MODULE 2

REGULAR EXPRESSION

→ The languages accepted by finite state machine are regular. And these languages can be represented by regular expression.

A regular expression is a string that can be formed according to the following rules

1. ϕ is a RE
2. ϵ is a RE
3. Every element in Σ is a regular expression (RE).
4. Given two regular expressions α & β, $\alpha\beta$ is a RE
5. Given two RE α and β, $\alpha \cup \beta$ is a RE
6. Given a RE α, α^* is a RE {ex: ϵ, a, aa, ab, . . }
7. Given a RE α, α^+ is a RE {ϵ is not included in α^+}
8. Given a RE, (α) is a RE

$$\alpha^* = \epsilon \cup \alpha^+$$

Find the regular expression for the following languages over the alphabet $\Sigma = \{0, 1\}$

1. Strings of 0's & 1's with two consecutive zeroes

$L = \{00, 001, 100, 1001, 11001, 101001, 110010, . .\}$

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

+ union
* repetation zero or more no of

2. strings of 0's & 1's ending with 011

$RE = (0+1)^* 011$

3. With no consecutive zeros

$\{0, 01, 10, 010, 110, 01011, 011010, 1, \epsilon . .\}$

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

4 Length of the string is exactly 3

RE = (0+1)(0+1)(0+1)

L = {000, 001, 010, 011, ...}

5 All 0's comes before all 1

RE = (0)*(1)*

6 String begin with zero & end with one 0

0(1)*0

RE = 0(0+1)*0

7 Strings that have atleast 2 0's

~~RE = (0+1)~~

L = {00, 100, 001, ~~1010~~, ...}

~~RE~~ (0+1)*00(0+1)* for consecutive

RE = (0+1)*0(0+1)*0(0+1)*

8 String that have exactly two 0's

RE = (1)*0(1)*0(1)*

9 Set of strings where 5th symbol from right hand is 1

RE = (0+1)*1(0+1)*(0+1)(0+1)(0+1)

10 Set of strings consisting atleast 1 a followed by string consist of atleast 1 b & string consisting atleast 1 c

RE = (a+b+c)*a(a+b+c)*b(a+b+c)*c(a+b+c)*

Page No.
Date: 12 | 11 | 21 YOUVA

Operations of RE:

1. Union

The union of two languages L and M denoted by LUM is set of strings that are either in L or M or both

eg: L = {001, 10, 111}

M = {ϵ, 001}

LUM = {ϵ, 10, 001, 111}

2. Concatenation

The concatenation of languages L and M is the set of strings that can be found by taking any string in L & concatenating with any string in M

LM = {001, 10, 111, 001001, 10001, 111001}

3. closure (Star or Kleene closure)

The closure of a language L denoted by L^* and it represents the set of strings by taking any number of string from L and concatenating them.

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

L^* is the string of 0's & 1's

$L^* = \{\epsilon, 0, 1, 01, 11, 011, \ldots\}$

$L = \{0, 11\}$

$L^* = \{\epsilon, 0, 11, 011, 1111, 1100, \ldots\}$

Note:

$\phi^* = \{\epsilon\}$

$\phi^0 = \{\epsilon\}$

$L^* = \bigcup_{i \geq 0} L^i$ or

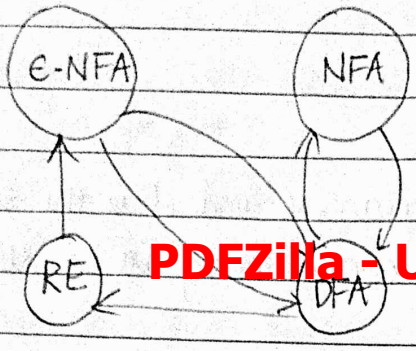
$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots$

$L^+ = \bigcup_{i \geq 1} L^i = L^1 \cup L^2 \cup L^3 \ldots$

Order of prudence

1. *
2. .
3. +

Equivalences



Built FSM from RE:

Conversion of RE to ϵ NFA:

Every language defined by regular expression is also defined by Finite Automata. Suppose L = L(R) for RE R,

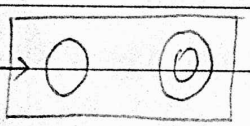
L = L(M) for some ϵNFA

(i) exactly 1 accepting state,

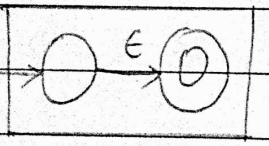
(ii) No arcs into the initial state

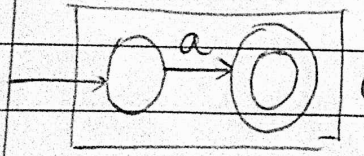
(iii) NO arcs out of accepting state

ϵ-NFA to

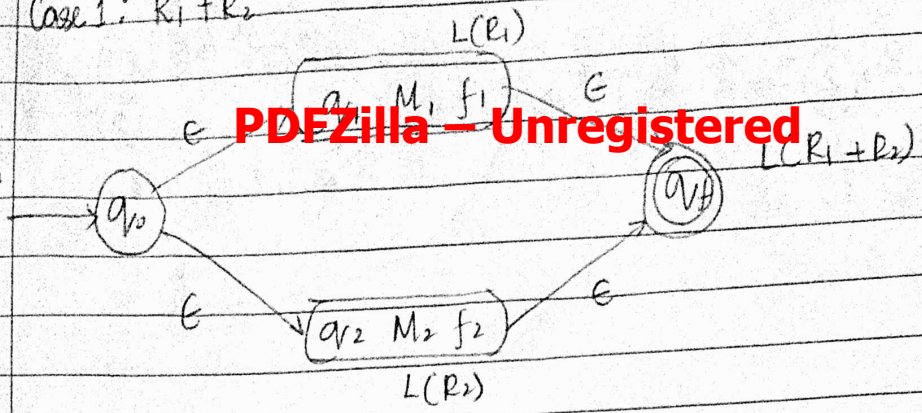
 ϕ Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ be FA - $L(R_1)$

