

ab



amb.

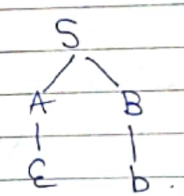
(f) $S \rightarrow (S) \mid \epsilon$

unambig.

Simplification of CFG / Minimization of CFG:

- reducing the Grammar size.
- eliminate ϵ prod^{n}s. (Why?)

$S \rightarrow AB$
$A \rightarrow \epsilon$
$B \rightarrow b$.



$\therefore S \rightarrow AB \; [A \rightarrow \epsilon]$
$B \; [B \rightarrow b]$
b

Steps = 3

$S \to AB \mid B$
$B \to b$

$S \to B \to b$ ∴ 2 steps

∴ Steps ↓

* (Due to ε production, no. of derivn steps increase)

Step 1: Elimination of ε-productions (null productions).
[$A \to \varepsilon$]

$S \to A$
$A \to B$
$B \to \varepsilon$
⇒ $S \to \varepsilon$

(1 NT in the RHS)

if exactly 1 NT → RHS
→ unit prodn (indirect reference)

Step 2: Elimination of unit prodn.
[$A \to B$, $S \to X$]

$S \to AB \mid B$
$B \to b$

A: useless symb.

⇒ symbols that dont contr. in gener. strings.

(i) useless/not (generates/not)
(ii) reachable from start/not.

useless (2 cond.)

eg: $S \to AB$
$A \to aA \mid b$
$B \to b$
$C \to a$

all prodn generate strings.

but, C not accessible from S.

Step 3: Elimination of useless productns.

1. ε productions
2. unit productions
3. useless productns → non-generating, → non-reachable

→ Simplification.

VI

Q. $S \rightarrow ABC$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow bB \mid \varepsilon$
$C \rightarrow c$

Eliminate the ε productn from this

Soln

$[A \rightarrow \varepsilon, B \rightarrow \varepsilon]$

$S \rightarrow ABC \mid BC \mid AC \mid C$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$
$C \rightarrow c$

After ε-elimination.

Q. Eliminate the unit production from the following grammar.

$S \rightarrow ABC$
$A \rightarrow aB \mid B$
$B \rightarrow aA \mid C$
$C \rightarrow b$.

Soln.

$[A \rightarrow B, B \rightarrow C]$

$S \rightarrow ABC$
$A \rightarrow aB \mid aA \mid b$
$B \rightarrow aA \mid b$
$C \rightarrow b$.

After elim. of unit prodn.

Q. Eliminate the useless productions from the following grammar.

$S \rightarrow ABC$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid bD \mid b$
$C \rightarrow cD \mid c$
$D \rightarrow dD$
$E \rightarrow e$

Soln:

S, A, B, C

generates string

$S \rightarrow ABC$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$.
$C \rightarrow c$.
$E \rightarrow e$

$\xrightarrow[\text{reachab.}]{}$

$S \rightarrow ABC$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$
$C \rightarrow c$.

C: unreachable.

Q1 $S \to A11B \mid 11A$ [useless symbols]
$A \to 0$
$B \to BB$

Q2 $S \to AaB \mid aaB$ [ε-prodn, unit & useless]
$A \to D$
$B \to bbA \mid \varepsilon$
$D \to E$
$E \to F$
$F \to aS$

Q3 $S \to aAa$
$A \to Sb \mid bCC \mid DaA$
$C \to abb \mid DD$
$D \to aDA$
$E \to aC$

Q4 $S \to AB$
$A \to a$
$B \to C \mid b$
$C \to D$
$D \to E \mid bC$
$E \to d \mid Ab$

Q5 $E \to E+T \mid T$
$T \to (E) \mid id$

Q6. $S \to 0A \mid 1B \mid C$
$A \to 0S \mid 00$
$B \to 1 \mid A$
$C \to 01$

Q7. $A \to 0B1 \mid 1B1$ → no further elimination.
$B \to 0B \mid 1B \mid 0 \mid 1$

Solution

1. $S \to 11A$ (elim of useless prodn).
$A \to 0$

2. $S \to AaB \mid aaB \mid Aa \mid aa$ $B \to \varepsilon$
$A \to \cancel{D}\ aS$
$B \to bbA$
$D \to \cancel{E}\ aS$
$E \to \cancel{F}\ aS$
$F \to aS$

$A \to D,\ D \to E,\ E \to F,\ F \to aS$

$A \rightarrow aS'$
$B \rightarrow bbA$.

Q3. $S \rightarrow aAa$
$A \rightarrow Sb \mid bCC$
$C \rightarrow abb$

Q4. $B \rightarrow C \rightarrow D \rightarrow E \rightarrow d$.

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b \mid bc \mid Ab \mid d$
$C \rightarrow bc \mid d \mid Ab$.

Q5. $E \rightarrow E+T \mid (E) \mid id$.
$T \rightarrow (E) \mid id$.

Q6. $S \rightarrow 0A \mid 1B \mid 01$
$A \rightarrow 0S \mid 00$.
$B \rightarrow 1 \mid 0S \mid 00$
$C \rightarrow 01$ X

. Ans

VI.

Lecture 4 [Toc]

* Normal Forms

If string length = n, how many steps req. to derive?

- Gram: specific format (to ans such qs)
  - → CNF
  - → GNF

CNF : Chomsky Normal Form
GNF : Greibach Normal Form

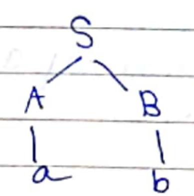
* CNF

RHS → exactly 2 NT / 1 T.
LHS → 1 NT.

$\boxed{[1\,NT \longrightarrow NT\,NT \mid T]}$ → VI.

eg. $S \rightarrow AB$ → CNF
$A \rightarrow a$
$B \rightarrow b$

ab

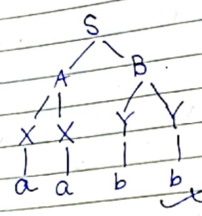


$S \rightarrow AB$ [$A \rightarrow a$]
aB [$B \rightarrow b$]
ab.
∴ 3 steps

$2 \rightarrow 3$

eg: $S \to AB$ → CNF
$A \to XX$
$B \to YY$
$X \to a$
$Y \to b$

→ aabb.



∴ 7 steps

4 → 7.

* if CNF:
no. of steps to derive n string → VI
(2n−1)   n→|W|

* if GNF:
no. of steps to derive string of length n
∴ n

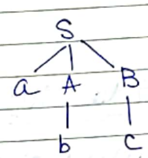
* GNF

[NT ⟶ T(NT)*]

LHS: 1 NT.
RHS: 1T followed by any no. of NTs.

eg: $S \to aAB$ → GNF.
$A \to b$
$B \to c$

→ abc |W| = 3.



S → aAB [A → b]
abB [B → c]
abc

∴ 3 steps

· CFG → CNF/GNF (Why?)

1. No. of steps req. to derive string is answered
CNF: 2n−1
GNF: n. → VI.

*

* CFG → CNF (Conversion)

Q. $S \rightarrow ABC$
$A \rightarrow aA \mid B$
$B \rightarrow bB \mid b$
$C \rightarrow c$

Convert the following CFG to CNF

Step 1: Check if gramm. is simplified or not.

$S \rightarrow ABC$
$A \rightarrow aA \mid bB \mid b$
$B \rightarrow bB \mid b$
$C \rightarrow c$

removing unit productions and useless symbols.

$S \rightarrow ABC$ ∴ $S \rightarrow XC$
$X \rightarrow AB$

$A \rightarrow aA$ ∴ $A \rightarrow X_aA$
$X_a \rightarrow a$

$A \rightarrow bB$ ∴ $A \rightarrow X_bB$
$X_b \rightarrow b$

$B \rightarrow bB$ ∴ $B \rightarrow X_bB$
$X_b \rightarrow b$

∴ $S \rightarrow XC$
$A \rightarrow X_aA \mid X_bB \mid b$
$B \rightarrow X_bB \mid b$
$C \rightarrow c$
$X \rightarrow AB$
$X_a \rightarrow a$
$X_b \rightarrow b$

→ Chomsky Normal Form.

$S \rightarrow XC$ [$X \rightarrow AB$]
ABC [$A \rightarrow b$]
bBC [$B \rightarrow b$]
bbC [$C \rightarrow c$]
bbc

5 steps

$l = 3$ ∴ $2 \cdot 3 - 1 = 5$
($2n-1$ steps)

Q. Convert the following grammars into CNF.

(ii) $E \rightarrow E+T \mid T$
$T \rightarrow (E) \mid id$.

(iii) $S \rightarrow ABCD$
$A \rightarrow aAB \mid a$
$B \rightarrow bB \mid b$
$C \rightarrow c$
$D \rightarrow d \mid dD$.

(iii) $A \rightarrow BC$
$B \rightarrow bb$
$C \rightarrow CC \mid b$

Soln

(i) Ans

$E \rightarrow E+T \mid (E) \mid id$
$T \rightarrow (E) \mid id$

$E \rightarrow E+T \Rightarrow E \rightarrow XT$
$X \rightarrow EX_+$
$X_+ \rightarrow +$

$E \rightarrow (E) \Rightarrow E \rightarrow YX_)$
$Y \rightarrow X_(E$
$X_(\rightarrow ($
$X_) \rightarrow)$

$T \rightarrow (E) \Rightarrow T \rightarrow YX_)$
$Y \rightarrow X_(E$
$X_(\rightarrow ($
$X_) \rightarrow)$

$$\boxed{\begin{array}{l} E \rightarrow XT \mid YX_) \mid id \\ T \rightarrow YX_) \mid id \\ X \rightarrow EX_+ \\ X_+ \rightarrow + \\ Y \rightarrow X_(E \\ X_(\rightarrow (\\ X_) \rightarrow) \end{array}}$$

$\rightarrow$ CNF

(ii) $S \rightarrow ABCD$: $S \rightarrow XY$
$X \rightarrow AB$
$Y \rightarrow CD$

$S \rightarrow XY$
$A \rightarrow X_a X \mid a$
$B \rightarrow X_b B \mid b$
$C \rightarrow c$
$D \rightarrow d \mid X_d D$
$X \rightarrow AB$
$Y \rightarrow CD$
$X_a \rightarrow a$
$X_b \rightarrow b$
$X_d \rightarrow d$

CNF

$$\boxed{1NT \longrightarrow 2NT \mid T}$$

(2n+1 steps).

(iii) $A \to BC$
$B \to X_b X_b$
$C \to X_c C \mid b$
$X_b \to b$
$X_c \to c$

Ans

GNF (Greiback Normal Form)

[n steps are required to derive string of length → n]

$\boxed{[NT \to T (NT)^*]}$

eg. $S \to ABC$
$A \to aA \mid a$
$B \to b$
$C \to c$

↓ ~~do~~ substitute.

$S \to aABC \mid aBC$
$A \to aA \mid a$
$B \to b$
$C \to c$

if abc

$S \to aBC$ [$B \to b$]
abC [$C \to c$]
abc.

∴ 3 steps

Q. $S \to AA \mid b$
$A \to SS \mid a$

∴ A not in GNF.
⇒ no subst.

$A \xrightarrow{\text{dep. upon}} S$
$S \xrightarrow{\text{dep upon}} A$

↓ Same algo.

VI

- $A_i \to A_j \alpha$
  [$i < j$ ⇒ No change]
  [$i > j$ ⇒ Substitution]
  [$i = j$ ⇒ Left Recursion method]
  (NT calls again NT)

$A \to Aa$ ⇒ LR.
$\cdot A \to aA$ ⇒ RR ∴ already in GNF.

A derives A, & A = left most symbol
∴ LRG

eg. $A \to A\alpha \mid \beta$
Left Recursive.

↓

$A \to \beta A'$
$A' \to \alpha A' \mid \varepsilon$

⇒ $A \to \beta A' \mid \beta$
$A' \to \alpha A' \mid \alpha$

Eliminating LR.

$A \rightarrow A\alpha_1 | A\alpha_2 | \cdots | A\alpha_n | \beta_1 | \beta_2 | \cdots | \beta_n$

$$\boxed{\begin{array}{l} A \rightarrow \beta_1 A' | \beta_2 A' | \cdots | \beta_n A' | \beta_1 | \beta_2 | \cdots \beta_n \\ A' \rightarrow \alpha_1 A' | \alpha_2 A' | \cdots | \alpha_n A' | \alpha_1 | \alpha_2 | \cdots \alpha_n \end{array}}$$

VI.

