UNIT-II: Regular Languages

Objective:

To familiarize how to employ regular expressions.

Syllabus:

Regular sets, regular expressions, identity rules, construction of finite Automata for a given regular expressions and its inter conversion, pumping lemma of regular sets, closure properties of regular sets (proofs not required),applications of regular languages.

Learning Outcomes:

Students will be able to:

- understand the regular sets and how to represent the regular expressions.
- construct finite Automata for a given regular expression and viceversa.
- list closure properties of regular languages.
- understand the different applications of regular languages.

2.Learning Material

2.1 Regular set:

A language is a regular set (or just regular) if it is the set accepted by some finite automaton.

Example:

L= {0, 1, 10, 00, 01, 11, 000, 101, ...} is a regular set representing any no of 0's and any no of 1's.

2.2 Regular expression:

The languages accepted by finite automata are easily described by simple expressions called regular expressions.

Let Σ be an alphabet. The regular expressions over Σ and the sets that they denote are defined recursively as follows.

1) Ø is a regular expression and denotes the empty set.

2) ε is a regular expression and denotes the set { ε }.

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 the languages R and S, respectively, then

(r + s), (rs), and (r*) are regular expressions that denote the sets R U S, RS, and R*, respectively.

2.2.1Some Examples on Regular expressions

1. Write regular expressions for each of the following languages over ∑={0, 1}.

a) The set representing {00}.

00

b) The set representing all strings of 0's and 1's.

(0+1)*

c) The set of all strings representing with at least two consecutive 0's.

(0 + 1)*00(0 + 1)*

d) The set of all strings ending in 011.

(0 + 1)*011

e) The set of all strings representing any number of 0's followed by any number of 1's followed by any number of 2's.

0*1*2*

f) The set of all strings starting with 011.

011 (0 + 1)*

2. Write regular expressions for each of the following languages over ∑={a, b}.

a) The set of all strings ending with either a or bb.

(a+b)* (a + bb)

b) The set of strings consisting of even no. of a's followed by odd no. of b's.

(aa)*(bb)*b

c) The set of strings representing even number of a's.

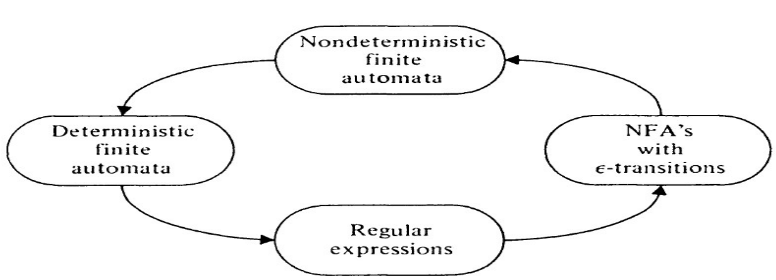
(b* a b* a b*) * + b*

2.3 Identity Rules Related to Regular Expressions

Given r, s and t are regular expressions, the following identities hold:

- $\emptyset^* = \varepsilon$
- $\varepsilon^* = \varepsilon$
- $r^+ = rr^* = r^*r$
- $r^*r^* = r^*$
- $(r^*)^* = r^*$
- $r + s = s + r$
- $(r + s) + t = r + (s + t)$
- $(rs)t = r(st)$
- $r(s + t) = rs + rt$
- $(r + s)t = rt + st$
- $(\varepsilon + r)^* = r^*$
- $(r + s)^* = (r^*s^*)^* = (r^* + s^*)^* = (r+s^*)^*$
- $r + \emptyset = \emptyset + r = r$
- $r\,\varepsilon = \varepsilon\, r = r$
- $\emptyset\, L = L\, \emptyset = \emptyset$
- $r + r = r$
- $\varepsilon + rr^* = \varepsilon + r^*r = r^*$

2.4 Construction of Finite automata for a given regular expression



Equivalence of Finite Automata and Regular Expressions

- The languages accepted by finite automata are precisely the languages denoted by regular expressions.
- For every regular expression there is an equivalent NFA with ε -transitions.
- For every DFA there is a regular expression denoting its language.

