UNIT-2

REGULAR LANGUAGES AND REGULAR EXPRESSIONS

UNIT - II (FLAT)

Finite Automata and Regular Expressions :

Regular expression :

→ The language accepted by FA can be easily described by simple expression called Regular Expressions.

→ It is most effective way to represent any language.

→ Regular expressions are used to match character combination in strings.

→ string searching algorithm used this pattern to find the operations on a string.

→ contains three operators

1. $\emptyset$ is a regular expression which denotes the empty set.
2. $\{\epsilon\}$ is a regular expression denotes null string.
3. For each a in Σ 'a' is a regular expression and denotes the set $\{a\}$.
4. If 'r' and 's' are regular expressions denoting L1 & L2 then $r+s$

Applications of Regular Expressions:

- operating systems (unix)
- compilation (lexical Analysis)
- programming languages (Java)
- finding patterns in text.

Regular expressions in unix:

→ Basically Regular expressions are used in search tools.
→ Text file search in unix (tool : egrep).
→ The egrep command searches for a text pattern in the file and list the words containing the pattern.

Example: File : xyz

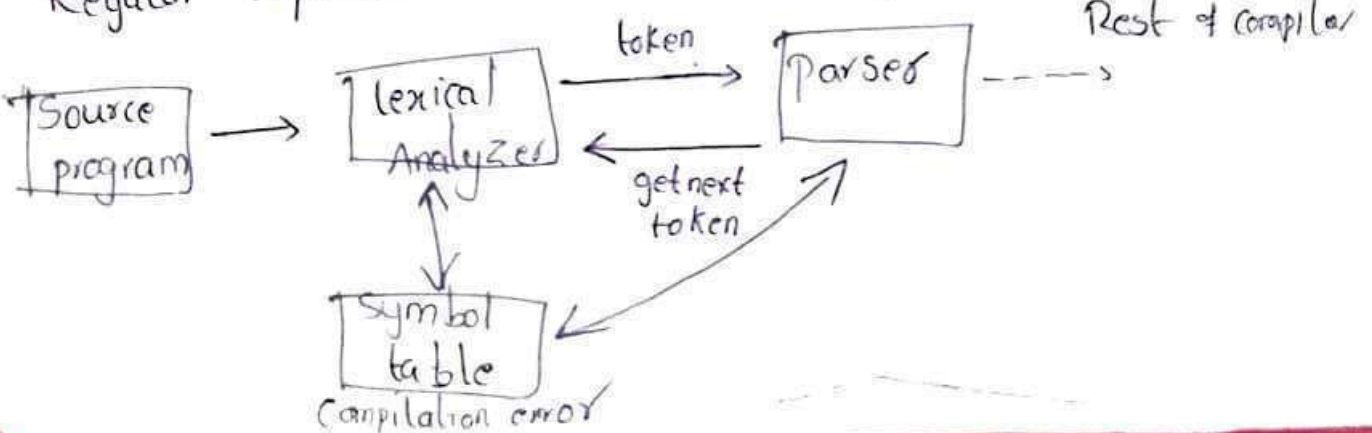
sam
Dexter
John
Raman

command: */< egrep 'n' xyz → searching for 'n' in the contents of the file.

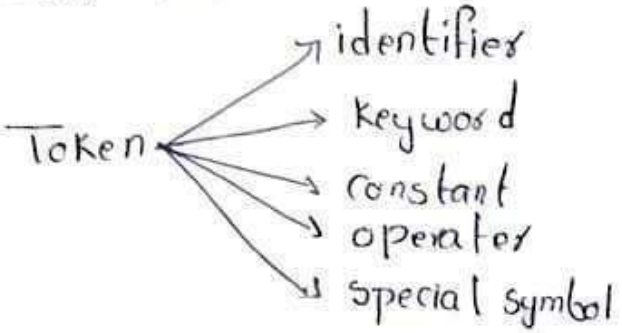
o/p: John
Raman.

Compilation:

Regular expression in lexical Analysis:



→ Lexical Analyzer takes the source program contents and converts them into tokens. Tokens are the individual units.



→ The role of the regular expressions in the lexical Analyzer is it specifies the patterns of the tokens.

Example: identifier

$$([A-Z]|[a-z])([A-Z]|[a-z]|[0-9])^*$$

In most of the programming languages the identifier should start with alphabet and then it can contain the combination of alphabet & digit or ϵ.

Programming languages (Java):

Regular Expressions are a language of string patterns built in to most moder programming languages including Java 1.4 on wards used for searching, extracting and modifying text.

Java.util.regex contains classes for working with regular expressions in Java.

Finding Patterns in Text:

By using regular expressions notation, it becomes easy to describe the patterns at high level.

→ A "compiler" for R.E is useful to turn the regular expressions in to executable code.

Example: "Find me a restaurant within 10 minutes drive of where I am now"

ex: focus on recognizing street addresses in particular. Some people live in "Avenues" or "Roads" or "streets"

Thus we might use R.E like

Street | st\. | Avenue | Ave\. | Road | Rd\.

Above expression is unix style notation with vertical bar rather than +, as union operator.

[0-9]+ [A-Z]? [A-Z][a-z]* ([A-Z][a-z]*)* (Street|st\.|Avenue|Ave\.|Road|Rd\.)

. → Any characters ex: "123A Main st"

? → zero or one characters

+ → one or more digits

Algebraic laws for Regular Expressions:

Like arithmetic expressions, the R.E have a number of laws that work for them. Union as addition & concatenation as multiplication.

Associativity → switch the order of operands & Commutativity → regroup the operands when operator is applied twice.

→ L+M = M+L (commutative law for union)

→ (L+M)+N = L+(M+N) (Associative law for union)

→ (LM)N = L(MN) (Associative law for concatenation)

Identities & Annihilators:

→ An identity for an operator is a value such that when the operator is applied to identity and some other value, the result is other value.

→ An annihilator for an operator is a value such that when the operator is applied to the annihilator & some other value, the result is annihilator.

⇒ $\emptyset + L = L + \emptyset = L$ ($\emptyset$ is identity for union)

⇒ $\epsilon L = L\epsilon = L$ (ϵ is identity for concatenation)

⇒ $\emptyset L = L\emptyset = \emptyset$ ($\emptyset$ is annihilator for concatenation)

Distributive Laws:

⇒ $L(M+N) = LM + LN$ (left distributive law of concatenation)

⇒ $(M+N)L = ML + NL$ (right distributive law of concatenation)

Idempotent law:

→ The result of applying it to two of the same values as arguments is that value is called as idempotent law.