$M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ FA - $L(R_2)$

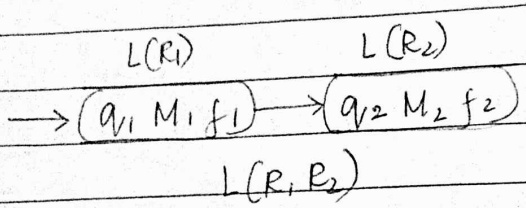
 ϵ

 a

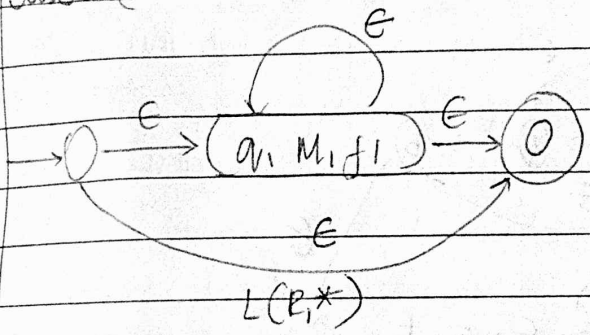
Case 1: $R_1 + R_2$



Case 2: $R_1 \cdot R_2$

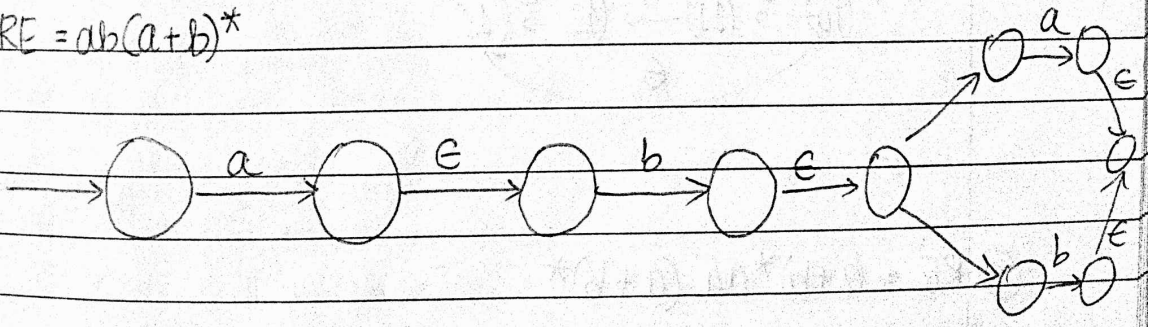


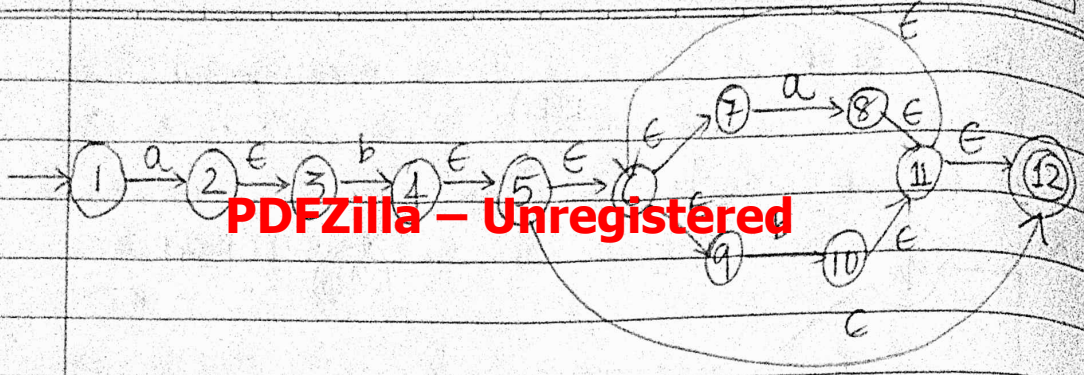
Case 3: $(R_1)^*$



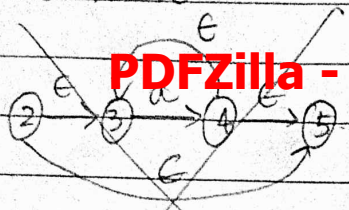
Construct ϵ-NFA for the following regular expression.

