Pharos University in Alexandria
Faculty of Computer Science & Artificial Intelligence
Course Title: Theory of Computation
Code: CS 307

Theory of Computation

Lecturer: Sherine Shawky

Text Books

1. Introduction to formal languages and automata, Peter Linz, 6th edition, 2017.

Week 4

Regular Expressions

Regular Expressions (RE)

- They are a way of representing regular languages.
- The algebraic description for regular languages is done using regular expressions.
- They can define the same language that various forms of finite automata can describe.
- Regular expressions offer something that finite automata do not, i.e. it is a declarative way to express the strings that we want to accept.
- They act as input for many systems.
- They are used for string matching in many systems(Java, python, etc.)
- Example: Lexical-analyzer generators, such as Lex or Flex.

Operations Performed on RE

- **1. Union**
- The union of two regular languages, L1 and L2, which are represented using L1 ∪ L2, is also regular and which represents the set of strings that are either in L1 or L2 or both.
  - **Example:**
  - L1 = (1+0).(1+0) = {00 , 10, 11, 01} and L2 = {∈ , 100}
    then L1 ∪ L2 = {∈, 00, 10, 11, 01, 100}.
- **2. Concatenation**
- The concatenation of two regular languages, L1 and L2, which are represented using L1.L2 is also regular and which represents the set of strings that are formed by taking any string in L1 concatenating it with any string in L2.
  - **Example:**
  - L1 = { 0,1 } and L2 = { 00, 11} then L1.L2 = {000, 011, 100, 111}.

Operations Performed on RE

- **3. Kleene closure**
- If L1 is a regular language, then the Kleene closure i.e. L1* of L1 is also regular and represents the set of those strings which are formed by taking a number of strings from L1 and the same string can be repeated any number of times and concatenating those strings.
  - **Example:**
  - L1 = { 0,1} = {∈, 0, 1, 00, 01, 10, 11 …….} , then L* is all strings possible with symbols 0 and 1 including a null string**.**

Algebraic Properties of RE

1. Associativity

- If r1, r2, r3 are RE, then
  r1+ (r2+r3) = (r1+r2) +r3

2. Distributed Property

- **If r1, r2, r3** are regular expressions, then
- (r1+r2).r3 = r1.r3 + r2.r3 i.e. Right distribution
- r1.(r2+ r3) = r1.r2 + r1.r3 i.e. left distribution
- (r1.r2) +r3 $\neq$ (r1+r3)(r2+r3)

3. Idempotent Law

- r1 + r1 = r1 $\Rightarrow$ r1 $\cup$ r1 = r1 , therefore the union operator satisfies idempotent property.
- r.r $\neq$ r $\Rightarrow$ concatenation operator does not satisfy idempotent property.

Algebraic Properties of RE

4. Identity

- In the case of union operators
  if r+ x = r ⇒ x= ∅ as r ∪ ∅= r, therefore ∅ is the identity for +.
  Therefore, ∅ is the identity element for a union operator.
  In the case of concatenation operator –
  if r.x = r , for x= ∈
  r.∈ = r ⇒ ∈ is the identity element for concatenation operator(.) .

5. Commutative Property

- If r1, r2 are RE, then r1+r2 = r2+r1.
- For example, for r1 =a and r2 =b, then RE a+ b and b+ a are equal.
- r1.r2 ≠ r2.r1.
- For example, for r1 = a and r2 = b, then RE a.b is not equal to b.a.

Identities for Regular Expression

- Let p, q and r are regular expressions.
  - $\emptyset + r = r$
  - $\emptyset.r = r.\emptyset = \emptyset$
  - $\in.r = r.\in = r$
  - $\in^* = \in$ and $\emptyset^* = \in$
  - $r + r = r$
  - $r^*.r^* = r^*$
  - $r.r^* = r^*.r = r^+$.
  - $(r^*)^* = r^*$
  - $\in + r.r^* = r^* = \in + r.r^*$
  - $(p.q)^*.p = p.(q.p)^*$
  - $(p + q)^* = (p^*.q^*)^* = (p^* + q^*)^*$
  - $(p+ q).r = p.r + q.r$ and $r.(p+q) = r.p + r.q$

Regular Expressions

- THEOREM 3.1
- Let *r* be a regular expression. Then there exists some nondeterministic finite accepter that accepts *L* (*r*). Consequently, *L* (*r*) is a regular language.

