Hall Ticket Number:

III/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

December, 2024 — Common to CB, DS, CS & IT

Fifth Semester — Automata Theory & Formal Languages

Time: Three Hours — Maximum: 70 Marks

Answer question 1 compulsorily. (14X1 = 14 Marks)

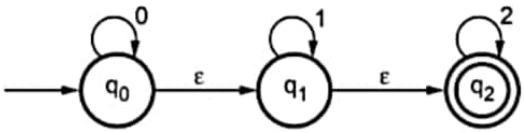
Answer one question from each unit. (4X14 = 56 Marks)

			CO	BL	M
1	a)	What is an alphabet in automata theory?	CO 1	L1	1M
	b)	List any two differences between a DFA and an NFA?	CO 1	L2	1M
	c)	Define transition function with an Example	CO 1	L2	1M
	d)	What is an epsilon ε transition in an NFA?	CO 1	L2	1M
	e)	What is a regular expression? Give an example.	CO 2	L2	1M
	f)	State the Pumping Lemma for regular languages.	CO 2	L1	1M
	g)	What are the closure properties of regular languages under intersection?	CO 2	L2	1M
	h)	What is a context-free language? Give an example.	CO 3	L2	1M
	i)	Define Chomsky Normal Form (CNF) for context-free grammars.	CO 3	L1	1M
	j)	Define a Pushdown Automaton (PDA) with respect to its components.	CO 3	L1	1M
	k)	List the decision properties of context-free languages.	CO 4	L1	1M
	l)	What is the definition of a Turing machine's tape?	CO 4	L2	1M
	m)	What is a language that is not recursively enumerable? Provide an example.	CO 4	L2	1M
	n)	Define an undecidable problem with respect to Turing machines.	CO 4	L1	1M

Unit-I

2 a) Construct a DFA that accepts all strings on {0,1} except those containing the substring 101. — CO1 L3 7M

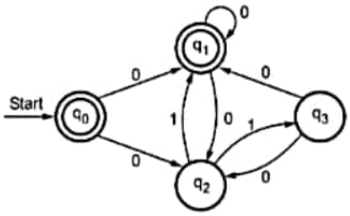
b) Convert the given NFA with ε to NFA without ε — CO1 L3 7M



(OR)

3 a) Design a DFA which accepts Even number of 0's and even number of 1's — CO1 L3 7M

b) Convert the following NFA to DFA — CO1 L3 7M



Unit-II

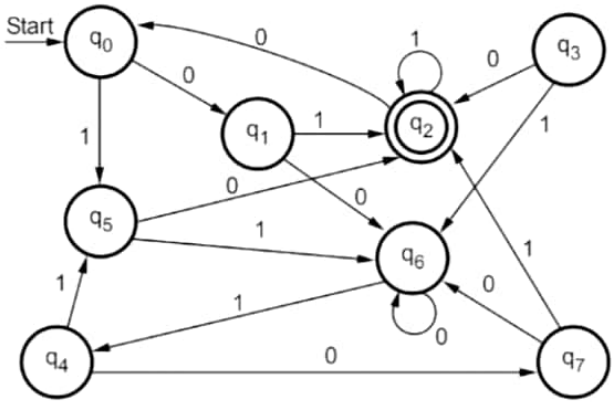
4 a) Write the regular expressions for the following languages — CO2 L2 6M

i) regular expression which contains L having strings which should have at least one 0 and at least one 1

ii) regular expression to denote the language L over {a, b} such that all the strings do not contain the substring "ab"

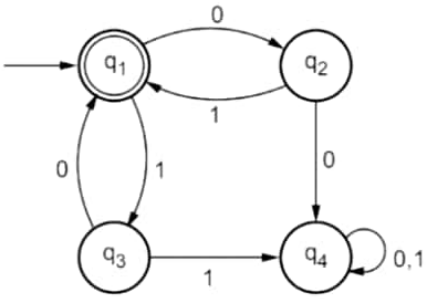
Scanned with OKEN Scanner

	b)	Construct a Minimum state DFA for the following Automata	CO2	L3	8M



(OR)

5	a)	Construct a Regular Expression for the given DFA using state elimination method.	CO2	L3	7M



	b)	Which of the following languages is regular? Justify 1. $L = \{a^n b^m \mid n, m \geq 1\}$ 2. $L = \{a^n b^n \mid n \geq 1\}$	CO2	L3	7M

Unit-III

6	a)	Consider the following productions S → aB | bA A → aS | bAA | a B → bS | aBB | b For the string aaabbabbba, find the left most derivation	CO3	L3	7M
	b)	Simplify the following grammar S → AACD A → aAb|ε C → aC|a D → aDa|bDb|ε	CO3	L3	7M

(OR)

7	a)	Construct a PDA for the Language $L = \{a^n b^{2n} \mid n \geq 1\}$	CO3	L3	7M
	b)	Define Pumping Lemma, Prove that $L = \{a^m b^{m+1} c^{m+2} \mid m>0\}$ is not a CFL	CO3	L3	7M

)

Unit-IV

8	a)	Explain closer properties of context free language.	CO4	L2	7M
	b)	Construct a Turing Machine for $L = \{a^n b^n c^n \mid n \geq 0\}$	CO4	L3	7M

(OR)

9	a)	What is Post Correspondence problem? Does PCP with two lists $x = (b, bab^3, ba)$ and $y = (b^3, ba, a)$ have a solution?	CO4	L3	7M
	b)	Explain about programming techniques for Turing Machine.	CO4	L2	7M

Scanned with OKEN Scanner

1a) what is an alphabet in automata theory?

A finite non-empty set of symbols. It is denoted by 'Σ'

1b) list any two Differences between a DFA and an NFA?

DFA	NFA
1) $\delta : Q \times \Sigma \rightarrow Q$	1) $\delta : Q \times \Sigma \rightarrow 2^Q$
2) Dead states occur	2) No dead states
3) Complement is possible	3) Complement is not possible
4) High efficiency	4) low efficiency

1c) Define transition function with an example

A function that determines the next state of an automata, given the current state and input symbol

Ex: $\delta(q_0, 0) = q_1$

1d) what is an epsilon ε transition in an NFA?

an NFA that allows an automata to change its state without consuming an input symbol

1e) what is a regular expression? Give an example

Regular language can be Represented by a simple expression format is known as Regular expression. Ex: $a^* b^* c^*$

Scanned with OKEN Scanner

1f) State the pumping lemma for regular languages.

Let L be a Regular language, there exist a positive Constant n, for every string of w in L, $|w| >= n$

1) we break string into three parts $w = xyz$

a) $y \neq \varepsilon$

b) $|xy| <= n$

c) $xy^k z \ \forall k$

if we pump y for any no. of k values then Result String must be Regular language.

1g) what are the closure properties of regular languages under intersection?

if L_1 and L_2 are two regular languages then $L_1 \cap L_2$ is also regular languages

1h) what is a Context free language? Give an Example

If the strings are generated by Context free grammar, then it is called Context free language. Ex $L = \{a^n b^n / n \geq 1\}$

1i) Define Chomsky Normal form (CNF) for Context-free grammars

A Grammar G is Said to be in chomsky normal form if the productions are of the form $A \rightarrow BB$ (or) $A \rightarrow \alpha$

where G is a simplified Context free grammar i.e. ε-productions, unit Productions and useless Symbols are Eliminated.