* $L + L = L$ (idempotent law for union)

Laws involving closure:

① $(L^*)^* = L^*$

② $\emptyset^* = \epsilon$

③ $\epsilon^* = \epsilon$

④ $L^+ = LL^* = L^*L$

⑤ $L^* = L^+ + \epsilon$

⑥ $L? = \epsilon + L$

⑦ $(L+M)^* = (L^*M^*)^* = (L^* + M^*)^*$

⑧ $L + LL^* = LL^* + + L = L^*$

⑨ $(PQ)^*P = P(PQ)^*$

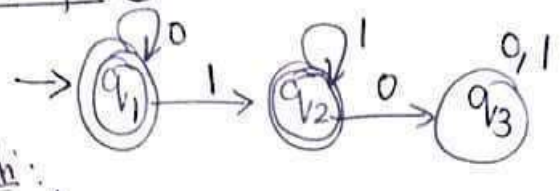
⑩ $(P+Q)R = PR + QR$

⑪ $R(P+Q) = RP + RQ$.

Conversion of F.A to Regular expression.

Ardeny theorem

Example: (1) construct R.E for the given DFA



Soln:

Equation for $q_1 = q_1 0 + \epsilon$

$q_2 = q_1 1 + q_2 1$

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

we solve q_1 & q_2 as they are final states

$q_1 = q_1 0 + \epsilon$ $(R = Q + RP)$

$R = q_1$, $Q = \epsilon$, $P = 0$ $(R = QP^*)$

$q_1 = \epsilon 0^*$ $(\epsilon R = R)$

$\boxed{q_1 = 0^*}$

$q_2 = q_1 1 + q_2 1$

$q_2 = 0^* 1 + q_2 1$

$R = Q + RP$

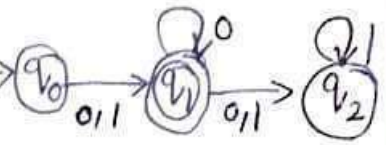
$R = q_2$ $Q = 0^*1$ $P = 1$

$R = QP^*$

$q_2 = 0^* 1 1^*$ $(RR^* = R^+)$

$\boxed{q_2 = 0^* 1^+}$

(2) Find R.E for the NFA



Solution: equation for

$q_0 = \epsilon$

$q_1 = q_0(0+1) + q_1 0$

$q_2 = q_1(0+1) + q_2 1$

As q_1 is final state, we solve for q_1

$q_1 = q_0(0+1) + q_1 0$

$q_1 = \epsilon(0+1) + q_1 0$

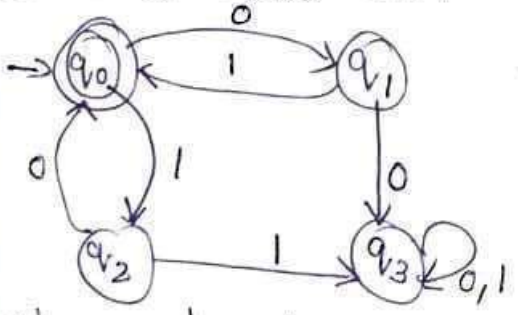
$R = q_1$ $Q = \epsilon(0+1)$ $P = 0$

$R = QP^*$

$q_1 = \epsilon(0+1)0^*$ $[\epsilon R = R]$

$\boxed{q_1 = (0+1)0^*}$

(3) construct R.E for DFA



Solution: Equations for

$q_0 = q_1 1 + q_2 0 + \epsilon$

$q_1 = q_0 0$ $q_2 = q_0 1$

$q_3 = q_1 0 + q_2 1 + q_3(0+1)$

← As q_0 is final state solve q_0

$q_0 = q_1 1 + q_2 0 + \epsilon$

$= q_0 01 + q_0 10 + \epsilon$ $[R = QP^*]$

$= q_0(01 + 10) + \epsilon$

$\boxed{q_0 = \epsilon(01+10)^*}$

Pumping lemma for Regular languages:

→ Pumping lemma is a powerful technique used for proving certain languages are not regular.

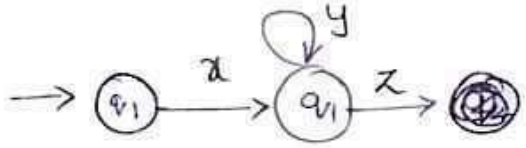
Theorem:

- Let L be a regular language → for a DFA
- Then there exists a constant 'n' → number of states of DFA such that for every string w in L, $|w| \geq n$

we can break w in to 3 strings, $w = xyz$ such that

1. $y \neq \epsilon$ (or) $|y| > 0$
2. $|xy| \leq n$
3. For all $K \geq 0$, the string xy^kz is also in L.

Example: $n = 3$ $\quad w = abc$



Prove that $L = \{0^n 1^n \mid n \geq 1\}$ is not regular.

→ Assume L is regular.

- Let n be a constant.

  Let $w = 0^n 1^n$ $\quad |w| \geq n$

  split $w = xyz$ such that

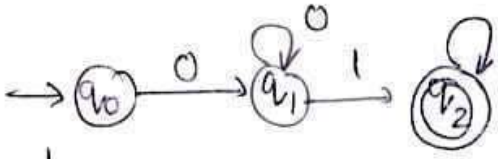
  1. $y \neq \epsilon$
  2. $|xy| \leq n$
  3. For all $K \geq 0$, $xy^kz \in L$

For given language the strings are 01, 0011, 000111, ...

- All strings having equal number of 1's followed by equal number of 1's.

Finite automata has limited memory, it cannot remember whether it read 'n' no. of zeros or 'm' no. of zeros

- consider DFA for given string



- Consider string three zero's, after reading these the state 'q1 is reached, now starts reading 1, but it cannot remember whether it has read 3, or 4, or 5 zero's. It cannot match number of zero's and 1's.

- Above DFA is acceptable for language

$$L = \{0^n 1^m \mid n \geq 1, m \geq 1\}$$

But the above DFA is not acceptable for language $L = \{0^n 1^n \mid n \geq 1\}$. For this language we cannot construct DFA. To prove this formally we need pumping lemma.

- Consider the string $w = 0^n 1^n$ & $n = 2$ then $w = 0011$.

- Split $w = xyz$ $\quad w = \underline{00}11$

$xy = 00$, $y = 0$, $x = 0$ and $z = 11$

Assume $k = 2$ then $xy^2z = 00011 \notin L$ as no. of 0's is not equal to no. of 1's.

