Formal Language and Automata Theory

Chapter Three

Regular expression and regular language

Regular Expressions

- A language is regular if there exists accepter for it.
- The notation of regular expressions involves a combination of strings of symbols from some alphabet Σ, parentheses, and the operators +, **.**, and *.

Formal definition of RE

- Let Σ be a, given alphabet. Then
  1. **φ**,λ and a ε Σ are all regular expressions. These are called primitive regular expressions.
  2. If r_1 and r_2 are regular expressions, so are r_1+r_2, r_1*r_2, $r_1.r_2$, r_1*, and (r1).
  3. A string is a regular expression if and only if it can be derived from the primitive regular expressions by a finite number of applications of the rules in (2).

Languages associated with RE

A **Regular Expression** can be recursively defined as follows:

1. **ε** is a Regular Expression indicates the language containing an empty string. **(L (ε) = {ε})**
2. **φ** is a Regular Expression denoting an empty language. **(L (φ) = { })**
3. **x** is a Regular Expression where **L={x}**

Con't

4. If **X** is a Regular Expression denoting the language **L(X)** and **Y** is a Regular Expression denoting the language **L(Y)**, then

   a. **X + Y** is a Regular Expression corresponding to the language **L(X) U L(Y)** where **L(X+Y) = L(X) U L(Y)**.

   b. **X . Y** is a Regular Expression corresponding to the language **L(X) . L(Y)** where **L(X.Y)= L(X) . L(Y)**

   c. **R*** is a Regular Expression corresponding to the language **L(R*)** where **L(R*) = (L(R))***

5. If we apply any of the rules several times from 1 to 5, they are Regular Expressions.

Con’t

Example:

- For $\Sigma=\{a,b\}$, the expression

  $r = (a+ b)^* (a + bb)$
- is regular. It denotes the language

  $L (r) : \{ a, bb, aa, abb, ba, bbb, ... \}$
- We can see this by considering the various parts of r. The first part, $(a + b)^*$, stands for any string of a's and b's. The second part, $(a + bb)$ represents either an a or a double b. Consequently, $L (r)$ is the set of all strings on $\{a,b\}$, terminated by either an a or a bb.

Con’t

Some RE Examples

Regular Expression	Regular Set
(0+10*)	L= { 0, 1, 10, 100, 1000, 10000, … }
(0*10*)	L={1, 01, 10, 010, 0010, …}
(0+ε)(1+ ε)	L= {ε, 0, 1, 01}
(a+b)*	Set of strings of a’s and b’s of any length including the null

Con’t

	string. So L= { ε, 0, 1,00,01,10,11,.......}
(a+b)*abb	Set of strings of a’s and b’s ending with the string abb, So L = {abb, aabb, babb, aaabb, ababb, ...}
(11)*	Set 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 number 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, ...}

Regular Sets

- Any set that represents the value of the Regular Expression is called a **Regular Set**.

Properties of Regular Sets

Property 1. *The union of two regular set is regular.*

Proof:

Let us take two regular expressions

RE1 = a(aa)* and RE2 = (aa)*

So, L1= {a, aaa, aaaaa,.....} (Strings of odd length excluding Null)

and L2={ ε, aa, aaaa, aaaaaa,.......} (Strings of even length including Null)

L1 ∪ L2 = { ε,a,aa, aaa, aaaa, aaaaa, aaaaaa,.......}

(Strings of all possible lengths excluding Null)

RE (L1 ∪ L2) = a* (which is a regular expression itself)

Con’t

Property 2. *The intersection of two regular set is regular.*

Proof:

Let us take two regular expressions

RE1 = a(a*) and RE2 = (aa)*

So, L1 = { a,aa, aaa, aaaa, ...} (Strings of all possible lengths excluding Null)

L2 ={ ε, aa, aaaa, aaaaaa,.......} (Strings of even length including Null)

L1 ∩ L2 = { aa, aaaa, aaaaaa,.......} (Strings of even length excluding Null)

RE (L1 ∩ L2) = aa(aa)* which is a regular expression itself.

Con’t

Property 3. *The complement of a regular set is regular.*

Proof:

Let us take a regular expression:

RE = (aa)*

So, L = {ε, aa, aaaa, aaaaaa, ...} (Strings of even length including Null)

Complement of **L** is all the strings that is not in **L**.

So, L’ = {a, aaa, aaaaa, ...} (Strings of odd length excluding Null)

RE (L’) = a(aa)* which is a regular expression itself.