Scanned with OKEN Scanner

1j) Define a pushdown Automaton (PDA) with respect to its Components

Combination of 7 tuples

$$M = (Q, \Sigma, \Gamma, Z_0, \delta, q_0, F)$$

where Q is finite set of states

Σ is finite set of i/p alphabets

Γ is finite set of stack symbols

Z_0 is initial stack symbol

δ is transition function

q_0 is initial state

F is final state

$\delta : Q \times \{\Sigma \cup \varepsilon\} \times \Gamma \longrightarrow Q \times \Gamma^*$ for DPDA

$\delta : Q \times \{\Sigma \cup \varepsilon\} \times \Gamma \longrightarrow 2^Q \times \Gamma^*$ for NPDA

1k) list the decision properties of Context-free languages

1) empty / non-empty

2) finite / infinite

3) membership (CYK-algorithm)

1l) what is the definition of a Turing Machine's Tape?

A semi infinite strip of tape that is divided into cells each containing a single symbol from the tape alphabet.

Scanned with OKEN Scanner

1m) what is a language that is not recursively enumerable? Provide an example

Diagonalization language

1n) Define an undecidable problem with respect to turing machines

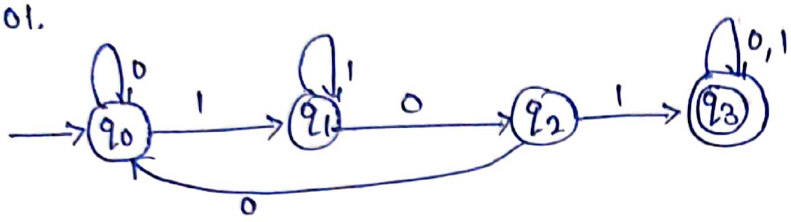
If the strings are generated by Recursive Enumerable language, then the problems are called undecidable problem

Scanned with OKEN Scanner

Unit - I

2(a) Construct a DFA that accepts all strings on {0,1} except those containing the substring 101.

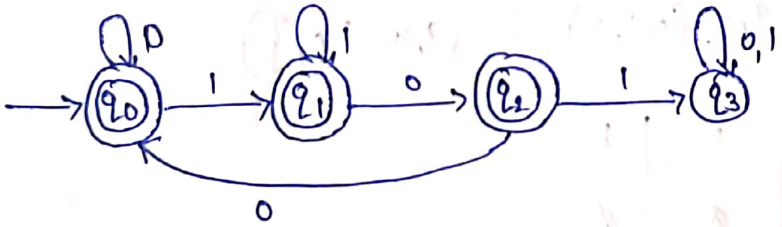
let us construct machine that accepts strings along with substring 101.



Now let us construct DFA that accepts all strings on {0,1} except those containing the substring 101.

→ Here simply perform complement to the above machine

1) Transition graph:



2) $M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, \{q_0\}, \{q_0, q_1, q_2\})$.

$\delta(q_0, 0) = q_0$ $\quad$ $\delta(q_1, 0) = q_2$

$\delta(q_0, 1) = q_1$ $\quad$ $\delta(q_1, 1) = q_1$

$\delta(q_2, 0) = q_0$ $\quad$ $\delta(q_3, 0) = q_3$

$\delta(q_2, 1) = q_3$ $\quad$ $\delta(q_3, 1) = q_3$

Scanned with OKEN Scanner

3) Transition Table:

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_3
q_3	q_3	q_3

2(b) Convert the given NFA with ε to NFA without ε

Conversion from ε-NFA to NFA:

Step1: Transitions for given ε-NFA

$M = (\{q_0, q_1, q_2\}, \{0, 1, 2\}, \delta, \{q_0\}, \{q_2\})$

δ	0	1	2
$\rightarrow q_0$	q_0	ϕ	ϕ
q_1	ϕ	q_1	ϕ
q_2	ϕ	ϕ	q_2

Step2: Transition Table for NFA

δ'	0	1	2
$* \rightarrow [q_0]$	$[q_0 q_1 q_2]$	$[q_1 q_2]$	$[q_2]$
$* [q_1]$	ϕ	$[q_1 q_2]$	$[q_2]$
$* [q_2]$	ϕ	ϕ	$[q_2]$

(∵ * represents final state)

(∵ you can also represent final state by circle) Ex: q_2

Scanned with OKEN Scanner

⇒ In NFA, the states will be same as of ε-NFA

⇒ initial state will be same as of ε-NFA

⇒ final states will be those, which have ε-transitions to the final state in ε-NFA

⇒ Here we use the transition function of epsilon NFA.

$\hat{\delta}([q_0],0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_0,\varepsilon),0))$

$= \varepsilon\text{-closure}(\delta(\{q_0,q_1,q_2\},0))$

$= \varepsilon\text{-closure}(\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0))$

$= \varepsilon\text{-closure}(q_0 \cup \phi \cup \phi)$

$= \{q_1 q_2 q_3\}$

$\therefore \hat{\delta}(q_0,\varepsilon) = \varepsilon\text{-closure}(q_0)$

$= \{q_0, q_1, q_2\}$

$\therefore \hat{\delta}(q_1,\varepsilon) = \varepsilon\text{-closure}(q_1)$

$= \{q_1 q_2\}$

$\therefore \hat{\delta}(q_2,\varepsilon) = \varepsilon\text{-closure}(q_2)$

$= \{q_2\}$

$\hat{\delta}([q_0],1) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_0,\varepsilon),1))$

$= \varepsilon\text{-closure}(\delta(\{q_0,q_1,q_2\},1))$

$= \varepsilon\text{-closure}(\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1))$

$= \varepsilon\text{-closure}(\phi \cup q_1 \cup \phi)$

$= \{q_1 q_2\}$

$\hat{\delta}([q_0],2) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_0,\varepsilon),2))$

$= \varepsilon\text{-closure}(\delta(\{q_0,q_1,q_2\},2))$

$= \varepsilon\text{-closure}(\delta(q_0,2) \cup \delta(q_1,2) \cup \delta(q_2,2))$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup q_2)$

$= \{q_2\}$

Scanned with OKEN Scanner

$$\hat{\delta}([q_1],0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_1,\varepsilon),0))$$
$$= \varepsilon\text{-closure}(\delta(\{q_1,q_2\},0))$$
$$= \varepsilon\text{-closure}(\delta(q_1,0) \cup \delta(q_2,0))$$
$$= \varepsilon\text{-closure}(\phi \cup \phi)$$
$$= \phi$$

$$\hat{\delta}([q_1],1) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_1,\varepsilon),1))$$
$$= \varepsilon\text{-closure}(\delta(\{q_1,q_2\},1))$$
$$= \varepsilon\text{-closure}(\delta(q_1,1) \cup \delta(q_2,1))$$
$$= \varepsilon\text{-closure}(q_1 \cup \phi)$$
$$= \{q_1,q_2\}$$

$$\hat{\delta}([q_2],2) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_2,\varepsilon),2))$$
$$= \varepsilon\text{-closure}(\delta(\{q_2\},2))$$
$$= \varepsilon\text{-closure}(\delta(q_2,2))$$
$$= \varepsilon\text{-closure}(q_2)$$
$$= \{q_2\}$$

$$\hat{\delta}([q_1],2) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_1,\varepsilon),2))$$
$$= \varepsilon\text{-closure}(\delta(\{q_1,q_2\},2))$$
$$= \varepsilon\text{-closure}(\delta(q_1,2) \cup \delta(q_2,2))$$
$$= \varepsilon\text{-closure}(\phi \cup q_2)$$
$$= \{q_2\}$$