Hence $L = \{0^n 1^n \mid n \geq 1\}$ is not regular.

Example: 2 show that $L = \{ b^{i^2} \mid i > 1 \}$ is not regular.

Proof: Assume that b^{i^2} is a regular language.

b^{i^2} length of the string $|w| = n^2$

Consider the pumping length is n

$n^2 > n$

Regular language for $L = \{ b^{i^2} \mid i > 1 \}$ is $\{ b^{2^2}, b^{3^2}, b^{4^2}, \ldots \}$

$= \{ b^4, b^9, b^{16}, \ldots \} = \{ bbbb, bbbbbbbbb, \text{---} \}$

condition 1: for $i \geq 0$, $xy^iz \in L$

$\underset{x}{b}\,\underset{y}{bb}\,\underset{z}{b}$

$x = b$, $y = bb$, $z = b$

Pumping y take $i = 0$, $(bb)^0 = \phi \Rightarrow bb$

take $i = 1$, $(bb)^1 = bb \Rightarrow bbbb$

take $i = 2$ $(bb)^2 = bbbb \Rightarrow bbbbbb$.

for $i = 0$ & $i = 2$ it is clear that our assumption is wrong.

Hence $L = \{ b^{i^2} \mid i > 1 \}$ is not regular.

Example 3: show that $\{ a^n b^{2n} \mid n > 0 \}$ is not regular.

Assume that L is regular & $L = \{ abb, aabbbb, aaabbbbbb, \ldots \}$

Length of the language is $n + 2n = 3n = |w|$

length of pumping lemma $= n$

$|w| > n$

take $n = 2$

$a^n b^{2n} = a^2 b^4 \Rightarrow \underset{x}{aa}\,\underset{y}{bb}\,\underset{z}{bb}$

case(i) y in b

$x = aa \quad y = bb \quad z = bb$

for $i \geq 0 \quad xy^iz \in L$

for $i = 0$

$aa(bb)^0bb = aabb \notin L$

case (ii) y in a

$\underbrace{aa}_{x}\underbrace{a}_{y}\underbrace{bbbb}_{z}$

$x = a$, $y = a$ & $z = bbbb$

- for $i = 0$

$xy^iz = a(a)^0bbbb = abbbb \notin L$

- for $i = 2$

$xy^2z = aaabbbb \notin L$

case (iii) y in a & b

$\underbrace{a}_{x}\underbrace{abbb}_{y}\underbrace{b}_{z}$

$x = a \quad y = abbb \quad z = bb$

for $i = 0$ ~~abbb~~ $ab \notin L$

$i = 2 \quad aabbbabbbb \notin L$

Since the language $\{a^nb^{2n} \mid n > 0\}$ is not regular.

Example 4:

Show that the language $L = \{a^ib^{2i} \mid i > 0\}$ is not regular.

Solution: The set of strings accepted by language L is

$L = \{abb, aabbbb, aaabbbbbb, \ldots\}$

Applying pumping lemma for any of the strings above.

Take the string abb

It is of the form xyz.

where $|xy| \leq i$, $|y| \geq 1$

To find i such that $xy^i z \notin L$

Take $i = 2$ here, then

$xyz = abb$

$xy^2z = a(bb)b$

$= abbb$

Hence $xy^2z = abbb \notin L$

Since abbb is not present in the strings of L.

$\therefore$ L is not regular.

Closure properties of Regular languages:

→ closure properties on regular languages are defined as certain operations on regular language which are guaranted to produce regular language.

→ closure refers to some language resulting in a new Languages that is of same "type" as originals operated i.e., regular.

consider L_1 & L_2 are regular languages, then

① The union of two regular languages $(L_1 \cup L_2)$ is regular.

② The intersection of two regular languages $(L_1 \cap L_2)$ is regular.

③ The complement of two regular languages is regular, L complement $(\bar{L})$ is $\Sigma^* - L$.

④ Reversal of a regular language is regular ie. L^R.

$L = \{0, 01, 100\}$ $L^R = \{0, 10, 001\}$

(5) Kleen closure: R is a regular expression from the languages L & M then R^* is a regular expression whose language is L^*.

(6) Positive closure: R is a regular expression from the language L & M, then R^+ is a regular expression whose language is L^+.

(7) Set difference operator:
$L_1 - L_2$ is regular language, strings in L_1 but not in L_2.

(8) Homomorphism:
A homomorphism on an alphabet is a function that gives a string for each symbol in that alphabet.

ex: $h(0) = ab$, $h(1) = \epsilon$ extend to strings by

$h(a_1 \ldots a_n) = h(a_1)\, h(a_2) \ldots h(a_n)$

ex: $h(01010) = abababab$

if L is a regular language, and h is homomorphism on its alphabet then $h(L) = \{h(w) \mid w \text{ is in } L\}$ is also a regular language.

(9) Inverse homomorphism:

$h^{-1}(L) = \{w \mid h(w) \text{ is in } L\}$

Inverse homomorphism of a language raise to regular language.

Decision Properties of Regular languages:

These are the algorithms that define the formal description of a regular language.

(i) Emptiness
(ii) Non-emptiness
(iii) Finiteness
(iv) Infiniteness
(v) Membership
(vi) Equality

Emptiness and Non-emptiness:

step 1: The states that cannot be reached from initial states delete them (ie; delete unreachable states)
step 2: If the finite automata contains any final state the language is not empty.
step 3: If the finite automata contains null final states the regular languages is empty.

Finite & Infiniteness:

step 1: Delete unreachable states.
step 2: Delete dead states as we cannot move from the dead state to the final state.
step 3: If the finite automata contains any loops then Regular language is infinite.
step 4: If the finite automata cannot contain any loops then the regular language is finite.

Membership:

If the string over an alphabet is accepted by the machine then we say that the string is member of the regular language.
Let us say the string from a language is w, it starts from the begining state and after certain transition if it reaches to final state then the string is accepted.

Equality:

let the automatas M_1 & M_2 are said to be equal if they have same regular languages.
The automatas are minimized to derive unique properties in both of them.

Minimization of Automata:

DFA minimization stands for converting a given DFA to its equivalent DFA with minimum number of states.
5 tuple notation for DFA $<Q, \Sigma, q_0, \delta, F>$, after minimization are $<Q', \Sigma, q_0, \delta', F'>$.
step 1: we will divide Q (set of states) in to two sets, one set contains all final states and other contains non-final states.
This partition is called P_0.
step 2: Initialize $K = 1$
step 3: Find P_K by partitioning the different sets of P_{K-1}. we will take all possible pair of

states If two states are distinguishable, we will split the sets into different sets in P_K.

step4: step when $P_K = P_{K-1}$ (No change in partition)

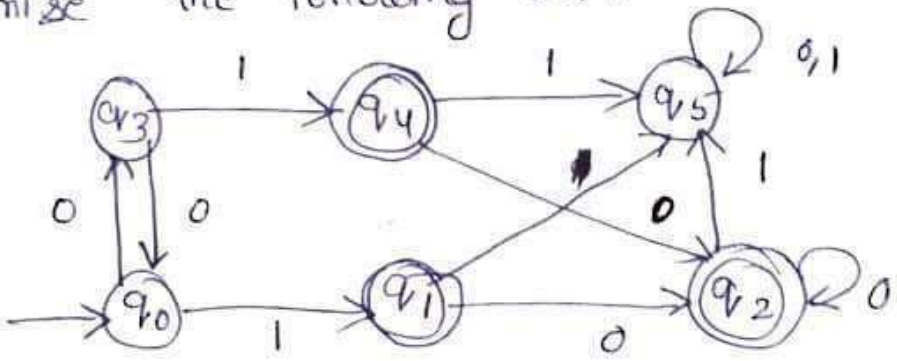
step5 And states of one set are merged into one. Number of states in minimized DFA will be equal to number of sets in P_K.

Distinguishable. If two states are not equivalent, then we say they are distinguishable i.e, two states P & q are distinguishable if there is atleast one string w such that one of $\hat{\delta}(P,w)$ & $\hat{\delta}(q,w)$ is accepting and other is not accepting.

How to find two states are distinguishable:

Two states (q_i, q_j) are distinguishable in partition P_K if for any input a, $\delta(q_i, a)$ and $\delta(q_j, a)$ are in different sets in partition P_{K-1}.

Ex: Minimize the following DFA



step1: $P_0 = \{\{q_1, q_2, q_4\}, \{q_0, q_3, q_5\}\}$

step2: To calculate P_1, we will check whether sets of partition P_0 can be partitioned or not.

(i) for set $\{q_1, q_2, q_4\}$ pairs are

(q_1, q_2) (q_1, q_4) (q_2, q_4)

$(q_1, q_2) \Rightarrow \left.\begin{aligned} \delta(q_1, 0) &= q_2 \\ \delta(q_2, 0) &= q_2 \end{aligned}\right\}$ not distinguishable (i.e, equivalent)

$\left.\begin{aligned} \delta(q_1, 1) &= q_5 \\ \delta(q_2, 1) &= q_5 \end{aligned}\right\}$ not distinguishable.

$(q_1, q_4) \Rightarrow \left.\begin{aligned} \delta(q_1, 0) &= q_2 \\ \delta(q_4, 0) &= q_2 \end{aligned}\right\}$

$\left.\begin{aligned} \delta(q_1, 1) &= q_5 \\ \delta(q_4, 1) &= q_5 \end{aligned}\right\}$ not distinguishable

$(q_2, q_4) \Rightarrow \left.\begin{aligned} \delta(q_2, 0) &= q_2 \\ \delta(q_4, 0) &= q_2 \end{aligned}\right\}$

$\left.\begin{aligned} \delta(q_2, 1) &= q_5 \\ \delta(q_4, 1) &= q_5 \end{aligned}\right\}$ not distinguishable.

(ii) for set $\{q_0, q_3, q_5\}$ pairs are

(q_0, q_3) (q_0, q_5) (q_3, q_5)

$(q_0, q_3) \Rightarrow \left.\begin{aligned} \delta(q_0, 0) &= q_3 \\ \delta(q_3, 0) &= q_0 \end{aligned}\right\}$ not distinguishable.

$\left.\begin{aligned} \delta(q_0, 1) &= q_1 \\ \delta(q_3, 1) &= q_4 \end{aligned}\right\}$ not distinguishable

$(q_0, q_5) \Rightarrow \left.\begin{aligned} \delta(q_0, 0) &= q_3 \\ \delta(q_5, 0) &= q_5 \end{aligned}\right\}$ not distinguishable.

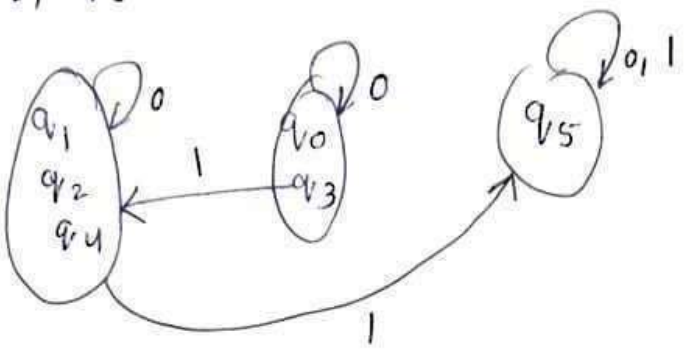
$\delta(q_0, 1) = q_1$ } distinguishable
$\delta(q_5, 1) = q_5$ }

we can partition the set $\{q_0, q_3, q_5\} \Rightarrow \{q_0, q_3\}, \{q_5\}$

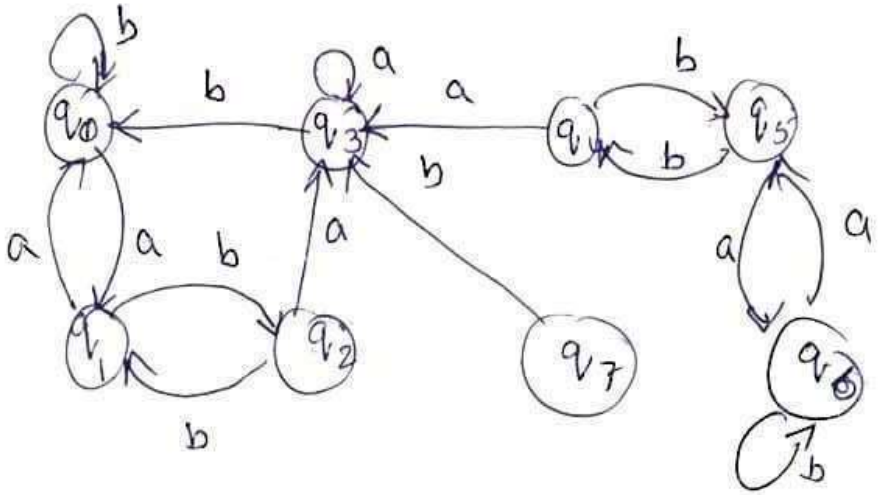
$\{q_3, q_5\} = \delta(q_3, 0) = q_0$ } not distinguishable
$\delta(q_5, 0) = q_5$

