10/02/20

Theory of Computation

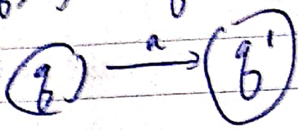
Transition system
Automata

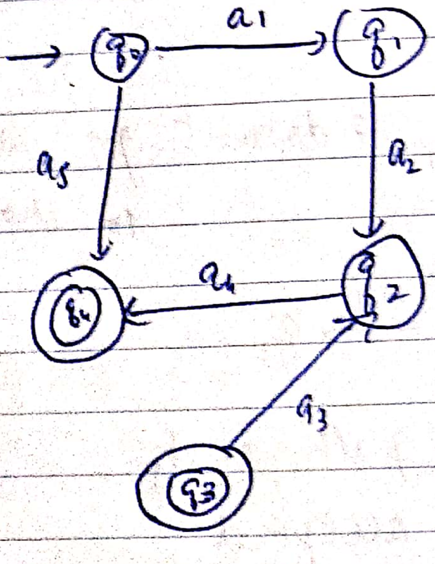
next transition func

$M = (Q, \Sigma, \delta, q_0, F) \rightarrow$ Automata
↳ can be represented graphically

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

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





Eg of automata

$w \in \Sigma^*$

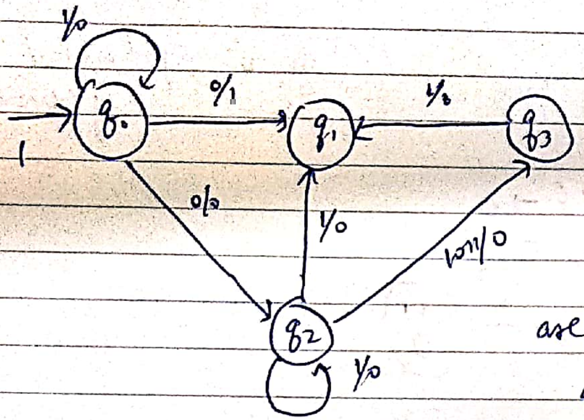
w is accepted by automata M iff

P.T.O

i) ∃ a path originates from the initial state, goes along the arrows & reaches some state.

ii) the path value obtained by concatenation (happen left to right) of edge labels of the path is w.

Q.1)



w = 101011

w = 111010

are above inputs accepted by automata?

A.1)

$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1$ doesn't go anywhere so choose q_2

$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_2 \xrightarrow{1011} q_3$

∴ path exists
hence accepted

w = 111010

$q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_2 \xrightarrow{1} q_2 \xrightarrow{0}$

path doesn't exist
hence not accepted

<u>Properties:-</u> (Imp)

1) $\delta(q, \Lambda) = q$

2) $\delta(q, aw) = \delta(\delta(q, a), w)$

and $\delta(q, wa) = \delta(\delta(q, w), a)$

$\boxed{***}$ $\forall\ w \in \Sigma^*$ & $a \in \Sigma$

using above, P.T. for any transition func. δ & for any $x, y \in \Sigma^*$

$\delta(q, xy) = \delta(\delta(q, x), y)$

<u>Proof</u>: Prove by PMI on length of the string

suppose length $= 1$, $|y| = 1$, $y = a$ (say)

so, $\delta(q, xy) = \delta(q, xa)$

$= \delta(\delta(q, x), a)$

PTO

$\therefore$ result is true for any string y of length $|y| \le m$

Let length of $y = m+1$

$y = y_1 a$, $|y_1| = m$, $a \in \Sigma$ $\qquad x_1 = xy_1$

$\delta(q, xy) = \delta(q, x_1 a)$

$= \delta(\delta(q, x_1), a)$

$= \delta(\delta(q, xy_1), a)$

LHS $= \delta(\delta(\delta(q, x), y_1), a)$

RHS: $\delta(\delta(q, x), y_1 a)$

$= \delta(\delta(\delta(q, x), y_1), a) = $ LHS

Hence proved

$\boxed{A18}$

Q2) P.T If $\delta(q, x) = \delta(q, y)$ then,

$\delta(q, xz) = \delta(q, yz) \quad \forall\, z \in \Sigma^*$

Proof: $\delta(q, xz) = \delta(\delta(q, x), z) = \delta(\delta(q, y), z)$

$= \delta(q, yz)$

$w \in \Sigma^*$ is accepted by the automata M iff

$\delta(q_0, w) = q$, $q \in F$

Q1) $Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$, $q_0 \to$ initial state

$F = \{q_0\}$ Give the entire sequence of states for the input string 110101

state	input 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

show all steps in exam

$\delta(q_0, 110101) = \delta(\delta(q_0, 1), 10101)$

$= \delta(q_1, 10101) = \delta(\delta(q_1, 1), 0101)$

$= \delta(q_0, 0101) = \delta(\delta(q_0, 0), 101)$

$= \delta(q_2, 101) = \delta(\delta(q_2, 1), 01)$

$= \delta(q_3, 01) = \underbrace{\delta(\delta(q_3, 0), 1)}_{\text{analog}} = \underbrace{\delta(q_1, 1)}_{\text{Hard}} = \underline{q_0}$

$q_0 q_1 q_0 q_2 q_3 q_1 q_0$

for all transitions

(DFA)

Deterministic Automata - You can determine the next state

(NDFA)

Non-Deterministic Automata - Cannot determine next state, you have more than one choice

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

$\delta : Q \times \Sigma \rightarrow 2^Q$

$\delta(q, a) = \{q_1, q_2, \ldots, q_n\}$

basis of || computation

$w \in \Sigma^*$ is accepted by an NDFA M iff $\delta(q_0, w)$ contains an element of F

Every DFA is an NDFA (put DFA's final state into singleton, it becomes NDFA

& converse is not true

We can construct DFA from NDFA s.t. it accepts all strings accepted by NDFA

→ Equivalence of DFA & NDFA :-

Thm: For every NDFA, ∃ a DFA which simulates the behaviour of NDFA

Alternatively,

If L is the set accepted by NDFA then L is also accepted by DFA.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NDFA accepting L

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

(i) $Q' = 2^Q$ (any state in Q' is denoted by $[q_1, q_2 \ldots q_j]$ where $q_1, q_2, \ldots q_j \in Q$)

(ii) $q_0' = [q_0]$

(iii) F' is the set of all subsets of Q containing an element of F.

(iv) $\delta'([q_1, q_2 \cdots q_j], a)$

$= \delta(q_1, a) \cup \delta(q_2, a) \cup \cdots \cup \delta(q_j, a)$

$\delta'([q_1, q_2 \ldots q_j], a) = [p_1, p_2 \ldots p_j]$ iff $\delta(\{q_1, q_2 \ldots q_j\}, a) = [p_1, p_2 \ldots p_j]$

TST $L = T(M')$ if $x \in T(M) = L$ then $x \in T(M')$

TST $\delta(q_0, x) = \{q_1, q_2 \ldots q_j\}$ iff $\delta'(q_0', x) = [q_1, q_2, \ldots q_j]$

$\forall x \in \Sigma^*$

prove by PMI on $|x|$

If $\delta(q_0, x) = \{q_1, q_2 \ldots q_j\}$ then Show that

$\delta'(q_0', x) = [q_1, q_2 \ldots q_j]$

PMI on $|x|$ If $|x| = 0$, $x = \Lambda$

$\delta(q_0, \Lambda) = q_0$

$\delta'(q_0', \Lambda) = q_0' = [q_0']$

Let the result is true for any string of length $\leq m$

Let $|x| = m+1$, $x = ya$, $|y| = m$, $a \in \Sigma$

$\delta'(q_0, y) = \{p_1, p_2, \ldots p_j\}$, $\delta(q_0, ya) = \{r_1, r_2, \ldots, r_k\}$