Q $S \rightarrow AA | b$
$A \rightarrow SS | a$

CFG $\rightarrow$ GNF

Soln:

$S = A_1$ (numbering)
$A = A_2$

$\downarrow$

$A_1 \rightarrow A_2 A_2 | b$
$A_2 \rightarrow A_1 A_1 | a$

- $A_1 \rightarrow A_2 A_2 | b$ $\quad i=1, j=2 \ (i<j)$
  $A_2 \rightarrow A_1 A_1 | a$ $\quad i=2, j=1 \ (i>j)$

$\underset{i}{A_2} \rightarrow \underset{j}{A_2} A_2 A_1 | bA_1 | a$

$\hookrightarrow$ LR. (elimin: of LR)

$A \rightarrow \beta A' | \beta$
$A' \rightarrow \alpha A' | \alpha$

$\overset{A_2}{A} \rightarrow \overset{A_2}{A} \overset{A_2 A_1}{\alpha} | \underset{bA_1}{\beta_1} | \underset{a}{\beta_2}$

$\Rightarrow \boxed{A_2 \rightarrow bA_1 A_2' | aA_2' | bA_1 | a} \rightarrow$ GNF.

$A_2' \rightarrow A_2 A_1 A_2' | A_2 A_1$

$\therefore A_2 \rightarrow bA_1 A_2' | aA_2' | bA_1 | a$

$$\boxed{\begin{array}{l} A_1 \rightarrow bA_1 A_2' A_2 | aA_2' A_2 | bA_1 A_2 | aA_2 | b \\ A_2 \rightarrow bA_1 A_2' | aA_2' | bA_1 | a. \\ A_2' \rightarrow bA_1 A_2' A_1 A_2' | aA_2' A_1 A_2' | bA_1 A_1 A_2' | \\ \quad aA_1 A_2' | bA_1 A_2' A_1 | aA_2' A_1 | bA_1 A_1 | \\ \quad aA_1 \end{array}}$$

$A_1 \rightarrow bA_1 A_2$ $\quad [A_1 \rightarrow b]$.
$\quad bbA_2$ $\quad [A_2 \rightarrow a]$.
$\quad bba$.

$\therefore$ 3 steps $\quad \therefore n)$

Page 26

CFG → GNF

Q. $E \to E+T \mid T$
$T \to (E) \mid id$

$E \to E+T \mid (E) \mid id$
$T \to (E) \mid id$

$\Rightarrow$ $E = A_1$
$T = A_2$

$A_1 \to A_1 + A_2 \mid (A_1) \mid id$
$A_2 \to (A_1) \mid id$

$\underset{A}{A_1} \to \underset{A}{A_1} \underbrace{+A_2}_{\alpha} \mid \underset{\beta_1}{(A_1)} \mid \underset{\beta_2}{id}$

Sol^n

$E \to E+T \mid (E) \mid id$
$T \to (E) \mid id$ $\Rightarrow$ $T \to (EX_) \mid id$
$X_) \to)$

$E \to E+T \mid (EX_) \mid id$
$T \to (EX_) \mid id$

Page 27

$\underset{A}{E} \to \underset{A}{E} \underset{\alpha}{+T} \mid \underset{\beta_1}{(EX_)} \mid \underset{\beta_2}{id}$

$$\boxed{\begin{aligned} &E \to (EX_)E' \mid id\,E' \mid (EX_) \mid id \\ &E' \to +TE' \mid +T \\ &T \to (EX_) \mid id \\ &X_) \to) \end{aligned}}$$

$A \to A\alpha \mid \beta_1 \mid \beta_2$

$\underset{A}{E} \to \underset{A}{E} \underset{\alpha}{+T} \mid \underset{\beta_1}{(EX_)} \mid \underset{\beta_2}{id}$

$A \to \beta_1 A' \mid \beta_2 A' \mid \beta_1 \mid \beta_2$
$A' \to \alpha A' \mid \alpha$

$E \to (EX_)E' \mid id\,E' \mid (EX_) \mid id$
$E' \to +TE' \mid +T$
$T \to (EX_) \mid id$
$X_) \to)$

$$\boxed{\begin{aligned} &\text{CNF: } 1NT \longrightarrow NT \cdot NT \mid T \\ &\text{GNF: } 1NT \longrightarrow T \cdot (NT)^* \end{aligned}}$$

→ 2n+1
→ n

$Q \times \Sigma \times T$ ...

* Push Down Automata (PDA)

(FA + Stack)

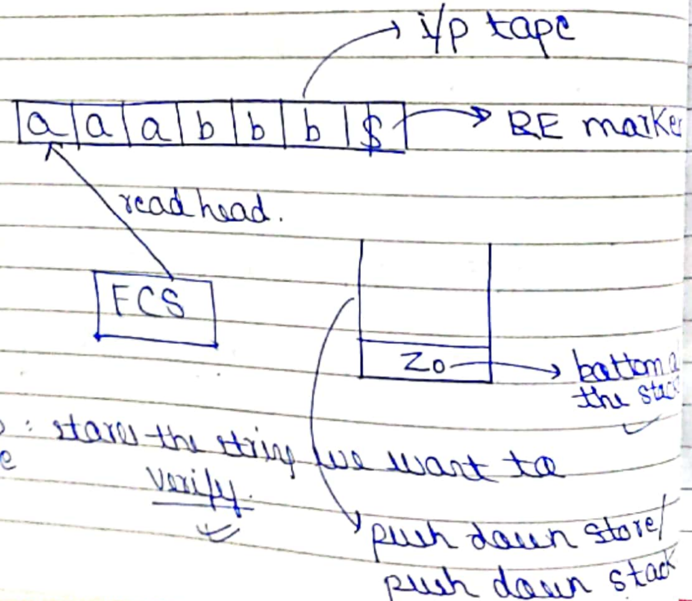
Limitations of FA : 1. memory problem — just the i/p buffer
2. can't compare symbols
eg: $a^n b^n | n$...

eg: $L = \{a^n b^n | n \geq 0\}$.
– same no. of a & b.

∴ add Stack

L(accepted by FA) → RL.

* Structure of PDA



i/p tape : stores the string we want to verify.

FA : RE marker
PDA : RE marker
LBA : LRE & REM (<, >)
TM : B (infinite tape)

$\boxed{Z_0 \in T}$

• 7 Tuple Notation.

$$M = (\underbrace{Q, \Sigma, q_0, \delta, F}_{FA}, \overbrace{Z_0, T}^{stack})$$

Z_0 → initial stack symbol
T → stack alphabets.

$\delta : Q \times \Sigma \rightarrow Q$: DFA.

δ : current state / curr. i/p sym / top of stack

∴ $\delta(PDA)$: $\boxed{Q \times \Sigma \times T \longrightarrow Q\,T^*}$ → DPDA.

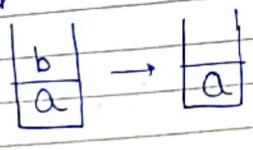
Q → V.I. ; T → TOS. ; Q → next state ; T^* → stack operations (push / pop / bypass).

$\boxed{Q \times \Sigma \times T \rightarrow 2^{Q \cdot T^*}}$ → NPDA

• push
• pop
• No change / bypass.

• PUSH ⇒ (q, State, i/p, TOS) ⟶ (next State, I/p-TOS)

POP (State, i/p, TOS)
↓
(next state, ϵ)



No change : (State, i/p, TOS)
↓
(next State, TOS)

F: per. string accepted / not

---

* PDA : 2 types of acceptance :

(i) → FS acceptance (universal acc.)

(ii) → empty stack acceptance (local acceptance)

(i) no. of stack symbols, accepted by PDA.

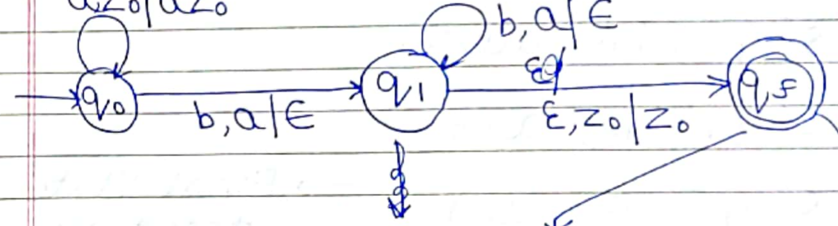
(FA/PDA/LBA/TM → final state accep.)
∴ universal.

(ii) by end of string, if stack becomes empty.
→ empty stack acceptance
→ local to PDA.

eg $a^n b^n \mid n \geq 1$. $L = \{ab, aabb, aaabbb \ldots\}$

aaabbb.

∵ Push all a's
for every b, pop 1 a



aabb

stack empty acceptance.

final state acceptance.

$$M = \left(\underset{Q}{\{q_0, q_1, q_F\}}, \underset{\Sigma}{\{a, b\}}, \underset{q_0}{\{q_0\}}, \right.$$

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, \epsilon)$
$\delta(q_1, b, a) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, Z_0) = (q_F, Z_0)$,

$\{q_F\}, Z_0, \{Z_0, a\}$

* Acceptance of Strings by PDA :

$\delta(q_0, aabb, Z_0)$ — Buffer, Stack — $Q \times \Sigma \times T \rightarrow Q \cdot T^*$

$\delta(q_0, abb, aZ_0)$

$\delta(q_0, bb, aaZ_0)$

$\delta(q_1, b, aZ_0)$

$\delta(q_1, \varepsilon, Z_0) \rightarrow$ Final State Acceptance

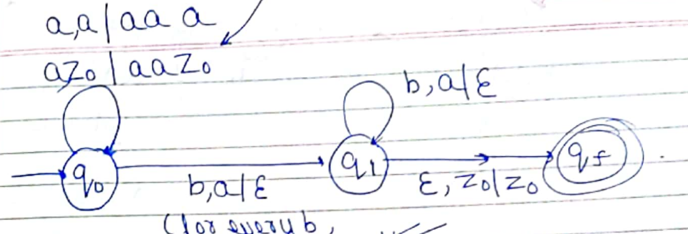
Buffer & Stack both empty.

Q. Construct PDA for following language :

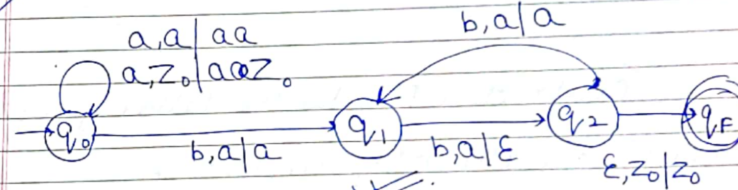
$L = \{a^n b^{2n} \mid n > 0\}$

$L = \{abb, aabbbb, aaabbbbbb \ldots\}$

for every a, push

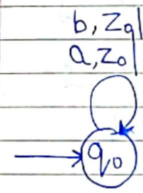


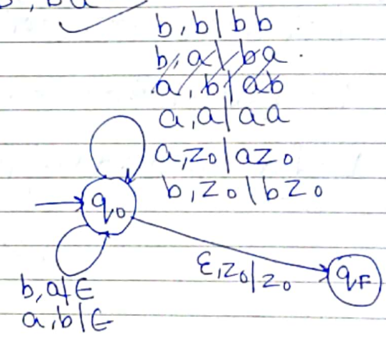
VI

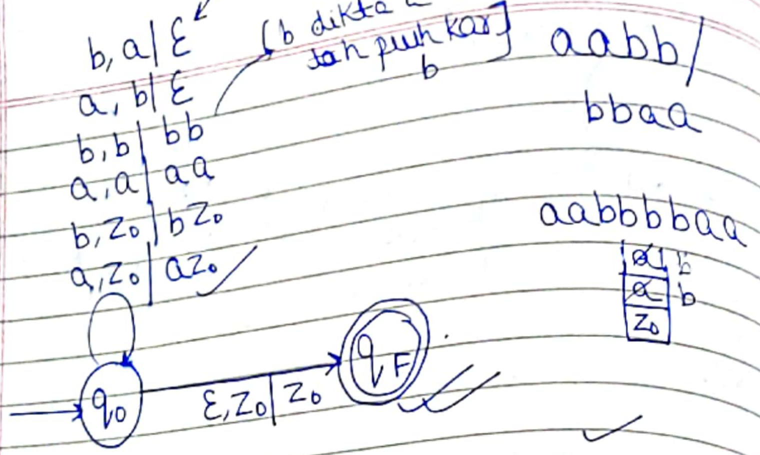


Q. $L = \{w \mid n_a(w) = n_b(w), w \in (a,b)^*\}$

$L = \{abba, abab, baba \ldots$
$\varepsilon, ab, ba \ldots\}$







Q. Construct a PDA for the following lang:

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

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

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

(iv) $L = \{a^{m+n} b^m c^n \mid m, n > 0\}$

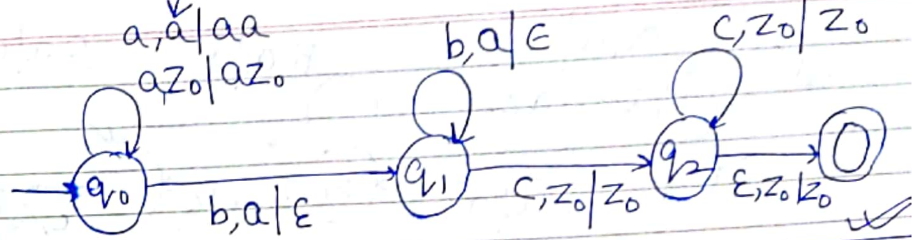
(v) $L = \{a^m b^{m+n} c^n \mid m, n > 0\}$