Let r be a regular expression. Then there exists an NFA with ε -transitions that accept L(r).

Zero operators:

The expression r must be ε, Ø, or a for some a in Σ. The NFA's for zero operators are

(a) $r = \epsilon$ (b) $r = \varnothing$ (c) $r = \mathbf{a}$

One or more operators:

Let r have i operators. There are three cases depending on the form of r.

Case 1: Union (r = rl + r2.)

There are NFA's M1 = **(Q1, ∑1, δ1, q1, {f1})** and M2=**(Q2, ∑2, δ2, q2, {f2})** with L(M1) = L(r1) and L(M2) = L(r2).

Construct

M = (Q1 ∪ Q2 ∪ {q0, f0}, ∑**1** ∪ ∑**2**, **δ, q0, {f0})** where **δ is defined by**

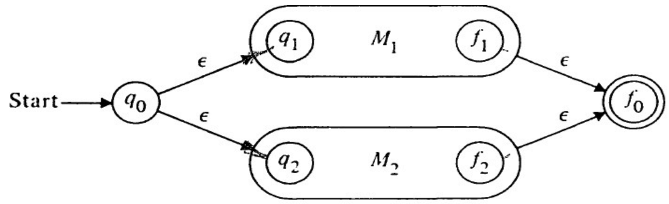
i) **δ (q0,** ε) = {q1,q2}

ii) **δ (q,** a) = **δ**1(q ,a) for q in Q1-{f1} and a in ∑**1** ∪ { ε }

iii) **δ (q,** a) = **δ**2(q ,a) for q in Q2-{f2} and a in ∑**2** ∪ { ε }

iv) **δ (f1,** ε) = **δ**1(f2, ε) = { f0 }

L(M) = L(M1) ∪ L(M2)



<u>Case 2:</u> Concatenation (r = r1 r2).

Let M1 and M2 be as in Case 1 and construct M = (Q1 ∪ Q2, **∑1** ∪ **∑2, δ, q1, {f2})**

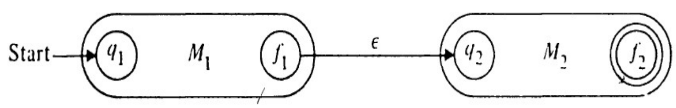
where **δ is defined by**

i) **δ (q,** a) = **δ**1(q ,a) for q in Q1-{f1} and a in ∑**1** ∪ { ε }

ii) **δ (f1,** ε) = {q2}

iii) **δ (q,** a) = **δ**2(q ,a) for q in Q2 and a in ∑**2** ∪ { ε }

L(M) ={xy| x is in L(M1) and y is in L(M2)} and L(M) = L(M1)L(M2)



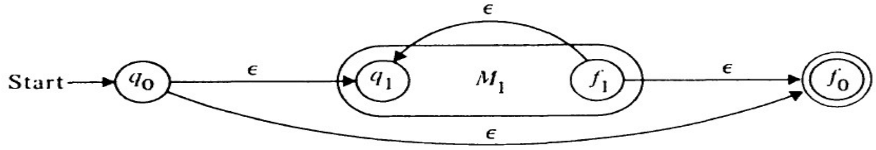
<u>Case 3:</u> Closure (r = r1*)

Let M1 = **(Q1, ∑1, δ1, q1, {f1}) and L(M1) = r1.**

Construct M = (Q1 ∪ {q0,f0}, **∑1** , **δ, q0, {f0}),** where **δ is defined by**

i) **δ (q0,** ε) = **δ (f1,** ε) = {q1,f0}

ii) **δ (q,** a) = **δ**1(q ,a) for q in Q1-{f1} and a in ∑**1** ∪ { ε }

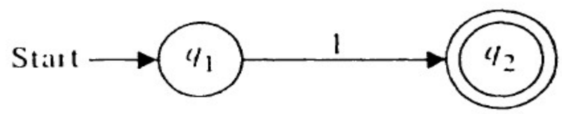


Example:

1. Construct an NFA for the regular expression 01*+1

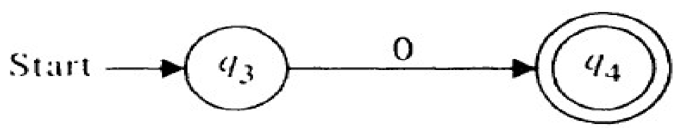
Regular expression is of the form r1 + r2, where r1 =01* and r2 = 1.

The automaton for r_2 is



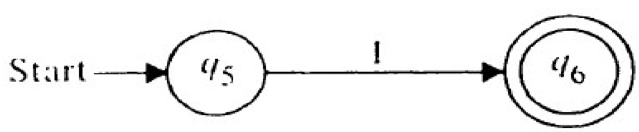
Express r_1 as r_3 and r_4, where r_3=0 and r_4= 1^*

The automaton for r_3 is

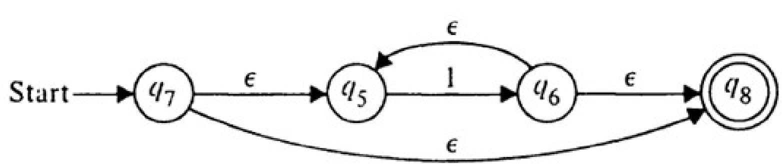


r_4 is r_5^* where r_5=1

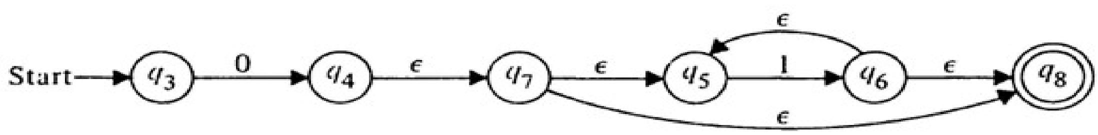
The NFA for r_5 is



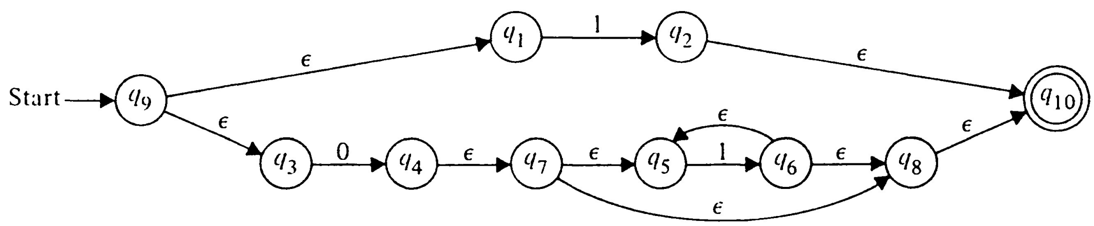
To construct an NFA for r4 = r_5^* use the construction of closure. The resulting NFA for r4 is



Then, for r1 = r3 r4 use the construction of concatenation.



Finally, use the construction of union to find the NFA for r = r1 + r2



2.5 Construction of regular expressions for the given finite Automata:

Arden's Theorem

Let P and Q be two regular expressions over Σ,and if P does not contain epsilon, then R=Q+RP has a unique solution R=QP*.

Procedure:

Assume the given finite automata should not contain any epsilons.

Step 1: Find the reachability for each and every state in given Finite automata.

Reachability of a state is the set of states whose edges enter into that state.

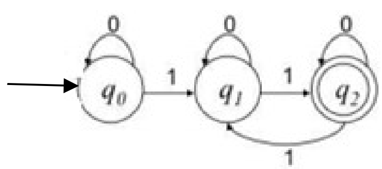
Step 2: For the initial state of finite automata ,add epsilon to the reachability equation.