- THEOREM 3.2
  - Let *L* be a regular language. Then there exists a regular expression *r*

such that $L = L(r)$.

Regular Expressions

Regular language	Regular set
∅	{ }
∈	{∈}
a*	{∈, a, aa, aaa …..}
a+ b	{a, b}
a.b	{ab}
a* + ba	{∈, a, aa, aaa,…… , ba}

Arden’s Theorem

- Arden’s theorem state that: “If P and Q are two regular expressions over “∑”, and if P does not contain “∈” , then the following equation in R given by R = Q + RP has a unique solution i.e., R = QP*.”
- That means, whenever we get any equation in the form of R = Q + RP, then we can directly replace it with R = QP*.
- So, here we will first prove that R = QP* is the solution of this equation and then prove that it is the unique solution of this equation.

Arden’s Theorem

- 1. **Proof R = QP* is the solution of R = Q + RP**

  - R = Q + RP .....(i)

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

  - R = Q + QP*P

  Taking Q as common,

  - R = Q(∈ + P*P) = QP*

  (As we know that ∈ + R*R = R*). Hence proved. Thus, R = QP* is the solution of the equation R = Q + RP. Now, we have to prove that this is the only solution to this equation.

Arden's Theorem

- **2. proof R = QP* is the unique solution of R = Q + RP**

Let me take this equation again:

- R = Q + RP

Now, replace R by R = Q + RP,

- R = Q + (Q + RP)P
  = Q + QP + RP2

Again, replace R by R = Q + RP :-

- R = Q + QP + (Q + RP) P2
  = Q + QP + QP2 + RP3
  = Q + QP + QP2 + .. + QP^n + RP(n+1)

Now, replace R by R = QP*, we get,

- R = Q + QP + QP2 + .. + QPn+ QP*P^{(n+1)}

Taking Q as common,

- R = Q(∈ + P + P2 + .. + Pn + P*P(n+1) = QP* [As ∈ + P + P2 + .. + Pn + P*P(n+1) represent the closure of P]

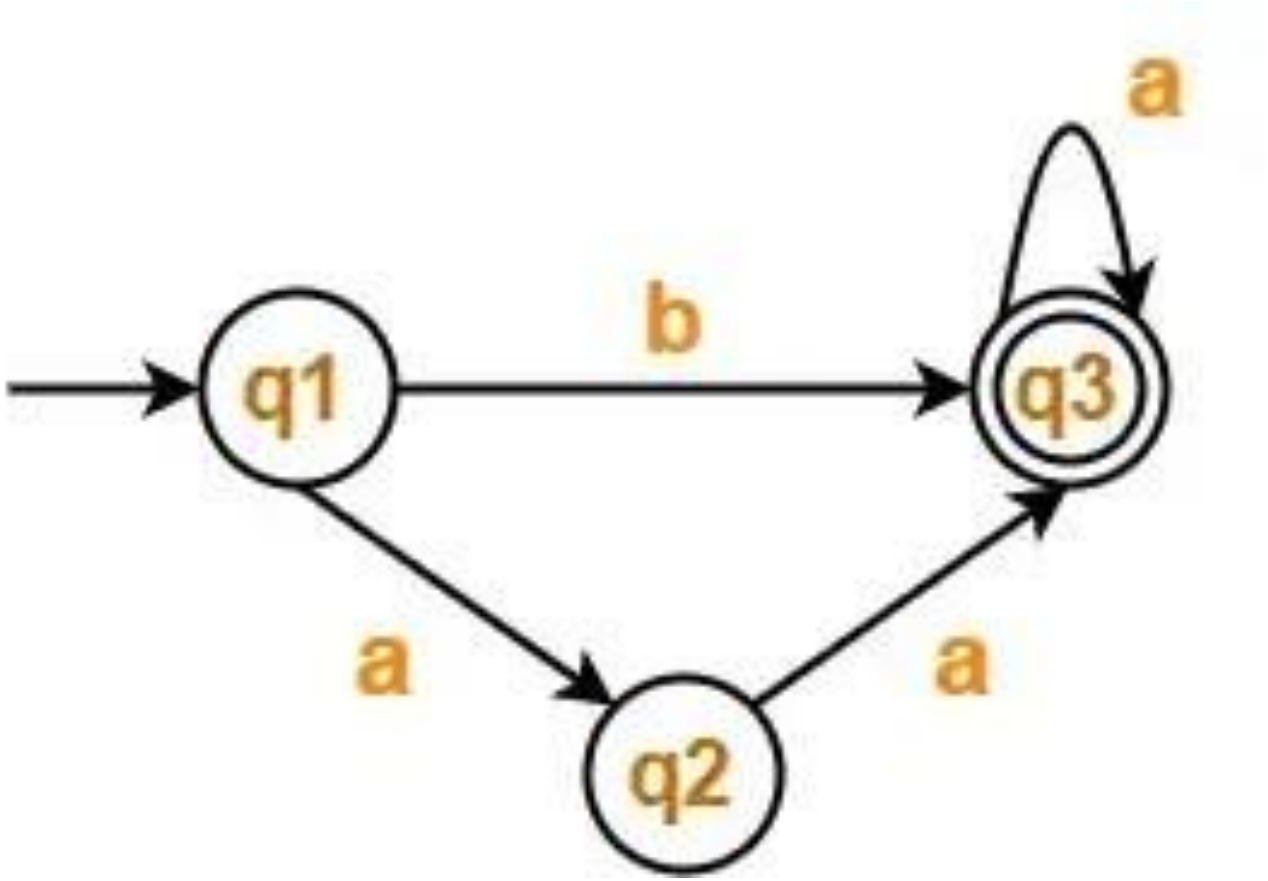
Hence proved. Thus, R = QP* is the unique solution of the equation R = Q + RP.