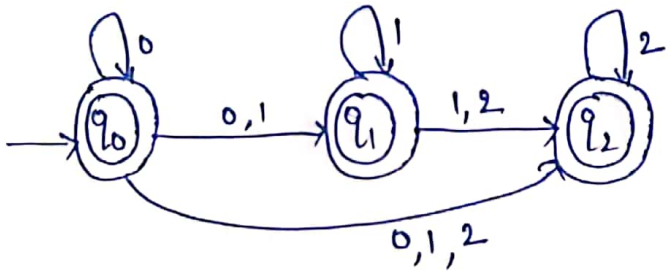
Scanned with OKEN Scanner

$$\hat{\delta}([q_2],0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_2,\varepsilon),0))$$
$$= \varepsilon\text{-closure}(\delta(\{q_2\},0))$$
$$= \varepsilon\text{-closure}(\phi)$$
$$= \phi$$

$$\hat{\delta}([q_2],1) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_2,\varepsilon),1))$$
$$= \varepsilon\text{-closure}(\delta(\{q_2\},1))$$
$$= \varepsilon\text{-closure}(\phi)$$
$$= \phi$$

Step 3:

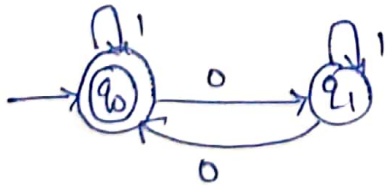
Transition graph: of NFA



3(a) Design a DFA which accepts Even number of 0's and even number of 1's

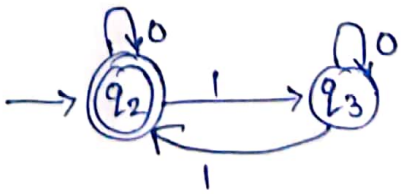
even number of 0's:

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



even number of 1's:

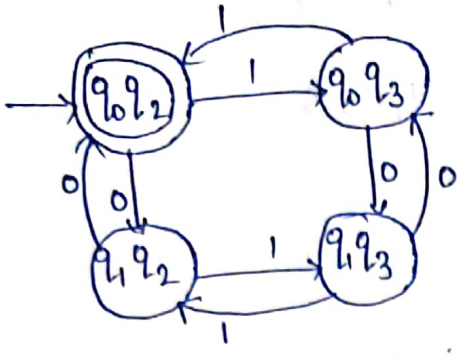
$L = \{\varepsilon, 11, 1111, \ldots\}$



Cartesian product

$$\{q_0, q_1\} \times \{q_2, q_3\} = \{(q_0,q_2); (q_0,q_3); (q_1,q_2); (q_1,q_3)\}$$

Scanned with OKEN Scanner



$\delta(q_{03}, 0) = q_{13}$

$\delta(q_0q_3, 1) = q_0q_2$

$\delta(q_1q_2, 0) = q_0q_2$

$\delta(q_1q_2, 1) = q_1q_3$

$\delta(q_0q_2, 0) = q_1q_2$

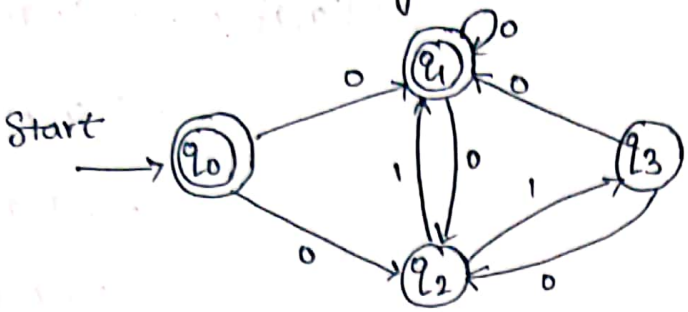
$\delta(q_0q_2, 1) = q_0q_3$

$\delta(q_1q_3, 0) = q_0q_3$

$\delta(q_1q_3, 1) = q_1q_2$

Scanned with OKEN Scanner

3(b) Convert the following NFA to DFA



Conversion from NFA to DFA:

Step1: Transition Table for NFA

δ	0	1
$\rightarrow q_0$	q_1q_2	ϕ
q_1	q_1q_2	ϕ
q_2	ϕ	q_1q_3
q_3	q_1q_2	ϕ

$M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, \{q_0\}, \{q_0, q_1\})$

Step2: Transition Table for DFA

δ'	0	1
$\rightarrow q_0$	q_1q_2	ϕ
q_1q_2	q_1q_2	q_1q_3
q_1q_3	q_1q_2	ϕ
ϕ	ϕ	ϕ

⇒ Initial state will be same as of NFA

⇒ final state: if the final state of NFA is a subset or complete state in DFA, then that will be final state

Scanned with OKEN Scanner

$\hat{\delta}([q_0],0) = \delta(\{q_0\},0)$

$= \{q_1,q_2\}$

$\hat{\delta}(q_0,1) = \delta(\{q_0\},1)$

$= \phi$

$\hat{\delta}([q_1q_2],0) = \delta(\{q_1,q_2\},0)$

$= \delta(q_1,0) \cup \delta(q_2,0)$

$= \{q_1,q_2\} \cup \{\phi\}$

$= \{q_1,q_2\}$

$\hat{\delta}([q_1,q_2],1) = \delta(\{q_1,q_2\},1)$

$= \delta(q_1,1) \cup \delta(q_2,1)$

$= \phi \cup \{q_1,q_3\}$

$= \{q_1,q_3\}$

$\hat{\delta}([q_1q_3],0) = \delta(\{q_1,q_3\},0)$

$= \delta(q_1,0) \cup \delta(q_3,0)$

$= \{q_1,q_2\} \cup \{q_1,q_2\}$

$= \{q_1,q_2\}$

$\hat{\delta}([q_1q_3],1) = \delta(\{q_1,q_3\},1)$

$= \delta(q_1,1) \cup \delta(q_3,1)$

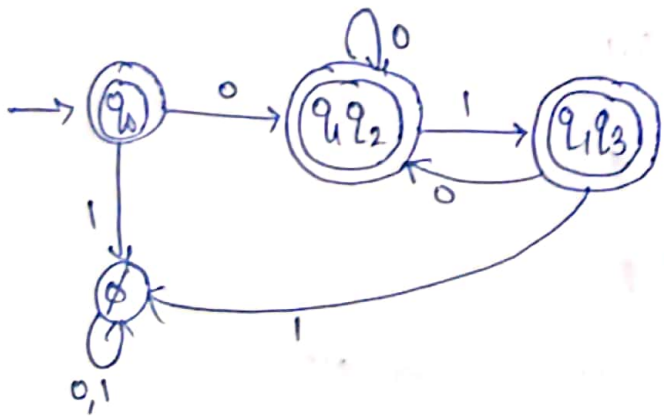
$= \phi \cup \phi$

$= \phi$

Step 3:

machine $M = (\{q_0, q_1q_2, q_1q_3\}, \{0,1\}, \delta, \{q_0\}, \{q_0, q_1q_2, q_1q_3\})$

Transition graph:



Unit - II

4(a) write the regular expressions for the following languages

(i) regular expression which contains L having Strings which Should have at least one 0 and at least one 1

$0\underline{\times}1 \quad 1\underline{\times}0 \quad 1\underline{\times}1 \quad 0\underline{\times}0$

