Ex.No.1 DESIGN A DFA AND NFA, RECOGNIZE AN INPUT STRING FROM AUTOMATA

DATE: 19/12/24

Aim:

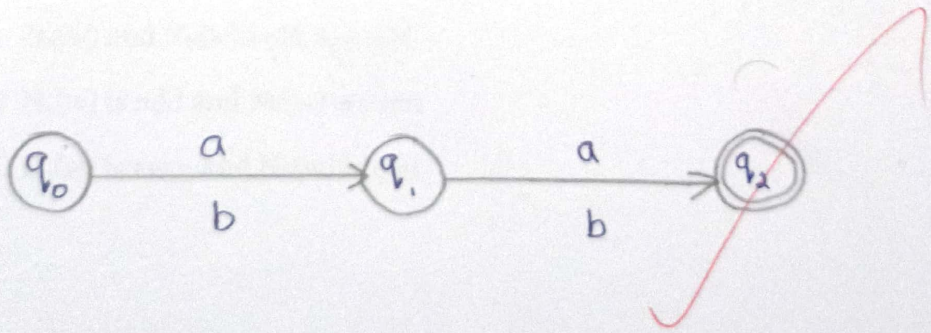
To construct the Deterministic Finite Automata (DFA) and Non-Deterministic Finite Automata (NFA) for the given language and to verify the given string is in the language of automata or not using JFLAP.

Algorithm/ Procedure:

1. **Deterministic Finite Automata:**

* From the current state and for the current input it has only one next state i.e, the next state, is clearly determined.
* Also from every state there exist a transition for all input alphabets.
* It is also represented by using 5 types.

Sample Deterministic Finite automata :



Problems on DFA: (Any Three)

1. **Construct the DFA, that accepts the below languages over $\Sigma=\{a,b\}$ (Draw the Transition diagram using JFLAP and recognize the string in Language)**

(i) $|w| = 2$ (String length exactly equals to 2)

(ii) $|w| <= 2$ (String length with atmost 2)

(iii) $|w| >= 2$ (String length with atleast 2)

(iv) $|w| \bmod 2 = 0$ (String length divisible by 2 or even string length)

(v) $|w| \bmod 2 = 1$ (String length not divisible by 2 or odd string length)

(vi) $N_a(w) = 0 \bmod 2$ (Even numbers of a's or No. of a's divisible by 2)

(vii) $N_a(w) = 1 \bmod 2$ (odd numbers of a's or No. of a's not divisible by 2)

(viii) $N_b(w) = 0 \bmod 2$ (Even numbers of b's or No. of b's divisible by 2)

(ix) $N_b(w) = 1 \bmod 2$ (odd numbers of b's or No. of b's not divisible by 2)

(x) $N_a(w) = 0 \bmod 3$ (No. of a's divisible by 3)

(xi) $N_a(w) = 1 \bmod 3$ (No. of a's divisible by 3 with remainder 1)

(xii) $N_a(w) = 2 \bmod 3$ (No. of a's divisible by 3 with remainder 2)

(xiii) $N_b(w) = 0 \bmod 3$ (No. of b's divisible by 3)

(xiv) $N_b(w) = 1 \bmod 3$ (No. of b's divisible by 3 with remainder 1)

(xv) $N_b(w) = 2 \bmod 3$ (No. of b's divisible by 3 with remainder 2)

(xvi) $N_a(w)$ and $N_b(w)$ both are even

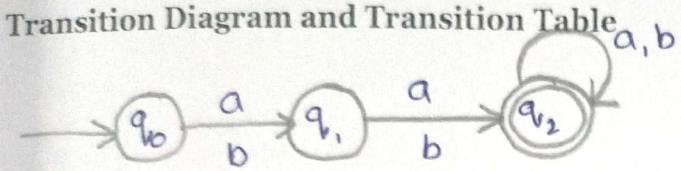
(xvii) $N_a(w)$ and $N_b(w)$ both are odd

(xviii) $N_a(w)$ is odd and $N_b(w)$ is even

(xix) $N_a(w)$ is even and $N_b(w)$ is odd

Solution (Q.No. iii)

Transition Diagram and Transition Table



state \ I/P	a	b
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
$* q_2$	q_2	q_2