(vi) $L = \{a^m b^n c^{m+n} \mid m, n > 0\}$

(vii) $L = \{a^m b^n c^n d^m \mid m, n > 0\}$

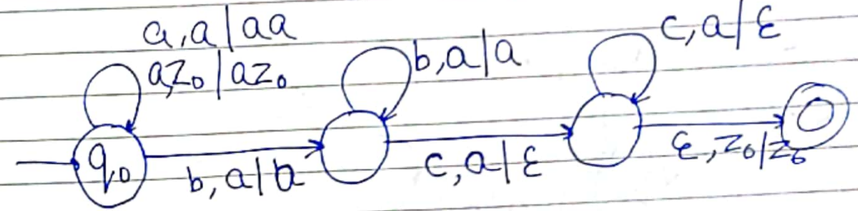
[agar a dikta hai a ke upar, toh a ka push kar over a]

(i)

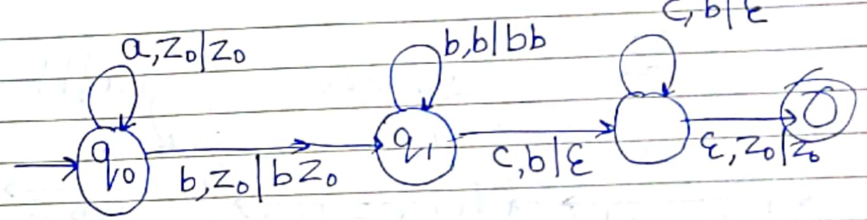


Try:

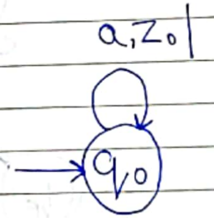
(ii)



(iii)

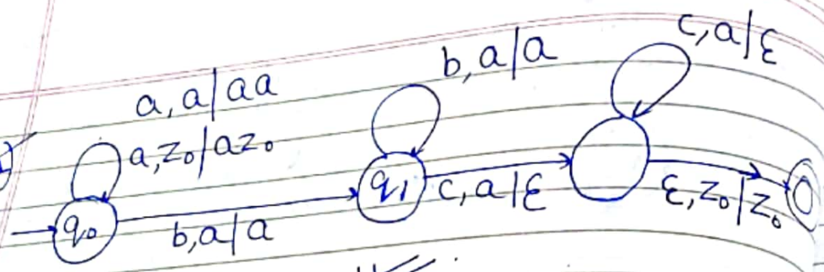


(iv)

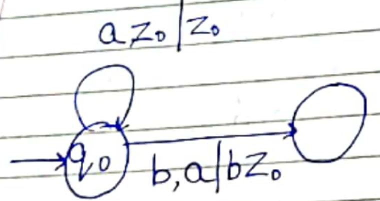


[TRY TRY TILL U SUCCEED]

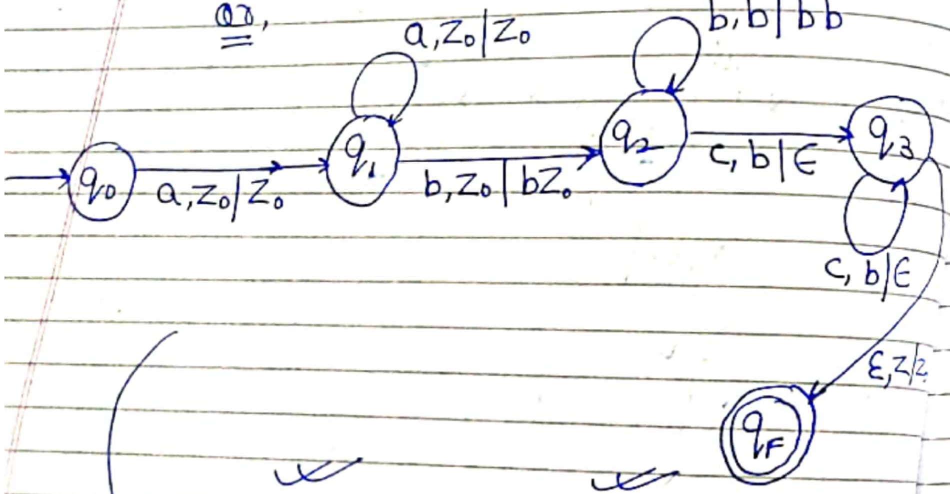
(ii)



(iii)



Or,



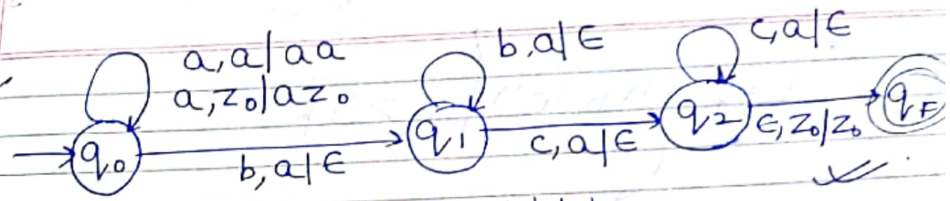
VI

$a^{m+n}b^mc^n$

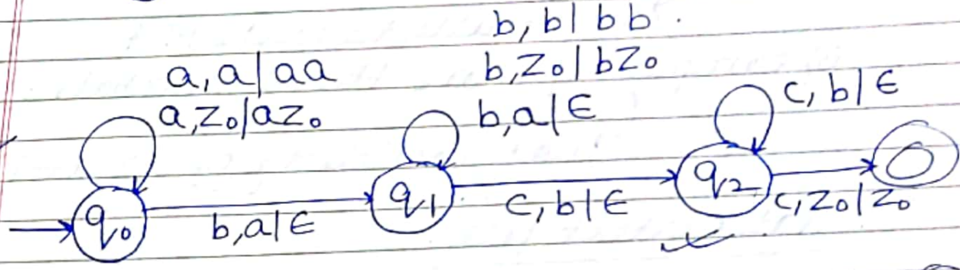
(v) $a^mb^{m+n}c^n$

$a^2b^5c^3$

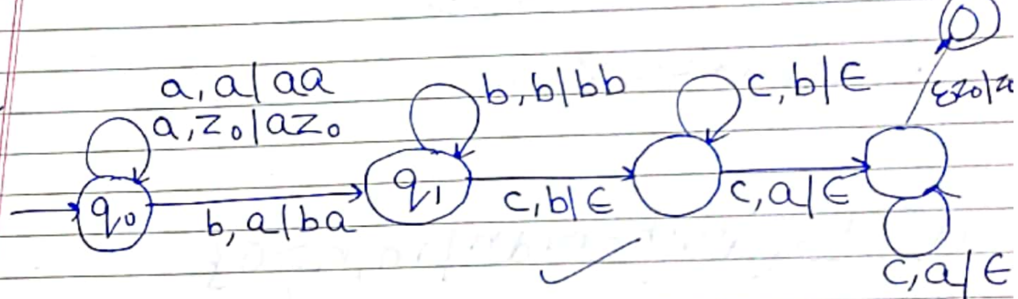
(iv)



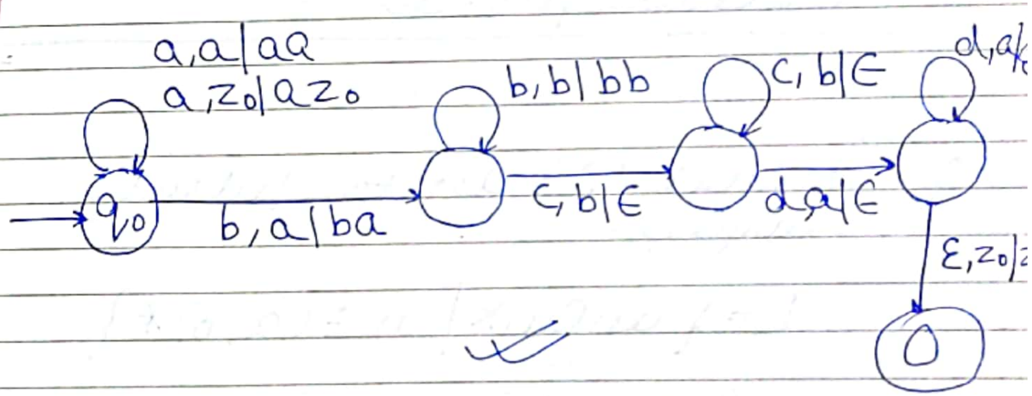
(v)



(vi)



(vii)



Q $L = \{a^n b^n c^n \mid n > 0\}$ → Linear Bound Automata solve

not possible to create PDA

if compar. of more than 2 symbols.

not possible to construct PDA

∴ Not context free.

· $a^n b^n c^{2n}$ X.

Q $L = \{a^m b^n c^m d^n \mid m, n > 0\}$

– Not context free.

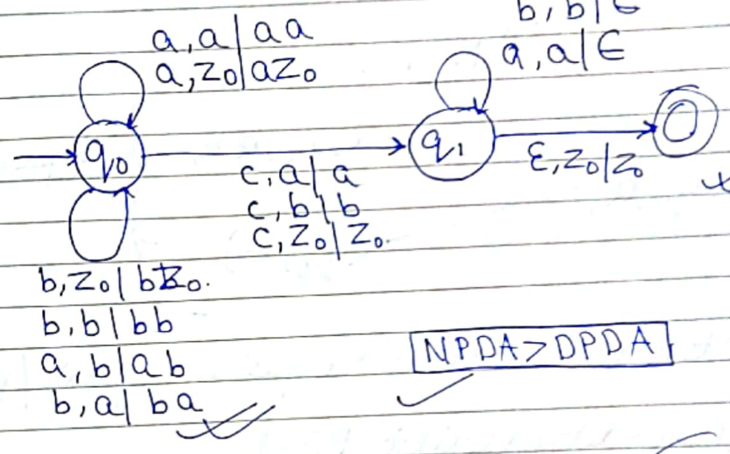
Q Construct PDA for the following language:

$L = \{wCw^R \mid w \in (a,b)^*\}$

$L = \{c, aCa, bCb, abCba, baCab, \ldots\}$

– till now DPDAs. → VI

$\underbrace{ab}_{Push} c \underbrace{ba}_{Pop}$

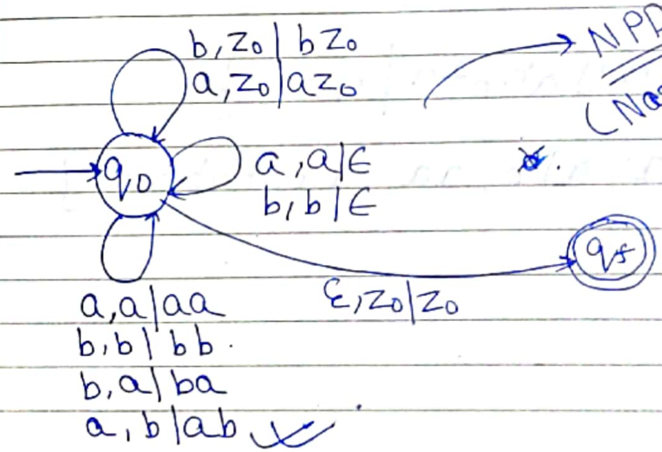


$\boxed{NPDA > DPDA}$

NPDA.

