Unit VI: Language and grammar

REGULAR EXPRESSION

Introduction

- The language accepted by finite automata can be easily described by simple expressions called Regular Expressions.
- The languages accepted by some regular expression are referred to as Regular languages.
- A regular expression can also be described as a sequence of pattern that defines a string
  - In a regular expression, x^* means zero or more occurrence of x. It can generate {e, x, xx, xxx, xxxx, ...}
  - In a regular expression, x^+ means one or more occurrence of x. It can generate {x, xx, xxx, xxxx, ...}

Primitive regular expression

- The regular expression $\emptyset$ describes the empty language $\emptyset$.
  - R= $\emptyset$; L(R)={}
- The regular expression ε describes the language containing just the empty string $\{\varepsilon\}$.
  - R= ε ; L(R)=$\{\varepsilon\}$
- For each $a \in \Sigma$ the regular expression a describes the language {a}
  - R=a ; L(R)={a}

- The above expression is called primitive regular expression. It is minimum language generated by regular expression.

Regular operators

- Union (+) of two regular expression (RE) is also regular
  - If L1 and L2 are languages, x ∈ L1 ∪ L2 if and only if x ∈ L1 or x ∈ L2
- Concatenation (.) of two regular expression (RE) is also regular
  - The concatenation of strings x and y is obtained by writing down x followed by y right after it. To get a concatenation of two languages L1 and L2, we consider all pairs of strings, one from each L1 and L2, and concatenate them
  - Let L1 and L2 be languages. The concatenation of L1 and L2 is the set L1L2 = {xy | x ∈ L1 ∧ y ∈ L2}.
- Kleene star of regular expression (RE) is also regular
  - We define $L* = \bigcup_{k=0}^{\infty} L^k$ where k is L concatenated with itself k times, i.e., $L^1 = L$, $L^2 = LL$, and so on. We can also define L^k inductively as $L^k = LL^{k-1}$ with the base case $L^0 = \{\in\}$.
- Positive closure of regular expression (RE) is also regular
  - We define $L* = \bigcup_{k=0}^{\infty} L^k$ where k is L concatenated with itself k times, i.e., $L^1 = L$, $L^2 = LL$, and so on. We can also define L^k inductively as $L^k = LL^{k-1}$ with the base case $L^0 = \{\in\}$.

Regular operators

- Union (+) of two regular expression (RE) is also regular
  - R_1=a R_2=b, $R_1 U R_2$=a+b generates language that contain either a or b.
- Concatenation (.) of two regular expression (RE) is also regular
  - R_1=a R_2=b, $R_1.R_2$=a.b generates language that contain either a and b.
- Kleene star of regular expression (RE) is also regular
  - R_1=a. R_1^*=a*
- Positive closure of regular expression (RE) is also regular
  - R_1=a. $R_1^+=a^+$

Examples

Find regular expression for following languages

- Language containing no string
  - R=Ø
- Language containing string of length 0
  - R=ε
- Language containing string of length 1
  - L=(a,b)   R= a+b
- Language containing string of length 2
  - L=(aa,ab,bb,ba)
  - R= aa+ab+bb+ba=(a+b)(a+b)

Examples

- Write the regular expression for the language accepting all combinations of a's, over the set ∑ = {a}
  - set of {ε, a, aa, aaa, ...}.
  - Solution: R = a* (Kleen closure)
- Write the regular expression for the language accepting all combinations of a's except the null string, over the set ∑ = {a}
  - L = {a, aa, aaa, ...}
  - R = a+
- Write the regular expression for the language accepting all the string containing any number of a's and b’s.
  - L = {ε, a, aa, b, bb, ab, ba, aba, bab, ...}
  - R= (a + b)*

Examples

- Write the regular expression for the language accepting all the string which are starting with 1 and ending with 0, over $\sum = \{0, 1\}$.
  - R = 1 (0+1)* 0
- Write the regular expression for the language starting and ending with a and having any having any combination of b's in between.
  - R = a b* a
- Write the regular expression for the language starting and ending with same symbol.
  - R = a(a+b)*a+b(a+b)*b+a+b
- Write the regular expression for the language starting and ending with different symbol.
  - R = a(a+b)*b+b(a+b)*a
- Language containing all string second symbol is a and fourth symbol is b over (a,b)
  - R=(a+b)a(a+b)b(a+b)*

Examples

(a+b)*	Set of strings of a’s and b’s of any length including the null string. So L = { ε, a, b, aa , ab , bb , ba, aaa…….}
(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, …………..}

Examples

- Which one of the following languages over the alphabet {0,1} is described by the regular expression?
  (0+1)*0(0+1)*0(0+1)*
  (A) The set of all strings containing the substring 00.
  (B) The set of all strings containing at most two 0's.
  (C) The set of all strings containing at least two 0's.
  (D) The set of all strings that begin and end with either 0 or 1
- Option C

Examples

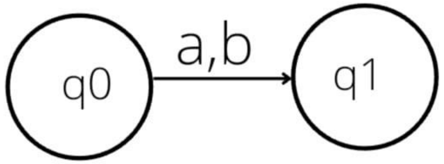
- **Question 2 :** Which of the following languages is generated by given grammar?
  S -> aS | bS | ∈
  (A) $\{a^n b^m \mid n,m \geq 0\}$
  (B) $\{w \in \{a,b\}^* \mid$ w has equal number of a’s and b’s$\}$
  (C) $\{a^n \mid n \geq 0\} \cup \{b^n \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$
  (D) $\{a,b\}^*$
- **Solution :** Option (A) says that it will have 0 or more a followed by 0 or more b.
  But S -> bS => baS => ba is also a part of language. So (A) is not correct.
  Option (B) says that it will have equal no. of a’s and b’s. But But S -> bS => b is also a part of language. So (B) is not correct.
  Option (C) says either it will have 0 or more a’s or 0 or more b’s or a’s followed by b’s. But as shown in option (A), ba is also part of language. So (C) is not correct.
  Option (D) says it can have any number of a’s and any numbers of b’s in any order. So (D) is correct.

Conversion of Regular Expression to Automata

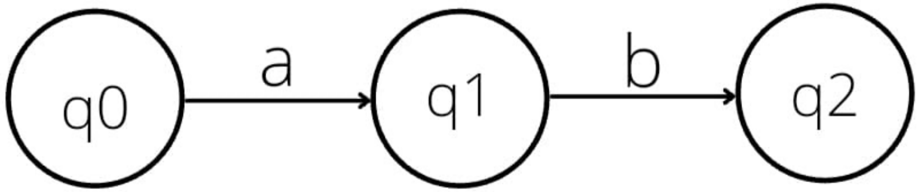
- **Step 1:** Make a transition diagram for a given regular expression, using NFA with ε moves.
- **Step 2:** Then, Convert this NFA with ε to NFA without ε.
- **Step 3:** Finally, Convert the obtained NFA to equivalent DFA.

Conversion of Regular Expression to Automata

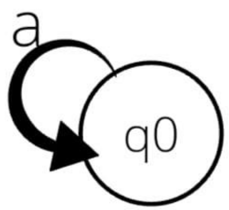
- 1.) If RE is in the form **a+b,** it can be represented as:



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

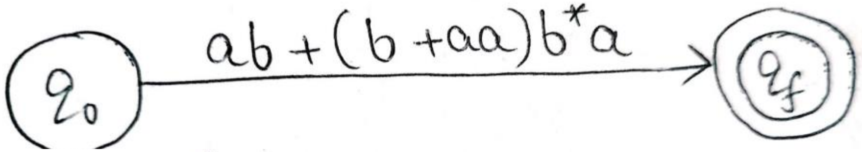


- 3.) If RE is in the form of **a*,** it can be represented as:

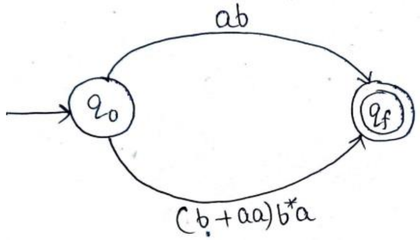


Conversion of Regular Expression to Automata

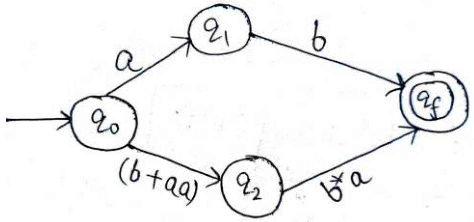
- Design a Finite Automata from the given RE **[ab + (b + aa)b* a]**.
- **Step 1:**



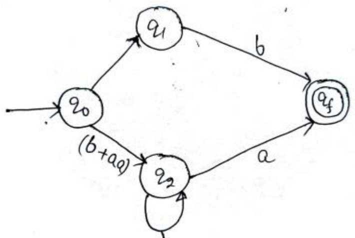
- **Step 2:**



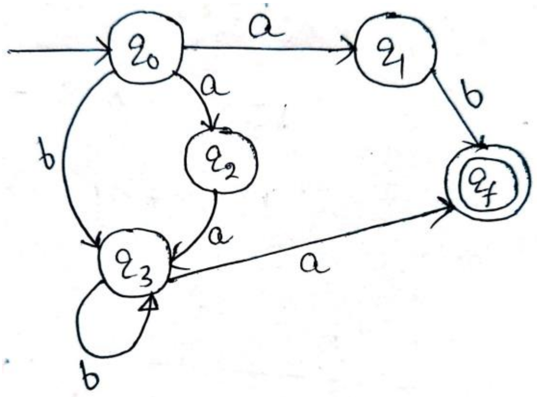
- **Step 3:**



- **Step 4:**



Step 5:



Conversion of Regular Expression to Automata

- Transition Table for NFA

State	a	b
→q0	{q1,q2}	q3
q1	ϕ	qf
q2	q3	ϕ
q3	qf	q3
*qf	ϕ	ϕ

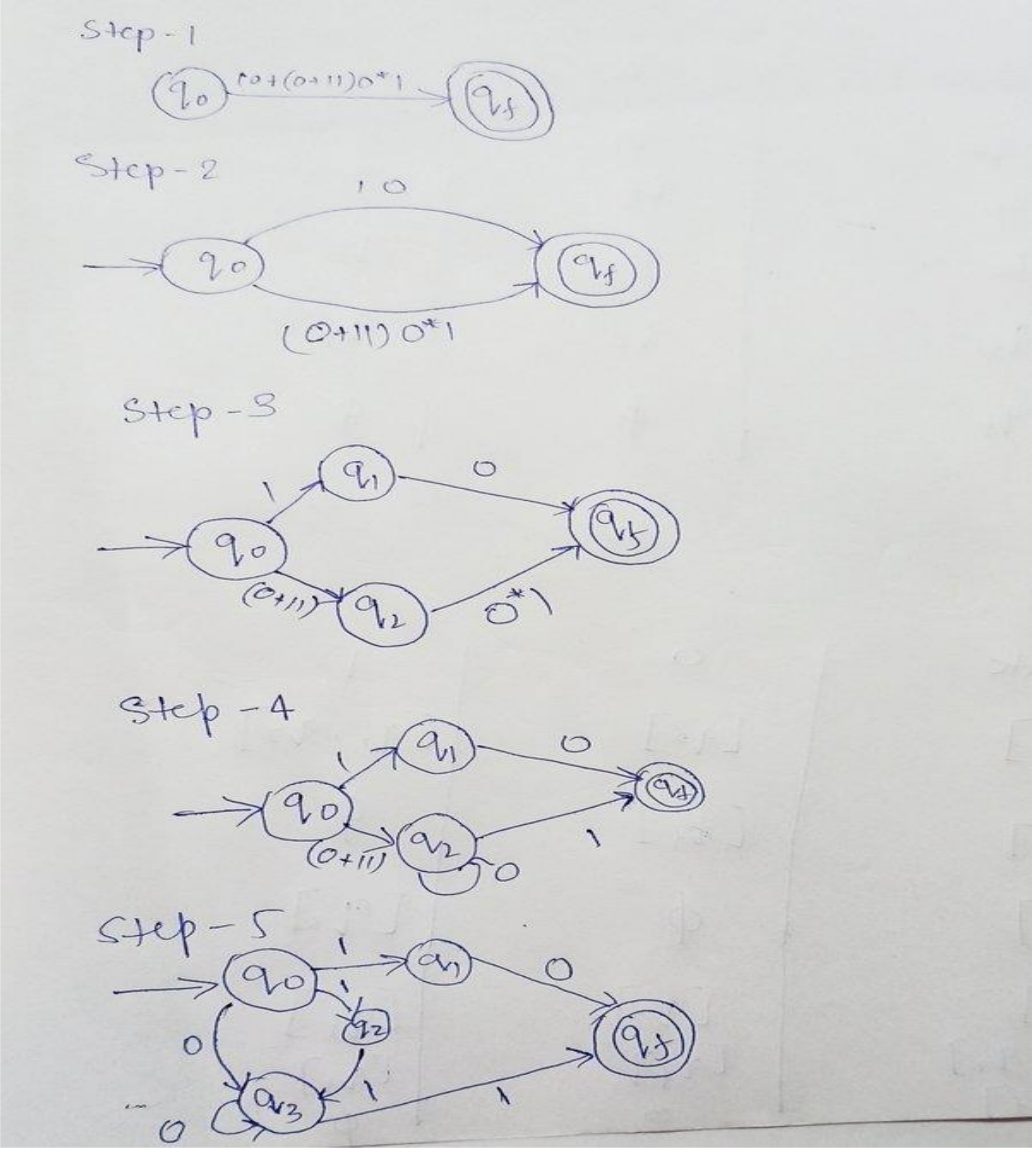
Conversion of Regular Expression to Automata

- Transition Table for DFA

State	a	b
→q0	[q1,q2]	q3
q1	ϕ	qf
q2	q3	ϕ
q3	qf	q3
[q1,q2]	qf	qf
*qf	ϕ	ϕ

Conversion of Regular Expression to Automata

- Convert RE to FA
- (10+(0+11)0*1



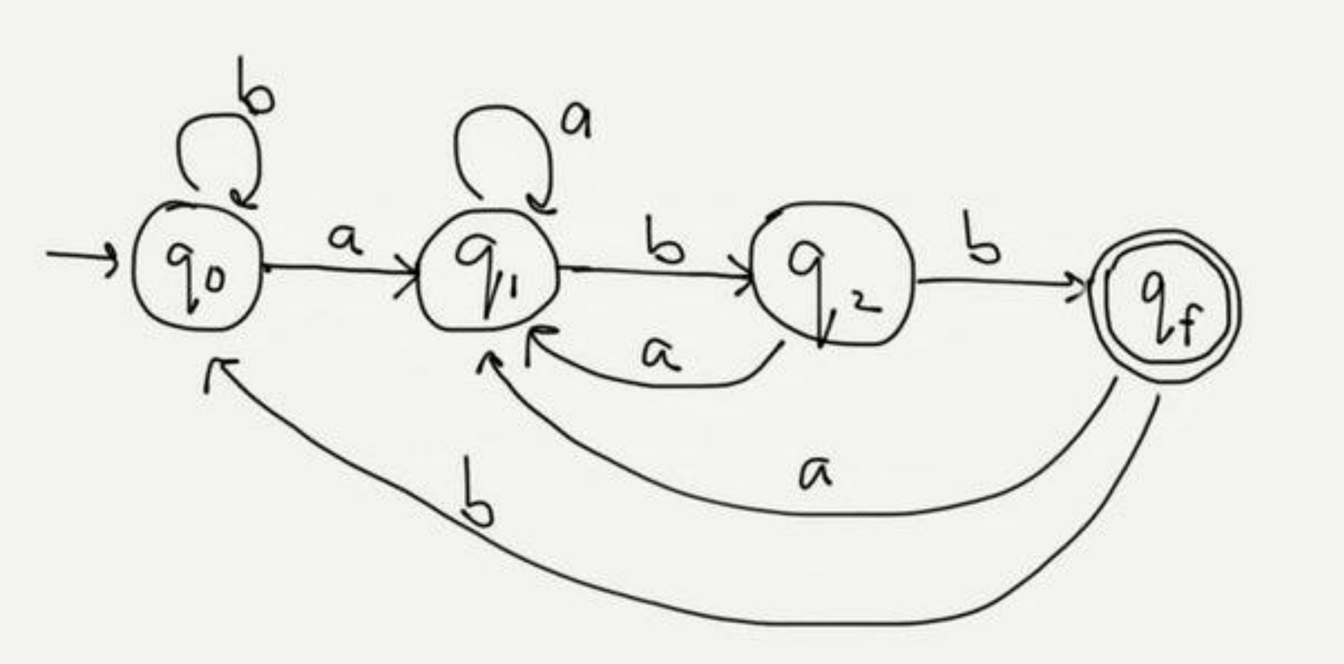
NFA and DFA

State	0	1
$\rightarrow q_0$	q_3	$\{q_1, q_2\}$
q_1	q_f	ϕ
q_2	ϕ	q_3
q_3	q_3	q_f
$* q_f$	ϕ	ϕ

State	0	1
$\rightarrow [q_0]$	$[q_3]$	$[q_1, q_2]$
$[q_1]$	$[q_f]$	ϕ
$[q_2]$	ϕ	$[q_3]$
$[q_3]$	$[q_3]$	$[q_f]$
$[q_1, q_2]$	$[q_f]$	$[q_f]$
$* [q_f]$	ϕ	ϕ

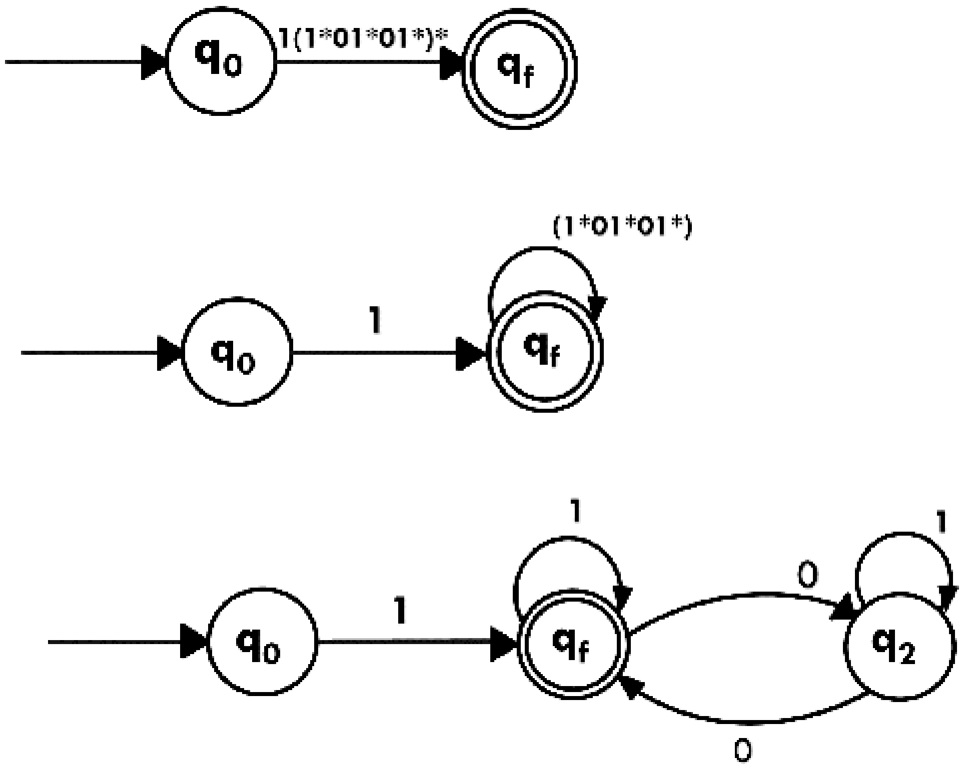
DFA

- DFA for regular expression (a+b)*abb?



Conversion of Regular Expression to Automata

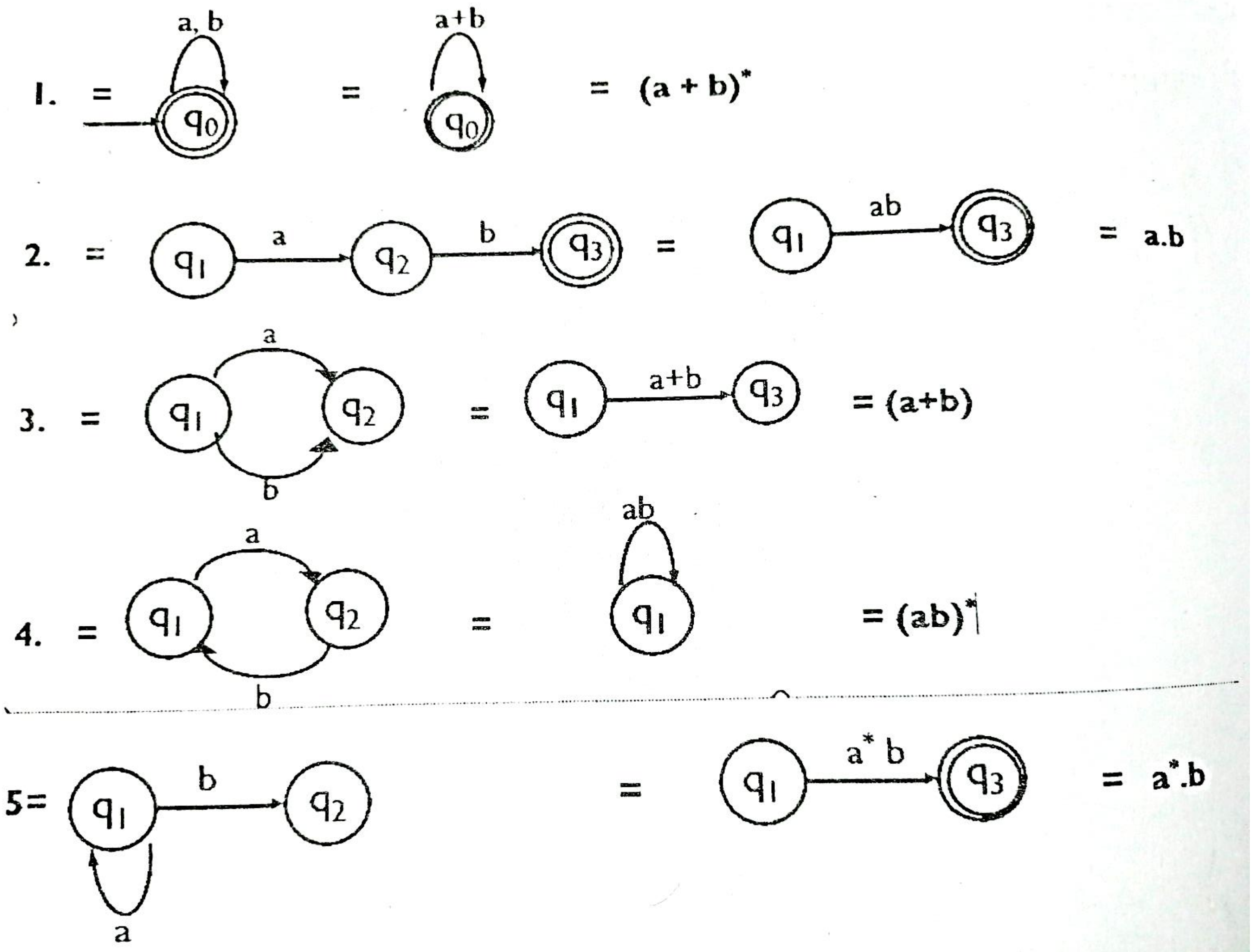
- Design a NFA from given regular expression 1 (1* 01* 01*)*.



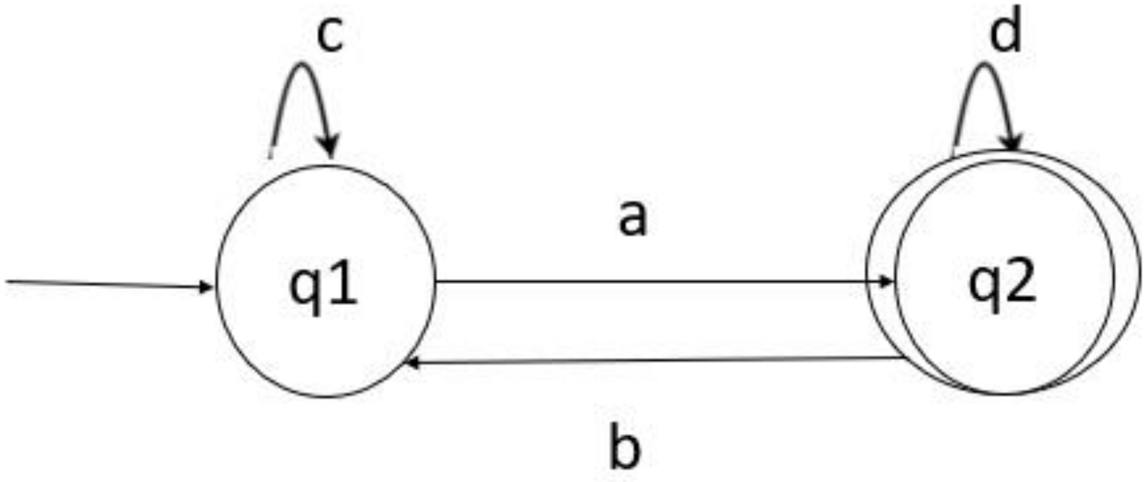
Conversion for Automata to Regular Expression (State elimination method)

- Rule 1
- The initial state of DFA must not have any incoming edge.
- If there is any incoming edge to the initial edge, then create a new initial state having no incoming edge to it.
- Rule 2
- There must exist only one final state in DFA.
- If there exist multiple final states, then convert all the final states into non-final states and create a new single final state.
- Rule 3
- The final state of DFA must not have any outgoing edge.
- If this exists, then create a new final state having no outgoing edge from it.
- Rule 4
- Eliminate all intermediate states one by one.

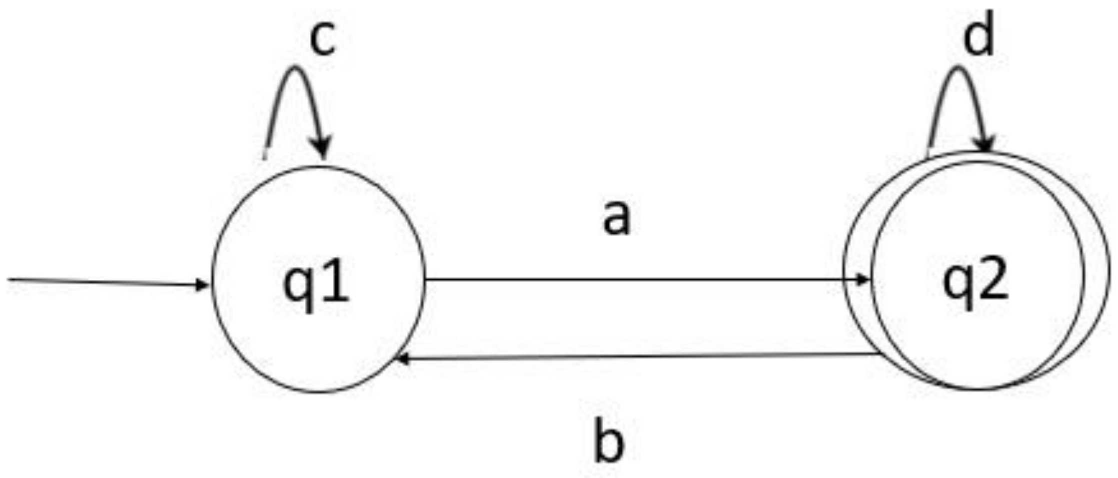
Some Conversions



Example Convert to regular expression

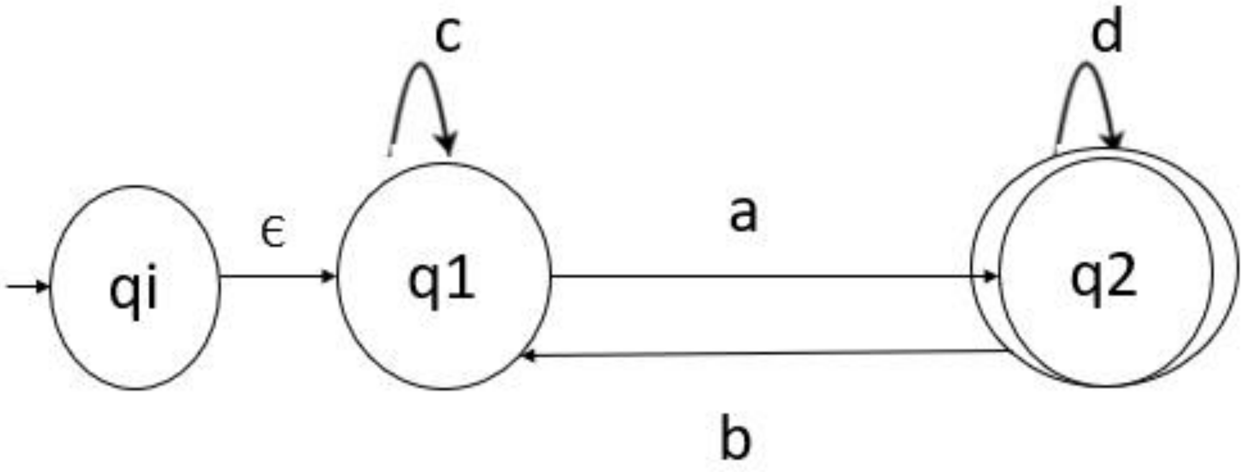


Example



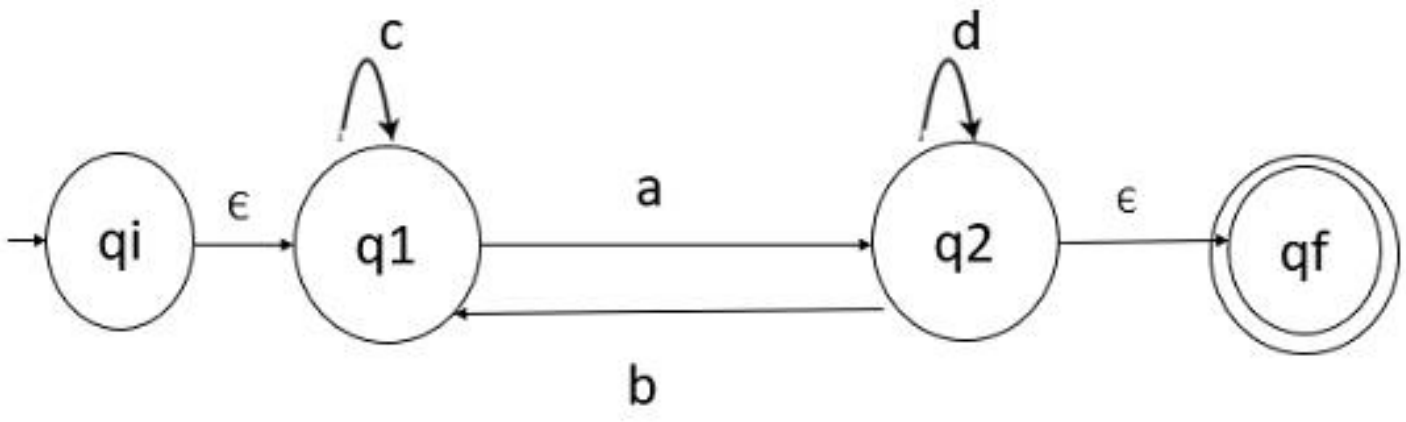
Example

- Step 1: Initial state q1 has an incoming edge so create a new initial state qi.



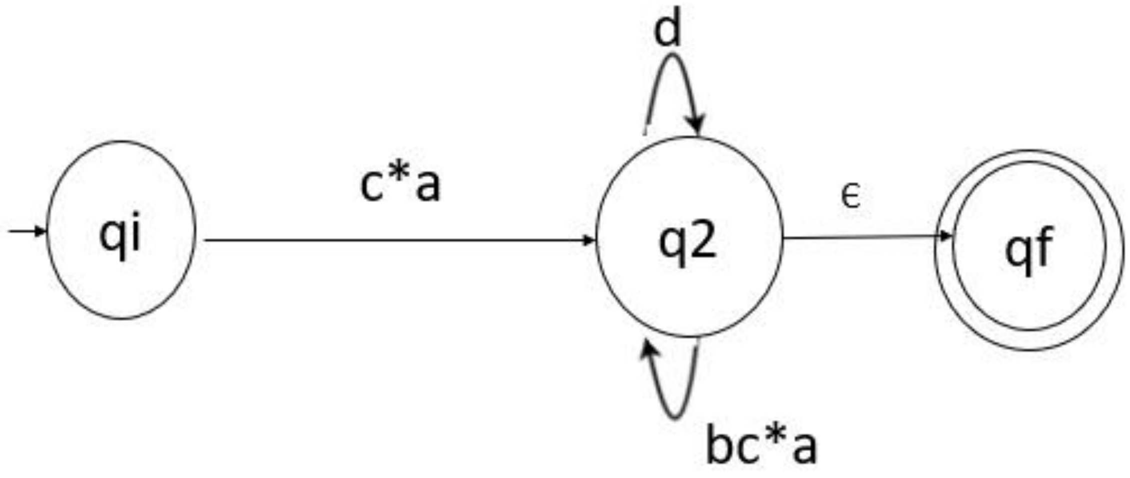
Example

- Step 2: Final state q2 has an outgoing edge. So, create a new final state qf.



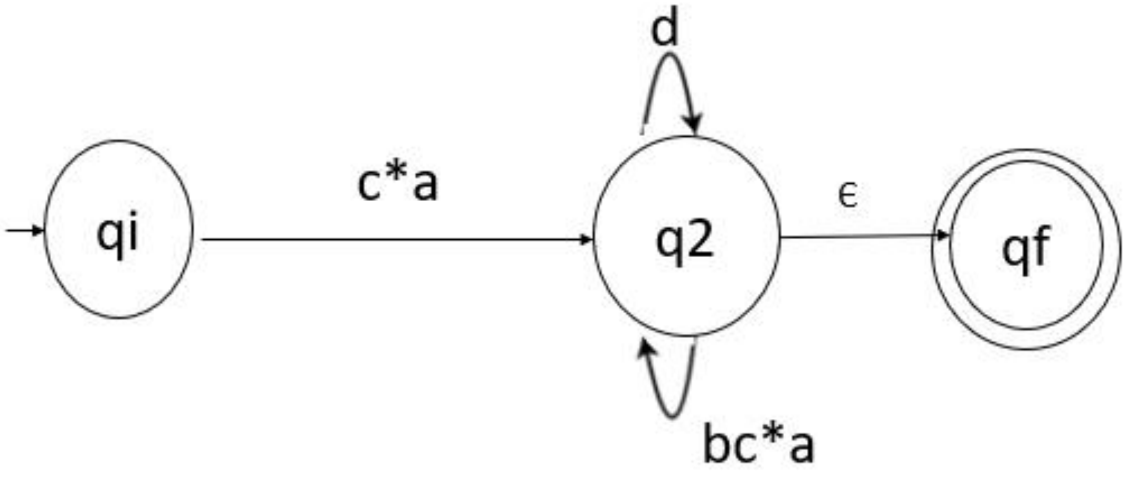
Example

- Step 3: Start eliminating intermediate states
- First eliminate q1
- There is a path going from qi to q2 via q1. So, after eliminating q1 we can connect a direct path from qi to q2 having cost.
- εc*a=c*a
- There is a loop on q2 using state qi. So, after eliminating q1 we put a direct loop to q2 having c
- b.c*.a=bc*a

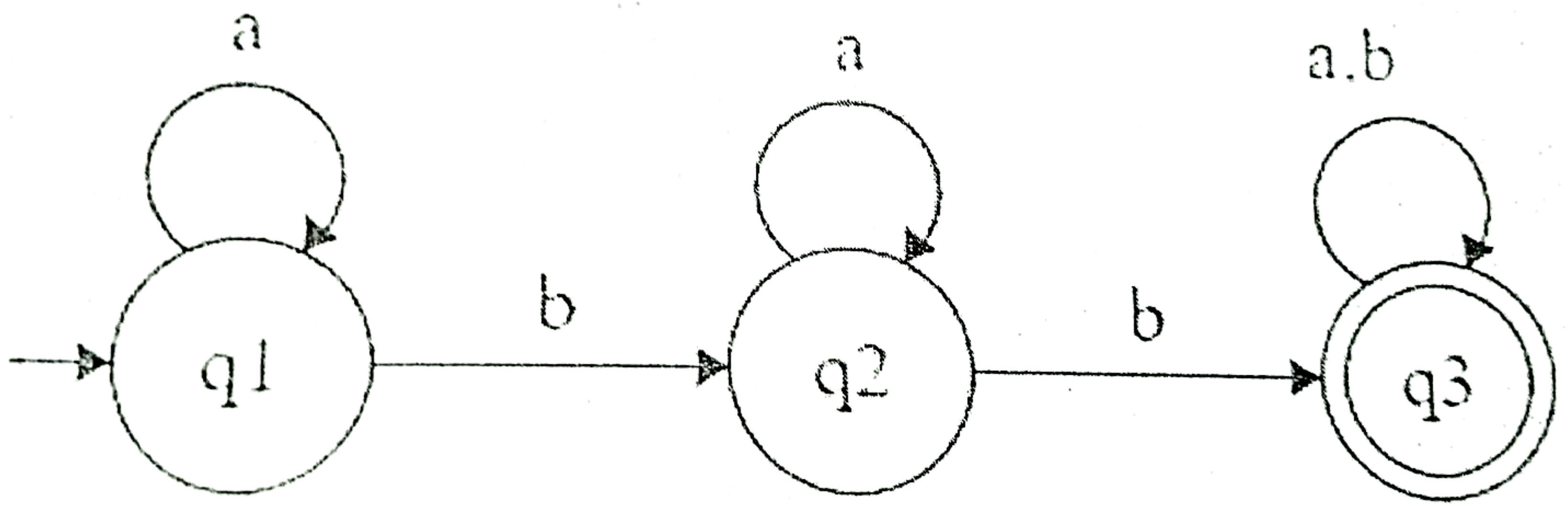


Example

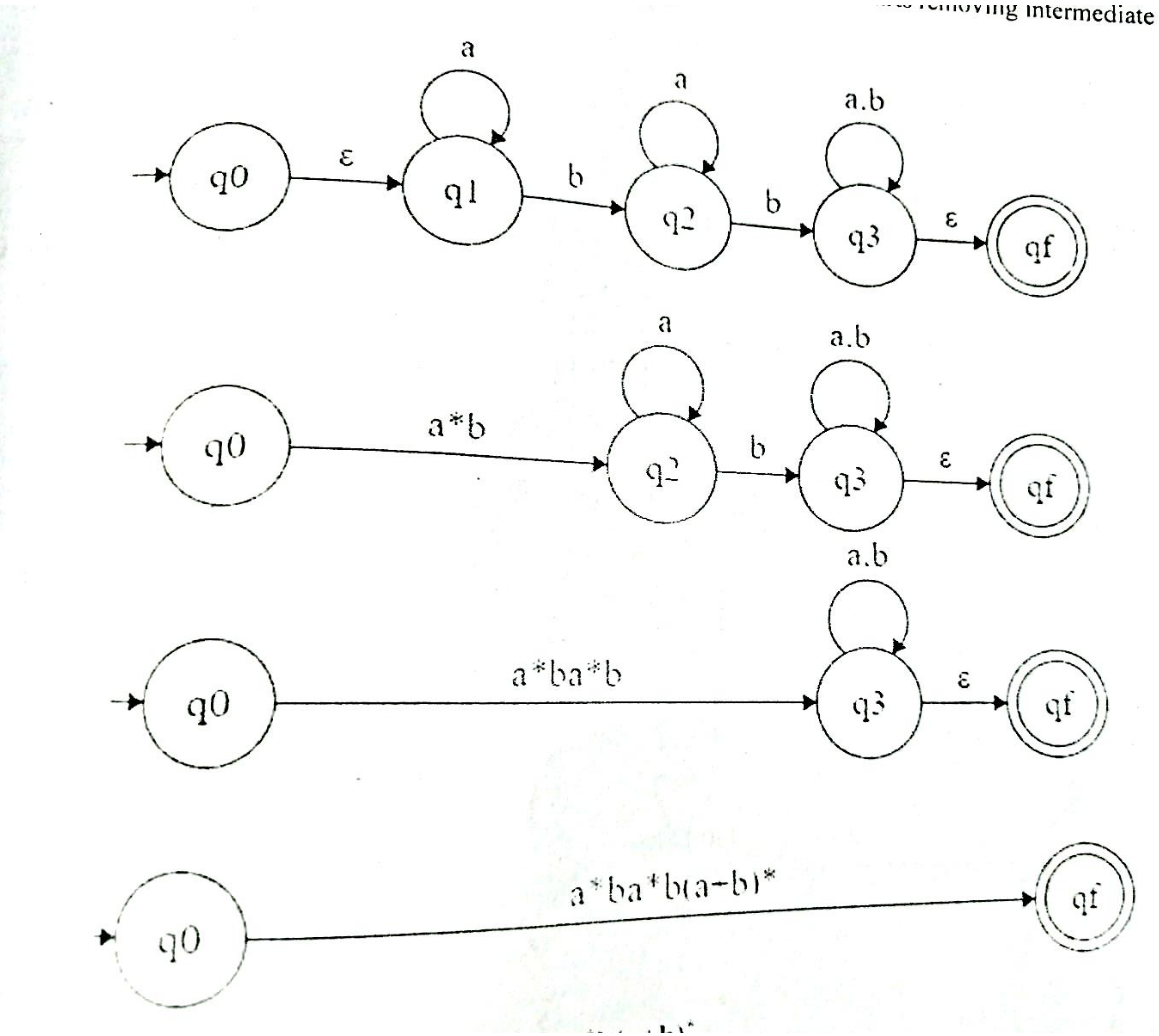
- Start eliminating intermediate states
- Second eliminate q2
- There is a direct path from qi to qf so, we can directly eliminate q2 having cost –
- C*a(d+bc*a)* ε = c*a(d+bc*a)*



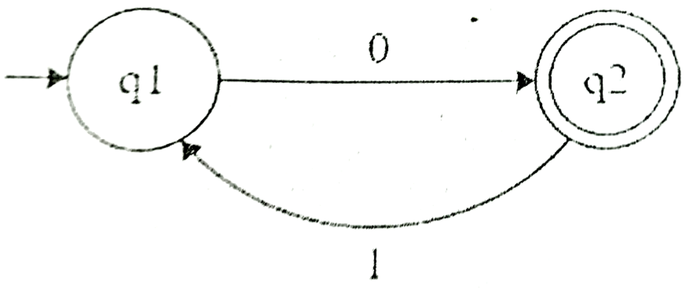
Convert FA to RE

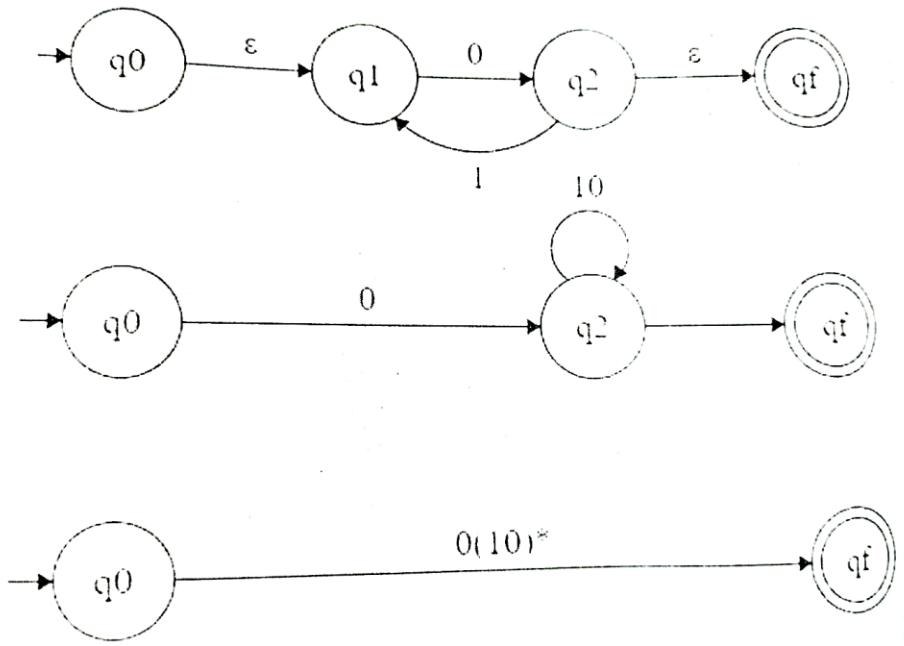


Convert FA to RE



Convert FA to RE





Generating Grammar for the language

- Steps
  - First write regular expression for language
  - Then write grammar

Examples

Write grammar that generates string over $\sum(a, b)$

- String of exactly length two
  - R.E: (a+b)(a+b)
  - S→AA
  - A→a/b
- Strings of at most length two
  - R.E: (a+b+∈)(a+b+ ∈)
  - S→AA
  - A→a/b/ ∈
- Strings of at most length two
  - R.E: a(a+b)*
  - S→aA
  - A→aA/bA/ ∈
- Ends with ba
  - R.E: (a+b)*ba
  - S→Aba
  - A→aA/bA/∈
- Strings starts with a and ends with b
  - R.E: a(a+b)*b
  - S→aAb
  - A→aA/bA/ ∈
- Starts and ends with same symbol
  - R.E: a(a+b)*a+b(a+b)*b+a+b
  - S→aAa/bAb/a/b
  - A→aA/bA/ ∈