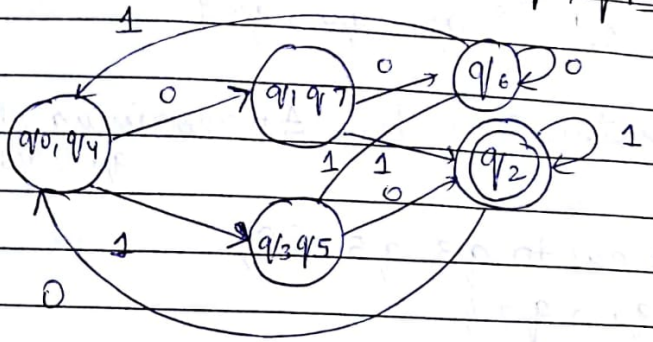
State/Σ	0	1
→ [q0, q4]	[q1, q7]	[q3, q5]
[q1], q7	[q6]	[q2]
[q2]	[q0, q4]	[q2]
[q3, q5]	[q2]	[q6]
[q6]	[q6]	[q0, q4]

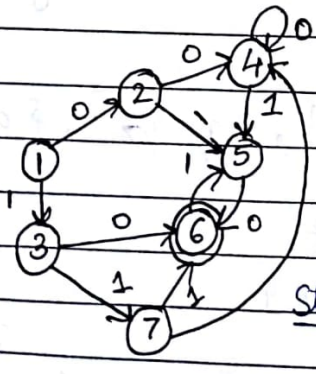
5-8 Marks Question in Test.

Bunk Pages - The Social Notebook

Class Test on Friday



(•) Myhill Nerode Theorem : Table Filling Method



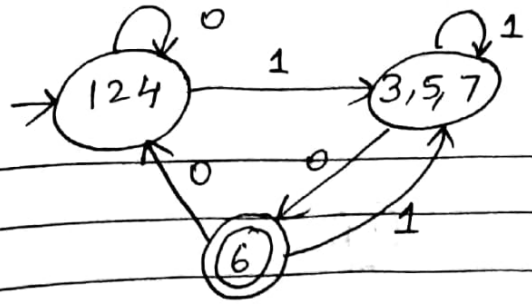
Step1 Making 6x6 half matrix. In row-wise & left to right → first STR states

Step2 In column wise, top-to-bottom ↓ Starting 2nd state

{(1,2) (1,4) (2,4) (3,5) (3,7) (5,7)}

Step3 First put x in final states in row and column And check for other pairs

	0	1	2	3	4	5	6
0							
1							
2		✓					
3		X	X				
4		✓	✓	X			
5		X	X	✓	X		
6		X	X	X	X	X	
7		X	X	✓	X	✓	X



Step4 Now, we check,

(1,2) —0→ (2,4) —0→ (4,4) Blocked
Put ✓ in that cell
(2,4) —1→ (3,5) —1→ (6,6) ✓
(3,5) —1→ (7,7) ✓
(3,5) —0→ (6,6) ✓

(1,3) —0→ (2,6) —0→ (4,∅)
(2,6) —1→ (3,7) —0→ (6,4) Non equivalent → X
(3,7) —1→ (7,6) → X
Non eq state → X

(1,4) —0→ (2,4) ✓
(1,4) —1→ (3,5) ✓

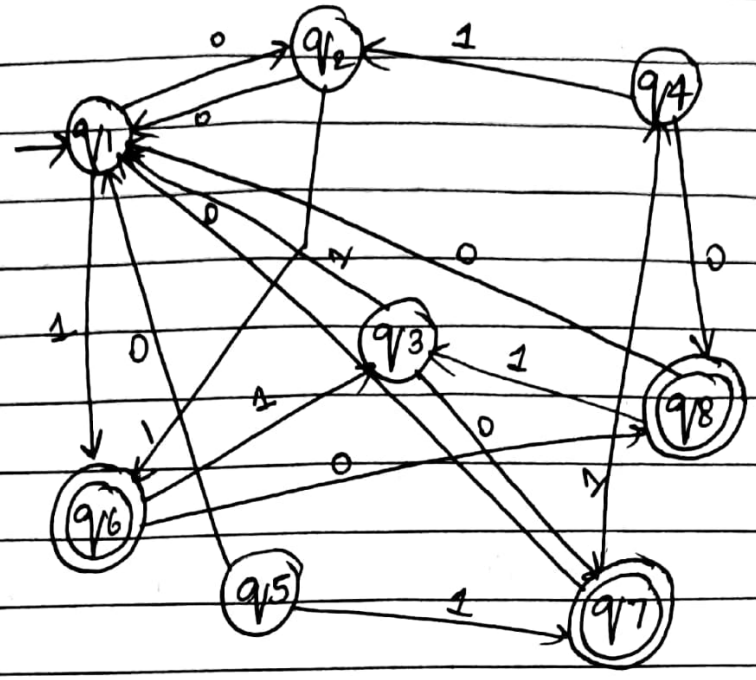
Step5 Construct a minimum state automata whose transitn is:-

