2. Regular expressions & Languages

=> Regular expressions

- An expression of strings which represents regular language is called regular expression.
- The regular expressions are useful for representing certain set of strings in an algebaric fashion.

=> Regular language

- Any set (language) represented by a regular expression is called a regular language. If for example, $a, b \in \Sigma$, then:

① $R = a$ denotes the $L = \{a\}$

② $R = a \cdot b$ denotes $L = \{ab\}$ concatenation

③ $R = a + b$ (OR) denotes $L = \{a, b\}$ union

④ $R = a^*$ denotes the set $\{\epsilon, a, aa, aaa, \ldots\}$ known as kleene closure

⑤ $R = a^+$ Positive closure $\{a, aa, aaa, \ldots\}$

⑥ $R = (a+b)^*$ denotes $\{a, b\}^*$

$(a+b)^3$

$(a+b)$ $(a+b)$ $(a+b)$

b a b => bab

=> Operator Precedence

- The precedence order to solve is

(1) ( ) Bracket
↓
(2) * Kleene closure
↓
(3) + Positive closure
↓
(4) Concatenation
↓
(5) Union

=> Identities for regular expression

- Two regular expression P and Q are equivalent, if P and Q represent the same set of strings.

- Every regular expression can generate only one regular language but, a regular language can be generated by more than one regular expression

- Two regular expression are said to be equal if they generate same language.

for eg: $r_1 = a^*$

$r_2 = a^* + (aa)^*$

Q) State the language

(1) $R = \{a\}$
$\Rightarrow L = \{a\}$

(2) $R = \{a + b\}$
$\Rightarrow L = \{a, b\}$

(3) $R = \{a + b + c\}$
$\Rightarrow L = \{a, b, c\}$

(4) $R = \{a . b\}$
$\Rightarrow L = \{ab\}$

(5) $R = \{a \cdot b + a\}b$
$\Rightarrow L = \{abb, ab\}$

Page No.:
Date:

Q1) Design a regular expression that represents a language 'L' where $L = \{a\}$ over the alphabet $\Sigma = \{a\}$

=> a

Q2) Design a regular " " " all strings over the alphabet $\Sigma = \{a, b\}$, where every accepted string 'w' starts with substring $s =$ 'abb'

=> abb $(a+b)^*$

Q3) Design a regular expression " " " .... where every accepted string 'w' ends with substring $s =$ 'bab'

=> $(a+b)^*$ bab

Q4) Design a regular expression " " " " .... where every accepted string 'w' contains sub string $s =$ 'aba'.

=> $(a+b)^*$ aba $(a+b)^*$

Q5) Design a regular expression ..... such that every accepted string start and end with a.

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

Q6) Start and end with same symbol

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

Q7) Start and end with different symbol

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

Page No.:
Date:

Q8) Every accepted string w is like $w = sx$,
$s =$ 'aaa / bbb'
$\Rightarrow (aaa + bbb)(a+b)^*$

Q9) Every accepted string w, is like $w = xs$,
$s =$ "aaa / bbb"
$\Rightarrow (a+b)^* (aaa + bbb)$

Q10) Every accepted string w, is like $w = xsx$,
$s =$ "aaa / bbb".
$\Rightarrow (a+b)^* (aaa + bbb) (a+b)^*$

Q11) Every string 'w' accepted must be like

(i) $|w| = 3$ (exactly length 3)
$\Rightarrow (a+b)^3$

(ii) $|w| >= 3$ (at least length 3)
$\Rightarrow (a+b)^3 (a+b)^*$

(iii) $|w| <= 3$ (at most length 3)
$\Rightarrow (a + b + \epsilon)^3$

OR

$\epsilon + (a+b) + (a+b)^2 + (a+b)^3$

Q12) Every accepted string 2^{nd} from left end is always b
$\Rightarrow (a+b)\, b\, (a+b)^*$

Q13) Every accepted string 4^{th} from right end is always a.
$\Rightarrow (a+b)^*\, a\, (a+b)^3$

