Page No.
Date

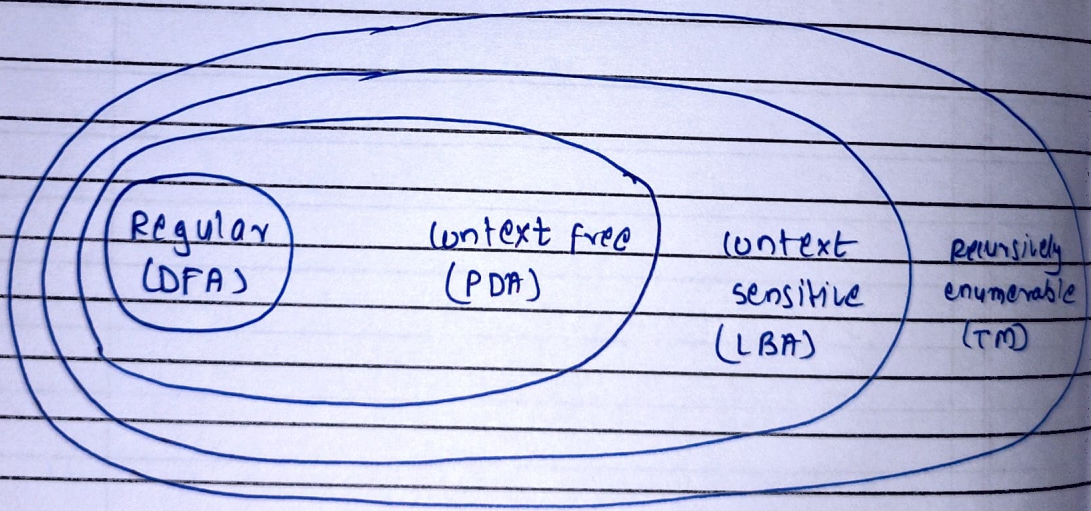
1.

- What is automata theory ?
  - Study of abstract computing devices or machines
  - Automaton - an abstract computing device
  - Alan Turing - A pioneer of automation theory

- Language & Grammer
  ① Language : Collection of sentences of finite length all constructed from alphabet of symbols.

  ② Grammer :- A grammer can be regarded as device the enumerates the sentences a language

Chomsky Hierarchy



Capacity ↓ increase

DFA - Diterministic Finite Automata
PDA - Push down Automata
LBA - Linear Bounded Automata
TM - Touring m/c

Alphabet

language → set of words, each word is comprises with alphabets

- A alphabet is a finite, non-empty set of symbols
- Σ → use to denote alphabet

ex.
Binary : $\Sigma = \{0,1\}$
All lower case letters : $\Sigma = \{a, b, c \cdots z\}$
Alphanumeric : $\Sigma = \{a\text{-}z, A\text{-}Z, 0\text{-}9\}$
DNA molecule letters : $\Sigma = \{a, c, g, t\}$

strings
- A string or word is a finite sequence of symbol chosen from Σ
- Empty string is ε (epsilon)

length of string is denoted by $|w|$ is equal to no. of (non-ε) in string

ex. (1) $x = 010100$ $|x| = 6$ (6 alphabet)
(2) $x = 01\varepsilon0\varepsilon1\varepsilon00\varepsilon$ $|x| = 6$

xy = concatenation of two string x and y

- Powers of an alphabet

Let Σ be an alphabet

Σ^k = the set of all strings of length k
$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$ (set of all binary string) (of any len)
$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

Σ^* string of length 0 is also present
Σ^+ string of all possible alphabet exept Σ/ϵ^0 epsilon

$\Sigma^1 = \{0,1\}$
$\Sigma^* = \{\epsilon, 0, 1, 00, 10, 11, 01 \ldots\}$ → all possible strings
$\Sigma^+ = \{0, 1, 00, 10, 11, 01 \ldots\}$

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001 \ldots 111\}$

* Language
$L \subseteq \Sigma^*$ L is subset of Σ^*

ϕ denotes the emply langvage

* The membership problem
Given a string $w \in \Sigma^*$ and a langvage L over Σ, decide whether or not $w \in L$.

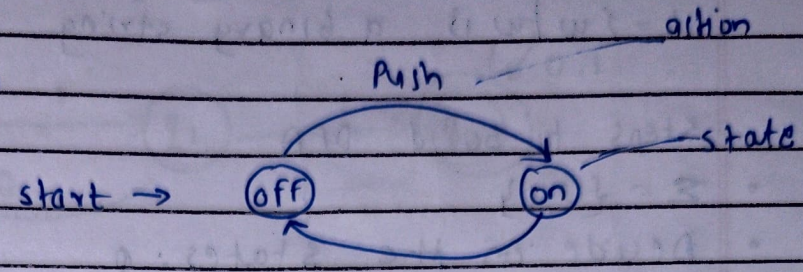
Example:
Let $w = 100011$

Q) Is $w \in$ the language of strings with equal no of 0's and 1's

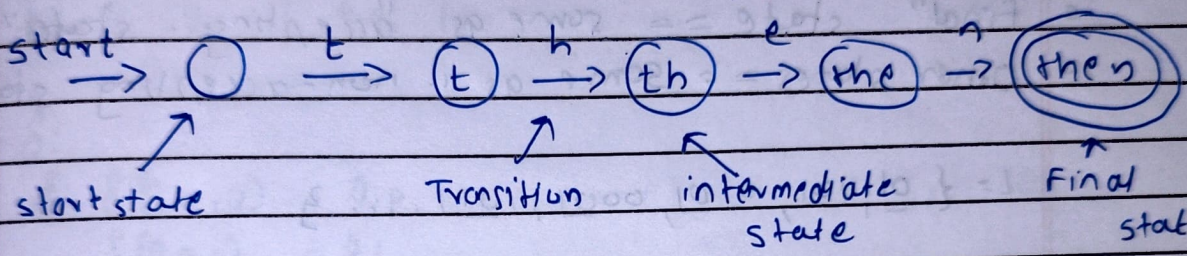
Page No.
Date

* Finite Automata

- on/off switch



- modeling recognition of the word "then"



- Grammers
- Regular expressions
  - eg unix style to capture city names such as "Palo Alto CA":
- [A-Z] [a-z]* ([ ] [A-Z] [a-z]*)* [ ] [A-Z] [A-Z]

start with letter

A string of other letters (possible empty)

other space delimited words (part of city name)

should end with 2 letter state code

Page No.
Date

Q. Build a DFA for following language
$L = \{w \mid w$ is a binary string containing 01 as substring$\}$

steps to build DFA
- $\Sigma = \{0, 1\}$
- Decide on the states: Q
- Designate start state and Final state (s)
- δ: Decide on the transitions

- "final" state == same as "accepting states"
- other states == same as "non-accepting states"

$L = \{01, 101, 0101, 0001, 1011 \ldots\}$

Q. $L = \{w \mid w$ is a bit string which contains the string 11$\}$

→ Clamping logic:-
A clamping circuit waits for a "1" input and turns on forever. However, to avoid clamping on spurious noise, we'll design a DFA that waits for two consecutive 1s in a row before clamping on.

