ending with the string abb. So,
L = {abb, aabb, babb, aaabb, ababb, ...}

$(11)^*$ Set of consisting of even number of 1's including empty string, so
L = {ε, 11, 1111, 111111, ...}

$(aa)^*(bb)^*b$ Set of strings consisting of even no. of a's followed by odd number of b's, so, L = {b, aab, aabbb, aabbbbb, aaaab, aaaabbb, ...}

$(aa+ab+ba+bb)^*$ String of a's and b's of even length can be obtained by concatenating any combination of the strings aa, ab, ba and bb including null, so
L = {aa, ab, ba, bb, aaab, aaba, ...}

a+b	{a, b}
a.b	{ab}
a^*+ba	{ε, a, aa, aaa..., ba}

Operations performed on Regular Expressions

1. Union
   L1 = {00, 10, 11, 01} and L2 = {ε, 100}
   Then, L1 ∪ L2 = {ε, 00, 10, 11, 01, 100}
2. Concatenation
   L1 = {0, 1} and L2 = {00, 11} then L1.L2 = {000, 011, 100, 111}
3. Kleene Closure
   $L1^* = \{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 1, ...\}$

✡ Algebraic Properties of Regular Expressions

1. Closure

If r_1 and r_2 are regular expression, then

- r_1^* is a RE
- $r_1 + r_2$ is a RE
- $r_1 \cdot r_2$ is a RE (eg. $(1+0).(1+0) = \{11, 10, 01, 00\}$

2. Closure Laws

- $(r^*)^* = r^*$
- $\phi^* = \varepsilon$
- $r^+ = r \cdot r^* = r^* r$ as $r^* = \varepsilon + r + rr + rrr$ and $r \cdot r^* = r + rr + rrr \ldots$
- $r^* = r^* + \varepsilon$

3. Associativity

If r_1, r_2, r_3 are RE, then

(i) $r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$

(ii) $r_1 \cdot (r_2 \cdot r_3) = (r_1 \cdot r_2) \cdot r_3$

4. Identity

In the case of union operators,

$r + \phi = \phi + r = r$

In the case of concatenation operator,

$r \cdot x = r$, for $x = \varepsilon$, $r \cdot \varepsilon = r$

5. Annihilator

- If $r + \phi = r_0 \Rightarrow r \cup \phi = \phi$, there is no annihilator for $+$.
- In the case of a concatenation operator $r \cdot \phi = \phi$, therefore ϕ is the annihilator for the $(\cdot)$ operator for eg $\{a, aa, ab\} \cdot \{\} = \{\}$

4K-36

6. Distributed property

If r_1, r_2, r_3 are regular expressions, then

- $(r_1+r_2)\cdot r_3 = r_1\cdot r_3 + r_2\cdot r_3$ i.e. Right distribution
- $r_1\cdot(r_2+r_3) = r_1\cdot r_2 + r_1\cdot r_3$ i.e. Left Distribution
- $(r_1\cdot r_2)+r_3 \neq (r_1+r_3)\cdot(r_2+r_3)$

7. Commutative property

If r_1, r_2 are RE, then

- $r_1+r_2 = r_2+r_1$
- $r_1\cdot r_2 \neq r_2\cdot r_1$

8. Idempotent law

- $r_1+r_1 = r_1 \Rightarrow r_1 \cup r_1 = r_1$
- $r\cdot r \neq r$

9. Identities for Regular Expression

- $\phi + r = r$
- $\phi\cdot r = r\cdot\phi = \phi$
- $\varepsilon\cdot r = r\cdot\varepsilon = r$
- $\varepsilon^* = \varepsilon$ and $\phi^* = \varepsilon$
- $r+r = r$
- $r^*\cdot r^* = r$
- $r\cdot r^* = r^*\cdot r = r^+$
- $(r^*)^* = r^*$
- $\varepsilon + r\cdot r^* = r^* = \varepsilon + r\cdot r^*$
- $(p\cdot q)^*\cdot p = p\cdot(q\cdot p)^*$
- $(p+q)^* = (p^*\cdot q^*)^* = (p^*+q^*)^*$
- $(p+q)\cdot r = p\cdot r + q\cdot r$ and $r\cdot(p+q) = r\cdot p + r\cdot q$