$\delta(q_3, 1) = q_4$ } not distinguishable
$\delta(q_5, 1) = q_5$

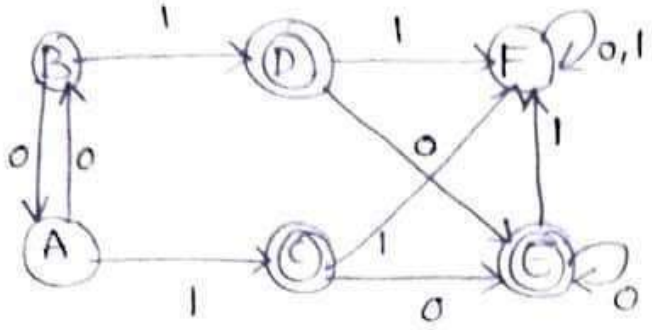
$\{q_1, q_2, q_4\}$ $\{q_0, q_3\}$ $\{q_5\}$



Example 2 minimize the following DFA



Minimization of DFA - Table Filling method
(Myhill - Nerode Theorem)



steps:

① Draw a table for all pairs of states (P, Q)

② Mark all pairs where $P \in F$ and $Q \notin F$ (ie., one state belongs to final state & other belongs to non-final state).

③ If there are any unmarked pairs (P, Q) such that $\delta(P, w)$, $\delta(Q, w)$ is marked, then mark [P, Q], where 'w' is input symbol.

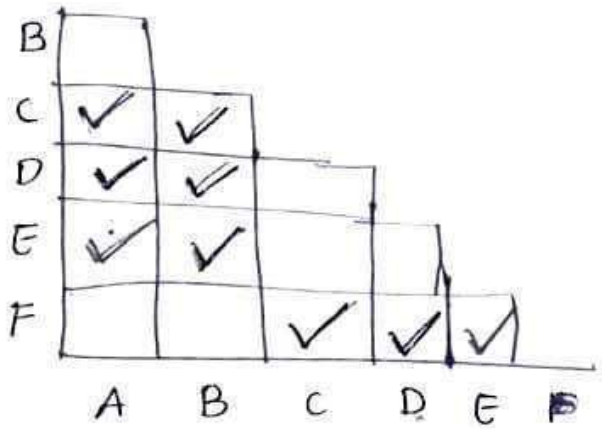
Repeat this until no more markings can be made.

④ Combine all the unmarked pairs and make them a single state in the minimized DFA.

Follow above steps for given F.A above.

step! E2

steps:



Step 3:

Now take the pairs that are unmarked ie, [B, A], [F, A], [F, B], [D, C], [E, C] [E, D]

→ [B, A] - $\delta(B,0) = A$, $\delta(A,0) = B$ } pair is not marked; $\delta(B,1) = D$, $\delta(A,1) = C$ } pair is not marked.

→ [F, A] - $\delta(F,0) = F$, $\delta(A,0) = B$ } Pair is unmarked; $\delta(F,1) = F$, $\delta(A,1) = C$ } Pair is marked

since [F, C] is marked mark [F, A] in table

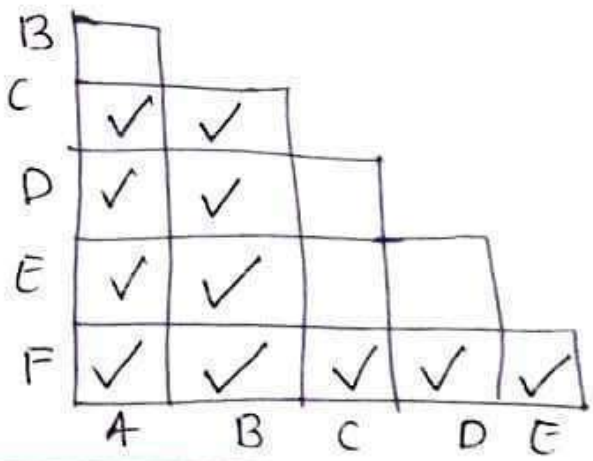
→ [F, B] - $\delta(F,0) = F$, $\delta(B,0) = A$ } pair is marked mark [F, B]; $\delta(F,1) = F$, $\delta(B,1) = D$ } pair is marked then mark [F, B]

→ [D, C] - $\delta(D,0) = E$, $\delta(C,0) = E$ } not marked; $\delta(D,1) = F$, $\delta(C,1) = F$ } unmarked

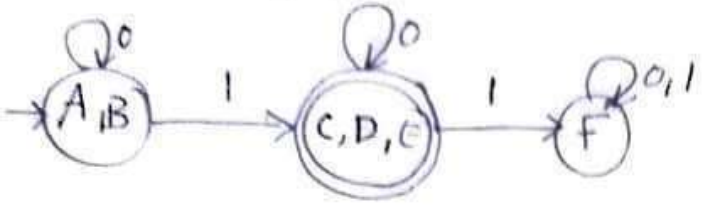
→ [E, C] - $\delta(E,0) = E$, $\delta(C,0) = E$ } not marked; $\delta(E,1) = F$, $\delta(C,1) = F$ } not marked

→ [E, D] = $\delta(E,0) = E$, $\delta(D,0) = E$ } not marked; $\delta(E,1) = F$, $\delta(D,1) = F$ } not marked

Finally the table after marking new pairs [F, A] & [F, B]



Step4 combine all the unmarked pairs and make them a single state i.e., unmarked pairs are [A,B], [C,D], [C,E] [D,E]



is the minimized DFA

3.2 REGULAR EXPRESSIONS

The languages accepted by FA are regular languages and these languages are easily described by simple expressions called regular expressions. We have some algebraic notations to represent the regular expressions.

Regular expressions are means to represent certain sets of strings in some algebraic manner and regular expressions describe the language accepted by FA.

If Σ is an alphabet then regular expression(s) over this can be described by following rules.

1. Any symbol from Σ, $\in$ *and* ϕ are regular expressions.
2. If r_1 and r_2 are two regular expressions then *union* of these represented as $r_1 \cup r_2$ or $r_1 + r_2$ is also a regular expression
3. If r_1 and r_2 are two regular expressions then *concatenation* of these represented as $r_1 r_2$ is also a regular expression.
4. The Kleene closure of a regular expression r is denoted by r^* is also a regular expression.
5. If r is a regular expression then (r) is also a regular expression.
6. The regular expressions obtained by applying rules 1 to 5 once or more than once are also regular expressions.