Q $L = \{ww^R \mid w \in (a,b)^+\}$

$L = \{aa, bb, abba, baab, \ldots\}$



→ NPDA (Non-deterministic PDA)

$\delta(q_0, a, a) \to (q_0, aa)$, (q_0, ϵ) } 2 paths.

NPDA.

$\delta(q_0, b, b) \to (q_0, bb)$, (q_0, ϵ) } 2 paths

→ Not known whether to perform push/pop.

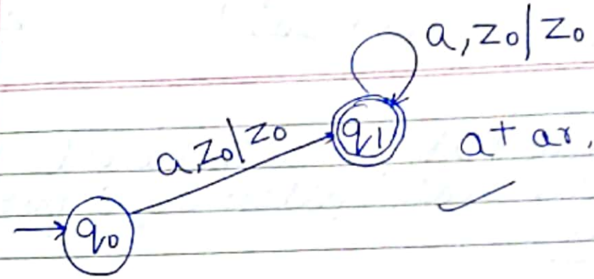
(non-deterministic PDA).

- NPDA & DPDA not equivalent

accepts more no. of languages.

→ $L = \{a^n \cup a^n b^{2n} \mid n > 0\}$

$L = \{a, abb, aa, aabbbb \ldots\}$



a^+ ar, $a^n \mid n > 0$

- PDA: useful for constructing syntax analyser in compiler.

1. balanced parencis.

( ), [ ], { }.

⇓

( ), [( )] ✓

({)} X

pop:), (| ϵ

], [| ϵ

}, { | ϵ.

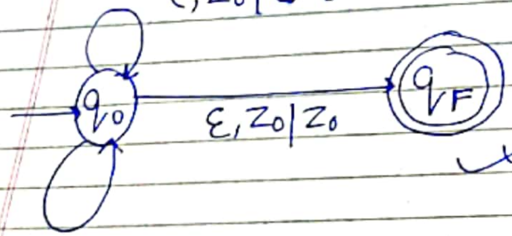
Q Construct a PDA for the balanced parenthesis using following input symbols: → [Balancing Parenthesis].

{

[, Z_0 | [Z_0

{, Z_0 | { Z_0

(, Z_0 | (Z_0



}, (| {((, (| ((

{ [| { [{, { | { {

[, { | [{   ), (| ϵ

[, (| [(   ], [| ϵ

, [| ([}, { | ϵ

{ | ({

Q Let N_F & N_P denote the classes of languages accepted by the NDFA and NPDA respectively and let D_F & D_P denote the classes of languages - DFA & DPDA respectively. Which of the following is true?

(a) $D_f \subset N_f$, $D_p \subset N_p$

(b) $D_f \subset N_f$, $D_p = N_p$

(c) $D_f = N_f$, $D_p = N_p$

(d) $D_f = N_f$, $D_p \subset N_p$

- for every NFA → there is an equivalent DFA.

eg: ww^R possible by NPDA but not DPDA so, $\underline{D_p \subset N_p}$.

→ [If a lang is accepted by DPDA, it has to be accepted by NPDA]

Q. Let P be a NPDA with exactly one state q and exactly 1 symbol Z in its stack alphabet.

State q is both starting & accepting state of PDA.
The stack is initialised with Z before start of the operation of the PDA.
Let the i/p alphabet of the PDA be Σ.

Let L(p) be the language accepted by PDA by reading a string and reaching its accepting state.
Let N(p) be the language accepted by the PDA by reading a string & emptying it's stack.

Which of the following statement is true?

(a) L(p) is necessarily a Σ^* but N(p) is not necessarily a Σ^*.

(b) N(p) is necessarily a Σ^* but L(p) not necessarily a Σ^*.

(c) Both N(p) & L(p) necessarily a Σ^*

Both L(p) & N(p) not necessarily a Σ^*.

(without any i/p symbol also, we can reach the final state).

* Regular Languages (RL), Context Free Languages (CFL), Pumping Lemma.

1. Every finite language is regular. (limited no. of strings).

eg: $L = \{a^n b^n \mid n \leq 100000\}$

· No. of C progs written in 2018 worldwide → regular. ∵ finite.

2. If $\Sigma = \{a\}$, then:

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

$\Rightarrow \{a, aa, aaa \ldots\}$
(AP)

[Some of infinite lang → regular].

[if same CD → regu...

if AP format → regular.

→ $L=\{a^n \mid n \text{ is even}\}$ → regular.

$\{a, aa, aaaa \cdots\}$

→ $L=\{a^n \mid n \text{ is odd}\}$ → regular

→ $L=\{a^{2n} \mid n \text{ is integer}\}$

$\Rightarrow \cancel{\{a^2, a^4, a^8, a^{16} \cdots\}}$

$\Rightarrow \{a^2, a^4, a^6, a^8, a^{10} \cdots\}$

(regular) ✓

→ $L=\{a^{n^2} \mid n \in I\}$

$=\{a^1, a^4, a^9, a^{16} \cdots\}$

(not regular) ✓

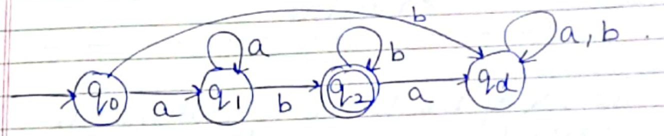
→ $L=\{a^n \mid n \text{ is prime}\}$

$=\{a^2, a^3, a^5, a^7, a^{11}, a^{13} \cdots\}$

(not regular)

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

→ $L=\{a^m b^n \mid m,n>0\}$



RE $= a^+ b^+$ ✓

→ $L=\{a^m b^n \mid m>n\}$ → TM

· need comparison.

(So not regular).

→ $L=\{a^m b^n \mid m=n\}$

or

$L=\{a^n b^n \mid n>0\}$

→ CFL (not regular)

→ $L=\{ww \mid w \in (a,b)^*\}$ ×

→ $L=\{ww^R \mid w \in (a,b)^*\}$ ×

→ NDPDA

4. If $\Sigma = \{a, b, c\}$
(if powers are different & no comparison
→ regular.

Q. Which of the following statement is true?

$S_1 : \{0^{2n} \mid n \geq 1\}$

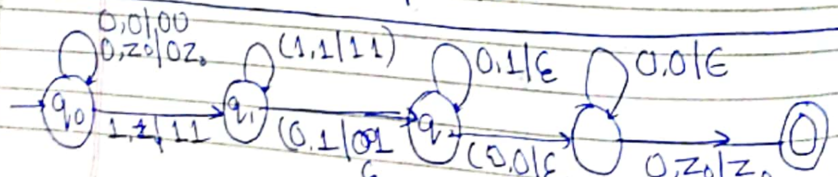
$S_2 : \{0^m 1^n 0^{m+n} \mid m, n \geq 1\}$

(a) Only S_1 is correct.
(b) Only S_2 is correct.
(c) Both S_1 & S_2 are correct.
(d) None of S_1 & S_2 are correct.

Q. $L_1 = \{0^2, 0^4, 0^6, 0^8 \ldots\}$

L_2 → comparison is reg.

L_1 : regular
L_2 : context free ✓



[PDA].

Q. Consider the following languages:

$L_1 = \{ww \mid w \in (a, b)^*\}$.
$L_2 = \{ww^R \mid w \in (a, b)^*,$
$w_R = \text{reverse of } w\}$

$L_3 = \{0^{2i} \mid i \in I\}$

$L_4 = \{0^{i^2} \mid i \in I\}$

Which of the following is regular?

(a) Only L_1 & L_2
(b) Only L_3 & L_4
(c) Only L_2, L_3 & L_4
(d) Only L_3 ✓

Q. Which of the following languages are regular?

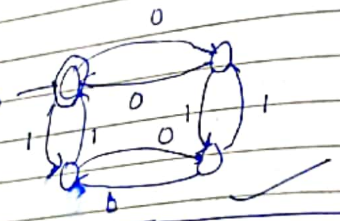
$L_1 = \{0^m 1^n \mid m, n \leq 10000\}$.

$L_2 = \{w \mid w$ reads both forward & backward$\}$ → WW^R

$L_3 = \{w \mid w \in (0, 1)^*$ and w contains even no. of 0s & even no. of 1s$\}$.

(a) L2 and L3
(b) L1 and L2
(c) L1 and L3
(d) L3 only.

· L_3 =



Q. Which of the following language is regular?
GATE

(a) $\{ww^R \mid w \in (0,1)^{*}\}$

(b) $\{ww^R x \mid x, w \in (0,1)^{*}\}$

(c) $\{wxw^R \mid x, w \in (0,1)^{*}\}$

(d) $\{xww^R \mid x, w \in (0,1)^{+}\}$

$L_c = \{0110 1110 ..$

- if we can define RE, we can claim it to be regular.

wxw^R

$w = 10$
$x = 011$
$w^R = 01$

$\Rightarrow$ 1001101.

$w = 0101$
$x = 10$
$w^R = 1010$

$\Rightarrow$ 0101 10 1010

(c) $0.(0+1)(0+1)^{*}0 +$
$1.(0+1)(0+1)^{*}1$

$\underset{w}{1}\underbrace{010}_{x}\underset{w^R}{1}$ → accepted by RL too.

Context Free Languages :

1. **All RLs are CFLs.**

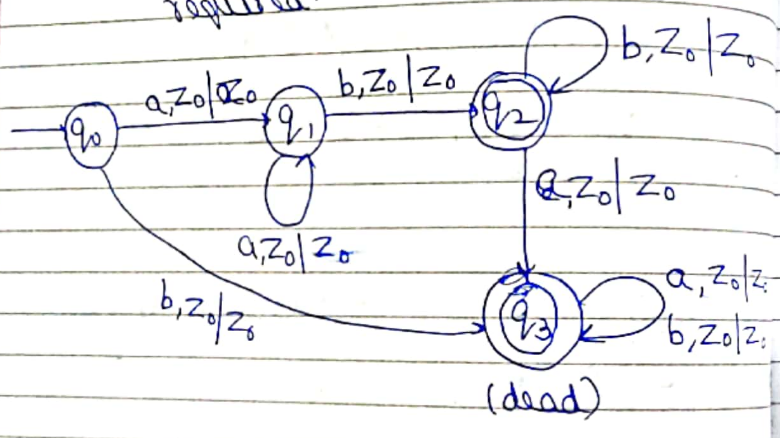
eg: a^n can be generated by PDA

- if can be gen. by FA, can be gen. by PDA

$L = \{a^m b^n \mid m,n > 0\}$ = regular.

PDA

- stack empty acceptance not required.



- All FLs are context free & regular.

2. If $\Sigma = \{a\}$

→ $L = \{a^n \mid n \rightarrow \text{prime}\}$

$L = \{a^2, a^3, a^5, a^7, a^{11} \ldots\}$ ✗

Not CFG.

[if some CD → CF also]

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

→ ⓐ $L = \{a^n b^n \mid n > 0\}$ → **CFL**

→ $L = \{a^n b^n c^n \mid n > 0\}$ → ✗

→ $L = \{a^m b^n c^{m+n} \mid m,n > 0\}$ → **CFL**

→ $L = \{a^m b^n c^m d^n \mid m,n > 0\}$ ✗

TOS = b
cant match with c.

→ $L = \{a^m b^n c^n d^m \mid m,n > 0\}$

Context Free.

→ $L = \{a^{m+n} b^{m+n} c^m \mid m,n > 0\}$
↳ not CF

but, $L = \{a^{m+n} b^{m+n} c^z \mid m,n,z > 0\}$
↳ context free.

∴ (more than 2 comp.)

→ $L = \{ww^R \mid w \in (a,b)^*\}$
↳ Non-deterministic context free language.