Page No.:
Date:

q14) Every string 'w' where $|w| = 0 \pmod 3$?
$\Sigma = \{a, b\}$
$\Rightarrow [(a+b)^3]^*$

q15) Every string 'w' where $|w| = 3 \pmod 4$
$\Rightarrow (a+b)^3 [(a+b)^4]^*$
remainder 3 ↑ ↓ mod 4

q16) $|w|_a = 0 \pmod 3$
$\Rightarrow b^*(b^* a\, b^* a\, b^* a\, b^*)^*$
mod 3

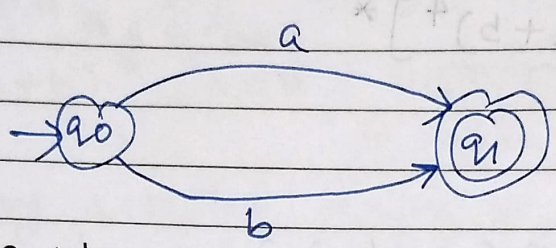
q17) $|w|_b = 2 \pmod 3$
$(a^* b\, a^* b\, a^* b\, a^*)^*$ $a^* b\, a^* b\, a^*$
mod 3 ↓ 2

Page No.:
Date:

⇒ Conversion from finite automata to regular expression.

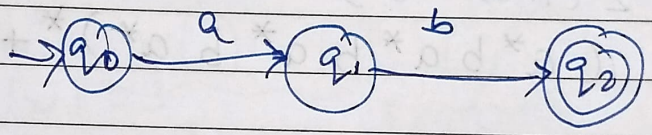
Q) Write regular expressions for the following machines

(1)



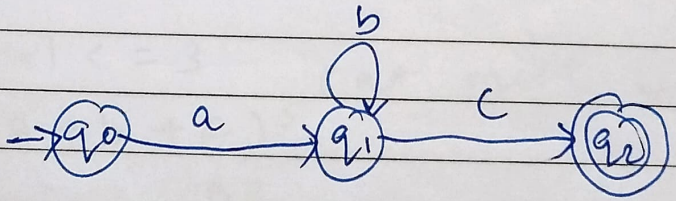
⇒ a+b

(2)



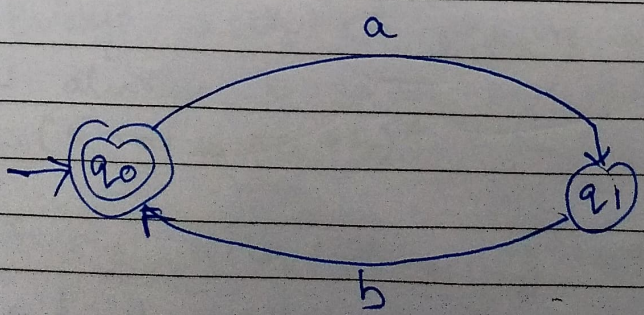
⇒ ab

(3)



⇒ ab^*c

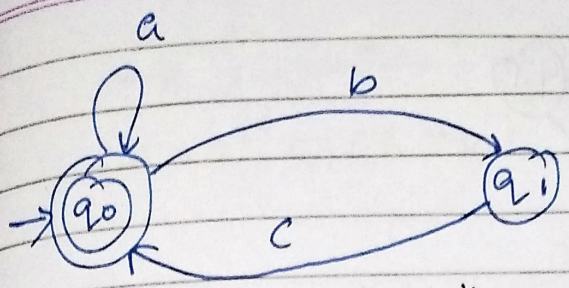
(4)



⇒ $(ab)^*$

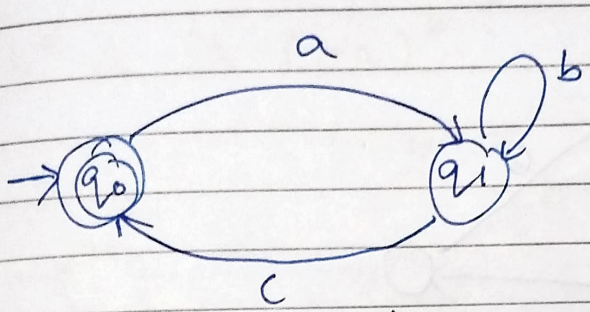
Page No.:
Date:

(5)



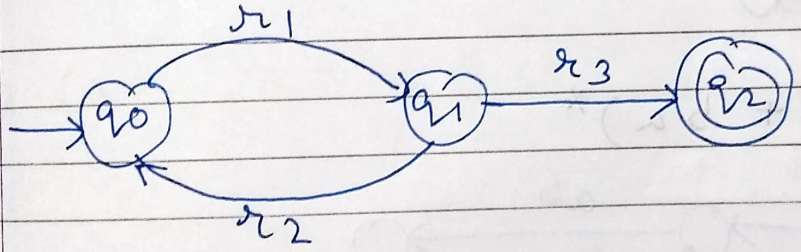
$(a + bc)^*$

(6)