Examples :

(1) If $\Sigma = \{a, b\}$, then

(a) a is a regular expression (Using rule 1)
(b) b is a regular expression (Using rule 1)
(c) $a + b$ is a regular expression (Using rule 2)
(d) b^* is a regular expression (Using rule 4)
(e) ab is a regular expression (Using rule 3)
(f) $ab + b^*$ is a regular expression (Using rule 6)

(2) Find regular expression for the following

(a) A language consists of all the words over $\{a, b\}$ ending in b.

(b) A language consists of all the words over $\{a, b\}$ ending in bb.

(c) A language consists of all the words over $\{a, b\}$ starting with a and ending in b.

(d) A language consists of all the words over $\{a, b\}$ having bb as a substring.

(e) A language consists of all the words over $\{a, b\}$ ending in aab.

Solution : Let $\Sigma = \{a, b\}$, and

All the words over $\Sigma = \{\in, a, b, aa, bb, ab, ba, aaa, \ldots\} = \Sigma^*$ *or* $(a + b)^*$ *or* $(a \cup b)^*$

Now let us compute for final state, which denotes the regular expression.

r_{11}^2 will be computed, because there are total 2 states and final state is q_1 whose start state is q_1.

$$r_{12}^2 = (r_{12}^1)(r_{22}^1)^*(r_{22}^1) + (r_{12}^1)$$

$$= 0(\epsilon)^*(\epsilon) + 0$$

$$= 0 + 0$$

$r_{11}^2 = 0$ which is a final regular expression.

3.6.1 Arden's Method for Converting DFA to RE

As we have seen the Arden's theorem is useful for checking the equivalence of two regular expressions, we will also see its use in conversion of DFA to RE.

Following algorithm is used to build the r. e. from given DFA.

1. Let q_0 be the initial state.
2. There are $q_1, q_2, q_3, q_4, \ldots q_n$ number of states. The final state may be some q_j where $j \leq n$.
3. Let α_{ij} represents the transition from q_i to q_j.
4. Calculate q_i such that

$$q_i = \alpha_{ij} \cdot q_j$$

If q_i is a start state

$$q_i = \alpha_{ij} \cdot q_j + \epsilon$$

5. Similarly compute the final state which ultimately gives the regular expression r.

Example 1 : Construct RE for the given DFA.

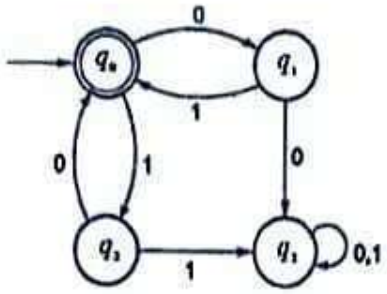
Solution :

Since there is only one state in the finite automata let us solve for q_0 only.

$$q_0 = q_0 0 + q_0 1 + \epsilon$$

$$q_0 = q_0(0 + 1) + \epsilon$$

Example 3 : Construct RE for the DFA given in below figure.



Solution : Let us see the equations

$$q_0 = q_1 1 + q_2 0 + \epsilon$$
$$q_1 = q_0 0$$
$$q_2 = q_0 1$$
$$q_3 = q_1 0 + q_2 1 + q_3 (0+1)$$

Let us solve q_0 first,

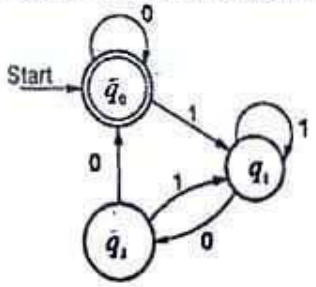
$$q_0 = q_1 1 + q_2 0 + \epsilon$$
$$q_0 = q_0 01 + q_0 10 + \epsilon$$
$$q_0 = q_0 (01+10) + \epsilon \qquad \because R = Q + RP$$
$$q_0 = \epsilon (01+10)^* \qquad \Rightarrow QP^* \text{ where}$$
$$q_0 = (01+10)^* \qquad R = q_0, Q = \epsilon, P = (01+10)$$

Thus the regular expression will be

$$r = (01+10)^*$$

Since q_0 is a final state, we are interested in q_0 only.

Example 4 : Find out the regular expression from given DFA.



$= (\{\epsilon, a, b, aa, bb, \ldots\})^*$

$= \{\epsilon, a, b, aa, bb, ab, ba, aaa, bbb, abb, baa, aabb, \ldots\}$

= {All the words over $\{a, b\}$ }

$= (a + b)^*$

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

3.3 IDENTITIES FOR REs

The two regular expressions P and Q are equivalent (denoted as P = Q) if and only if P represents the same set of strings as Q does. For showing this equivalence of regular expressions we need to show some identities of regular expressions.

Let P, Q and R are regular expressions then the identity rules are as given below

1.	$\epsilon R = R\epsilon = R$	
2.	$\epsilon^* = \epsilon$	ϵ is null string
3.	$(\phi)^* = \epsilon$	ϕ is empty string.
4.	$\phi R = R\phi = \phi$	
5.	$\phi + = R = R$	
6.	$R + R = R$	
7.	$RR^* = R^* R = R^+$	
8.	$(R^*)^* = R^*$	
9.	$\epsilon + RR^* = R^*$	
10.	$(P + Q)R = PR + QR$	
11.	$(P + Q)^* = (P^*Q^*) = (P^* + Q^*)^*$	
12.	$R^*(\epsilon + R) = (\epsilon + R)R^* = R^*$	
13.	$(R + \epsilon)^* = R^*$	
14.	$\epsilon + R^+ = R^*$	
15.	$(PQ)^* P = P(QP)^*$	
16.	$R^+ R + R = R^* R$	

3.3.1 Equivalence of two REs

Let us see one important theorem named Arden's Theorem which helps in checking the equivalence of two regular expressions.