Σ	a	b
→ q0	q1	q2
q1	q1	q3
q2	q3	q4
q3	q1	q5
q4	q4	q2
(q5)	q6	q6

∴ ∅ → NFA

Read: Equivalence of 2 finite automata

Ques



	ε	0	1	ε	0	1
→	q_1	q_2	q_6	$q_1\ q_2$	$q_1 q_2$	q_6
	q_2	q_1	q_6	$q_3\ q_4$	$q_8\ q_7$	$q_1 q_2$
	q_3	q_7	q_1	q_6	$q_8\ q_7$	$q_3 q_4$
	q_4	q_8	q_2	$q_7 q_8$	$q_1 q_2$	$q_3 q_4$
	q_5	q_1	q_7 X			
	q_6	q_8	q_3			
	q_7	q_1	q_4			
	q_8	q_1	q_3			

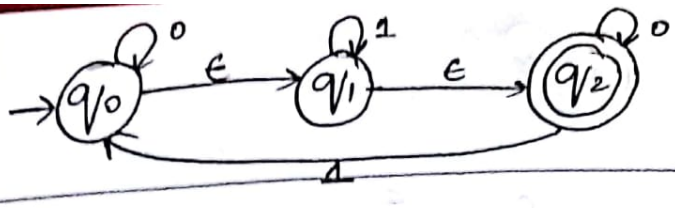
$Q_0 = \{q_1\ q_2\ q_3\ q_4\}$, $\{q_6\ q_7\ q_8\}$

$Q_1 = \{q_1\ q_2\}$, $\{q_3\ q_4\}$ $\{q_6\}$ $\{q_7\ q_8\}$

27 Aug'18

NFA with ε - Transition

If A finite automata is transitive w/o input symbols, then we get a NFA with null transitions. Transitions w/o symbols are called ε transitions



It is possible to make transition from q_0 to q_1 w/o consuming input symbol and finite automata a will be denoted by

$(Q, \Sigma, \delta, q_0, F)$ → usual meaning δ is $Q \times (\Sigma \cup \epsilon) \to 2^Q$

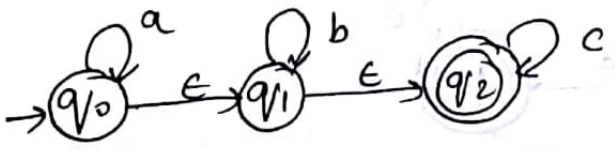
(*) Acceptance of a string by NFA with ϵ- Transition
A string w in S^K will be accepted by NFA with ϵ-transitivity if there exists atleast one path corresponding to which starts in an initial state and ends in of the final state. Since this path may be formed by ϵ-transition as well as non ϵ-transition.

To find out whether w is accepted or not by the NFA with ϵ-moves - we require to define a function ϵ-closure(q) where q is the state of the automata.

ϵ-closure(q): It is a set of all states we reach by following the transition function from the given state. that are labelled ~~in~~ ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$
ϵ-closure $(q_1) = \{q_1, q_2\}$
ϵ-closure $(q_2) = \{q_2\}$

Illustⁿ Conversion of NFA with ϵ-transition to NFA w/o ϵ-conversion

$(Q, \Sigma, \delta, q_0, F)$ $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b, c\}$ and ϵ-moves
q_0 – initial state
$F = \{q_2\}$



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

Find the states of NFA w/o ϵ-transitions including initial-states as follows :-

Rule 1 (a) Language accepted by above NFA with ϵ-transitions is a set of strings over a, b, c followed by any no. of b's and c's.

(b) There will be same no. of states but the names can be constructed by writing names as set of states in the ϵ-closure

(c) Initial state will be ϵ-closure of initial state of NFA

ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

new initial state w/o ϵ transitions

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

(d) Final state of NFA w/o ϵ transitions are all these new states which contain all states of NFA with ϵ-transitions as NFA.