→ $L = \{wcw^R \mid w \in (a,b)^*, c\}$
↳ DCFL ✓

Q Let $L_1 = \{0^{m+n} 1^n 0^m \mid m,n \geq 0\}$
$L_2 = \{0^{n+m} 1^{n+m} 0^m \mid m,n \geq 0\}$
$L_3 = \{0^{n+m} 1^{n+m} 0^{n+m} \mid n,m \geq 0\}$

L_3 : CSL

Which of these lang. are not context free?

(a) L_1 only
(b) L_3 only
(c) L_1 & L_2 only
(d) L_2 & L_3 only ✓

• $w\#w^R \rightarrow$ CFL.

Q Which of the following language over $\Sigma = \{a,b,c\}$ is accepted by the DPDA?

(a) $w \cdot c \cdot w^R \mid w \in (a,b)^*$
(b) $L = \{w \cdot w^R \mid w \in (a,b,c)^*\}$
(c) $L = \{a^n b^n c^n \mid n > 0\}$
(d) $L = w \mid w$ is a palindrome over $\{a,b,c\}$.

* Pumping Lemma

- used if the lang. is regular / not ?
  ↳ decidable ✓

- Some of the lang. are proved as not regular due to Pumping Lemma.

$L = \{ \varepsilon, aa, aaaa, \ldots \}$

- (same set of string is pumped again & again)

Q. $L = \{ a^n b^n \mid n > 0 \}$

Prove L is not regular.

— Solⁿ:

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

1. $|z| \geq n$

$z = uvwx$

$z = uvw$

Conditions

$$\boxed{\begin{array}{l} 1.\ |z| \geq n \\ 2.\ |uv| \leq n \\ 3.\ |v| \geq 1 \end{array}}$$

- $|uv| \leq n$
- $|v| \geq 1$
- $z = uv^i w$

$i = 0$ to k times.

$\therefore z = uv^i w \quad i \in (0 \text{ to } k)$

Let $z = aabb \qquad n = 2$.

$|z| \geq n \quad \therefore 4 \geq 2$ ✓

$z = \underset{u}{a}\,\underset{v}{a}\,\underset{w}{bb}$

$|uv| \leq n \quad \Rightarrow 2 \leq 2$ ✓

$|v| \geq 1 \qquad 1 \geq 1$ ✓ (atleast same string to be pumped).

- $i = 1$

$\therefore z = aa^1bb = aabb$ ✓

$i = 2$

$z = aa^2bb \Rightarrow aaabb$ ✗. $\notin L$.

$\therefore$ (Not regular).

Q $L = \{a^n / n \text{ is prime}\}$

Soln

$L = \{a^2, a^3, a^5, a^7, a^{11} \dots \}$

$n = 5$

$z = \underbrace{aa}_{u}\underbrace{aa}_{v}\underbrace{a}_{w}$

1. $|z| \geq n$ ✓
2. $|uv| \leq n$ ✓
3. $|v| \geq 1$ ✓

$z = uv^i w$

$\Rightarrow aa \cdot aa^i \cdot a \qquad i = 2$

$aa \cdot aaaa \cdot a$

$\Rightarrow a^6$ ✗

∴ Not regular

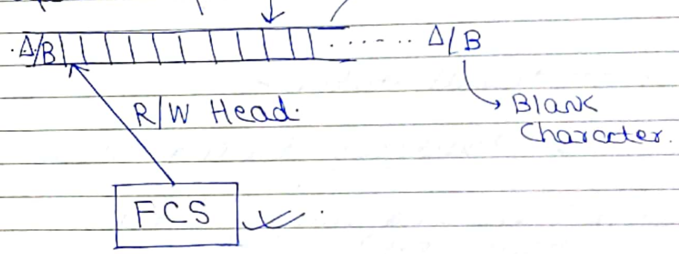
Drawbacks of FA & PDA

- Math operations not performed using FA, PDA
- unidirectional. (head moment is unidirection)
- only read operation is allowed.

* Turing Machine

- same as FA
- infinite tape

i/p tape



- head moment is bidirectional.

$M = (Q, \Sigma, q_0, \delta, F, \underset{B}{\overset{}{\Delta}}\text{ or }, T)$

7 tuples.

tape alphabets
· (i/p symb. + Δ/B. write sym.)

$\boxed{\delta : Q \times T \rightarrow Q \times T \times \{L, R\}}$

write same sym into buffer

· The tape contains $\underbrace{\text{i/p / op / } \Delta}_{T}$ symbols

· F → hault state

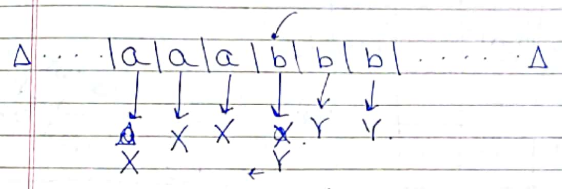
FS → outgoing edge is available

Q · (There is no outgoing edge from hault state)

· Δ/B: end/start of String.

(transition diag / trans. table).

→ $L = \{a^n b^n \mid n > 0\}$

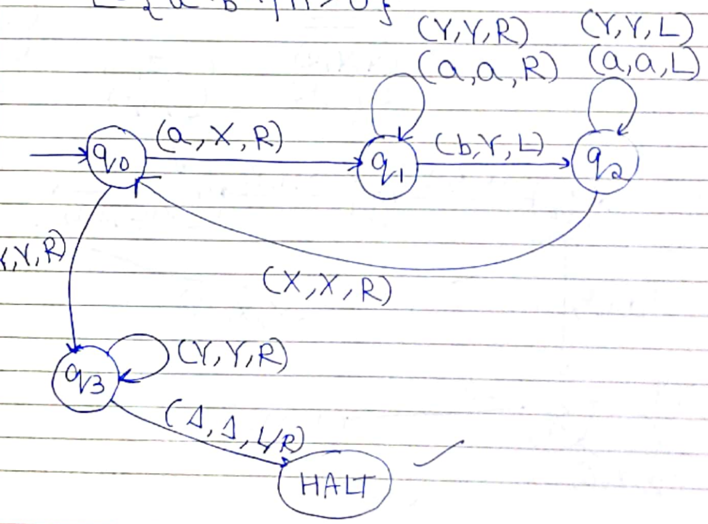


(search till we get first b).

Q Construct a TM:

$L = \{a^n b^n \mid n > 0\}$

a a a b b b → X X X Y Y Y



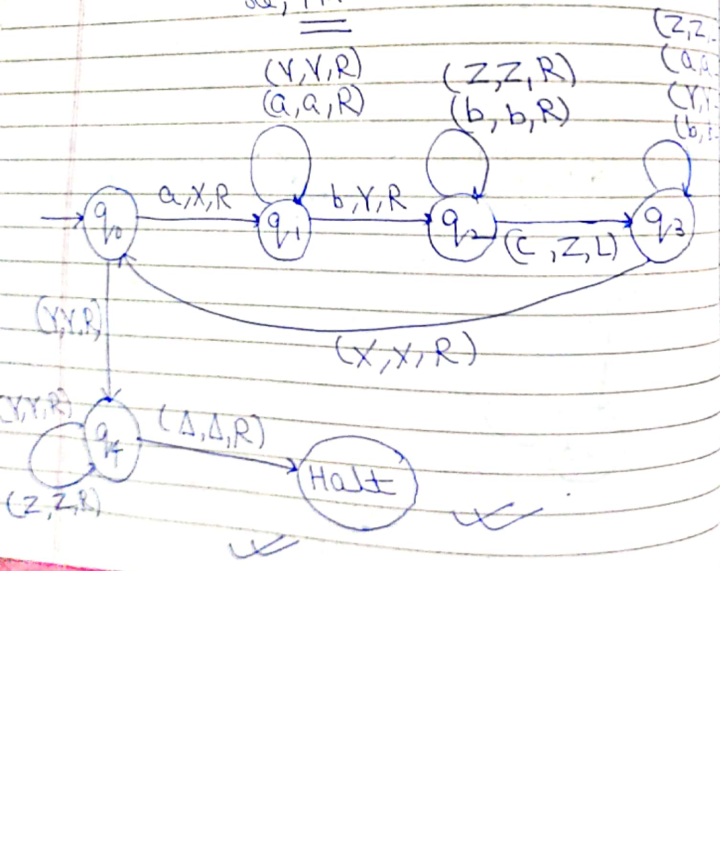
δ	a	b	X	Y	Δ
→ q_0	(q_1,X,R)	–	–	(q_3,Y,R)	–
q_1	(q_1,a,R)	(q_2,Y,L)	–	(q_1,Y,R)	
q_2	(q_2,a,L)	–	(q_0,X,R)	(q_2,Y,L)	–
q_3	–	–	–	(q_3,Y,R)	HALT
HALT	–	–	–	–	–

no outgoing edge from HALT state

Q $\{a^n b^n c^n \mid n > 0\}$

↳ not be done by PDA.

So, TM.



Lecture 5 [TOC]

Q Construct TM for

$L_1 = \{wcw^R \mid w \in (a,b)^*\}$

$L_2 = \{ww^R \mid w \in (a,b)^*\}$

Soln:

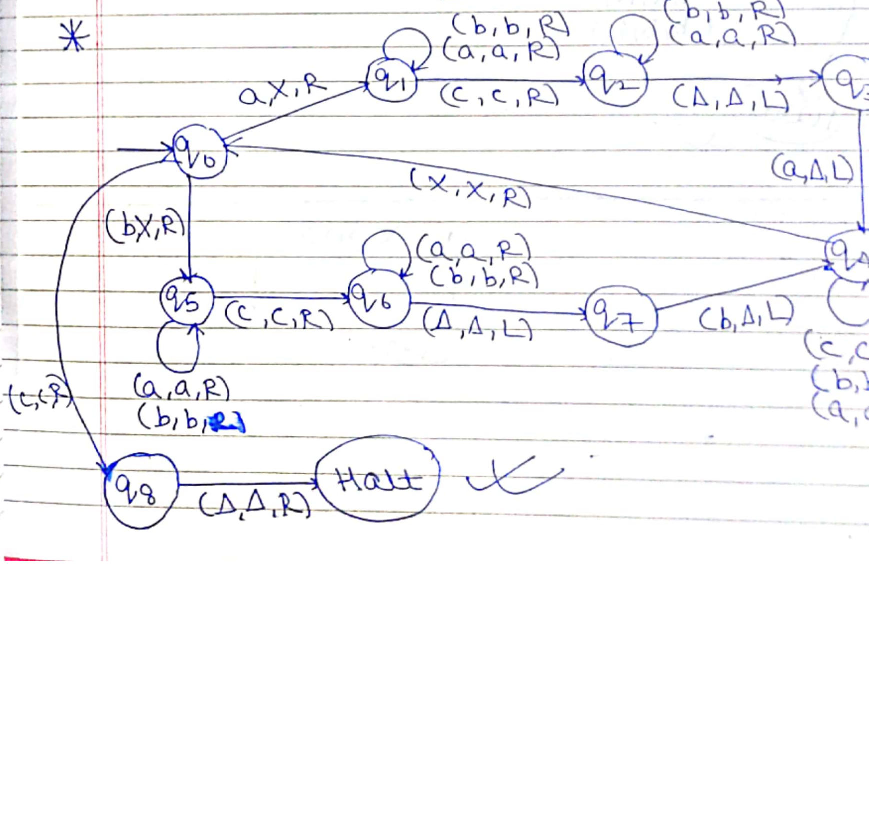
$L_1 = \{aca, bcb, abcba \ldots$

a b b c b b a Δ...

b b a c a b b

X X X Δ Δ Δ

*

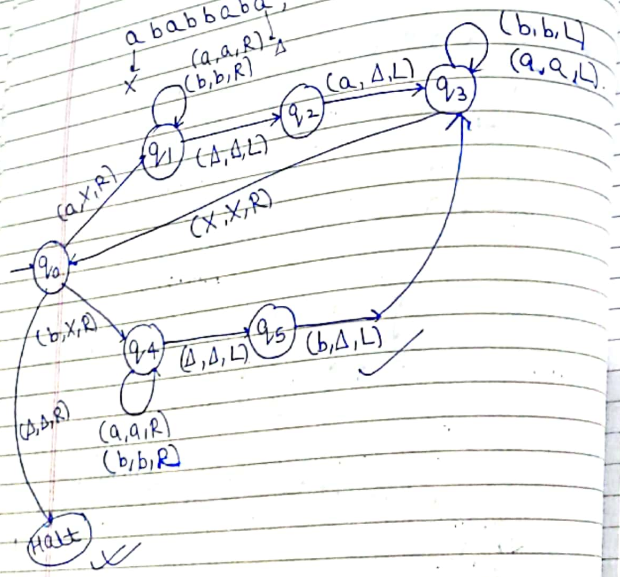


even length palindrome

$L_2 = \{ ww^R, w \in (a,b)^* \}$

babaabab.

a babbaba,



ababbaba.