$\downarrow \quad \downarrow$

$\phi \quad \phi$

$R.E = \{0(0+1)^*1 + 1(1+0)^*0 + (0+1)^*0(0+1)^*1 + (0+1)^*1(0+1)^*0\}$

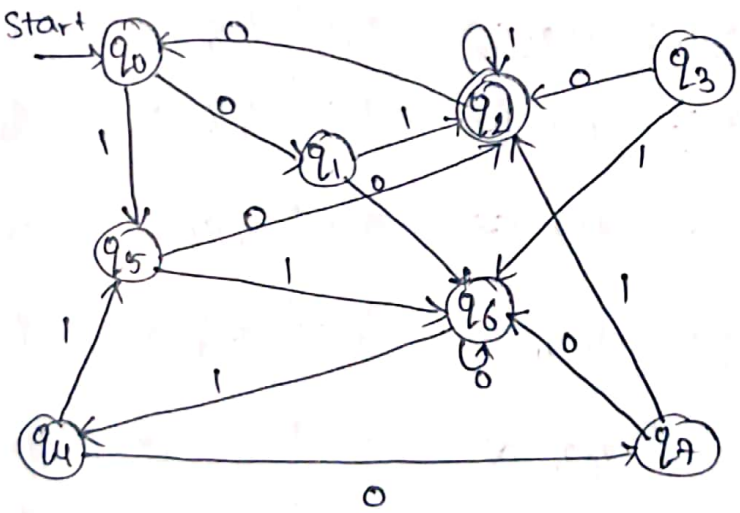
4(a) (ii) regular expression to denote the language L over {a, b} Such that all the Strings do not Contain the SubString "ab"

$\Rightarrow$ Any one of a's should not be followed by any of b's

$\Rightarrow$ But Any no. of b's Can be followed by any no. of a's

$\therefore R.E = \{b^* a^*\}$

4(b) Construct a Minimum state DFA for the following Automata.



Scanned with OKEN Scanner

Tabular method:

	q_0	q_1	q_2	q_3	q_4	q_5	q_6
q_1	×						
q_2	×	×					
q_3	×	×	×				
q_4	✓	×	×	×			
q_5	×	×	×	✓	×		
q_6	×	×	×	×	×	×	
q_7	×	✓	×	×	×	×	×

$q_0q_7 \xrightarrow{0} q_1q_6$ (×)
$q_0q_7 \xrightarrow{1} q_5q_2$ (×)

$q_0q_2 \xrightarrow{0} q_1q_0$ (×)
$q_0q_2 \xrightarrow{1} q_5q_2$ (×)

$q_1q_4 \xrightarrow{0} q_6q_7$ (×)
$q_1q_4 \xrightarrow{1} q_2q_5$ (×)

$q_2q_5 \xrightarrow{0} q_0q_2$ (×)
$q_2q_5 \xrightarrow{1} q_2q_6$ (×)

$q_0q_6 \xrightarrow{0} q_1q_6$ (×)
$q_0q_6 \xrightarrow{1} q_5q_4$ (×)

$q_0q_1 \xrightarrow{0} q_1q_6$ (×)
$q_0q_1 \xrightarrow{1} q_5q_2$ (×)

$q_1q_3 \xrightarrow{0} q_6q_2$ (×)
$q_1q_3 \xrightarrow{1} q_2q_6$ (×)

$q_2q_4 \xrightarrow{0} q_0q_7$ (×)
$q_2q_4 \xrightarrow{1} q_2q_5$ (×)

$q_0q_5 \xrightarrow{0} q_1q_2$ (×)
$q_0q_5 \xrightarrow{1} q_5q_6$ (×)

$q_1q_7 \xrightarrow{0} q_6q_6$ (✓)
$q_1q_7 \xrightarrow{1} q_2q_2$ (✓)

$q_1q_2 \xrightarrow{0} q_6q_0$ (×)
$q_1q_2 \xrightarrow{1} q_2q_2$ (×)

$q_2q_3 \xrightarrow{0} q_0q_2$ (×)
$q_2q_3 \xrightarrow{1} q_2q_6$ (×)

$q_0q_4 \xrightarrow{0} q_1q_7$ (✓)
$q_0q_4 \xrightarrow{1} q_5q_5$ (✓)

$q_1q_6 \xrightarrow{0} q_6q_6$ (×)
$q_1q_6 \xrightarrow{1} q_2q_4$ (×)

$q_2q_7 \xrightarrow{0} q_0q_6$ (×)
$q_2q_7 \xrightarrow{1} q_2q_2$ (×)

$q_3q_7 \xrightarrow{0} q_2q_6$ (×)
$q_3q_7 \xrightarrow{1} q_6q_2$ (×)

$q_0q_3 \xrightarrow{0} q_1q_2$ (×)
$q_0q_3 \xrightarrow{1} q_5q_6$ (×)

$q_1q_5 \xrightarrow{0} q_6q_2$ (×)
$q_1q_5 \xrightarrow{1} q_2q_6$ (×)

$q_2q_6 \xrightarrow{0} q_0q_6$ (×)
$q_2q_6 \xrightarrow{1} q_2q_4$ (×)

$q_3q_6 \xrightarrow{0} q_2q_6$ (×)
$q_3q_6 \xrightarrow{1} q_6q_4$ (×)

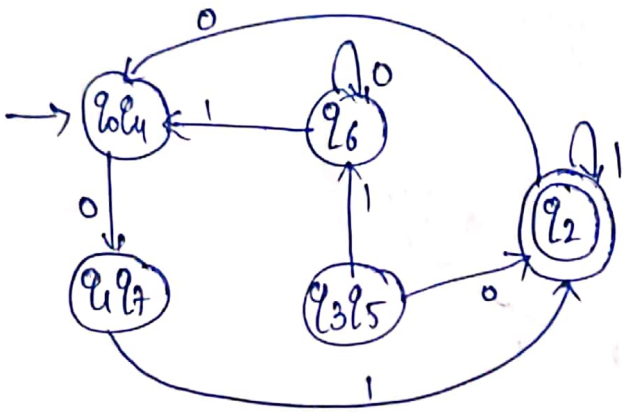
Scanned with OKEN Scanner

$q_3q_5 \xrightarrow{0} q_2q_2$ (✓) $\quad q_4q_7 \xrightarrow{0} q_7q_6$ (✗) $\quad q_4q_5 \xrightarrow{0} q_2q_2$ (✗) $\quad q_5q_6 \xrightarrow{0} q_2q_6$ (✗)

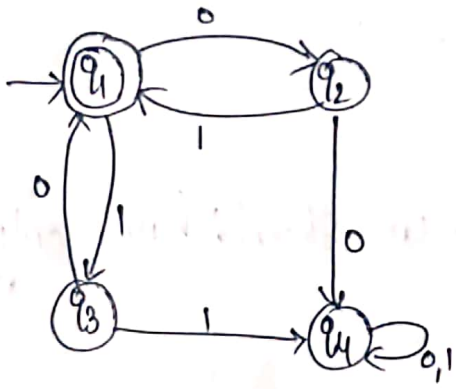
$q_3q_5 \xrightarrow{1} q_6q_6$ (✓) $\quad q_4q_7 \xrightarrow{1} q_5q_2$ (✗) $\quad q_4q_5 \xrightarrow{1} q_5q_6$ (✗) $\quad q_5q_6 \xrightarrow{1} q_6q_4$ (✗)