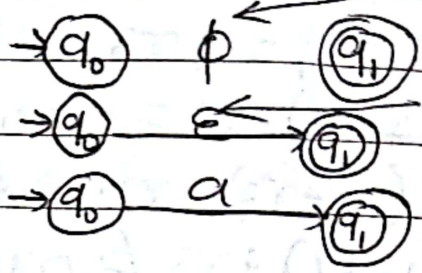
HK-37

✡ Automata for Regular Expression

(i) NFA accept ϕ

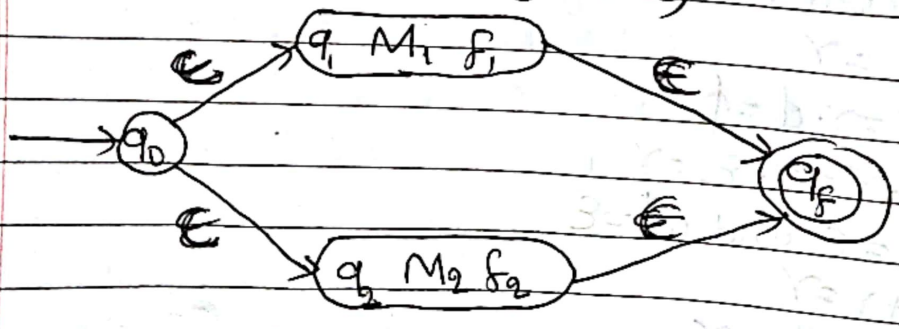
(ii) NFA accept ϵ

(iii) " " a

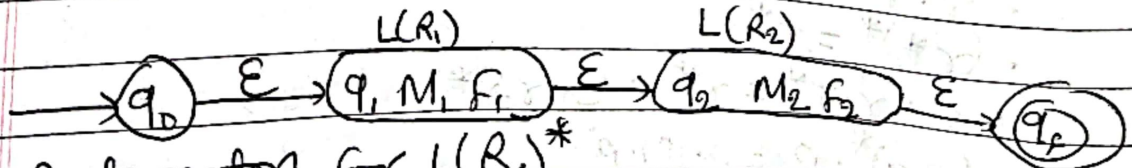


(iv) Schematic representation of regular expression 'R' to accept the language L(R) is show in figure where q is the start state and f is the final state of the machine M.

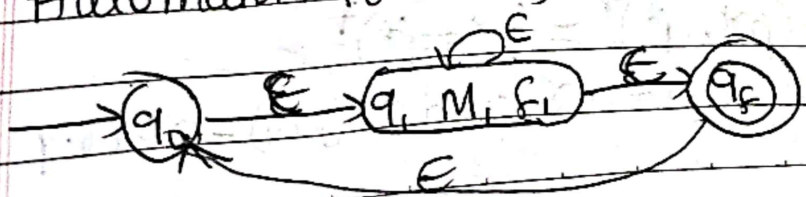
(v) Automaton for $L(R_1+R_2)$



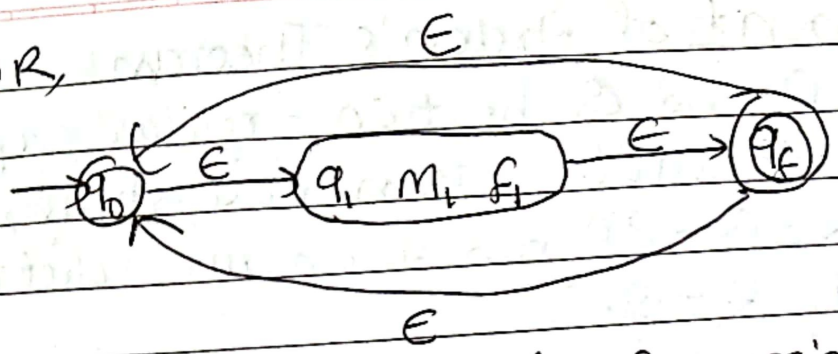
(vi) Automaton for $L(R_1 \cdot R_2)$



(vii) Automaton for $L(R_1)^*$



OR,



✡ Application of Regular Expression

→ Data Validation
→ Data Scraping
→ Data wrangling
→ Simple Parsing
→ The creation of syntax highlighting system and a variety of other tasks are typical applications.

1.7. Arden's Theorem

→ Arden's Theorem is a fundamental result in the TOC used to solve regular expressions from a given linear equation.
→ It is particularly useful when converting finite automata into regular expressions and vice versa.
→ This theorem allows us to automate the construction of regular expressions from finite automata or simplify complex regular expressions.