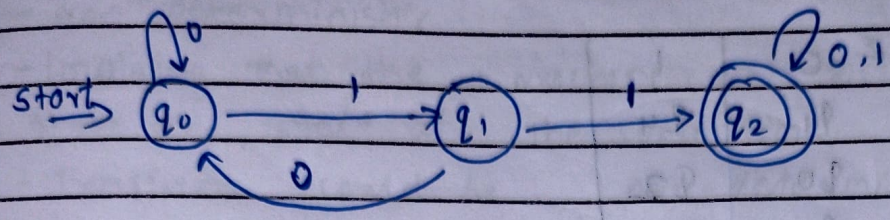
⇒ $L = \{11, 011, 110, 01101, 101110 \ldots\}$

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

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

Page No.
Date

δ



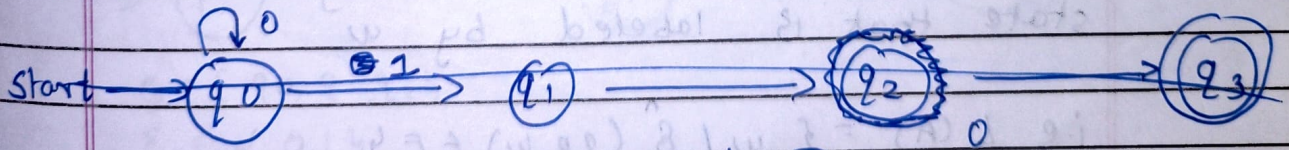
	δ	0	1
→	q_0	q_0	q_1
	q_1	q_0	q_2
*	q_2	q_2	q_2

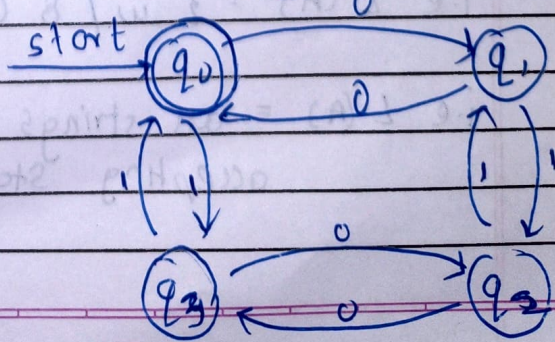
δ:

$\delta(q_0, 0) = q_0$
$\delta(q_0, 1) = q_1$
$\delta(q_1, 0) = q_0$
$\delta(q_1, 1) = q_2$
$\delta(q_2, 0) = q_2$
$\delta(q_2, 1) = q_2$

Q. L = {w | w is a binary string that has even number of 1's and even number of 0s}

=> L = {ε, 0011, 1100, 0101, 1010, 1001, 0110 ... }





Q = { q_0, q_1, q_2, q_3 }
Σ = { 0, 1 }

δ	Σ 0	1
q0	q1	q4
q1	q0	q3
q2	q3	q1
q3	q2	q0

~~Using~~

* Extension of transitions (δ) to path ($\hat{\delta}$)

using the input sequence $w = 10010$ $\quad a = 1$

$$\hat{\delta}(q_0, w_a) = \hat{\delta}(q_0, 10010) = q_1$$

$$\hat{\delta}(q_0, w_0) = \delta(\hat{\delta}(q_0, w)\, a)$$
$$= \delta(q', 1)$$
$$= q_2$$

* Language of DFA

A DFA A accepts string w if there is a path from q_0 to an accepting or (Final) state that is labeled by w

i.e $L(A) = \{ w \mid \hat{\delta}(q_0, w) \in F \}$

i.e $L(A)$ = all strings that leads to an accepting state from q_0

Page No.
Date

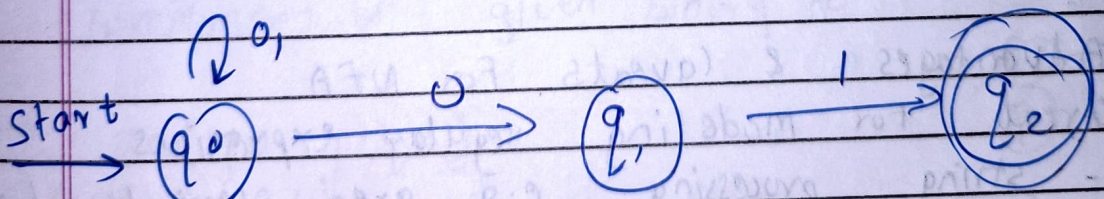
* Non - deterministic Finite Automata (NFA)
- Non deterministic
- Implying that the machine can exist in more than one state at the same time.
- Transitions could be non-deterministic

Q -> a Finite set of states
Σ -> Finite set of input symbols (alphabet)
q_0 -> a start state
F -> set of accepting states
δ -> a transition function, which is mapping between Q × Σ -> subset of Q

- An NFA is also defined by the 5 tuple $(Q, \Sigma, q_0, F, \delta)$

Q Build an NFA for the following language
$L = \{ w | w \text{ ends in } 01 \}$

-> $L = \{01, 101, 0001, 11101 \ldots\}$

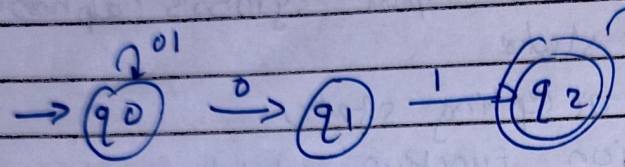


$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$F = \{q_2\}$

Page No.
Date

* Language of an NFA
An NFA accepts w if there exists at least one path from the start state to and accepting (or Final) state that is labeled by w

$$L(N) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset \}$$



$w = 101$

Path 1 = $q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$

Path 2 = $q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2$

$\delta(q_0, 101) = \{q_0, q_2\}$

$F = \{q_2\}$ $\{q_0, q_2\} \cap \{q_2\} = \{q_2\} \neq \emptyset$

Advantages & Caveats For NFA
- Great For modeling regular expressions
  - String processing e.g grep, lexical analyzer
- Could a non-deterministic state machine be implemented in pratice?
  - Probabilistic models could be viewed as extension.

Difference bet^n DFA and NFA

* Finite Automata
  - Moore m/c
  - Mealy mc

• Moore m/c

Six types $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

$\delta = Q \times \Sigma = Q$

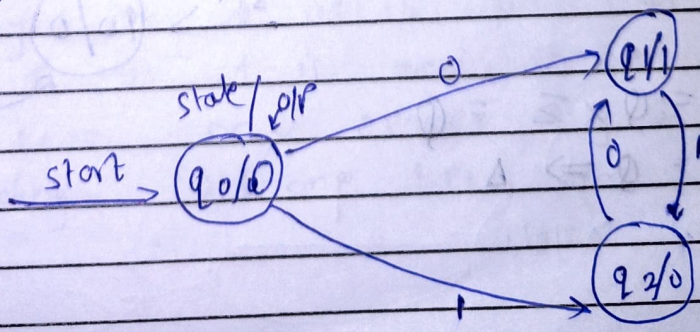
$\lambda = Q \Rightarrow \Delta$

Q = Finite set of state
Σ = Finite set of i/p symbol
Δ = Finite set of o/p symbol
δ = transition state
λ = machine f^n
q_0 = initial state

Q. Design a moore machine to generate one's compliment of given binary no.

q_1	0	$\to 1$
q_2	1	$\to 0$

→ $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\Delta = \{0, 1\}$



Date

$\delta : Q \times \Sigma \Rightarrow Q$

Q / Σ	0	1
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_1	q_2

$\lambda = Q \Rightarrow \Delta$

q_0	0
q_1	1
q_2	0

I/P (w) = 101 O/P = 010

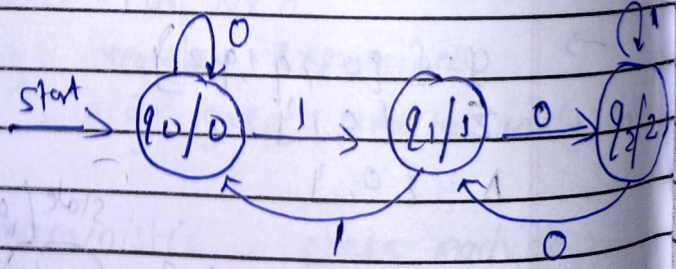
current state	I/P	Next state	O/P
q_0	1	q_2	0
q_2	0	q_1	1
q_1	1	q_2	0

Q. Construct Moore machine for finding remainder mod 3 for binary number.

→ $Q = \{q_0, q_1, q_2\}$

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

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



$\delta = Q \times \Sigma = Q$

$\lambda = Q \Rightarrow \Delta$

Page No.
Date

δ

Q \ Σ	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_2	q_1

$\lambda = Q \Rightarrow \Delta$

q_0	0
q_1	1
q_2	2

w = 111

Present state	I/P	Next state	O/P
q_0	1	q_1	1
q_1	1	q_0	0
q_0	1	q_1	1

Mealy m/c

six tuples ⇒ $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

$\delta : Q \times \Sigma \Rightarrow Q$

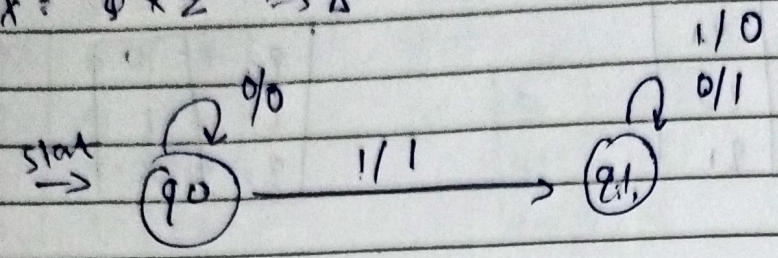
$\lambda : Q \times \Sigma \Rightarrow \Delta$

Q. Design a Mealy m/c to find 2's compliment

1. read the inputs from least significant bit (LSB)
2. keep binary no as it is until we read 1st 1
3. keep first 1 as it is and change remaining ones by zero and zeroes by 1.
4. Befor reading First one keep zero as it is.