Transition Function

ababa

$\delta(q_0, a) = q_1$

$\delta(q_0, ab) = \delta(q_1, b) = q_2$

$\delta(q_0, aba) = \delta(q_2, a) = q_2$

$\delta(q_0, abab) = \delta(q_2, b) = q_2$

$\delta(q_0, ababa) = \delta(q_2, a) = q_2$

$\therefore$ ababa $\in L \rightarrow$ accepted

Transition

$\delta(q_0, a) = q_0$

$\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_2$

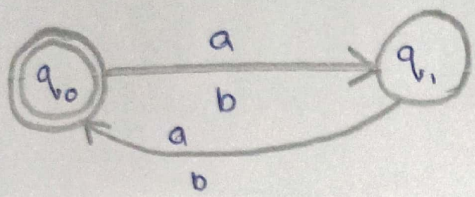
$\delta(q_1, b) - q_2$

$\delta(q_2, a) - q_2$

$\delta(q_2, b) - q_2$

Solution (Q.No. iv)

Transition Diagram and Transition Table



i/p / state	a	b
→ * q_0	q_1	q_1
q_1	q_0	q_0

Transition Function

abaabb

$\delta(q_0, a) - q_1$

$\delta(q_0, ab) - \delta(q_1, b) - q_0$

$\delta(q_0, aba) - \delta(q_0, a) - q_1$

$\delta(q_0, abaa) - \delta(q_1, a) - q_0$

$\delta(q_0, abaab) - \delta(q_0, b) - q_1$

$\delta(q_0, abaabb) - \delta(q_1, b) - q_0$

abaabb $\in L \rightarrow$ accepted

Transition

$\delta(q_0, a) - q_1$

$\delta(q_0, b) - q_1$

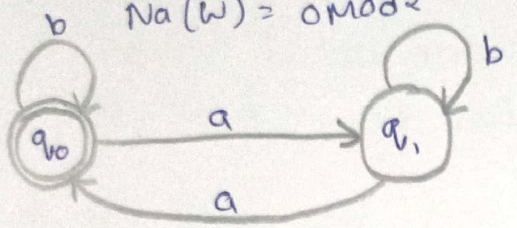
$\delta(q_1, a) - q_0$

$\delta(q_1, b) - q_0$

Solution (Q.No. V;)

Transition Diagram and Transition Table

$N_a(w) = 0 \bmod 2$



i/p / state	a	b
→ *q_0	q_1	q_0
q_1	q_0	q_1

Transition Function

aabaab

$\delta(q_0, a) = q_1$

$\delta(q_0, aa) = \delta(q_1, a) - q_0$

$\delta(q_0, aab) = \delta(q_0, b) - q_0$

$\delta(q_0, aaba) = \delta(q_0, a) - q_1$

$\delta(q_0, aabaa) = \delta(q_1, a) - q_0$

$\delta(q_0, aabaab) = \delta(q_0, b) - q_0$

aabaab $\in L \rightarrow$ accepted

Transitions

$\delta(q_0, a) - q_1$

$\delta(q_0, b) - q_0$

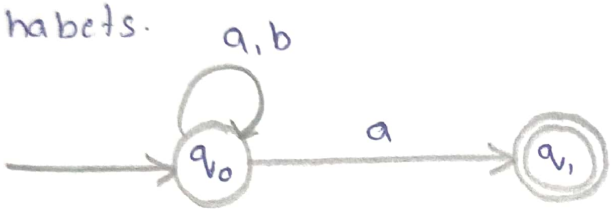
$\delta(q_1, a) - q_0$

$\delta(q_0, b) - q_1$

2. **Non-Deterministic Finite Automata (NFA):**

* From the current state and for the current input, it has more than one next state that is that next state is non deterministic.

* It may not have transition for all the inputs Alphabets.



Problem:(Any Three)

2. **Construct the NFA that accepts the below language over ∑={a,b}**

(i) Strings start with 'a'

(ii) Strings ends with 'ab'

(iii) Strings that have 3 consecutive a's

(iv) String that has 'ab' as sub-string

Solution (Q.No.)

Transition Diagram and Transition Table

i/p / state	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
$*q_2$	ϕ	ϕ

Transition Function

aabab

$\delta(q_0, a) - q_0$

$\delta(q_0, aa) - \delta(q_0, a) - q_0$

$\delta(q_0, aab) - \delta(q_0, b) - q_0$

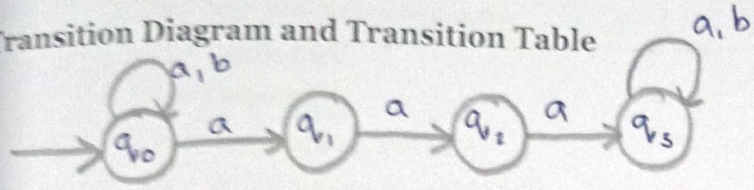
$\delta(q_0, aaba) - \delta(q_0, a) - q_1$

$\delta(q_0, aabab) - \delta(q_1, b) - q_2$

$\therefore$ aabab $\in L \rightarrow$ accepted

Solution (Q.No. iii)