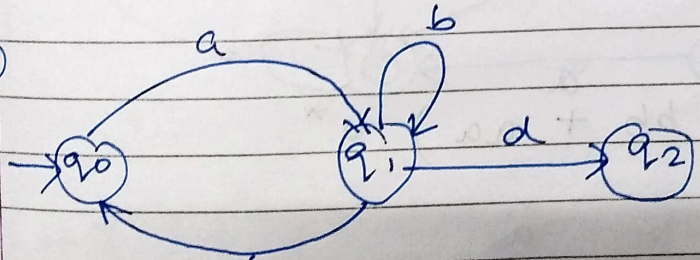
$a\,(b+(a)^*\,c$ OR $(ab^*\,c)^*$

(7)



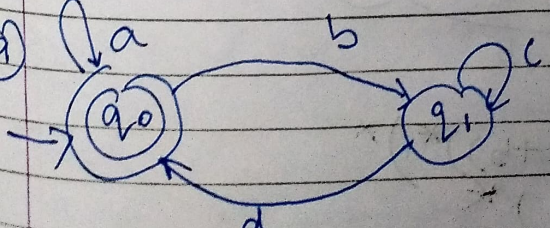
$r_1\,(r_2 \cdot r_1)^*\,r_3$

(8)



$a(b+(a)^*\,d$

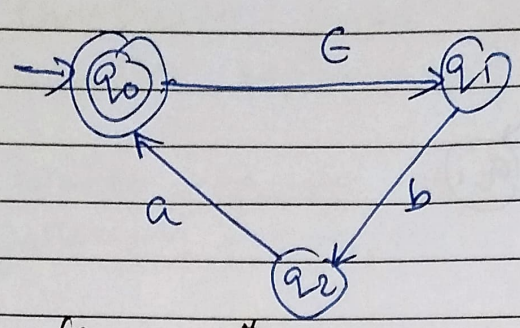
(9)



$(a + b\,c^*\,d)^*$

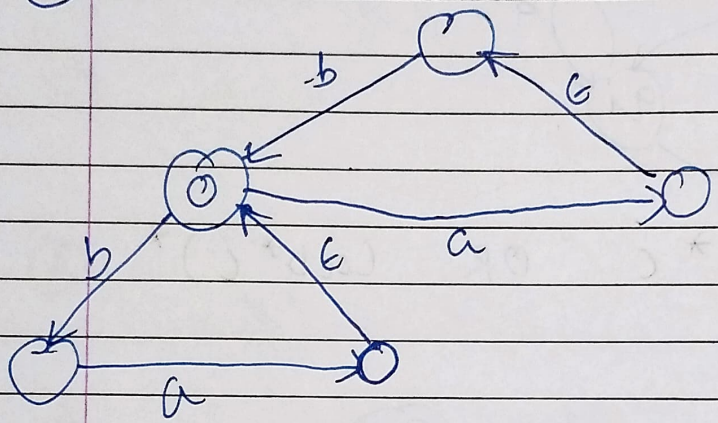
Page No.:
Date:

(10)



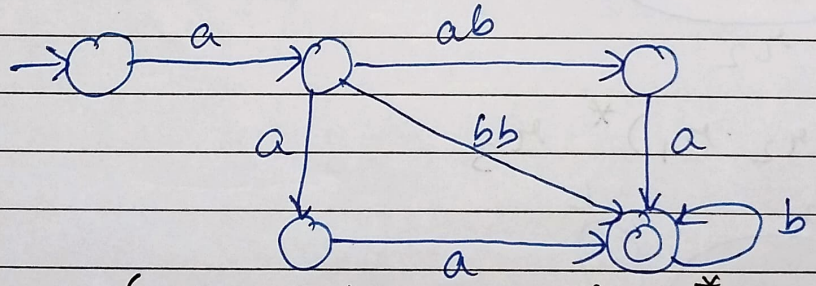
$\Rightarrow (\epsilon\, b\, a)^*$ $\Rightarrow (b\, a)^*$

(11)