NFA without ϵ-transition

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

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

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

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

δ' = graph

(•) Consider each old machine transition in each row, we include ϵ-entry in ϵ-transition column. In the old machine we
have $\delta(q_0, a) = q_0$

$M' = \delta_0'(\{q_0 q_1 q_2\}, a) = \{q_0 q_1 q_2\}$

This is because the new machine accept the same language must atleast have the same transition for the new state names.

$\delta'(\{q_0, q_1 q_2\}, a) = \epsilon\text{-}c(\delta(q_0, q_1, q_2), a)$

$= \epsilon\text{-}c(\delta(q_0, a), \delta(q_1, a), \delta(q_2, a))$

$= \epsilon\text{-}c(q_0 \cup \phi \cup \phi) = \epsilon\text{-}c(q_0)$

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

$\delta'(\{q_0, q_1 q_2\}, b) = \epsilon\text{-}c(\delta(q_0, q_1, q_2), b)$

$= \epsilon\text{-}c(\delta(q_0, b), \delta(q_1, b), \delta(q_2, b))$

$= \epsilon\text{-}c(q_1) = \{q_1, q_2\}$

$\delta'(\{q_0 q_1 q_2\}, c) = \epsilon\text{-}c(\delta(q_0, q_1, q_2\}, c)$

$= \epsilon\text{-}c(\phi \cup \phi \cup q_2\} = \{q_2\}$

(•) Similarly,

$\delta'(\{q_1 q_2\}, a\} = \epsilon\text{-}c(\delta(q_1 q_2), a) = \phi$

$\delta'(\{q_1, q_2\}, b) = \epsilon\text{-}c(\delta(q_1, q_2), b) = \{q_1, q_2\}$

$\delta'(\{q_1, q_2\}, c) = \epsilon\text{-}c(\phi \cup q_2) = \{q_2\}$

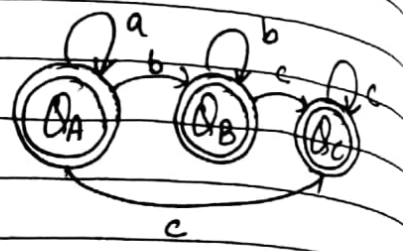
$\delta'(\{q_2\}, a) = \phi$

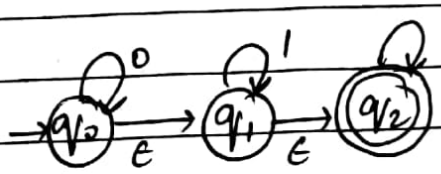
$\delta'(\{q_2\}, b) = \phi$

$\delta'(\{q_2\}, c) = \epsilon\text{-}c(q_2) = \{q_2\}$

Using ϵ-closure, NFA w/o Transition

Q | Σ	a	b	c
Q_A	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
Q_B	$\{q_1, q_2\}$	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
Q_C	$\{q_2\}$	ϕ	ϕ	$\{q_2\}$



Ques 

Convert the given NFA to NFA without ϵ-transitions

$\epsilon\text{-}C(q_0) = \{q_0, q_1, q_2\}$

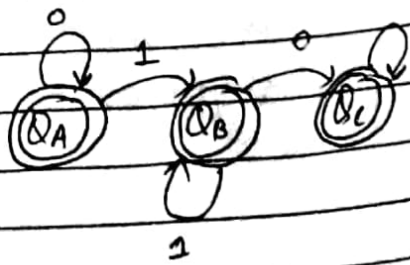
$\epsilon\text{-}C(q_1) = \{q_1, q_2\}$

$\epsilon\text{-}C(q_2) = \{q_2\}$

- $\delta'(\{q_0 q_1 q_2\}, 0) = \delta' \epsilon = \epsilon\text{-}C(\delta(q_0, q_1, q_2), 0)$
  $= \epsilon\text{-}C(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)) = \epsilon\text{-}C(q_0 \cup \phi \cup q_2)$
  $= \{q_0, q_1, q_2\}$
- $\delta'(\{q_0, q_1, q_2\}, 1) = \epsilon\text{-}C(\phi \cup q_1 \cup \phi) = \{q_1 q_2\}$
- $\delta'(\{q_0, q_2\}, 0) = \epsilon\text{-}C(\phi \cup q_2) = \{q_2\}$
- $\delta'(\{q_1, q_2\}, 1) = \epsilon\text{-}C(q_1 \cup q_2) = \{q_1, q_2\}$
- $\delta'(\{q_2\}, 0) = \epsilon\text{-}C(q_2) = \{q_2\}$
- $\delta'(\{q_2\}, 1) = \phi$