$\delta : Q \times \Sigma \Rightarrow Q$

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



$w = 101$

Current state	Current I/P	Next state	O/P
q_0	1	q_1	1
q_1	0	q_2	0
q_2	1	q_1	1
q_1	1	q_1	0

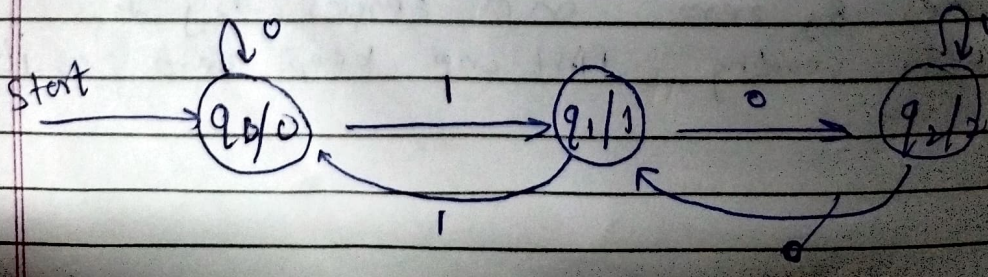
Moore m/c to equivalent Mealy m/c conversion

Moore m/c $(M_1) = (Q_E, \Sigma, \Delta, \delta, \lambda, q_0)$

then equivalent Mealy m/c $(M_2) = (Q, \Sigma, \Delta, \delta, \lambda', q_0$

$$\boxed{\lambda'(q, a) = \lambda(\delta(q, a))}$$

Q. Construct mealy m/c for given moore m/c Finding remainder mod 3 for binary number



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

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

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

$\lambda'(q_0, 0) = \lambda(\delta(q_0, 0))$

$= \lambda(q_0)$

$= 0$

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

$= 1$

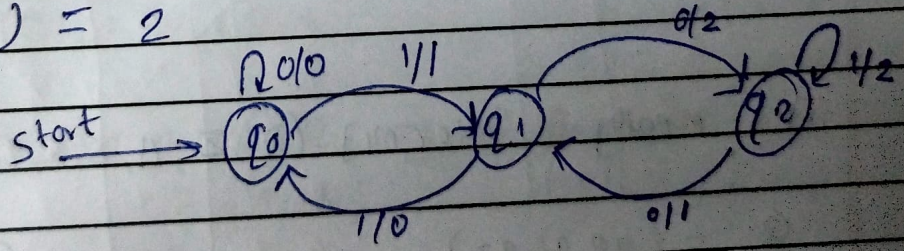
$\lambda'(q_1, 0) = 2$

$\lambda'(q_1, 1) = 0$

$\lambda'(q_2, 0) = 1$

$\lambda'(q_2, 1) = 2$

Q \ Σ	0	1
q_0	0	1
q_1	2	0
q_2	1	2



Mealy m/c to equivalent Moore m/c

If mealy m/c $(M_1) = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ ... (1)

where $\delta = Q \times \Sigma \Rightarrow Q$

$\lambda = Q \times \Sigma \Rightarrow \Delta$

the equivalent Moore m/c $(M_2) = (Q', \Sigma, \Delta, \delta', \lambda', [q_0, b])$

$Q' = [Q \times \Delta]$

$\delta'([q, b], a) = [\delta(q, a), \lambda(q, a)]$

$\lambda'([q, b]) = b$

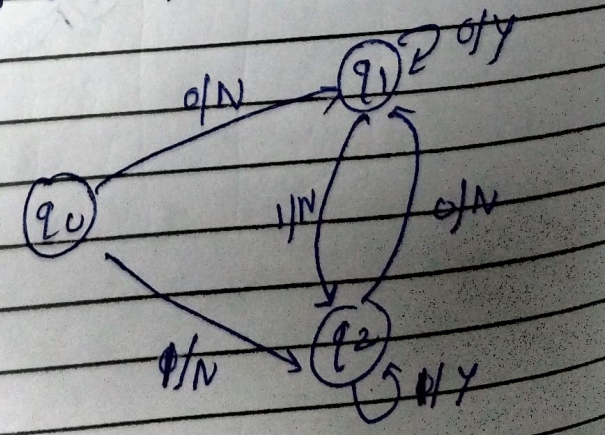
Q Construct a mealy M/C which will o/p Y (yes) if string ends in 00 or 11 otherwise o/p is N (no) also convert this mealy m/c into Moore M/C.