Create a RE

- Write the regular expression for the language having a string which should have at least one 0 and at least one 1.
- **Solution:**
- The regular expression will be:
- R = [(0 + 1)* 0 (0 + 1)* 1 (0 + 1)*] + [(0 + 1)* 1 (0 + 1)* 0 (0 + 1)*]

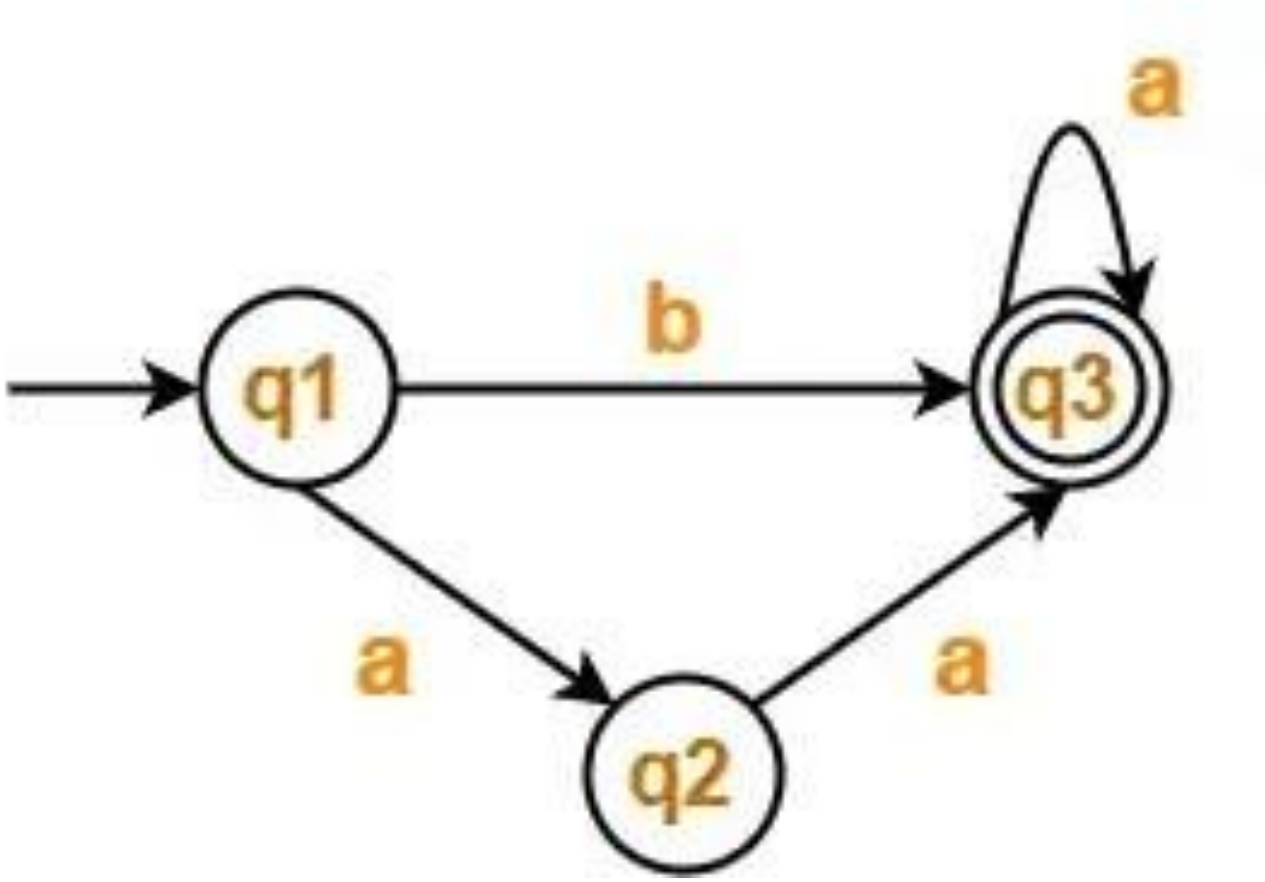
Convert FA to RE

- Problem 1



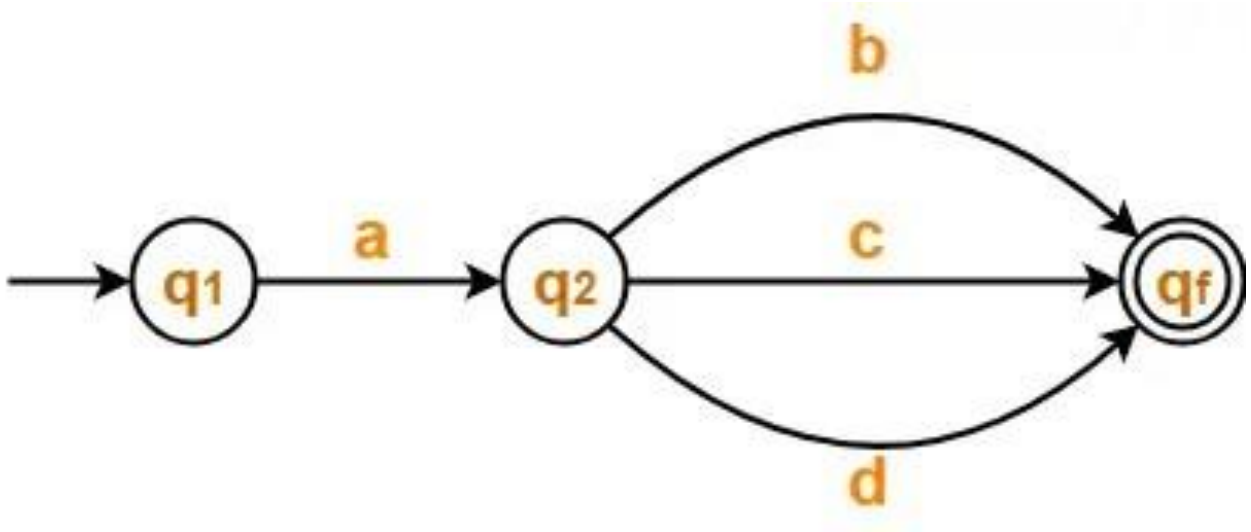
Convert FA to RE