Transition Diagram and Transition Table



i/p / state	a	b
→ q_0	$\{q_0, q_1\}$	q_0
q_1	q_2	ϕ
q_2	q_3	ϕ
* q_3	q_3	q_3

Transition Function

baaab

$\delta(q_0, b) - q_0$

$\delta(q_0, ba) - \delta(q_0, a) - q$

$\delta(q_0, baa) - \delta(q_1, a) - q_2$

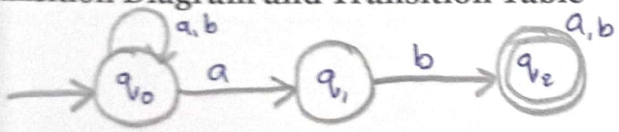
$\delta(q_0, baaa) - \delta(q_2, a) - q_3$

$\delta(q_0, baaab) - \delta(q_3, b) - q_3$

$\therefore$ baaab $\in L \rightarrow$ accepted

Solution (Q.No. iv)

Transition Diagram and Transition Table



i/p / state	a	b
→ q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
* q_2	q_2	q_2

Transition Function

babab

$\delta(q_0, b) = q_0$

$\delta(q_0, ba) = (q_0, a) - q_1$

$\delta(q_0, bab) = (q_1, b) - q_2$

$\delta(q_0, baba) = (q_2, b) - q_2$

$\delta(q_0, babab) = (q_2, b) \rightarrow q_2$

$\therefore$ babab $\in L \rightarrow$ accepted.

Ex.No.2 AUTOMATA TRANSLATION USING JFLAP

(NFA TO DFA)

DATE: 02-01-25

Aim:

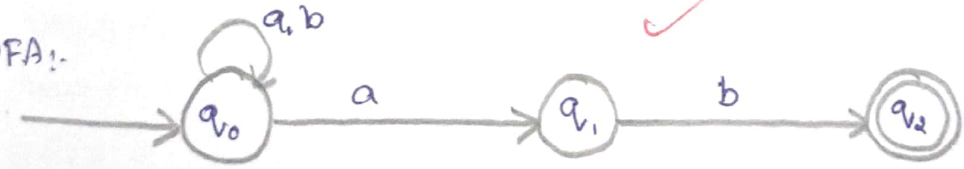
To convert the Non-Deterministic Finite Automata (NFA) to Deterministic Finite Automata (DFA) using JFLAP.

Algorithm/ Procedure:

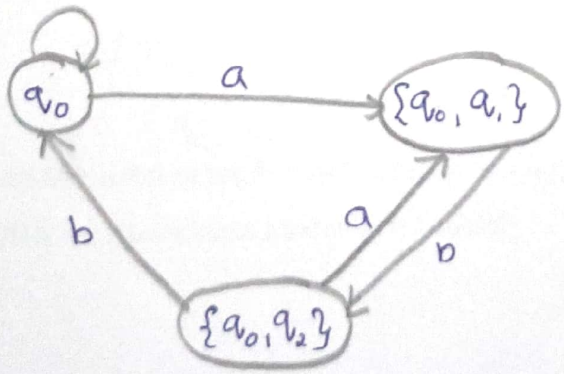
Steps in conversion of NFA to DFA:

1. Construct the transition table for the given NFA (If transition table is given take as it is)
2. Find the new states from the NFA's transition table and find the next states.
3. Repeat step2 until no more new states can be fa
4. Construct the DFA's Transition table and find reachable state.
5. Construct the DFA using reachable states.

NFA:-



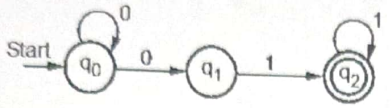
DFA:-



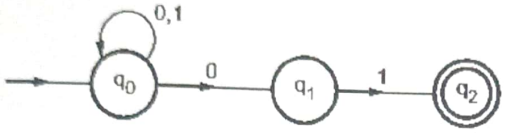
Problems:

1. Convert the following NFA to DFA (Any one)

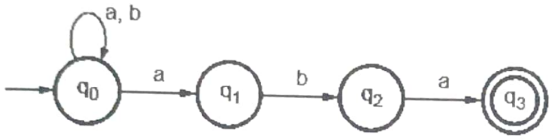
(i)



(ii)



(iii)



Solution (Q.No.)