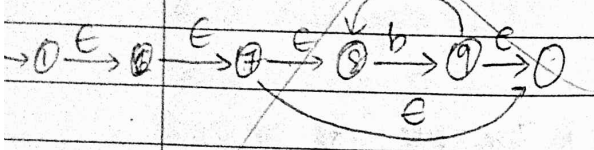
① RE = $ab(a+b)^*$

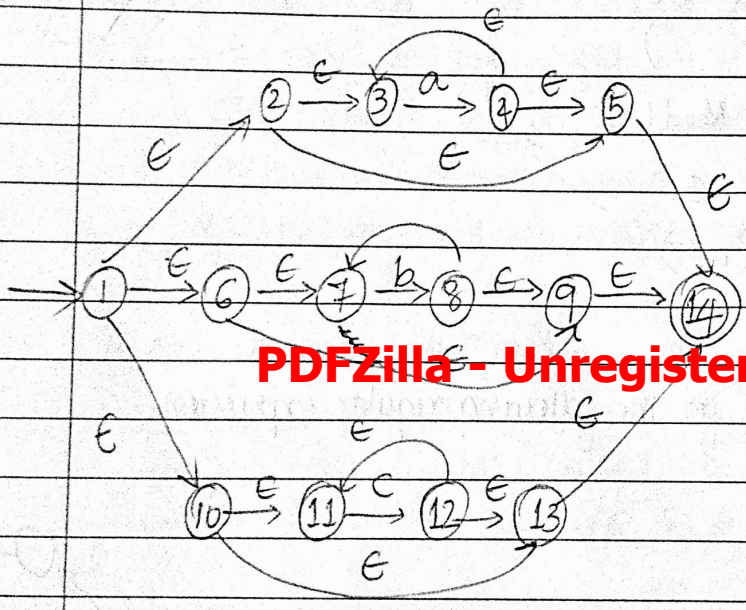




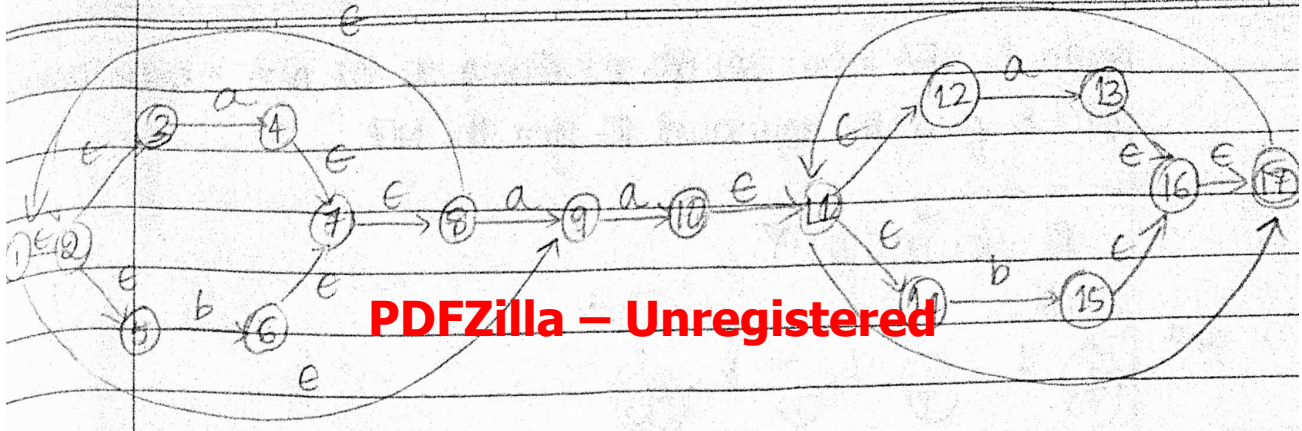
(2) RE = $a^* + b^* + c^*$

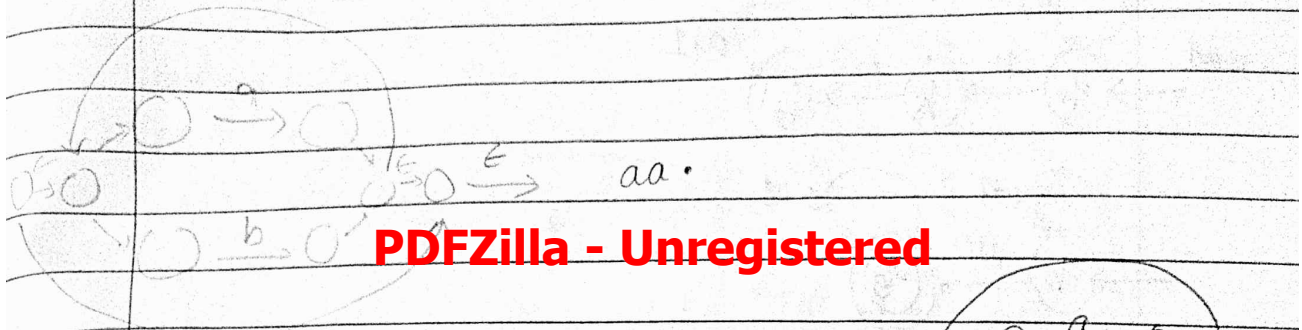


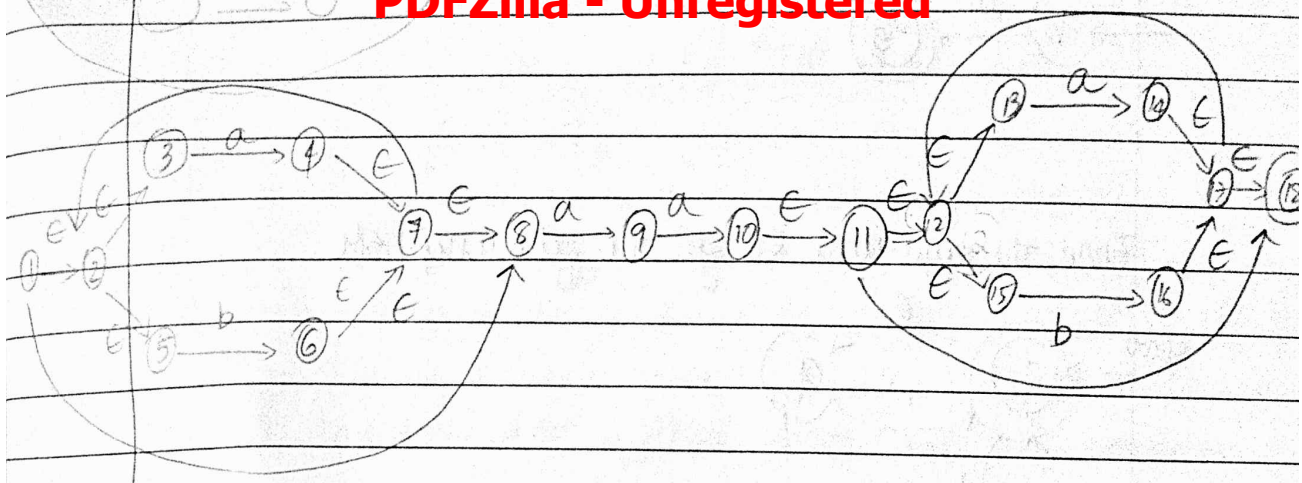




(3) RE = $(a+b)^* aa (a+b)^*$

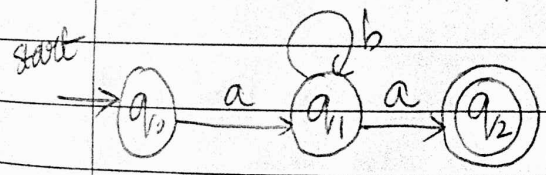


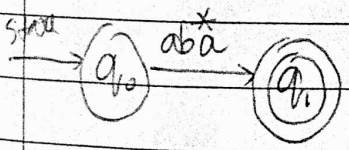




15/11/21 From FSM to RE:

Write the equivalent regular expression for the following FSM

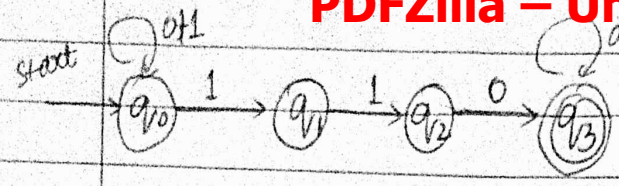


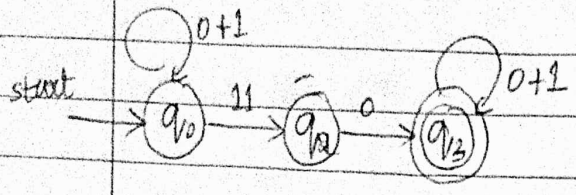


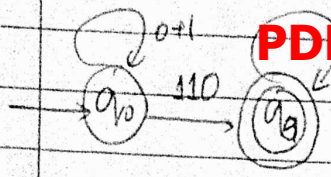
Remove the state q_1 & try to write equivalent expression for q_0 & q_2