- Answer
- (b + aa)a*



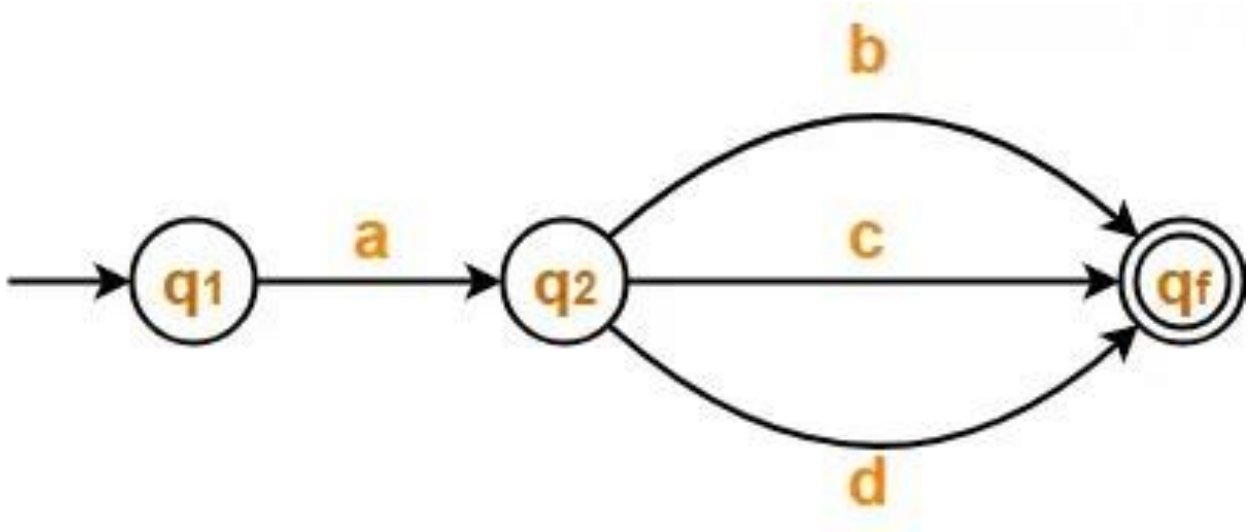
Convert FA to RE

- Problem 2



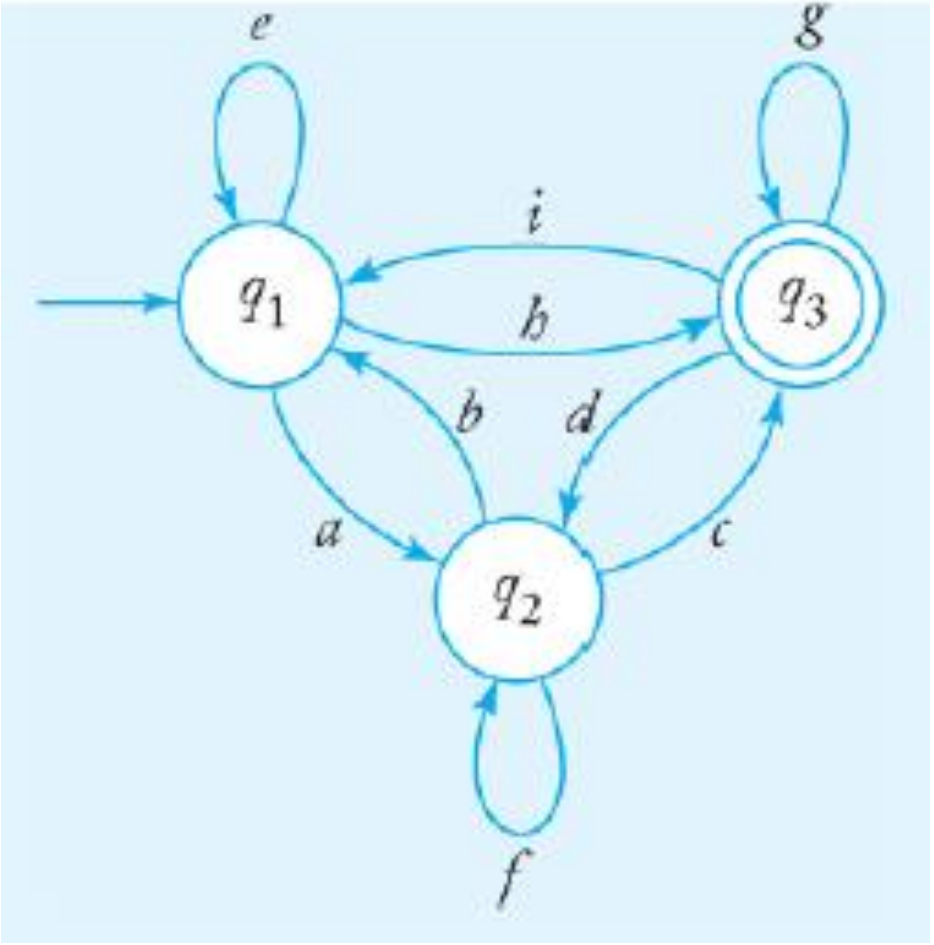
Convert FA to RE

- Answer
- a(b+c+d)