Con’t

Property 4. *The difference of two regular set is regular.*

Proof:

Let us take two regular expressions:

RE1 = a (a*) and RE2 = (aa)*

So, L1= {a,aa, aaa, aaaa, ...} (Strings of all possible lengths excluding Null)

L2 = { ε, aa, aaaa, aaaaaa,.......} (Strings of even length including Null)

L1 – L2 **= {**a, aaa, aaaaa, aaaaaaa, ...}

(Strings of all odd lengths excluding Null)

RE (L1 – L2) = a (aa)* which is a regular expression.

Con’t

Property 5. *The reversal of a regular set is regular.*

***Proof*:**

We have to prove **L(R)** is also regular if **L** is a regular set.

Let, L= {01, 10, 11, 10}

RE (L)= 01 + 10 + 11 + 10

L(R)= {10, 01, 11, 01}

RE (LR)= 01+ 10+ 11+10 which is regular

Con’t

Property 6. *The closure of a regular set is regular.*

***Proof*:**

If L = {a, aaa, aaaaa, ...} (Strings of odd length excluding Null)

i.e., RE (L) = a (aa)*

L*= {a, aa, aaa, aaaa , aaaaa,……………} (Strings of all lengths excluding Null)

RE (L*) = a (a)*

Con’t

Property 7. *The concatenation of two regular sets is regular.*

Proof:

Let RE1 = (0+1)*0 and RE2 = 01(0+1)*

Here, L1 = {0, 00, 10, 000, 010, ...} (Set of strings ending in 0)

and L2 = {01, 010,011,.....} (Set of strings beginning with 01)

Then, L1 L2 = {001,0010,0011,0001,00010,00011,1001,10010,.............}

Set of strings containing 001 as a substring which can be represented by an RE: (0+1)*001(0+1)*

Identities Related to Regular Expressions

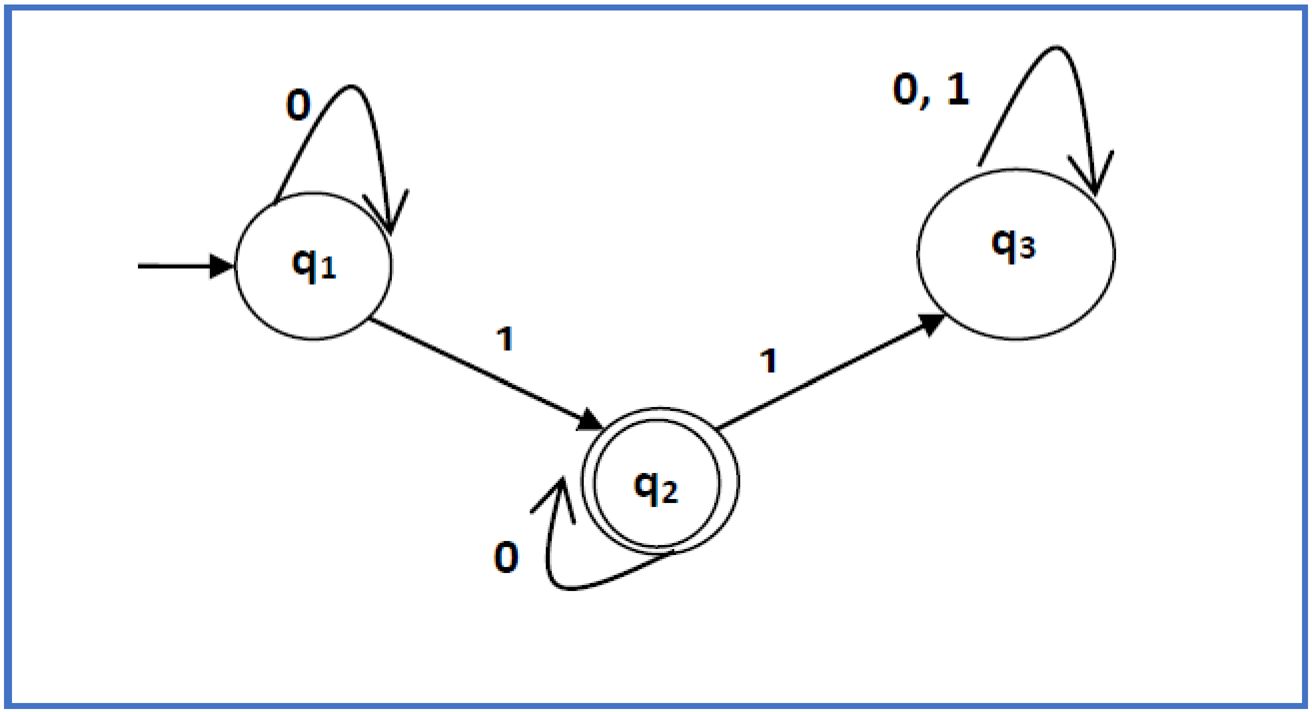
- Given R, P, L, Q as regular expressions, the following identities hold:

1. $\emptyset^* = \varepsilon$
2. $\varepsilon^* = \varepsilon$
3. $R^+ = RR^* = R^*R$
4. $R^*R^* = R^*$
5. $(R^*)^* = R^*$
6. $RR^* = R^*R$
7. $(PQ)^*P = P(QP)^*$
8. $(a+b)^* = (a^*b^*)^* = (a^*+b^*)^* = (a+b^*)^* = a^*(ba^*)^*$
9. $R + \emptyset = \emptyset + R = R$ (The identity for union)
10. $R\varepsilon = \varepsilon R = R$ (The identity for concatenation)
11. $\emptyset L = L\emptyset = \emptyset$ (The annihilator for concatenation)
12. $R + R = R$ (Idempotent law)
13. $L (M + N) = LM + LN$ (Left distributive law)
14. $(M + N) L = LM + LN$ (Right distributive law)
15. $\varepsilon + RR^* = \varepsilon + R^*R = R^*$

Construction of an RE from an FA

Problem

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



Finite automata

Con’t

Solution:

Here the initial state is q1 and the final state is q2

Now we write down the equations:

q1 = q10 + ϵ

q2 = q11 + q20

q3 = q21 + q30 + q31

Now, we will solve these three equations:

q1 = ϵ0* [As, εR = R]

So, q1 = 0* s

q2 = 0*1 + q20

So, q2 = 0*1(0)* [By Arden’s theorem]

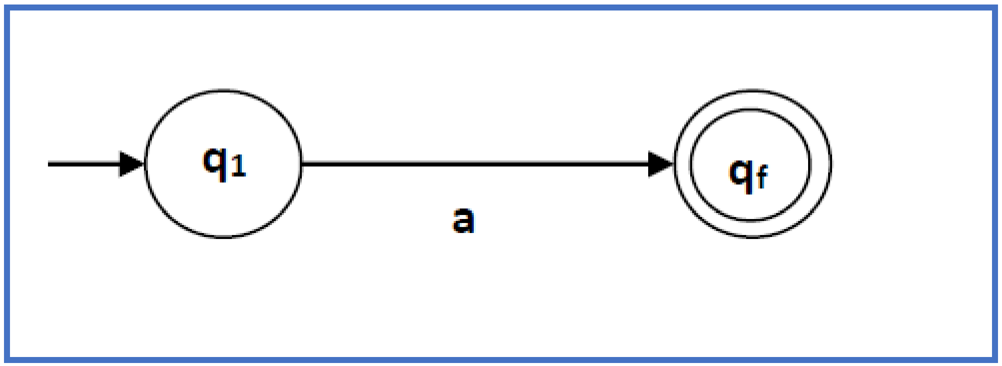
Hence, the regular expression is 0*10*.

Construction of an FA from an RE

- We can use Thompson's Construction to find out a Finite Automaton from a Regular Expression.
- We will reduce the regular expression into smallest regular expressions and converting these to NFA and finally to DFA.

Con't

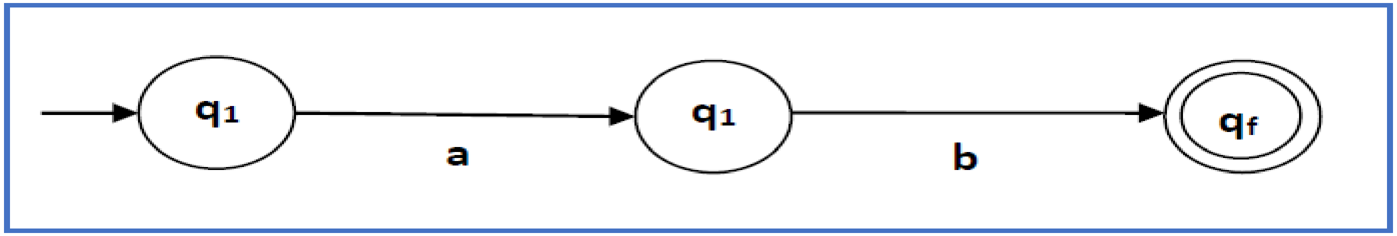
- ***Case 1*:** For a regular expression 'a', we can construct the following FA:



Finite automata for RE = a

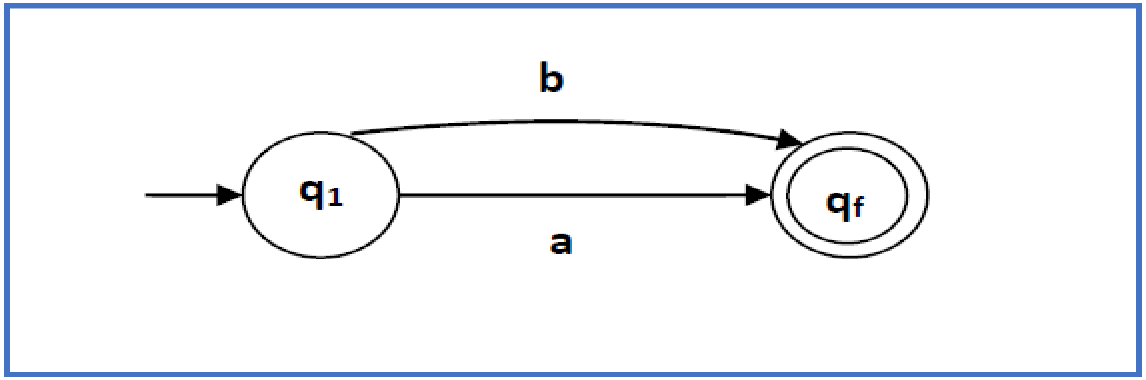
Con't

- ***Case 2*: For a regular expression 'ab', we can construct the following FA:**



Finite automata for RE = ab

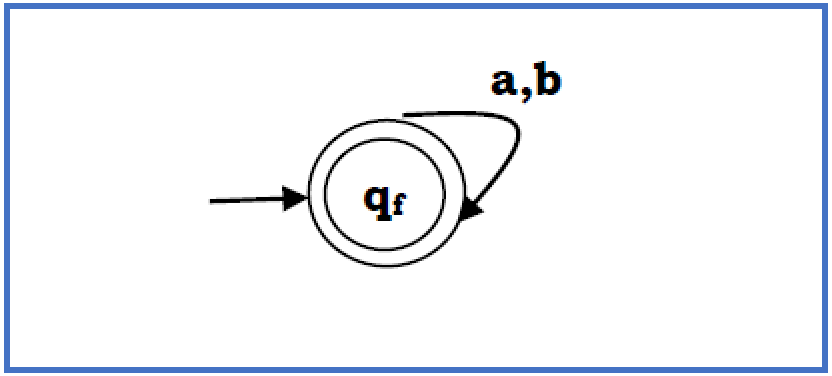
- ***Case 3*: For a regular expression (a+b), we can construct the following FA:**



Finite automata for RE= (a+b)

Con’t

- ***Case 4***: For a regular expression (a+b)*, we can construct the following FA:



Finite automata for RE= (a+b)*

Con't

Method:

- **Step 1** Construct an NFA with Null moves from the given regular expression.
- **Step 2** Remove Null transition from the NFA and convert it into its equivalent DFA.

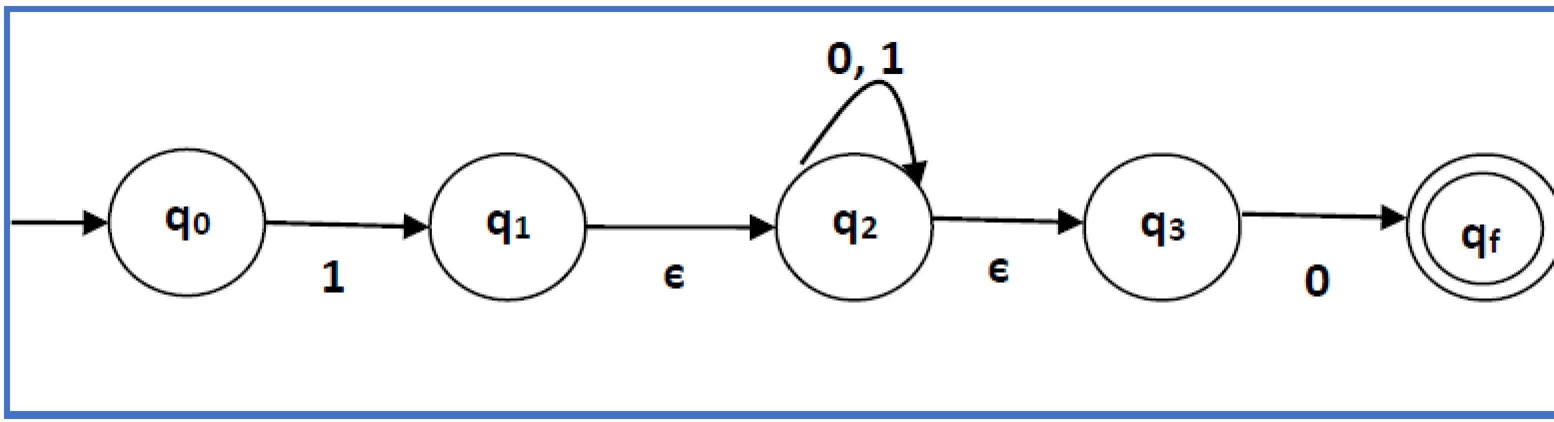
Con’t

Problem

Convert the following RA into its equivalent DFA: 1 (0 + 1)* 0

Solution:

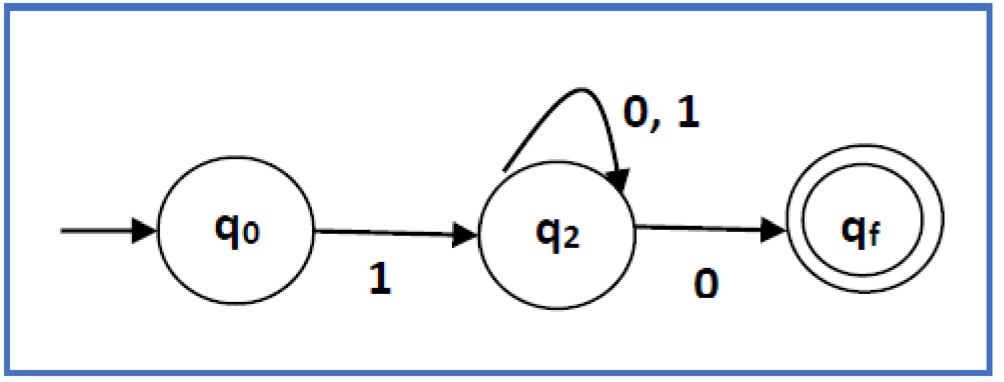
We will concatenate three expressions "1", "(0 + 1)*" and "0"



NDFA with NULL transition for RA: 1 (0 + 1)* 0

Con’t

- Now we will remove the **ϵ** transitions. After we remove the **ϵ** transitions from the NDFA, we get the following:



NDFA without NULL transition for RA: 1 (0 + 1)* 0

- If you want to convert it into a DFA, simply apply the method of converting NDFA to DFA discussed in Chapter 2.

Pumping Lemma for Regular Languages

Theorem

Let **L** be a regular language. Then there exists a constant '**c**' such that for every string **w** in **L:**

$|\mathbf{w}| \geq \mathbf{c}$

We can break **w** into three strings, **w = xyz**, such that:

1. $|y| > 0$
2. $|xy| \leq c$
3. For all $k \geq 0$, the string xy^kz is also in L.

Con’t

Applications of Pumping Lemma

Pumping Lemma is to be applied to show that certain languages are not regular. It should never be used to show a language is regular.

1. If **L** is regular, it satisfies Pumping Lemma.
2. If **L** is non-regular, it does not satisfy Pumping Lemma.

Con’t

Method to prove that a language L is not regular:

1. At first, we have to assume that **L** is regular.
2. So, the pumping lemma should hold for **L**.
3. Use the pumping lemma to obtain a contradiction:

(a) Select **w** such that $|\mathbf{w}| \geq \mathbf{c}$

(b) Select **y** such that $|\mathbf{y}| \geq \mathbf{1}$

(c) Select **x** such that $|\mathbf{xy}| \leq \mathbf{c}$

(d) Assign the remaining string to **z**.

(e) Select **k** such that the resulting string is not in **L**.

Con’t

Hence L is not regular.

Problem

Prove that **L = {a^ib^i | i ≥ 0}** is not regular.

Solution:

1. At first, we sassume that **L** is regular and **n** is the number of states.
2. Let w = *anbn*. Thus |w| = 2n ≥ n.
3. By pumping lemma, let w = xyz, where |xy|≤ n.
4. Let x = *ap*, y = *aq*, and z = *arbn*, where p + q + r = n.p ≠ 0, q ≠ 0, r ≠ 0. Thus |y|≠ 0
5. Let k = 2. Then xy2z = *apa2qarbn*.
6. Number of ***a***s = (p + 2q + r) = (p + q + r) + q = n + q
7. Hence, xy2z = *an+q bn*. Since q ≠ 0, xy2z is not of the form *anbn*.
8. Thus, xy2z is not in L. Hence L is not regular.