Step 3: Solve the equations by using Arden's Theorem.

Step 4: Substitute the results of each state equation into the final state equation,to get the regular expression for the given DFA.

Example:

1. Construct regular expression for the given finite automaton.



The given Finite Automata is not having any ε's(epsilons).

Step 1: Find the reachability for each and every state in given Finite automata.

Reachability of a state is the set of states whose edges enter into that state.

$q_0 = q_0\, 0$ ——— 1

$q_1 = q_0\, 1 + q_1 0 + q_2\, 1$ ——— 2

$q_2 = q_1\, 1 + q_2 0$ ——— 3

Step 2: For the initial state of finite automata, add epsilon to the reachability equation.

$q_0 = q_0\, 0 + \varepsilon$

Step 3: Solve the equations by using Arden's Theorem.

After applying arden's theorem for equation 3

$q_2 = q_1\, 10^*$ ——— 4

Substitute equation 4 in equation 2

$q_1 = q_0\, 1 + q_1 0 + q_1\, 10^*$

$q_1 = q_0\, 1 + q_1(0+10^*)$ ——— 5

Apply arden's theorem on equation 5

$q_1 = q_0 1\, (0+10^*)^*$ ——— 6

Apply arden's theorem on equation 1

$q_0 = q_0\, 0 + \varepsilon$

$q_0 = \varepsilon\, 0^*$ ——— 7

Substitute equation 7 in equation 6

$q_1 = \varepsilon\, 0^*\, 1\, (0+10^*)^*$ ——— 8

Step 4: Substitute the results of each state equation into the final state equation, to get the regular expression for the given DFA.

$q_2 = \varepsilon\, 0^*\, 1\, (0+10^*)^*\, 10^*$

Therefore, the regular expression for the given DFA is **0* 1 (0+10*)* 10*.**

2.6 Pumping Lemma for Regular Sets:

- Pumping lemma, which is a powerful tool for proving certain languages non-regular.
- It is also useful in the development of algorithms to answer certain questions concerning finite automata, such as whether the language accepted by a given FA is finite or infinite.

Lemma

Let L be a regular set. Then there is a constant n such that if z is any word in L, and $|z| > n$, we may write z=uvw in such a way that $|uv| \leq n$, $v \geq 1$, and for all $i > 0$, uv^iw is in L. Furthermore, n is no greater than the number of states of the smallest FA accepting L.

Example:

The set L = $\{0^{i^2} \mid i$ is an integer, $i \geq 1]$, which consists of all strings of 0's whose length is a perfect square, is not regular.

Assume L is regular and let n be the integer in the pumping lemma.

Let $z = 0^{n^2}$.

By the pumping lemma, 0^{n^2}may be written as uvw, where $1 \leq |v| \leq n$ and uv^iw is in L for all i. Let i = 2, $n^2 < |uv^2w| < n^2 + n < (n+1)^2$.

That is, the length of uv^2w lies properly between n^2 and $(n+1)^2$, and is thus not a perfect square.

Thus uv^2w is not in L, a contradiction.

We conclude that L is not regular.

2.7 Closure Properties of Regular Sets:

- The regular sets are closed under union, concatenation, and Kleene closure.
- The class of regular sets is closed under complementation. That is, if L is a regular set and $L \subseteq \sum^*$, then $\sum^* - L$ is a regular set.
- The regular sets are closed under intersection.
- The class of regular sets is closed under substitution.
- The class of regular sets is closed under homomorphism and inverse homomorphism.
- The class of regular sets is closed under quotient with arbitrary sets.

2.8 Applications of Regular Languages:-

- Efficient string searching .
- Pattern matching with regular expressions (example: Unix grep utility)
- Lexical analysis (a.k.a. scanning, tokenizing) in a compiler.