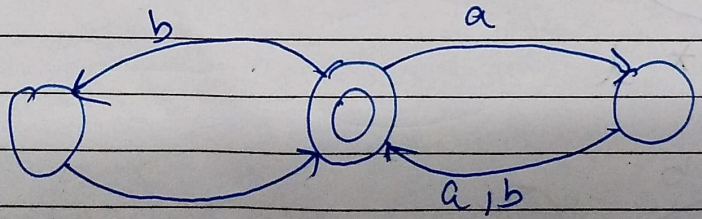
$\Rightarrow (ab + ba)^*$

(12)



$\Rightarrow a\,(aba + bb + aa)\,b^*$

(13)

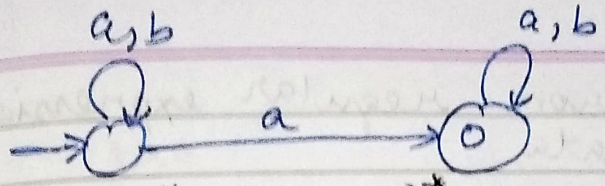


$\Rightarrow (b(a+b)^* + a(a+b)^*)^*$

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

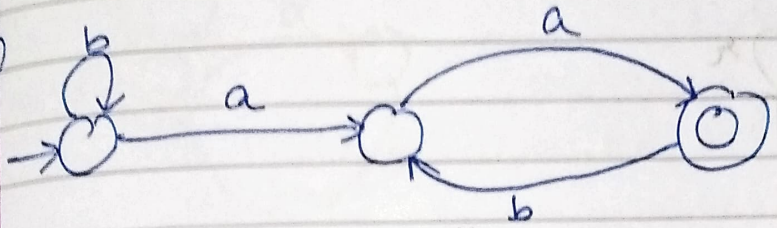
$((a+b)^2)^*$

(12)



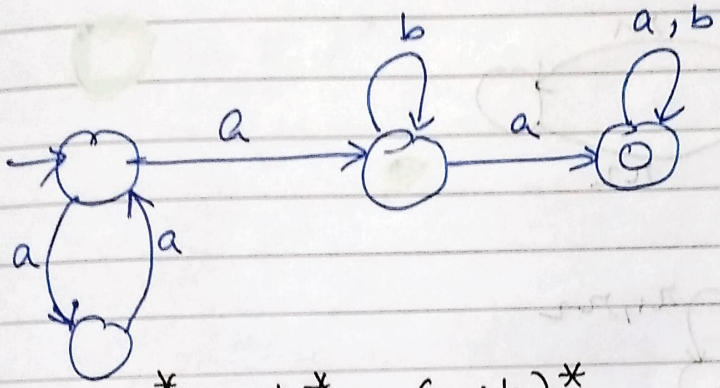
$\Rightarrow (a+b)^* a (a+b)^*$

(13)



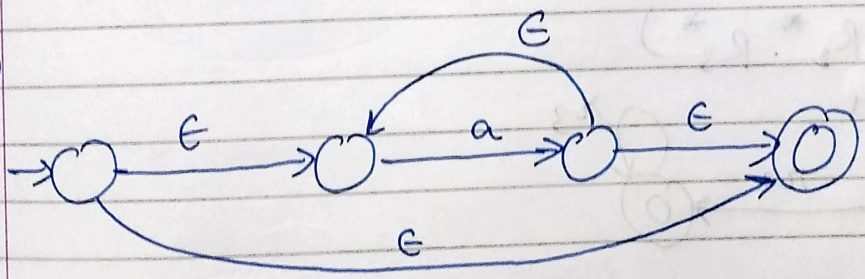
$\Rightarrow b^* a \; a (ba)^*$

(14)



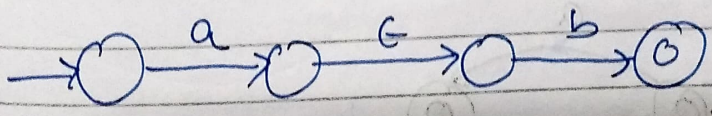
$\Rightarrow (aa)^* a b^* a (a+b)^*$

(15)