If Mealy M/C $[M_1] = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

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

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

$\Delta = \{Y, N\}$



δ

Q \ Σ	0	1
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_2	q_2

λ

Q \ Σ	0	1
q_0	N	N
q_1	Y	N
q_2	N	Y

equivalent Moore M/C

$Q' = [Q \times \Delta]$ $\quad Q = \{q_0, q_1, q_2\}$

$= \{[q_0, Y], [q_0, N]$ $\quad \Delta = \{Y, N\}$

$[q_1, Y], [q_1, N]$ $\quad \Sigma = \{0, 1\}$

$[q_2, Y], [q_2, N]$

$\delta'([q, b], a) = [\delta(q, a), \lambda(q, a)]$

1. $\delta([q_0, Y), 0) = [\delta(q_0, 0), \lambda(q_0, 0)]$
$= [q_1, N]$

2. $\delta([q_0, Y), 1) = [\delta(q_0, 1), \lambda(q_0, 1)]$
$= [q_2, N]$

3. $\delta[q_0, N), 0] = [\delta(q_0, 0), \lambda(q_0, 0)]$
$= [q_1, N]$

4. $\delta[(q_0, N), 1) = [\delta(q_0, \phi), \lambda(q_0, \phi)]$
$= [q_2, N]$

5. $\delta[(q_1, Y), 0] = [\delta(q_1, 0), \lambda(q_1, 0)]$
$= [q_1, Y]$

6. $\delta[(q_1, Y), 1] = [\delta(q_1, 1), \lambda(q_1, 1)]$
$= [q_2, N]$

7. $\delta[[q_1, N), 0] = [q_1, Y]$

8. $\delta[[q_1, N), 1] = [q_2, N]$

9. $\delta[(q_2, Y), 0] = [\delta(q_2, 0), \lambda(q_2, 0)]$
$= [q_1, N]$

10. $\delta(q_2, Y), 1] = [\delta(q_2, 1), \lambda(q_2, 1)]$
$= [q_2, Y]$

11. $\delta[[q_2, N), 0)] = [q_1, N]$

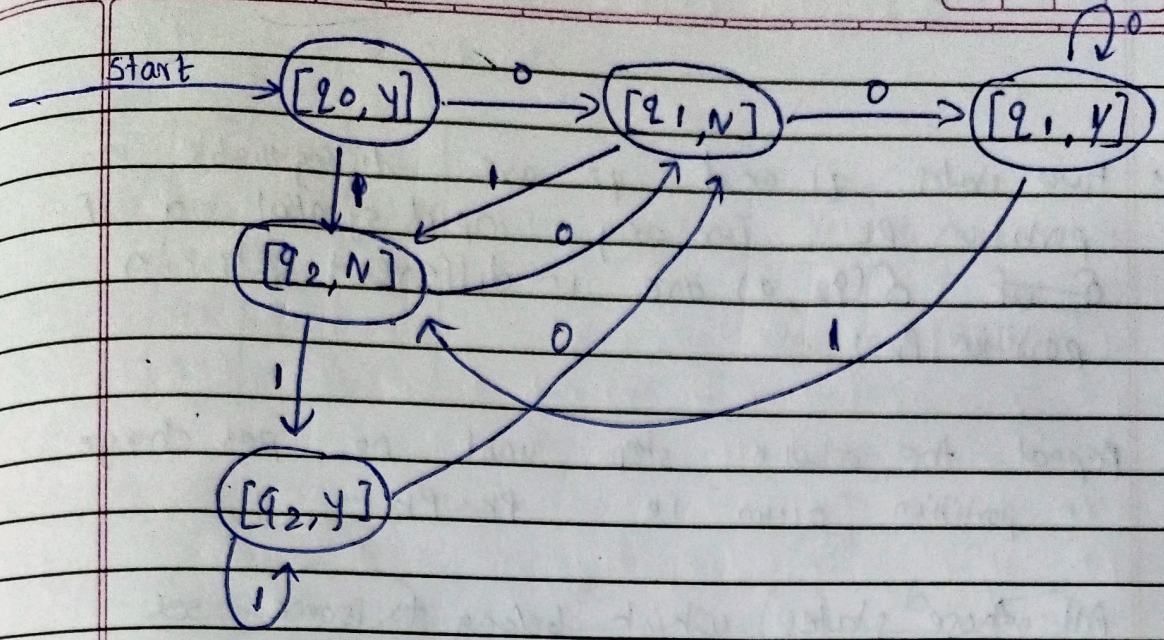
12. $\delta[[(q_2, N), 1)] = [q_2, Y]$

$\delta' : Q' \times \Sigma \Rightarrow Q'$

Q / Σ	0	1
$q_0 Y$	$q_1 N$	$q_2 N$
$q_0 N$	$q_1 N$	$q_2 N$
$q_1 Y$	$q_1 Y$	$q_2 N$
$q_1 N$	$q_1 Y$	$q_2 N$
$q_2 Y$	$q_1 N$	$q_2 Y$
$q_2 N$	$q_1 N$	$q_2 Y$

$\lambda'([q, b]) = b$

Q / Σ	0	1
q_0, Y	Y	Y
q_0, N	N	N
q_1, Y	Y	Y
q_1, N	N	N
q_2, Y	Y	Y
$q_2 N$	N	N



Minimization of DFA using Equivalence Thm

① Dead states — Q = { }

② Inaccessible states — P0 = { Final state, Non-final state }

③ Apply equivalence Thm — P1 =

- Divide set of states Q in two states such that one set contain all the non final state and other set contain all the Final state i.e P0
  P0 = { Final state, Non-final state }

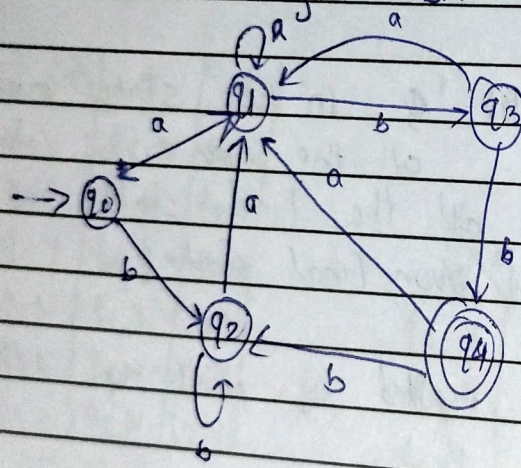
- Partition P_k is computed by partitioning different sets of P_{k-1}.
  In each set of P_{k-1} consider all the possible pair of states within each set and if two states are distinguishable partition the state into two diff sets in P_k.

- Two states q_1 and q_2 are distinguishable in partition P_k for any input symbol a if $\delta(q_2, a)$ are in different sets in partition P_{k-1}

- Repeat the above step until no change in partition occurs ie $P_k = P_{k-1}$

- All these states which belong to some set are equivalent states and they are merged to form a single state

Q. Minimize the given DFA



δ / Σ	a	b
q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
q_4	q_1	q_2

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

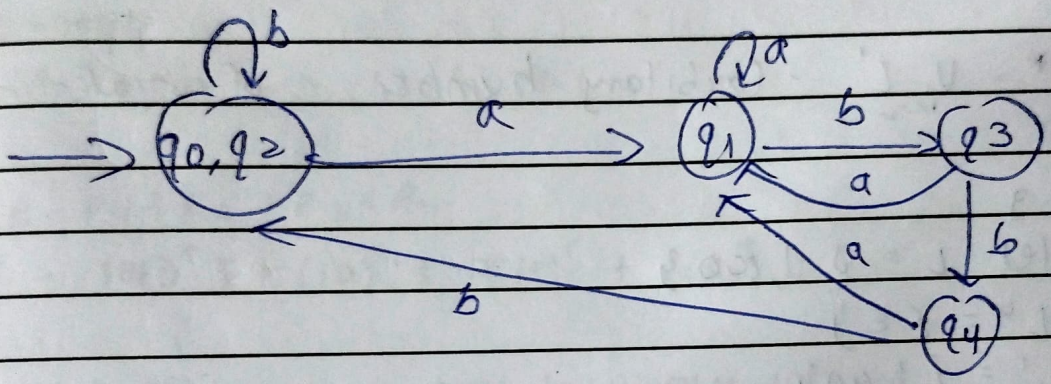
$P_0 = \{\{q_4\}, \{q_0, q_1, q_2, q_3\}\}$

$P_1 = \{(q_4), \{q_0, q_1, q_2\}, \{q_3\}\}$

$P_2 = \{\{q_4\}, \{q_3\}, \{q_1\}, \{q_0, q_2\}\}$

$P_3 = \{\{q_4\}, \{q_3\}, \{q_1\}, \{q_0, q_2\}\}$

Σ / δ	a	b
q_0q_2	q_1	q_2q_1
q_1	q_1	q_3
q_3	q_1	q_4
q_4	q_1	q_0q_2



Page No.
Date 20 | 8 | 24

2. Regular Expressions Vs Finite Automata

* Kleene Closure of a given language L

$L^0 = \{\varepsilon\}$

$L^1 = \{w \mid$ for some $w \in L\}$

$L^2 = \{w_1 w_2 \mid w_1 \in L, w_2 \in L$ (duplicate allowed)$\}$

$L^i = \{w_1 w_2 \ldots w_i \mid$ all w's chosen are $\in L$ (duplicates allowed)$\}$

$L^* = \bigcup_{i=0} L^i$ (arbitrary number of concatenations)

e.g.

(a) $L = \{1, 00\}$

$L^0 = \{\varepsilon\}$

$L^1 = \{1, 00\}$

$L^2 = \{11, 100, 001, 0000\}$

$L^3 = \{111, 1100, 1001, 10000, 000000, 00001, 00100\}$

$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots$

L^* is an infinite set iff $|L| \geq 1$ and $L \neq \{\varepsilon\}$

If $L = \{\varepsilon\}$ then $L^* = \{\varepsilon\}$

If $L = \phi$ then $L^* = \{\varepsilon\}$

Precedence of Operator

- Highest to lowest
  * Operator (star)
  . (concatenation)
  + operator

Example

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

$R = R_1 + R_2 + R_3 + R_4$

$R = (01)^* \Sigma + (10)^* + 0(10)^* + 1(01)^*$

$L = \{ 0, 1, 01, 10, 0101, 1010, 014, 101 \cdots \}$

$R_1 = (01)^*$

$L(R_1) = \{ \varepsilon, 01, 0101, 010101 \cdots \}$

$R_2 = (10)^*$

$L(R_2) = \{ \varepsilon, 10, 1010, 101010 \cdots \}$

$R_3 = 0(10)^+$

$L(R_3) = \{ 0, 010, 01010 \cdots \}$

$R_4 = 1(01)^*$

$L(R_4) = \{ 1, 101, 10101, 1010101 \cdots \}$

$r = (\epsilon + 1)(01)^*(\epsilon | 0)$

$R: 01^*11$

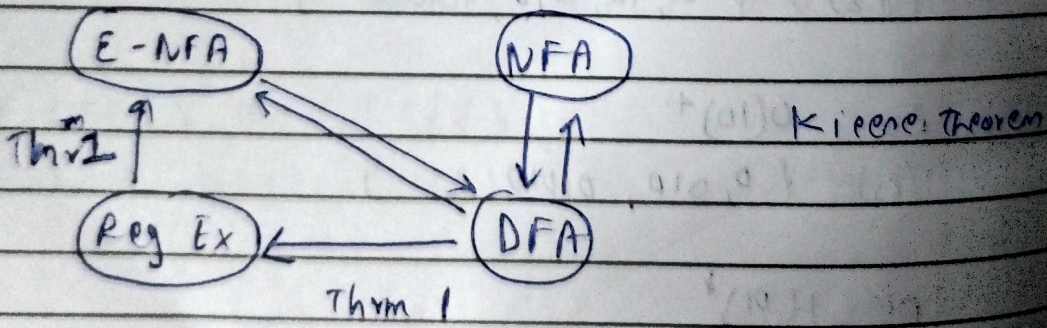
$L(r) = \{1, 0, 01, 011, 0111, 01111 \ldots\}$

Finite Automata (FA) & Regular Expressions (Reg Ex)

To show that they are interchangeable consider the following theorems.