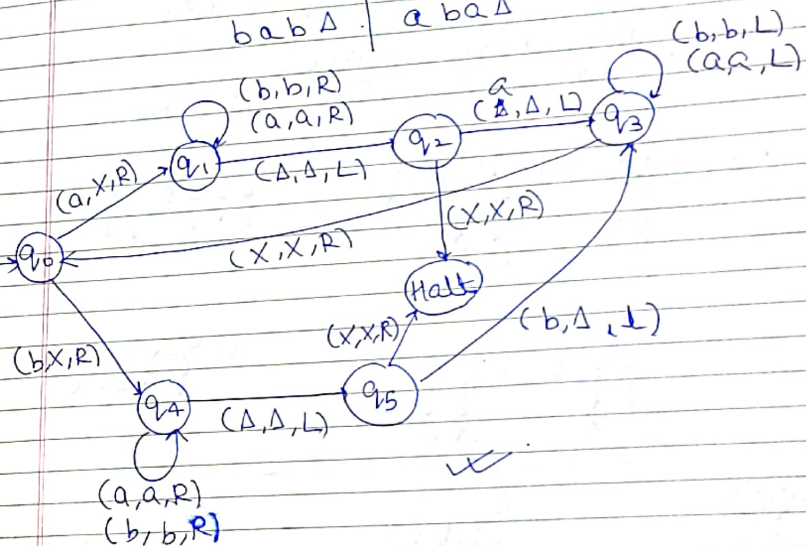
X Δ

VI

Q odd length palindrome

$L = \{$ ababbbaba, bababab.

bab Δ | a ba Δ



eg: abab b baba.

X X X X X Δ Δ Δ Δ

Q.
a = 111, b = 11 o/p → a>b

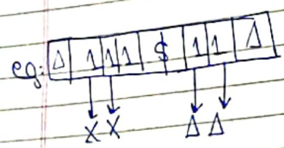
111 _ 111 o/p → a=b.

11 _ 111 o/p → a<b

FA → not possible.

TM: Comparator (Comp. of 2 both)

eg:

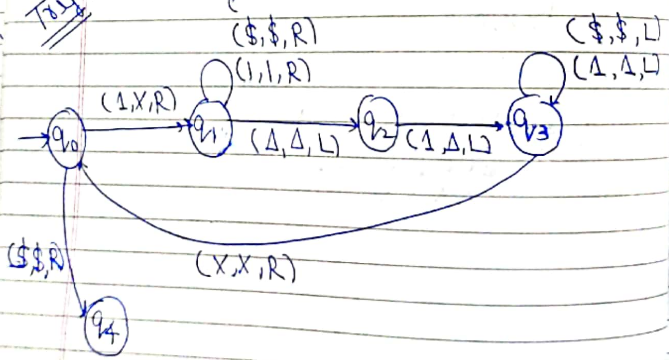


1 1 1 $ 1 1 Δ
X X X Δ Δ

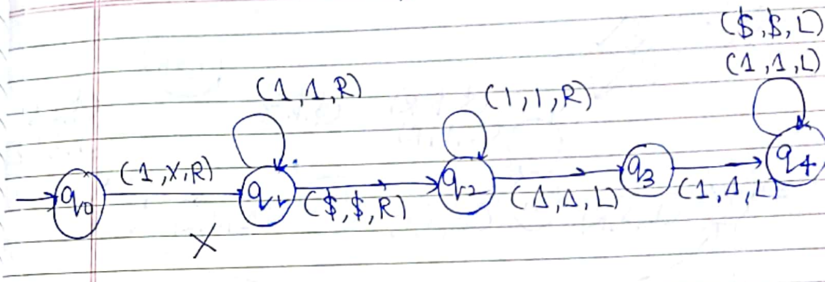
i) $ → Δ : a=b
$ → 1 : a<b.

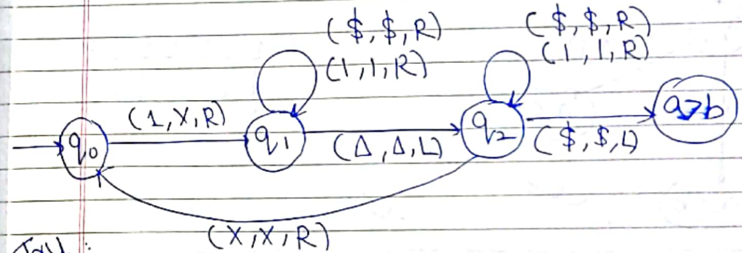
1 1 $ 1 1 1 Δ
X X Δ Δ

Try:



1 1 1 $ 1 1 Δ
X Δ

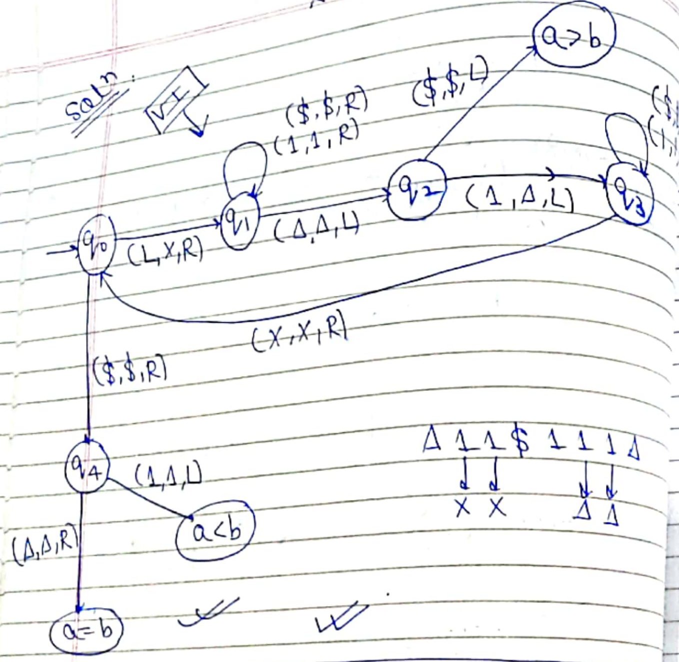




Try:

Δ 1 1 1 1 1 1
X X X Δ Δ

Solⁿ: V.I.



Δ 1 1 $ 1 1 1 Δ
X X Δ Δ

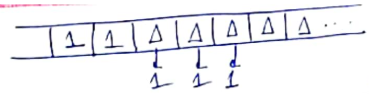
Q $f(x) = x+3$

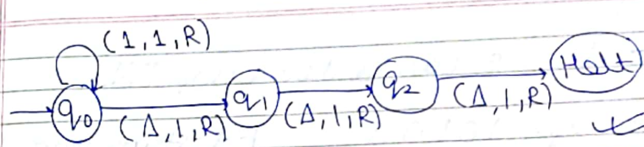
$1+3=4 \quad x \to 1$

|1|1|Δ|Δ|Δ| 1 → 4 5 1s

$x = 2$

$f(x) = x+3$



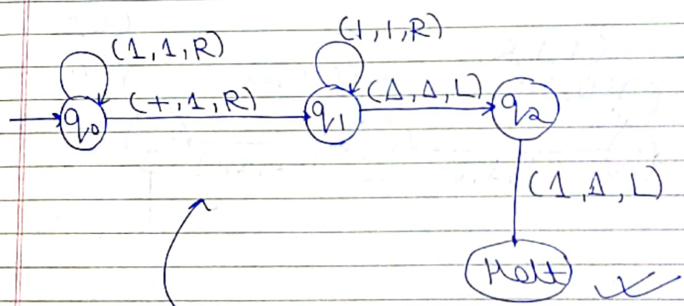


Q $x+4$

2 ones + 3 ones = 5 ones

1 1 + 1 1 1
1 1 1 1 1 Δ

(same will be concat of 2 strings).



Addition of 2 unary no.s

- 1s, 2s can be done using TM

eg: 1010Δ   FA starts from LSB

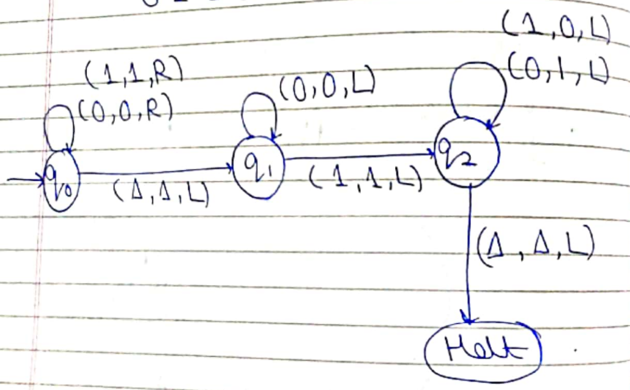
0110.

Q Construct TM for generating output as 2s complement of a given binary number.

1 0 1 0 Δ

↓

0 1 1 0



Subtraction of 2 unary no:s

X − Y

Δ 1 1 1 − 1 1 Δ

Δ Δ Δ   Δ Δ Δ

* Multiplication of 2 no:s :

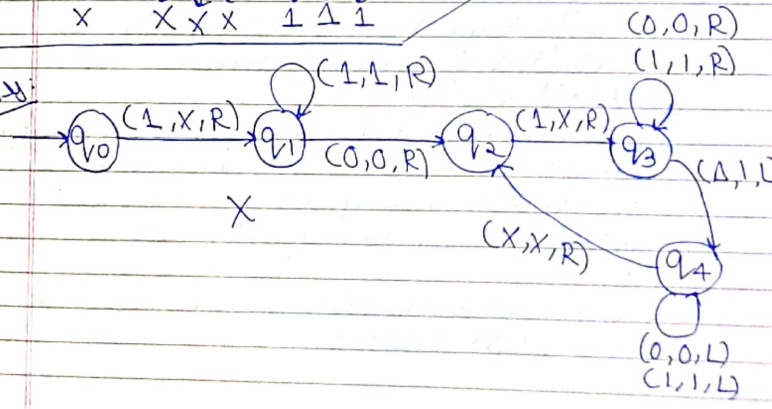
Δ 1 1 0 1 1 0 Δ

X   X X

eg: ... 1 1 0 1 1 1 0 Δ Δ Δ ...

X   X X X   1 1 1

Try:

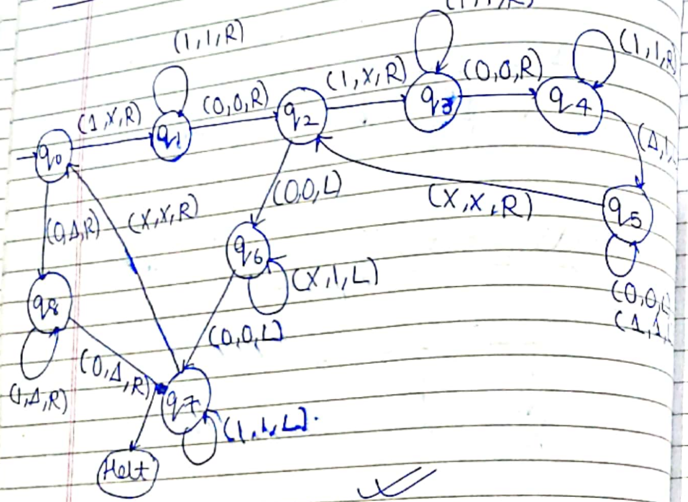


Page 72

ΔΔΔ 1 1 0 1 1 1 0 Δ Δ Δ Δ
X X X X 1 1 1

ΔΔΔ 1 1 0 1 1 1 0 Δ Δ Δ Δ
X X X X X 1 1 1 1 1 1

Solⁿ:

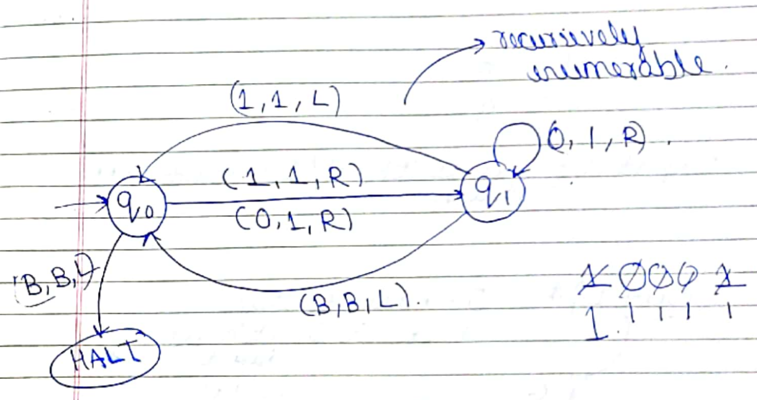


Q. A single tape turing machine M has 2 states q_0 and q_1 where q_0 is initial state.
The tape alphabet (T) of M is {0,1,B} where B → Blank Symbol used to indicate end of an i/p string.
where {0,1} are i/p symbols.