Construct NFA which accepts all strings of 0's and 1's containing 110 & find the equivalent RE from the NFA

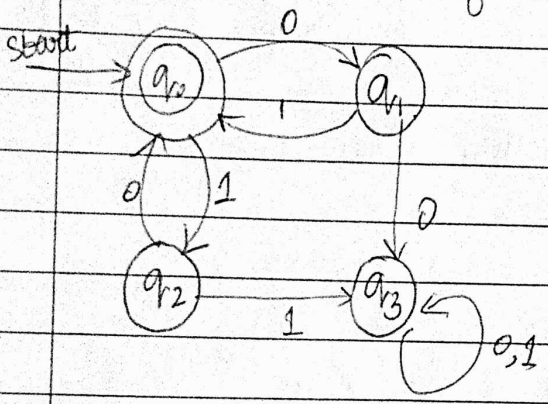
$RE = (0+1)^*110(0+1)^*$

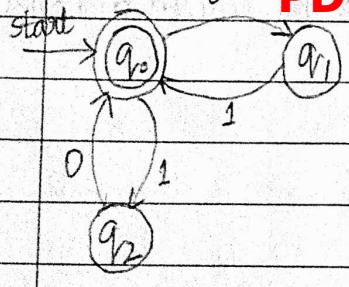




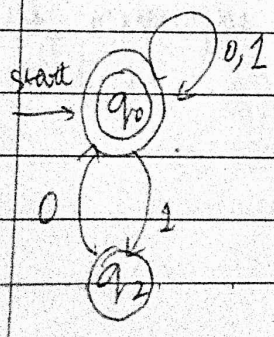


Find the equivalent RE for the following FSM

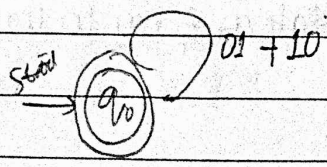




Remove q_1



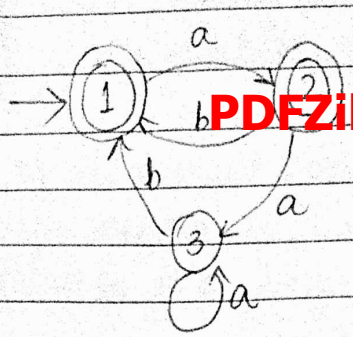
Remove q_2



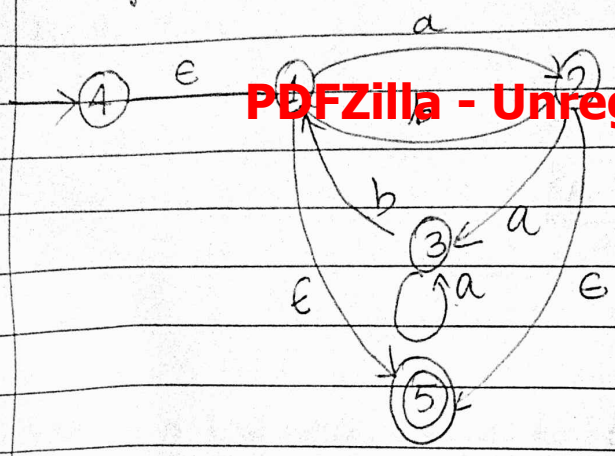
$RE = (01+10)^*$

Date: YOUVA

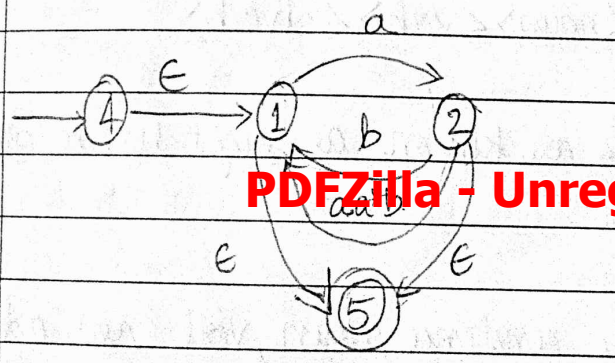
Build RE from the following FSM

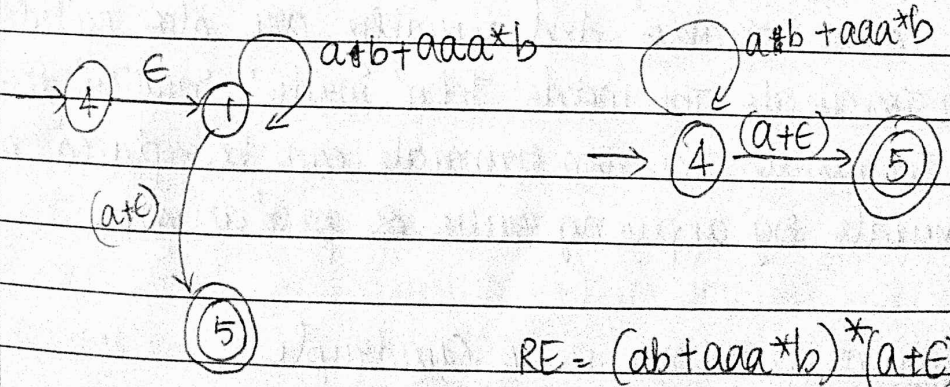


Adding states



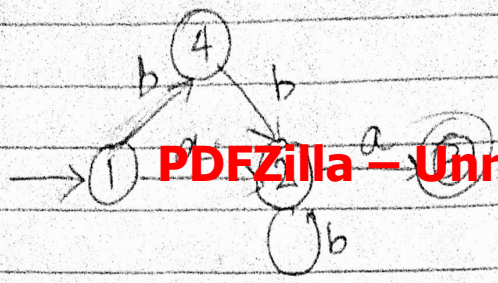
Create a new start state 4 & accepting state 5 & link them to M.

