TOC

Dt. B+
Pg.

Books : 1. AHO Ullman
2. Reference : i) KLP Mishra
ii) Adish K Pandey
iii) Udit Aggarwal

→ Automata
→ Usage
→ Alphabets
→ Strings
→ languages

Automata

It is difined as a system where energy, material and information are transformed and transmitted for performing some function without direct participation of a man.
Eg : Automatic Machine Tool, Automatic Packaging Machine, Automatic Printing Machine.

Study of Automata Thory

1. Automata Thory plays an imp. role when we are making software for designing and checking the behaviour of a digital ckt.
2. The lexical analyzer of the typical compiler i.e, the compiler content that breaks the I/P text into logical units called tokens
3. Software for scanning large bodies of text such as the collection of web pages to find occurances of words, phrases or other patterns
4. It is used for natural language processing

Dt.

Pg.

Alphabets

Symbol Σ : I/P alphabets

$\Sigma = \{A \text{ to } Z\}$ $\Sigma = \{a \text{ to } z\}$

Numerical $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$

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

(1.) Terminals	(2.) Non Terminals
• Numerical • Binary • Small letters	• Capital letters

a^* – Multiplicative closure / Kline closure

- It includes Null value

min. value = $\epsilon / \lambda / \wedge$

$\epsilon, a, aa, aaa, \ldots\ldots$

a^+ – Positive closure

- It doesn't includes Null value

min. value = a

$a, aa, aaa, \ldots\ldots$

Alphabets are defined as finite set of symbols or letters.

Eg. : characters and digits

Be Positive...

Dt. B+
Pg.

String / Word

A string or a word is the finite sequence of symbols selected from some alphabet length of a string is denoted by [| |]

Concatination of String

a = "Hello" b = "World"

$w_1 = a \cdot b$ = "Hello World"

w_2 = aaa

Let w_1 and w_2 be two strings, then concatination of strings is formed by making a copy of w_1 followed by a copy of w_2

$\Rightarrow w_1 \cdot w_2$ = abaaa

Reverse of a String

$(w_1)^R$ = ba

Reverse of a string can be achieved by flipping over the last symbol.

Be Positive...

Dt. 10/1/18
Pg. B+

Languages

Q1. language of all strings consisting of n 0's followed by n 1's where $n \geq 0$

Q2. Set of strings of 0's and 1's with equal no. of each

Q3. Set of binary string no. whose value is prime.

A language is a subset of Σ^* for some alphabet Σ

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

Ans = $\{\epsilon, 01, 0011, 000111, \ldots\}$

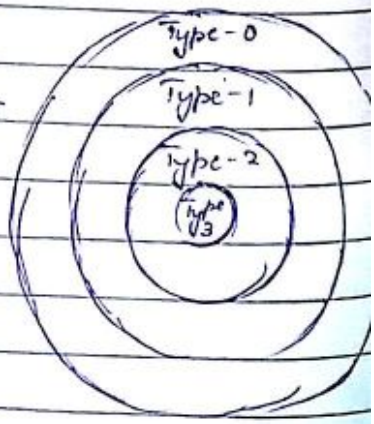
Ans 2 If condition is not given, then use Σ^*

Ans = $\{\epsilon, 01, 10, 0011, 0101, 1010, 1100, 0110, 1001 \ldots\}$

Ans 3 = $\{10, 11, 101, 111 \ldots\}$

CHOMSKY CLASSIFICATION OF LANGUAGE

Type-0 Unrestricted Gramar
Type-1 Context Sensitive Gramar
Type-2 Context free Gramar
Type-3 Finite Automata



Be Positive...

Dt. _____ B+
Pg. _____

Type	Name of Language	Production Restriction	Acceptor
0	Unrestricted / Recursively Enumerable language	A. = Any string with non-terminal B. = Any string	Turing Machine
1	Context sensitive	A. = Any string with non terminal B. = Any string as long as A	Linear bound automata
2	Context free	A. = One non-terminal B. = Any string	Push down automata
3	Regular	A. = One non terminal B. = ax or B = a where a = terminal x = Non-terminal [ε: works as 1]	Finite automata

Gramar : (V, T, P, S)

V → Non terminal
T → Terminal
P → Production
S → Start Symbols

eg :
S → aSb
S → bSb
S → a/b → Production

↓
S

Be Positive...

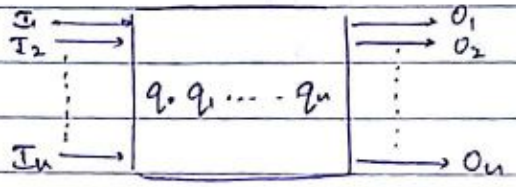
Dt.
Pg.

B+

Finite Automata

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

$Q \rightarrow$ Set of States
$\Sigma \rightarrow$ I/P Alphabet
$\delta \rightarrow$ Transitions funct"
$q_0 \rightarrow$ Initial State
$F \rightarrow$ Final State

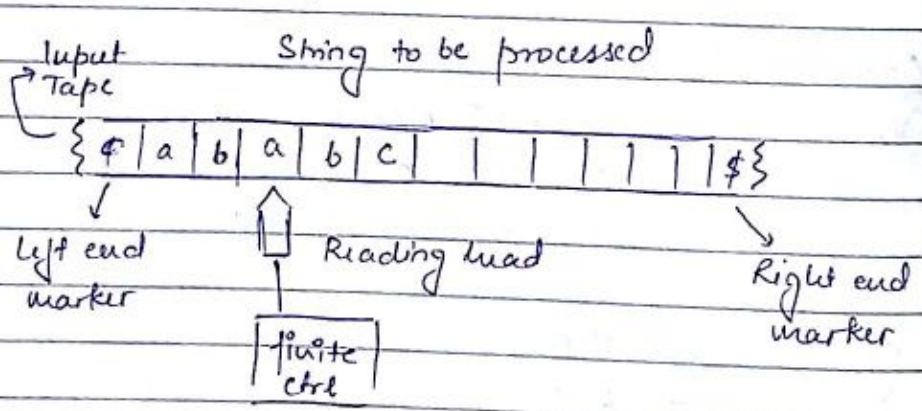


Automata

Characterstics

1. Input
2. Output
3. State relation
4. O/P relation
5. State

Block Diagram of F.A.

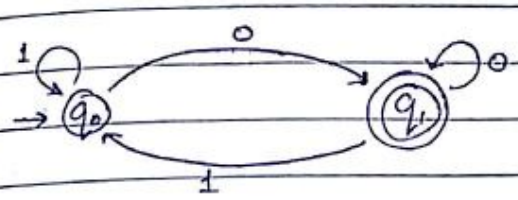


Be Positive...

Dt. _____ B+
Pg. _____

1. $\rightarrow q_0$ / $\hookrightarrow (q_0)$ ⇒ Initial State

2. (q_0) / (q_1) / $((q_2))$ ⇒ Finite State (q1 labeled)

3. 

Possibilities
→ 0011
→ 101
→ 1100011*

• Self loops can be neglected

11/1/18

Acceptability of a String

A string 'x' is accepted by finite automata $M(Q, \Sigma, \delta, q_0, F)$
if $\delta(q_0, x) = q$ for some $q \in F$

Ques Consider a FSM where transition function δ is given in the form of transition table.
Here $V = \{q_0, q_1, q_2, q_3\}$ $\Sigma = \{0,1\}$, $F = \{q_0\}$
Check for acceptability of string 110010

S/Σ	0	1
→ (q_0)	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Transition Table

Ans ⇒ $\delta(q_0, 110010)$
$\delta(q_1, 10010)$
$\delta(q_0, 0010)$
$\delta(q_2, 010)$
→ $\delta(q_0, 10)$
$\delta(q_1, 0)$
$\delta(q_3)$

Be Positive...

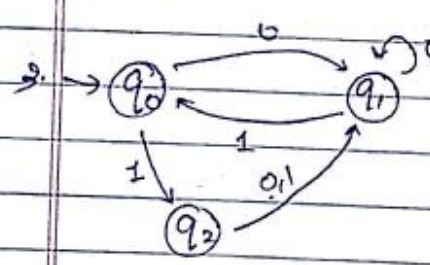
Dt.
Pg.

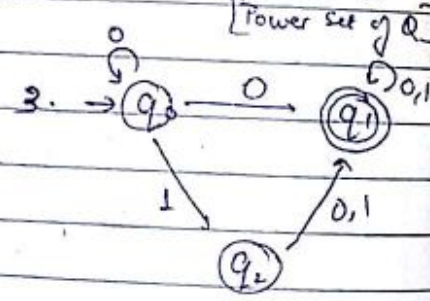
ii) 110101

$\delta(q_0, 110101)$
$\delta(q_1, 10101)$
$\delta(q_0, 0101)$
$\delta(q_2, 101)$
$\delta(q_3, 01)$
$\delta(q_1, 1)$
$\delta(q_0)$

Finite Automata

DFA	NDFA / NFA
1. Deterministic	1. Non-Deterministic
$(Q, \Sigma, q_0, F, \delta)$	$(Q, \Sigma, q_0, F, \delta)$
2. $Q \times \Sigma = Q$	2. $Q \times \Sigma = 2^Q$ [Power set of Q]





Qm Construct a DFA equivalent to

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

q_0 ↓ Initial state; $\{q_0\}$ ↓ Final state

where δ is given by

iv)

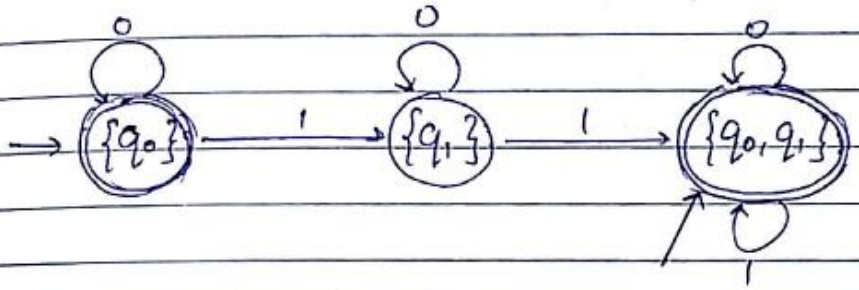
δ / Σ	0	1
→ q_0	q_0	q_1
q_1	q_1	$q_0 q_1$

Be Positive...