$q_3q_4 \xrightarrow{0} q_2q_7$ (✗) $\quad q_4q_6 \xrightarrow{0} q_7q_6$ (✗) $\quad q_5q_7 \xrightarrow{0} q_2q_6$ (✗). $\quad q_6q_7 \xrightarrow{0} q_6q_6$ (✗)

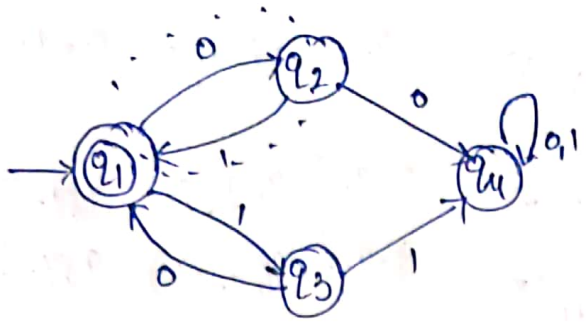
$q_3q_4 \xrightarrow{1} q_6q_5$ (✗) $\quad q_4q_6 \xrightarrow{1} q_5q_4$ (✗) $\quad q_5q_7 \xrightarrow{1} q_6q_2$ (✗) $\quad q_6q_7 \xrightarrow{1} q_4q_2$ (✗)



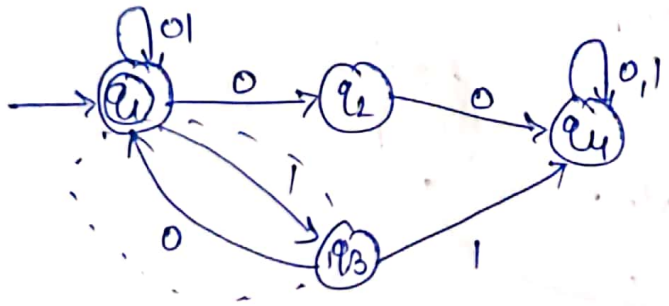
5(a) Construct a Regular Expression for the given DFA using State Elimination method.



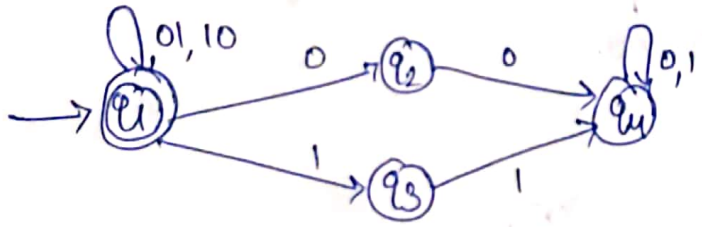
Scanned with OKEN Scanner



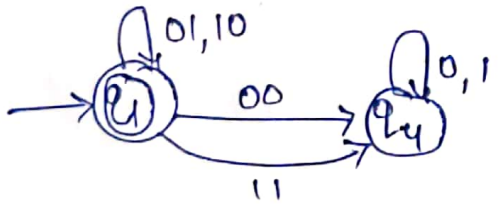
Step 1: Removing the 1st cycle

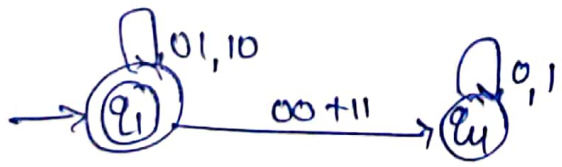


Step2: Removing the 2nd cycle



Step3: In the final DFA, we should have only initial state and final state





Scanned with OKEN Scanner

Step 4: Removal of dead states

$\therefore$ Regular Expression $= (01+10)^*$

5(b) which of the following languages is regular? justify

1) $L = \{a^n b^m / n, m \geq 1\}$ 2) $L = \{a^n b^n / n \geq 1\}$

Pumping Lemma:

Let 'L' is a Regular language, there exist +ve Constant n such that for every string of w in L, $|w| \geq n$

1) we break the string into 3 parts $w = xyz$

where a) $y \neq \varepsilon$

b) $|xy| <= n$

c) $xy^k z$, $\forall k$

for all values of k, if we pump y for any no. of k values then result string must be Regular language

Scanned with OKEN Scanner

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

the language will be $L = \{ab, aab, abb, \ldots\}$

$\boxed{\text{here the regular Expression is } a^+ b^+}$

1) let us consider a String such that $|w| \geq n$

$|n+m| >= n+m$

2) Break the String into three parts $w = xyz$

$a^n b^m$

$a^{n-1} a b^m$

$x \quad y \quad z$

$\therefore y \neq \varepsilon$

$|xy| <= m+n$

3) $\Rightarrow x y^k z$

$a^{n-1} a^k b^m$

$\underline{\underline{K=1}}$: $a^{n-1} a b^m$ (True)

$\underline{\underline{K=2}}$: $a^{n-1} a^2 b^m$

for $n=1$ & $m=1$ $\quad a^0 a^2 b^2$ (True)

for $n=2$ & $m=1$ $\quad a^1 a^2 b^1$ (True)

$\underline{\underline{K=3}}$: $a^{n-1} a^3 b^m$

for $n=1$ & $m=1$ $\quad a^0 a^3 b^1$ (True)

for $n=2$ & $m=1$ $\quad a^1 a^3 b^1$ (True)

Scanned with OKEN Scanner

$\therefore$ The given language is Regular language.

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

The language will be $L = \{ab, aabb, aaabbb, ..\}$

1) let us consider a String such that $|w| \geq n$

$|n+n| \geq n$

$|2n| \geq n$

2) Break the String into three parts $w = xyz$

$a^n b^n$

$a^{n-1} a b^n$

$x \quad y \quad z$

3) $a^{n-1} a^k b^n$

$\underline{\underline{K=1}}$: $a^{n-1} a b^n$ (True)

$\underline{\underline{K=2}}$: $a^{n-1} a^2 b^n$

$a^{n+1} b^n$ (false)

$\therefore$ given language is not Regular language.

Scanned with OKEN Scanner

6(a) Consider the following productions

$S \rightarrow aB \mid bA$

$A \rightarrow aS \mid bAA \mid a$

$B \rightarrow bS \mid aBB \mid b$

for the String aaabbabbba, find the left most derivation.