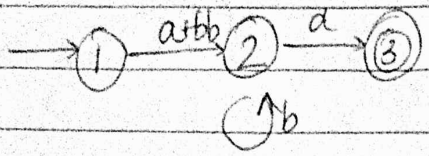


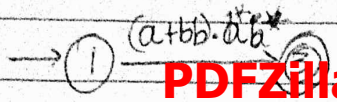


$RE = (ab+aaa^*b)^*(a+\epsilon)$

Build RE from the fdl FSM







$RE = (a+bb)\cdot b^* \cdot a$

26|11|21 Regular Grammers:

Right linear Grammers

John writes neatly
noun verb adverb

<sentence> → <noun><verb><adverb>

Regular grammers are known as right linear grammers
Consider

In the above example sentence, noun, verb and adverb are called variables. And variables are also called as non terminals. The words 'John' 'writes', 'neatly' these are terminals. The non terminals can be replaced by terminals by applying rules or production.

∴ Regular Grammer is a Quadruple

(V, Σ, R, S)

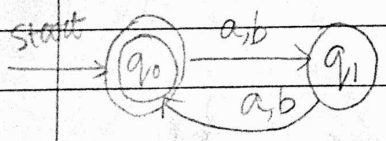
where V - set of variables or non terminals
Σ - set of terminals or is set of rules
S - start symbol & it is a non-terminal

All rules must follow
1. LHS should be a single non terminal
2. RHS is ∈ or single terminal followed by non terminal
3.

Legal Rules
S → a
S → ∈
T → aS

Not legal
S → aSa
S → TT
aSa → T
S → T

① Consider a language L = {w ∈ {a,b}* : |w| is even}
L = {aa, bb, ab, ba, aabb, abab, baba, bbaa, ... }



RE = ((a+b)(a+b))*

G = (V, Σ, R, S)
V = {S, T}
Σ = {a, b}

S → ∈
S → aT/bT
T → aS/bS

eg: aabb

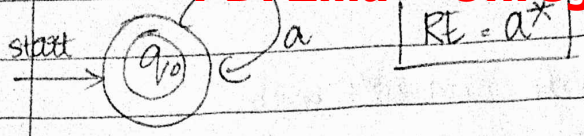
S ⇒ aT
⇒ aaS
⇒ aabT
⇒ aabb

S ⇒ aT
⇒ abS
⇒ abaT
⇒ ababS
⇒ abab

② Obtain Grammer to generate string consisting of any no of a's

$L = \{a^n : n \geq 0\}$

$L = \{a, aa, \ldots\}$



RE = a^*

$G = (V, \Sigma, R, S)$

$S \rightarrow aS / \epsilon$

eg: aa

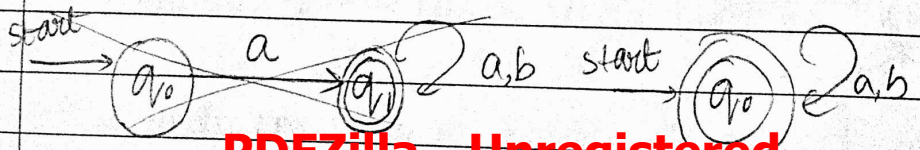
$S \rightarrow aS$ $\{S \rightarrow aS\}$

$\rightarrow aaS$ $\{S \rightarrow aS\}$

$\rightarrow aa$ $\{S \rightarrow \epsilon\}$

③ Obtain Grammer to generate string consisting of any no of a's & b's

$L = \{a, aa, bb, b, abbb, \ldots\}$



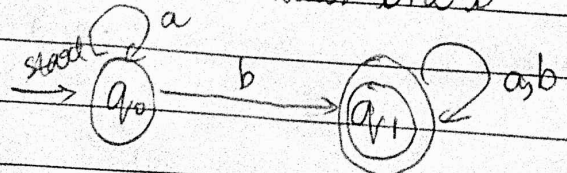
$G = (V, \Sigma, R, S)$

$S \rightarrow aS$

$S \rightarrow bS$

$S \rightarrow \epsilon$

④ Obtain Grammer to generate string consisting of any no of a's & b's with atleast one b



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

$G = (V, \Sigma, R, S)$

$S \rightarrow aS$ $\quad S \rightarrow as/bA$

$S \rightarrow bA$ $\quad A \rightarrow aA/bA/\epsilon$

$\delta(S,b) = A$ $\quad S \rightarrow bA$

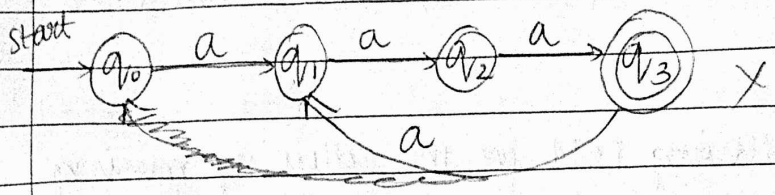
$\delta(A,a) = A$ $\quad A \rightarrow aA$

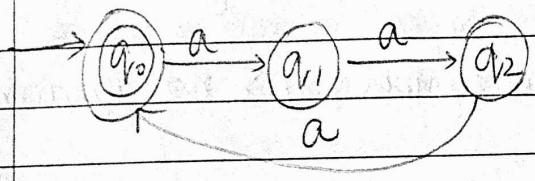
$\delta(A,b) = A$ $\quad A \rightarrow bA$

$A \rightarrow \epsilon$

(5) Obtain Grammer to accept language $L = \{w : |w| \bmod 3 = 0\}$

$L = \{a^3, a^6, \ldots\}$





RE =

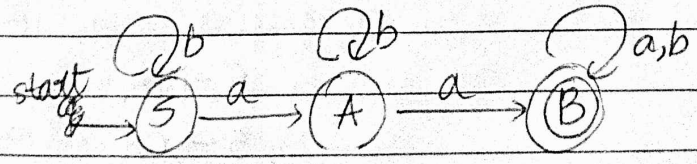
$\delta(S,a) = A$ $\quad S \rightarrow aA$

$\delta(A,a) = B$ $\quad A \rightarrow aB$

$\delta(B,a) = S$ $\quad B \rightarrow aS$

(6) Obtain grammer to generate string consisting of atleast 2 a's where $\Sigma = \{a,b\}$