$\delta_0'(q_0', x) = \delta'(q_0', ya) = \delta'(\delta'(q_0', y), a)$

$= \delta'([p_1, p_2 \ldots p_j], a)$
$= [r_1, r_2, \ldots r_k]$

Prove the rest, $L = T(M')$ if $x \in T(M) = L$ then $x \in T(M')$

$x \in L = T(M)$ $\delta(q_0, x)$ reaches to some final state

$\delta'(q_0', x)$

13/01/20

Q:1) Construct a DFA equivalent to NDFA, $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$, where δ is defined by the state table:

state/Σ	0	1
$\to q_0$	q_0	q_1
q_1	q_1	$q_0 q_1$

$Q' = \{\phi, [q_0], [q_1], [q_0 q_1]\}$

$\Sigma = \{0, 1\}$, $q_0' = \{q_0\}$, $F' = \{[q_0], [q_0, q_1]\}$

State/Σ	0	1
$\to [q_0]$	$[q_0]$	$[q_1]$
$[q_1]$	$[q_1]$	$[q_0 q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

represents initial state

represents final state

$\to \delta'([q_0, q_1], 0)$

$\delta'([q_0, q_1], 0) = \delta([q_0, 0]) \cup \delta([q_1, 0])$

Q.2) Suppose NDFA $M = (\{q_0, q_1, q_2, q_3\}, \{a,b\}, \delta, q_0, \{q_3\})$

State/Σ	a	b
$\rightarrow q_0$	$q_0 q_1$	q_0
q_1	q_2	q_1
q_2	q_3	q_3
q_3	—	q_2

Ans 2) DFA,

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

$Q' = 2^Q = 16$ elements

$= \{\phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_3\}, \{q_0, q_1\}, \dots\}$

$\Sigma = \{a, b\}, \quad q_0' = [q_0]$

F' = all states containing q_3

$= \{[q_3], [q_0 q_3], [q_1 q_3], [q_2 q_3], [q_0, q_1, q_3], [q_0, q_2, q_3], [q_1, q_2, q_3], [q_0, q_1, q_2, q_3]\}$

Now find state table

$\delta'([q_0, q_1], a) = \delta(q_0, a) \cup \delta(q_1, a)$:

State/Σ	a	b
$[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$
$[q_0, q_1, q_3]$	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$

stop now, only explore states reachable by initial state. stop when all new states explored. ← better than exploring all 16 states

Draw state diagram in the end.

How to minimize an automata (the no. of states)

→ Minimization of an Automata:-

suppose we have automata, $M = (Q, \Sigma, \delta, q_0, F)$

Def^n: Two states q_1 & q_2 are s.t.b. k-equivalent ($k \geq 0$) iff both $\delta(q_1, x)$ & $\delta(q_2, x)$ are either final or non-final $\forall\, x \in \Sigma^*$ where $|x| \leq k$

P.T.O

Two final states are 0-equivalent

Two non-final states are 0-equivalent

Equivalence relation on Q.

Equivalence relation partitions the set into equivalence classes.

$\Pi_k = \{ Q_1^k, Q_2^k, \ldots Q_i^k, \ldots Q_n^k \}$

Check: If q_1 & q_2 are $(k+1)$ eq., are they k equivalent.

$\delta(q_1, x)$ & $\delta(q_2, x)$ are either final or non-final $\forall x \in \Sigma^*$, $|x| \leq k+1$

this is true, but converse need not be true

$\not\leftarrow$

P.T.O.

★★★

Result: Two states q_1 & q_2 are $(k+1)$ eq. if

(i) they are k-eq.

(ii) $\delta(q_1, a)$ & $\delta(q_2, a)$ are also k-eq. $\forall a \in \Sigma$

Proof: Let q_1 & q_2 are not $(k+1)$ eq.

suppose $|w| = k+1$

i.e. you have strings such as one is final & other is not.

$\delta(q_1, w)$ → final $\qquad \delta(q_2, w)$ → non final

$w = aw_1$

$|w_1| \leq k$

$\delta(q_1, aw_1)$ → final $\qquad \delta(q_2, aw_1)$ → non-final

$\delta(\delta(q_1, a), w_1) \qquad \delta(\delta(q_2, a), w_1)$

Contradiction

Hence they are k+1 eq.

14/01/20

Π_k = set of k-equivalence classes

$= \{Q_1^k, Q_2^k, \dots, Q_i^k\}$

<u>Construction of Minimum Automata</u>

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

<u>Step 1</u>: Find out Π_0 → set of 0 equivalent classes

$\Pi_0 = \{Q_1^0, Q_2^0, \dots, Q_i^0\}$

Only 2 equivalence classes in Π_0 (final & non-final states).

$\Pi_0 = \{Q_1^0, Q_2^0\}$ → $Q - Q_1$, set of non-final states

Q_1^0 → set of final states

<u>Step 2</u>: Construction of Π_{k+1} from Π_k

if $q_1, q_2 \in Q_i^k \Rightarrow q_1$ & q_2 are k-equivalent

for q_1 & q_2 to be k+1 equivalent,

$\delta(q_1, a)$ & $\delta(q_2, a)$ $\forall a \in \Sigma$ should also be k-equivalent

$\Rightarrow q_1$ & q_2 are k+1 eq.

<u>Step 3</u>: for some k, $\Pi_{k+1} = \Pi_k$ then we stop.

Q.1)

State/Σ	0	1
$\to q_0$	q_1	q_5
q_1	q_6	q_2
q_2	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

$\Sigma = (0, 1)$

Ans.1) $Q_1^0 = [q_2]$ → only one final state

$Q_2^0 = [q_0, q_1, q_3, q_4, q_5, q_6, q_7]$

Don't use curly brackets for representing classes

$\Pi_0 = \{Q_0^1, Q_0^2\}$ → set, so curly bracket used

$\Pi_1 = \{[q_2], [q_0, q_4, q_6], [q_1, q_7], [q_3, q_5]\}$

q_2 can't be compared w/ anyone

for q_0 & q_1 P.T.O

$\delta(q_0, a)$, $\delta(q_1, a)$ $\forall$ $a \in \Sigma$

$a \in \{0, 1\}$

$(q_0, 0)$, $(q_1, 0)$
$= q_1$ $= q_6$ = both non-final
hence 1-equ.

$(q_0, 1)$, $(q_1, 1)$
$= q_5$ $= q_2 \Rightarrow$ not equ.
hence not possible

q_0 q_3 also fails

q_0, q_4 succeeds

Next is q_0, q_5
can also check
q_4, q_5 as q_4 & q_0 are equ.
fails

q_4, q_6 passes

q_6, q_7 fails

now repeat process starting from q_1

Don't check q_1 with q_0, q_4, q_6

check w/ q_2, $q_3 \rightarrow$ fails

then $q_1, q_5 \rightarrow$ fails

& lastly $q_1, q_7 \rightarrow$ passes

starting w/ q_3 (skip q_0, q_1, q_2, q_4)

check q_3 & q_5

final $\Pi_1 = \{[q_2], [q_0, q_4, q_6], [q_1, q_7], [q_3, q_5]\}$

get Π_2, check w/ $\uparrow$

$\Pi_2 = \{[q_2], [q_0, q_4], [q_6], [q_1, q_7], [q_3, q_5]\}$

$\Pi_3 = \{[q_2], [q_0, q_4], [q_6], [q_1, q_7], [q_3, q_5]\}$

$\Pi_3 = \Pi_2$
so we stop

so, now the states are

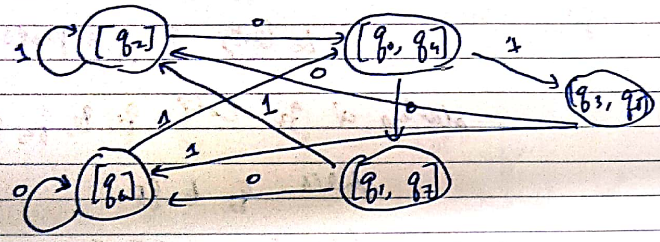
$[q_2]$ $[q_0, q_4]$ $[q_3, q_5]$
$[q_6]$ $[q_1, q_7]$

P.T.O.