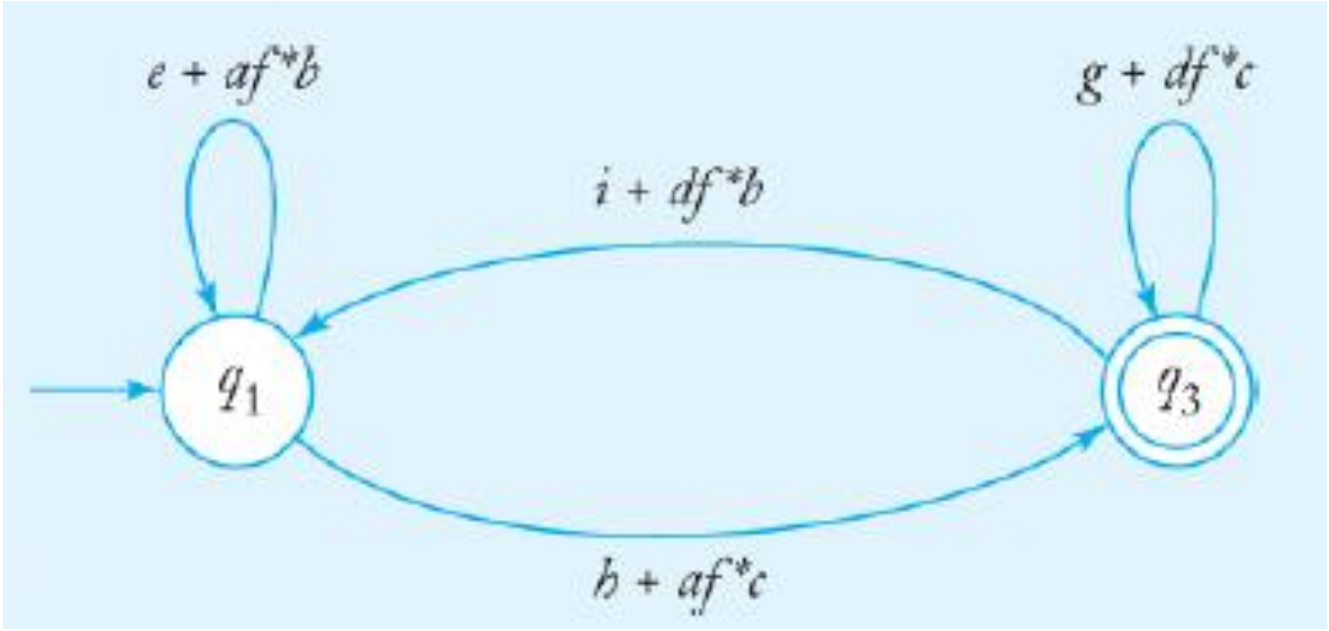
Convert FA to RE using GTG

- Example:
- To remove $q2$, we first introduce some new edges.
- We create an edge
  - from $q1$ to $q1$ and label it $e + af^*b$,
  - from $q1$ to $q3$ and label it $h + af^*c$,
  - from $q3$ to $q1$ and label it $i + df^*b$,
  - from $q3$ to $q3$ and label it $g + df^*c$.



Convert FA to RE using GTG

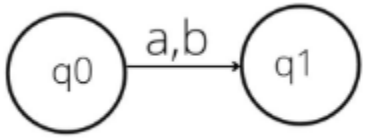
- When this is done, we remove $q2$ and all associated edges.
- This gives the output GTG.



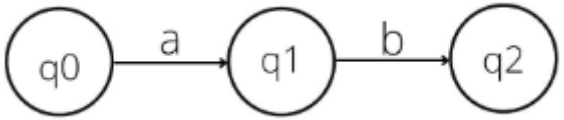
Convert RE to FA

- Some standard rules help in the conversion of RE to NFA are:-

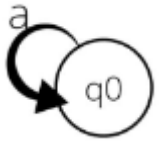
1.) If RE is in the form **a+b,** it can be represented as:



2.) If RE is in the form **ab,** it can be represented as:



3.) If RE is in the form of **a*,** it can be represented as:



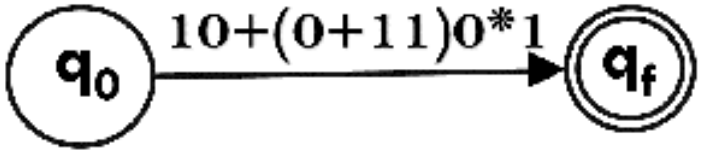
Convert RE to FA

- Design a FA from given regular expression 10 + (0 + 11)0* 1.