Ardon's Theorem : Let P and Q be the two regular expressions over the input set Σ. The regular expression R is given as

$$R = Q + RP$$

Which has a unique solution as $R = QP^*$

Proof : Let, P and Q are two regular expressions over the input string Σ.
If P does not contain ϵ then there exists R such that

$$R = Q + RP \qquad \ldots (1)$$

We will replace R by QP^* in equation 1.
Consider R. H. S. of equation 1.

$$= Q + QP^*P$$

$$= Q(\epsilon + P^*P)$$

$$= QP^* \qquad \because \epsilon + R^*R = R^*$$

Thus $R = QP^*$

is proved. To prove that $R = QP^*$ is a unique solution, we will now replace L.H.S. of equation 1 by Q + RP. Then it becomes

$$Q + RP$$

But again R can be replaced by Q + RP.

$$\therefore \quad Q + RP = Q + (Q + RP)P$$

$$= Q + QP + RP^2$$

Again replace R by Q + RP.

$$= Q + QP + (Q + RP)P^2$$

$$= Q + QP + QP^2 + RP^3$$

Thus if we go on replacing R by Q + RP then we get,

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

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

From equation 1,

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

Where $i \geq 0$

Consider equation 2,

$$R = Q\underbrace{(\epsilon + P + P^2 + \ldots\ldots + P^i)}_{P^*} + RP^{i+1}$$

$$\therefore \quad R = QP^* + RP^{i+1}$$

Let w be a string of length i.

$= \{\epsilon, 0, 00, 1, 11, 111, 01, 10, \ldots\ldots\}$

$=$ { ϵ, any combination of 0's, any combination of 1's, any combination of 0 and 1 }

Hence, L. H. S. $=$ R. H. S. is proved.

3.4 RELATIONSHIP BETWEEN FA AND RE

There is a close relationship between a finite automata and the regular expression we can show this relation in below figure.

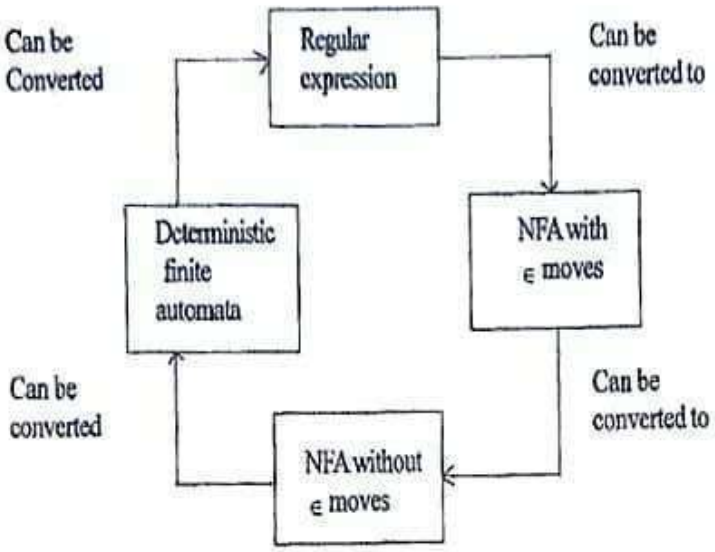


FIGURE : Relationship between FA and regular expression

The above figure shows that it is convenient to convert the regular expression to NFA with ϵ moves. Let us see the theorem based on this conversion.

3.5 CONSTRUCTING FA FOR A GIVEN REs

Theorem : If r be a regular expression then there exists a NFA with ϵ - moves, which accepts $L(r)$.

Proof : First we will discuss the construction of NFA M with ϵ - moves for regular expression r and then we prove that $L(M) = L(r)$.

Let r be the regular expression over the alphabet Σ.

Construction of NFA with ϵ - moves

Case 1 :

(i) $r = \phi$

NFA $M = (\{s, f\}, \{\ \}\delta, s, \{f\})$ as shown in Figure1 (a)

(No path from initial state s to reach the final state f.)

Figure 1 (a)

(ii) $r = \epsilon$

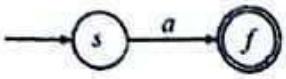
NFA $M = (\{s\}, \{\ \}, \delta, s, \{s\})$ as shown in Figure 1 (b)

(The initial state s is the final state)

Figure 1 (b)

(iii) $r = a$, for all $a \in \Sigma$,

NFA $M = (\{s, f\}, \Sigma, \delta, s, \{f\})$



(One path is there from initial state s to reach the final state f with label a.)

Figure 1 (c)

Case 2 : $|r| \geq 1$

Let r_1 and r_2 be the two regular expressions over Σ_1, Σ_2 and N_1 and N_2 are two NFA for r_1 and r_2 respectively as shown in Figure 2 (a).

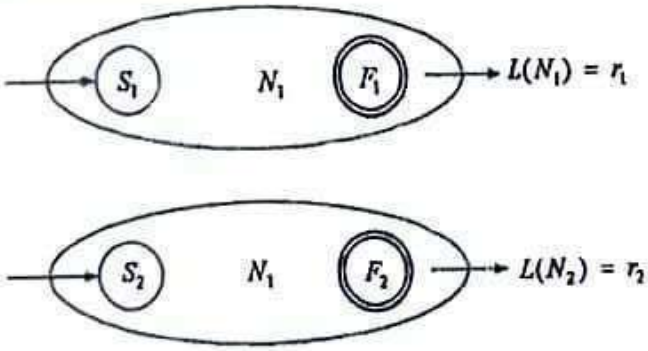


Figure 2 (a) NFA for regular expression r_1 and r_2

Now let us compute for final state, which denotes the regular expression.

r_{12}^1 will be computed, because there are total 2 states and final state is q_1 whose start state is q_0.

$$r_{12}^2 = (r_{12}^1)(r_{22}^1)^*(r_{22}^1) + (r_{12}^1)$$
$$= 0(\in)^*(\in) + 0$$
$$= 0 + 0$$

$r_{12}^2 = 0$ which is a final regular expression.

3.6.1 Arden's Method for Converting DFA to RE

As we have seen the Arden's theorem is useful for checking the equivalence of two regular expressions, we will also see its use in conversion of DFA to RE.

Following algorithm is used to build the r. e. from given DFA.

1. Let q_0 be the initial state.
2. There are $q_1, q_2, q_3, q_4, \ldots q_n$ number of states. The final state may be some q_j where $j \leq n$.
3. Let α_{ij} represents the transition from q_i to q_j.
4. Calculate q_i such that

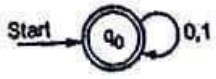
$$q_i = \alpha_{ij} \cdot q_j$$

If q_i is a start state

$$q_i = \alpha_{ij} \cdot q_j + \in$$

5. Similarly compute the final state which ultimately gives the regular expression r.

Example 1 : Construct RE for the given DFA.