∴ initial state is $[q_0, q_4]$ (bcz q_0 is initial)

$[q_2]$ is final state

now check w/ inputs,



We have to check whether string accepted by automata is also accepted by above min. automata

Q.2)

State/Σ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_3	q_0
q_4	q_3	q_5
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	q_3

$\Pi_0 = \{[q_3], [q_0, q_1, q_2, q_4, q_5, q_6, q_7]\}$

$\Pi_1 = \{[q_3], [q_0, q_1, q_5, q_6], [q_2, q_4], [q_7]\}$

$\Pi_2 = \{[q_3], [q_0, q_6], [q_1, q_5], [q_2, q_4], [q_7]\}$

$\Pi_3 = \{[q_3], [q_0, q_6], [q_1, q_5], [q_2, q_4], [q_7]\}$

$\Pi_3 = \Pi_2$

Draw diagram yourself

17/01/20

→ **Grammar** :

Ram ate quickly
Shyam walked slowly
Anu ran slowly

S → Terminals <noun> <verb> <adverb>
S → <noun> <verb>

Description of the sentence

Variables, Terminals, rules

Rule - 1	Rule 2
<noun> → Ram	<verb> → ate
<noun> → Shyam	<verb> → walked
<noun> → Anu	<verb> → ran

<adverb> → quickly
<adverb> → slowly
Rule 3

Rules are known as production rules
for english, hindi, all lang.s

$||^{ly}$ we can make rules for formal language

A grammar is defined as,

$G = (V_N, \Sigma, P, S)$, where

1. V_N is a finite non-empty set of variables (non-terminal symbols)
2. Σ is a finite non-empty set of terminals
3. $V_N \cap \Sigma = \phi$.
4. $S \in V_N$ known as start symbol/ start variable
5. P is a set of production rules, whose elements are denoted by $\alpha \to \beta$, where $\alpha, \beta \in (V_N \cup \Sigma)^*$ & α contains at least one symbol from V_N

$\alpha \to \beta \simeq$ <noun> → Ram

Variables denoted by uppercase letters
Terminals " " lowercase "

~~XXXX~~ Suppose $(\alpha \to \beta) \in P$

$\gamma, \delta \in (V_N \cup \Sigma)^*$

$\gamma \alpha \delta \to \underbrace{\gamma \beta \delta}$ derivable from $\gamma \alpha \delta$

1st step derivative

S → Ram <verb> <adverb>

↳ Ram <verb> quickly

2nd step derivation

Ram ate quickly

3rd step derivation

$\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \overset{n}{\Rightarrow} \alpha_n = \beta$

$\alpha \underset{G}{\overset{n}{\Rightarrow}} \beta$ → n-step derivation

$\alpha \underset{G}{\overset{*}{\Rightarrow}} \beta$ $\qquad$ $\alpha \Rightarrow \beta$

↳ no step means it is directly derivable (1st step derivation)

Q1) $G = (\{S\}, \{0, 1\}, \{S \to 0S1, S \to 01\}, S)$

Ans 1) $S \to 01$ → terminal, no further steps

$S \to 0S1$

↳ here we have variable which can be replaced by production rules (we have 2)

$S \to 0S1 \xrightarrow{S \to 01} 0011$ (terminated)

$S \to 0S1 \xrightarrow{S \to 0S1} 00S11 \xrightarrow{S \to 01} 0^3 1^3$ (terminated)

$\downarrow S \to 0S1$

$000S111 \xrightarrow{S \to 01} 0^4 1^4$

we can keep going

all terminals reachable through S

$\{0^n 1^n / n \geq 1\}$ → all strings reachable

Language generated by a Grammar G

Denoted by $L(G)$

$L(G) = \{ w \in \Sigma^* / S \overset{*}{\Rightarrow} w \}$

↳ sentence

Two grammars G_1 & G_2 are equivalent iff $L(G_1) = L(G_2)$

P.T.O

Q.1) $G = (\{S\}, \{0,1\}, \{S \to 0S1, S \to \wedge\}, S)$ Find LG

Ans 1) $S \to \wedge$ (empty string cannot be considered as a terminal but if production rules states so, then only $\wedge \in L(G)$)

$\wedge \in L(G)$

$S \to 0S1 \to 01$ $(\because \wedge)$

$\downarrow$

$0^2 S 1^2 \to 0^2 1^2$ $0 \wedge 1 = 01$

$\downarrow$

$0^3 S 1^3 \to 0^3 1^3$

$\downarrow$

$\therefore \{0^n 1^n / n \geq 0\} = L(G)$

Q.2) $(\{S\}, \{a\}, \{S \to SS\}, S)$

Ans 2) ~~L(G)=~~ $S \to SS$ only PR

LHS & RHS are both variables

No terminals will be achieved

$\therefore L(G) = \phi$

$S \notin \Sigma^*$

PTO

Q.3) Suppose $L(G) = \{b, bb, bbb\}$, find out grammar

Ans 3) Find PR, rest is straightforward

Start w/ length of min. string,

$\therefore S \to b$ is a production rule

Now we can't consider $S \to Sb$ as it will go on infinitely.

$G = (\{S\}, \{b\}, \{S \to b / bb / bbb\}, S)$

Q.4) L(G) = any non-empty string on $\{0,1\}$. Find grammar.

Ans 4) $S \to 0/1/0S/$ (start with either).
1S

we can use recurrance as it is infinite
satisfies all scenarios

$S \to 0S \to 00$

$\downarrow$

01

01S $\to$

I look funny when I squint?

20/01/20

→ **Chomsky Classification of Languages**

4 types
type 0 – type 3

Type 0 Grammar (Phase structure Grammar):

No restriction of production rules
Any production rule is type 0.

If all production rules are type 0, then grammar is type 0.

Any grammar is type 0.

The language generated by type 0 grammar is called type 0 language.

Type 1 Grammar (Context sensitive Grammar):

$\phi A \psi \rightarrow \phi \alpha \psi$ $\quad A \in V_N \quad \alpha, \phi, \psi \in (V_N \cup \Sigma)^*$

ϕ ↓ left context, ψ ↓ right context

Eg. $abAbcd \rightarrow abABbcd$
$A = A$
$\alpha = AB$
$\phi = ab$
$\psi = bcd$

A production of the type $\phi A \psi \rightarrow \phi \alpha \psi$ a type 1 production if $\alpha \neq \wedge$ (α is not an empty string).

aka context sensitive production (if production is of above type)

If all productions are type 1, grammar produced is called context sensitive grammar, generated language is called context sensitive language.

CFG (in short)
Type 2 Grammar (Context free Grammar):-

Any production rule of the type $A \rightarrow \alpha$, $A \in V_N$, $\alpha \in (V_N \cup \Sigma)^*$ is called type 2 production

rest defn. from above.

Type 3 Grammar (Regular Grammar):-

Any production of the type,
$A \rightarrow a$, $A \rightarrow aB$
$A, B \in V_N$, $a \in \Sigma$
is called type 3 production.

Also called regular grammar.
Regular language.

~~Handwritten~~

$L_0 \rightarrow$ family of type 0 languages

each L_i is a type 0 language

llly

$L_{csl} \rightarrow$ family of context sensitive languages

$L_{cfl} \rightarrow$ " " " free "

$L_{rl} \rightarrow$ " " regular "

$L_{rl} \subseteq L_{cfl}$

$A \rightarrow a$ or $A \rightarrow ab$

both cases don't have any contexts & satisfy conditions of L_{cfl}

$L_{cfl} \subseteq L_{csl}$

$A \rightarrow \alpha$

$\wedge A \wedge \rightarrow \wedge \alpha \wedge$ (becomes csl)

$L_{csl} \subseteq L_0$

$\therefore L_{rl} \subseteq L_{cfl} \subseteq L_{csl} \subseteq L_0$

<u>Thm</u>: Each of the classes $L_0, L_{csl}, L_{cfl}, L_{rl}$ is closed under union.

<u>Proof</u>: Let L_1 & L_2 be two languages of the same type say i.

$G_1 = (V_N', \Sigma_1, P_1, S_1)$ and

$G_2 = (V_N'', \Sigma_2, P_2, S_2)$ s.t.