$L = \{aab, aa, baa, ababa, \ldots\}$



$\delta(S,b) \Rightarrow bS$ $\quad \delta(B,b) = B$

$\delta(S,a) = A$ $\quad \delta(A,b) = A$

$\delta(A,a) = B$ $\quad \delta(S,b) = S$

$\delta(B,a) = B$

$S \to aA$
$A \to aB$
$B \to aB$

$B \to bB$
$A \to bA$
$S \to bS$

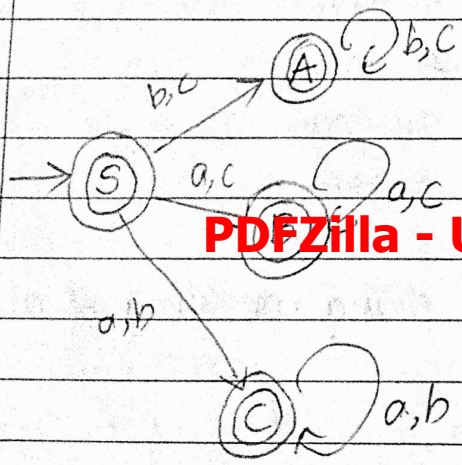
Algorithm from FSM to Grammar:
1. Make M deterministic
2. Create a non terminal for each state
3. Start state becomes start symbol
4. For each transition $\delta(T,a) = V$, $T \to aV$
5. For each accepting state T, make a rule $T \to \epsilon$

24/11/21 Construct following FSM for the following missing letter & also write equivalent languag for it

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

$L_{missing} = \{w : \text{there is a symbol } a \in \Sigma \text{ not appearing in } w\}$

$L = \{ab, ac, bc, ba, ca, cb, abb, bcb, acac, \ldots\}$



$G = (V, \Sigma, R, S)$

$\delta(S,b) = A$
$\delta(S,c) = A$
$\delta(S,a) = B$
$\delta(S,c) = B$
$\delta(S,a) = C$
$\delta(S,b) = C$
$\delta(A,b) = A$

$S \to bA$
$S \to cA$
$S \to aB$
$S \to cB$
$S \to aC$
$S \to bC$
$A \to bA$

$\delta(A,C) = A$ $\quad A \to CA$
$\delta(B,a) = B$ $\quad B \to aB$
$\delta(B,C) = B$ $\quad B \to cB$
$\delta(C,a) = C$ $\quad C \to aC$
$\delta(C,b) = C$ $\quad C \to bC$

$S \to bA/CA/aB/CB/aC/bC/\epsilon$
$A \to CA/bA/\epsilon$
$B \to aB/cB/\epsilon$
$C \to aC/bC/\epsilon$

$L = \{a^n 1^n / n \geq 0\}$

Properties of regular languages:

The class of languages known as regular languages has four and regular languages are the languages accepted by DFA, NFA, ϵ-NFA. The two important properties of regular expression are

1. Closure property
2. Decision property

1. closure Property:

Closure property of regular languages is useful to building complex automata. Using the closure property, we can build language recognizers.

ex: union, intersection etc.

2. Decision Property:

Using this property we can decide whether 2 automata define same language. If so we can minimize the states of automata. The minimization of automata is very important in the design of switching circuit. as no of states are reduced, cost reduced.

Consider the following languages which are not regular

1. $L = \{ w : w \in \{0,1\}^* \}$ and has equal no of 0's & 1's $\}$
2. $L = \{0^n 1^n \mid n \geq 0\}$
3. $L = \{a^i$ | i is prime no$\}$
4. language consisting of matching parenthesis

We can prove that certain languages are not regular using pumping lemma.

State and prove Pumping lemma for regular languages

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a Finite Automata & small n is no of states

L - be the regular language which is accepted by M

Let for every string x in L, there exist a constant n such that $|x| \geq n$ then x can be broken into 3 substring $x = uvw$ satisfy the constraint

$\rightarrow v \neq \epsilon$ is $|v| \geq 1$

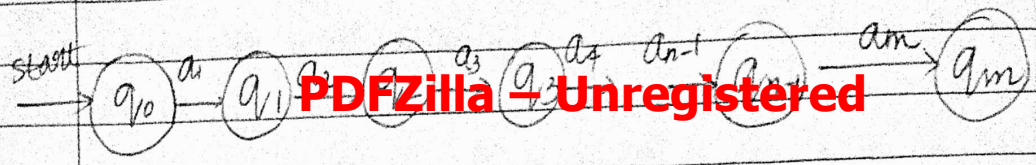
$\rightarrow |uv| \leq n$ then uvw is in L for $i \geq 0$

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a Finite Automata & L is the language accepted by DFA.

Let $x = a_1 a_2 a_3 \ldots a_m$ where $m \geq n$ & $a_i \in \Sigma$

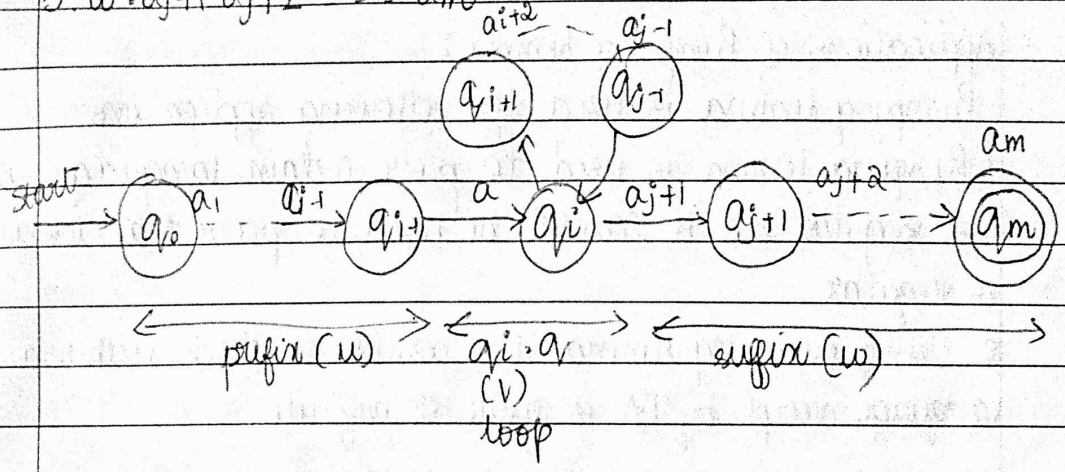
n - no of states

Since we have m input symbols then we should have m+1 states in the sequence $q_0, q_1, q_2 \ldots q_m$ where q_0 is the start state & q_m will be final state as shown below



Since $|x| \geq n$ by the pum pigon hole pumciple it is not possible to have distinct transitions i.e one of the states must have loop. let string x is divided into three substrings as shown below.