The transition function of M is described in the foll. table:

Page 73

	0	1	B
q_0	$(q_1, 1, R)$	$(q_1, 1, R)$	HALT
q_1	$(q_1, 1, R)$	$(q_0, 1, L)$	(q_0, B, L)



Which of the following statement is true about M?

(a) M doesnot hault on any string in $(0+1)^+$.
(b) M doesnot hault on any String in $(00+1)^*$
(c) M haults on all strings end with 0.
(d) M haults on all strings ends with 1.

1	B	B

1 B

0 B

1 B

· if string ~~ends~~ starts with 0/1 : won't halt

$(00+1)^*$ 0 → doesn't belong
but, in 0 : it ~~halts~~ doesn't

{ ∈, 00, 1, 001 . . . }

* Turing M. accepted language is recursively innumerable language.

* Recursive & Recursively Enumerable languages.

L → TM → Y / N.

· Recursive language : accepted/rejected by TM.

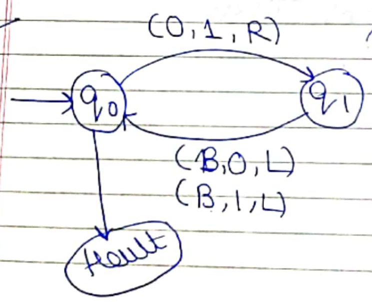
→ L is accepted / enters in a loop.
→ not rejected.
→ recursively enumerable.

[Haulting problem of TM]

→ By reading i/p symbols, TM won't never hault.

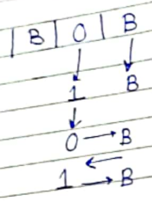
→ undecidable.

Q



→ haulting problem of TM.

Recursively Enumerable ... Loop



- By reading any i/p symbol, goes into loop → recursively enumerable language
→ Halting problem → Accept / Reject / Don't Know

· Recursive → Accept
→ Reject

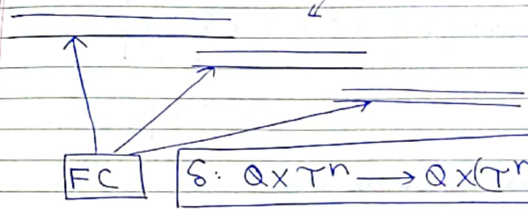
---

* Types of TM:

(i) Single Tape → normal TM.
(ii) Multi Tape
(iii) Multihead TM.
(iv) TM with stay option.
(v) Non-deterministic TM.
(vi) Multi-dimensional TM.
(vii) Universal TM.

(ii) Multi Tape

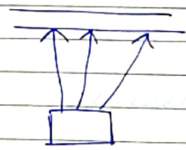
Multitape Multihead.



$$\boxed{\delta : Q \times \Gamma^n \longrightarrow Q \times (\Gamma^n) \times (L, R)^n}$$

- we can read i/p symbols from diff tapes at a time.

(iii) Multihead : Same concept.



O O O

Multi head Single Tape

$$\boxed{\delta : Q \times \Gamma^n \longrightarrow Q \times \Gamma^n \times (\{L, R\})^n}$$

$\boxed{Q \times \Gamma \rightarrow Q \times \Gamma \times \ldots}$

(iii) TM with stay option.

$$\boxed{Q \times \Gamma \rightarrow Q \times \Gamma \times \{S\}}$$

nor left/right.

(iv) Multi-directional

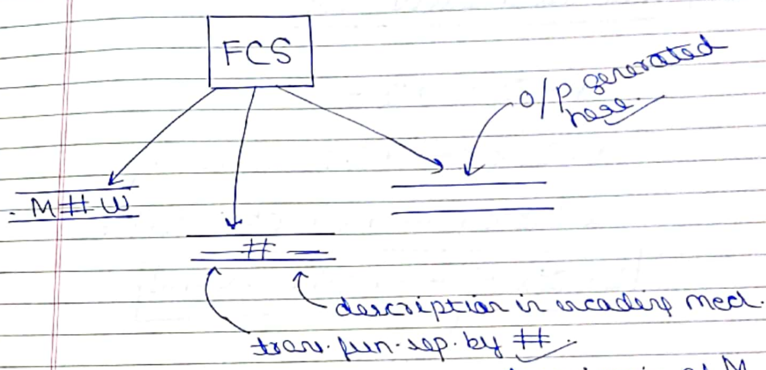
$\{L, R, \underbrace{U, D}_{\text{vertical}}\}$

$\Rightarrow \boxed{Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}}$

(v) Universal TM:

- It accepts all languages.
- 3 tapes

no need to design the TM structure again & again.
→ Just change the second tape (its encode).



$\delta(q_0, 0) = (q_1, X, R)$ → description of M.

$q_0 \rightarrow 0$	$0 \rightarrow 1$	$L \rightarrow 1$
$q_1 \rightarrow 1$	$1 \rightarrow 11$	$R \rightarrow 11$
$q_2 \rightarrow 2$	$B \rightarrow 111$	
	$X \rightarrow 1111$	

For add, sub, mul → only 2nd tape is required.
→ no need to change the structure.

(vi) Non-deterministic TM :

$Q \times \Sigma \rightarrow Q$

$Q \times \Sigma \times \Gamma \rightarrow Q.\Gamma^*$: PDA.

$Q \times \Sigma \times \Gamma \rightarrow 2^{Q.\Gamma^*}$: NPDA

$\boxed{Q \times \Sigma \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R\}}}$ → NTM

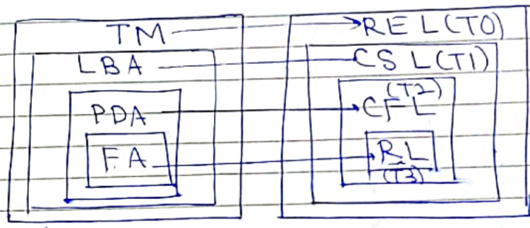
$\boxed{Q \times \Sigma \times \Gamma \rightarrow 2^{Q.\Gamma^*}}$ → NPDA

* CHOMSKY HIERARCHY

→ no. of m/cs,
no. of languages

- FA, PDA, TM, LBA.

FA



- RL derived from RE
- CFL derived from CFG
- CSL derived from CSG.
- REL derived from unrestricted Gramm.
  → any format is ok.

- TM → Lang. acceptance
  → Math operations.

→ Type 0 : TM.
  → Lang : REL
  → M/C : TM
  → Gram : UG

→ Type 1 : LBA.
→ Type 2 : PDA.
→ Type 3 : FA.

* Context Sensitive Grammar

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

$\alpha \longrightarrow \beta$

1. $\alpha \in (V, T)^*$, $\beta \in (V, T)^*$ } CSG.
2. $|\beta| \geq |\alpha|$.

eg: $Aa \longrightarrow aB$ ✓

$|\beta| = 2 \geq |\alpha| = 2$.

$A \longrightarrow \varepsilon$ X

$|\beta| = 0 < |\alpha| = 1$

- (In CSG : ε productions are not allowed). ✓

[Except ε prodn, all prod in CFG are CSG] ✓ → VI

Q Find the language for the following Grammar.

$S \longrightarrow aSBC \mid aBC$
$CB \longrightarrow BC$
$aB \longrightarrow ab$
$bB \longrightarrow bb$
$bC \longrightarrow bc$
$cC \longrightarrow cc$

→ CSG

Soln

$L = \{$

$S \longrightarrow aBC$ [$aB \rightarrow ab$]
abC [$bC \rightarrow bc$]
abc.

$S \longrightarrow aSBC$ [$S \rightarrow aSB$]
$aa\cancel{S}B$
aBC

aa

$S \longrightarrow aSBC$ [$S \rightarrow aBC$]
$aaBCBC$
ab _ _
bc
aa

$S \rightarrow aSBC$ [$S \rightarrow aBC$]
$\rightarrow aaBCBC$ [$aB \rightarrow ab$]
$\rightarrow aabCBC$ [$CB \rightarrow BC$]
$\rightarrow aabBCC$ [$bB \rightarrow bb$]
$\rightarrow aabbCC$ [$bC \rightarrow bc$]
$\rightarrow aabbcC$ [$cC \rightarrow cc$]
$aabbcC$
$aabbcc$ $\rightarrow$ CSL

$\therefore L = \{a^n b^n c^n \mid n > 0\}$

Q Find the language for the following CSG.

$S \rightarrow ABC \mid ABCS$
$AB \rightarrow BA$
$BC \rightarrow CB$
$BA \rightarrow AB$
$CB \rightarrow BC$
$A \rightarrow a$
$B \rightarrow b$
$C \rightarrow c$

$L = \{abc, abcabc, acbabc \ldots\}$

ABCS [$AB \rightarrow BA$]
BACS [$BA \rightarrow AB$]
ABCS [$BC \rightarrow CB$]
ACBS
abc

ABCS [
ACBS [
acbabc

ABCS
BACS
bacABC~~S~~ABC
abcabc.

$\therefore L = \{n_a = n_b = n_c(w) \mid w \in (a,b)^+\}$

UNIVERSAL TM

Q X_1 will always be 0.
X_2 will be 1.
X_3 will be B.

Refer direction $L \rightarrow D_1$ and refer direction $R \rightarrow D_2$
Encode the transition rule :

$\delta(q_i, x_j) = (q_K, x_L, D_m)$ for some integers i, j, K, l, m by the string $0^i 1 0^j 1 0^K 1 0^l 1 0^m$.

(i) Let TM-{$M = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, \{q_1\}, B, q_3)$

$\Upsilon \rightarrow \{0, 1, B\}$.

Encode the transition:

$$\delta(q_3, 1) = (q_2, 0, R)$$

(a) 0100100010100

(b) 0001010100 1010

(c) 00010010010100

(d) None of the above.

$X_j = 1$

$$\delta(q_i, X_j) = (q_k, X_l, D_m)$$

$i=3, j=2, k=2, l=1, m=2$

$(0^i 1 0^j 1 0^k 1 0^l 1 0^m)$.

⇒ 00010010010100 (c). Ans.

the transition is encoded into the input buffer.

Q. If the transition function:

δ of given TM consist of the following rules:

$$\delta(q_1, 1) = (q_3, 0, R)$$
$$\delta(q_1, 0) = (q_2, 1, R)$$
$$\delta(q_2, 1) = (q_2, 0, R)$$
$$\delta(q_2, B) = (q_3, 1, R)$$
$$\delta(q_3, B) = (q_3, 1, L)$$

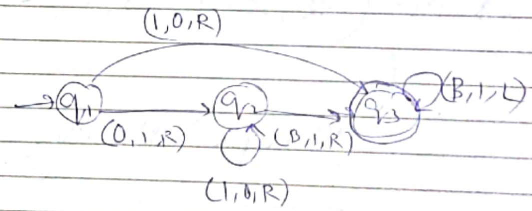
Which of the following is valid code of a transition rule of a given TM.