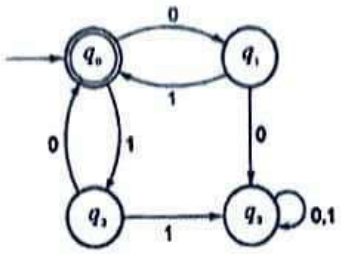


Solution :

Since there is only one state in the finite automata let us solve for q_0 only.

$$q_0 = q_0 0 + q_0 1 + \in$$
$$q_0 = q_0 (0 + 1) + \in$$

Example 3 : Construct RE for the DFA given in below figure.



Solution : Let us see the equations

$$q_0 = q_1 1 + q_2 0 + \epsilon$$
$$q_1 = q_0 0$$
$$q_2 = q_0 1$$
$$q_3 = q_1 0 + q_2 1 + q_3 (0+1)$$

Let us solve q_0 first,

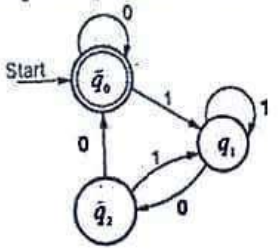
$$q_0 = q_1 1 + q_2 0 + \epsilon$$
$$q_0 = q_0 01 + q_0 10 + \epsilon$$
$$q_0 = q_0 (01+10) + \epsilon \qquad \because R = Q + RP$$
$$q_0 = \epsilon (01+10)^* \qquad \Rightarrow QP^* \text{ where}$$
$$q_0 = (01+10)^* \qquad R = q_0, Q = \epsilon, P = (01+10)$$

Thus the regular expression will be

$$r = (01+10)^*$$

Since q_0 is a final state, we are interested in q_0 only.

Example 4 : Find out the regular expression from given DFA.



Example 8 : Show that the language $L = \{a^i b^{i^2} | i > 0\}$ is not regular.

Solution : The set of strings accepted by language L is,

$L = \{abb, aabbbb, aaabbbbbb, aaaabbbbbbbb...\}$

Applying Pumping lemma for any of the strings above.

Take the string abb.

It is of the form uvw.

Where, $|uv| \leq i, |v| \geq 1$

To find i such that $uv^i w \notin L$

Take i = 2 here, then

$uv^2w = a(bb)b$

$= abbb$

Hence $uv^2w = abbb \notin L$

Since abbb is not present in the strings of L.

$\therefore$ L is not regular.

Example 9 : Show that $L = \{0^n | n$ is a perfect square $\}$ is not regular.

Solution :

Step 1 : Let L is regular by Pumping lemma. Let n be number of states of FA accepting L.

Step 2 : Let $z = 0^n$ then $|z| = n \geq 2$.

Therefore, we can write z = uvw; Where $|uv| \leq n, |v| \geq 1$.

Take any string of the language L = { 00, 0000, 000000 .... }

Take 0000 as string, here u = 0, v = 0, w = 00 to find i such that $uv^i w \notin L$.

Take i = 2 here, then

$uv^i w = 0(0)^2 00$

$= 00000$

This string 00000 is not present in strings of language L. So $uv^i w \notin L$.

$\therefore$ It is a contradiction.

3.9 PROPERTIES OF REGULAR SETS

Regular sets are closed under following properties.

1. Union
2. Concatenation

3. Kleene Closure
4. Complementation
5. Transpose
6. Intersection

1. **Union :** If R_1 and R_2 are two regular sets, then union of these denoted by $R_1 + R_2$ or $R_1 \cup R_2$ is also a regular set.

Proof : Let R_1 and R_2 be recognized by NFA N_1 and N_2 respectively as shown in Figure1(a) and Figure1(b).

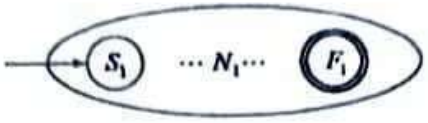


FIGURE 1(a) NFA for regular set R_1

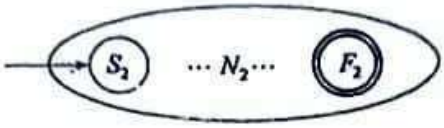


FIGURE 1(b) NFA for regular set R_2

We construct a new NFA N based on union of N_1 and N_2 as shown in Figure 1 (c)

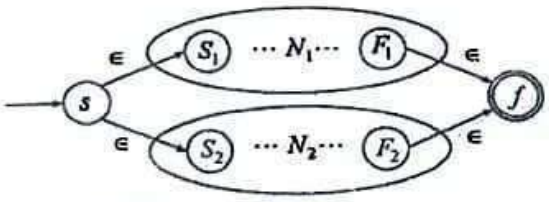


FIGURE 1(c) NFA for $N_1 + N_2$

Now,

$$L(N) = \in L(N_1) \in + \in L(N_2) \in$$
$$= \in R_1 \in + \in R_2 \in$$
$$= R_1 + R_2$$

Since, N is FA, hence $L(N)$ is a regular set (language). Therefore, $R_1 + R_2$ is a regular set.

2. **Concatenation :** If R_1 and R_2 are two regular sets, then concatenation of these denoted by R_1R_2 is also a regular set.

Proof : Let R_1 and R_2 be recognized by NFA N_1 and N_2 respectively as shown in Figure 2(a) and Figure 2(b).

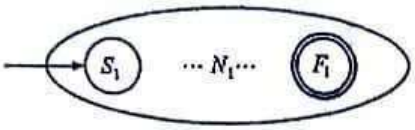


FIGURE 2(a) NFA for regular set R_1

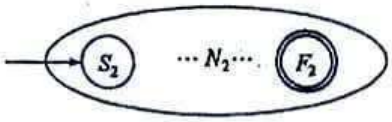


FIGURE 2(b) NFA for regular set R_2

We construct a new NFA N based on concatenation of N_1 and N_2 as shown in Figure2(c).

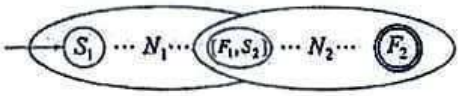


FIGURE 2(c) NFA for regular set R_1R_2

Now,

$L(N)$ = Regular set accepted by N_1 followed by regular set accepted by $N_2 = R_1R_2$

Since, $L(N)$ is a regular set, hence R_1R_2 is also a regular set.

3. **Kleene Closure :** If R is a regular set, then Kleene closure of this denoted by R^* is also a regular set.

Proof : Let R is accepted by NFA N shown in Figure 3(a).

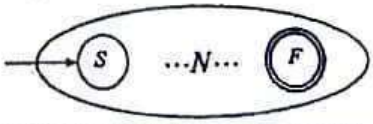


FIGURE 3(a) NFA for regular set R