1. $u = a_1 a_2 \ldots a_n$
2. $V = a_{i+1} a_{i+2} \ldots a_{j-1} a_j$
3. $w = a_{j+1} a_{j+2} \ldots a_m$



29/11/21 From the above fig prefix string u takes the machine from q_0 to q_i and loops string V takes the machine from $q_i \to q_i$ ($\because q_i = q_i$) and the suffix string w takes the machine from $q_j \to q_s$. The min string that can be accepted by the above DFA which is $uv^i w$ - $i=0$

In general if string x is split into substring

After processing the string

Applications of Pumping lemma:

Pumping lemma is used for following applications.

1. Pumping lemma is used to prove certain languages are not regular but it cannot be used to prove that language is regular.
2. Using pumping lemma it is possible to check wether a language accpt by FA is finite or infinite.

① ST the language $L = \{a^n b^n \mid n \geq 0\}$ is non regular

Assume language is regular and apply pumping lemma

$x = uvw$

$|uv| \leq n$

$v \neq \epsilon \quad |v| > 1$

$x = \underbrace{aa}_{u}\underbrace{a}_{v}\underbrace{bbb}_{w}$

~~$x = vu$~~ $x = uv^i w$, $i = 0, 1, 2, \ldots$

when $i=0$, uw, $aabbb \notin L$

$i=1$, uvw, $(aa)(a)(bbb) \in L$

$i=2$, uv^2w, $(aa)(a)^2(bbb) \notin L$

② The given language $a^{n+1}b^n \notin L$ hence it fails Pumping lemma.
∴ language is not regular.

② ST the language $L=\{a^i b^j \mid i>j\}$ is not regular

$x = uvw$

$L = \{a^4b^3, a^3b^2, a^5b^4, \ldots\}$

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

↓

$\underbrace{a^n}_{u}\ \underbrace{a}_{v}\ \underbrace{b^n}_{w} \Rightarrow a\,a\,a\,a\,b \Rightarrow \underbrace{a_1\ a_2\ a_{n-1}}_{u}\ \underbrace{a_n}_{v}\ \underbrace{a_{n+1}\ b^n}_{w}$

$x = uv^iw \quad , i = 0, 1, 2, \ldots$

when $i=0$, uw, $a_1a_2a_{n-1}ab^n = a^nb^n \notin L$

when $i=0$, the language $a^{n-1}, ab^n \notin L$. Hence language is not regular.

③ ST the language $L=\{ww^R : w\in\{0,1\}^*\}$ is non regular.

w^R is reverse of w

$w = 011$

$w^R = 110$

$ww^R = \underbrace{011}_{w}\underbrace{110}_{w^R}$

Let $w = 0^n1^n$

$w^R = 1^n0^n$

$0^n1^n1^n0^n \Rightarrow \underbrace{0^{n-1}}_{u}\ \underbrace{0}_{v}\ \underbrace{1^n1^n0^n}_{w}$

when $i=0$, uw, $010 \notin L$

$i=1$, uvw, 00110

④ ST the language $L=\{w \mid w\in\{a,b\}^*, n_a(w)=n_b(w)\}$ is non regular

$w=a^nb^n$

Let $w=a^nb^n$ as ur string contain n no of a's & b's

Already proved a^nb^n is not regular

⑤ ST the language $L=\{a^n b^l c^{n+l} \mid n,l\geq 0\}$ is not regular.

$L=\{abcc, aabbbc, ccccc, \ldots\}$

$a^n b^l c^{n+l}$

$\underbrace{a \ldots a^{n-1}}_{u}\ \underbrace{a}_{v}\ \underbrace{b^l c^{n+l}}_{w}$

when $i=0$, $uv=ac$

$i=$

⑥ $L=\{ww \mid w\in\{a,b\}^*\}$

1/12/21

Assignment

4. ST the following languages are not regular using pumping lemma.

1. $L=\{a^n \mid n \text{ is prime}\}$
2. $L=\{b^{i^2} \mid i\geq 1\}$
3. Balanced parenthesis
   $L=\{w\in\{),(\}^*\}$
4. $L=\{a^{n!} \mid n>0\}$
5. $L=\{w \mid n_a(w)<n_b(w)\}$

Date: YOUVA

Closure Property of RE Regular languages (RL):

Union
→ Union of two RL is regular

Intersection
→ Intersection of two RL is regular

→ Compilment of two RL is regular
→ Closure (star) of RL is regular
→ Concatenation of 2 RL is regular
→ Differance of 2 RL is regular
→ Reversal of 2 RL is regular
→ Homomorphism of 2 RL is regular
→ Inverse of 2 RL is regular.

Union, Concatination, & Star Closure are regular by the difinition of RE

If L_1 & L_2 are RL then L_1+L_2 is a RL
By $L_1 \cdot L_2$ i.e concatination of 2 languages is Regular & so L_1 is a RL then L_1^* is also regular.

Complement:

If L is a RL, then compelmunt of L denoted by $\bar{L}$ is regular. let $L = L(M_1)$ · $M_1 = (Q, \Sigma, \delta, q_0, F)$ be DFA that accepts the lang L.

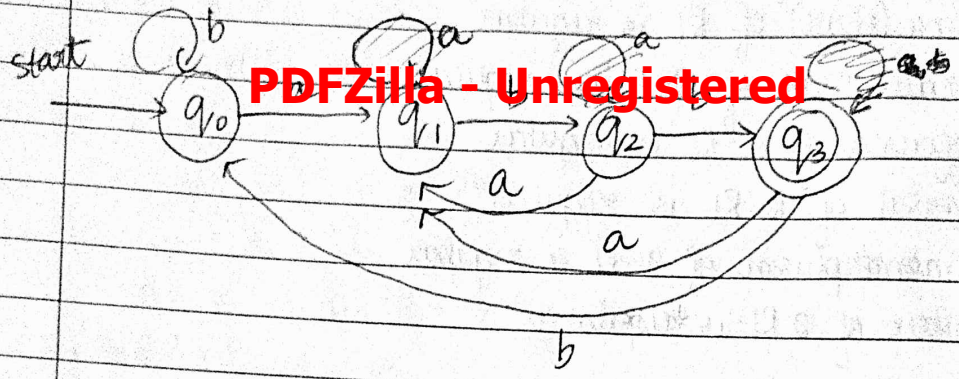
$\bar{L} = L(M_2)$ $M_2 = (Q, \Sigma, \delta, q_0, Q-F)$
M_2 is exactly like M_1 but the final states of M_1 have become non final states of M_2 & vice versa. Thus, we have a DFA which accepts all strings of $\bar{L}$ & rejected by M_1.