$L(G_1) = L_1$ and $L(G_2) = L_2$

$G_u = (V_N' \cup V_N'' \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_u, S)$, where S is new symbol

i.e $S \notin V_N' \cup V_N''$

$P_u = P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}$

we have T.S.T $L(G_u) = L_1 \cup L_2$

let $w \in L_1 \cup L_2$ i.e $w \in L_1$ or $w \in L_2$

i.e. $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w$ or $S_2 \overset{*}{\Rightarrow} w$

$S \Rightarrow S_1 \overset{*}{\Rightarrow} w$ i.e. $S \Rightarrow w$

or if

$S \Rightarrow S_2 \overset{*}{\Rightarrow} w$ i.e. $S \Rightarrow w$

$\therefore w \in L(G_u)$ $(L_1 \cup L_2) \subseteq L(G_u)$

for converse

P.T.O

Let $w \in L(G_u)$ i.e. $S \overset{*}{\underset{G_u}{\Rightarrow}} w$

for G_u, first step derivation would be
$S \Rightarrow S_1$ or $S \Rightarrow S_2$

$S \Rightarrow S_1 \Rightarrow \dots \Rightarrow w$ i.e. $w \in L_1$ or
$S \Rightarrow S_2 \Rightarrow \dots \Rightarrow w$ i.e. $w \in L_2$

i.e. $w \in L_1 \cup L_2$

Thm: Each class (category) is closed under concatenation

Proof: L_1, L_2 two lang.s of same type i.

$L(G_1) = L_1$, $L(G_2) = L_2$ (write stuff from last proof)

$G_{con} = (V_N' \cup V_N'' \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_{con}, S)$

$P_{con} = P_1 \cup P_2 \cup \{S \to S_1S_2\}$

T.S.T. $L_1L_2 = L(G_{con})$

Let $w \in L_1L_2$ i.e. $w = w_1w_2$, $w_1 \in L_1$, $w_2 \in L_2$

$S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w_1$ $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} L_2$

first step derivation

$S \underset{G_{con}}{\Rightarrow} S_1S_2 \Rightarrow \dots \Rightarrow w_1S_2 \Rightarrow \dots \Rightarrow w_1w_2 = w$

$S \overset{*}{\Rightarrow} w$ i.e. $w \in L(G_{con})$

P.T.O

$L_1L_2 \subseteq L(G_{con})$

for converse,

Let $w \in L(G_{con})$ i.e. $S \overset{*}{\Rightarrow} w$

first step will be $S \Rightarrow S_1S_2$,

$S \Rightarrow S_1S_2 \Rightarrow \dots \Rightarrow w_1S_2 \Rightarrow \dots \Rightarrow w_1w_2$

Hence proved

21/01/20

$\boxed{\text{H.W}}$ S.T each of the categories are closed under transpose

$G = (V_N, \Sigma, P, S)$
$L = L(G)$
$\alpha \to \beta$ iff $\alpha^T \to \beta^T$ in P^T

$G' = (V_N, \Sigma, P^T, S)$

$L^T = L(G^T)$

Do rest as exercise

P.T.O

24/01/20

→ **Algo. to test whether $w \in L(G)$:-**

$G = (V_N, \Sigma, P, S)$

1. $W_0 = \{S\}$ [construct tree]

$W_{i+1} = W_i \cup \{\beta \in (V_N \cup \Sigma)^* \mid \exists\, \alpha \in W_i$ s.t. $\alpha \Rightarrow \beta$ & $|\beta| \le n\}$

obviously $W_i \subseteq W_{i+1}$

for some k, $W_{k+1} = W_k$ (stop at this point)

2. If $w \in W_k$ then $w \in L(G)$ otherwise $w \notin L(G)$.

Q.1) Consider G given by $S \to 0SA_12$, $S \to 012$, $2A_1 \to A_12$, $1A_1 \to 11$.

Test whether $00112 \in L(G)$ & $001122 \in L(G)$

Ans 1) Take $w = 00112$, $|w| = 5$

$W_0 = \{S\}$

$W_1 = W_0 \cup \{012, 0SA_12\}$

$W_1 = \{S, 012, 0SA_12\}$

$W_2 = W_1 \cup \{012, 0SA_12\}$

$0SA_12 \xrightarrow{012} 0012A_12$ } both strings length > 5

$\xrightarrow{0SA_12}$ $00SA_12A_12$: not included

$W_2 = \{S, 012, 0SA_12\} = W_1$ (stop)

$00112 \notin W_1$

$\therefore$ doesn't belong to $L(G)$

$w = 001122$ $|w| = 6$

$W_0 = \{S\}$

$W_1 = \{S, 012, 0SA_12\}$

$W_2 = W_1 \cup \{012, 0SA_12, 0012A_12\}$

$0SA_12 \to 0012A_12$

$0SA_12 \to 00SA_12A_12$ (length = 7)

$0012A_12 \to 001A_122 \to 0011122$ (length = 7)

$W_3 = W_2 \cup \{012, 0SA_12, 0012A_12, 001A_122\}$

$W_3 = \{S, 012, 0SA_12, 0012A_12, 001A_122\}$

$W_4 = W_3 \cup \{012, 0SA_12, 0012A_12, 001A_122, 001122\}$

$W_5 = W_4 \cup \{012, 0SA_12, 0012A_12, 001A_122, 001122\}$

$W_5 = W_4$

$001122 \in W_4$ $\therefore w \in L(G)$

→ **Regular Expressions :-**

Regular expressions are defined recursively.

Σ = input alphabet

we define reg. exp. over Σ, R.e over Σ is defined as follows :-

1) Any symbol $a \in \Sigma$ is a r.e (Λ is a r.e)

2) If R_1 & R_2 are r.e then their union written as $R_1 + R_2$ is also a r.e.

3) If R_1 & R_2 are r.e over Σ then their concatenation R_1R_2 is also a r.e over Σ.

4) If R is a r.e over Σ then iteration R^* is also a re over Σ.

5) If R is a r.e over Σ then (R) is also a r.e over Σ.

6) Any expression obtained by using the above rules 1-5 is also a r.e over Σ.

a^* = set of all strings

$a \in \Sigma$

$a = \{a\}$

Bold a

set represented by the regular expression a.

$a + b = \{a\} \cup \{b\} = \{a, b\}$

$ab = \{ab\}$

$a^* = \{\phi, a, aa, aaa, \dots\}$

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

Q.1) {101}, write corresponding re.

Ans.1 101

Q.2) {01, 10}

Ans.2) 01 + 10

Q.3) {Λ, ab}

Ans.3) Λ + ab

Q.4) {1, 11, 111, ...}.

Ans.4) $1(1^*)$ (multiply w/ 1 to get rid of ϕ)

$\therefore 1 \times \phi$ = nothing

Q.5) The set of all strings of 0's and 1's ending in 00.

Ans 5) $(0+1)^* 00$

Identities for Regular Expressions:

P, Q – re over Σ

$P \equiv Q$ iff they have same set of strings
(equivalent)

Identities for re.

1. $\Lambda R = R\Lambda = R$
2. $\Lambda^* = \Lambda$
3. $R + R = R$
4. $R^* R^* = R^*$
5. $R^* R = RR^*$
6. $(R^*)^* = R^*$
7. $\Lambda + RR^* = R^* = \Lambda + R^* R$
8. $(PQ)^* P = P(QP)^*$
9. $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$
10. $(P+Q)R = PR + QR$ & $R(P+Q) = RP + RQ$

In string, $x(a+by) \neq (a+by)x$
order of concatenation matters

27/01/20

☆DD (Imp. for Exam)

Arden's Theorem –

Let P & Q be 2 r.e. over Σ. If P does not contain Λ, then the following eqn.

$R = Q + RP$ — ①

has a unique solution given by $R = QP^*$

Proof: T.S. soln. exists & it is unique.

$Q + RP = Q + (QP^*)P = Q + QP^*P$

$= Q(\Lambda + P^*P) = QP^* = R$

$\Rightarrow R = QP^*$ is soln. of the eq. ①

Uniqueness

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

$= Q + QP + RP^2$

$= Q + QP + (Q+RP)P^2$

$= Q + QP + QP^2 + RP^3$

$= Q + QP + QP^2 + (Q+RP)P^3 = Q + QP + QP^2 + QP^3 + RP^4$

$= Q + QP + QP^2 + \dots + QP^i + RP^{i+1}$

$= Q(\Lambda + P + P^2 + \dots + P^i) + RP^{i+1}$ — (2)

(1) & (2) are equivalent

Let w be any string of length i in R

$w \in R$, $|w| = i$

$w \in Q(\Lambda + P + P^2 + \dots + P^i) + RP^{i+1}$

$+$ means Union

but $w \notin P^{i+1}$

$\because P$ doesn't contain Λ, hence every string is of length $i+1$.

$\therefore w \in Q(\Lambda + P + P^2 + \dots + P^i)$

$\underbrace{\Lambda + P + P^2 + \dots + P^i}_{= P^*}$

$\Rightarrow w \in QP^*$

Now, suppose $w \in QP^*$

~~...~~

then $w = QP^k$ for some k

$\Rightarrow w \in Q(\Lambda + P + P^2 + \dots P^i)$

$\Rightarrow w \in Q(\Lambda + P + P^2 + \dots P^i) + RP^{i+1}$

$\underline{\underline{w \in R}}$ Hence proved

Q.1) Prove that

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

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

Start from LHS, use identities & solve.

LHS $= (1 + 00^*1)[\Lambda + (0 + 10^*1)^*(0 + 10^*1)]$

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

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

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

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

Hence proved

$\rightarrow$ <u>Transition system w/ Λ-moves:-</u>

Suppose we want to replace a Λ-move from a vertex v_1 to v_2.

Step 1: Find all the edges starting from v_2.

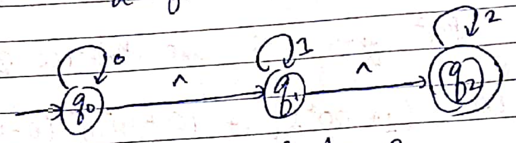
Step 2: Duplicate all these edges starting from v_1 w/o changing the edge ~~...~~ labels.

P.T.O

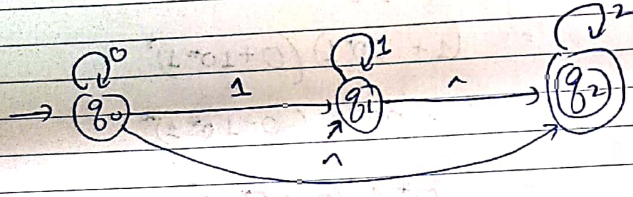
Step 3: If v_1 is an initial state, make v_2 also an initial state.

Step 4: If v_2 is a final state, make v_1 also a final state

Q1)

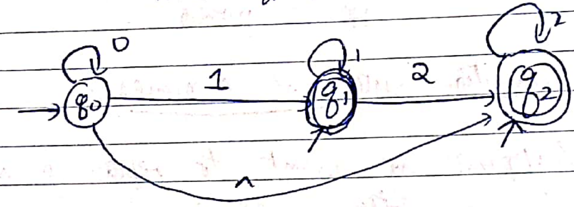


Remove $\wedge$-move q_0 to q_1

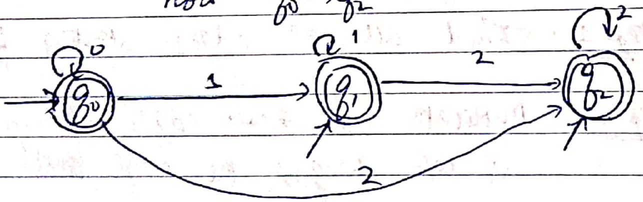


Remove $q_1 \to q_2$ $\wedge$-move $\qquad q_2 \xrightarrow{2} q_2$

Q.2)



now $q_0 \to q_2$



w = 012

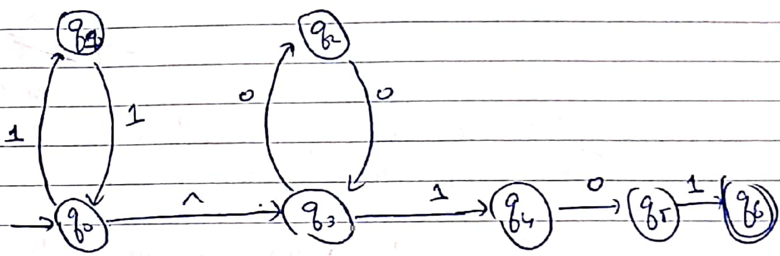
$q_0 \xrightarrow{0} q_0 \xrightarrow{\wedge} q_1 \xrightarrow{1} q_1 \xrightarrow{\wedge} q_2 \xrightarrow{2} q_2$

w is accepted

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

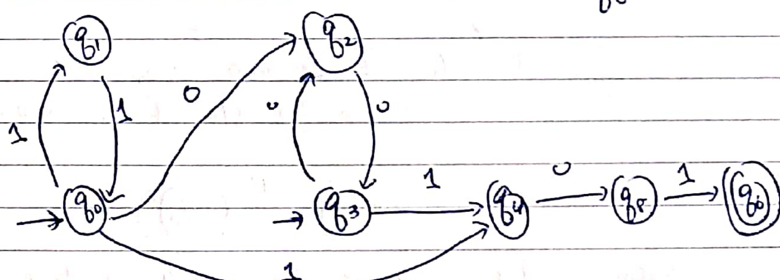
accepted by both systems

Q2)



$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{\wedge} q_3 \xrightarrow{1} q_4 \xrightarrow{0} q_5 \xrightarrow{1} q_6$ $\qquad w = 11101$

Ans 2)



$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{1} q_4 \xrightarrow{0} q_5 \xrightarrow{1} q_6$

w is accepted.

28/01/20

Kleene's Theorem: If R is a regular expression over Σ representing $L \subseteq \Sigma^*$, then $\exists$ an NDFA M w/ Λ-moves s.t.

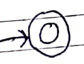
$L = T(M)$

Proof: Prove by PMI on the no. of characters in R

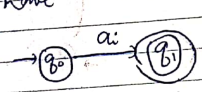
($+$, $*$, count as characters, $1^* + 00^* + 11^*$ = 10 char)

Let the no. of characters in R is 1.
Then, either $R = \Lambda$ or $R = a_i$

for $R = \Lambda$, NDFA has one state which is initial & final state



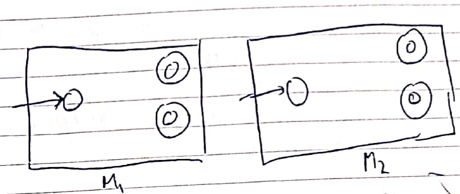
for $R = a_i$, since one character; we have 2 states in NDPA



Induction step: Suppose the result is true for any re R having n characters
Let the no of characters in R be $(n+1)$

if R has $(n+1)$ characters, then R is of the form,
$R = P + Q$ or $R = PQ$ or $R = P^*$ where P & Q are re having $\leq n$ characters.

$\Rightarrow \exists$ NDFA M_1 & M_2 s.t. $L(P) = T(M_1)$ & $L(Q) = T(M_2)$

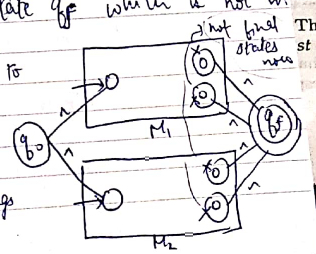


Case I: when $R = P + Q$
we have two machines M_1 & M_2

Introduce new initial state q_0 which is not in M_1 or M_2.

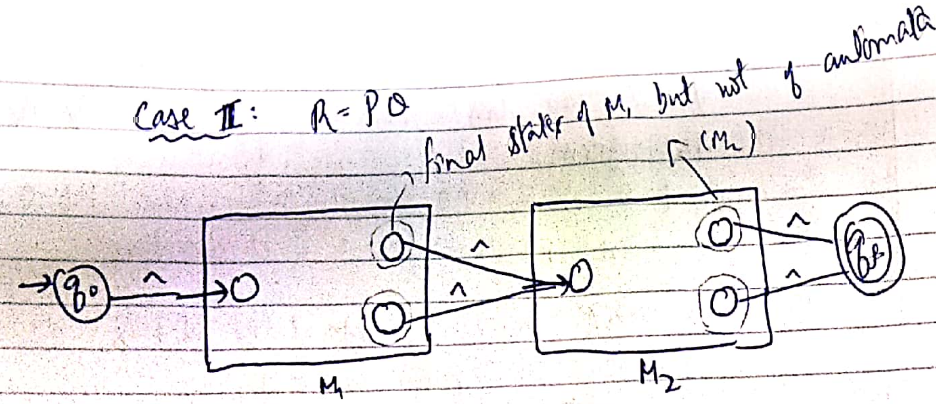
Introduce new final state q_f which is not in M_1 or M_2

By using Λ-move, reach to following states



This is an automata which accepts all strings in $R = P + Q$

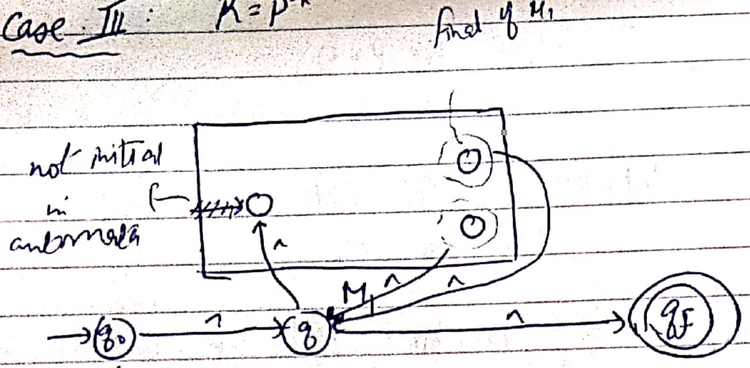
Case II: $R = PQ$



Introduce q_0 & q_f which are not a part of M_1 ~~and~~ or M_2.

Any string of $L(PQ)$ is accepted by this automata.

Case III: $R = P^*$



Introduce 3 states q_0, q & q_f

Any string in $L(P^*)$ is accepted by this automata

→ Algebraic Method using Arden's theorem :-

This is used to find a r.e recognized by a T.S (automata) / transition system

Following assumptions are made of T.S :-

1) Transition graph does not have $\wedge$-moves

2) It has only one initial state

3) If α_{ij} denotes the r.e representing the set of labels of edges from v_i to v_j, then we can get the following:-

$V_1 = V_1\alpha_{11} + V_2\alpha_{21} + \dots + V_n\alpha_{n1} + \wedge$ (only come for initial state)

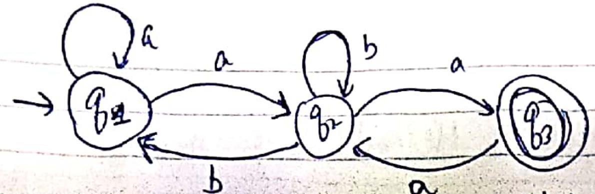
$V_2 = V_1\alpha_{12} + V_2\alpha_{22} + \dots + V_n\alpha_{n2}$

$\vdots$

$V_n = V_1\alpha_{1n} + V_2\alpha_{2n} + \dots + V_n\alpha_{nn}$

By applying arden's theorem, express V_i in terms of α_{ij}'s. For getting the string accepted by the T.S, we have to take union of all V_i's corresponding to final states

Q.1)



$q_1 = q_1 a + q_2 b + \Lambda$ → (add empty string ∵ q_1 is initial) → (no edge from $q_3 \to q_1$)

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

$q_3 = q_2 a$

final state is q_3, write q_3 in terms of α_{ij}'s (a, b, etc)

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

$= q_1 a + q_2 b + q_2 aa$ (∵ $q_3 = q_2 a$)

$\underset{R}{q_2} = \underset{Q}{q_1 a} + \underset{R}{q_2}\underset{P}{(b + aa)}$

$R = Q + RP$ (Arden's thm)

$R = QP^*$

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

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

$\underset{R}{q_1} = \underset{R}{q_1}\underset{P}{(a + a(b+aa)^* b)} + \underset{Q}{\Lambda}$

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

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

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

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

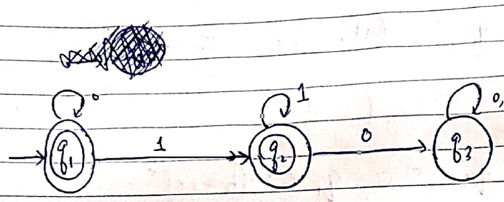
$q_3 = q_2 a$

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

Used to identify string accepted by T.S.

07/02/20

Q.1)



Ans. 1)

$q_1 = q_1 0 + \wedge$

$q_2 = q_2 1 + q_1 1$

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

$q_1 = q_1 0 + \wedge = \wedge + q_1 0$

$R \qquad\qquad Q \quad RP$

$q_1 = \wedge 0^* = 0^*$

$R = QP^*$

$q_2 = q_2 1 + \wedge 0^* 1$

$q_2 = 1(q_2 + \wedge 0^*)$

$q_2 = 1(q_2 + 0^*)$

$q_2 = 0^* 1 + q_2 1$

$R \qquad Q \qquad RP$

$q_2 = 0^* 1 1^*$

string accepted by machine is string accepted by final state

P.T.O

$= q_1 + q_2 = 0^* + 0^* 1 1^*$

$= 0^*(\wedge + 11^*)$

$\wedge + PP^* = P^*$

$= \underline{0^* 1^*}$

If any r.e. is given, we can create finite automata which accepts r.e., by use of Kleene's thm.

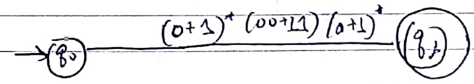
→ **Construction of DFA equivalent to R.E :-**

Kleene's thm helps create NDFA accepting r.e.

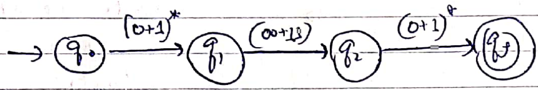
we convert NDFA to DFA

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

Ans. 1)

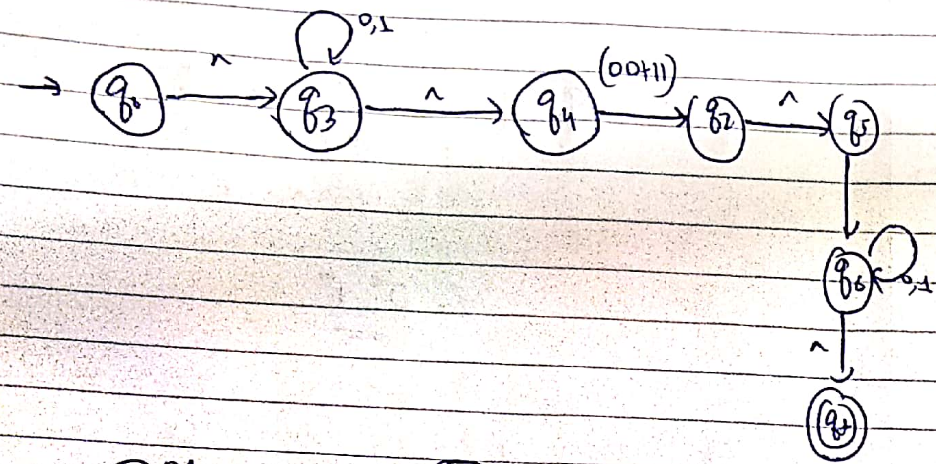


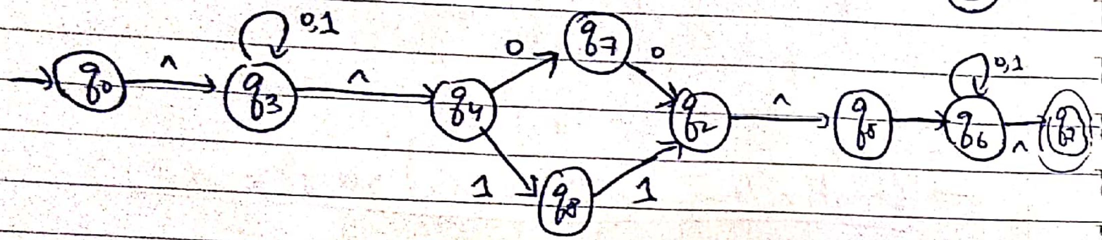
concatenation solved first



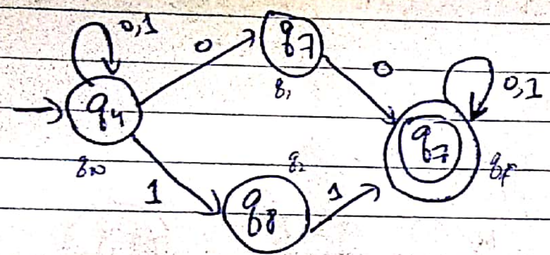
we add a loop to get $(0+1)^*$

P.T.O





Now remove all null moves



Make transition table for this

Q\Σ	0	1
→ q_0	$q_0 q_1$	$q_0 q_2$
q_1	q_f	–
q_2	–	q_f
q_f	q_f	q_f

this is NDFA bcs ~~each~~ one input gives multiple outputs

So convert to DFA

if NDFA has 4 states, DFA will have $2^4 = 16$ states

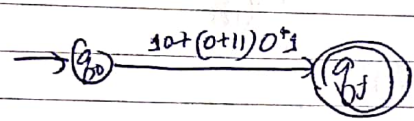
in DFA, $[q_0]$ (w/ bracket) is initial & $[q_f]$ is final

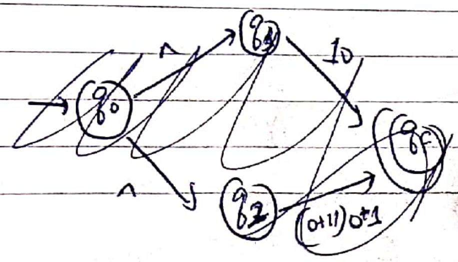
	0	1
$[q_0]$	$[q_0, q_1]$	$[q_0, q_2]$
$[q_0, q_1]$	$[q_0, q_1, q_f]$	$[q_0, q_2]$
$[q_0, q_2]$	$[q_0, q_1]$	$[q_0, q_2, q_f]$
$[q_0, q_1, q_f]$	$[q_0, q_1, q_f]$	$[q_0, q_2, q_f]$
$[q_0, q_2, q_f]$	$[q_0, q_1, q_f]$	$[q_0, q_2, q_f]$

we only include states reachable from initial state.

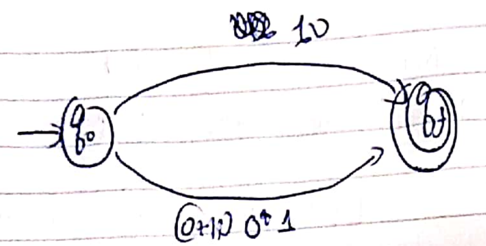
Q2) Construct DFA w/ respect to $10 + (0+11)0^*1$

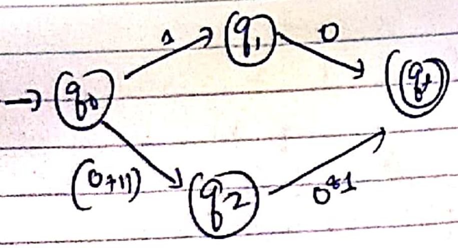
Ans 2)

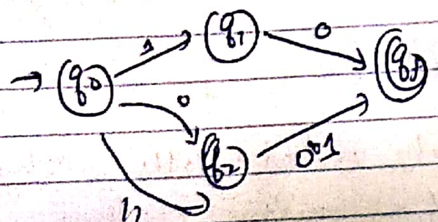


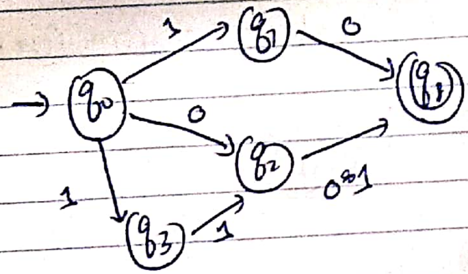


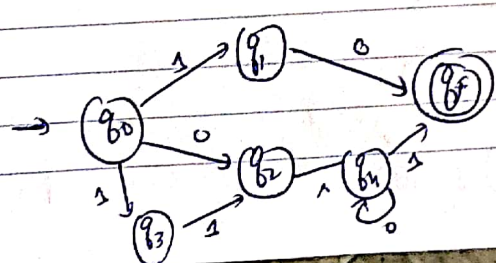
P.T.O











10/02/20

→ <u>Pumping lemma for Regular sets</u> :-

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton w/ n states. Let L be the regular set accepted by M. Let $w \in L$ & $|w| \geq n$. Then $\exists\, x, y, z$ s.t $w = xyz$, $y \neq \wedge$, $|xy| \leq n$ & $xy^iz \in L$ $\forall\, i \geq 0$

<u>Pf</u>: Let $w = a_1 a_2 a_3 \ldots a_m$, $m \geq n$

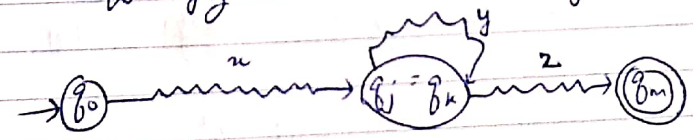
$\delta(q_0, a_1 a_2 a_3 a_4 \ldots a_i) = q_i$ for $i = 1, 2, \ldots, m$

$\left(\Rightarrow q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \ldots \xrightarrow{a_m} q_m\right)$

Suppose string is, $w = \underbrace{a_1 a_2 \ldots q_j}_{x} \underbrace{a_{j+1} \ldots q_k}_{y} \underbrace{a_{k+1} \ldots a_m}_{z}$

As w can be written as xyz, $0 \leq j \leq k \leq n$

$w = xyz$



$\therefore q_j = q_k$ (why?)

$xy^iz \in L$

w/ help of this lemma, we can show a <u>set is not valid</u> by the following :

Step 1: Assume that L is regular. Let n be the no. of states in the corresponding finite automata.

Step 2: Choose a string w s.t. $|w| \geq n$. Use pumping lemma to write $w = xyz$, $|xy| \leq n$, $|y| > 0$.

Step 3: Find a suitable i s.t. $xy^iz \notin L$. This contradicts our assumption. Hence L is not regular.

Q.1) Show that $L = \{a^{i^2} \mid i \geq 1\}$ is not regular

Ans 1) Suppose L is regular. Let n be the no. of states in the corresponding F.A.

Let $w = a^{n^2}$, $|w| = n^2 > n$

$w = xyz$, $|xy| \leq n$, $|y| > 0$

Consider xy^2z, $|xy^2z| = |x| + 2|y| + |z| >$ $|x| + |y| + |z| = n^2$

$\Rightarrow |xy^2z| > n^2$

$|xy^2z| = |x| + 2|y| + |z| = (|x| + |y| + |z|) + |y|$

$= n^2 + |y| \leq n^2 + n$

P.T.O

$n^2 < |xy^2z| \leq n^2 + n < n^2 + n + (n+1) = n^2 + 2n + 1 = (n+1)^2$

$n^2 < |xy^2z| < (n+1)^2$

Q.2) $L = \{a^p \mid p \text{ is prime}\}$ s.t. L is not regular

Ans 2) Suppose L is regular & the corresponding automaton have n states.

Let $w \in L$, $|w| \geq n$ $|w| = |xyz| = a^{|p|}$, p is prime

Suppose $y = a^m$ for some m.

Consider $|xy^{p+1}z| = \underbrace{|x| + |y| + |z|} + p|y|$

$= p + pm$

$= p(1+m)$

$=$ not prime ∵ a multiplication of p & 1+m, hence composite no.

∴ $xy^{p+1}z \notin L$

∴ Our assumption was wrong

11/02/20

→ Closure Property of Regular Sets :-

Thm: If L is regular then L^T is also regular

Proof: Let L be regular. We can construct a F.A, $M = (Q, \Sigma, \delta, q_0, F)$ s.t. $T(M) = L$.

Now we construct automata M', from M by reversing the arrows of the state diagram wrt M.

if $q_1 \xrightarrow{M} q_2$, then

$q_1 \xleftarrow{M'} q_2$

$q_0 \to q_1 \to \cdots \to q_f$ (M)

final state $\xleftarrow[M',\ w^T]{}$ initial state

If $w \in T(M)$, $\delta(q_0, w) = q_f$

---

essentially mean L' (complement)

Thm: If L is regular, then $(\Sigma^* - L)$ is also regular

Proof: Let L be regular. $M = (Q, \Sigma, \delta, q_0, F)$ s.t. $T(M) = L$

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

$w \in T(M) \Rightarrow \delta(q_0, w) \in F$

$w \in T(M') \Rightarrow \delta(q_0, w) \in F'$

$w \notin L$

Thm: If L_1 & L_2 be two regular sets over Σ, then $L_1 \cap L_2$ is also regular over Σ.

Proof: $L_1 \cap L_2 = (L_1' \cup L_2')'$ (by De Morgan's)

$= [(\Sigma^* - L_1) \cup (\Sigma^* - L_2)]'$

$= \Sigma^* - \underbrace{[(\Sigma^* - L_1) \cup (\Sigma^* - L_2)]}_{\text{both regular}}$

∴ whole thing is also regular

→ Equivalence of two F.A. :-

Comparison Method: Let M & M' be two finite automata over Σ.

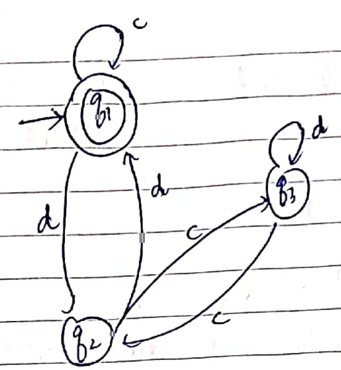
We construct a comparison table consisting of $(n+1)$ columns, where n is the no. of input symbols. The 1st column consists of pairs of the form (q, q'), where $q \in M$ & $q' \in M'$. If (q, q') appears in same row of 1st column, then the correspon-ding entry in the a-columns ($a \in \Sigma$) is (q_a, q_a') where q_a & q_a' are reachable from q & q' respectively on applications of a.

The table is constructed by starting with the pair of initial states q_{in}, q_{in}' of M & M'.

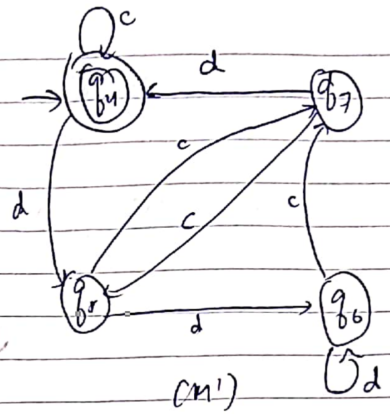
step 1: If we reach a pair (q, q') s.t. q is final in M & q' is final in M' or vice versa, we terminate the construction & M & M' are not eqv.

step 2: Construction is terminated when no new element appears in 2nd & subsequent columns, which are not in 1st column. In this case, M & M' are eqv.

Q.1)



(M)



(M')

(q, q')	(q_c, q_c')	(q_d, q_d')
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_7)	(q_1, q_6)

(q_1, q_6): q_1 final, q_6 non-final, hence stop, not eqv.

→ F.A. w/ Outputs :-

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

Mealy Machine

q_0 - initial state

Q - non-empty finite set of states

Σ - " " " " " input symbols

Δ - output symbols

λ - " function, $\lambda : Q \to \Delta$

δ - $\delta : Q \times \Sigma \to Q$

P.T.O

output is dependent on state only - Moore Machine

In case of Mealy machine,

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

only diff. from moore machine

given moore machine, we can construct mealy machine, given mealy machine, we can construct moore machine.

18/02/20

Moore-Machine

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

Δ - output alphabets

λ - output function.

$\lambda : Q \to \Delta$

Mealy Machine

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

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

we can construct Moore machine from mealy machine & vice versa

→ **Transformation of a Moore Machine into Mealy Machine:-**

$\lambda \to$ Moore machine

$\lambda' \to$ Mealy machine

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

Eg.

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

$\delta(q_0, 0) = q_3$

$\delta(q_0, 1) = q_1$

⋮

$\lambda(q_0) = 0$

$\lambda(q_1) = 1$

$\lambda(q_2) = 0$

$\lambda(q_3) = 0$

now to convert into a mealy machine,

P.T.O

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

$\lambda'(q_0, a)$
$= \lambda(\delta(q_0, a))$
$= \lambda(\delta(q_0, 0)) = \lambda(q_3) = 0$

$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1))$
$= \lambda(q_1) = 1$

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

$\lambda'(q_1, 1) = \lambda(q_2) = 0$

IIIly do for all

→ Transformation of Mealey into Moore :-

present state	Next State a = 0 state	Output	a = 1 state	Output
→ 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

PTO

$\Delta = (0, 1)$

$\delta(q_1, 0) = q_3$, $\delta(q_1, 1) = q_2$
$\delta(q_2, 0) = q_1$, $\delta(q_2, 1) = q_4$
... go on

To convert to moore machine,

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

q_1 has output 1 in both cases, so leave as it is

same for q_3

q_2 & q_4 have diff. outputs so break them into multiple states

not in standard form of moore machine.

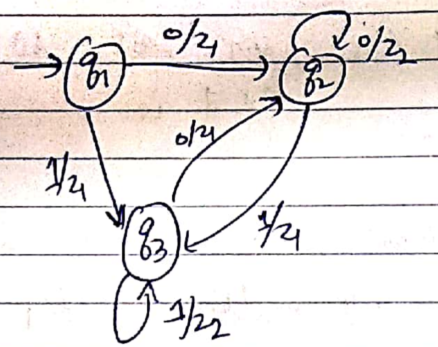
so standardise

P.T.O

Present state	Next state a=0	Next state a=1	Output
→ q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

look at circles behind to get output

Q1) Consider Mealy machine & get equivalent Moore machine



Construct transition table first.

Ans 1)

Present State	a=0 state	a=0 output	a=1 state	a=1 output
→ q_1	q_2	z_1	q_3	z_1
q_2	q_2	z_2	q_3	z_1
q_3	q_2	z_1	q_3	z_2

q_2 & q_3 take upon diff states & hence we will split them

Present State	a=0 state	a=0 output	a=1 state	a=1 output
→ q_1	q_{21}	z_1	q_{31}	z_1
q_{21}	q_{22}	z_2	q_{31}	z_1
q_{22}	q_{22}	z_2	q_{31}	z_1
q_{31}	q_{21}	z_1	q_{32}	z_2
q_{32}	q_{21}	z_1	q_{32}	z_2

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

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

$\Delta = \{z_1, z_2\}$

Now we have to standardize

Present state	Next state		Output
	$a=0$	$a=1$	
$\rightarrow q_1$	q_{21}	q_{31}	$\wedge$ $\rightarrow$ ∵ doesn't affect original
q_{21}	q_{22}	q_{31}	z_1
q_{22}	q_{22}	q_{31}	z_2
q_{31}	q_{21}	q_{32}	z_1
q_{32}	q_{21}	q_{32}	z_2