Statement of Arden's Theorem

Let P and Q be two regular expressions. If P does not contain null string, then $R = Q + RP$ has a unique solution that is $R = QP^*$

Proof

$$R = Q + RP \quad \text{...(i)}$$

$$R = Q + (Q + RP)P \quad [\text{After putting the value } R = Q + RP]$$

$$= Q + QP + RPP$$

When we put the value of R recursively again and again, we get the following equation

$$R = Q + QP + QP^2 + QP^3 + \cdots$$

$$R = Q(\varepsilon + P + P^2 + P^3 + \cdots)$$

$$R = QP^* \quad [\text{As } P^* \text{ represents } (\varepsilon + P + P^2 + P^3 + \cdots)]$$

Hence, proved.

Reverse Proof

$R = QP^*$ is the solution of $R = Q + RP$

$$R = Q + RP \quad \text{...(i)}$$

Now, replacing R by $R = QP^*$, we get

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

Taking Q as common

$$R = Q(\varepsilon + P^*P)$$

$$= QP^* \quad [\varepsilon + R^*R = R^*]$$

Hence proved.

HK-40

Applying Arden's Theorem in TOC

- There must be no null transitions in the transition diagram.
- It must only have one initial state

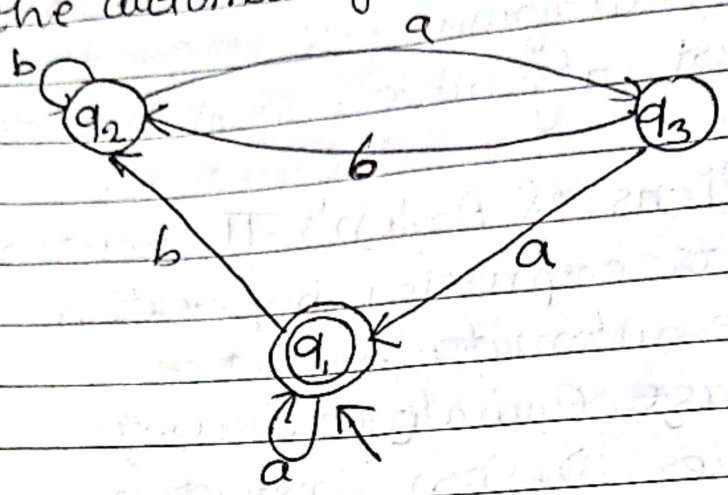
✡ Applications of Arden's Theorem in TOC

1. Regular expression simplification
2. Finite automata construction
3. language Equivalence checking
4. Compiler Design Construction
5. Text Processing and Pattern Matching
6. Automata Conversion

✡ Features of Arden's theorem in the Context of TOC

(i) Solving systems of equations
(ii) Unique Solution
(iii) Efficiency and systematic procedure
(iv) Applicability to regular languages
(v)

Example: Construct a regular expression corresponding to the automata given below:



Soluⁿ⇒ In this case, the initial and final states are q_1. The following are the equations for the three states q_1, q_2 and q_3.

$q_1 = q_1a + q_3a + \varepsilon$ (move is because q_1 is the starting point, Soluⁿ⇒

$q_2 = q_1b + q_2b + q_3b$

$q_3 = q_2a$

Now, we will solve three equations :-

$q_2 = q_1b + q_2b + q_3b$

$= q_1b + q_2b + (q_2a)b$ (substituting value of q_3)

$= q_1b + q_2(b + ab)$

$= q_1b(b + ab)^*$ (Applying Arden's Theorem

$q_1 = q_1a + q_3a + \varepsilon$

$= q_1a + q_2aa + \varepsilon$ (substituting value of q_3)

$= q_1a + q_1b(b + ab)^*aa + \varepsilon$ (Substituting value of q_2)