- Theorem 1: For every DFA A there exists a regular expression R such that $L(R) = L(A)$

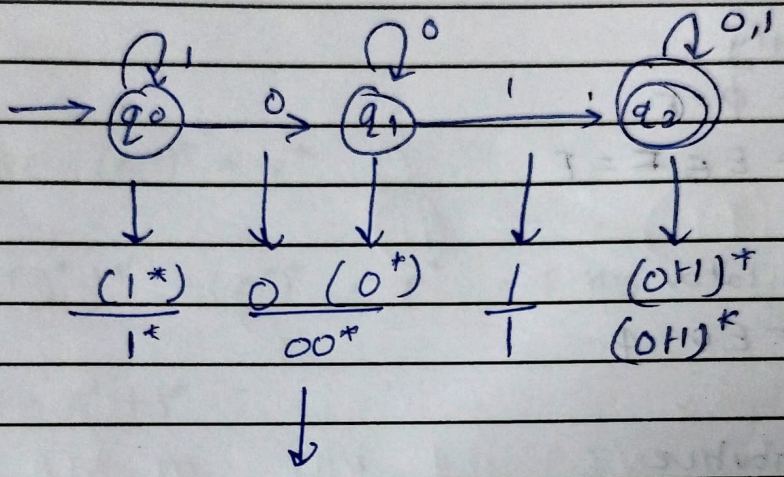
- Theorem 2: For every regular expressions R there exist an E-NFA E such that $L(E) = L(R)$



DFA to RE

Informally, trace all distinct paths (traversing cycles only from the start to each of the final states and enumerate all the expressions along the way

e.g.



$\Rightarrow 1^*00^* | (0+1)^*$

$R = 1^*00^* | (0+1)^*$

$L(R) = \varepsilon\ 0 \cdot \varepsilon \cdot 1 \cdot \varepsilon$

$= 01$

$L = \{01, 101, 1001, 0010, 0011\}$

$L(R) = 100 | \varepsilon$

$= 1001$

Algebraic laws of Regular Expressions

- Commutative union
  - $E + F = F + E$

- Associative
  $(E+F)+G = E+(F+G)$

- Identity
  $E + \phi = E$
  $E = E \in = E$

- Annihilator
  $\phi E = E \phi = \phi$

- Distributive
  $E(F+G) = EF + EG$
  $(E+G)E = FE + GE$

- Idempotent = $E + E = E$

- Involving Kleene closures
  - $(E^*)^* = E^*$
  - $\phi^* = \varepsilon$
  - $E^+ = E E^*$
  - $E? = \varepsilon + E$

Page No.
Date

Q. Check whether following regular expressions are equivalent or not

① $((R^*)^*)^* = R^*$

$R = ((R^*)^*)^*$

$S = R^*$

For R

subsitute $(R^*)^* = R^*$

$R = ((R^*)^*)^* = (R^*)^* = R^* = R.H.S$

② $(R+S)^* = R^* + S^*$

subsitute $R = a \quad S = b$

$L.H.S = (a+b)^* = \{ \varepsilon, a, b, ab, ba, aba, aab, abab, \ldots \}$

$R.H.S = a^* + b^* = \{ \varepsilon, a, aa, bb, aaa, bbb \ldots \}$

$\therefore L.H.S \neq R.H.S$

③ $(RS+R)^* RS = (RR^*S)^*$ $\quad LHS \neq RHS$

Replace $R = a \quad S = b$

$L.H.S = (RS+R)^* RS = (ab+a)^* ab = \{ ab, abab, aab, \ldots \}$

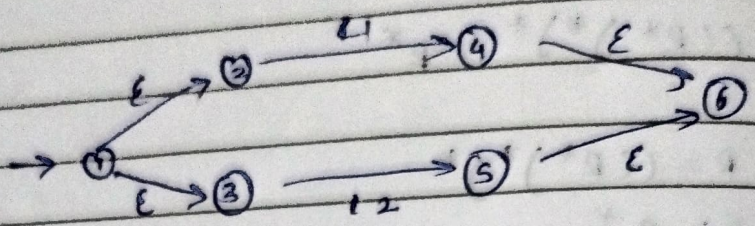
$R.H.S = (RR^*S)^* = (aa^*b)^* = \{ \varepsilon, ab, aab, aaab \ldots \}$

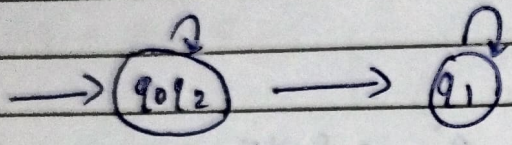
$L.H.S \neq R.H.S$

RE to ε-NFA (Thompson's construction)

① R_1, R_2

$R = R_1 + R_2$

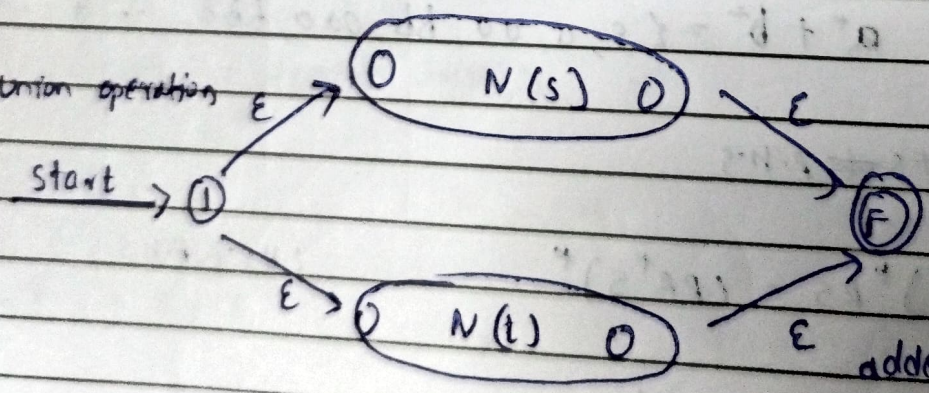




Induction :-

Suppose N(s) and N(t) or NFA, for regular expressions s and t respectively

Step a) For the regular expression s|t (s or t)

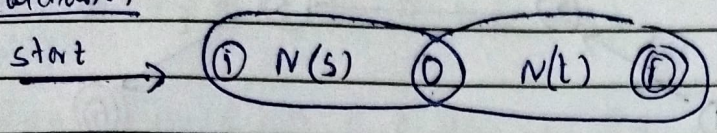


added new initial and newsi Final making transition of ε

Page No.
Date

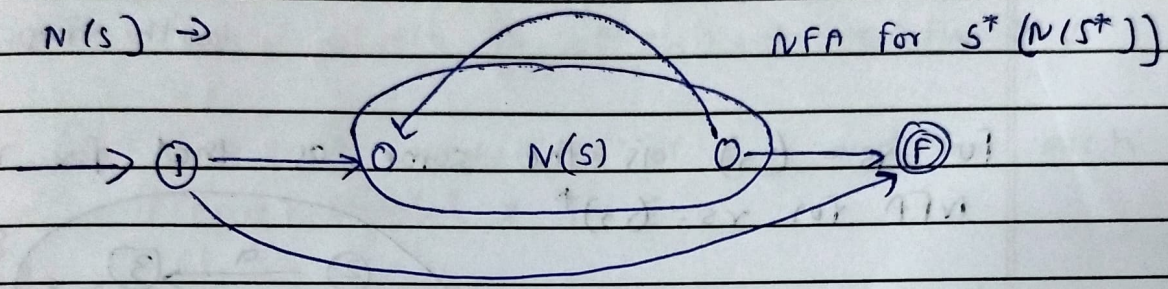
step b. For the regular expression st,

concatenation



merging final state of N(s) with initial state of N(t)

step c. For the regular expression, ~~st~~ s^*



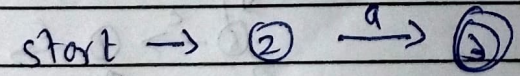
step d. Finally suppose $r = (s)$

Then $L(r) = L(s)$ and

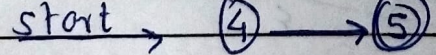
we can use the NFA N(s) as N(r)