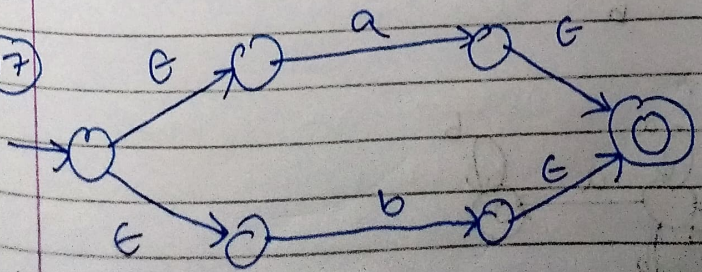
$\Rightarrow a^*$

(16)



$\Rightarrow a b$

(17)

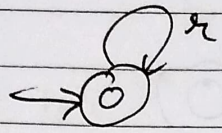


$\Rightarrow a + b$

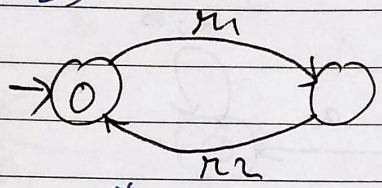
Page No.:
Date:

⇒ Conversion from regular expressions to finite automata

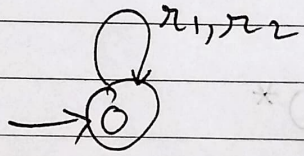
① R^*



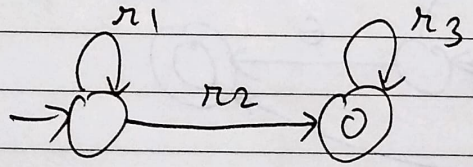
② $(R_1 \cdot R_2)^*$



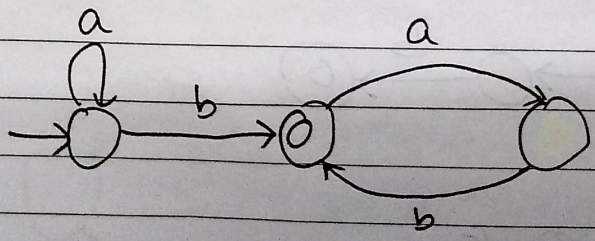
③ $(R_1 + R_2)^*$



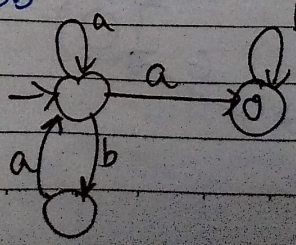
④ $(R_1^* R_2 \cdot R_3^*)$



⑤ $a^* b(ab)^*$

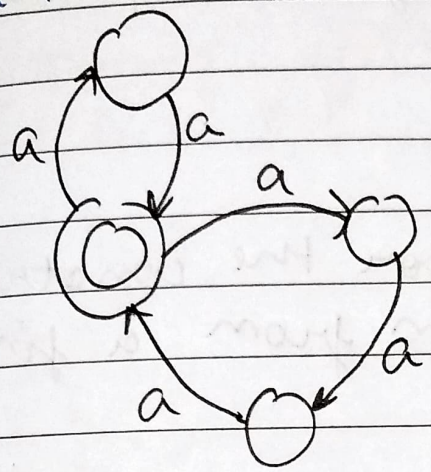


⑥ $(a + ba)^* ab^*$

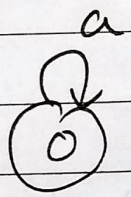


Page No.:
Date:

(7) $(aa + aaa)^*$



(8) $(a + aaaaa)^*$



Page No.:
Date:

Equivalence between regular expression and finite automata

- **Arden's theorem**

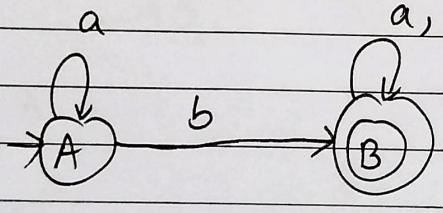
It is the mechanism for the construction of a regular expression from a finite automaton

Q1) Consider a DFA and convert it into regular expression using Arden's theorem

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

A is the initial state and B is the final state

⇒



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

OR

⇒ Steps for Arden's theorem

- For, every individual state of the DFA, write an expression for every incoming and outgoing input alphabet

- Apply Arden's theorem as follows:

(1) If P is free from NULL, then equation $R = Q + RP$ has unique solution, $R = QP^*$

(2) If P contains NULL, then equation $R = Q + RP$ has infinety many solutions

Page No.:
Date:

$A = \epsilon + Aa$

comparing with

$R = Q + RP$

We get

$R = A$

$Q = \epsilon$

$P = a$

$R = QP^*$

$A = \epsilon a^*$

$A = a^*$

$B = Ab + B(a+b)$

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

Comparing with

$R = Q + RP$

$R = B$

$Q = a^*b$

$P = (a+b)$

$R = QP^*$

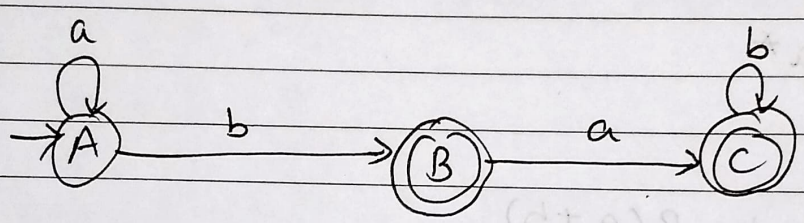
$= a^*b(a+b)^*$

Q2) Consider a DFA and convert it into regular expression using Arden's theorem

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

A is the initial state and B, C is the final state.

=>



$A = \epsilon + Aa$

Comparing with $R = Q + RP$

$Q = \epsilon$
$R = A$
$P = a$

$R = QP^*$
$A = \epsilon a^*$
$A = a^*$

$B = Ab$
$B = a^*b$

$C = Ba + C(a+b)$
$C = a^*ba + C(a+b)$

Comparing with, $R = Q + RP$

$Q = a^*ba$, $R = C$, $P = a+b$

$$R = qp^*$$
$$= a^*ba\,(a+b)^*$$

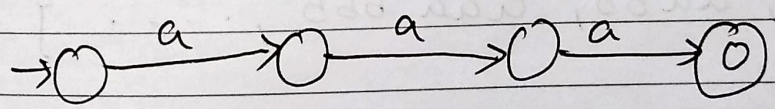
Final = $a^*ba\,(a+b)^* + a^*b$

As F is final state

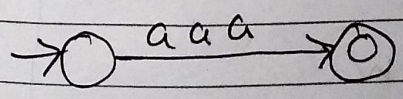
⇒ Transition graph

- Same as ε-nFa can have more than one initial state
- Non-deterministic in nature
- Can have string as a input to transition from one state to another.

Eg: Normal NFA



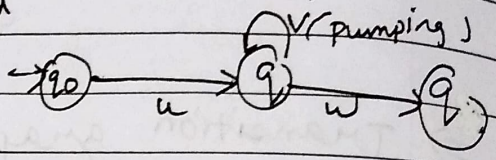
Transition graph



Page No.:
Date:

⇒ Pumping lemma for regular languages

- For any regular language L, there exists an integer n, such that all $z \in L$ with $|z| \geq n$, there exists $u, v, w \in \Sigma^*$, such that $z = uvw$, and
  - $|uv| \leq n$
  - $|v| \geq 1$
  - for all $i \geq 0$: $uv^i w \in L$



- In simple terms, this means that if a string v is 'pumped', i.e., if v is inserted any number of times, the resultant string still remains in L.

Q1) Prove that language $L = \{ a^m b^n \mid m = n \}$ is non-regular?

⇒ $L = \{ ab, aabb, aaabbb, \ldots \}$

Suppose

$z \in L$

$z = a^K b^K$ ←

Dividing z into u, v, w

$\underbrace{a^{K-1}}_{u} \quad \underbrace{a}_{v} \quad \underbrace{b^K}_{w}$

The combination of both gives a^K

for $i = 1 \rightarrow uv^i w \rightarrow a^K b^K$

for $i = 2 \rightarrow uv^i w \rightarrow a^{K+1} b^K$

$i = 2$ does not satisfy the language. Therefore the language is non-regular

Page No.:
Date:

Q2) Proof that language $L = \{ a^m b^n \mid m < n \}$ is non-regular?

$\Rightarrow L = \{ abbb, aabbb, aaabbbb, \ldots \}$

Basically no of b's should be more than a

Suppose

$z \in L$

$z = a^{k-1} b^k$

Dividing z into u, v, w.

$\underbrace{a^{k-2}}_{u} \underbrace{a}_{v} \underbrace{b^k}_{w}$

for $i = 1 \rightarrow a^{k-1} b^k$

for $i = 2 \rightarrow a^k b^k$

$i=2$ does not satisfy the language. Therefore the language is non-regular.