Given grammar,

$S \rightarrow aB \mid bA$

$A \rightarrow aS \mid bAA \mid a$

$B \rightarrow bS \mid aBB \mid b$

String: aaabbabbba

left most derivation: $S \; (S \rightarrow aB)$

$aB \; (B \rightarrow aBB)$

$aaBB \; (B \rightarrow aBB)$

$aaaBBB \; (B \rightarrow b)$

$aaabBB \; (B \rightarrow bS)$

$aaabbSB \; (S \rightarrow aB)$

$aaabbaBB \; (B \rightarrow b)$

$aaabbabB \; (B \rightarrow bS)$

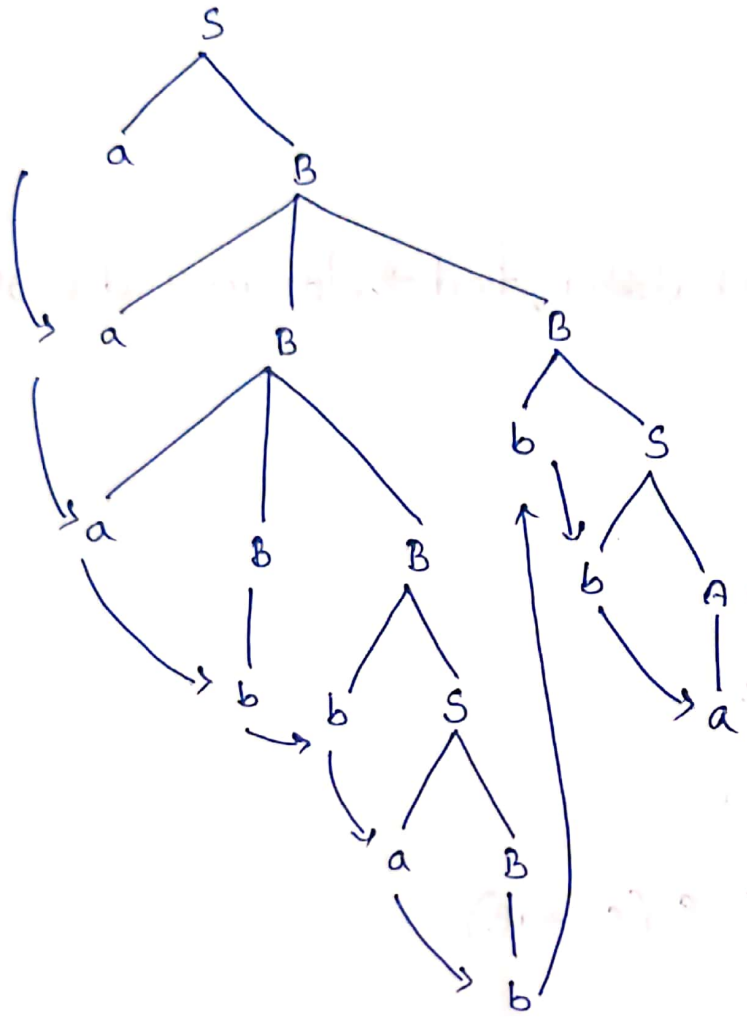
$aaabbabbS \; (S \rightarrow bA)$

$aaabbabbbA \; (A \rightarrow a)$

$\therefore$ aaabbabbba

Scanned with OKEN Scanner

Parse Tree for left most derivation:



Scanned with OKEN Scanner

6(b) Simplify the following grammar

$S \rightarrow AACD$

$A \rightarrow aAb / \varepsilon$

$C \rightarrow aC / a$

$D \rightarrow aDa / bDb / \varepsilon$

1) Eliminate ε-productions :

when $A = \varepsilon$

$S \rightarrow AACD / CD / ACD$

$A \rightarrow aAb / ab$

$C \rightarrow aC / a$

$D \rightarrow aDa / bDb / \varepsilon$

when $D = \varepsilon$

$S \rightarrow AACD / CD / ACD / AAC / C / AC$

$A \rightarrow aAb / ab$

$C \rightarrow aC / a$

$D \rightarrow aDa / bDb / aa / bb$

2) Eliminate unit productions:

$S \rightarrow AACD / CD / ACD / AAC / C / AC$

$A \rightarrow aAb / ab$

$C \rightarrow ac / a$

$D \rightarrow aDa / bDb / aa / bb$

(i.e. In $S \rightarrow C$ C will be replaced with its value)

Scanned with OKEN Scanner

we call a production is unit production, when R.H.S of production have only one variable.

$S \rightarrow AACD / CD / ACD / AAC / aC / a / AC$

$A \rightarrow aAb / ab$

$C \rightarrow aC / a$

$D \rightarrow aDa / bDb / aa / bb$

3) Eliminate <u>useless</u> symbols:

(i) non-generating property: if any non-terminal doesn't generate atleast one string then it has to be eliminated.

∴ we don't have this property in above grammar.

(ii) unReachable states:

which means, not reachable from start symbol

∴ Simplified grammar is

$S \rightarrow AACD / CD / ACD / AAC / ac / a / AC$

$A \rightarrow aAb / ab$

$C \rightarrow aC / a$

$D \rightarrow aDa / bDb / aa / bb$

Scanned with OKEN Scanner

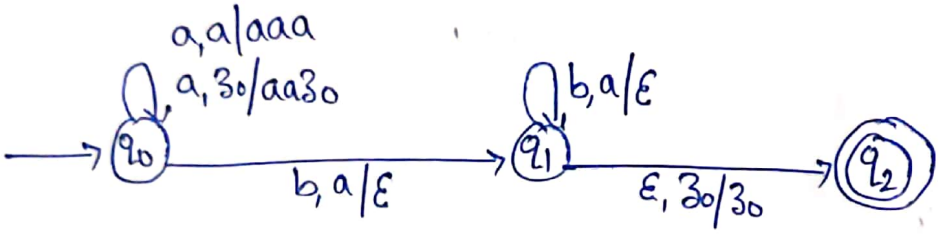
7(a) Construct a PDA for the Language $L=\{a^n b^{2n} \mid n>=1\}$

PDA Construction:

Process: given language $L=\{a^n b^{2n} / n>=1\}$

1) Here push two a's whenever we see one 'a' in input data.

2) we pop one 'a' for each one 'b'.



Here we use "accepted by final state" approach.

transitions:

$\delta(q_0,a,Z_0)=(q_0,aaZ_0)$ $\quad$ $\delta(q_1,b,a)=(q_1,\epsilon)$

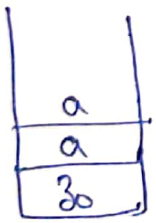
$\delta(q_0,a,a)=(q_0,aaa)$ $\quad$ $\delta(q_1,\epsilon,Z_0)=(q_2,Z_0)$

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

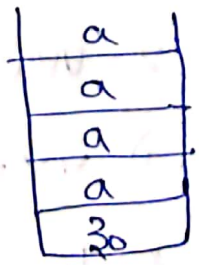
Example: let $n=2$ then $w=a^2b^4$

$=aabbbb$

1) a a b b b b
↑
push



2) a a b b b b
↑
push



Scanned with OKEN Scanner

3) a a b b b b
↑ pop

α
a
a
a
z_0

4) a a b b b b
↑ pop

α
a
a
z_0

5) a a b b b b
↑ pop

α
a
z_0

6) a a b b b b
↑ pop

α
z_0

7) a a b b b b
↑ skip

z_0

$\therefore w = a^2b^4$ is accepted.

7(b) Define pumping Lemma, prove that $L = \{a^m b^{m+1} c^{m+2} / m > 0\}$ is not a CFL

pumping lemma:

let L be a CFL, there exist a +ve Constant n, such that we choose a String z, $|z| >= n$ then we divide the String into 5 parts

$z = uvwxy$

$\therefore$ 1) $vx \neq \varepsilon$

2) $|vwx| <= n$

Scanned with OKEN Scanner

$\forall k$, $uv^k wx^k y$, if we pump v and x for every value of k, then the result String must be CFL.

given $L = \{a^m b^{m+1} c^{m+2} / m > 0\}$

$L = \{a^1 b^2 c^3, a^2 b^3 c^4, \ldots\}$

Let us Consider a String with $m = 2$

$w = a^2 b^3 c^4$

$= aabbbcccc$

divide String into 5 parts

$w = uvwxy$

$u = aab \quad v = b \quad w = \varepsilon \quad x = b \quad y = cccc$

Now $\forall k$, $w = uv^k wx^k y$

$\Rightarrow (aab)(b)^k \varepsilon (b)^k (cccc)$

Now for $k = 1$: $(aab)(b)^1 (b)^1 (cccc)$

$= aabbbcccc$ (True)