Construct NFA for $r = (a|b)^* a$

For $r_1 = a$,

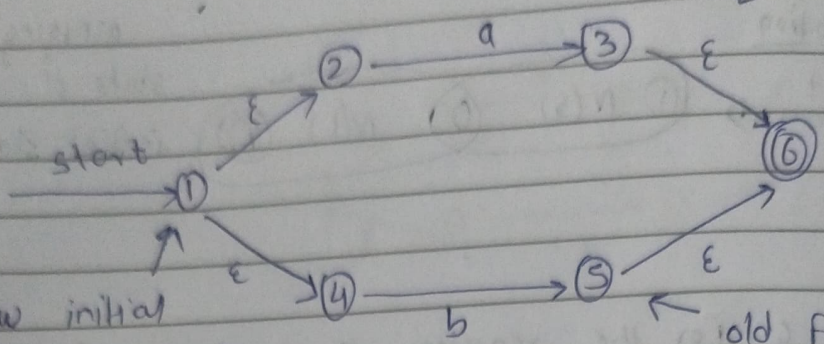


For $r_2 = b$,



Page No.
Date

For $r_1 = a|b$ $\quad s = a|b \cdot \rightarrow N(s)$

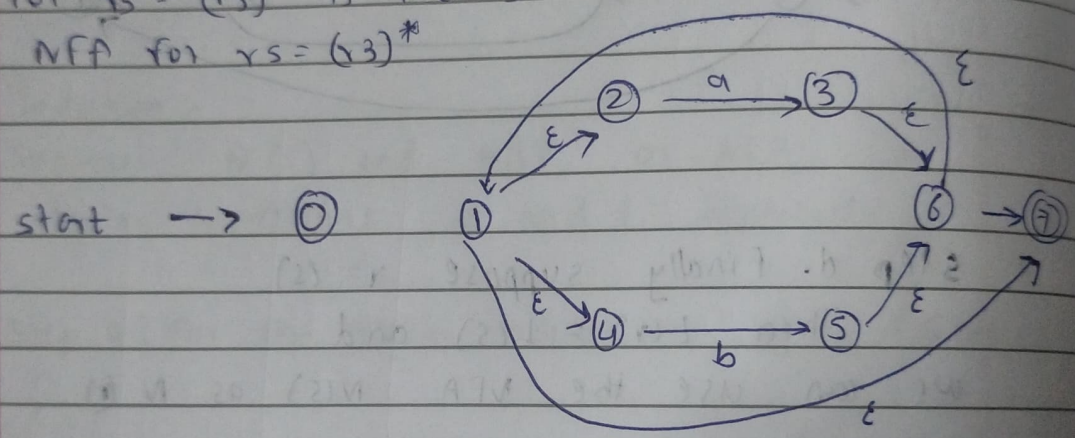


new initial state to enter initial add ε

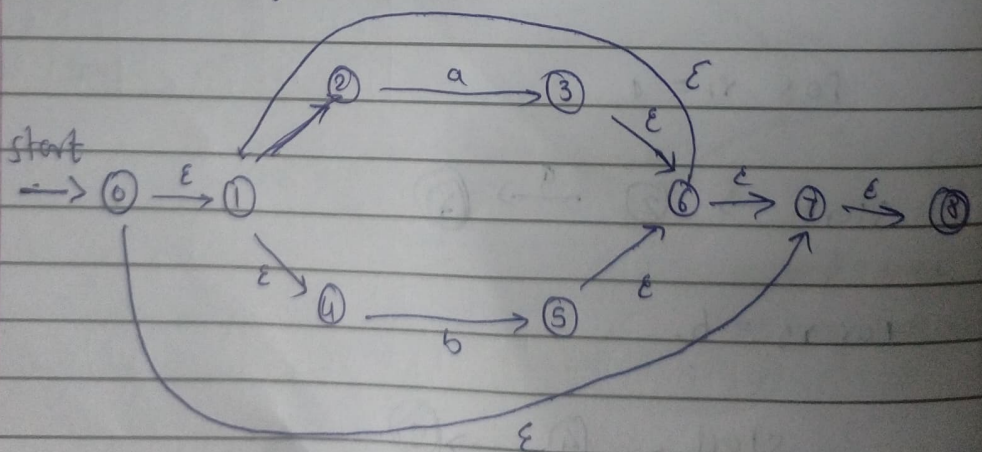
old final added to new final with transition of ε

for $r_5 = (r_3)^*$ is the same as that for r_3

NFA for $r_5 = (r_3)^*$



Finally NFA For $r = (a|b)^*a$



NFA to DFA by subset construction

- Let $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$
- Goal : Build $D = \{ Q_D, \Sigma, \delta_D, \{q_0\}, F_D \}$ s.t $L.D = L(N)$
- Construction:

  Q_D = all subsets of Q_N (i.e power sets)

  F_D = set of subsets S of Q_N s.t $S \cap F_N \neq \phi$

  δ_D : For each subset S of Q_N and for each input symbol a in Σ :

$$\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a)$$

$Q_N = \{q_0, q_1, q_2\}$ $\quad 2^3 = 8$

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

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

$$\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a)$$

$\delta_D([q_0, q_1], 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$

$= \{q_0, q_1\} \cup \phi$

$= \{q_0, q_1\}$

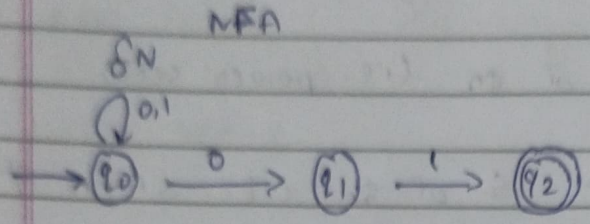
$\delta_D([q_0, q_1, q_2], 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= \{q_0\} \cup \{\phi\} \cup \{\phi\}$

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

Q.

NFA



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

$F_N = \{q_2\}$

δ_D

$F_D = \{[q_2], [q_0, q_2], [q_1, q_2], [q_0, q_1, q_2]\}$

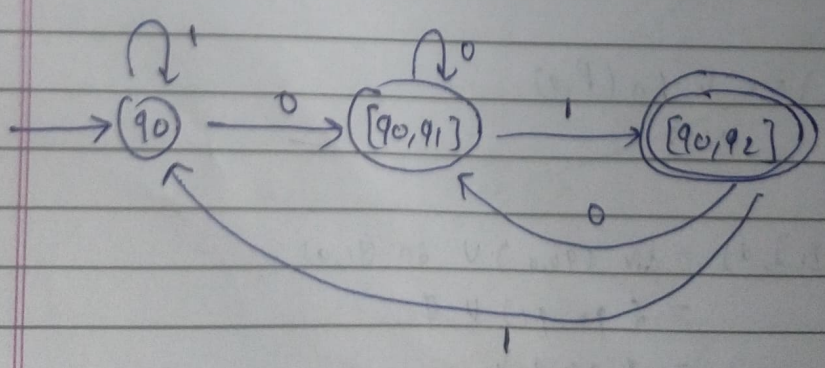
δ_D / Σ	0	1
$[\phi]$	$[\phi]$	$[\phi]$
→ ✓ $[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_1]$	$[\phi]$	$[q_2]$
* $[q_2]$	$[\phi]$	$[\phi]$
✓ $[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
✓ * $[q_0, q_2]$	$[q_0, q_1]$	$[q_0, q_2]$
* $[q_1, q_2]$	$[\phi]$	$[q_2]$
* $[q_0, q_1, q_2]$	$[q_0, q_1]$	$[q_0, q_2]$