(a) 01010100101

(b) 0011010100

(c) 0001000100010010

(d) 101001000100

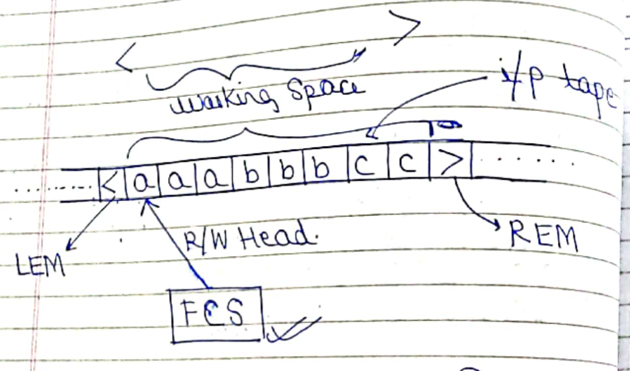


1 0
~~0~~1 0101 00101

~~0~~ 0010001000
1

* LBA (Linear Bounded Automata)

- Tape length is finite.
- Static memory (need to specify mem. before) ∴ Bounded
- By default: Left Boundary & Right Boundary symbol.



* $M = (Q, \Sigma, q_0, \delta, F, <, >, T)$

8 tuple Notation

$\boxed{\text{TM : 7 tuple notation}}$

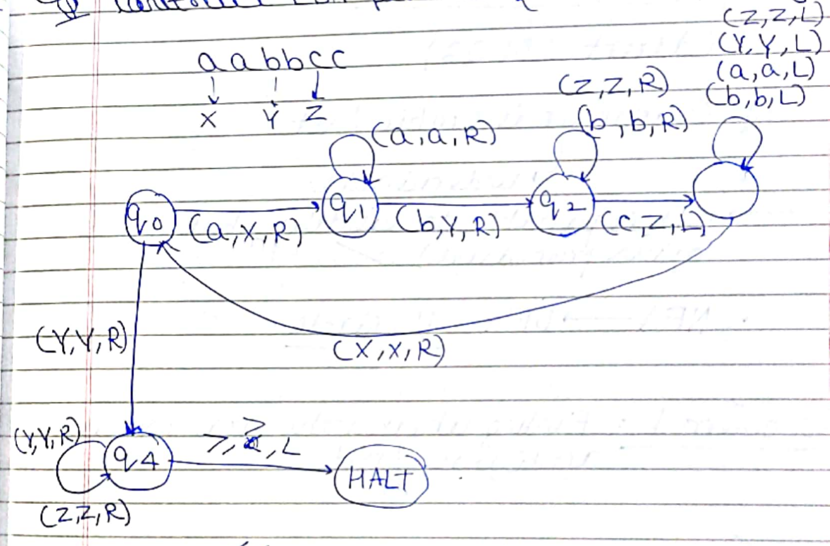
Transition function

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

if we read $>$, R X
if we read $<$, L X

eg: $a^n b^n c^n \rightarrow$ equivalent LBA.

Q construct LBA for $L = \{a^n b^n c^n \mid n > 0\}$



eg. $x = x+3$.

$x=3$, $x=100$.

↓ ↓

$x=6$ $x=103$

· for every i/p symb; we need to specify the memory size.

↳ Drawback ✓

* UNDECIDABILITY :

(direct 1M QS)

eg: Grammar is ambig./not.

↳ undecidable.

→ (It is possible to identify, but no given procedure) ✓

· NFA ⟶ DFA : Decidable.

Problem 1 : Finding whether the grammar is ambiguous/ not.

2. equiv. of 2 FAs : decidable

· equiv. of 2 RLS : decidable.

3. 2 CFLS/ CFGS are equivalent/ not :

↳ Undecidable.

↓

PDA₁ PDA₂

cant comp. 2 PDAS.

→· q_0, a P_0, a

⇒ in both m/c, a & b path is available in DFA

but, in PDA ⟶ no such Rule.

– no guarantee of the symbol.

q_0 : $a, z_0 / az_0$

← b path not available.

4. Halting problem of TM.

↳ undecidable.

– there is no specific procedure.

5. Post correspondence Problem.

↳ Undecidable.

$\rightarrow X_1 = (w_1, w_2, w_3 \cdots\cdots w_n)$

$X_2 = (z_1, z_2, z_3 \cdots\cdots z_n)$

Count of strings is same.

· We need to find String matching.

$$\begin{array}{l} \quad\ w_1 \quad w_2 \quad w_3 \\ X_1 = (ab, ab, c) \\ X_2 = (ab, a, a) \\ \quad\ x_1 \quad x_2 \quad x_3 \end{array}$$

$w_1 w_2 w_3$

eg: $X = (a, ab, c)$ $2 \rightarrow ab$

$Y = (ab, ab, a)$ $2 \rightarrow ab$

∴ PCP has soln.

eg: $w = (ab, a, aa)$ (1 2 3)

$X = (b, aa, a)$ (1 2 3)

21 → aab

21 → aab

(21 & 32).

∴ PCP has a solution.

Q Verify ~~the~~ whether the following PCP problems has solution / not.

(i) $w = (0, 100, 110)$

$X = (01, 001, 10$

(ii) $w = (0, 01)$

$X = (100, 001)$

(iii) $w = (b, a, ca, abc)$ (1 2 3 4)

$X = (ca, ab, a, c)$ (1 2 3 4)

(i) 13 : 0110,

0110 ✓

123 : 0100110

0100110 ✓

∴ Solution ✓

(ii) ~~w=~~ no solution

(iii) ~~no solution~~.

2 1 3 2 4

a b c a a a b c

a b c a a a b c

{2, 1, 3, 2, 4} → solution

* Properties

1. Union of 2 RLS is Regular.

Closure Properties of RL :

(i) Union :

$L_1 \cup L_2$

$0(00)^* \; (00)^* = 0^*$ (RL).

⇒ Decidable.

(ii) Concatenation.

- RL closed under concat, intersection, complement, reverse, Kleen closure, difference.
- [Not closed under RL].
  - subset operation.

(i) Union.

$\boxed{RL \cup RL = RL}$

eg: $0.(00)^* \cup (00)^* = 0^*$

(ii) Concat : $\boxed{RL \cdot RL = RL}$

$L_1 = \{ \varepsilon, 00, 0000 \cdots \}$

$L_2 = \{ 0, 000, 00000 \cdots \}$

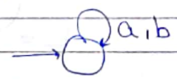
$L_1 \cdot L_2 = \{ 0, 000 \cdots ; 000, 00000 \cdots \cdots ; \cdots \cdots \}$.

⇒ L_2 (odd no. of zeros).

$L_1 \cdot L_2 = 0.(00)^*$

(iii) Intersection : $\boxed{RL \cap RL = RL}$

$L_1 \cap L_2 = \phi$ ← regular.



(iv) complement :

RL

$\boxed{\text{if } L_1 = RL \quad \overline{L_1} = RL}$

$L_1 = \{(00)^*\}$

$\overline{L_1} = 0.(00)^*$

(v) Kleen closure

$(L_1)^* = \{\epsilon, 0000, 08, \dots\}$

$(L_1)^* = L_1$

$\boxed{\text{if, } L_1 = RL \quad L_1^* = RL}$

(vi) Reverse

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

$L_R = \{b^n a^m \mid m, n > 0\}$.

(vii) Subset operation

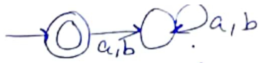
$(a+b)^* \longrightarrow RE$.

$L = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb \dots\}$

$\{ab, aabb, aaabbb, aaaabbbb \dots\}$

$L_1 = \{a^n b^n \mid n > 0\}$.

is L_1 regular?



$L = \epsilon$.

$(a+b)^* \longrightarrow RL$

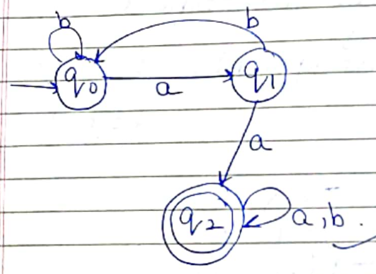
Subset

$(a^n b^n \mid n \geq 0) \longrightarrow$ not RL [CFL]

$\therefore$ Not closed.

* Decision Properties of RL:

1. The L is finite / not?

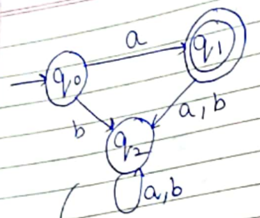


$RL = (b + a \cdot b)^* \cdot aa(a+b)^*$

if $\because$ loops $\longrightarrow$ infinite.

$L = \{a\}$

$\because$ finite

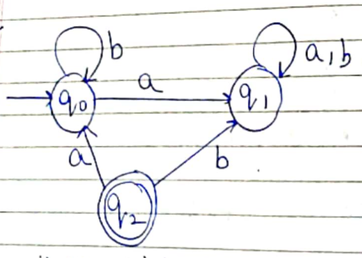
* 

$\because$ unreachable state has loop.

2. emptiness/not.
   - emptySet/not.
   - ϵ also not accepted.

$\{\}$ (no string is accepted). $\{\} \to \phi$

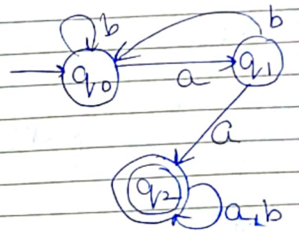
$\{\epsilon\} \to \epsilon$

eg. 

- if the final state is unreachable from the init. state.

3. Membership. → decidable.

- is the string the member of the FA/not?



a : member.

Q. if we read aa i/p string how many states are member?
   - $\{q_0, q_1, q_2\}$

- CFL is finite/infinite.

$A \to aA/a$ → recursive.

- empty/non empty.

- $A \to aB$
  $B \to aB$ ~~X root std~~
  - no string is generated.

· CFL not closed under:
- intersection
- complement

e.g.: $L_1 = \{a^i b^i c^k \mid i, k > 0\}$
$L_2 = \{a^i b^k c^k \mid i, k > 0\}$ } CFL.

$L_1 = \{abc, aabbcc, aabbcc\}$

$L_2 = \{abc, abbcc, aabbcc\}$.

intersection

$L_1 \cap L_2 = \{abc, aabbcc, aaabbbccc \ldots\}$

$\Downarrow$

$\{a^n b^n c^n \mid n > 0\} \rightarrow$ CSL

∴ Not closed.

∴ Complement.

Assume $L_1 =$ CFL

$\rightarrow \overline{L_1} =$ CFL

$\overline{\overline{L_1} \cup \overline{L_2}} \rightarrow$ CFL

$\Rightarrow \overline{\overline{L_1}} \cap \overline{\overline{L_2}} = L_1 \cap L_2$ ∴ not CFL.

∴ complement → CFL ✗.

Q Which of the following statement is false?

(a) The halting problem of TM is undecidable.

(b) Determining whether a CFG is ambig. is undecidable.

(c) Given 2 CFGs → G_1 & G_2 it is undecidable whether $L(G_1) = L(G_2)$

(d) Given 2 RGs → G_1 & G_2, it is undecidable whether $L(G_1) = L(G_2)$.

(e) PDA can't be compared.

Q which of the following problem is undecidable?

(a) membership problem for CFGs.
(b) ambiguity problem for CFGs.
(c) Finiteness problem for FAs.
(d) Equivalence problem for FAs.

Q which of the following statements are decidable?

S1: The intersection of 2 RLs is infinite.

S2: whether 2 PDAs accept the same language.

S3: whether the given grammar is context free.

(a) 1 & 2.
(b) 1 & 3
(c) 2 & 3
(d) None of the above.

Soln.

S1: intersection always decidable

$a^* \cup a\cdot(aa)^* = a(aa)^*$.

decidable. infinite.

S2: 2 PDAs can't be compared.

S3: → decidable

$1NT \longrightarrow (V \cup T)^*$.

∴ (b)

Q which of the following statement is false?

S1: DFA & NFA has the same power.

S2: DPDA & NPDA has the same power.

S3: Mealy & Moore m/c has the same power.

(a) 1, 2, 3
(b) 2
(c) 1, 2
(d) None of the above.

power: same set of strings.

→ DFA ≅ NFA (DFA more efficient).
↳ same power. (S1)

→ S2: NPDA > DPDA. ✗.

$L_D \subset L_N$.

↳ w.r.t. can be accepted.

→ interconvertible.

(S3) Melay & Moore same power.

→ is CFG empty/ finite → decidable.

---

1. FA & RE → [Most imp]
2. CFG → (1 Q)
3. PDA & CFG
4. Chomsky Hierarchy → Direct Qs.

TM → Halting problem.

—X—X—X—X—X