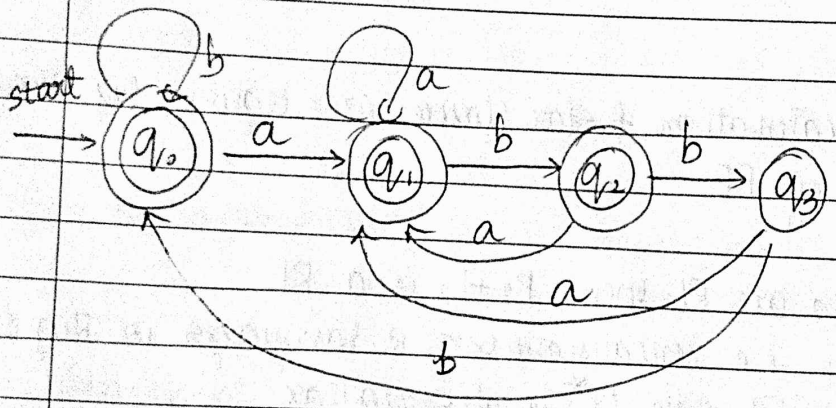
Steps:

1. Convert RE to ϵ-NFA
2. Convert ϵ-NFA to DFA
3. Complement the accepting states of DFA
4. Turn the complement DFA to RE.

Find the complement of DFA ending with abb, $w \in \{a, b\}^*$

$L = \{abb, babb, \ldots\}$





Closure under Intersection:

If L and M are RL, then $L \cap M$ is also RL

Let L be the language accepted by $M_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$

$M_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$

You accept language $L \cap M$, we have to construct DFA that simulates both M_L & M_M, where states of the

machine are the pair (P,q) where $P \in M_1$ and $q \in M_m$

$Q = Q_1 \times Q_2$

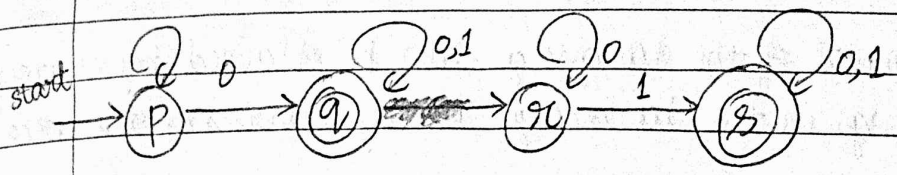
$q_1 = (q_{11}, q_{12})$ where q_{11} is the start state of n_1

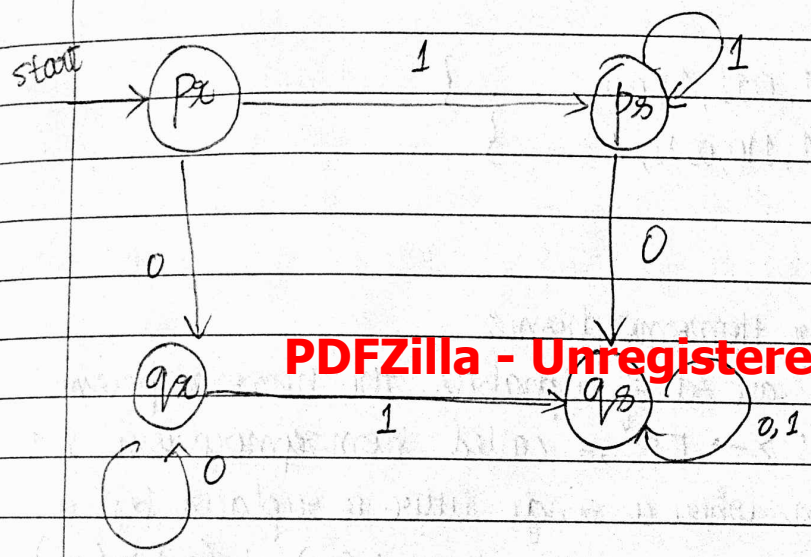
$F = (p,q) \mid p \in F_1$ and $q \in F_2$

$\delta : Q \times \Sigma$ to Q

$\delta((p,q),a) = (\delta_1(p,a), \delta_2(q,a))$

Q Let M_1 be DFA which accepts strings with atleast 1 zero & M_2 be DFA which accepts 0's & 1's with atleast one 1





Closure under difference:

If L_1 & L_2 are RL then the RL is closed under difference i.e $L_1 - L_2$ is also regular

Let us consider $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) - L_1$

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) - L_2$

$L_1 - L_2$: $Q \to Q_1 \times Q_2$

$\Sigma \to$ is same

$\delta : Q \times \Sigma \to Q$

$\delta((p,q),a) = (\delta_1(p,a), \delta_2(q,a))$

$q_0 = (q_{01}, q_{02})$ is the start state

$F = \{(p,q) \mid p \in F_1 \text{ and } q \notin F_2\}$

Closure under Reversal

Let $w = a_1 a_2 \ldots a_n$ & reversal of the string w^R written as $w^R = a_n a_{n-1} \ldots a_2 a_1$

The reversal of the language can be obtained by reversing the strings in the language or by reversing the arcs of DFA.

$L = \{\epsilon, 0, 1, 011, 1100, \ldots\}$

$L^R = \{\epsilon, 0, 1, 110, 0011, \ldots\}$

3/12/21 Closure under Homomorphism:

Let Σ & Γ are set of alphabets, the homomorphism function $h : \Sigma \to \Gamma^*$ is called homomorphism i.e a substitution where a single letter is replaced by a string. If $w = a_1 a_2 \ldots a_n$, then $h(w) = h(a_1) h(a_2) \ldots h(a_n)$

ex: Let $\Sigma = \{a, b\}$ $\Gamma = \{0, 1, 2\}$

$h(a) = 011$

$h(b) = 201$

Date:

Find the homomorphism of string $w = aba$

$h(w) = h(aba)$

$= h(a)\ h(b)\ h(a)$

$= 011\ 201\ 011$

Find the homomorphism of language $L = \{ab, aba\}$

$h(ab) = h(a)\ h(b) = 011201$

$h(aba) = 011201011$

$h(L) = \{h(ab), h(aba)\}$

$= \{011201, 011201011\}$

Find the homomorphism of expression $(a+b)^*(aa)^*$

$h(a) = (011 + 201)^*(011011)^*$