$\delta_D([q_0, q_1], 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$

$= \{q_0, q_1\} \cup \{\phi\}$

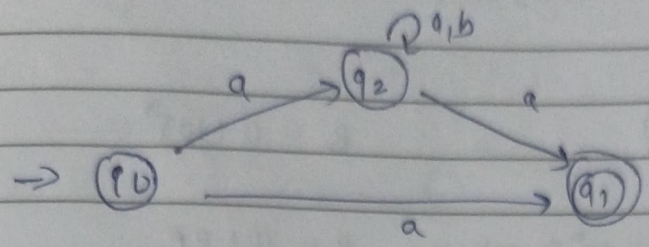
$= [q_0, q_1]$

DFA



Page No.
Date

Q. Convert NFA to DFA

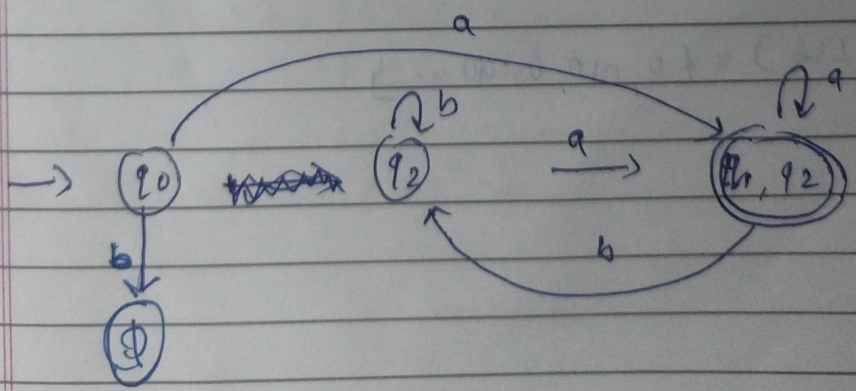


$Q_n = \{ q_0, q_2, q_1 \}$

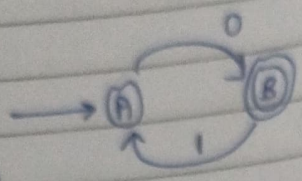
$F_n = \{ q_1 \}$

$F_D = \{ [q_1], [q_0, q_1], [q_1, q_2], [q_0, q_1, q_2] \}$

	Q_D \ Σ	a	b
	$[\phi]$	ϕ	ϕ
✓ →	$[q_0]$	$[q_1, q_2]$	$[\phi]$
	$[q_1]$	$[\phi]$	$[\phi]$
✓	$[q_2]$	$[q_1, q_2]$	$[q_2]$
	$[q_0, q_1]$	$[q_1, q_2]$	$[\phi]$
	$[q_0, q_2]$	$[q_1, q_2]$	$[q_2]$
✓	$[q_1, q_2]$	$[q_1, q_2]$	$[q_2]$
	$[q_0, q_1, q_2]$	$[q_1, q_2]$	$[q_2]$



Automata to RE using Arden's Theory



$B = 0 \cdot (10)^*$

incoming edge and its label (B,1)

writing eq^n for each step

Step1 : $A = \varepsilon + B \cdot 1$ — ①

$B = A \cdot 0$ ②

$R = Q + RP$

$\Downarrow$

$R = QP^*$

ε add in initial step only

Step2: substitute $A = \varepsilon + B \cdot 1$ in eq^n ②

$B = A \cdot 0$

$= (\varepsilon + B \cdot 1) 0$

$B = \varepsilon \cdot 0 + B \cdot 1 \cdot 0$

$= 0 + B \cdot 1 \cdot 0$

where $R = B$

$Q = 0$

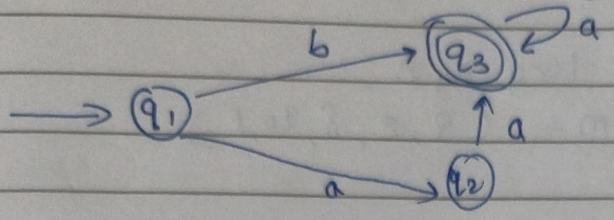
$P = 10$

Then $B = Q + R \cdot P$

$\therefore B = 0 \cdot (10)^*$

$L(B) = \{ 0, 010, 0 \cdot 1010 \dots \}$

Q. Convert following eq^n using Ardens theorem



Step ① : $q_1 = \varepsilon$ — ①

$q_2 = q_1 \cdot a$ — ②

$q_3 = q_1 \cdot b + q_2 \cdot a + q_3 \cdot a$ — ③

Step ② : simplify eq^n ②

$q_2 = q_1 \cdot a$

$q_2 = \varepsilon \cdot a$

$q_2 = a$ — ④

where $R = q_3$

$Q = b + aq$

$P = a$

substitute $q_1 = \varepsilon$ & $q_2 = a$ in eq. ③

$q_3 = \varepsilon \cdot b + q_2 \cdot a + q_3 \cdot a$

$q_3 = \varepsilon \cdot b + a \cdot a + q_3 \cdot a$

$q_3 = b + aa + q_3 a$

$R = Q + R \cdot P$

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

Pumping Lemma

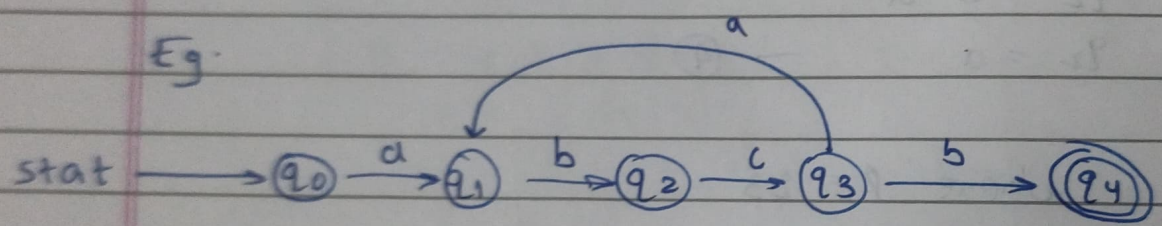
Let 'L' is a regular language accepted by FA $M = (Q, \Sigma, \delta, q_0, F)$

Let $w \in L$ and $|w| \geq n$

then 'w' can be written as xyz where

1. $|y| > 0$
2. $|xy| \leq n$
3. $xy^i z \in L \quad \forall i \geq 0$ $\quad xy^i z \in L$

Eg.



$Q = \{q_0, q_1, q_2, q_3, q_4\}$ $\quad n = 5$

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

$F = \{q_4\}$

$L = \{abcb, abcabcb, abcabcabcb, \ldots\}$

$w = abcabcb$

$|w| = 7 \geq n = 5$

$w = xyz$

$x = a$ $\quad |y| = 3 > 0$

$y = bca$ $\quad |xy| = 4 \leq n$

$z = bcb$

$xy^i z = a(bca)^i bcb = abcabcb$ Here $i=1$

$a(bca)^2 bcb = abcabcabcb$ Here $i=2$

Here $i=0$ $a(bca)^0 bcb = abcb$

Q. Show that language is not regular using pumping lemma.

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

→ Step ① Assume L is regular
L is accepting by FA having 'n' states

Step ② $w = a^n b^n$

$|w| \geq 2n \geq n$

write $w = xyz$
where $|y| > 0$ and $|xy| \leq n$

$w = a^n b^n = \underbrace{a^s \cdot a^r}_{xy} \cdot \underbrace{a^{n-s-r} \cdot b^n}_{z}$ $r > 0$

$x = a^s$
$y = a^r$
$z = a^{n-s-r} \cdot b^n$ $xy = a^s a^r = a^{s+r}$

step ③ $xy^iz \in L$, $i \geq 0$

put $i=0$

$xy^0z = xz = a^s \cdot a^{n-s-r} b^n$

$= a^{n-r} b^n \notin L$

∴ Given language is not a regular language.

put $i=2$

$xy^2z = xyyz = a^s a^r a^r a^{n-s-r} b^n$

$= a^{n+r} b^n$

→ ∴ This string not belong to language

Q. use pumping lemma ~~and~~ prove that Following language is not regular.

$L = \{0^i 1^{2i} \mid i \geq 0\}$

step ① :- Assume L is regular

L is accepting by FA having n states.

step ② $w = 0^n 1^{2n}$

$|w| = 3n \geq n$

write $w = xyz$

where $|y| > 0$ $|xy| \leq n$

$w = 0^n 1^{2n} = 0^s 0^t 0^{n-s-r} 1^{2n}$

$x = 0^s$

$y = 0^r$

$z = 0^{n-s-r} 1^{2n}$ $\quad xy = 0^r 0^s = 0^{s+r}$

Step ③ $xy^i z \in L \quad i \geq 0$

put $i=0$

$xy^0 z = xz = 0^s 0^{n-s-r} 1^{2n}$

$= 0^{n-r} 1^{2n} \notin L$

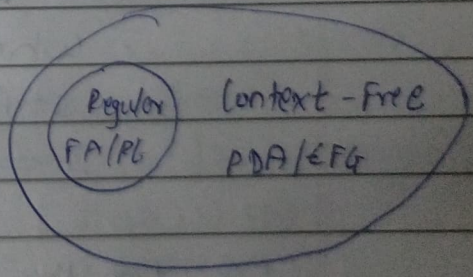
Given language is not regular language

---

3. Chapter

Context Free languages

- A language class larger than ~~the~~ the class of regular language.
- Supports natural, recursive notation called context free grammar
- Applications
  - Parse trees, compilers
  - XML



- Example

A palindrome is a word that reads identical from ends
e.g. madam, rediveder, malayalam. 010010010

Let $L = \{w \mid$ is a binary palindrome$\}$

Is L regular? No

Proof:
Let $w = 0^N 1^N$
By Pumping lemma w can be rewritten as xyz such that $xy^i z$ is

But $|xy| < N$ and $y \neq 0$
$\Rightarrow y = 0^+$
$\Rightarrow xy^+z$ will NOT be in L for $k=0$
$\Rightarrow$ contradiction.

Context Free Grammer

Def:
A context free Grammer $G = (V, T, P, S)$ where
V: set of variables or non-terminals
T: set of terminals (= alphabet Σ {ε})
P: set of productions, each of which is of the
form $V \Rightarrow \alpha_1 | \alpha_2 | \cdots$
. where each α_i is an arbitary string of variables and terminals.
S $\Rightarrow$ start variable.

CFG for the language binary palindrome
$G = (\{A\}, \{0,1\}, P, A)$
P: $A \Rightarrow 0A0 | 1A1 | 0 | 1 | \varepsilon$

More eg
- Parenthesis matching in code
- syntax checking
- In scenarios where there is a general need for:
  - matching a symbol with another symbol or
  - Matching a count of one symbol with that of another symbol or
  - recursively substituting one symbol with a string of other symbols.

Eg-2
Language of balanced paranthesis
eg () (() ()) (()) ....
CFG?

G :
S => (S) | SS | ε

V : {S}
T : { (,), ε }
P : { S → (S) | SS | ε }
s : {S}

w = ((()))()()
S => (S)
= (SS) (S → SS)
= (SSS) (∵ S → SS)
=> ((S)SS) (S → (S))
=> (((S))SS) (S → (S))
=> ((((S))SS)
=> ((((ε))))(S)(S)) (S → ε)
=> ((((ε))))()()

eg A grammar For $L = \{0^m 1^n \mid m \geq n\}$

CFG?

G:
$S \Rightarrow 0S1 \mid A$
$A \Rightarrow 0A \mid \varepsilon$

$\Rightarrow$ $S \rightarrow 0S1 \mid A$
$A \rightarrow 0A \mid \varepsilon$
$V = \{S, A\}$
$T = \{0, 1, \varepsilon\}$
$S = \{S\}$

$w = 00000111$

$S \rightarrow 0S1$
$\rightarrow 00S11$
$\rightarrow 000S111$
$000A111$
$0000A111$
$00000A1111$
$0000\varepsilon 111$

$\because$ $S \rightarrow 0S1$
$S \rightarrow 0S1$
$S \rightarrow 0S1$
$S \rightarrow A$
$A \rightarrow 0A$
$A \rightarrow 0A$
$A \rightarrow \varepsilon$

$w = 0001$

$S \rightarrow 0S1$
$0A1$
$00A1$
$000A1$
0001

$S \rightarrow 0S1$
$S \rightarrow A$
$A \rightarrow 0A$
$A \rightarrow 0A$
$A \rightarrow \varepsilon$

Page No.
Date

* Structure of a production

* CFG convention
  - Terminal symbols <== a, b, c
  - Non-terminal symbols => A, B, C
  - Terminal or non-terminals symbol <== X, Y, Z
  - Terminal strings <== w, x, y, z
  - Arbitary strings of terminals and non terminals α, β, δ

* String membership

w = 01110

A => 0A0 | 1A1 | 0 | 1 | ε

- Simple expression

G = (V, T, P, S)

V = {E, F}

T = {0, 1, a, b, +, *, (,)}

S = {E}

w = (a+b) * 10

P:

E ==> E+E | E*E | (E) | F

F ==> aF | bF | 0F | 1F | a | b | 0 | 1

#left

E => E * E

=> (E) * E

=> (E + E) * E

=> (F + E) * E

(a+F) * E

(a+E) * E

(a+b) * E

(a+b) x F

(a+b) * 1F
(a+b) * 10

Right w = (a+b) x 1

E ⇒ E x E

E x F	E → F
E x 1	F → 1
(E) x 1	E → (E)
(E+E) x 1	E → E+E
(E+F) x 1	E → F
(E+b) x 1	F → b
(F+b) x 1	E → F
(a+b) x 1	F → a

*

Page No.
Date

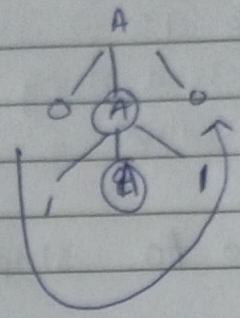
$P = \{A \rightarrow 0A0 \mid 1A1 \mid 0 \mid 1 \mid \varepsilon\}$

① $w = 0110$

$T = \{0, 1, \varepsilon\}$

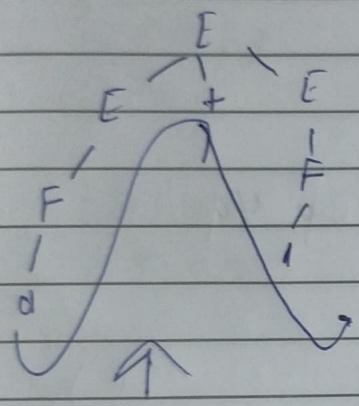
$V = \{A\}$

$A \Rightarrow 0A0$
$\Rightarrow 01A10$
$\Rightarrow 01\varepsilon10$
$\Rightarrow 0110$



$01\varepsilon10$
$= 0110$

②



Parse tree for a+1

G :

$E \Rightarrow E + E \mid E * E \mid (E) \mid F$
$F \Rightarrow aF \mid bF \mid 0F \mid 1F \mid 0 \mid 1 \mid a \mid b$

$w = a+1$

$E \Rightarrow E+E$
$\Rightarrow F+E$
$\Rightarrow a+E$
$\Rightarrow a+1$

$E \Rightarrow E+E$
$F+E$
$F+F$
$a+F$
$a+1$

$E \rightarrow E+E$
$\Rightarrow F+E$
$\Rightarrow a+E$
$\Rightarrow a+F$
$\Rightarrow a+1$

Page No.
Date

Ambiguity in CFGs

The CFG is said to be ambiguous if there exists a string which has more than one left most derivation.

How to simplify 'CFGs'?

$P = \{A \to 0A0 \mid 1A1 \mid 0 \mid 1 \mid \epsilon\}$

$V = \{A\}$

$T = \{0, 1, \epsilon\}$

$S = \{A\}$

$L = \{0, 1, 01010, 00, \ldots\}$

- 3 ways to simplify / clear a CFG

(clean)

- Eliminate useless symbols.

(Simplify)

- Eliminate ϵ-productions $A \Rightarrow \epsilon$
- Eliminate unit productions

- Eliminating useless symbols.

A symbol x is reachable if there exist $S \xrightarrow{*} \alpha x \beta$

A symbol X is generating if there exists $x \xrightarrow{*} w$, for some $w \in T^*$

for a symbol X to be 'useful' it has to be both reachable and generating

$S \xrightarrow{*} \alpha X \beta \xrightarrow{*} w$, for some $w' \in T^*$
reachable generating

Algorithm to detect useless symbols

First, eliminate all symbols that are not generating
Next eliminate all symbols that are not reachable

e.g. $S \xrightarrow{P} AB \mid a$
$A \xrightarrow{P} b$

$V = \{S, A, B\}$
$T = \{a, b\}$