Step 1: Construct Transition Table:

state \ i/p	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
* q_2	ϕ	ϕ

Step 2: Identify the new states from transition table

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

$\delta(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$

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

$= \{q_0, q_1\}$

$\delta(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$

$= q_0 \cup q_2$

$= \{q_0, q_2\}$

Step 3: Find the next state for new states in step 2 using transition function (Repeat until no more new states are found)

New state - $\{q_0, q_2\}$

$$\delta(\{q_0, q_2\}, 0) = \delta\{q_0, 0\} \cup \delta\{q_2, 0\}$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\}$$

$$\delta(\{q_0, q_2\}, 1) = \delta(\{q_0, 1\} \cup \delta\{q_2, 1\})$$
$$= q_0 \cup \phi$$
$$= q_0$$

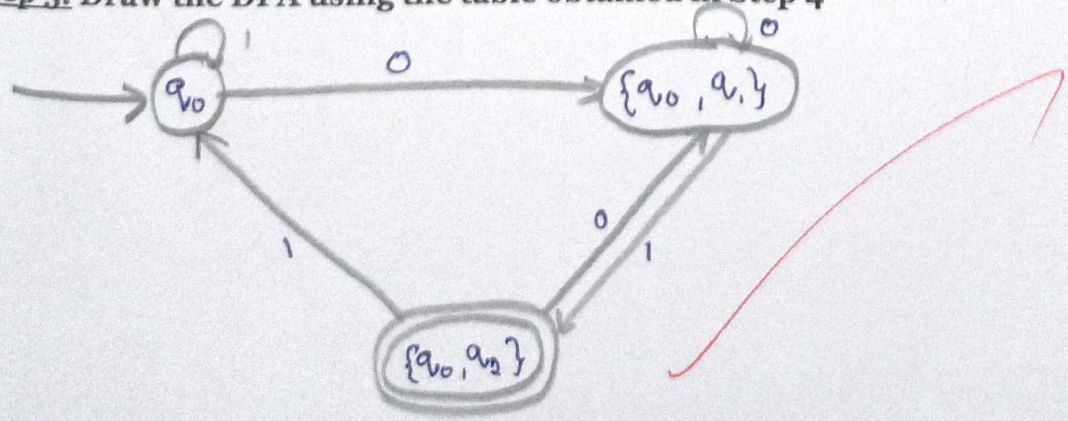
Step 4: **Construct the new Transition Table and find the reachable states**

i/p / state	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
q_2	ϕ	ϕ
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
* $\{q_0, q_2\}$	$\{q_0, q_1\}$	q_0

Reachable states

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

Step 5: **Draw the DFA using the table obtained in Step 4**



Ex.No.3 AUTOMATA AND REGULAR EXPRESSION

TRANSLATION USING JFLAP

DATE: 30-1-25

Aim:

(i) To convert the Regular Expression (RE) to Finite Automata and Vice-Versa using JFLAP.

(ii) To convert the Regular Expression (RE) to Epsilon NFA using JFLAP.

Algorithm/ Procedure:

Regular Expression-Definition:

* Both finite automata and regular expression accepts the regular language
* The representation of regular language using an expression is called regular expression.

ε is a RE where $\varepsilon = \{\ \}$

'a' is a RE where $\varepsilon = \{a\}$

'b' is a RE where $\varepsilon = \{b\}$

'a+b' is a RE where $\varepsilon = \{a, b\}$ → union of two RE

'a.b' is a RE where $\varepsilon = \{a, b\}$ → combination of two RE

a^* is a RE where $\varepsilon = \{a\}$ → kleene closure (0 to n)

a^+ is a RE where $\varepsilon = \{a\}$ → positive closure (1 to n)

Closure:

Kleene Closure:

kleene closure (*) - (0 to n) $\Sigma = \{a, b\}$

$R_1 = a$ $\boxed{R_1^* = a^*}$

$= \{\epsilon, a, aa, aaa, \ldots\}$

Positive Closure:

Positive closure (+) - (1 to n)

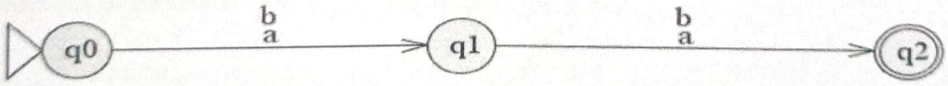
$R_1 = a$ $R_1^+ = a^+$

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

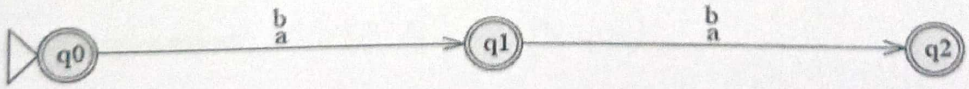
(i) **Finite Automata to Regular Expression:**

Problems:

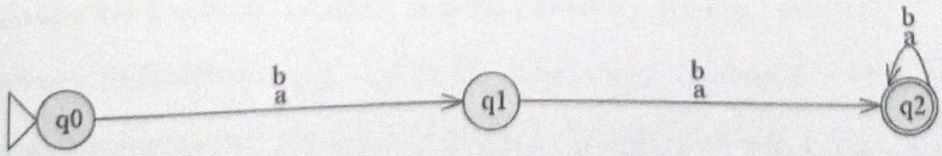
1. **Convert the following Finite Automata to Regular Expression**



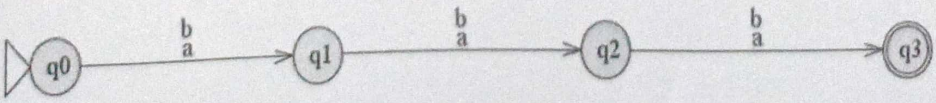
(i)



(ii)



(iii)



(iv)

(v)

Solutions:

(i) **Language L=** { aa, ab, ba, bb }

Language Definition: set of all string whose length is exactly 2

Regular Expression: $(a+b)(a+b)$

(ii) **Language L=** { ε, a, b, aa, ab, ba, bb }

Language Definition: set of all strings whose length is almost 2

Regular Expression: $\lambda + a + b + (a+b)(a+b)$

(iii) **Language L=** {aa, ab, bb, ba, aab, aaa, baa, bab }

Language Definition: set of all strings whose length is atleast 2

Regular Expression: $(a+b)(a+b)(a+b)^*$

(iv) **Language L=** {aaa, bbb, aab, baa, aba, bba, bab, abb }

Language Definition: set of all strings whose length is exactly 3

Regular Expression: $(a+b)(a+b)(a+b)$

(v) **Language L=** { aaaa, bbbb, abbb, baaa, bbba, aaab ..... }

Language Definition: set of all strings whose length is exactly 4

Regular Expression: $(a+b)(a+b)(a+b)(a+b)(a+b)^+$

(ii) **Regular Expression to Epsilon NFA:**

Procedure:

Thomson Construction Method

(**Precedence order:** Closure, Union, Concatenation)

(i) **Union Case:** $(+) \rightarrow OR$

* Let R_1 and R_2 be the regular expression.

$R_1 = a$, $R_2 = b$ to find $R_3 = R_1 + R_2$.

i) construct the Automata for RE R_1 and R_2

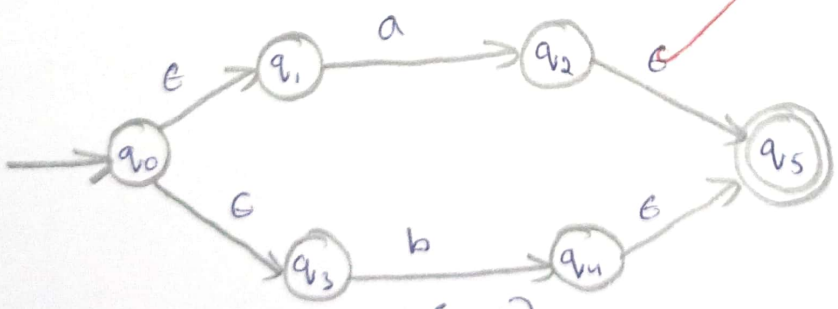
$R_1 = a$ $\qquad$ $R_2 = b$

ii) Draw the two Automatas parallely. Then create a new start state and new final state.

iii) Create the Null transition as follows.

iv) Create ϵ transition from new start state to existing Automata start state.

v) Create ϵ transition from existing Automatas to new final state.



$(a+b)$

(ii) **Concatenation Case:** $(\cdot) \rightarrow$ Followed by AND let R_1 and R_2 be two regular expression

$R_1 = a$, $R_2 = b$ to find $R_3 = R_1 \cdot R_2$

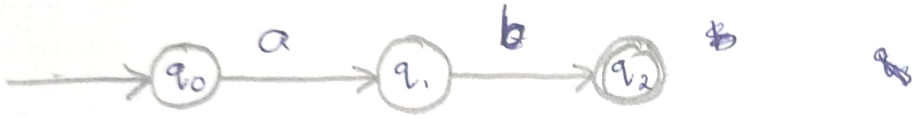
Construct the Automata for R.E R_1 and R_2

$R_1 = a$ $\qquad$ $R_2 = b$

ii) Draw the two Automata serially such that the first Automata start state will be start state of resulting Automata.

iii) Second Automata's final state will be the final state of resulting Automata.

iv) Provide transition from first Automata final state to second Automata start state.



(iii) Closure Case:

Kleene Closure:

Kleene closure (*) - (0 to n)

Let R_1 be the RE $R_1 = a$ then $R_1^* = a^*$

i) Construct the Automata for the RE R_1 $R_1 = a$

ii) Create a new start state and a new final state

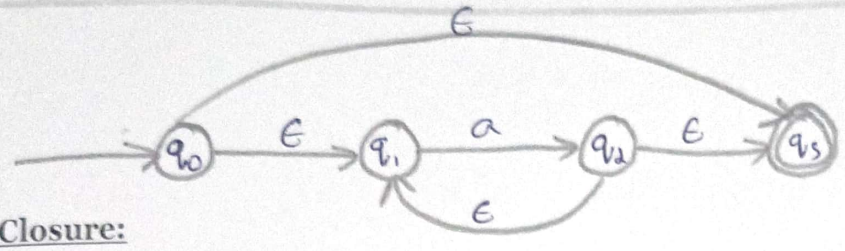
iii) Provide the ϵ transition as follows.

b) Create a ϵ transition from existing Automates final state

a) Create a ϵ transition from new state to existing automata start state.

b) Create a ϵ transition from new start state to new final state

a) Create a ϵ transition from old final state to old start state.

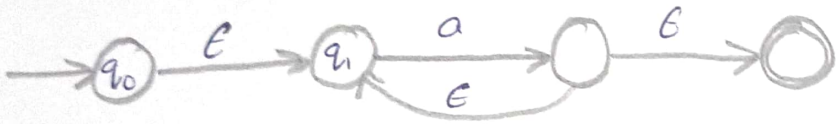


Positive Closure:

Positive closure (+) → (1 to n)

a^+

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



Problem:

Convert the Regular Expression (a+b)*abb to epsilon NFA.

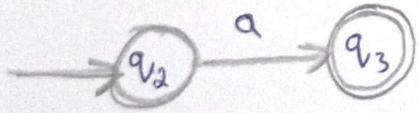
Solution:

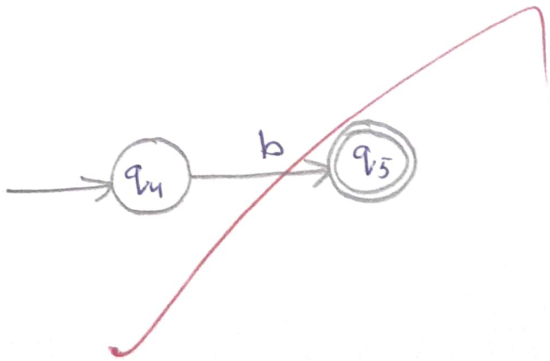
Union Case: (a+b)

R1= a

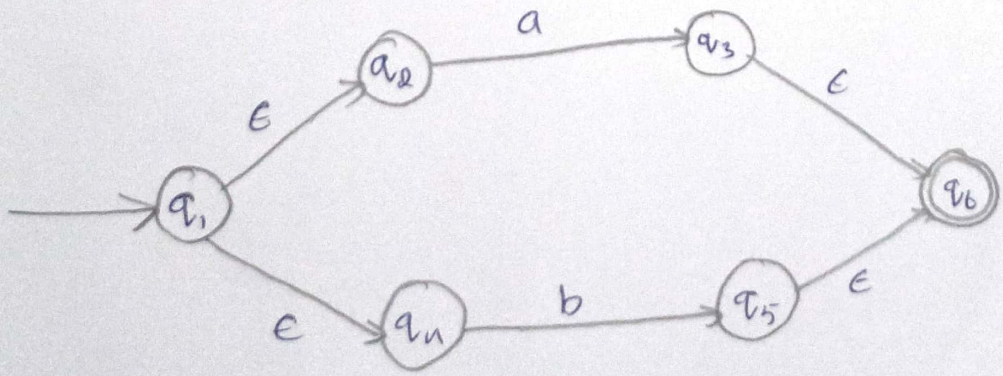
R2=b

$R_3 = R_1 + R_2$



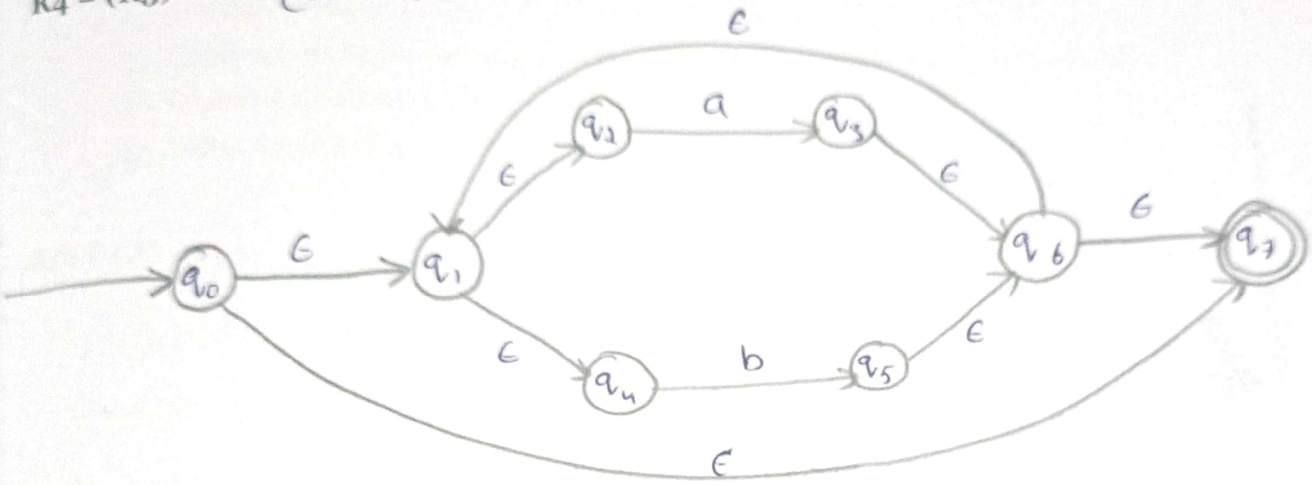


R3= (R1 + R2) => (a+b)



Closure Case: (a+b) * [Kleene Closure]

R4 = (R3) * = $(a+b)^*$



Concatenation Case: (abb)

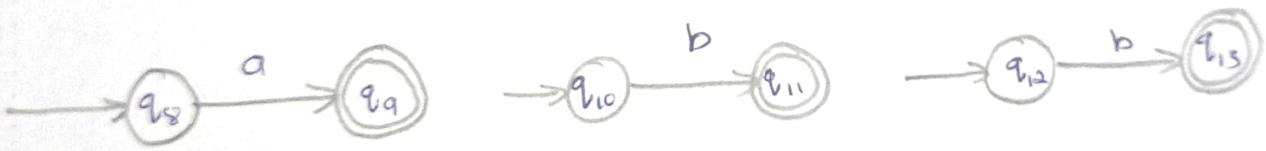
R5=abb

$R_6 = a$

$R_5 = R_6 . R_7 . R_8$

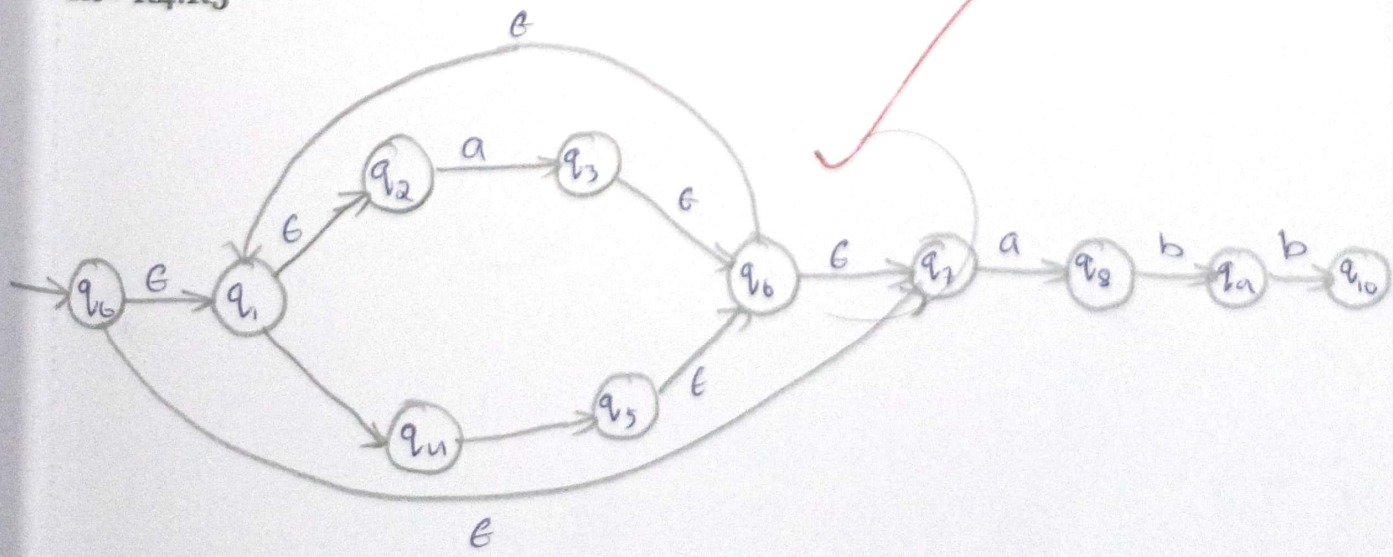
$R_7 = b$

$R_8 = b$



Final Epsilon NFA:

R6= R4.R5



Ex.No.4 **PROVING LANGUAGE NOT TO BE REGULAR**

DATE: 6/1/25

Aim:

To prove the given language is regular or not with Pumping Lemma using JFLAP.

Algorithm/ Procedure:

Pumping Lemma -Definition:

* Pumping Lema for Regular language is used to prove the given language is regular or not.
* It states that, it takes the string 'w' from the Language 'L' and divides it into 3 parts $\boxed{w = xyz}$

Let n be the no. of states in the Finite Automata then it satisfies the below

i) $|xy| \leq n$

ii) $|y| > 0$

iii) for all $k > 0$, $xy^k z$ is regular

If the above three coordinates satisfied then $z \in L$ and the language 'L' is said to be regular.

Problems:

1. Prove that the given Language is not Regular using Pumping Lemma

$L = \{a^n b^n \mid n > 0\}$

Solution:

Language L ={ ab , aabb , aaabbb , aaaabbbb, ------- }

Case 1: (V contains Only 'a')

$z = aa|a|bbb$
$\quad\;\; u \;\; v$

i) $|uv| \le n$

$|aaa| \le n$

$3 \le 3$

ii) $|v| > 0$

$|a| > 0$

iii) $z = uv^iw \qquad \{i = 2\}$

$= aa(a)^2 bb$

$= aaaabbb \notin L$

Case 2: (V contains Only 'b')

$z = aaa|b|bb$
$\quad\;\; u \;\; v \;\; z$

i) $|uv| \le n$

$|aaab| \le n$

$4 \le 3$ ✗

ii) $|v| > 0$

$|b| > 0$

$z = uv^iw$ $\{i=2\}$

$= aaa(b)^2bb$

$= a^3b^4 \notin L$

Case 3: (V contains both 'a' and 'b')

$z = \underset{u}{aa}\underset{v}{ab}\underset{w}{bb}$

i) $|uv| \leq n$

$|aaab| \leq n$ ✗

ii) $|v| \geq 1$

$|ab| > 0$

iii) $z = uv^iw$ $\{i=2\}$

$= aa(ab)^2bb$

$= aaababbb \notin L$

∴ It is not a Regular language

2. **Prove that the given Language is not Regular using Pumping Lemma**

$L = \{a^n b^k c^{n+k} \mid n,k > 0\}$

Solution:

Language L ={abcc , aabcc , abbccc, aabbcccc..... }

Case 1: (V contains Only 'a')

$z = |aabbcccc| \Rightarrow \underset{u}{a}|\underset{v}{a}|\underset{w}{bbcccc}$

i) $|uv| \le n$

$aa \le n$

$2 \le 2 \checkmark$

ii) $|v| > 0$

$|a| > 0$

iii) $z = uv^iw$ $\{i = 2\}$

$z = a(a)^2 bbcccc$

$z = aaabbcccc \notin L$

$\therefore z = a^3 b^2 c^4 \notin L$

Case 2: (V contains Only 'b')

$z = \underset{u}{aa}b|\underset{v}{b}|\underset{w}{ccc}c$

i) $|uv| \le n$

$|aabb| \le n$

$4 \le 2 \times$

ii) $|v| > 0$

$|b| > 0$

iii) $z = uv^iw$ $\{i = 2\}$

$= aab(b)^2 cccc$

$= aa\ bbb\ cccc$

$z = a^2 b^3 c^4 \notin L$

Case 3: (V contains both 'a' and 'b')

$z = a|ab|b\,cccc$

$\quad\;\; u \quad v \quad w$

i) $|uv| \le n$

$|aab| \le 2$

$3 \le 2$ ✗

ii) $|v| > 0$

$|ab| > 0$

iii) $z = uv^iw$

$= a(ab)^2 bcccc$

$= aababbcccc$

$z = a^3b^3c^4 \notin L$

Case 3: (V contains only 'c')

$z = aa\,bb\,|\,c\,|\,ccc$

$\quad\;\; u \quad v \quad w$

i) $|uv| \le n$

$|aabbc| \le 2$ ✗

ii) $|v| > 0$

$|c| > 0$

iii) $z = uv^iw$

$= aa\,bb\,(c)^2\,ccc$

$= aabbccccc$

$z = a^2b^2 \cdot c^5 \notin L$

$\therefore$ It is not a Regular language