Theory of Computation

BY

MR. VIPIN WANI

UNIT 2

REGULAR EXPRESSIONS (RE)

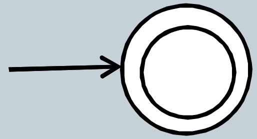
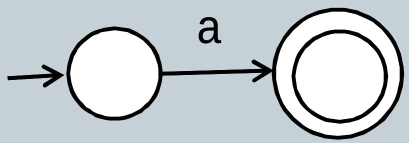
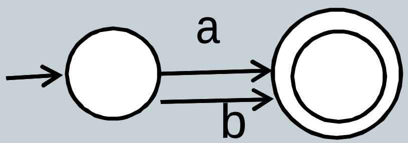
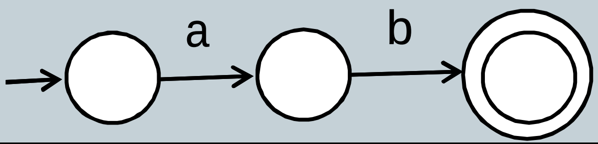
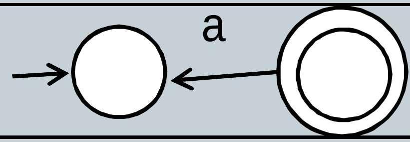
Regular Expressions Introduction

- Regular Language: The set of strings accepted by finite automata is known as regular language.
- Such language can also be represented in compact form using set of operators.
- These operators are
  1. + union operator
  2. . Concatenation operators
  3. * Star or closure operator

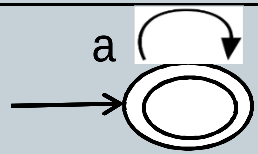
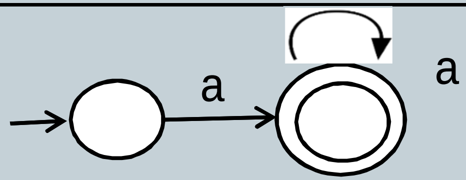
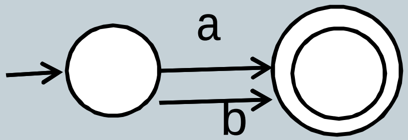
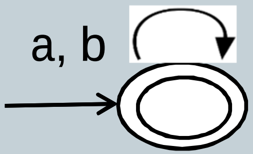
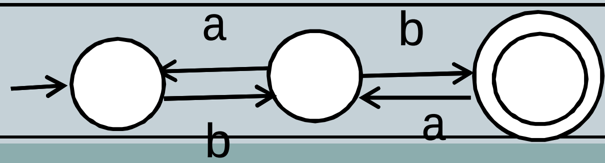
Regular Expressions Introduction

- Regular Expressions: An expressions written using set of operators such as (+, . , *) and describing regular language is known as regular expressions.
- They are used to represent the regular language in compact form.
- Let us see some of the examples of Regular Expressions.

Regular Expressions Introduction

Sr. No.	Automata	Language	Regular Expression
1		{ε}	RE=ε
2		{a}	RE=a
3		{a ,b}	RE=a+b
4		{a ,b}	RE=a.b
5		φ	RE=φ

Regular Expressions Introduction

Sr. No.	Automata	Language	Regular Expression
6		{ε, a, aa, aaa,...}	RE=a*
7		{ a, aa, aaa,...}	RE=aa* or a^+
8		{a ,b}	RE=a+b
9		{ε, a, b, aa, ab, ba, aaa,...}	RE=(a+b)*
10		{ab,ba}	RE=ab+ba

Regular Expressions Introduction

- A Regular Expression can be recursively defined as follows –
- ε is a Regular Expression indicates the language containing an empty string. (L (ε) = {ε})
- φ is a Regular Expression denoting an empty language. (L (φ) = { })
- x is a Regular Expression where L = {x}
- If X is a Regular Expression denoting the language L(X) and Y is a Regular Expression denoting the language L(Y), then
  - X + Y is a Regular Expression corresponding to the language L(X) ∪ L(Y) where L(X+Y) = L(X) ∪ L(Y).
  - X . Y is a Regular Expression corresponding to the language L(X) . L(Y) where L(X.Y) = L(X) . L(Y)
  - X* is also a Regular Expression.
  - Y* is also a Regular Expression.

Regular Expressions Introduction

- X^+ is also a Regular Expression.
- Y^+ is also a Regular Expression.
- R^* is a Regular Expression corresponding to the language $L(R^*)$ where $L(R^*) = (L(R))^*$

Regular Expressions Examples

Sr. No.	Regular Language	Regular Expression
1	The Set {1010}	RE=1010
2	The Set {10, 1010}	RE=10+1010
3	The Set {ε, 10, 01}	RE=ε+10+01
4	The Set {ε, 0, 00, 000...}	0^*
5	The Set {0, 00, 000...}	0^+
6	The set of Strings over alphabet {0, 1} starting with 0.	$0(0+1)^*$
7	The set of Strings over alphabet {0, 1} ending with 1.	$(0+1)^*1$
8	The set of Strings over alphabet {a, b} starting with a and ending with b.	$a(a+b)^*b$
9	The set of string Recognized by $(a+b)^3$	(a+b) (a+b) (a+b)

Precedence of Operators

- Similar to arithmetic operators, operators of regular expressions follow a predefined precedence.
- Precedence for operators of regular expressions are as follow.
- Star (*) or closure operator has the highest precedence.
- Dot (.) or concatenation operator has the next highest precedence.
- Union (+) or operator has the lowest precedence.

FA representing RE

- Finite Automata and Regular Expressions are equivalent.
- To show this:
  - we can express a DFA as an equivalent RE
  - we can express a RE as an ε-NFA using operators on alphabets, ε , φ, ∑.
  - The ε-NFA can be converted to a DFA and the DFA to an NFA,
  - then RE will be equivalent to all the automata we have described.

FA representing RE

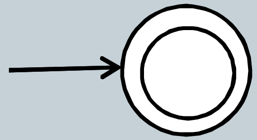
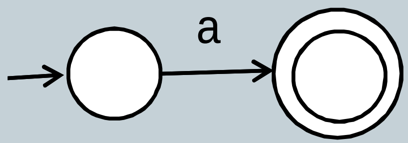
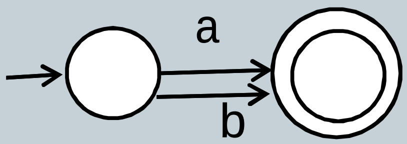
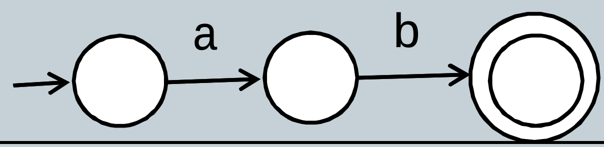
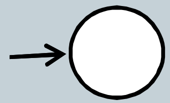
- Recursive nature of regular expressions
- A regular expression R= (1+0(10)*)* can be written as by doing following replacement.
- Replace 1+0(10)* by P so R= P*
- Rewrite P as P1 + P2 where P1=1 and P2=0(10)*
- P2 can be written as P3. P4 where P3= 0 and P4=(10)*
- P4 can be written as P5* where P5=10
- P5 can be written as P6.P7 where P6 is 1 and P7 is 0.

- From this recursive nature of RE we can conclude that if FAS for two RE R1 and R2 are given and if we can construct composite FAs for

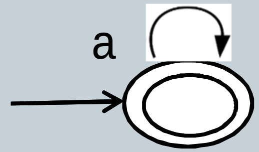
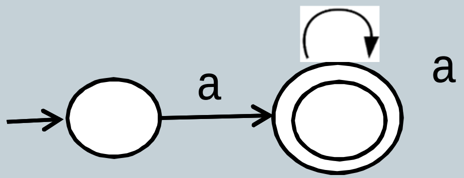
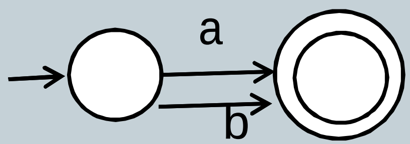
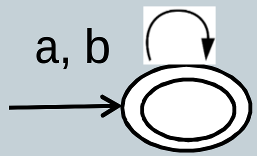
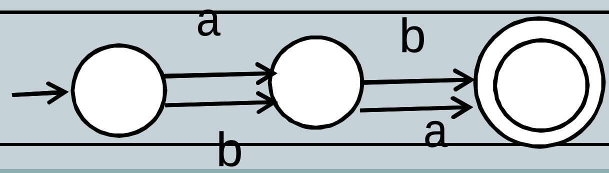
  1. R1+R2 2. R1.R2 3. R1*

  Then we can construct Fas for any RE.

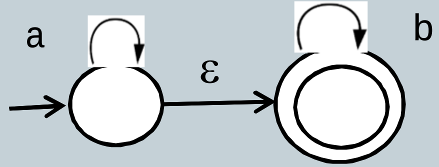
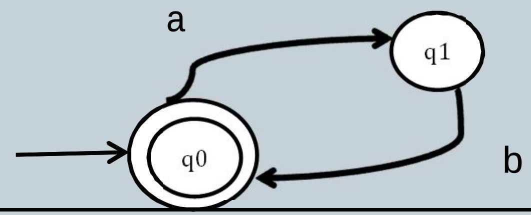
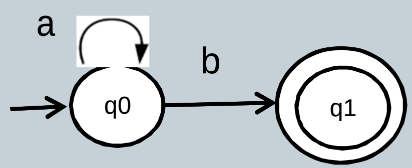
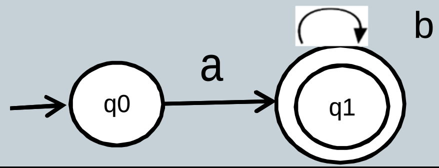
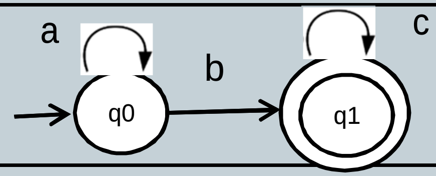
FA representing RE

Sr. No.	Regular Expression	Automata
1	RE=ε	
2	RE=a	
3	RE=a+b	
4	RE=a.b	
5	RE=φ	

FA representing RE

Sr. No.	Regular Expression	Automata
6	RE=a*	
7	RE=aa* or a^+	
8	RE=a+b	
9	RE=(a+b)*	
10	RE=ab+ba	

FA representing RE

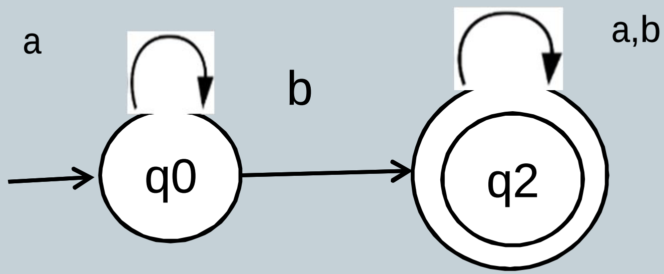
Sr. No.	Regular Expression	Automata
6	RE=a*b*	
7	RE=(ab)*	
8	RE=a*b	
9	RE=ab*	
10	RE=a*bc*	

Examples on RE to Fas conversion

- Ex. 1: Convert following RE to FAs
  1. a*b(a+b)*
  2. (ab)*ab*
  3. (a+ba)*ba
  4. (aa+aaa)*
  5. (0+1(01)*)*

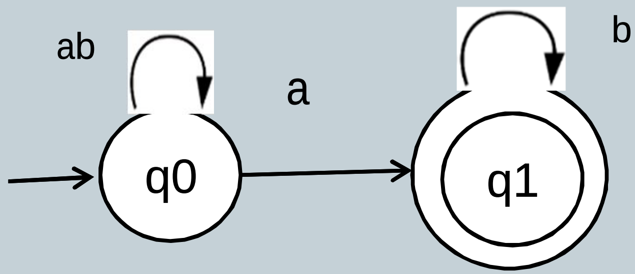
Solution:

1. a*b(a+b)*

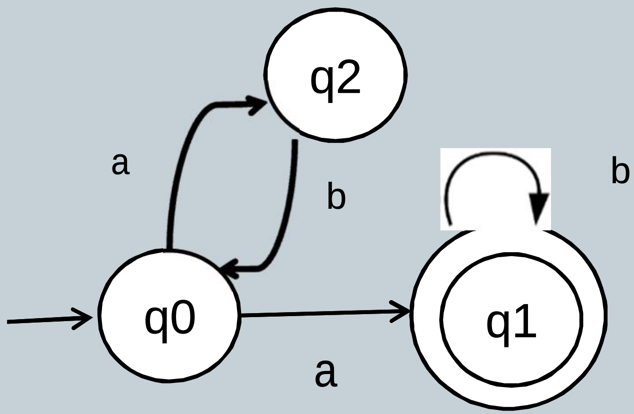


Examples on RE to Fas conversion

2. (ab)*ab*

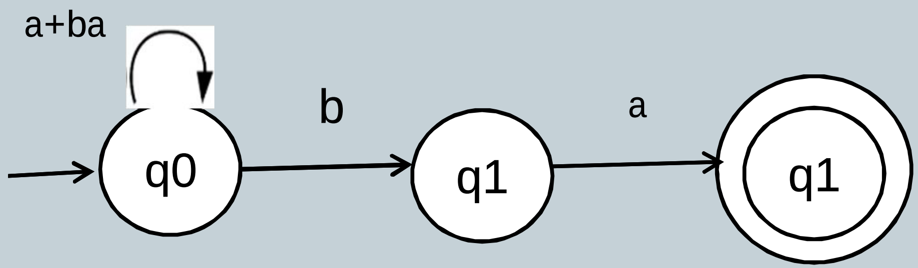


Divide transition ab by adding one more state.

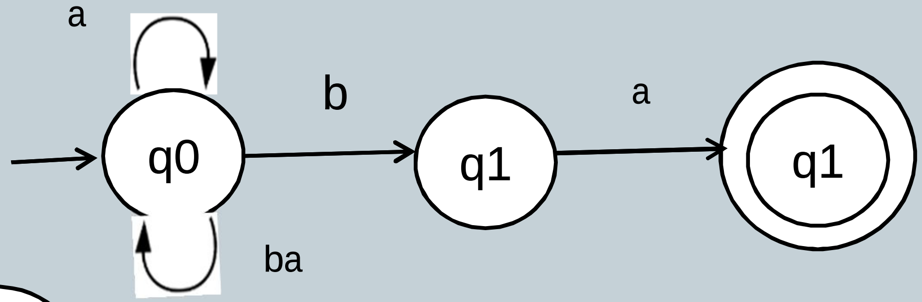


Examples on RE to Fas conversion

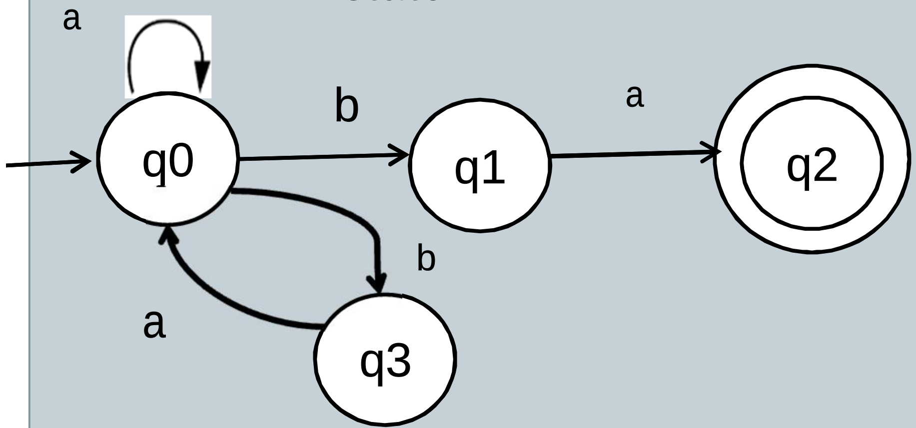
3. (a+ba)*ba



Divide transition a+ba by adding one more transition.

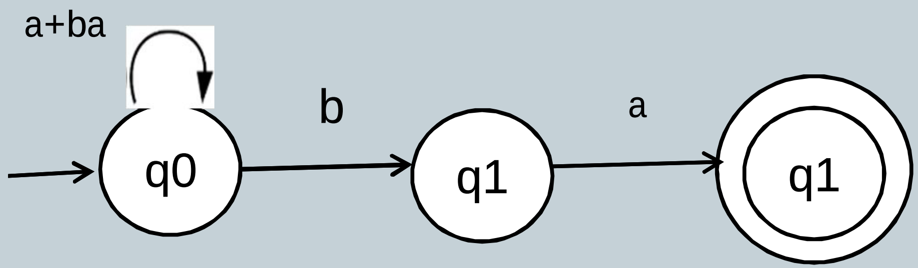


Divide transition ba by adding one more state.

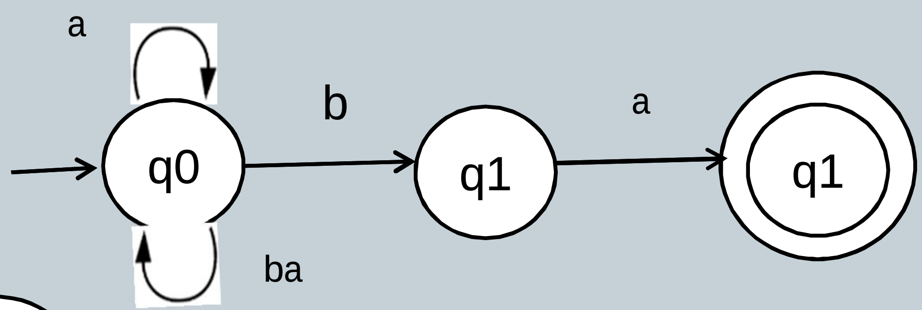


Examples on RE to Fas conversion

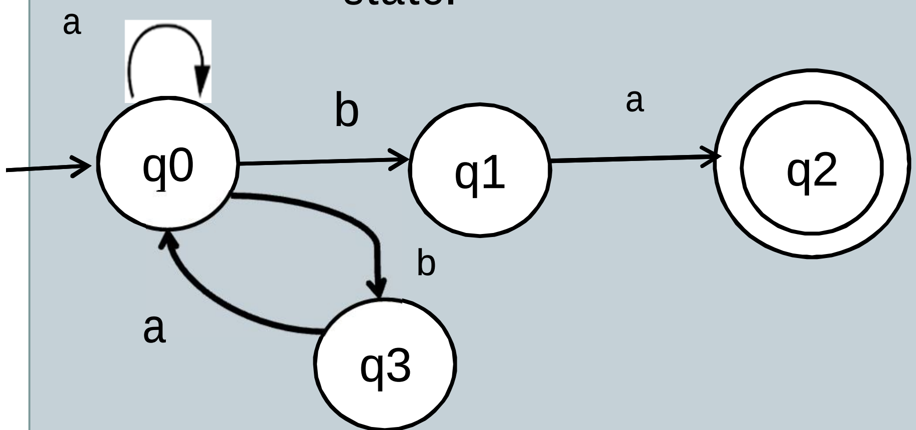
3. (a+ba)*ba



Divide transition a+ba by adding one more state.

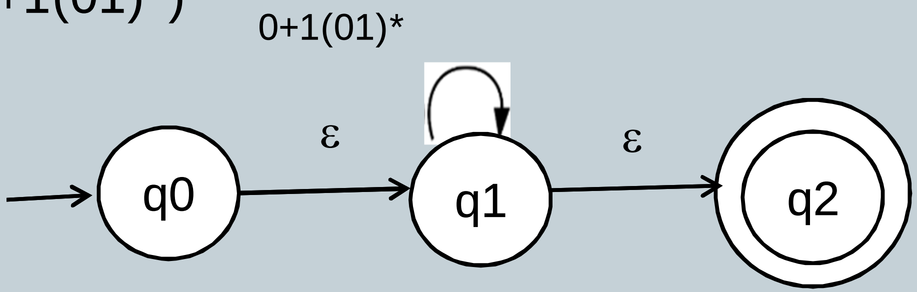


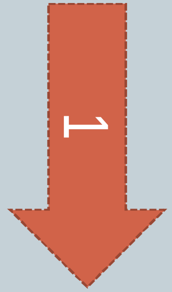
Divide transition ba by adding one more state.



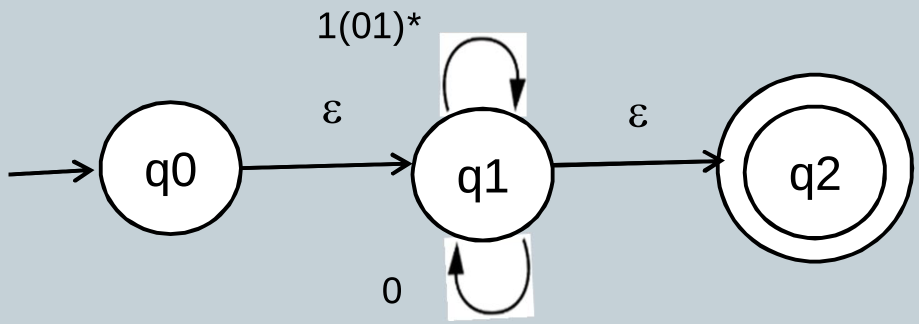
Examples on RE to Fas conversion

5. (0+1(01)*)*

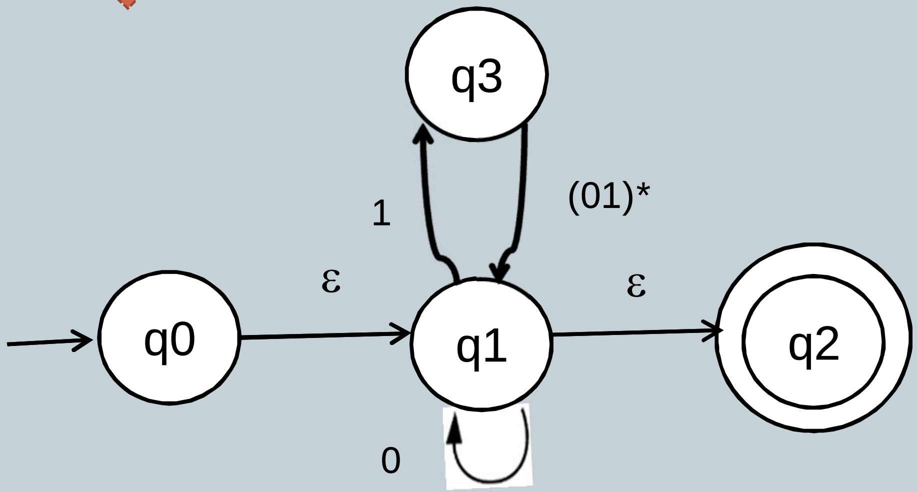


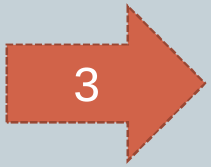


Divide transition 0+1(01)* by adding one more transition.



Divide transition 1(01)* by adding one more state.

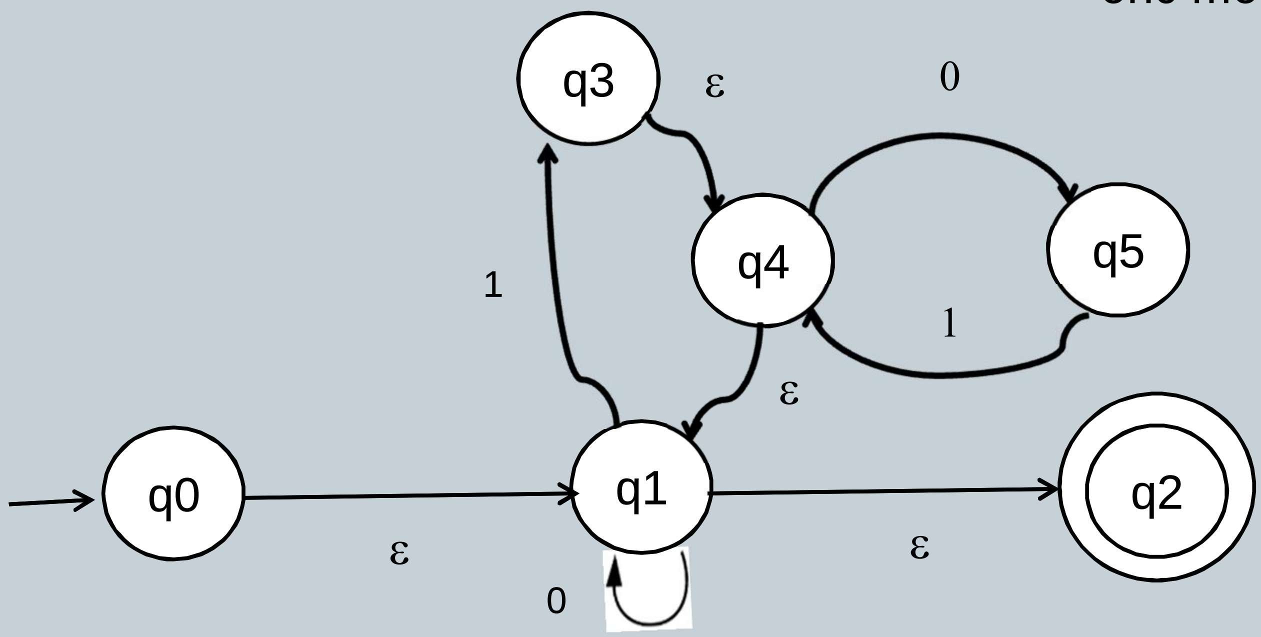




Examples on RE to Fas conversion

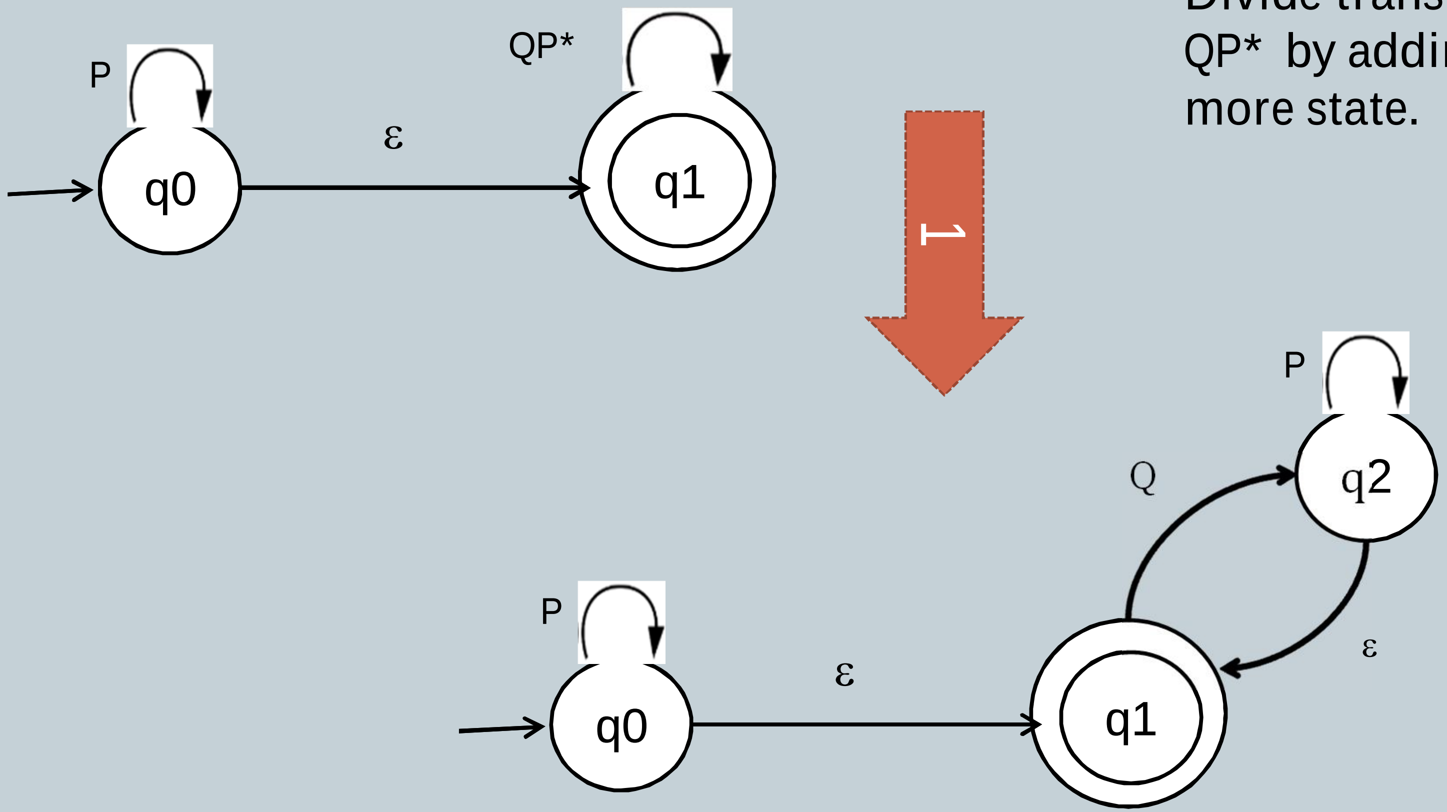
5. (0+1(01)*)*

Divide transition 1(01)* by adding one more state.



Examples on RE to Fas conversion

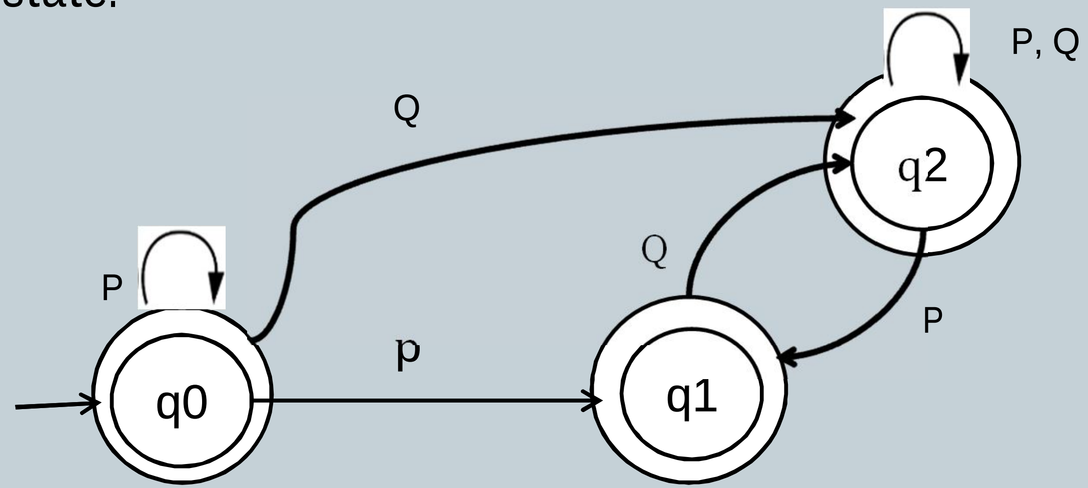
Ex. 2: Show that P*(QP*)* = (P+ Q)*



Divide transition QP* by adding one more state.

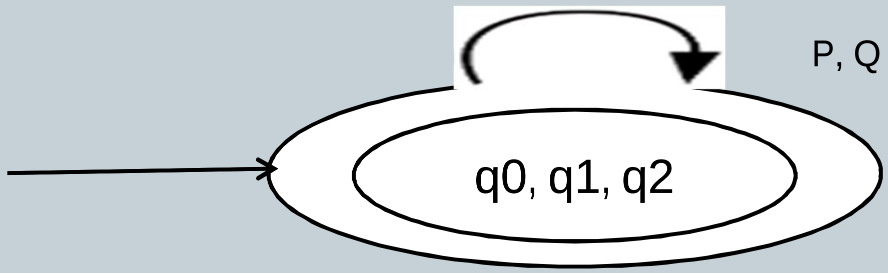
Examples on RE to Fas conversion

- Now remove ε move from q1 and q0
- To remove ε move from q2 to q1
  - Duplicate transition of q1 on q2
  - Make q2 as final state.
- To remove ε move from q0 to q1
  - Duplicate transition of q1 on qo
  - Make q0 as final state.



Examples on RE to Fas conversion

- As all q0, q1, q2 are final state so they can be merged in to single state.

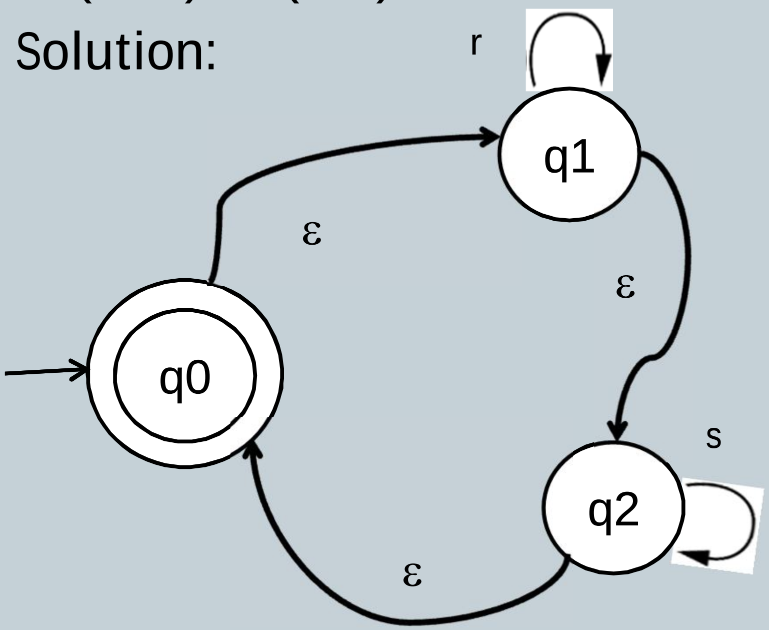


- Thus both Expressions are equivalent so P*(QP*)* = (P+ Q)*.

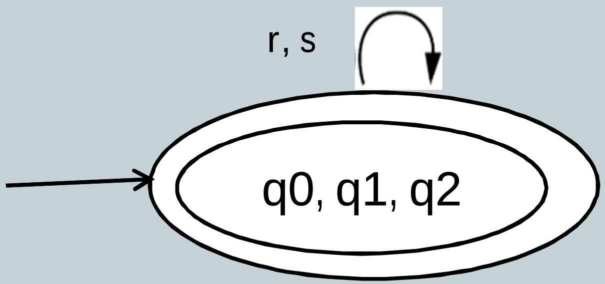
Examples on RE to Fas conversion

- Ex 3: Prove or disprove the following for a regular expressions.
- $(r^*s^*)^* = (r+s)^*$

Solution:



- ε -NFA to DFA
- ε Closure of q0= {q0, q1, q2}
- ε Closure of q1= {q0, q1, q2}
- ε Closure of q2= {q0, q1, q2}
- So the equivalent DFA can be shown as bellow



- So we can say both RE are equivalent.
- $(r^*s^*)^* = (r+s)^*$

Examples on Regular Expressions

- Ex. 4: Check the equivalence of the regular expression.

1. (a* bbb)*a*   AND a*(bbb a*)*

Solution: Substitute P for a* and Q for bbb we will get:

- (PQ)*P and P(QP)* so both are equivalent from the besic lemma on RE.
- Thus both Expressions are equivalent so (QP)*P = P(QP)*.

2. ((a+bb)* aa)*   AND ε+(a+bb)* aa

Solution: Substitute P for a+bb and Q for aa we will get:

(P*Q)* and ε+ P*Q

Thus both Expressions are not equivalent.

Examples on Regular Expressions

- Ex. 5: Check the equivalence of the regular expression.
- (ab)* is not equal to a*b*

Solutions:

The string abab $\in$ (ab)*

But abab $\notin$ a*b*

- And hence both are not equal.

Algebraic Laws of Regular Expressions

- There are a number of laws for algebraic laws, such as:

1. Associative And commutative
2. Identities and annihilators
3. Distributive law
4. The idempotent law
5. Laws involving closures

Algebraic Laws of Regular Expressions

1. **Associative And commutative**

- The commutative laws for union says that the union of two regular languages can be taken in any order.
- For any two language X and Y

$$X+Y = Y+X$$

- The associative laws holds for union of regular languages.

$$(X+Y)+Z = (X+Z)+Y$$

Algebraic Laws of Regular Expressions

2. Identities and annihilators

- ε Is Identities and Φ is annihilators. There are three rules for regular expression involving ε and Φ .
- $\Phi + L = L + \Phi = L$ Φ is identity for Union +.
- $\varepsilon . L = L . \varepsilon = L$ ε is identity for concatenation.
- $\Phi . L = L . \Phi = L$ Φ is identity for concatenation .

Algebraic Laws of Regular Expressions

3. Distributive Law

- The left distributive law of concatenation over union hold for regular language.

$$Z(X+Y) = ZX + ZY$$

- The right distributive law of concatenation over union hold for regular language.

$$(X+Y)Z = XZ + YZ$$

Algebraic Laws of Regular Expressions

4. The idempotent Law

- It says that the union of two identical expression can be replaced by one copy of expression.

$$L + L = L$$

Algebraic Laws of Regular Expressions

5. Laws involving closures:

- These laws includes:

1. $(L^*)^* = L$ Closure of the closure does not change the language.
2. $\Phi^* = \varepsilon$
3. $\varepsilon^* = \varepsilon$
4. $L^* = LL^* = L^*L$
5. $L^* = L^* + \varepsilon$

Determination of Regular Expressions

- A regular expression over alphabet

  q= (a1, a2, a3..........an) is defined recursively as follows.

1. A null string ε is a Regular Expression.
2. An empty set φ is a Regular Expression .
3. An alphabet a is a regular expression.
4. If x & y are regular expressions then
5. Their concatenation is regular expressions .
6. Closure of x (x*) is regular expressions .

Basic properties of Regular Expressions

- If P, Q & R are regular expressions then basic properties of RE are given bellow.

1. $\Phi + R = R$
2. $\Phi . R = R . \Phi = R$
3. $\varepsilon . R = R . \varepsilon = R$
4. $\varepsilon * = \varepsilon$
5. $\Phi^* = \varepsilon$
6. $R + R = R$
7. $PQ + PR = P(Q + R)$
8. $QP + RP = (Q + R)P$

Basic properties of Regular Expressions

9. $R^* R^* = R^*$

10. $R R^* = R^* R$

11. $(R^*)^* = R^*$

12. $\varepsilon + R R^* = R^*$

13. $(PQ)^* P = P (QP)^*$

14. $(P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^*$

15. $(P^*Q)^* = \varepsilon + (P + Q)^*Q$

Examples on Regular Expressions

- Ex. 1: Prove that (1+00*1)+ (1+00*1) (0+10*1)* (0+10*1)=0*1 (0+10*1)
- LHS= (1+00*1)+ (1+00*1) (0+10*1)* (0+10*1)

= (1+00*1) (ε + (0+10*1)* (0+10*1)) .... By Eq. 7

= (1+00*1) (0+10*1)* ....By Eq. 12

= [(ε + 00*) 1] (0+10*1)* ....By Eq. 8

= 0* 1 (0+10*1)*

= RHS

Examples on Regular Expressions

- Ex. 2: If S= {a, bb} find the set of all strings in S* with string length less than or equal to 5. Also for given S, Prove whether following is true or false.
  - (S*)*=(S+)*

Solution:

String of length 0= ε

String of length 1= a

String of length 2= aa, bb

String of length 3= aaa, abb, bba

String of length 4= aaaa, bbbb, aabb, bbaa, abba

String of length 5= aaaaa, abbbb, bbabb, bbbba, abbaa, aabba, bbaaa, aaabb

And (S*)*=(S+)* is true as ε is belonging to both expressions.

Examples on Regular Expressions

- Ex. 2: Find all regular expressions for following languages over {a, b}.
  1. The set of all strings ending in b.
  2. The set of all strings ending in ba.
  3. The set of all strings ending neither in b nor in ba.
  4. The set of all strings ending in ab.
  5. The set of all strings ending neither in ab nor in ba.

  Solutions:
  1. The set of all strings ending in b ➔ $(a+b)^*b$
  2. The set of all strings ending in ba ➔ $(a+b)^*ba$
  3. The set of all strings ending neither in b nor in ba. ➔ $(a+b)^*aa+a+\varepsilon$
  4. The set of all strings ending in ab. ➔ $(a+b)^*ab$
  5. The set of all strings ending neither in ab nor in ba.

     ➔ $\varepsilon + a + b + (a+b)^*(aa+bb)$

Examples on Regular Expressions

- Ex. 3: Find all regular expressions for following subset of (0, 1)*.
    1. The Language of all strings containing exactly two 0's.
    2. The Language of all strings containing at least two 0's.
    3. The Language of all strings that do not end with 01.
    4. The Language of all strings starting with 11.

Solutions:

1. The Language of all strings containing exactly two 0's.

   RE ➔ 1*01*01*

2. The Language of all strings containing at least two 0's.

   RE➔ 1*01*0(1+0)*

3. The Language of all strings that do not end with 01.

   RE ➔ (1+0)*(00+11+10)

4. The Language of all strings starting with 11.

   RE ➔ 11(0+1)*

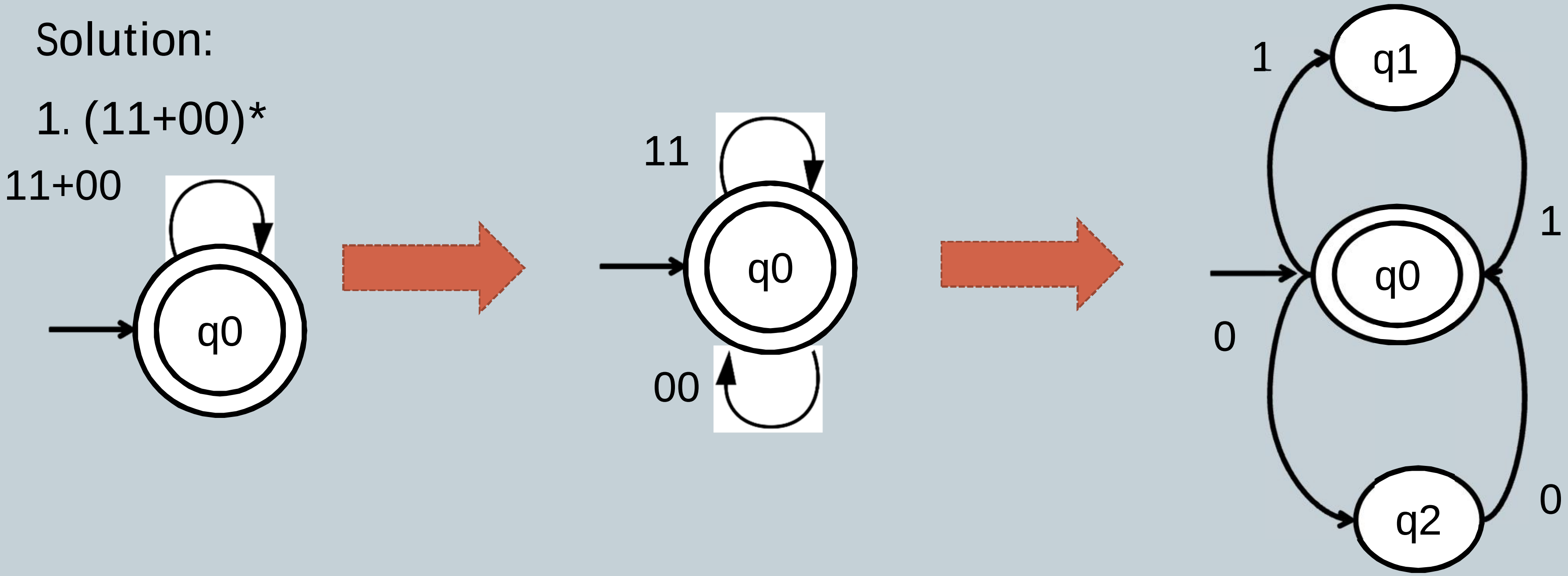
Examples on Regular Expressions

- Ex. 4: For each of the following RE draw DFA.

1. (11+00)* 2. (111+100)*0 3. 0+10*+01*0 4. 10+ (0+11)0* 1

Solution:

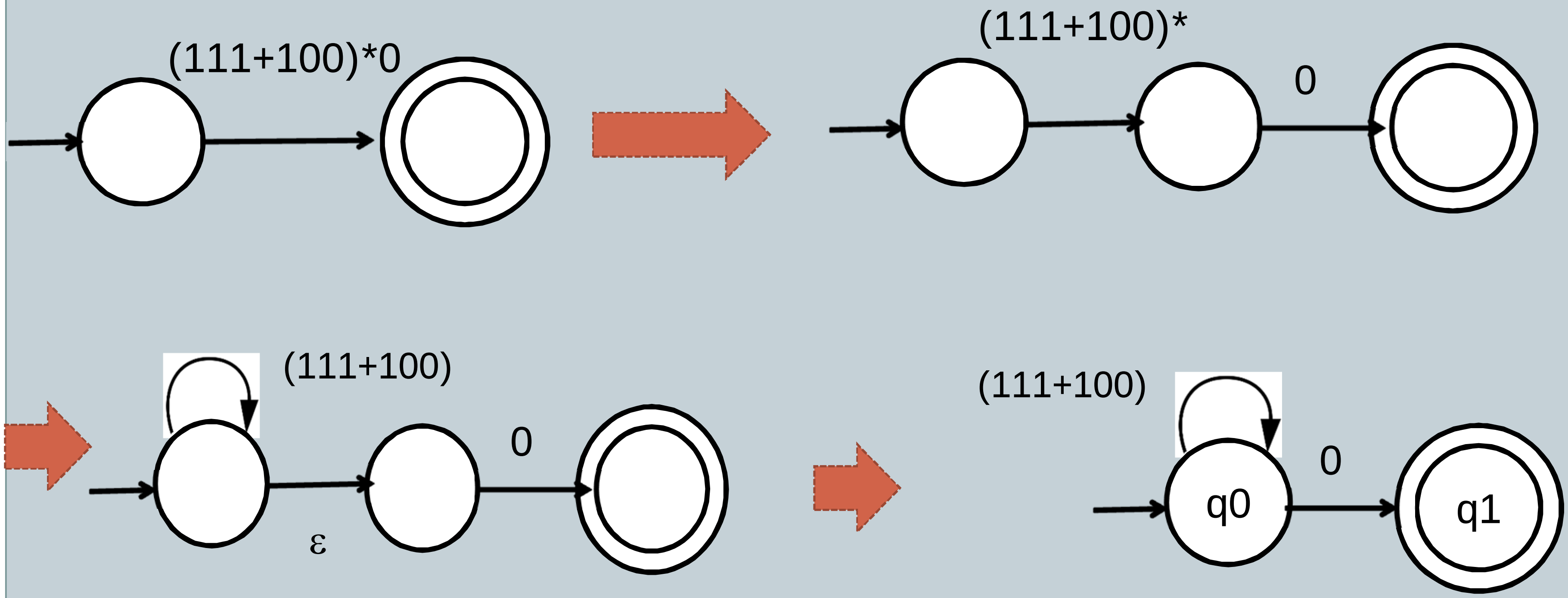
1. (11+00)*



Examples on Regular Expressions

Solution:

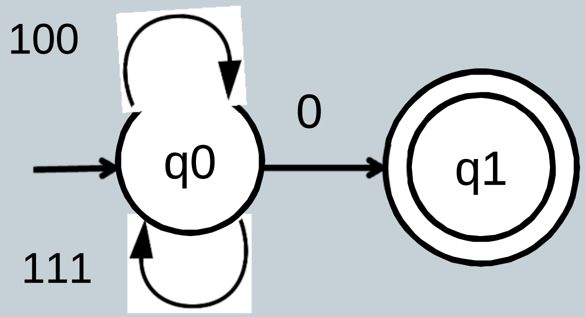
2. (111+100)*0

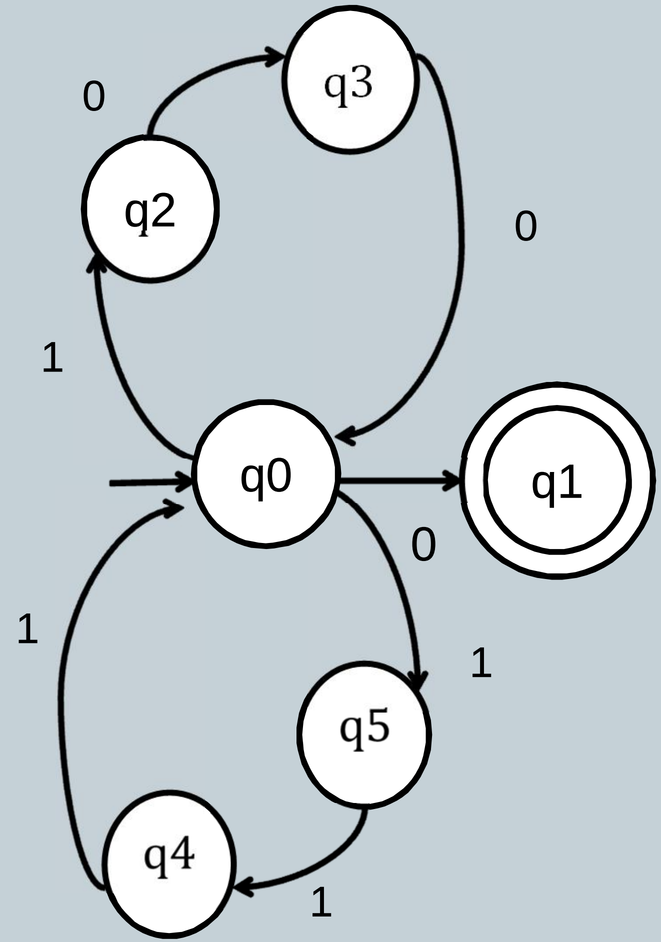


Examples on Regular Expressions

Solution:

2. (111+100)*0

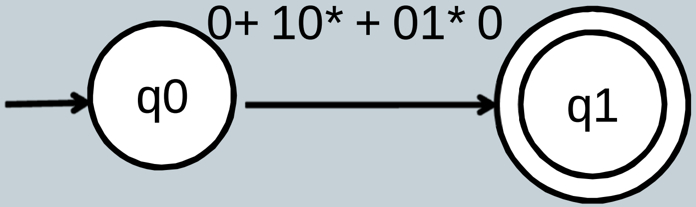


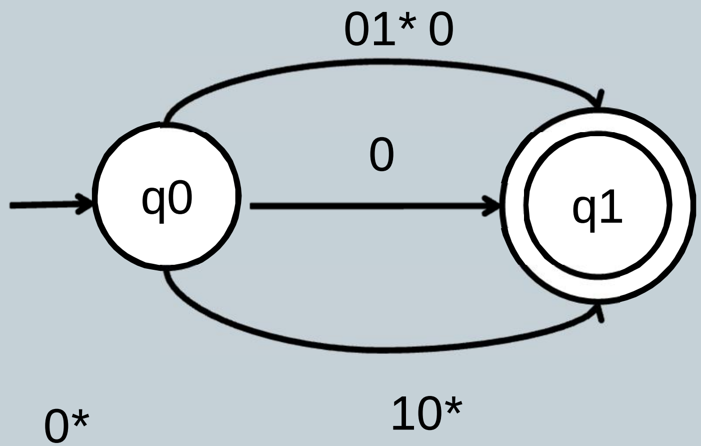


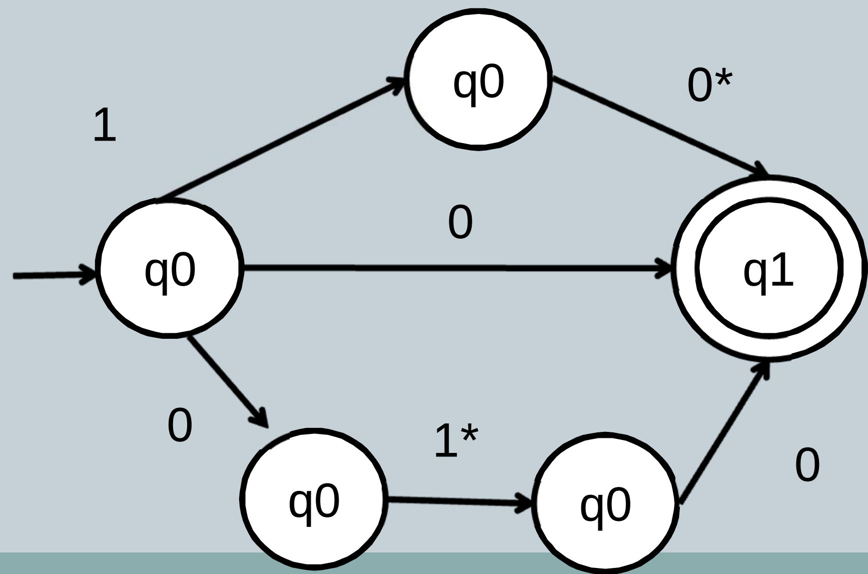
Examples on Regular Expressions

Solution:

3. 0+ 10* + 01* 0



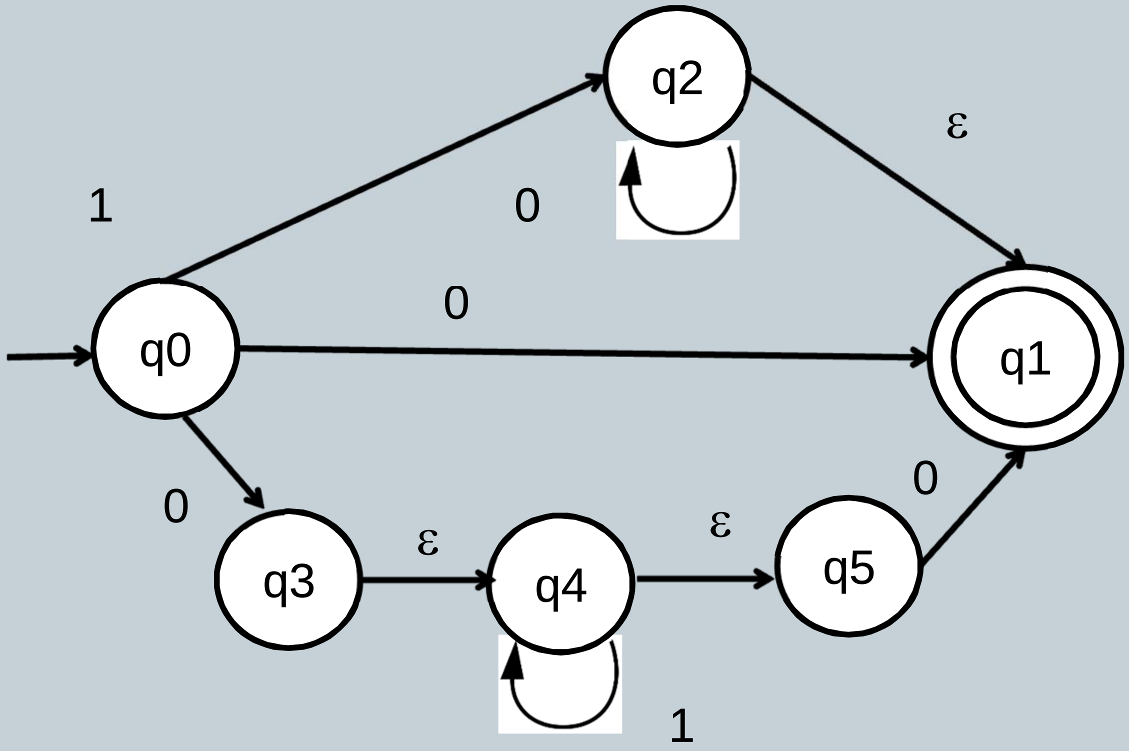




Examples on Regular Expressions

Solution:

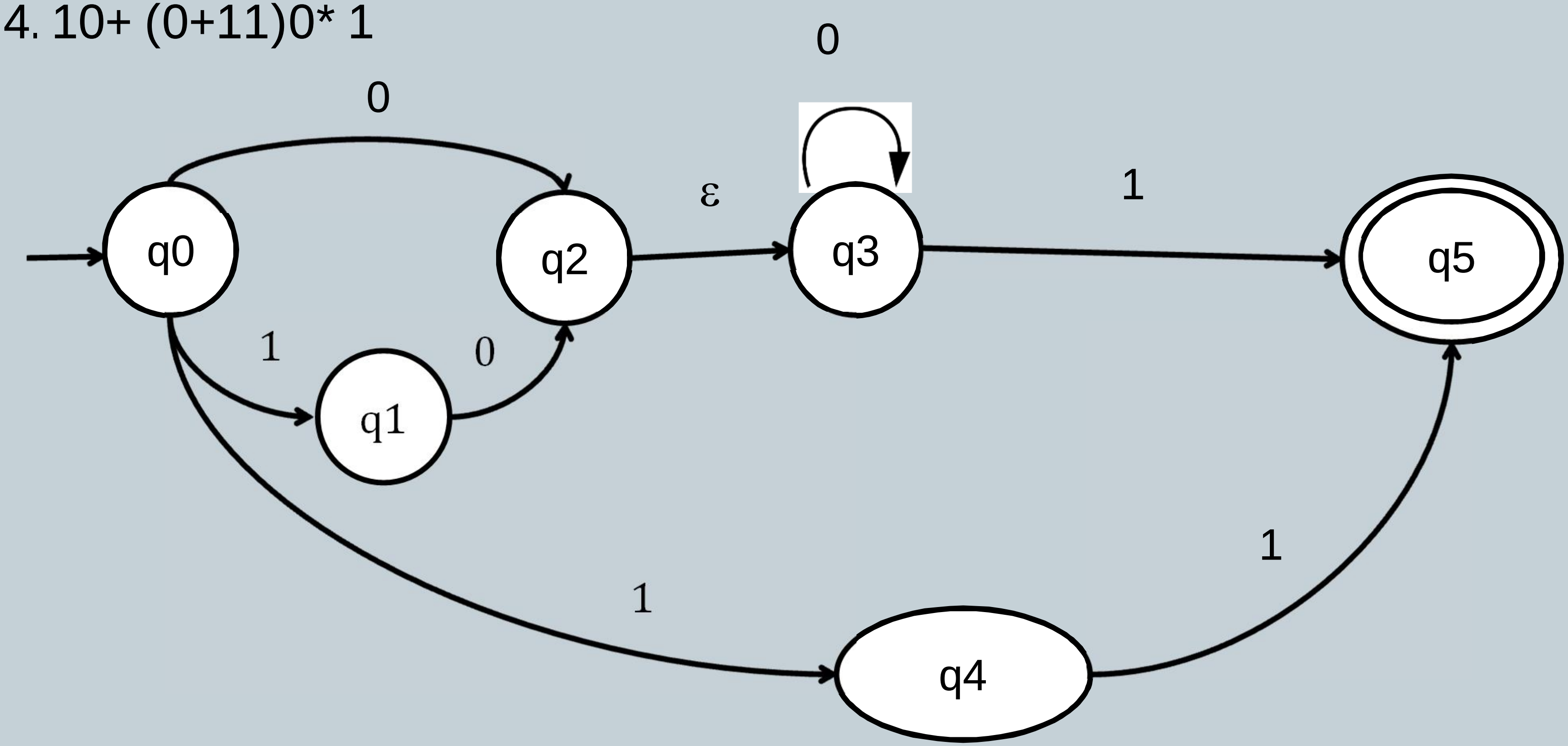
3. 0+ 10* + 01* 0



Examples on Regular Expressions

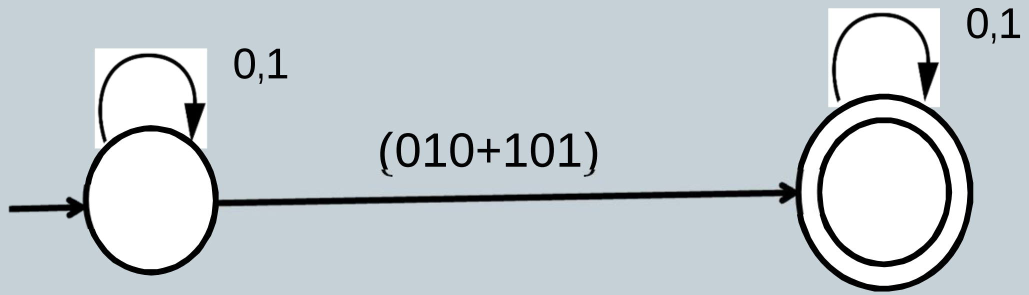
Solution:

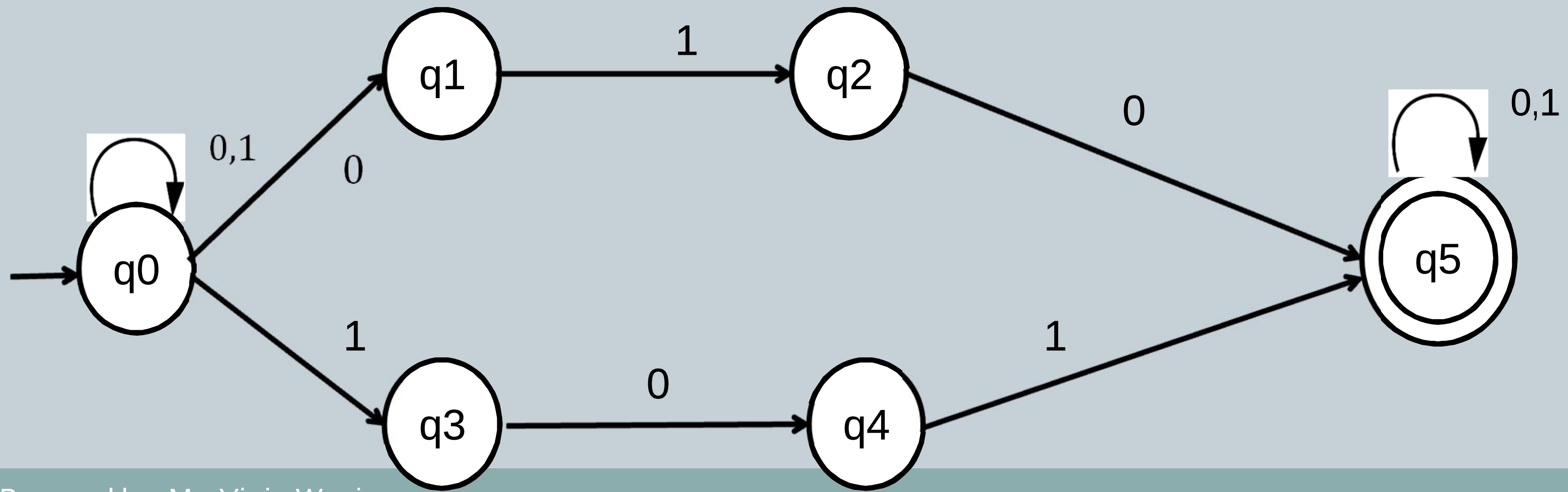
4. 10+ (0+11)0* 1



Examples on Regular Expressions

Ex 5: Find FA for given RE (0+1)* . (010+101) . (0+1)*



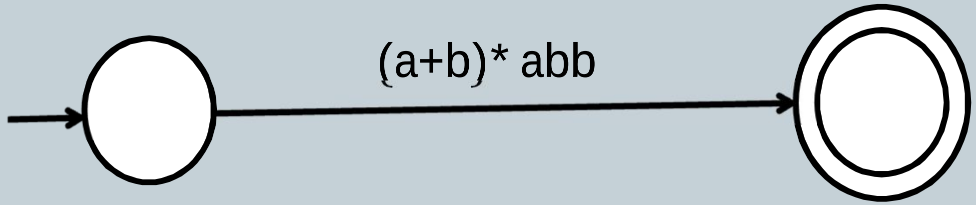


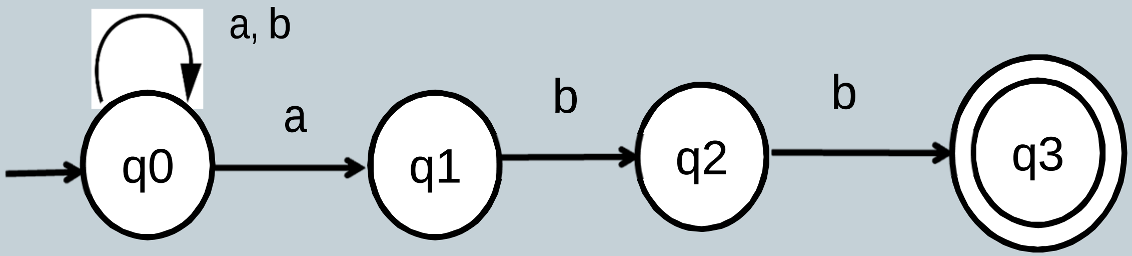
Examples on Regular Expressions

Ex 5: Find FA for given RE   1. (a+b)* abb 2. (11)* . 010 . (11)*

Solution:

1. (a+b)* abb

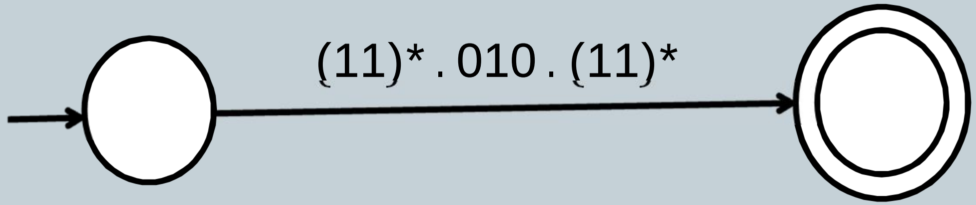


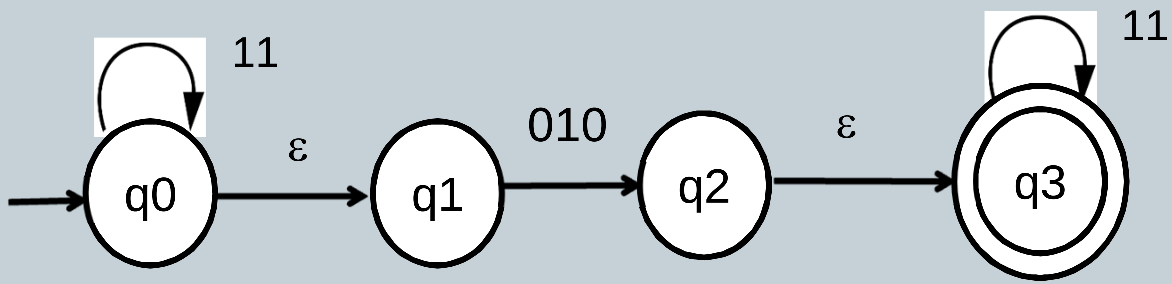


Examples on Regular Expressions

Solution:

2. (11)* . 010 . (11)*

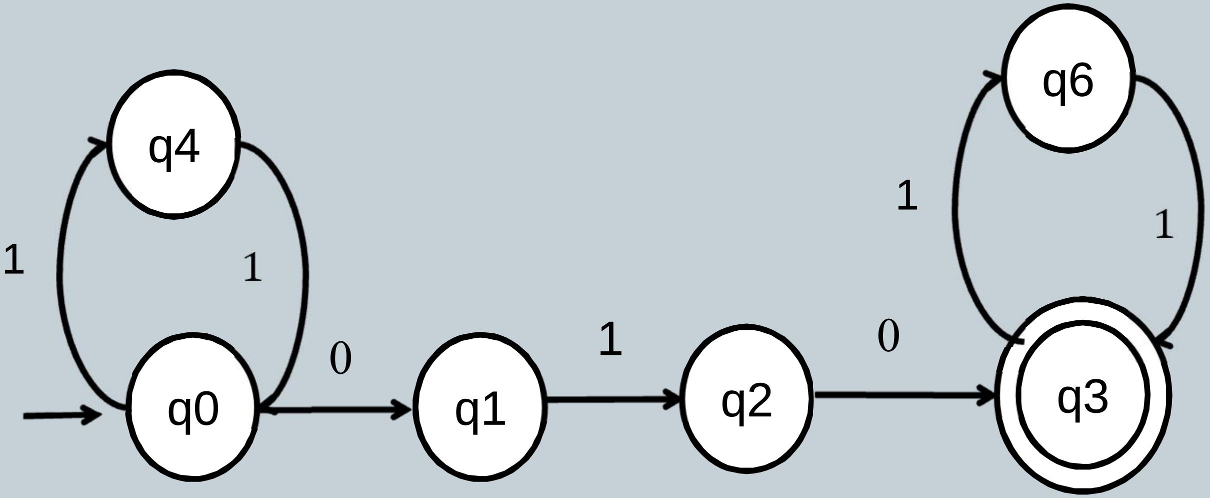




Examples on Regular Expressions

Solution:

2. $(11)^* . 010 . (11)^*$



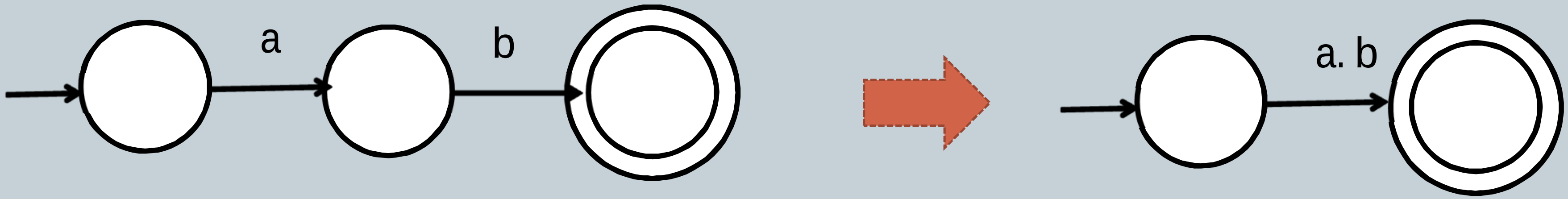
DFA to Regular Expressions

- There are two methods to convert DFA to regular expressions.
- By State or Loop elimination.
- Arden's Theorem

DFA to Regular Expressions : By State or Loop elimination.

- The process to Eliminate State/loop:

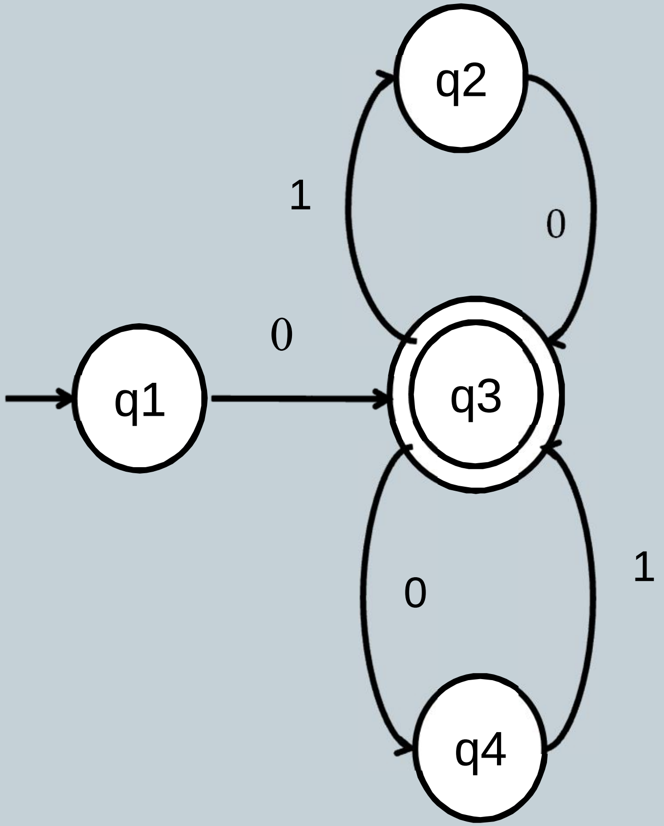
1. Every state qi $\notin$ F can be eliminated if qi is not initial state.
2. That is qi can be eliminated if it is not a initial or final state.
3. Elimination of state involve writing of regular expression as label on arc.
4. Example:

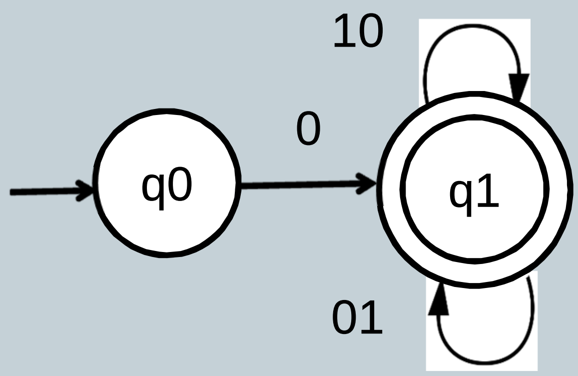


RE= ab

DFA to Regular Expressions : By State or Loop elimination.

Example: 2

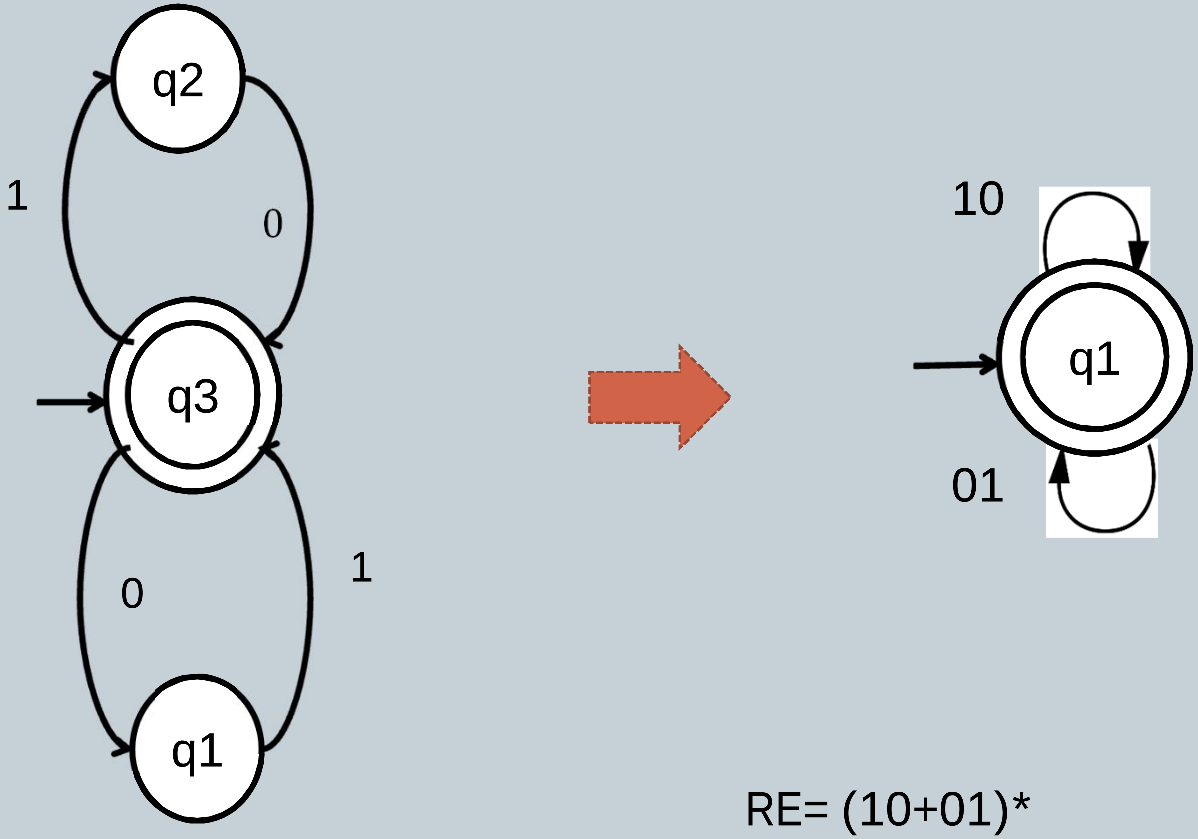




RE= 0(10+01)*

DFA to Regular Expressions : By State or Loop elimination.

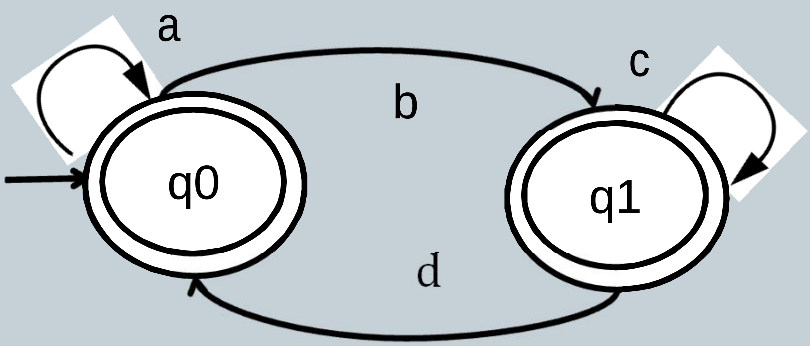
Example: 3



RE= (10+01)*

Examples on DFA to RE: By State or Loop elimination

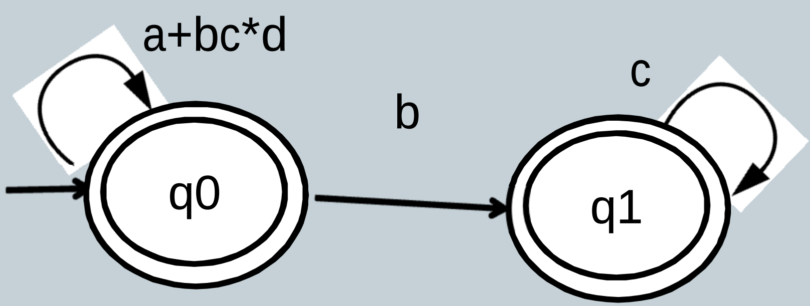
Examples 1: Find a Regular expression for the given two State machine.



Solution:

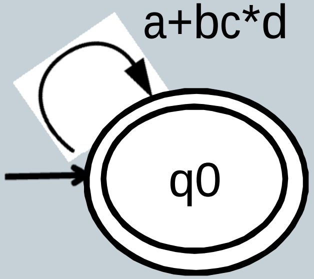
Now here nor q0 or q1 can be eliminated as both are final states.

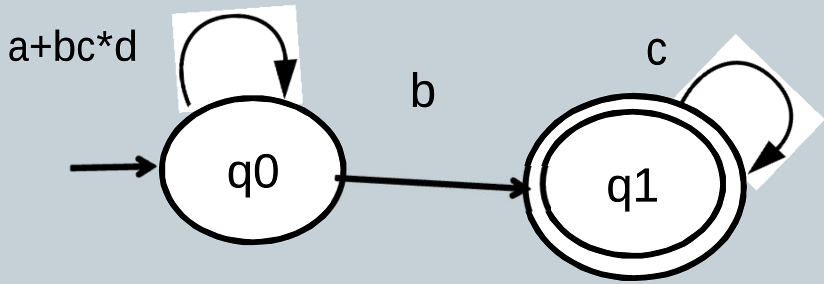
But loop can be eliminated by moving its effect either to state q0 or q1, so let us move it to q0.



Examples on DFA to RE: By State or Loop elimination

The above machine has two final states. The machine can be divided in to two machines, Machine with q0 as final state and machine with q1 as final state.



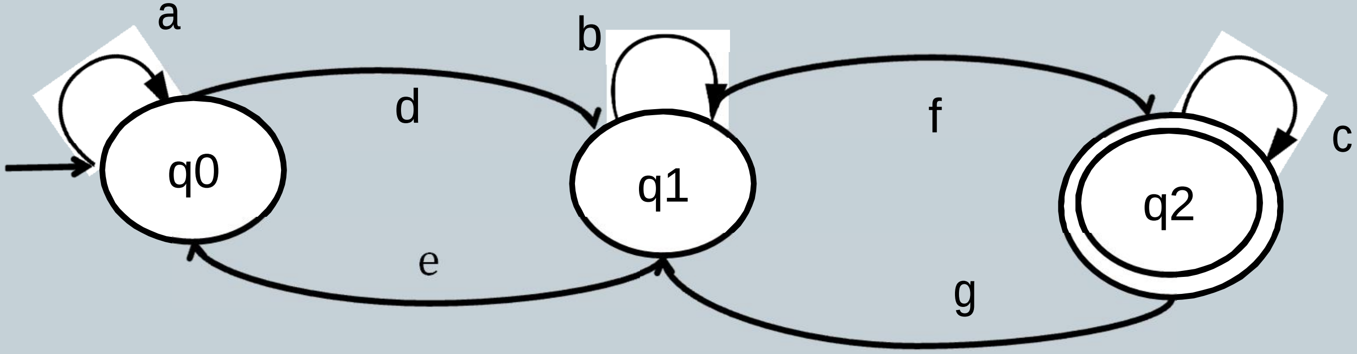


So RE for first machine with q0 final state (RE1)= (a+bc*d)*
And RE for Second machine with q1 final state (RE2)= (a+bc*d)* bc*
So the final RE can be obtained by taking union of both the RE.

So RE= RE1+RE2
RE= (a+bc*d)* + (a+bc*d)* bc*
= (a+bc*d)* (ε + bc*)

Examples on DFA to RE: By State or Loop elimination

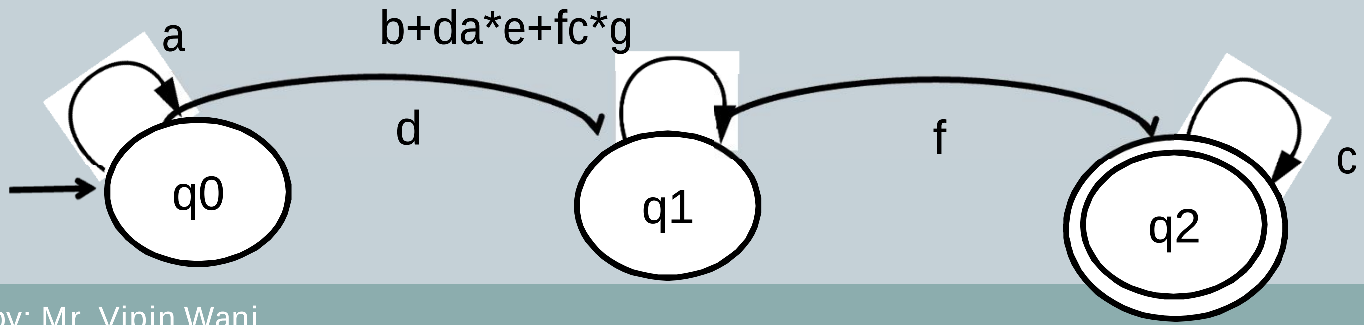
Example 2: Find Regular Expression for given three state machine.



Solution: For given machine state q1 is the crossing point o three loops

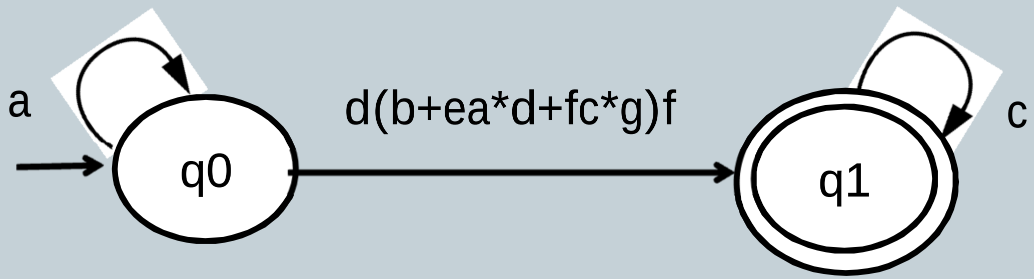
1. q1 to q1 on input b.
2. Loop between q1 and q0 on input ea*d
3. Loop between q1 and q2 on input fc*g

Now eliminate the loop by moving these loop effects on state q1.



Examples on DFA to RE: By State or Loop elimination

Example 2:
Now eliminate the state q1 can be eliminated.



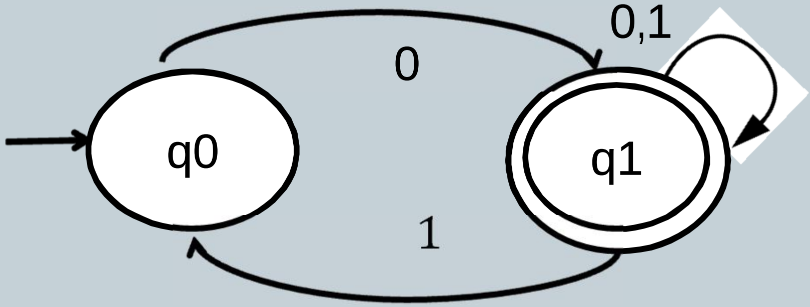
Now the RE for given machine is

RE = a* d(b+ea*d+fc*g)f c*

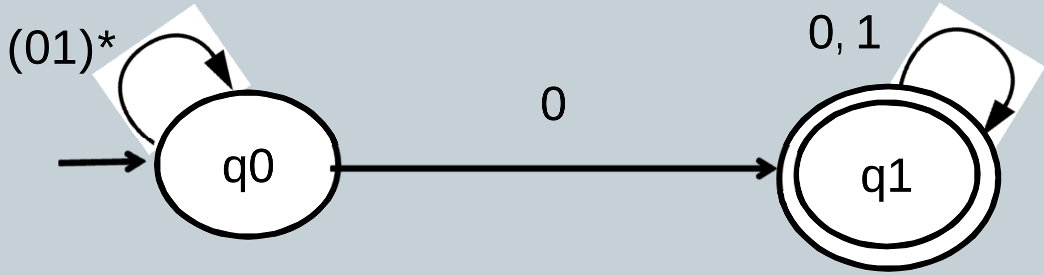
Examples on DFA to RE: By State or Loop elimination

Example 3: Find Regular Expression for given three state machine

1.



Solutions: Machine is reduced by moving effect of loop between q0 and q1 to q0

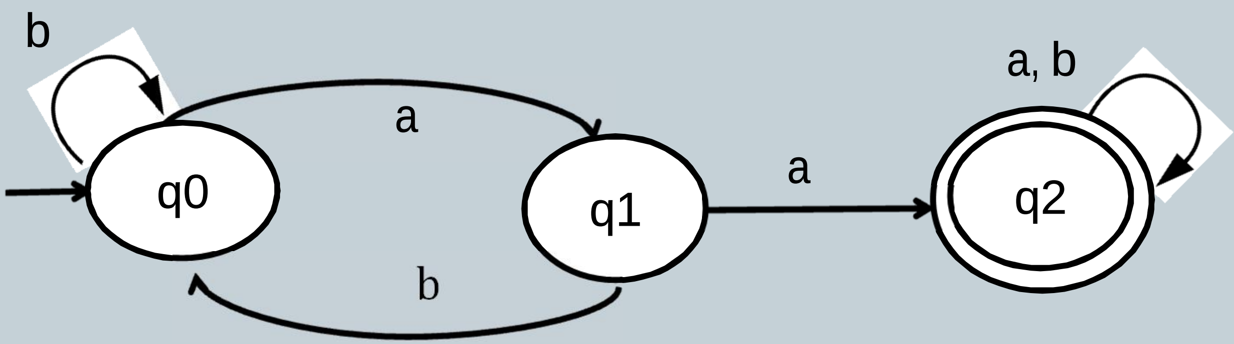


So final require RE= 01*0 (0+1)*

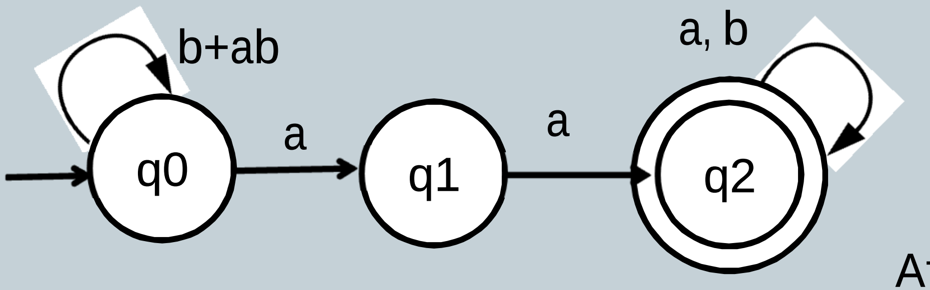
Examples on DFA to RE: By State or Loop elimination

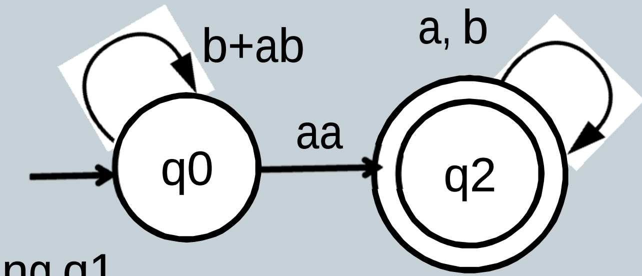
Example 3: Find Regular Expression for given three state machine

2.



Solutions: Machine is reduced by moving effect of loop between q0 and q1 to q0





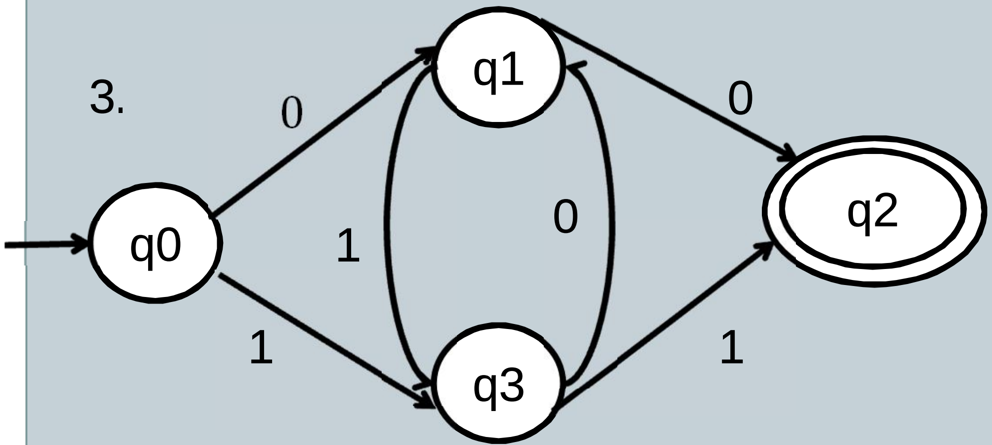
After eliminating q1

So final required RE= (b+ab)* aa (a+b)*

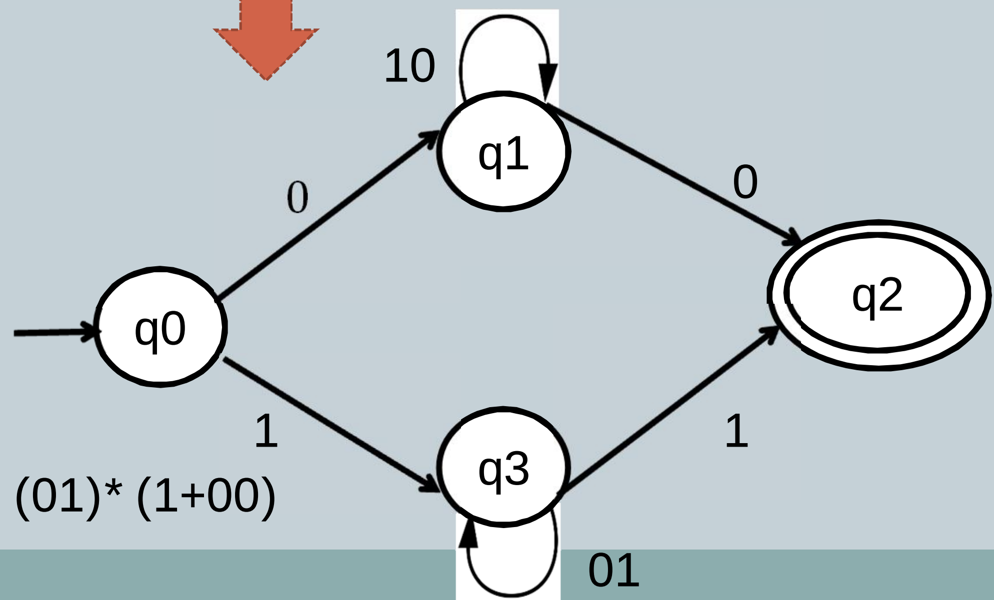
Examples on DFA to RE: By State or Loop elimination

Example 3: Find Regular Expression for given three machines

3.



Solutions: Machine is reduced by moving effect of loop between q0 and q1 to q0



So final required RE= 0(10)* (0+11)+ 1 (01)* (1+00)

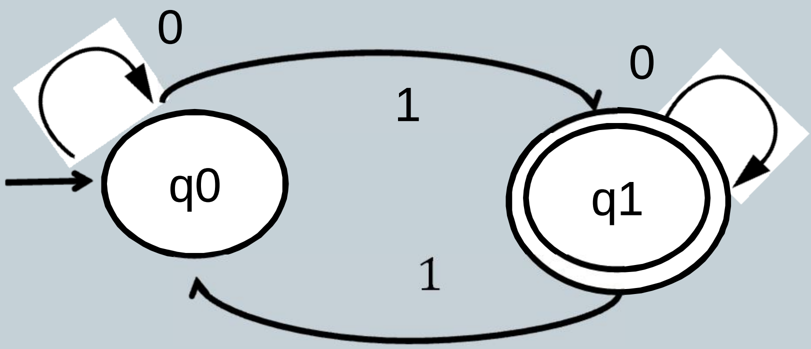
Examples on DFA to RE: By State or Loop elimination

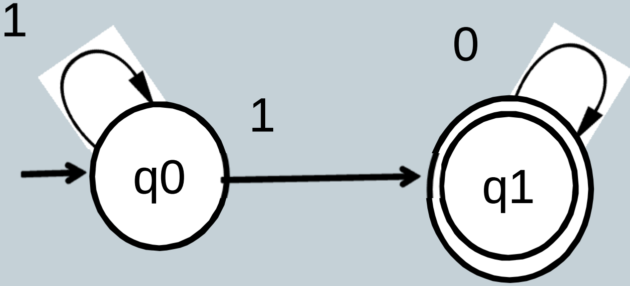
Example 4: Write a RE for the following.

1. **∑ =(0, 1) Odd number of 1's in String.**
2. **∑ =(0, 1) Triple 0 must never appear in Strings.**
3. **Identities in C language.**
4. **Obtain plain English language represented by RE.**
   a. **0* (10* 10*)* 1(0* 10* 1)* 0***
   b. **0* (0* 10* 1)* 0***

Solution:

i. DFA for Given Language is

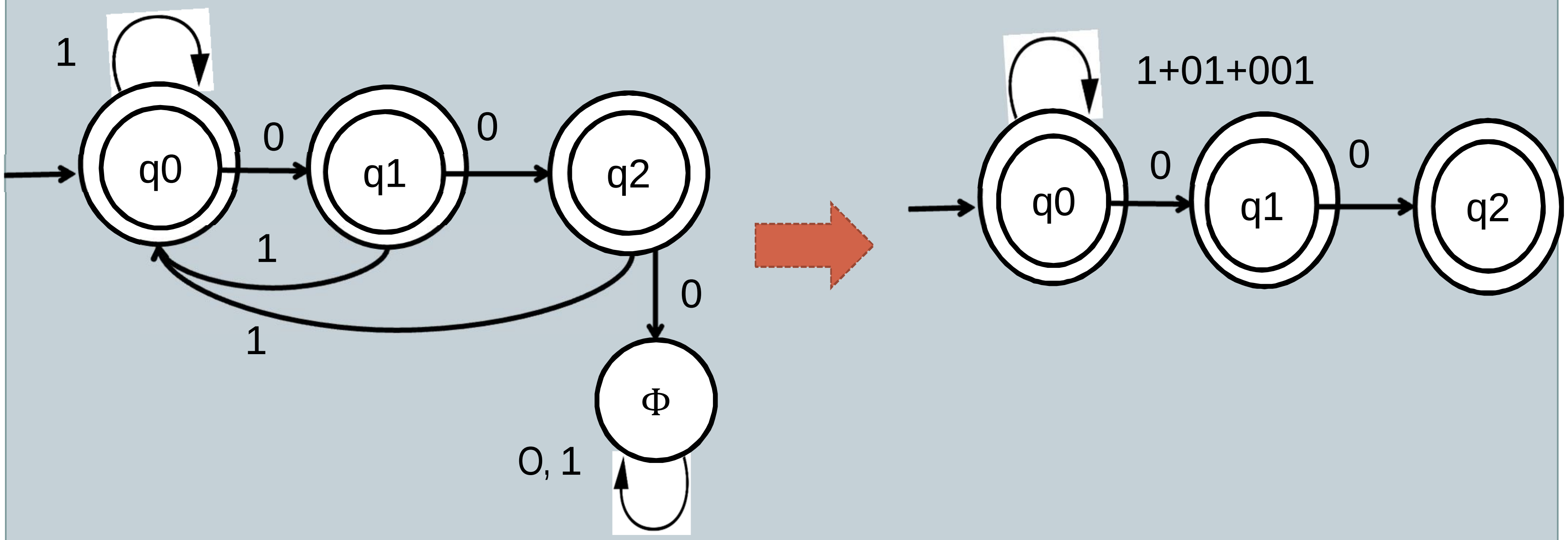




RE=(0+10*1)*10*

Examples on DFA to RE: By State or Loop elimination

ii. DFA for Given Language is



RE= (1+01+001)*+0(1+01+001)*+00(1+01+001)*)

RE=(1+01+001)*(ε+0+00)

Examples on DFA to RE: By State or Loop elimination

iii. Identities in C language

RE= (A+ B + C +........+ Z + a + b + c + ... + z + 0 + 1 +........+ 9)

3. **iv. Obtain plain English language represented by RE.**

   a. **0* (10* 10*) 1(0* 10* 1)* 0***

   ➔ **The Language with all strings containing odd numbers of 1s.**

   b. **0* (0* 10* 1)* 0***

   ➔ **The Language with all strings containing even numbers of 1s.**

Examples on DFA to RE: By State or Loop elimination

Example 5: Write RE for following language over {0, 1}*.

a. The set of strings that begin with 110.

b. The set of all strings not containing 101 as a substring.

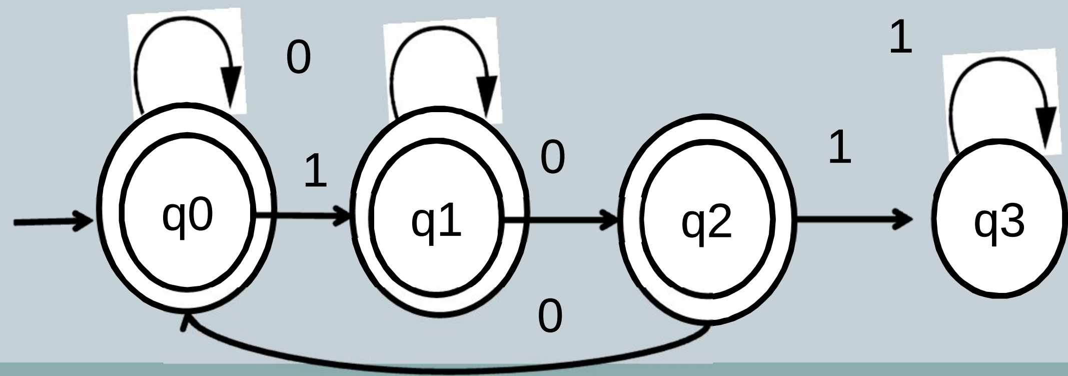
Solutions:

a. The set of strings that begin with 110.

RE ➔ 110 (1+0)*

b. The set of all strings not containing 101 as a substring.

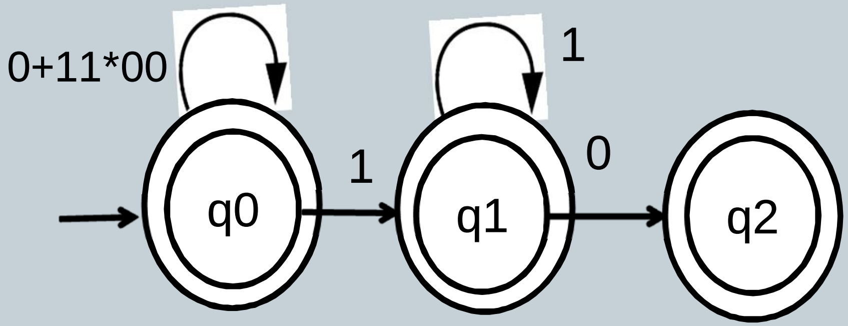
DFA for given language is.



Examples on DFA to RE: By State or Loop elimination

Example 5:

b. The set of all strings not containing 101 as a substring.



RE = (0+11* 00) * $(\varepsilon + 11* + 11* 0)$

Arden's Theorem

Let P, Q and R be regular expressions on Σ. Then if p does not contain ε then the following equation

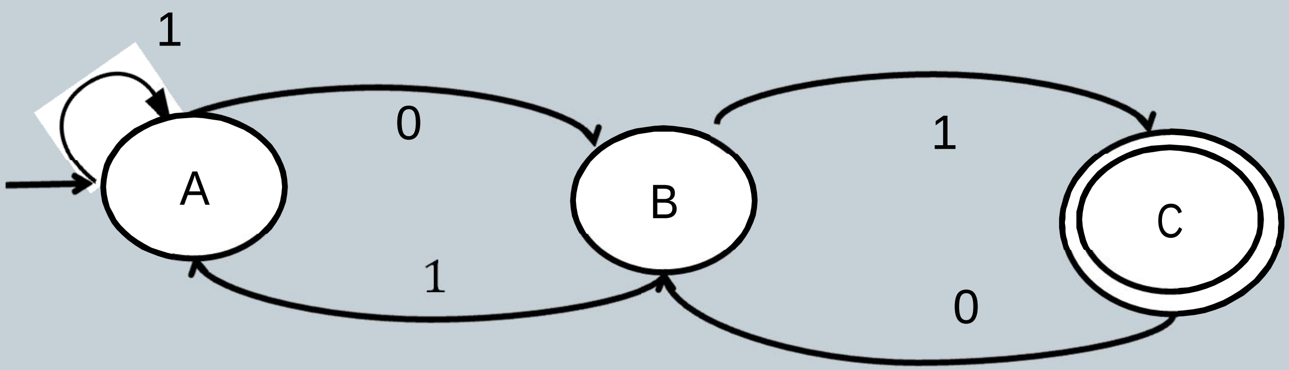
R=Q + RP ....(1)

Has the unique solution given by

R= QP*. ---------------------(2)

Arden's Theorem Solved Examples

Example 1: Consider the following transition diagram convert it to an equivalent RE using Arden's theorem.



Solution: The Set of equations for regular sets (Incoming Transition to every state (A, B & C)) represented as follow.

$A = A1 + B1 + \varepsilon$ ....(1)

$B = A0 + C0$ ....(2)

$C = B1$ ....(3)

Arden's Theorem Solved Examples

Example 1:

So now Equation (1) can be rewritten as

A= A1+(B1+ ε) = (B1+ ε) + A

Now compare above equation with first equation of Arden's theorem. It is of the form R=Q+RP. With R=A, Q= (B1+ ε) & P=1 so it's solution is given by.

R=QP*;

So A= (B1+ ε) 1* ....(4)

Now substitute the value of A from equitation 4 & the value of C from equitation 3 in eq. 2

B= (B1+ ε) 1* 0 + B10

=1*0+B11*0+B10

=1*0+B(11*0+10)

Arden's Theorem Solved Examples

Example 1:

Now compare above equation with first equation of Arden's theorem. It is of the form R=Q+PR. With R=B, Q= 1*0 & P=(11*0+10) so it's solution is given by R=QP*;

B= 1*0(11*0+10)* ....(5)

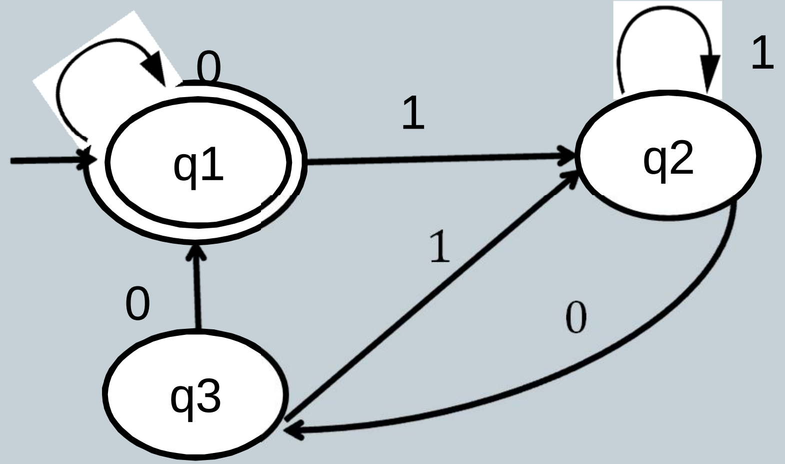
Now substitute the value of B from equitation 5 in equitation 3

C= 1*0(11*0+10)* 1

Since the state C is final state the regular set represented by C is required RE for given FA.

Arden's Theorem Solved Examples

Example 2: Consider the following transition diagram convert it to an equivalent RE using Arden's theorem.



Solution: The Set of equations for regular sets (Incoming Transition to every state (q1, q2 & q3)) represented as follow.

$q_1 = q_1 0 + q_3 0 + \varepsilon$ ....(1)

$q_2 = q_1 1 + q_3 1 + q_2 1$ ....(2)

$q_3 = q_2 0$ ....(3)

Arden's Theorem Solved Examples

Now substitute the value of eq. 3 in eq. 2 we get

$$q_2=q_1 1+q_2 1+q_2 10$$
$$=q_1 1+q_2(1+10)$$

Now compare above equation with first equation of Arden's theorem. It is of the form R=Q+PR. so it's solution is given by R=QP*.

So $q_2=q_1 1(1+10)^*$ ....(4)

Now put value of q3 and q2 in eq1 we get

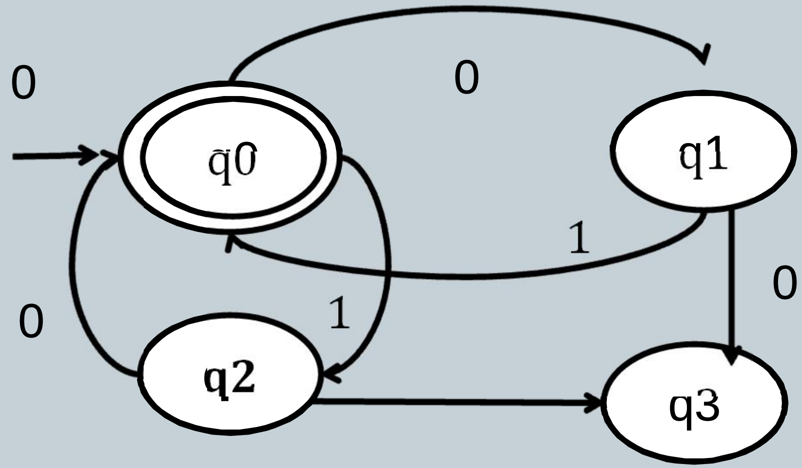
$$q_1= q_1 0+ q_2 00 + \varepsilon$$
$$=q_1 0+ q_1 1(1+10)^* 00 + \varepsilon$$
$$=\varepsilon + q1\ (0+1(1+10)^*00)$$

This eq. is of the form R=Q+PR so by Arden's theorem solution is given by R=QP*.

So q1= (0+1(1+10)*00)* which is required RE.

Arden's Theorem Solved Examples

Example 3: Prove that following FA accepts strings with equal number of 0's and 1's such that each prefix has atmost one more 0 than 1's or at most one more 1 than 0's.



Solution: As q3 is dead state so can be eliminated. The Set of equations for regular sets (Incoming Transition to every state (q1, q2 & q0)) represented as follow.

$q_0 = q_1 1 + q_2 0 + \varepsilon$ ....(1)

$q_1 = q_0 0$ ....(2)

$q_2 = q_0 1$ ....(3)

Arden's Theorem Solved Examples

Example 3:

Now substitute q1 and q2 in eq. 1

$$q_0 = q_0 01 + q_0 10 + \varepsilon \text{ ....(1)}$$

$$q_0 = e + q_0 (01+10)$$

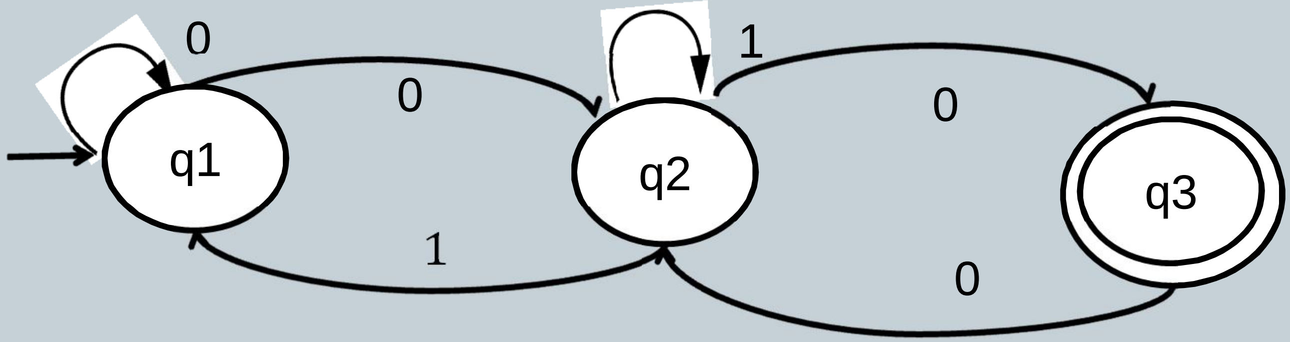
Now compare above equation with first equation of Arden's theorem. It is of the form R=Q+RP. so it's solution is given by R=QP*.

$$q_0 = (01+10)^*$$

RE= (01+10)* As q0 is final state.

Arden's Theorem Solved Examples

Example 4: Consider the following transition diagram and prove that the string recognized are (0 + 0(1+00)*1*) 0 (1+00)* 1 using Arden's theorem.



Solution: The Set of equations for regular sets (Incoming Transition to every state (q1, q2 & q3)) represented as follow.

$q_1 = q_1 0 + q_2 1 + \varepsilon$ ....(1)

$q_2 = q_1 0 + q_2 1 + q_3 0$ ....(2)

$q_3 = q_2 1$ ....(3)

Arden's Theorem Solved Examples

Example 4:

Now substitute value of q3 in eq. 2

$$q_2 = q_1 0 + q_2 1 + q_2 10$$

$$q_2 = q_1 0 + q_2 (1+10)$$

Now compare above equation with first equation of Arden's theorem. It is of the form R=Q+RP. so it's solution is given by R=QP*.

$$q_2 = q_1 0 (1+10)^* \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

So now substitute the value of f q2 from eq. 4in eq. 1

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

$$= \varepsilon + q1 (0+0(1+10)^* 1)$$

Now compare above equation with first equation of Arden's theorem. It is of the form R=Q+RP. so it's solution is given by R=QP*.

Arden's Theorem Solved Examples

Example 4:

So $q_1 = \varepsilon . (0+0(1+10)*1)*$

$= (0+0(1+10)*1)*$ ....(5)

Now put value of q1 from eq. 5 in eq. 4

$q_2 = (0+0(1+10)*1)* \, 0 \, (1+10)*$ ....(6)

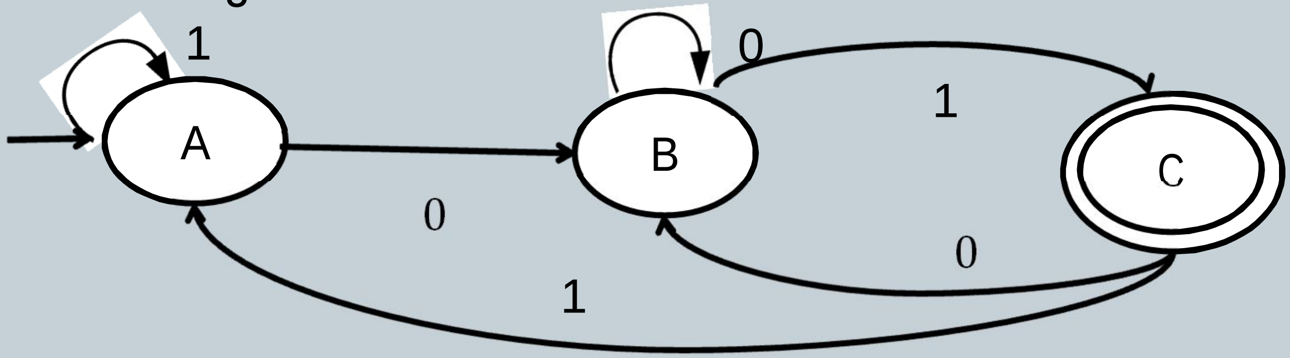
Now put value of q2 from eq. 6 in eq. 3

$q_3 = (0+0(1+10)*1)* \, 0 \, (1+10)* \, 1$

Hence Proved.

Arden's Theorem Solved Examples

Example 5: Consider the following transition diagram convert it to an equivalent RE using Arden's theorem.



Solution: The Set of equations for regular sets (Incoming Transition to every state (A, B & C)) represented as follow.

A= A1+C1 + ε ....(1)

B= A0 + B1 + C0 ....(2)

C= B1 ....(3)

Arden's Theorem Solved Examples

Example 5:

Now substitute value of q3 in eq. 2

$$B = A0 + B1 + B1\,0$$

$$= A0 + B(1 + 1\,0)$$

Now compare above equation with first equation of Arden's theorem. It is of the form R=Q+RP. so it's solution is given by R=QP*.

$$B = A0\,(1 + 1\,0)^* \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

So substitute the value of C in eq. 1

$$A = A1 + B11 + \varepsilon \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$$

Now substitute the value of b from eq. 4 in eq. 5

$$A = A1 + A0\,(1 + 1\,0)^*\,11 + \varepsilon$$

$$= \varepsilon + A(1+0(1+10)^*11)$$

Now compare above equation with first equation of Arden's theorem. It is of the form R=Q+RP. so it's solution is given by R=QP*.

$$A = (1+0(1+10)^*11)^*$$

Arden's Theorem Solved Examples

Example 5:

Hence B from Eq. 4 is

$$B=(1+0(1+10)^{*}11)^{*}\,0\,(1+1\,0)^{*}$$

Hence C from Eq. 3 is

$$C=(1+0(1+10)^{*}11)^{*}\,0\,(1+1\,0)^{*}\,1$$

Limitations of Finite Automata

Pumping Lemma

- Pumping Lemma is used as a proof for irregularity of a language.
- Thus, if a language is regular, it always satisfies pumping lemma.
- If there exists at least one string made from pumping which is not in L, then L is surely not regular.
- The opposite of this may not always be true.
- That is, if Pumping Lemma holds, it does not mean that the language is regular.

Pumping Lemma

If A is a regular language, then there is a number p (the pumping length), where, if x is any string in A of length at least p, then s may be divided into three pieces, s=xyz, satisfying the following conditions: 2

1. For each i ≥ 0, xyi z ∈ A,
2. 2. y≠ ϵ, and
3. 3. |xy|≤p

Pumping Lemma

The Pumping Lemma says that is a language A is regular, then any string in the language will have a certain property, provided that it is ‘long enough’ (that is, longer than some length p, which is the pumping length). Inside any string in A that’s longer than p, we can find a piece that can be repeated (pumped) as many times as we want, and the result will always be in A. Moreover, this piece can be found within the first p letter of our string. That is, given any string s in A longer than p, we can find a substring in s that can be pumped. We’ll call this substring y. Then anything before y we’ll call x, and anything after y we’ll call z. Then the whole string can be rewritten as x – y – z. (Remember that these are strings and not letters!) By repeating y zero or more times, we get: Xz, xyz, xyyz, xyyyz, ..., xyyyyyyyyyyyyyyyyyz, ... What the Pumping Lemma says is that each of these must be in A.

Pumping Lemma

Condition 1: For each $i \geq 0$, $xy^i z \in A$ xy^2z is the same as xyyz, etc. So this says that sticking in multiple copies of y will give you strings that are still in the language. For I=0, you get no copies of y, i.e., the string xz.

Pumping Lemma

Condition 2: $y \neq \epsilon$ While x or z may have length zero, the length of y is not zero. That is, y is not the empty string. If you allowed y to be the empty string, the theorem would be trivially true. This is because if y was the empty string, you would end up with xz, which is just s, the original string you started with, no matter how many times you pumped y.

Pumping Lemma

Condition 3: $|xy| \le p$ Since x is the piece before y, this says that all of y must come from the first p letters of our string s, so that the combined length of x and y is at most p.

How to use the Pumping Lemma to prove that a language is not regular?

The pumping lemma is most useful when we want to prove that a language is not regular, We do this by using a proof by contradiction. To prove that a given language L is not regular:

1. Assume that L is regular.
2. 2. Use the pumping lemma to guarantee the existence of a pumping length p such that all strings of length p or greater in L can be pumped.
3. 3. Find a string s in L that has length p or greater but that cannot be pumped.
4. 4. Demonstrate that s cannot be pumped by considering all ways of dividing s into x, y, and z, and for each division, finding a value I where xyi z∉L Ö The existence of s contradicts the pumping lemma if L were regular. Hence L cannot be regular.

Pumping Lemma

- Pumping Lemma is used as a proof for irregularity of a language.
- Thus, if a language is regular, it always satisfies pumping lemma.
- If there exists at least one string made from pumping which is not in L, then L is surely not regular.
- The opposite of this may not always be true.
- That is, if Pumping Lemma holds, it does not mean that the language is regular.

How to use the Pumping Lemma to prove that a language is not regular?

Let L be a regular language and M=(Q, ∑, δ, q_0, F) be FA with n states. Language L is accepted by M. Let some string w ∈ L and |w| >=n then w can be written in the form of xyz such that

i. |y|>0
ii. |xy|<=n
iii. $xy^i z \in L$ for all i>0 here y^i denotes that y is repeated or pumped i times where i>0.

Pumping Lemma Examples

Example 1: Using Pumping Lemma Show that the language $L=\{a^n b^{2n}\}$ is not regular.

Solution:

Step 1: Let us assume that L is a regular and L is accepted by FA with n states.

Step 2: Let us choose a string represented by given language L and accepted by FA.

According to language no of occurrences of a should be followed by double no. of occurrences of b. So let us consider the string |w|

$L = a^n b^{2n}$ $\quad\quad\quad$ $|\,w\,| = aabbbb$

Now let us write w as xyz with $|y|>0$ and $|xy| <= n$

$$\underbrace{aa}_{x}\quad\underbrace{bb}_{y}\quad\underbrace{bb}_{z}$$

Pumping Lemma Examples

Example 1:

Now let us pump the y i times write i >0

Let us consider i=2

$$\underbrace{aa}_{x}\quad \underbrace{bbbb}_{y^i}\quad \underbrace{bb}_{z}$$

Now let us check $xy^2 z$ belongs to L or not.

$xy^2 z$ = aabbbbbb

So as per given language no of occurrences of a should be followed by double no. of occurrences of b is not followed by string after pumping so which indicates $|w| \notin L$, and hence the given language is not regular.

Pumping Lemma Examples

Example 2: Using Pumping Lemma Show that the language $L=\{a^n b^n\}$ is not regular.

Solution:

Step 1: Let us assume that L is a regular and L is accepted by FA with n states.

Step 2: Step 2: Let us choose a string represented by given language L and accepted by FA.

According to language no of occurrences of a should be followed by equal no. of occurrences of b. So let us consider the string |w|

$L= a^n b^n$

$| w |= aaabbb$

Now let us write w as xyz with |y|>0 and |xy| <= n

$$\underbrace{aa}_{x}\quad \underbrace{abb}_{y}\quad \underbrace{b}_{z}$$

Pumping Lemma Examples

Example 2:

Now let us pump the y i times write i >0

Let us consider i=2

$$\underbrace{aa}_{x}\quad \underbrace{abbabb}_{y^i}\quad \underbrace{bb}_{z}$$

Now let us check $xy^2 z$ belongs to L or not.

$xy^2 z$ = aaabbabbbb

So as per given language no of occurrences of a should be followed by same no. of occurrences of b is not followed by string after pumping so which indicates |w| $\notin$ L, and hence the given language is not regular.

Pumping Lemma Examples

Example 3: Show that the language L={0^m1^n 0^{m+n} m>=1 and n>=1} is not regular.

Solution:

Step 1: Let us assume that L is a regular and L is accepted by FA with n states.

Step 2: Step 2: Let us choose a string represented by given language L and accepted by FA. According to language no of m times occurrences of 0 should be followed by n times occurrences of 1 followed by m+n times occurrences of 0 should be followed by n times occurrences of 1 followed by m+n times occurrences of 0 . So let us consider the string |w|

L= 0^m1^n 0^{m+n} | w |= 00110000

Now let us write w as xyz with |y|>0 and |xy| <= n

$$\underbrace{001}_{x}\ \underbrace{10}_{y}\ \underbrace{000}_{z}$$

Pumping Lemma Examples

Example 3:

Now let us pump the y i times write i >0

Let us consider i=2

$$\underbrace{001}_{x}\ \underbrace{0101}_{y^i}\ \underbrace{000}_{z}$$

Now let us check xy^2z belongs to L or not.

$xy^2z = 0010101000$

So as per given language m times occurrences of 0 should be followed by n times occurrences of 1 followed by m+n times occurrences of 0 should be followed by n times occurrences of 1 followed by m+n times occurrences of 0 . is not followed by string after pumping so which indicates $|w| \notin L$, and hence the given language is not regular.

Example 4: Using Pumping Lemma Show that the given language is not regular. $L=\{ww^R \mid w \in \{0, 1\}^*\}$

Solution:

Step 1: Let us assume that L is a regular and L is accepted by FA with n states.

Step 2: Let us choose a string represented by given language L and accepted by FA.

According to language is set of string of {0 & 1} such that string consist of string followed by reverse of itself . So let us consider the string |w|

$$|w| = 1001$$

Now let us write w as xyz with |y|>0 and |xy| <= n

$$\underbrace{10}_{x} \quad \underbrace{0}_{y} \quad \underbrace{1}_{z}$$

Pumping Lemma Examples

Example 4:

Now let us pump the y i times write i >0

Let us consider i=2

$$\underbrace{10}_{x}\ \underbrace{00}_{y^i}\ \underbrace{1}_{z}$$

Now let us check $xy^2 z$ belongs to L or not.

$xy^2 z = 10001$

So as per given is set of string of {0 & 1} such that string consist of string followed by reverse of itself . So is not followed by string after pumping so which indicates |w| $\notin$ L, and hence the given language is not regular.

Pumping Lemma Examples

Example 5: Using Pumping Lemma Show that the given language is not regular. $L=\{ww \mid w \in \{0, 1\}^*\}$

Solution:

Step 1: Let us assume that L is a regular and L is accepted by FA with n states.

Step 2: Step 2: Let us choose a string represented by given language L and accepted by FA.

According to language is set of string of {0 & 1} such that string consist of string followed by itself . So let us consider the string |w|

$$|w| = 1010$$

Now let us write w as xyz with |y|>0 and |xy| <= n

$$\underbrace{1}_{x} \quad \underbrace{01}_{y} \quad \underbrace{0}_{z}$$

Pumping Lemma Examples

Example 5:

Now let us pump the y i times write i >0

Let us consider i=2

$$\underbrace{1}_{x}\quad\underbrace{0101}_{y^i}\quad\underbrace{0}_{z}$$

Now let us check xy^2z belongs to L or not.

$xy^2z = 101010$

So as per given is set of string of {0 & 1} such that string consist of string followed by itself . So is not followed by string after pumping so which indicates $|w| \notin L$, and hence the given language is not regular.

Closure Properties of Regular Language

- **Closure properties** on regular languages are defined as certain operations on regular language which are guaranteed to produce regular language.
- Regular languages are closed under following operations.

1. **Union**
2. **Intersection**
3. **Difference**
4. **Concatenation**
5. **Kleene Closure**
6. **Reversal**
7. **Complementation**

Closure Properties of Regular Language

1. **Regular languages Closed under Union:**

- **Let** M1=(Q1 ,Σ, δ1 , q0 , F1) and M2=(Q2 ,Σ, δ2 , S0 , F2) are given FA and are closes Under union if another FA M3 which acceptes all string generated by M1 & M2.
- **Machine M3 is constructed to accept L (M1) U L (M2) then**
- M3=(Q3 ,Σ, δ3, R0 , F3) where
  - Q3= Q1 U Q2 U {R0} R0 new initial state generated.
  - F3= F1 U F2
  - δ3 = δ1 U δ2

Closure Properties of Regular Language

2. Regular languages Closed under Complementation :

- **Let** M=(Q1 ,Σ, δ , q0 , F) is a given FA. Then The set of regular languages is closed under complementation.
- The complement of language L, written L, is all strings not in L but with the same alphabet. The statement says that if L is a regular language, then so is L

Closure Properties of Regular Language

3. Regular languages Closed under Difference :

- If L and M are regular languages, then the difference L – M is set of strings in L but not in M.

Closure Properties of Regular Language

4. Regular languages Closed under Intersection :

- Let L and M be the languages of regular expressions R and S, respectively then it a regular expression whose language is L intersection M.

Closure Properties of Regular Language

5. Regular languages Closed under Reversal:

- Given language L which is regular language the reverse of it is represented by L^R is the set of strings whose reversal is in L and is also regular.
- Example: L = {0, 01, 100}; L^R = {0, 10, 001}.

Closure Properties of Regular Language

6. Regular languages Closed under Kleen Star:

- **Let** M1=($Q1$,Σ, $\delta1$, $q0$, $F1$) is given FA representing regular language. So we can construct FA M2 such that L (M2) =L(M1)* which is also represent regular language.
- R is a regular expression whose language is L, and automata M then R* is a regular expression whose language is L*.

Closure Properties of Regular Language

7. Regular languages Closed under Concatenation:

- **Let** M1=(Q1 ,Σ, δ1 , q0 , F1) and M2=(Q2 ,Σ, δ2 , S0 , F2) are given FA and are closes Under concatenation if another FA M3 such that
- **L (M3) =L (M1) . L (M2)**.
- **Machine M3 is constructed to accept L (M1) . L (M2) then**
- M3=(Q3 ,Σ, δ3, R0 , F3)

Applications of Regular Expressions

- Followings are the applications of RE.

1. **Regular Expressions in Unix:** The grep utility in Unix scans a file for occurrences of patters and displays those lines in which the given patters are found.
2. **Regular Expressions in Lexical Analysis:** Lexical Analysis is a important phase of compiler . It scans the source program and convert it into a steam of tokens. A token is a string of consecutive symbol defining an entity.

Thank You