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

HK-42

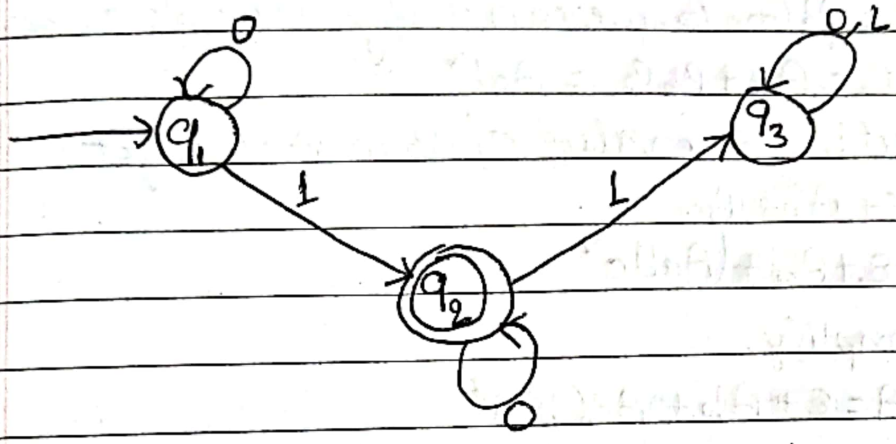
$= \varepsilon(a+b(b+ab)aa)^*$

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

Hence, the regular expression is $(a+b(b+ab)^*aa)^*$

Example

Construct a regular expression corresponding to the automata given below



Soln ⇒ Here, the initial state is q_1 and the final state is q_2.

Now, we write down the equations:

$q_1 = q_1 0 + \varepsilon$

$q_2 = q_1 1 + q_2 0$

$q_3 = q_2 1 + q_3 0 + q_3 1$

Now, we will solve these three equations

Now, we will solve these three equations:-

$q_1 = \varepsilon 0^*$ [As, $\varepsilon R = R$]

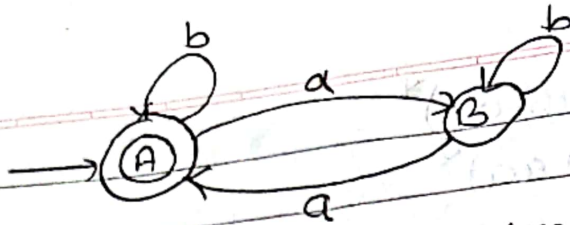
So, $q_1 = 0^*$

$q_2 = 0^*1 + q_2 0$

So, $q_2 = 0^*1(0)^*$ [By Arden's Theorem]

Hence, the regular expression is $0^*1(0)^*$

Example3



Construct regular expression for above DFA

Soln→ We see that A is final state.

$A = \varepsilon + Ab + Ba$

$B = Aa + Bb$

Taking eqn for B, we can apply Arden's theorem

$B = Aa + BbB = Aab^*$

Substituting the value of B in A we get

$A = \varepsilon + Ab + Ba$

$A = \varepsilon + Ab + (Aab)a^*$

Simplify,

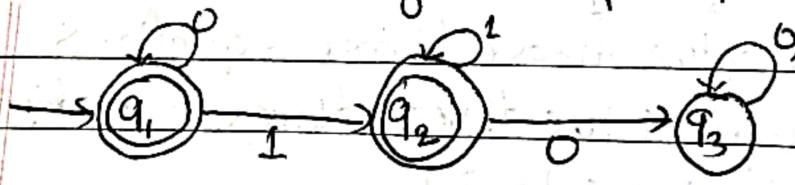
$A = \varepsilon + Ab + Aa(b)a^*$

Combined and simplify further:

$A = \varepsilon(b + aba^*)$

$A = (b + aba^*)^*$

Example4 Construct Regular Expression for



Soln→ $q_1 = \varepsilon + q_1 0$ ---(i)

$q_2 = q_1 1 + q_2 1$ ----(ii)

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

Now, we will solve the three equations

$q_1 = \varepsilon + q_1 0$

$q_1 = \varepsilon 0^*$

$= 0^*$

(Arden's Theorem
$R = Q + RP$
$R = Qp^*$)

$q_2 = q_1 1 + q_2 1$

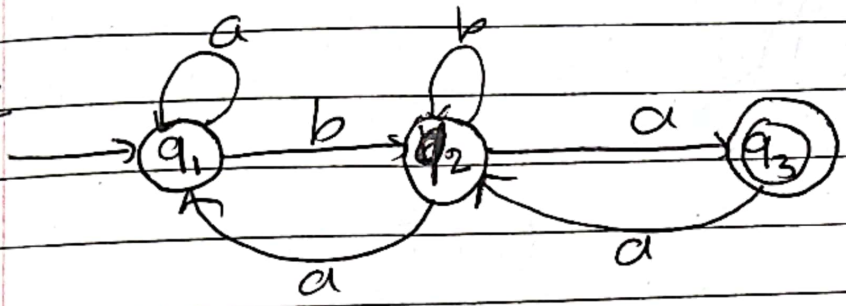
$q_2 = 0^* 1 + q_2 1$

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

[Arden's theorem
$R = Q + RP$
$R = QP^*$]

$q_2 = 0^* 1 1^*$

Samples



Ans

$q_3 = [a + b(b + aa)^* a]^* (b + aa)^* a$

Construct Regular expression for above DFA

Sol^n →