$k = 2$: $(aab)(b)^2 (b)^2 (cccc)$

$= aabbbbbcccc$ (false) $\therefore$

$\therefore$ It is not in CFL

$\therefore$ given $L = \{a^m b^{m+1} c^{m+2} / m > 0\}$ is not a CFL

Scanned with OKEN Scanner

Unit-IV

8(a) Explain closer properties of Context free language

closer properties of CFL:

1) Union
2) Concatenation
3) Kleen closure
4) Reverse
5) Homomorphism
6) Inverse homomorphism
7) Subtraction

1) Union:

let L_1 and L_2 are two CFL then $L_1 \cup L_2$ is also CFL

proof:

let $G_1 = (V_1, T_1, P_1, S_1)$ be the grammar that represented by the language L_1 $\quad L_1 = L(G_1)$

let $G_2 = (V_2, T_2, P_2, S_2)$ be the grammar that represented by the language L_2 $\quad L_2 = L(G_2)$

let us construct a CFG that accept the language $L_1 \cup L_2$

Let $G = (V, T, P, S)$ be the grammar constructed from G_1, G_2 where

$$V = V_1 \cup V_2$$

$$T = T_1 \cup T_2$$

$$P = P_1 \cup P_2 \cup (S \rightarrow S_1 / S_2)$$

$$S = S \rightarrow S_1 / S_2$$

$\therefore L_1 \cup L_2$ is also CFL

Scanned with OKEN Scanner

2) Concatenation:

let L_1 and L_2 are two CFL then $L_1 . L_2$ is also CFL

Proof: Let $G_1 = (V_1, T_1, P_1, S_1)$ be the grammar that represented by the language L_1 $\quad L_1 = L(G_1)$

let $G_2 = (V_2, T_2, P_2, S_2)$ be the grammar that represented by the language L_2 $\quad L_2 = L(G_2)$

let us construct a CFG that accept the language $L_1 . L_2$

let $G = (V, T, P, S)$ be the grammar constructed from G_1, G_2 where

$V = V_1 \cup V_2$

$T = T_1 \cup T_2$

$P = P_1 \cup P_2 \cup (S \rightarrow S_1 S_2)$

$S = S$

$L_1 . L_2$ is also CFL

3) Kleen closure:

Let L is a CFL then L^* is also a CFL

Proof: let $G_1 = (V_1, T_1, P_1, S_1)$ be the grammar that represented by the language L_1 $\quad L_1 = L(G_1)$

let us construct a CFG that accept the language L_1^*

let $G = (V, T, P, S)$ be the grammar constructed from G_1 where

$V = V_1$

$T = T_1$

$P = P_1 \cup (S \rightarrow S1 / \epsilon)$

Scanned with OKEN Scanner

$S = S \rightarrow S_1 / \epsilon$

$\therefore L_1^*$ is also CFL

4) <u>Reverse:</u>

Let L be a CFL then L^R is also a CFL

Proof: Let $G_1 = (V_1, T_1, P_1, S)$ be the grammar that represented by the language L_1 $\quad L_1 = L(G_1)$

let us Construct a CFG that accept the language L_1^R

let $G = (V, T, P, S)$ be the grammar Constructed from G_1 where

$V = V_1$

$T = T_1$

$P = P_1^R \cup \{S \rightarrow S_1\}$

$S = S \rightarrow S_1$

$\therefore L_1^R$ is also CFL

Scanned with OKEN Scanner

8(b) Construct a Turing Machine for $L = \{a^n b^n c^n / n \geq 1\}$

Turing Machine Construction:

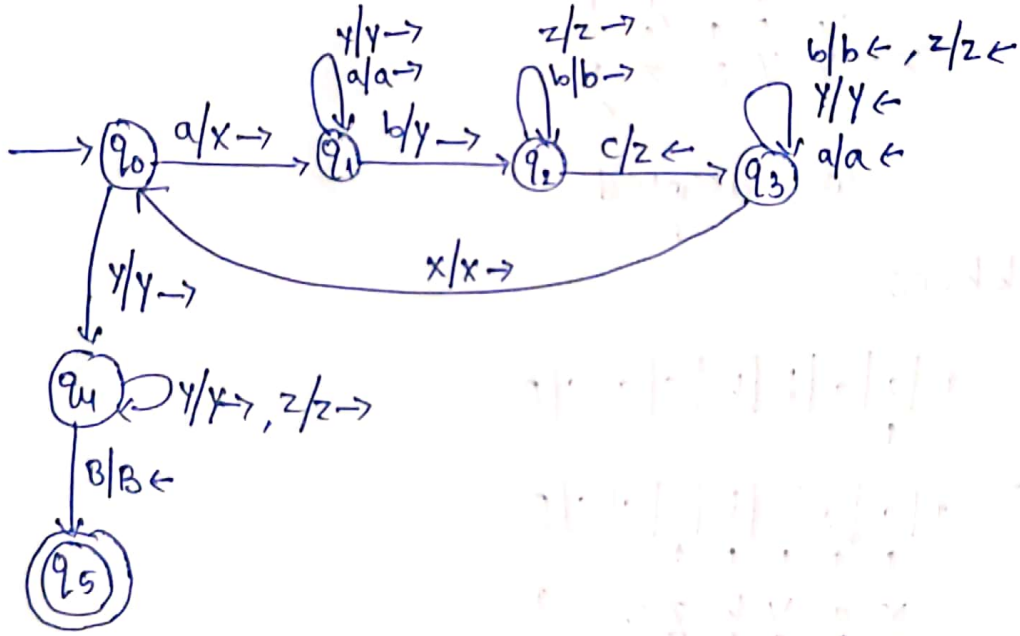
given language, $L = \{a^n b^n c^n / n \geq 1\}$

→ let $n=2$, then a string will be aabbcc

→ while traversing the string we replace 'a' with 'x', 'b' with 'y' and 'c' with z

→ after the string is completed, if the state is final state then string is accepted otherwise rejected.

→ Here we use Acceptor rather than transducer.



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

$\delta(q_1, b) = (q_2, y, \rightarrow)$ $\quad$ $\delta(q_2, z) = (q_2, z, \rightarrow)$

$\delta(q_2, c) = (q_3, z, \leftarrow)$ $\quad$ $\delta(q_3, a) = (q_3, a, \leftarrow)$

$\delta(q_1, a) = (q_1, a, \rightarrow)$ $\quad$ $\delta(q_3, b) = (q_3, b, \leftarrow)$

$\delta(q_1, y) = (q_1, y, \rightarrow)$ $\quad$ $\delta(q_3, y) = (q_3, y, \leftarrow)$

Scanned with OKEN Scanner

$\delta(q_3, z) = (q_3, z, \leftarrow)$ $\quad$ $\delta(q_4, Y) = (q_4, Y, \rightarrow)$

$\delta(q_3, X) = (q_0, X, \rightarrow)$ $\quad$ $\delta(q_4, z) = (q_4, z, \rightarrow)$

$\delta(q_0, Y) = (q_4, Y, \rightarrow)$ $\quad$ $\delta(q_4, B) = (q_5, B, \leftarrow)$

Ex:

1) $w = aabbcc$

B	a	a	b	b	c	c	B

B	a	a	b	b	c	c	B

X a Y b z c B

X a Y b z c B

X X Y Y Z Z B

X X Y Y Z Z B

$\therefore$ w is accepted

2) $w = aabbccc$

B	a	a	b	b	c	c	c	B

B	a	a	b	b	c	c	c	B

X a Y b z c c B

X X Y Y z z c B

X X Y Y Z Z c B

X X Y Y Z Z

$\Rightarrow$ here we will be at state q_4 where there is no transition with c

$\therefore$ w is rejected.

Scanned with OKEN Scanner

9(a) What is post Correspondence problem? Does pcp with two lists $X = (b, bab^3, ba)$ and $Y = (b^3, ba, a)$ have a solution?

Post Correspondence problem:

→ Introduced by Emil Post in 1946

→ If a problem can be reduced to the "post Correspondence problem", then we know that the problem is undecidable

→ Formal definition:

⇒ Given two lists of strings A and B (equal length)

$A = w_1, w_2, \ldots w_k \qquad B = x_1, x_2, \ldots x_k$

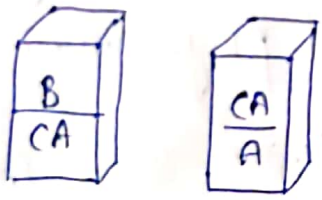
The problem is to determine if there is a sequence of one or more integers $i_1, i_2, \ldots i_m$ such that:

$$w_{i_1} w_{i_2} \ldots w_{i_m} = x_{i_1} x_{i_2} \ldots x_{i_m}$$

(w_i, x_i) is called a corresponding pair

Post ~~p~~ Correspondence problems can be represented in two ways:

1. Domino's Form:



2. Table Form:

	Numerator	Denominator
1	B	CA
2	CA	A

Scanned with OKEN Scanner

Given two lists A (or) $X = (b, bab^3, ba)$ and

B (or) $Y = (b^3, ba, a)$

	X	Y
i	w_i	x_i
1	b	b^3
2	bab^3	ba
3	ba	a

This pcp instance has a solution : 2, 1, 3

$$w_2 w_1 w_3 = x_2 x_1 x_3 = bab^3ba$$

∴ note each and every pair should be taken once (atleast)

$$\therefore \quad \frac{bab^3ba}{bab^3ba}$$

9(b) Explain about programming techniques for Turing Machine

The turing machines can be designed with the help of some conceptual tools called programming tools those are

1) Storage in the finite. State control
2) multiple tracks
3) Sub-routines

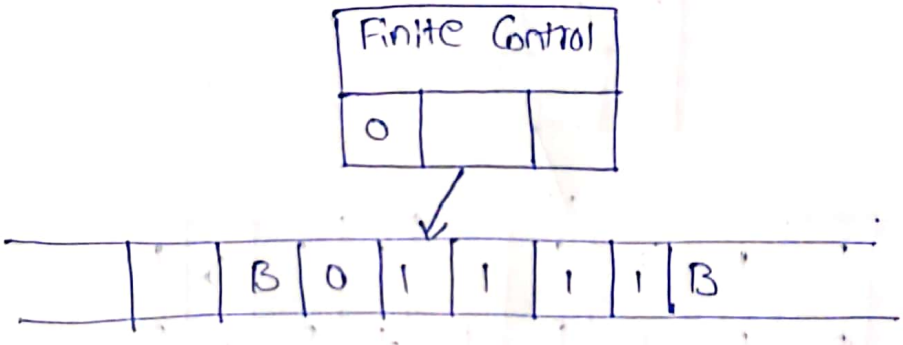
Scanned with OKEN Scanner

1) Storage in the finite State Control:

→ The model of the Turing machine has a finite State Control, this finite state Control Can be used to hold Some amount of information in addition to the State.

→ The finite State Control Stores the information in a pair of elements such as the Current State and the Current input Symbol pointed by the tape head as shown in figure.

Construct a TM to recognize the language $01^* + 10^*$



$\delta([q,B],0) = ([p,0],0,R)$

$\delta([q,B],1) = ([p,1],1,R)$

$\delta([p,0],1) = ([p,0],1,R)$

$\delta([p,1],0) = ([p,1],0,R)$

$\delta([p,0],B) = ([r,B],B,L)$

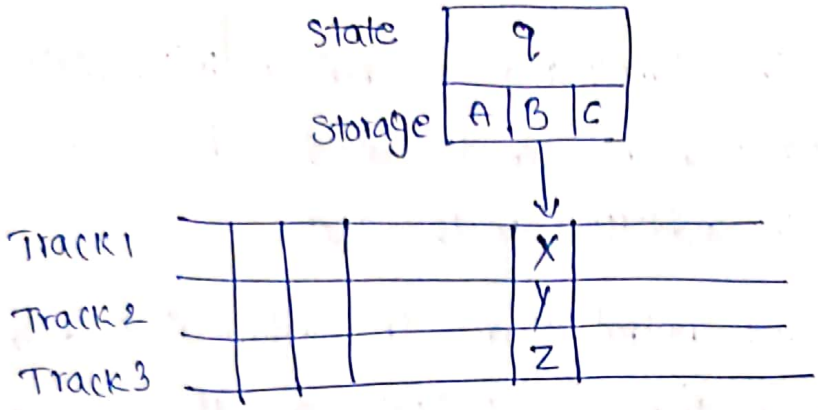
$\delta([p,0],B) = ([r,B],B,L)$

2) multiple Tracks:

→ Another useful tool is a tape with multiple tracks each cell in the track can held one symbol and tape alphabet of the Turing machine Consists of a tuple with one Component of each track

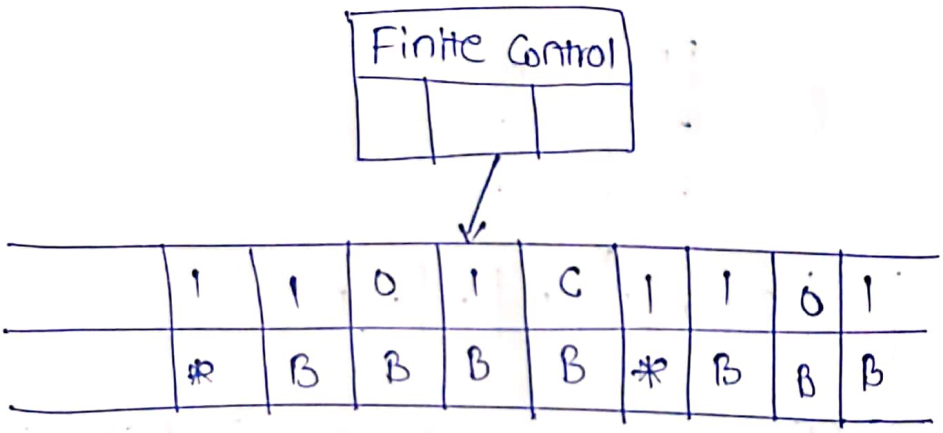
Scanned with OKEN Scanner

⇒ one tape head to read K symbols from the K tracks at one step



let $w = 1101$

$wcw = 1101c1101$



$\delta([q,B],[1,B]) = ([p,1],[1,*],R)$

$\delta([p,1],[1,B]) = ([p,1],[1,B],R)$

$\delta([p,1],[0,B]) = ([p,1],[0,B],R)$

$\delta([p,1],[c,B]) = ([r,1],[c,B],R)$

$\delta([r,1],[1,B]) = ([s,B],[1,*],L)$

Scanned with OKEN Scanner

3) Sub-routines:

→ In high level programming language the use of it is to build modularity in program development process, the same type of concept can be used in the Turing machine

→ A problem with same tasks to be repeated for many number of times, can be programmed using subroutines

→ A Turing Machine with subroutine is a set of states that perform some useful process.

Scanned with OKEN Scanner