Answer

Q | Σ	0	1
Q_A	$\{q_0 q_1 q_2\}$	$\{q_0 q_1 q_2\}$	$\{q_1 q_2\}$
Q_B	$\{q_1 q_2\}$	$\{q_2\}$	$\{q_1, q_2\}$
Q_C	$\{q_2\}$	$\{q_2\}$	ϕ



FORMAL LANGUAGES

1) GRAMMAR

A phase structure grammar is a set comprising of (V_N, Σ, P, S) where

- V_N is a finite non empty set whose elements are called variables
- Σ is " " " " " " " " terminals
- S is a special variable called the start symbol and it is an element of V_N
- P is a finite set whose elements are $\alpha \rightarrow \beta$ where α and β are strings on $V_N \cup \Sigma$, α as atleast one symbol from V_N. Elements of P are Productions or Production Rules or Re-writing Rules.

* If $\alpha \rightarrow \beta$ is a production in the grammar g and γ, δ are any 2 strings in $V_N \cup \Sigma$, we say that $\gamma\alpha\delta$ directly derives $\gamma\beta\delta$, i.e

$$\gamma\alpha\delta \overset{G}{\Rightarrow} \gamma\beta\delta$$

This process is called one-step derivation.

* The lang generated by grammar g denoted by L_G is defined as

$$\{ w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

(* → many production rules)

L_G is the set of terminal strings derived from the start symbol S.

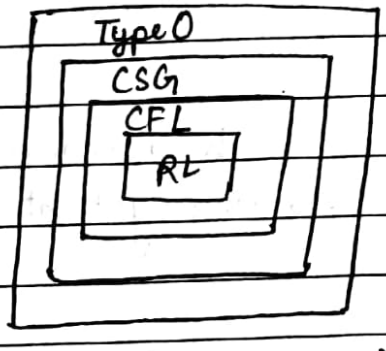
* If $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$ then α is called sentential form. We can write that the elements of L_G are sentential forms.

* G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$

* The string generated by the most recent application of production is called the working string.

* The derivation of a string is complete when the working string cannot be modified. If the final string does not contain any variable, it is called as the sentential form.

(•) Chomsky classification of Languages



① Type 0 → Turing m/c
② Type 1 → Context Sensitive (LBA)
③ Type 2 → Context free (PDA)
④ Type 3 → Regular Grammar
FA → DFA
→ NFA

In the def^n of $G(V_n, \Sigma, P, S)$ V_n & Σ are sets of symbols and $S \in V_n$, so if we want to classify grammars we have to do it only by considering the form of productions

→ Chomsky classified the grammar into 4 types in terms of productions

1) Type 0 grammar (Recursively enumerable grammar)

• Type 0 grammar is any phase structure grammar w/o any restriction. The production of the form

$$\phi A \psi \longrightarrow (\phi \alpha \psi)$$

Replacement string

A → variable; ϕ → left context; ψ → right context

① eg: ab**A**bcd → ab**AB**bcd

~~AB~~ ab → L.C bcd → R.C $\alpha = AB$

② eg: $AC \longrightarrow A$

$A : LC$
$\wedge : RC$
$\alpha \rightarrow \wedge$

The production simply erases C when left content is A and right content is $\wedge$.

2) Type 1

- A production of the form :-

$\phi A \psi \longrightarrow \phi \alpha \psi$ is called type 1 production when $\alpha \neq \wedge$

In Type 1 productions, erasing of A is not permitted.

① eg $AB \longrightarrow AbBc$

$A : LC$
$\wedge : RC$
$\alpha = bBC \neq \wedge$

- A grammar is called Type 1 or content sensitive or context dependent if all its productions are type 1 productions. The production $S \longrightarrow \wedge$ is also allowed in Type 1 grammar but in this case S does not appear on the right hand side of any production

$S \longrightarrow aA/\lambda$	✓
$S \rightarrow aAS/\wedge$	X

- The language generated by Type 1 grammar is called content sensitive language.

3) Type 2

- Type 2 production is a product^n of the form

$A \longrightarrow \alpha$

where A is an element of V_n and α is the element of $(V_n \cup \Sigma)^*$ or in other words LHS has no left content and right content

① eg $S \rightarrow Aa$
$A \rightarrow a$
$B \rightarrow abc$
$A \rightarrow \Lambda$

- A grammar is called Type 2 if it contains only Type 2 productions. It is called as content free grammar.
- A language generated by content free grammar is called Type 2 lang or context free lang.

4) Type 3

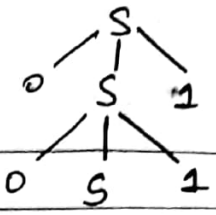
- A production of the form
$A \xrightarrow{P} aB$
$A, B \in V_n$
$a \in \Sigma$ is called a type 3 production
- if the production rules are of the form in which RHS contains atmost one non terminal symbol is called type 3 grammar.
- The language generated by Type 3 grammar are recognized FSM

These grammars are identical to Regular expression grammar

Formal Languages

Q If $G = (\{S\}, \{0, 1\}, \{S \rightarrow 0S1, S \rightarrow \Lambda\}, S)$
find $L(G)$ Λ is also in $L(G)$

$S \Rightarrow_G 0^n 1^n$



$0^n 1^n \in L(G)$ for $n \geq 0$

$\Rightarrow L(G) \subseteq \{ 0^n 1^n \mid n \geq 0 \}$

Q. Let $G = (\{S, A_1, A_2\}, \{a, b\}, P, S)$

P $S \rightarrow aA_1A_2a$, $A_1 \rightarrow baA_1A_2b$, $A_2 \rightarrow A_1ab$, $aA_1 \rightarrow baa$

$bA_2b \rightarrow abab$

Test whether $w = baababaaaabbaba$

Soln

$S \Rightarrow \underline{a}A_1A_2\underline{a}$

$baa\underline{A_2}a$

$baa\underline{A_1}aba$

$baa\,b\underline{aA_1}A_2baba$

$baabbaaA_2baba$

$baabbaa\underline{A_1}abbaba$

$baababaaabbaba$

Q. $L_1 = \{ a^n b^{2n} \mid n \geq 1 \}$

Construct a grammar generating the Lang L_1

$n=1$ $L_1 = abb$

$n=2$ $L_2 = aabbbb$

$L_3 = aaabbbbbb$

$G_1 = (\{S\}, \{a, b\}, P, S)$

where P is $S \rightarrow aSbb$ $S \rightarrow abb$

Q $L_2 = \{ a^m b^n \mid m > n,\ m, n \geq 1 \}$

Construct a grammar generating the Lang L_2

$m=2\ n=1$ aab

$m=3\ n=2$ aaabb

$m=4\ n=3$ aaaabbb

$S \rightarrow aS \mid aA$

$A \rightarrow aAb$ $A \rightarrow ab$

There is always a unique way

Q1. $L = \{ a^n b^n c^i \mid n \geq 1, i \geq 0 \}$

Q2. $L = \{ a^j b^n c^n \mid n \geq 1, j \geq 0 \}$

Find the grammar generating these lang

Q3. $L = \{ a^n b^n c^n \mid n \geq 1 \}$

Q4. Construct a grammar which generates all even integers upto 998

Q5. Find L(G) for $G = (\{S, A\}, \{0, 1\}, P, S)$ where P is $S \to 0A \mid 1S \mid 0 \mid 1$

$A \to 1A \mid 1S \mid 1$

4. ~~2 4 8~~ Ans 1. $a^n b^n c^i$ ($a^n b^n$ = A, c^i = B)

$S \to AB$

$A \to aAb \mid ab$

$B \to cB \mid \epsilon$

$B = \epsilon, c, cc, ccc$ $A = ab, a(ab)b$ $S \to AB$

$B \to cB \mid \epsilon$ $A \to aAb \mid ab$ Combine both

Ans 2.

$S \to AB$

$A \to aA \mid \epsilon$

$B \to bBC \mid bc$

Ans 3.

$S \to ABC$

$A \to aA \mid a$ ✗

$B \to bB \mid b$

$C \to cC \mid c$

(•) Regular Expressions

The Language accepted by finite auto -mata is described by simple expressions called regular expressions. A regular lang over an alphabet Σ is one that can be obtained from these language using the operations of Union, Concatenation, Kleene*

(•) Regular Set: A set represented by a regular exp^{r} is called a regular set.

eg: $a, b \in \Sigma$ (i) $\{a\}$, (ii) $a+b$ $(\{a, b\})$

(iii) ab $\{ab\}$

(iv) a^* $\{\wedge, a, aa, \ldots\}$

The set represented by R is denoted by L(R)

Q Describe the following sets by regular expressions

1) $\{101\}$ $\{1\}$ $\{0\}$

↓ ↓

1 0

101 is obtained by concatenating 1, 0, 1

2) $\{abba\}$ abba

3) $\{01, 10\}$ $\{01\}$ $\{10\}$ $01+10$

4) $\{\wedge, ab\}$ $\wedge + ab$

5) Any element in this set is obtained by concatenating 1 to existing :-

$\{1, 11, 111, \ldots\} \Rightarrow 1(1)^*$

We have the following operations on Reg. expressions.

① Union $L_1 = \{ab, ba\}$
$L_2 = \{bb, aa\}$ $L_1 \cup L_2$

② Concatenation L_1L_2 , L_2L_1

A regular lang can be described by an explicit formula. We think of a regular exp^n as representing the most typical string in the corresponding lang.

③ Kleene*

(•) Identities for Regular Exp^n

Two regular exp^n P & Q are equivalent if P and Q represent the same set of strings

1) $\phi + R = R$
2) $\phi R + R\phi = \phi$
3) $\wedge R = R\wedge = R$
4) $\wedge^* = \wedge$ and $\phi^* = \wedge$
5) $R + R = R$
6) $R^*R^* = R^*$
7) $RR^* = R^*R$
8) $(R^*)^* = R$
9) $\wedge + RR^* = R^* = \wedge + R^*R$
10) $(PQ)^*P = P(QP)^*$
11) $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$
12) $(P+Q)R = PR + QR$ | $R(P+Q) = RP + RQ$

(•) Ardenis Theorem

let P and Q be two regular exp^n over Σ, if P does not contain NULL ($\wedge$) then the following eq^n in R has a unique sol^n given by

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

(o) Proof

$R = QP^*$

$Q + (QP^*)P = Q(\wedge + P^*P)$

$= QP^*$

By identity 9, hence 1st is satisfied when $R = QP^*$

Therefore, $R = QP^*$ is a soln of first.

To proove uniqueness consider first here by replacing R by $Q+RP$ on the RHS we get

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

$= Q + QP + RP^2$

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

$= Q(\wedge + P + P^2 + \ldots + P^i) + RP^{i+1}$

from ①

$R = Q(\wedge + P + P^2 + \ldots + P^i) + RP^{i+1}$ for $i \geq 0$ ↳ ②

We now show that any soln of ① is eq. to QP^*.

Suppose R satisfies ① then it satisfies ②. Let w be a string of length i in the set R. Then w belongs to the set

$Q(\wedge + P + P^2 + \ldots + P^i) + RP^i + 1$

As P does not contain NULL, $\therefore RP^i + 1$ has no string of length less than $i+1$, so w is not in the set $RP^i + 1$. This means w belongs to the set

$Q(\wedge + P + P^2 + \ldots + P^i) + RP^i + 1$

$\downarrow$

QP^*

Consider a string w in QP^*, then w is in the set QP^k for some $k \geq 0$ and hence in $Q(0+P+P')$. So w is in the RHS of (2)
So w is in R. Thus R and QP^* represent the same set.

⇒ Proves the uniqueness of (1)

Prove

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

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

~~$10+110^*1+00^*10+00^*+10^*1$~~

Taking common,

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

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

$\Lambda + R^*R = R^+$

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

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

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

(•) <u>Closure Properties of Regular sets</u> (Imp for Exam)

<u>Thm</u> The regular sets are closed under union concatenation and Kleen closure (Here closed means if a and b belong to set A and a+b also belong to set A) Then A is closed under addition

operation.

Union R_1 and R_2 denote any 2 reg exp^n
Any string in $R_1 + R_2$ is a string from R_1 or R_2
Thus the set represented by $R_1 + R_2$ is union of sets represented by R_1 and R_2 Hence by def^n regular sets are closed under Union.

#Concat^n If R_1 and R_2 denote any 2 reg exp^n, A string in R_1R_2 is a string from R_1 followed by a string from R_2 Thus the set represented by R_1R_2 is the concatenation of sets represented by R_1R_2. Hence by def^n regular sets are closed under concatenation

#Keen clos Let R denote a reg exp^n. The string in R^* is a string obtained concatenating elements for some $n \geq 0$. Thus the set represented by R^* is $(w_1 w_2 \ldots w_n) w_i$ is the set represented by R and $n \geq 0$.
Hence by def^n regular sets are closed under Keen closure.

⇒ Here define Kleeneclosure, Union, Concatenation and everything.