Dt.
Pg. B+

Ans

S/Σ	0	1	
→{q0}	{q0}	{q1}	
{q1}	{q1}	{q0,q1}	
{q0,q1}	{q0,q1}	{q0,q1}	→ reducing the repeated state



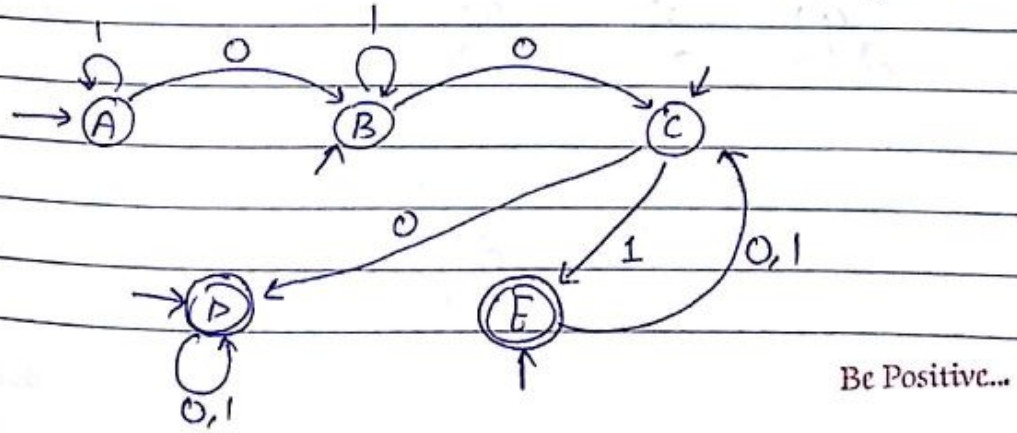
Since q_0 is initial & final state
∴ where ever it is present will be treated as initial & final state.

ii)

S/E	0	1
→ q0	q0q1	q0
q1	q2	q1
q2	q3	q3
(q3)	—	q2

Ans..

S/Σ	0	1
→A {q0}	{q0q1}	{q0}
B {q0q1}	{q0q1q2}	{q0q3}
C {q0q1q2}	{q0q1q2q3}	{q0q1q3}
D {q0q1q2q3}	{q0q1q2q3}	{q0q1q2q3}
E {q0q1q3}	{q0q1q2}	{q0q1q2}

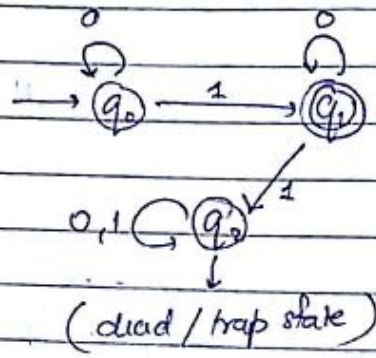


Be Positive...

Dt.
Pg.

Ques Design a F.A. that Accepts strings containing exactly '1' over alphabet {0,1} ('1' only once)

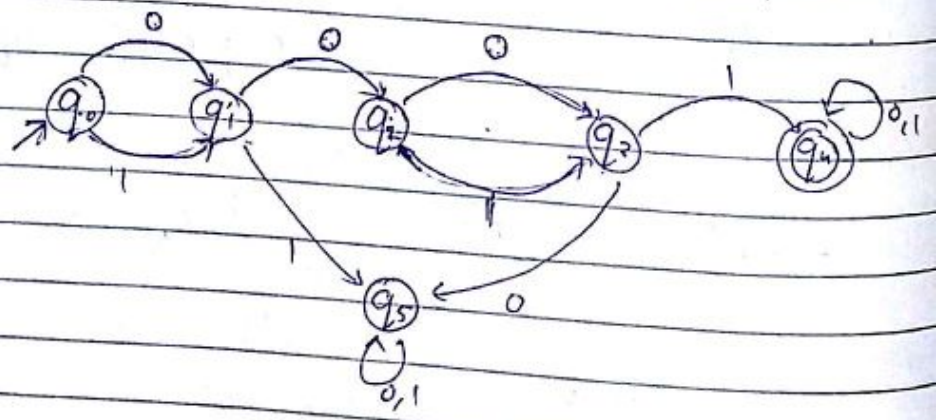
Possibilities : 1, 10, 01, 010, 100, 001, 00100, ...



Ques Design a FA which accepts the language
$L = \{w \in (0,1)^*\}$
such that second symbol of w is 0 and fourth symbol is 1

Q2 Design a finite automata that accepts even no. of zeros and even no. of ones.

Ans: Possibilities 0001, 0011, 1000, 1010, 1100, 1101

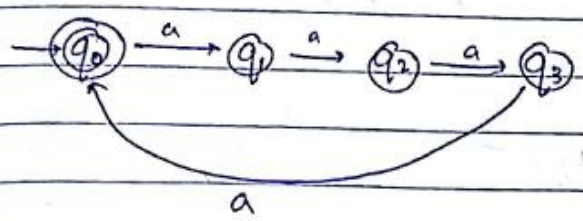


Be Positive...

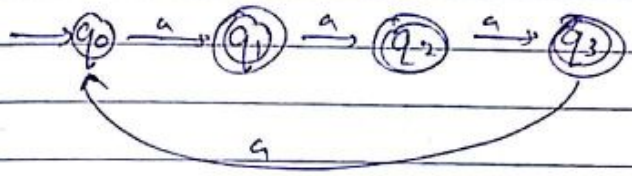
Dt.
Pg.

B+

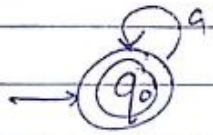
Q1 $L = \{ a^n \mid n \bmod 4 = 0 \}$



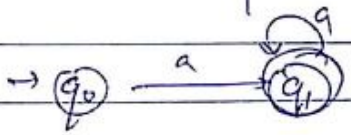
$L = \{ a^n \mid n \bmod 4 \neq 0 \}$



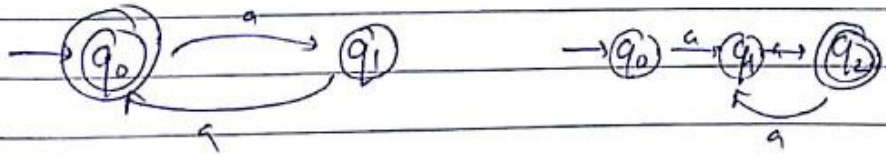
Q3 Construct a DFA for a^*



Q3 Construct a DFA for a^+



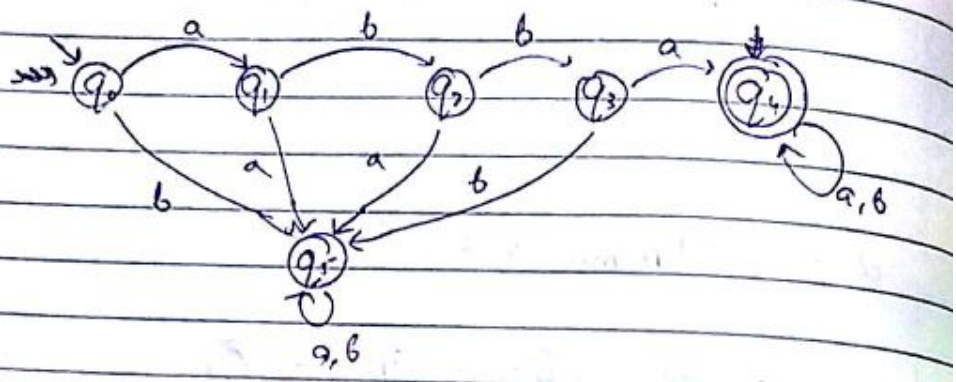
Q5 $L = \{ a^{2n} \mid n \geq 0 \}$ $n \geq 1$



Be Positive...

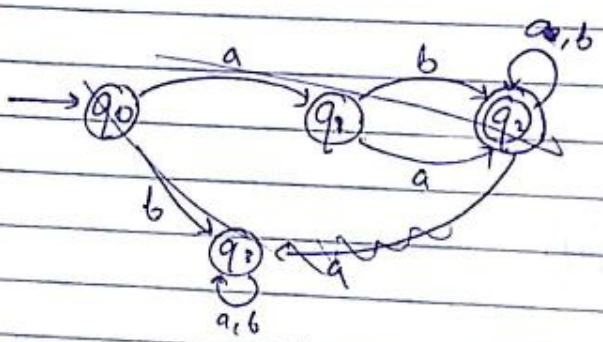
Dt.
Pg.

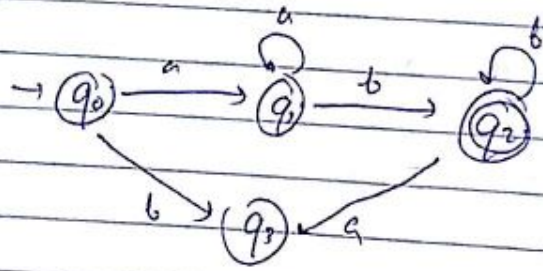
Q) Construct a DFA that accepts all a set of all strings over {a, b} which starts with abba.



Q) $L = \{a^+b^+, \Sigma = \{a, b\}\}$

Pombilities
ab a^2b ab^2

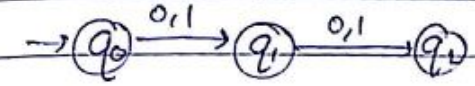


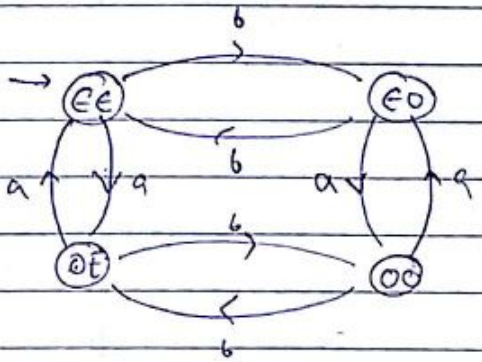


Be Positive...

Dt.
Pg. B+

Q2 ε, 0011, 0101, 1001, 0110, 1010, 1100, 00001111, 0000, 1111, 00011101



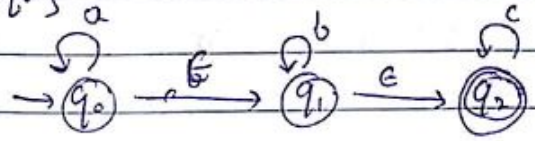


Final state : the one given in ques."

17/1/18

Conversion of NFA with ε-Transition to NFA without ε-Transition

Q Consider the NFA with ε-transition $M = (Q, \Sigma, q_0, \delta, F)$
$Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b, c\}$ and ε moves $IE. = \{q_0\}$
F.S. $= \{q_0\}$



Sol. Make a transition table

S/E	a	b	c	ε
→ q_0	$\{q_0\}$	$\{\phi\}$	$\{\phi\}$	$\{q_1\}$
q_1	$\{\phi\}$	$\{q_1\}$	$\{\phi\}$	$\{q_2\}$
q_2	$\{\phi\}$	$\{\phi\}$	$\{q_2\}$	$\{\phi\}$

Be Positive...

Dt.
Pg. B+

1. ϵ - Closure$(q_0) = \{q_0, q_1, q_2\}$
2. ϵ Closure$(q_1) = \{q_1, q_2\}$
3. ϵ Closure$(q_2) = \{q_2\}$

Now we have to find δ

$\delta(\{q_0, q_1, q_2\}, a) = \epsilon$ Closure $(\delta\{q_0, q_1, q_2\}, a)$
$= \epsilon$ Closure $(\delta\{q_0, a\} \cup \delta\{q_1, a\} \cup \delta\{q_2, a\})$
$= \epsilon$ Closure $(q_1 \cup \phi \cup \phi)$
$= \epsilon$ Closure $(q_0) \Rightarrow \{q_0, q_1, q_2\}$

Similarly

$\delta'(\{q_0, q_1, q_2\}, b) = \epsilon$ closure $(\delta(q_0 q_1 q_2), b)$
$= \epsilon$ Closure $(\delta(q_0, b) \cup \delta(q_0, b) \cup \delta(q_0, c))$
$= \epsilon$ closure $(\phi \cup q_1 \cup \phi)$
$= \epsilon$ Cosure $(q_1) \Rightarrow \{q_1, q_2\}$

Similarly

$\delta'(\{q_0, q_1, q_2\}, c) = \epsilon$ Closure $(\delta(q_0, q_1, q_2), c)$
$= \epsilon$ Closure $(\delta(q_0, c) \cup \delta(q_1, c) \cup \delta(q_2, c))$
$= \epsilon$ Cosure $(\phi \cup \phi \cup q_2)$
$= \epsilon$ Closure $(q_2) \Rightarrow \{q_2\}$

Similarly we will find ϵ Closure of other state ~~with~~ as well

$\delta'(\{q_1, q_2\}, a) = \epsilon$ Closure $(\delta(q_1, a) \cup \delta(q_2, a))$
$= \epsilon$ Cosure $(\phi \cup \phi)$
$= \phi$

Similarly

$\delta'(\{q_1, q_2\}, b) = \epsilon$ Closure $(\delta(q_1, b) \cup \delta(q_2, b))$
$= \{q_1, q_2\}$

Be Positive...

Dt. _____ B+
Pg. _____

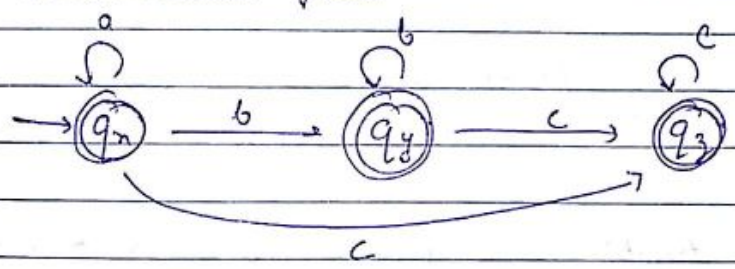
$\delta'(\{q_1, q_2\}, c) = \{q_2\}$
$\delta'(q_2, a) = \phi$
$\delta'(q_2, b) = \phi$
$\delta'(q_2, c) = \{q_2\}$

So transition table without ϵ transition is

δ / Σ	a	b	c
$\{q_0, q_1, q_2\}$	$\{q_0 q_1 q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_2\}$	ϕ	ϕ	$\{q_2\}$

Let $\{q_0 q_1 q_2\} = q_x$
$\{q_1 q_2\} = q_y$
$\{q_2\} = q_z$

Transition diagram



Properties of Finite State Machine

1. Recognition problem can be solved in linear time
2. There is an algo. to transform each automata into unique equivalent automata with least number of states
3. Finite acceptors provide a way to describe potentially

Dt.
Pg.

infinite languages in finite ways.

④ Determining whether a string is accepted or rejected by the acceptor is linear in the length of the string.

Limitations of Finite Automata

① It can have finite no. of states
② It can recognize and process regular language
③ It does not have any auxillary memory, so it works in process and forget mode.
Thus is no finite automata that recognizes set of binary strings consisting of equal no. of a's and equal no. of b's.
(Push down stack -)

18/1/18

Mealy and Moore Machine

Moore Machine

It is a 6 tuple machine $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$
where Q : Finite set of states
Σ : I/P alphabet
Δ : O/P alphabet
δ : transition function
λ : O/P funct^n mapping $Q \to \Delta$
q_0 : Initial State

Be Positive...

Dt. _____ B+
Pg. _____

If the O/P function $Z(t)$ depends only on the present state and is independent of the current I/P then the O/P function is given by

$$Z(t) = \lambda[q(t)]$$

Mealy Machine

A Mealy machine is also a 6 tuple machine $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where all the symbols have same meaning except 'λ', which is an O/P funct^n mapping $\Sigma \times Q \to \Delta$

The value of the O/P funct^n $Z(t)$ is a funct^n of present state $q(t)$ and present I/P $x(t)$

$$Z(t) = \lambda(q(t) \cdot x(t))$$

Representation of Moore Machine

Present State	Next State		Output
	a=0	a=1	
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_0	q_0	0

Representation of Mealy Machine

Present State	Next State			
	a=0		a=1	
	State	O/P	State	O/P
→ q_0	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Be Positive...

Dt
Pg. B+

Q. Consider a Mealy Mach. described by the transition table. Construct a Moore Mach. which is equivalent to Mealy Mach

Present State	Next State			
	a=0		a=1	
	State	O/P	State	O/P
→ q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Ans

Present State	Next State			
	a=0		a=1	
	State	O/P	State	O/P
→ q_1	q_3 0	0	q_2 0	0
q_2 0	q_1	1	q_4 0	0
q_2 1	q_1	1	q_4 0	0
q_3	q_2 1	1	q_1	1
q_4 0	q_4 1	1	q_3	0
q_4 1	q_4 1	1	q_3	0

Breaking of states takes place if the O/P is diff, and the state changes from q_1 to q_1A where A is the O/P of q_1 for either a=0 or 1

Be Positive...

Dt.
Pg. B+

Present State	Next State a=0	Next State a=1	O/P
→ q_1	q_3	q_{20}	0
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	1
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Qu

Present state	Next State a=0	Next State a=1	O/P
→ q_1	q_1	q_2	0
q_2	q_1	q_3	0
q_3	q_1	q_3	1

Consider a Moore Mach. described by the transition table. Construct an equivalent Mealy Mach.

Ans

Present State	Next St. a=0 S.	Next St. a=0 O/P	Next St. a=1 S	Next St. a=1 O/P
→ q_1	q_1	0	q_2	0
q_2	q_1	0	q_3	1
q_3	q_1	0	q_3	1

(Check the O/P of P.S. and substitute it in the next state)

Be Positive...

Dt. 19/1/18
Pg.

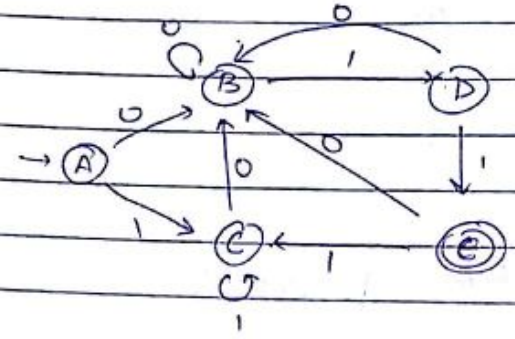
Minimization of DFA

Conditions

$\delta(A, x) \rightarrow F$ OR $\delta(A, x) \nrightarrow F$

$\delta(B, x) \rightarrow F$ $\delta(B, x) \nrightarrow F$

Ques



State table

S/Σ	0	1
→ (A)	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C

Zero Equivalence

0E) → {A, B, C, D}, {E} → Final state forms seperate group

1E) → {A, B, C}, {D}, {E} → value of D at 1 is E, which lies out the other group, hence D is seperated

*[we check for final state in same group]

2E) → {A, C}, {B}, {D}, {E} → in reference to 1E)

↓

Value of A & C at 0 lies in the same group and at 1 also lies in the same group

∴ A & C can be referred at same state

Be Positive...

Dt.
Pg. B+

Que

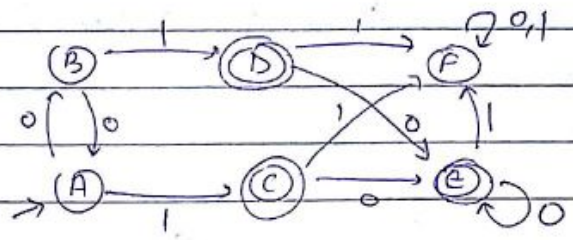
S/Σ	0	1
q_0	q_1^{24}	q_5^{25}
q_1	q_6^{23}	q_2^{11}
q_2	q_0^{23}	q_2^{11}
q_3	q_2^{11}	q_0^{23}
q_4	q_7^{24}	q_5^{25}
q_5	q_2^{11}	q_0^{23}
q_6	q_6^{23}	q_4^{27}
q_7	q_5^{23}	q_2^{1}

0E) → $\{q_0\ q_1\ q_3\ q_4\ q_5\ q_6\ q_7\}\ \{q_2\}$ (2) (1)

1E) → $\{q_0\ q_4\ q_6\}\ \{q_1\ q_5\ q_7\}\ \{q_3\ q_5\}\ \{q_2\}$ (3) (4) (5)

2E) → ~~$\{q_2\}\ \{q_4\}\ \{q_5\}\ \{q_7\}$~~ $\{q_0\ q_7\}\ \{q_3\ q_5\}\ \{q_0\ q_4\}\ \{q_2$

Que



S/Σ	0	1
→ A	B^{13}	C^{22}
B	A^{13}	D^{22}
(C)	E^{22}	F^{14}
(D)	E^{22}	F^{14}
(E)	E^{22}	F^{14}
(F)	F^{14}	F^{14}

Be Positive...

Dt.
Pg.

0E) {AB F} {CDE}
1 2

1E) {AB} {F} {CDE}
3 4 2

2E → {AB} {F} {CDE}

	0	1
AB	AB	CD
CDE	E	F
F	F	F

22/1/18

Operators of Regular Expression

1. Union

Union of two regular expression or union of two regular language denoted by L_1 and L_2 is the set of strings that are either in L_1 or L_2 or both.

$L_1 = \{001, 10, 111\}$

$L_2 = \{\epsilon, 001\}$

$L_1 \cup L_2$ / $L_1 + L_2 = \{001, 10, 111, \epsilon\}$

2. Concatination

Concabination of two language L_1 and L_2 is a set of strings that can be formed by taking any string in L_1 and concatinating it with any string in L_2. It is denoted by

$L_1 \cdot L_2 = \{001 \cdot \epsilon, 001 \cdot 001, 10 \cdot \epsilon, 10 \cdot 001, 111 \cdot \epsilon, 111 \cdot 001\}$

$= \{001, 001001, 10, 10001, 111, 111001\}$

Be Positive...

Dt.
Pg. B+

3. Kleene Closure (L^*)

Kleene Closure of a language is denoted by L^* and represents the set of those strings that can be formed by taking any no. of strings from L, possibly with repitition and concatenating all of them.

Eg.: If $L = \{0,1\}^*$,
then L^* is a set of all strings of 0's and 1's including null string.

4. Positive Closure (L^+)

It is a set of those strings that can be formed by taking any no. of strings from L possibly with repitition and concatinating all of them excluding the null string.

$$L^+ = L^+ - \epsilon$$

Ques The set of all strings of 0's and 1's ending in 00
$(0+1)^*00$

Ques Set of all strings of 0's and 1's beginning with 0 and ending with 1
$0(0+1)^*1$

Ques $L = \{\epsilon, 11, 1111, 111111 \ldots\}$
$(11)^*$

Ques Set of all strings of a's and b's containing exactly one a
b^*ab^*

Be Positive...

Dt.
Pg. B+

Qw Set of all strings of a's and b's having at most 2 a's

$b^* + b^*ab^* + b^*ab^*ab^*$

Qw Set of all strings of a's and b's having even no. of a's followed by odd no. of b's

~~$(aa)^*(aa)^*bb$~~

~~$(aa)^*(b+(bbb)^*)$~~ $(aa)^*(bb)^*b$

Qw Set of all strings of a's and b's having atleast one a and atleast one b

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

* Qw Find a regular expression consisting of 0's and 1's having atleast one pair of consecutive zeros

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

Qw Write a regular expression for the set of strings of a's and b's whose 5th symbol from the right end is 'c'

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

$(a+b)^*a(a+b)^4$

Identities for RE

I_1	$\phi + R = R$	I_7	$RR^* = R^*R$
I_2	$\phi R = R\phi = \phi$	I_8	$(R^*)^* = R^*$
I_3	$\Lambda R = R\Lambda = R$	I_9	$\Lambda + RR^* = R^* = \Lambda + R^*R$
I_4	$\Lambda^* = \Lambda$	I_{10}	$(PQ)^*P = P(QP)^*$
I_5	$R + R = R$	I_{11}	$(P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^*$
I_6	$R^*R^* = R^*$	I_{12}	$(P+Q)R = PR + QR$ and $R(P+Q) = RP + RQ$ Be Positive...

Dt. _____ B+
Pg. _____

Qu) $R = \Lambda + \underbrace{1^*(011)^*}_{P}\underbrace{(1^*(011)^*)^*}_{P^*}$

Identity 9

$\Rightarrow R = (1^*(011)^*)^*$

Identity 11

$\Rightarrow R = (1 + 011)^*$

Qu) $(1 + 00^*1) + (1 + 00^*1)(0 + 10^*1)^*(0 + 10^*1) = 0^*1(0 + 10^*1)^*$

Prove LHS = RHS

LHS

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

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

$= (\epsilon + 00^*)1(0 + 10^*1)^*$

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

$= RHS$

24/1/16

Finite Automata with Λ-moves

Suppose we want to replace a Λ-move from vertex 1 to vertex 2

Step 1: Find all the edges starting from V_2

Step 2: Duplicate all the edges starting from V_1 without ~~having~~ changing edge labels

Step 3: If V_1 is the initial state, make V_2 an initial state

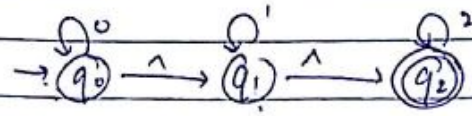
Step 4: If V_2 is the final state, make V_1 also as a final state.

Be Positive...

Dt.

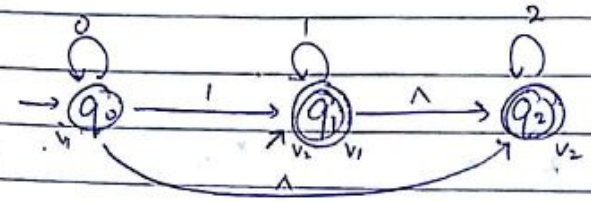
Pg.

Que: Consider a F.A. with ∈ moves Obtain an equivalent automata without ∧ move

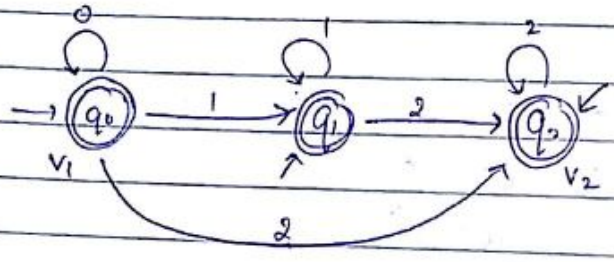


Sol^n

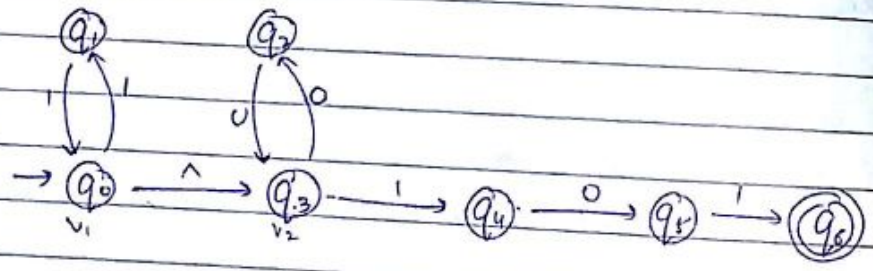
(1.)



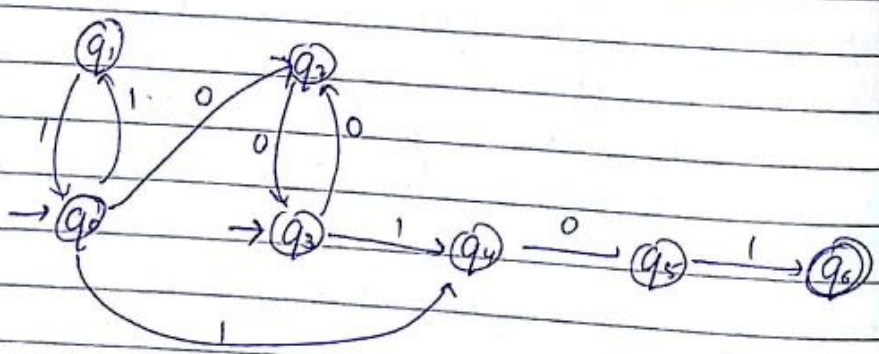
(2)



Que



(1)



Be Positive.

Dt.
Pg.

ARDEN's Theorem

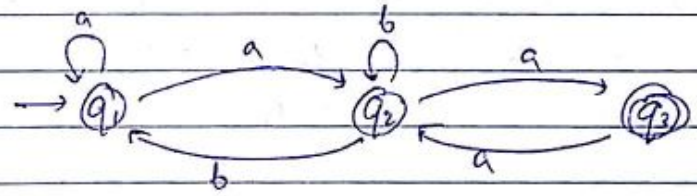
Let P & Q be two regular exp. over i/p alphabet (Σ). If P does not contain ϵ. Then the following eq^n in R is given by $R = Q + RP$ has a unique sol^n i.e. $R = QP^*$

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

Ques Consider the transition system given by the diagram prove the string as

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

Sol^n



$q_1 = \epsilon + q_1a + q_2b$ ——— (1)

$q_2 = q_1a + q_2b + q_3a$ ——— (2)

$q_3 = q_2a$ ——— (3)

Put the value of q_3 from (3) to (2)

$\Rightarrow q_2 = q_1a + q_2b + q_2aa$

$\Rightarrow q_2 = q_1a + q_2(b+aa)$

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

$\therefore q_2 = q_1a(b+aa)^*$ ——— (4)

Put the value of q_2 from (4) to (1)

$q_1 = \epsilon + q_1a + q_1a(b+aa)^*b$

$= \epsilon + q_1(a + a(b+aa)^*b)$

Be Positive...

Dt.

Pg.

$R = Q + RP$

$\Rightarrow q_1 = \epsilon \cdot (a + a(b+aa)^* b)^*$

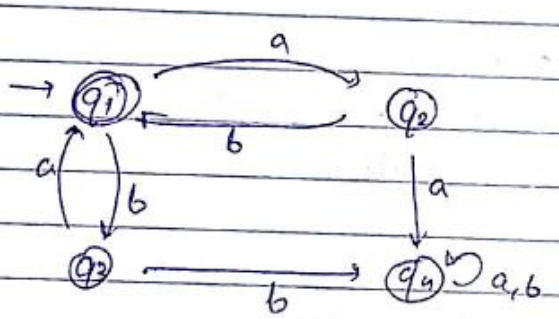
$\Rightarrow q_1 = (a + a(b+aa)^* b)^*$ ——— (5)

Put value of q_1 from (5) to (4)

$q_2 = (a + a(b+aa)^* b)^* \cdot a(b+aa)^*$ ——— (6)

$q_3 = (a + a(b+aa)^* b)^* \cdot a(b+aa)^* a$

Qus Prove that finite automata whose transition diagram is given that accepts the set of all strings over the alphabet a b with equal no. of a's and b's such that each prefix has atmost one more a than the b's and atmost one more b's than a's



$q_1 = \epsilon + q_2 b + q_3 a$

$q_2 = q_1 a$

$q_3 = q_1 b$

$q_4 = q_2 a + q_3 b + q_4 (a+b)$

$q_4 = q_1 aa + q_1 bb + q_4 (a+b)$

$q_4 = q_1 (aa+bb) + q_4 (a+b)$

$R = Q + RP$

Be Positive...

Dt.
Pg. B+

$\Rightarrow q_4 = q_1(aa+bb)\cdot(a+b)^*$

$q_1 = \epsilon + q_1 ab + q_1 ba$

$q_1 = \epsilon + q_1(ab+ba)$

$q_1 = (ab+ba)^*$

21/1/18

How to construct finite automata from Regular Expression

Step 1: Take two states, one initial and other one as final.
Write the regular expression as path value from initial state to final state

Step 2: Now divide the regular expression in two or more sub expressions

Step 3: Introduce new intermediate states b/w initial and final state

Step 4: In the end remove emsylon is required.

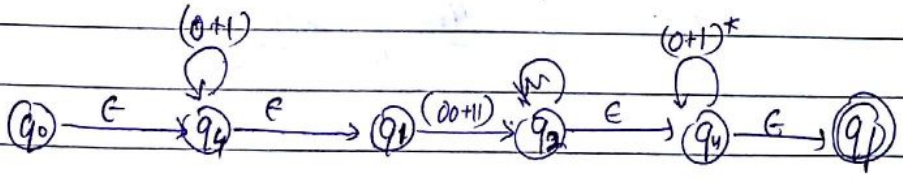
Ques: $(0+1)^*(00+11)(0+1)^*$

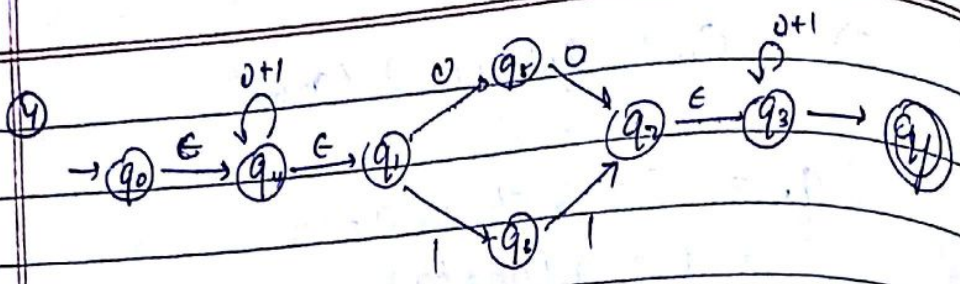
① $\rightarrow (q_0) \xrightarrow{(0+1)^*(00+11)(0+1)^*} (q_f)$

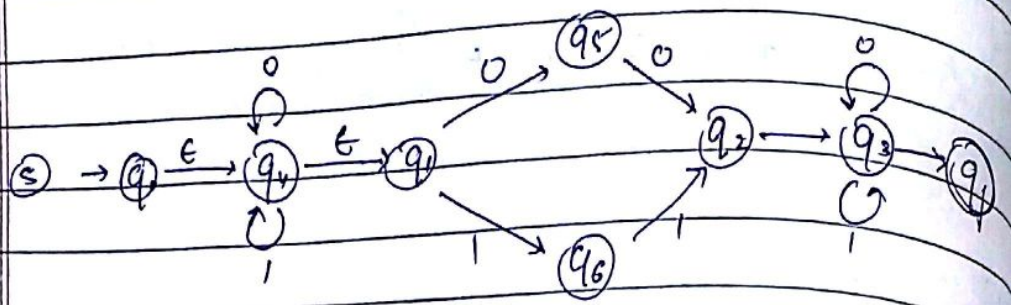
multiplication → series
addition → parallel

② $(q_0) \xrightarrow{(0+1)^*} (q_1) \xrightarrow{(00+11)} (q_2) \xrightarrow{(0+1)^*} (q_f)$

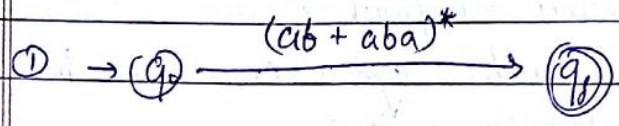
③

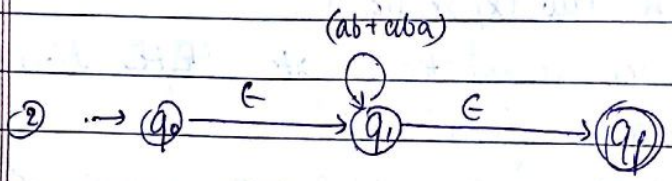


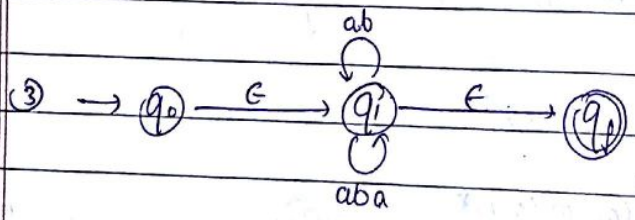


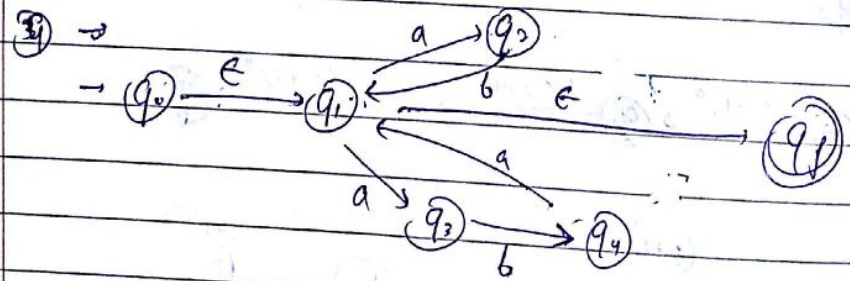


Que: $r = (ab + aba)^*$



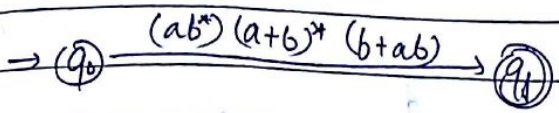


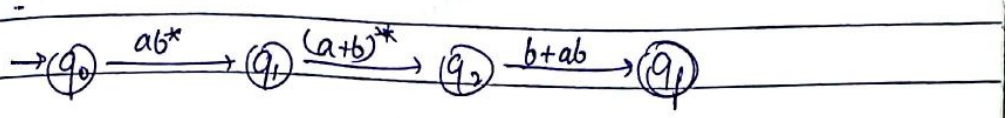


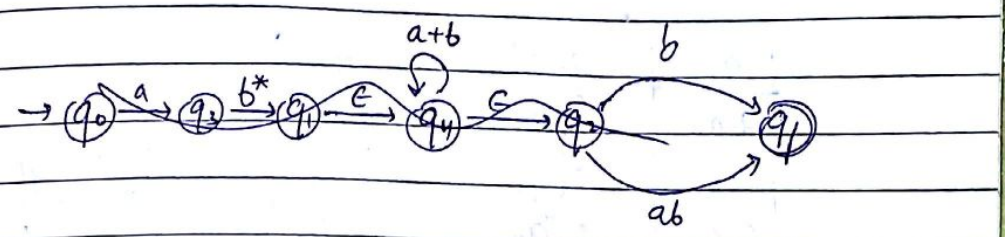


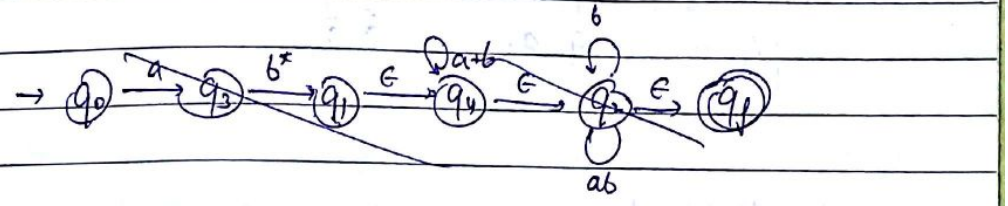
Dt. _____ B+
Pg. _____

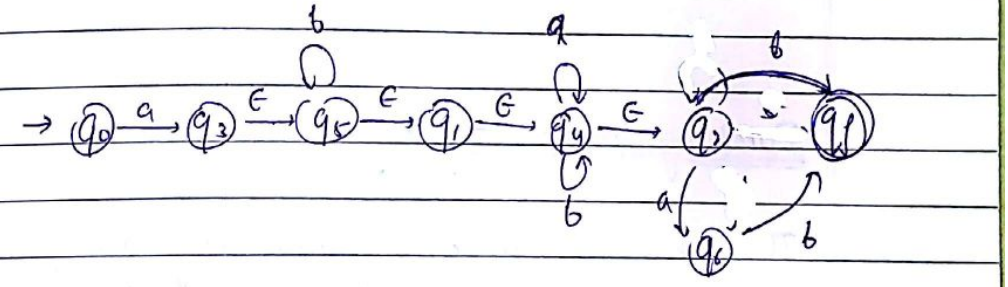
Q RE = $(ab^*)(a+b)^*(b+ab)$









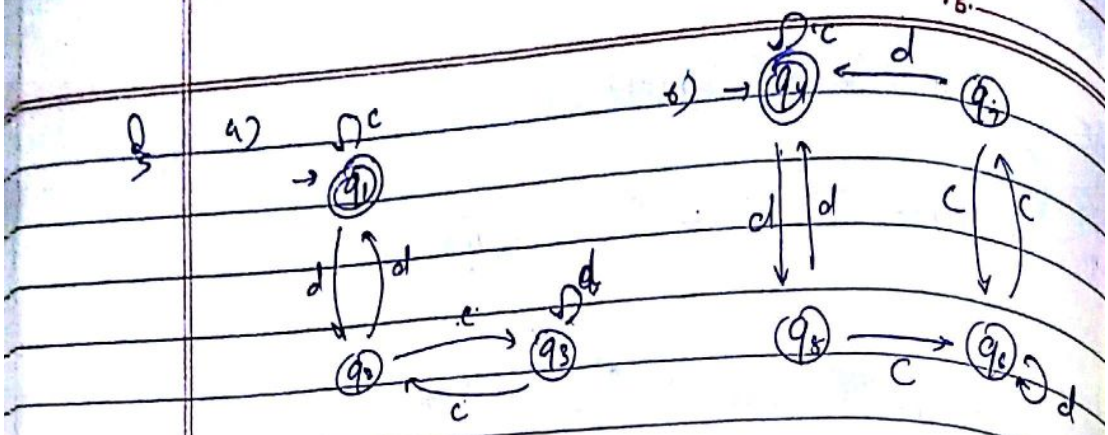


Equivalence of two finite automata

Two finite automatas are equivalent if they accept same set of strings; when two finite automata's are not equivalent then there is some string 'w' over I/P alphabet (Σ) by which one automata reaches a final state whereas other reaches at a non-final state.

Be Positive...

Dt.
Pg.



qq'	q_1q_1'	q_2q_2'
q_1q_4	$q_1^F q_4^F$	$q_2^{NF}q_5$
q_2q_5	$q_3^{NF}q_6$	$q_1^F q_4^F$
q_3q_6	$q_2^{NF}q_7$	$q_3^{NF}q_6$
q_2q_7	$q_3^{NF}q_6$	$q_1^F q_4^F$

If any 1 pair gets diff. labels, then they are non equivalent

Hence the given FA are equivalent

Pumping lemma for Regular languages

The pumping lemma is made up of two 5 words Pumping + lemma, where pumping refers to generate many VP strings by pushing a symbol in an VP string, again and again. The word lemma refers to intermediate theorem. It is a powerful tool for proving certain languages as non regular

It is based on the pigeon hole principle i.e any sufficiently long string in 'L' should repeat some states in the DFA and therefor it contains a cycle.

Be Positive.

Dt. 1/2/18
Pg. B+

Application of pumping lemma

Steps to prove that certain language is not regular

1. Assume that L is regular. Let 'n' be the number of states in corresponding finite automata
2. Choose a string 'w' such that $|w| \geq n$. Use pumping lemma to write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$
3. Find a suitable integer 'i' such that $xyz \notin L$. This contradicts our assumption. Hence L is not regular.

Ques Prove that languages $L = \{a^n b^n \text{ for } n = 0, 1, 2 \ldots\}$ is not regular.

Let us assume L is regular

Case I : $xy^i z$

$a^2 b^2 \in$

According to pumping lemma there must be string xyz such that all the words are of the form $x(y)^i z$

Case I: If middle part y is made up of entirely 0's, i.e.

$x\,aaa\ldots\ldots z$

If we pump it as xy^2z, xy^3z, then number of a's will increase, but it is not required.

Be Positive...

Case I: xy^iz

$\in a^2b^2$ $\quad i=1 \Rightarrow a^2b^2$

$\quad i=2 \Rightarrow a^4b^2$ X

Case II:

xy^iz

$a^2b^2 \in$ $\quad i=1 \Rightarrow a^2b^2$

$\quad i=2 \Rightarrow a^2b^4$ X

Case III:

xy^iz

$\in (ab^2)^i \in$ $\quad i=1 \Rightarrow a^2b^2$

$\quad i=2 \Rightarrow (a^2b)(a^2b)$

Case III

Ques $L = \{a^n b a^n$ for $n = 0, 1, 2, \ldots\}$

Case I: xy^iz

$a^n b a^n$ $\quad i=1$

Be Positive

Dt.
Pg.

UNIT - II

Context Free Grammar

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

- V → non-terminal
- T → terminal
- P → Production
- S → start symbol

$\alpha \rightarrow \beta$

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

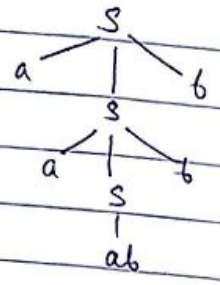
Q Construct a CFG for the language

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

Ans Pass $[ab, a^2b^2, a^3 \cdot b^3]$

$S \rightarrow aSb$

$S \rightarrow ab$



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

Ans ⇒ $b, ab^2, a^2b^3 \ldots$

For $n = 1$

$S \rightarrow b$

$S \rightarrow aSb$

$S \rightarrow abb$

$S \rightarrow aSb$

Qu: $L = \{a,b\}^* = \{\epsilon, a, b, aa, bb, ab, ba..\}$

$S \to \epsilon$

$S \to aS$

$S \to bS$

Qu: $L = \{a,b\}^+ = \{a, b, aa, bb, ab, ba ---\}$

$S \to a$ $\quad$ $S \to aS$

$S \to b$ $\quad$ $S \to bS$

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

$[S \to a/b \;,\; S \to as/bS]$ → Bakus Naur Form

Qu: $L = \{ w \in \{a,b\}^* \}$ $\quad N(a) = N(b)$

$\epsilon, ab, ba, a^2b^2, b^2a^2$

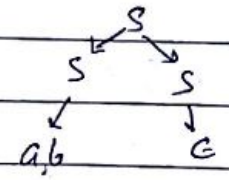
$S \to ab\epsilon$ $\quad$ $S \to ab$

$S \to aS$ $\quad$ $S \to \epsilon$

$S \to bS$ $\quad$ $S \to asb$

$S \to asb$ $\quad$ $S \to bsa$

$S \to SS$ [main]



Qu: $L = \{ w \in \{a,b\}^+ \}$ $\quad N(a) = N(b)$

$S \to ab/ba$

$S \to asb$

$b \to bsa$

$S \to ss$

Dt.
Pg.

Q. Write the grammar over VP alphabet ab.
i) Set of all strings starting with ab
ii) Set of all strings ending with bba

Ans

i) $S \to abS$
$S \to a$
$S \to bSab$
$S \to \epsilon$
$S \to SS$

$S \to ab$
$S \to \epsilon$
$S \to SS$

$S \to abA$
$A \to \epsilon$
$A \to aA$
$A \to bA$

ii) $S \to Abba$
$A \to \epsilon$
$A \to aAa / aA$
$A \to bAb / bA$

$(a+b)^* bba$

Que Set of all binary strings containing exactly two 1's

$0^* 1 0^* 1 0^*$

$S \to A1A1A$
$A \to \epsilon$
$A \to 0A$

Dt. _____ B+

Pg. _____

ASSIGNMENT

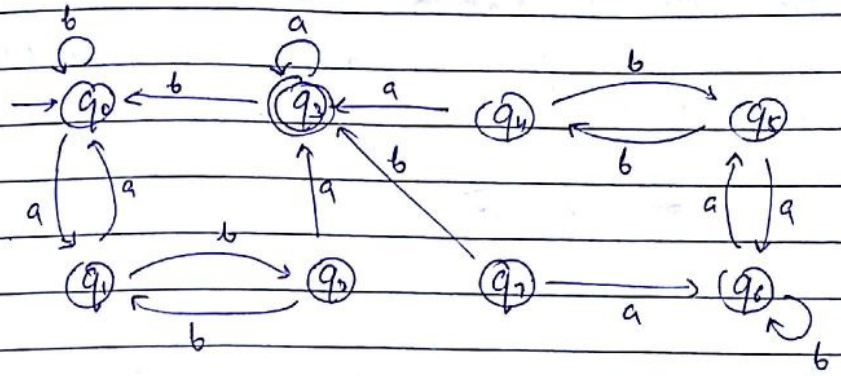
17 Feb

① Differentiate b/w NFA & DFA

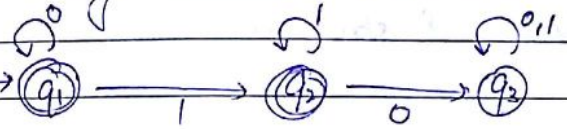
② What are the different closure properties of regular expression / grammar

③ Differentiate b/w Mealy & Moore Machine

④ Construct the minimum state ~~and~~ F.A. equivalent to given state diagram



⑤ Solve using Ardin's Theorem



⑥ Prove that the language $L = \{a^n b^n c^n ; n \geq 1\}$ is not regular

⑦ Design a DFA for set of all string of $\Sigma(a,b)$ which starts and ends with different symbol

⑧ Write a R.E. for set of all string having atleast two 1's and $\Sigma(0,1)$

⑨ Write a grammar for set of all string having atmost two 1's over $\Sigma\{0,1\}$.

Be Positive...

Left Most Derivation

In this we replace the left most variable by one of its production. It is denoted by

$\Rightarrow_{lm}$ / $\overrightarrow{lm}^*$

Right Most Derivation

In this we replace the right most variable by one of its production. It is denoted by

$\Rightarrow_{rm}$ / $\overrightarrow{rm}^*$

Parse Tree / Syntax tree / Derivation Tree

The strings generated by a CFG can be represented by a hierarchical structure

Characteristics of a Parse Tree

1. Every vertex of a parse tree has a label which is a non-terminal, terminal or epsylum
2. Root of the tree has a label 'S' [start symbol]
3. labels in an intermidiate vertex will be non-terminals
4. If a vertex 'A' has 'k' children with label $A_1, A_2 \dots A_k$ then there will be a production $A \to A_1, A_2 \dots A_k$ in a C.F.G.
5. A vertex small 'v' is a leaf if it contains a terminal or epsylum

Dt. _____ B+
Pg. _____

Que Let G be the gramar

$S \rightarrow 0B / 1A$

$A \rightarrow 0/0S/1AA$

$B \rightarrow 1/1S/0B$

Find the string 00110101 using

a) LMS b) RMS c) PT

a) LMS: $S \rightarrow 0B$

$00BB$

$001B$

$0011S$

$00110B$

$001101S$

$0011010B$

00110101

RMS: $S \rightarrow 0B$

$00BB$

$00B1$

$001S1$

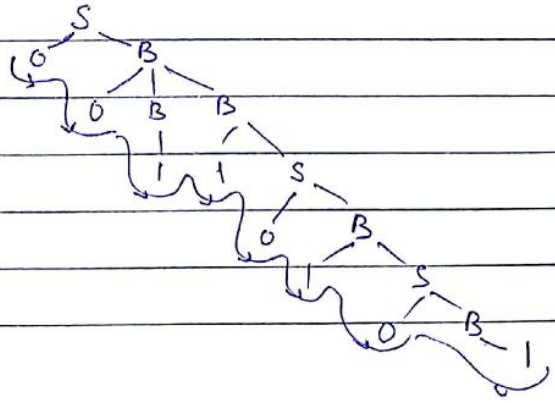
$0011A1$

$00110S1$

$001101A1$

$0011010B1$

PT:



Be Positive...

Qus $S \to a / aAS$

$A \to bS$

abaabaa by a) LMS b) RMS c) P.T.

a) LMS ⇒ $S \to aAS$

$abSS$

$abaS$

$abaaAS$

$abaabSS$

$abaabaS$

$abaabaa$

b) RMS ⇒ $S \to aAS$

$aAaAS$

$abSaAaAa$

$abaASaAabSa$

$abaAaaAabaa$

$abSabaa$

$abaabaa$

c) PT :

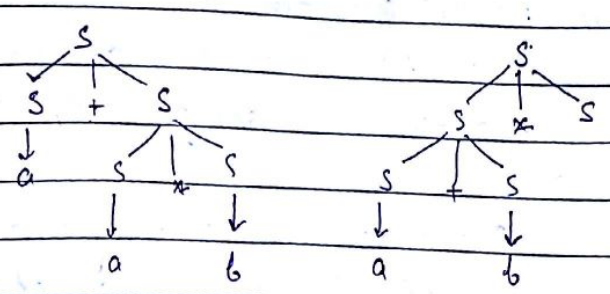
Dt. 7/2/18 B+
Pg.

Ambiguity in CFG

A CFG is called ambiguous if there exist two or more derivation tree

Eg. $S \rightarrow S+S \mid S*S \mid a \mid b$

$a+a*b$



dec.

	^	*, /	+, -
1. Precedance			
2. Associativity	R to L	L to R	L to R

[highest precedance operator should lie at the bottom of the tree]

Removal of Ambiguity

1. Precedance of the operators is respected
2. Sequence of identical operators can group either from left or right. We would see two diff. parse tree for $E+E+E$.

Since addition is associative so in order to avoid ambiguity we must follow a conventional approach that is L to R associativity

Reduction of CFG

Be Positive...

Dt.

Pg.

Reduction of CFG

Let L be a non-empty CFL (context free language) then it can be generated by CFG using following properties

① Eliminate useless symbols i.e. those variables or terminals that do not appear in any derivation of a terminal string from the start symbol.

② Eliminate ε-productions of the form $X \to \epsilon$

③ We must eliminate the unit productions of the form $X \to Y$ where X and Y are non terminals

Useless → Generating
↳ Reachable

$S \to AB$ — A & B are reachable

$A \to a$

↳ A is Generating

① Eliminate Useless Productions

A symbol Y in a CFG is useful if and only if

① $Y \Rightarrow w$ where $w \in L(G)$ and w in V_t^*,
i.e. Y leads to a string of ~~from~~ terminals
then Y is a generating symbol

② If there is a derivation $S \Rightarrow \alpha Y \beta \Rightarrow w$, $w \in L(G)$ for some α and β then Y is said to be reachable.

Dt. _____ B+

Pg. _____

Que $S \to AB/a$

$A \to b$

Sol. $S \to AB$ ✗

$S \to a$

$A \to b$ ✓

B is non generating symbol

⇒ $S \to a$

$A \to b$ ✗

A is not reachable from the start symbol (S)

⇒ $S \to a$

Que $S \to AB$ ✓

$S \to bX$

$A \to BAd$ ✗

$A \to BSX$ ✗

$A \to a$

$B \to aSB$ ✗

$B \to bBX$ ✗

$X \to SBD$ ✗

$X \to aBX$ ✗

$X \to ad$

⇒ $S \to bX$

$A \to a$ ✗

$X \to ad$

⇒ $S \to bX$

$X \to ad$

Que $S \to dC$

$S \to SB$

$A \to bSCa$ ✗

$B \to aSB/bBC$

$C \to abC/ad$

⇒ $S \to aC$

$S \to SB$ ✗

$B \to aSB/bBC$ ✗

$C \to aBC/ad$ ✗

⇒ $S \to aC$

$C \to ad$

Be Positive...

② Removal of ε Production

If $A \to \varepsilon$ is a production to be eliminated then we look for all productions whose right side contain A and replace each occurance of A in those production to obtain a non-ε production

Ques $S \to aA$, $A \to b/\varepsilon$ ⇒ ~~$S \to a/b/A/\varepsilon$~~, $A \to b$ — $S \to aA/a$, $A \to b$

Ques $S \to ABAC$, $A \to aA/\varepsilon$, $B \to bB/\varepsilon$, $C \to c$ ⇒ $S \to ABAC/BAC/ABC/BC/AAC/AC/C$, $A \to aA/a$, $B \to bB/b$, $C \to c$

③ Removal of Unit Production

A production of the form NT → 1 NT, eg: A → B is called unit production

Removal of Unit Production ⇒ while (there exist a unit production A → B)
{ Select a unit product^n A → B such that there exist a product^n B → α, where α is a terminal
for (every non-unit product^n B → α)
add product^n A → α to the grammar
eliminate A → B from the grammar
}

Be Positive

Dt. 8/2/16 B+
Pg.

Ques $S \to AB$
$A \to a$
$B \to C/b$
$C \to D$
$D \to E$
$E \to a$

Unit product'
$B \to C \Rightarrow B \to a$
$C \to D \to C \to a$
$D \to E \Rightarrow D \to a$

$\Rightarrow$

✓ $S \to AB$
✓ $A \to a$
✓ $B \to b$
$E \to a$
✓ $B \to a$
$C \to a$
$D \to a$

$\Rightarrow$

$S \to AB$
$A \to a$
$B \to a/b$

CHOMSKY NORMAL FORM

If a CFG has only production of the form
NT $\to$ string of exactly two NT
or NT $\to$ Single terminal

Eg: $A \to BC$
Eg: $E \to a$

Ques $S \to 1A / 0B$
$A \to 1AA / 0S / 0$
$B \to 0BB / 1$

$\Rightarrow$

$S \to 1A$
$S \to 0B$
$A \to 1AA$
$A \to 0S$
$A \to 0$ ✓
$B \to 0BB$
$B \to 1$ ✓

Ans Replace 1 by C_1 and 0 by C_0

$\to S \to C_1A$; ✓
$S \to C_0B$ ✓
$A \to C_1AA$
$A \to C_0S$ ✓
$A \to 0$ ✓
$B \to C_0BB$
$B \to 1$ ✓

Ans Replace AA by D_A BB by D_B

$S \to C_1A$ $\quad C_1 \to 1$
$S \to C_0B$ $\quad C_0 \to 0$
$A \to C_1D_A$ $\quad D_A \to AA$
$A \to C_0S$ $\quad D_B \to BB$
$A \to 0$
$B \to C_0D_B$
$B_1 \to 1$

Be Positive...

Dt.
Pg.

Qwl $S \to abSb \mid a \mid aAb$
$A \to bS \mid aAAb$

$S \to abSb$
$S \to a$ ✓
$S \to aAb$ ⇒
$A \to bS$
$A \to aAAb$

let $C_1 \to ab$
$S \to C_1Sb$
$S \to a$
$S \to C_1$

let $aA \to C_1$ let $B \to b$
$S \to abSb$

$S \to aBSB$
$S \to a$
$S \to aAB$
$A \to BS$
$A \to aAAB$

let $A_a \to a$
$S \to A_aBSB$
$S \to a$
$S \to A_aAB$
$A \to BS$
$A \to A_aAAB$

let $C_A \to A_aA$ $C_B \to A_aB$
$D_A \to SB$ $D_B \to AB$
$S \to C_BD_A$
$S \to a$
$S \to C_AB$
$A \to BS$
$A \to C_AD_B$

Greibach Normal Form
In this

12/2/18
Qwl Convert the grammar
$S \to AB$ ✓ $B \to SA \mid b$
$A \to BS \mid a$ into GNF (Greibach Normal Form

Be Positi

Dt. _____ B+
Pg. _____

Sol: Let us change the name of variables as A_1 A_2 for A_3 for S, A, B respectively

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 / a$

$A_3 \rightarrow A_1 A_2 / b$

2. Production must be in the form that RHS of a production must start with terminal or with higher numbered variable

$A_1 \rightarrow A_2 A_3$ —— ①

$A_2 \rightarrow A_3 A_1 / a$ —— ②

$A_3 \rightarrow b$ —— ③

as in required form

But $A_3 \rightarrow A_1 A_2$ is not

So in order to convert it,

put value of A_1 from ① in A_3

$A_3 \rightarrow A_2 A_3 A_2$

Again put the value of A_2 from ②

$A_3 \rightarrow A_3 A_1 A_3 A_2 / a A_3 A_2$

So now all eq^n are in form

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 / a$

$A_3 \rightarrow A_3 A_1 A_3 A_2 / c A_3 A_2 / b$ —— ④

Left recursion

If $A \rightarrow A\alpha / \beta$

then $A \rightarrow \beta A' / \beta$

$A' \rightarrow \alpha A' / \alpha$

Be Positive...

Dt.
Pg.

where A' is new variable, using it in (4)

$A_3 \rightarrow aA_3A_2B_3 / bB_3 / aA_3A_2 / b$

$B_3 \rightarrow A_1A_3A_2B_3 / A_1A_3A_2$

So,

$A_1 \rightarrow A_2A_3$ —— (5)

$A_2 \rightarrow A_3A_1 / a$ —— (6)

$A_3 \rightarrow aA_3A_2B_3 / bB_3 / aA_3A_2 / b$ —— (7)

$B_3 \rightarrow A_1A_3A_2B_3 / A_1A_3A_2$ —— (8)

Now all the A_3 productions start with a terminal, substitute the value of A_3, A_2 i.e. from (7) to (8)

$\therefore$ Eq^n (6) becomes

$A_2 \rightarrow aA_3A_2B_3A_1 / bB_3A_1 / aA_3A_2A_1 / bA_1 / a$

Put value of A_2 in A_1 i.e from (6) to (5)

$A_1 \rightarrow aA_3A_2B_3A_1A_3 / bB_3A_1A_3 / aA_3A_2A_1A_2 / bA_1A_2 / aA_2$

Also convert B_3 by putting value of A_1

$B_3 \rightarrow A_1A_3A_2B_3 / A_1A_3A_2$

$B_3 \rightarrow aA_3A_2B_3A_1A_3A_2A_3B_3 / bB_3A_1A_3A_2A_3B_3 / aA_3A_2A_1A_3A_2A_3$
$bA_1A_3A_2A_3B_3 / aA_3A_2A_3b_3 / aA_3A_2B_3A_1A_3A_2A_3 /$
$bB_3A_1A_3A_2A_3 / aA_2A_3A_1A_3A_2A_3 / bA_1A_3A_2A_3 /$
$aA_3A_2A_3$ are in GNF

Be Posit

Dt. _____ B+
Pg. _____

Greibach Normal Form

In this, every production must be of the form

$A \rightarrow aV/a$

where a: is a terminal which is exactly one
V: is a string of one or more variables (N.T.)

Ques Convert the Grammar

$S \rightarrow AB/BC$

$A \rightarrow aB/bA/a$

$B \rightarrow bB/cC/b$

$C \rightarrow \epsilon$ is in GNF

Sol^n: $S \rightarrow AB$ ✓ $\Rightarrow S \rightarrow aBB$, $S \rightarrow bAB$

$S \rightarrow BC$ ✗

$A \rightarrow aB$ $B \rightarrow bC$

$A \rightarrow bA$

$A \rightarrow a$

$B \rightarrow bB$

$B \rightarrow cC$

$B \rightarrow b$

$C \rightarrow \epsilon$

Imp

Q. • Equivalence of NFA and DFA (M.I.)

Q. • Kleene Closure

• Arden's Theorem *

• Closure properties of regular language

• Pumping lemma for regular languages *

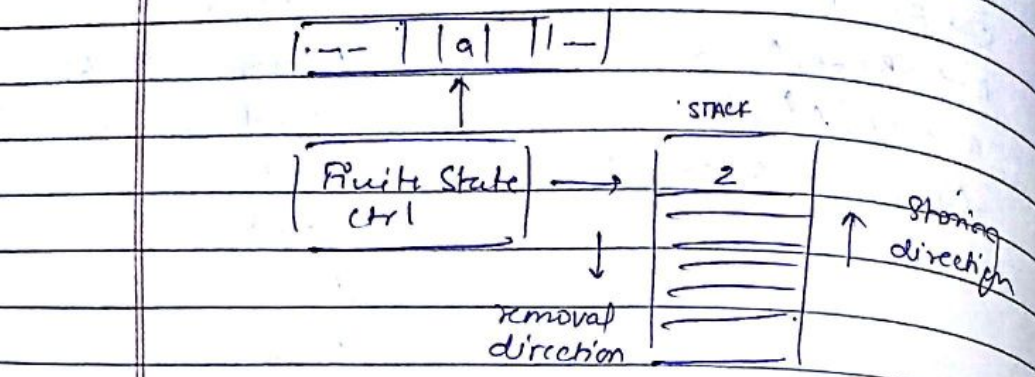
• Pumping lemma for CFL

Be Positive...

Dt. 15/4/19
Pg.

Model of PDA

(i) Read only tape
(ii) I/P alphabet
(iii) Finite state ctrl
(iv) Set of final states
(v) Initial state
(vi) Stak called as PDS (Push down State)



Formal definition

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

Γ → tape symbol

Z_0 → top most symbol on stack

State before transition — I/P to be inserted — I/P in stack

Push $\delta(q, a, b) = (q, a, b)$

Pop $\delta(q, 1, 1) = (q, \epsilon)$

Ques Let $A(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where

$Q = \{q_0, q_1, q_f\}$ $\quad \Sigma = \{a, b\}$

$\Gamma = \{a, Z_0\}$ $\quad F = \{q_f\}$

and δ is given by

Be Positive

Dt. _____ B+
Pg. _____

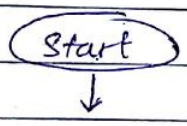
$\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, b, a) = (q_1, \epsilon)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$
$\delta(q_0, b, a) = (q_1, \epsilon)$ Process "aab"

Sol^n (q_0, aab, z_0)

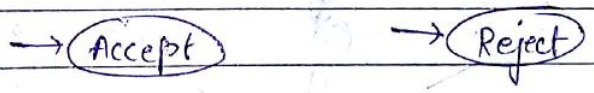
$\Rightarrow (q_0, ab, az_0)$ Rule 1.
$\Rightarrow (q_0, b, aaz_0)$ Rule 3
$\Rightarrow (q_0, \epsilon, az_0)$

Graphical Notations of PDA

Rule 1: Our start state that has only out edges with in edges denoted by



Rule 2: It consists of two half states,
1st : Accept
2nd Reject
having only in edges and no out edges

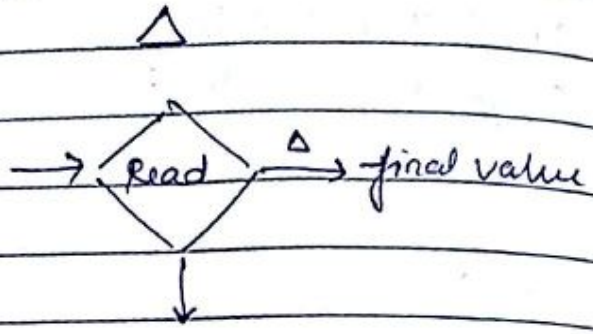


Rule 3: State that reads the next unused letter from the tape which may have half out

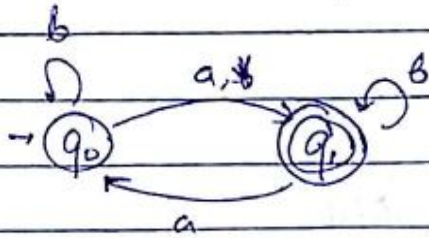
Be Positive...

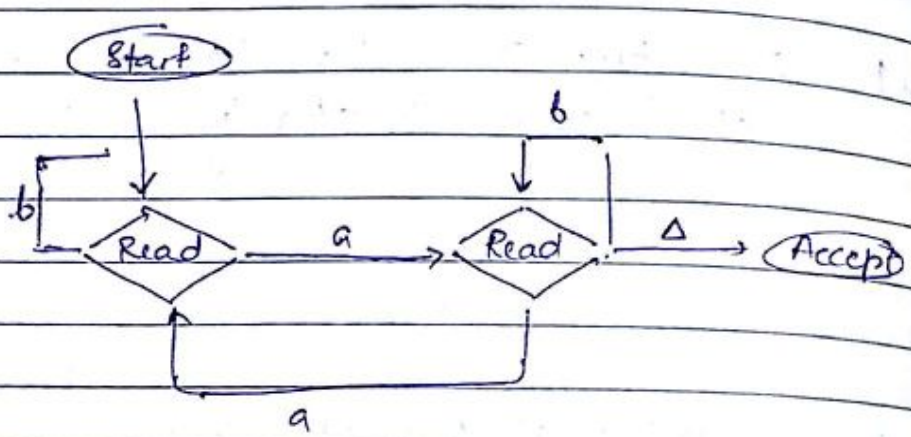
Dt.
Pg.

edges labled within letters from the alphabet and a blank character denoted by

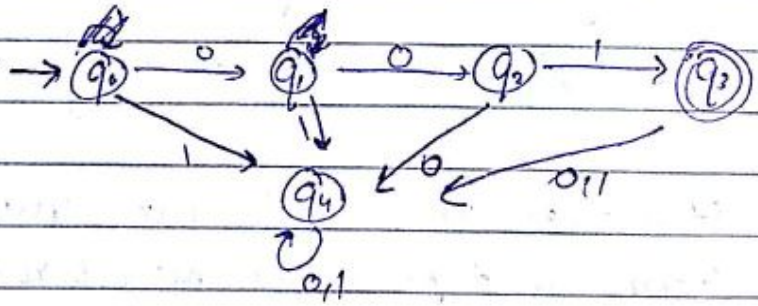


Qu1 Design a PDA that accepts only odd no. of 'a' over input alphabet {a, b}



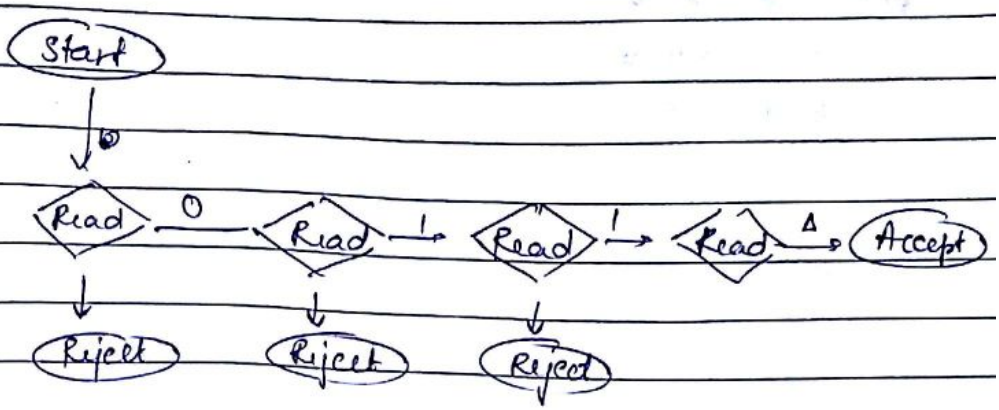


Qu2 Design a PDA for the language L = 001



Dt.
Pg.

B+



Que Construct a PDA equivalent to CFG
$S \to OBB$, $B \to OS \mid 1S \mid 0$,
Test for 010^4

Ans $R_1(q, A, S) = (q, \wedge, OBB)$
$R_2(q, \wedge, B) = (q, \wedge, OB) \mid (q, \wedge, 1S) \mid (q, \wedge, 0)$
$R_3(q, 0, 0) = (q, \wedge)$
$R_4(q, 1, 1) = (q, \wedge)$

$(q, 010^4, S)$
$(q, 010^4, OBB)$ ①
$(q, 10^4, BB)$ ②
$(q, 10^3, 1SB)$ ③
$(q, 0^3, SB)$ ④
$(q, 000, OBBB)$ ①
(q, e, e)

Be Positive...