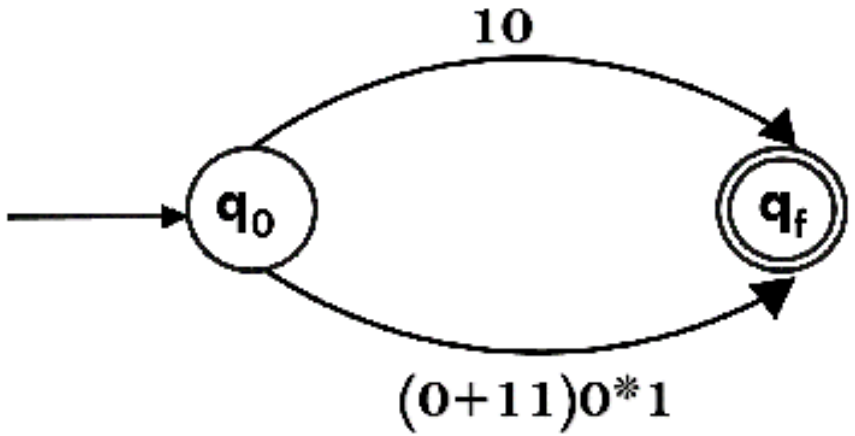
Convert RE to FA

- Design a FA from given regular expression 10 + (0 + 11)0* 1.
- Answer

Step 1:

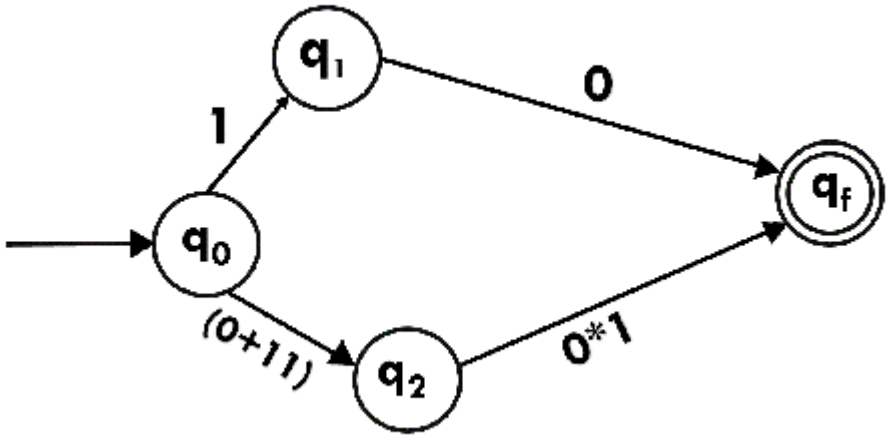


Step 2:

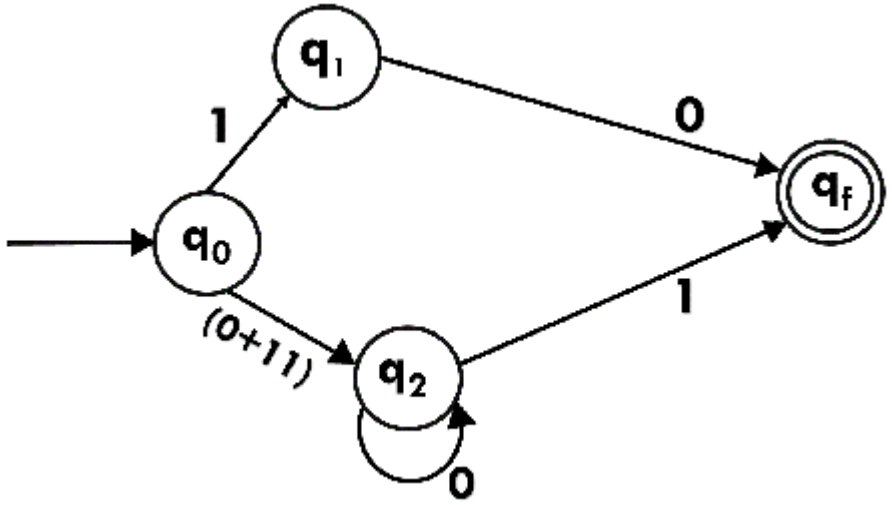


Convert RE to FA

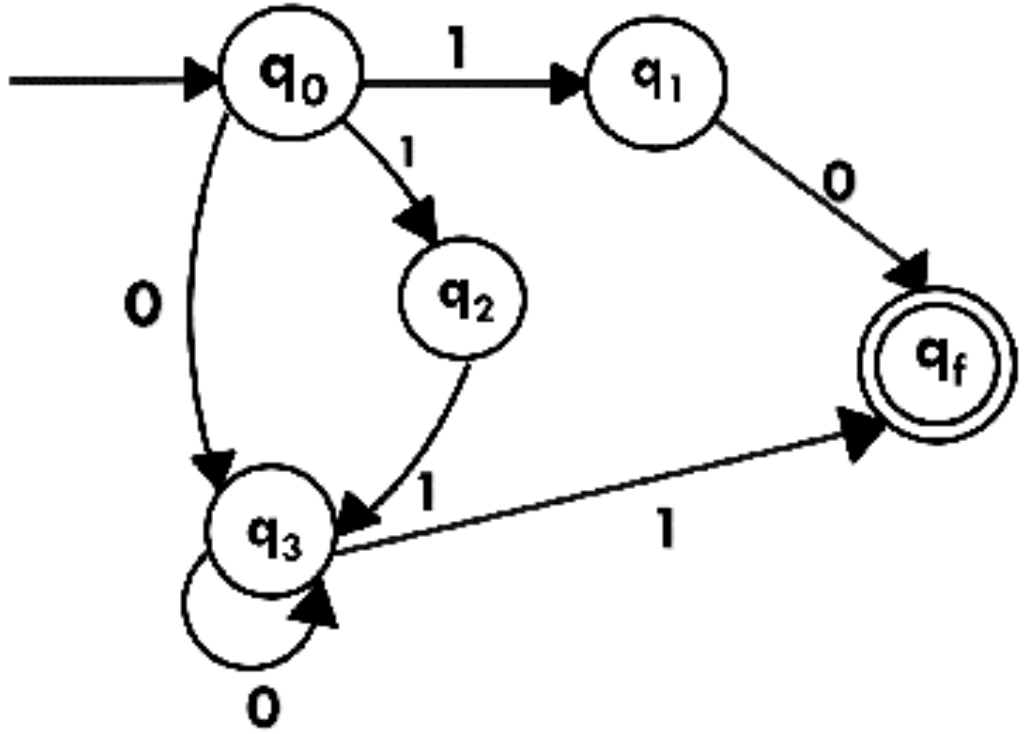
Step 3:



Step 4:



Step 5:



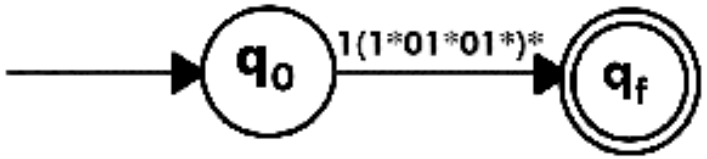
Convert RE to FA

- Design a NFA from given regular expression 1 (1* 01* 01*)*.

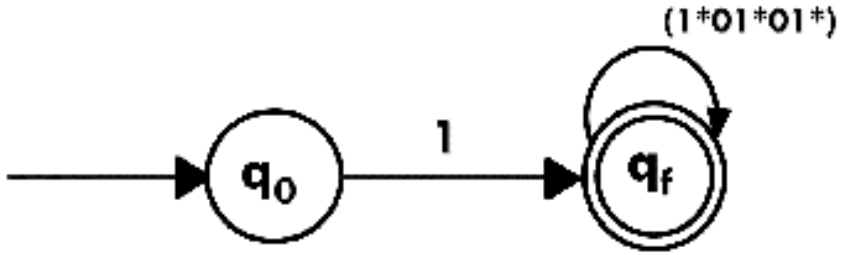
Convert RE to FA

- Design a NFA from given regular expression 1 (1* 01* 01*)*.
- Answer

Step 1:



Step 2:



Step 3: