23MT2014

THEORY OF COMPUTATION

Topic:

RE ↔ NFA ↔ DFA

Session - 7

AIM OF THE SESSION

To familiarize students with the basic concept of Regular Expressions

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Define Regular Expression
2. Demonstrate the construction of RE for the given languages
3. Describe properties and Identity rules of RE
4. Demonstrate the equivalence of RE and Finite Automata (FA)

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Define Regular Expression
2. Construct Regular Expression for given language
3. Convert between RE and FA

REGULAR EXPRESSION

- **Regular expressions describe regular languages**
- **These are an algebraic representations of languages**
- **They have applications in**
  - **Text editors: search a word using RE (st*)**
  - **Compiler Design: Comments (*/ … */)**

REGULAR EXPRESSION - definition

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

ϕ is a regular expression and denotes empty set

ε is a regular expression and denotes the set $\{\varepsilon\}$

For each 'a' in $\sum$, 'a' is a regular expression and denotes the set {a}

If 'r' and 's' are regular expressions denoting the languages R and S respectively, then the regular expression

r+s denotes R U S

rs denotes RS

r* denotes R*

REGULAR EXPRESSION - operations

Given regular expressions r_1 and r_2

Union	$r_1 + r_2$
Concatenation	$r_1 \cdot r_2$
Star Closure	$r_1 *$
	(r_1)

Are regular expressions

REGULAR EXPRESSION - example

A regular expression: $(a+b\cdot c)^*\cdot(c+\varnothing)$

Not a regular expression: $(a+b+)$

REGULAR EXPRESSION & Language

Regular Expression	Language
0*	Set of all strings of the form 0^n (n>=0) L = {ε, 0, 00, 000, 0000, 00000, . . . }
1+0	Set of 0 and 1 L = { 0, 1 }
01	L = { 01 }
(01)*	Set of all strings containing n '01's (n >= 0) L = {ε , 01, 0101, 010101, 01010101, . . . }
(1+0)*	Set of all strings containing 1 and 0 L = {ε, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, …. }
01*	Set of all strings start with 0 followed by n 1s (n>=0) L = { 0, 01, 011, 0111, 01111, 011111, … }
(0+1)00	L = {000, 100}

Language & REGULAR EXPRESSION

- **All strings of 0's and 1's starting with 0 and ending with 1**
- **0(0+1)*1**
- **All strings of 0's and 1's with even number of 0's**
- **1 * (01*01*) ***
- **All strings of 0's and 1's with at least two consecutive 0's**
- **(0+1)*00 (0+1)***
- **All strings of 0's and 1's without two consecutive 0's**
- **((1+01) * (ε+0))**

Equivalence of Regular Expressions

- **Regular expressions r_1 and r_2**
- **are equivalent if**

$$L(r_1) = L(r_2)$$

Equivalence of Regular Expressions - example

L = { all strings without two consecutive 0 }

$$r_1 = (1+01)^*(0+\varepsilon)$$

$L(r_1)$ = { ε, 0, 1, 10, 01, 11, 110, 0101,…. }

$$r_2 = (1^*011^*)^*(0+\varepsilon)+1^*(0+\varepsilon)$$

$L(r_2)$ = { ε, 0, 1, 10, 01, 11, 110, 0101,…. }

$$L(r_1) = L(r_2) = L$$ ⇒ r_1 and r_2 are equivalent regular expr.

Identity Rules for Regular Expression

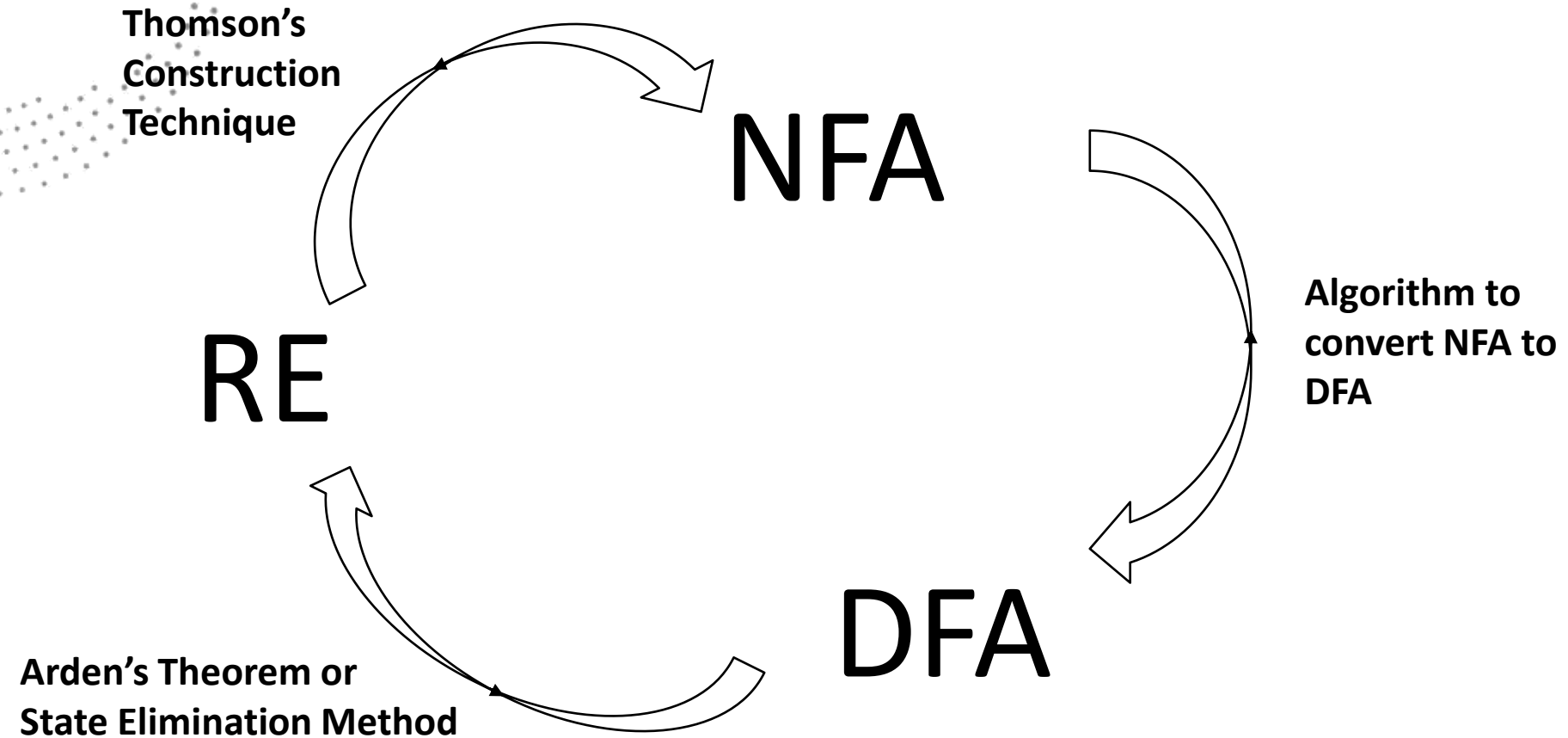
The two regular expression's P and Q are equivalent (denoted as P=Q) if and only if P represents the same set of strings as Q does.

For showing the equivalence of two regular expressions we need to show some identities of regular expression's

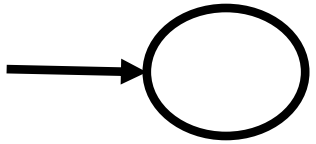
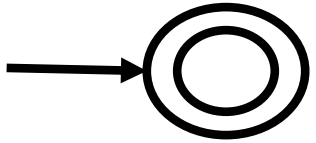
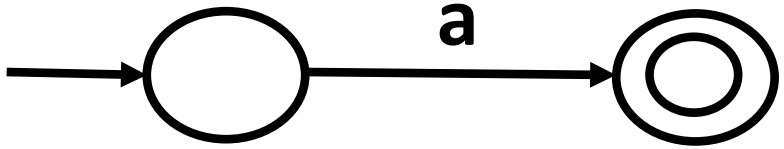
Identity Rules for Regular Expression

1. $\varepsilon R = R\ \varepsilon = R$
2. $\varepsilon^* = \varepsilon$
3. $(\Phi)^* = \varepsilon$
4. $\Phi R = R\ \Phi = \Phi$
5. $\Phi + R = R$
6. $R + R = R$
7. $RR^* = R^*R = R+$
8. $(R^*)^* = R^*$
9. $E + RR^* = R^*$
10. $(P+Q)R = PR + QR$
11. $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$
12. $R^*(\varepsilon + R) = (\varepsilon + R)R^* = R^*$
13. $(R+\varepsilon)^* = R^*$
14. $E + R^* = R^*$
15. $(PQ)^*P = P(QP)^*$
16. $R^*R + R = R^*R$

Equivalence of Regular Expression & FA

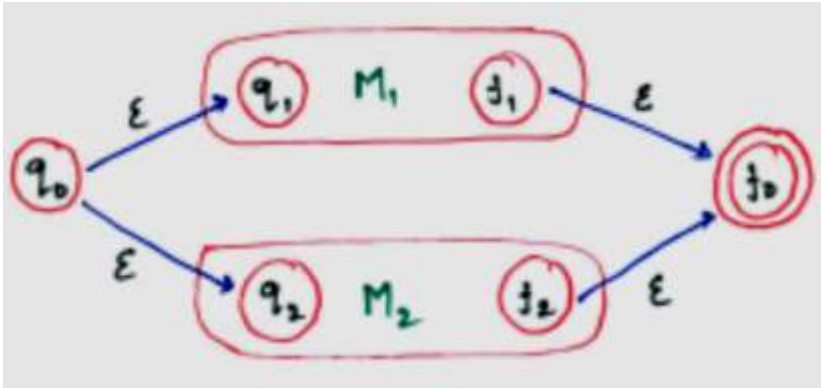


- **Thomson's Construction Technique**

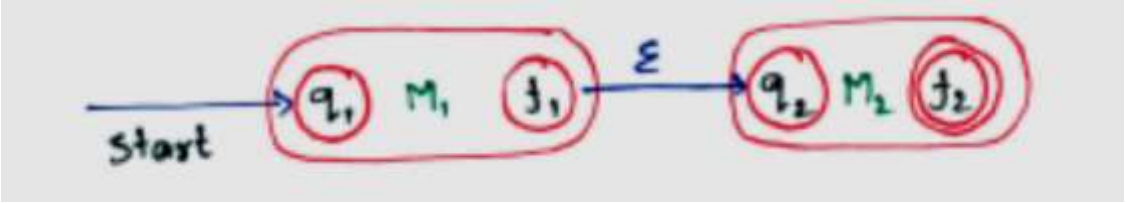
RE	NFA
Φ	
λ	
a	

RE to FA Conversion

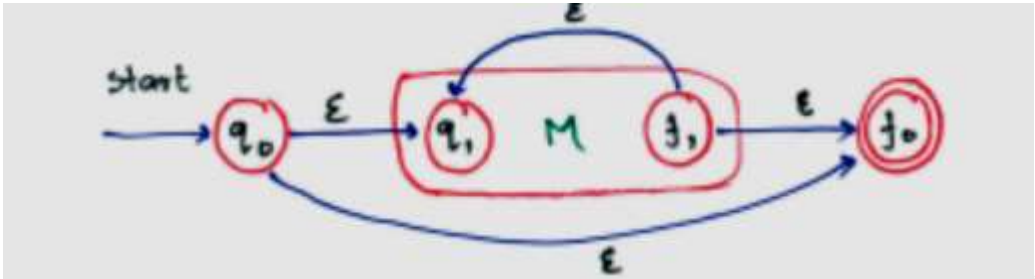
- **Case 1 : Construction of $L_1 + L_2$ (Union)**



- **Case 2 : Construction of L_1L_2 (Concatenation)**



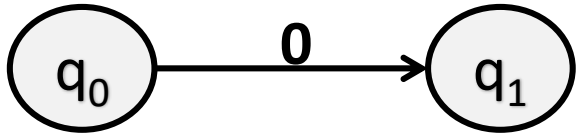
- **Case 3 : Construction of L_1* (Star closure)**



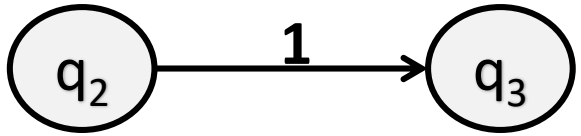
RE to FA Conversion - Example

RE → 01

$r_1 = 0$



$r_2 = 1$

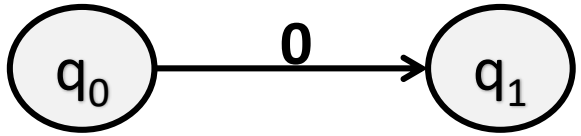


$r_3 = r_1 . r_2$

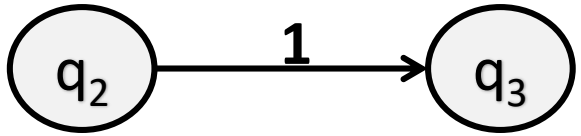
RE to FA Conversion - Example

RE → 01*

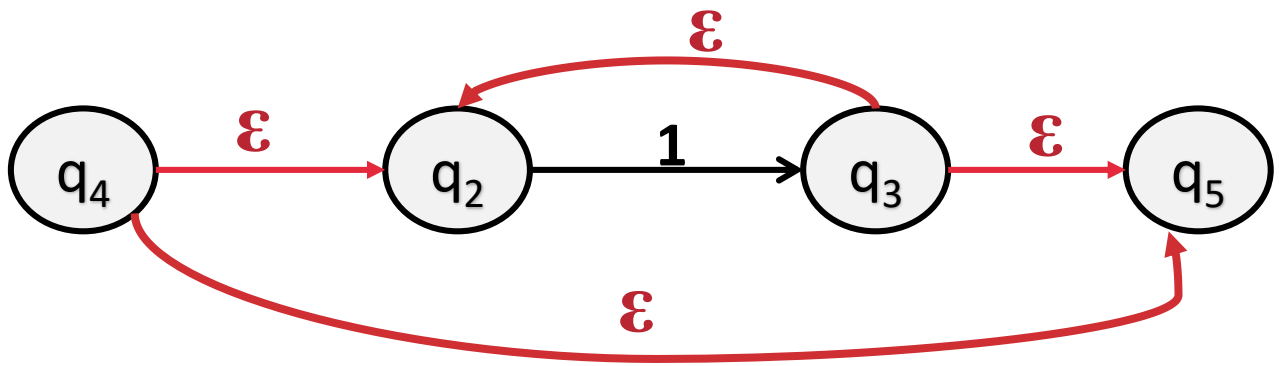
$r_1 = 0$



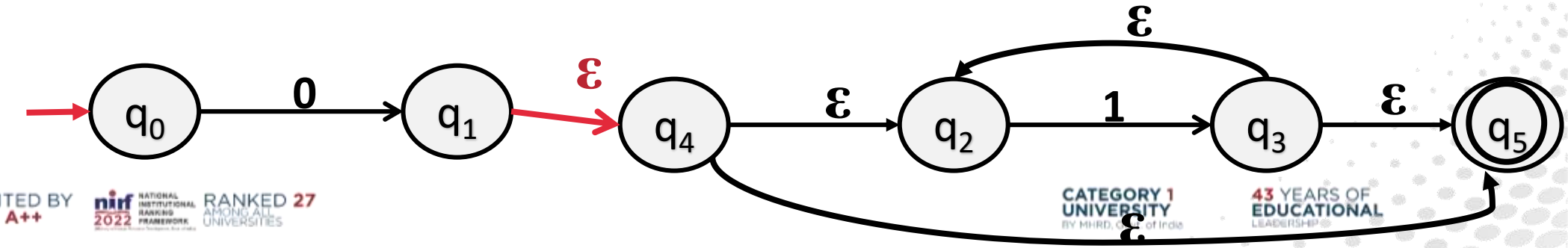
$r_2 = 1$



$r_3 = r_2^*$



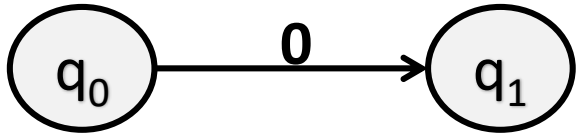
$r_4 = r_1 . r_3$



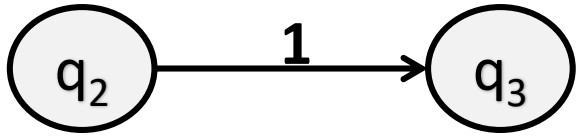
RE to FA Conversion - Example

RE → 0 + 1

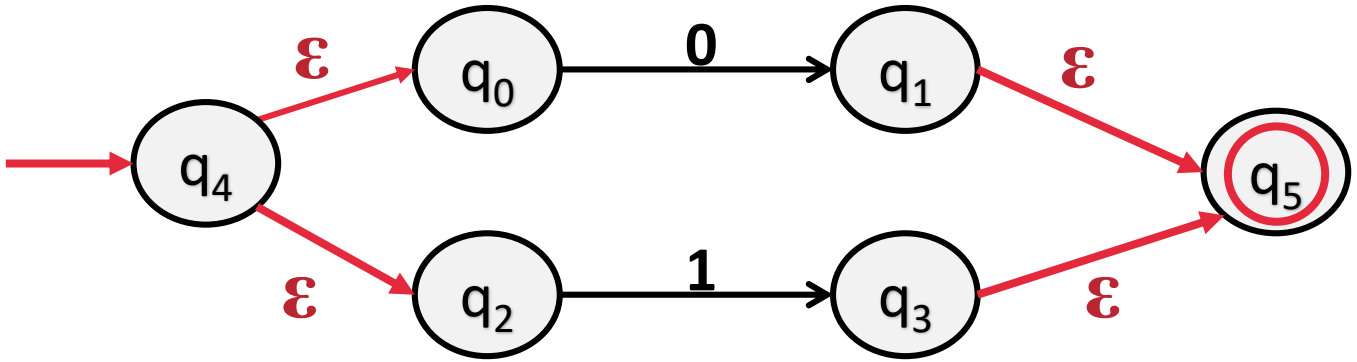
$r_1 = 0$



$r_2 = 1$



$r_3 = r_1 + r_2$



RE to FA Conversion – More Example

1*0 + 0

(0+1)*

0*1*

(01)*(10)*+00*

Two Methods :

- **Arden's theorem**
- **State Elimination Method**

Arden's Theorem:

- Let P and Q be two regular expressions over $\sum$.

  If P does not contain λ, then the following equation in R,

$$R = Q + RP$$

has a unique solution given by

$$R = QP^*$$

Conditions-

The transition diagram must not have any ∈ transitions.

There must be only a single initial state.

Step-01:

Form equation for each state considering the transitions which comes towards that state.

Add '∈' in the equation of initial state.

Step-02:

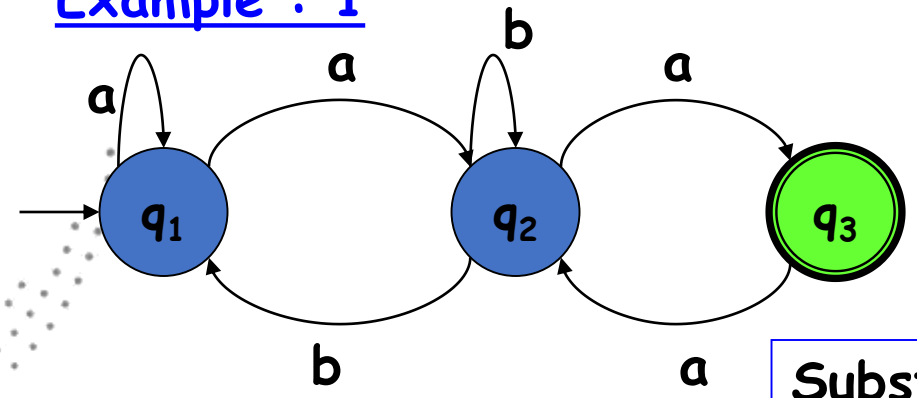
Bring final state in the form R = Q + RP to get the

Arden's Theorem can be used to find a regular expression for both DFA and NFA.

If there exists multiple final states, then-

- **Write a regular expression for each final state separately.**
- **Add all the regular expressions to get the final regular expression.**

Example : 1



Initial Equations

$q_1 = q_1a + q_2b + \lambda$

$q_2 = q_1a + q_2b + q_3a$

$q_3 = q_2a$

Substituting the value of q_3 in q_2 we get:

$q_2 = q_1a + q_2b + q_2aa$

$q_2 = q_1a + q_2(b + aa)$

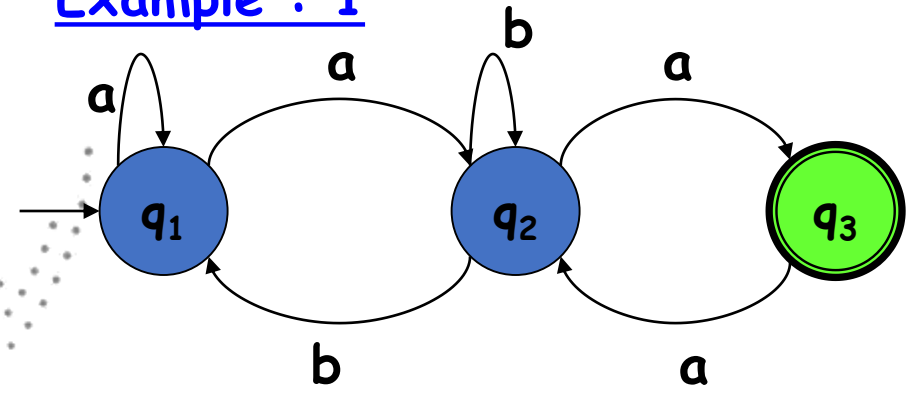
By Arden's Theorem

$R = Q + RP$ has a solution $R = QP^*$

Therefore

$q_2 = q_1a(b + aa)^*$

Example : 1



Substituting the value of q_2 in q_1 we get:

$q_1 = q_1a + q1a(b+aa)^* b + \lambda$

$q1 = q1\ (a + a(b+aa)^*b) + \lambda$

By Arden's Theorem

$R = Q + RP$ has a solution $R = QP^*$

Therefore

$q_1 = \lambda\ (a + a(b+aa)^*b\ )^*$

$q1 = (a + a(b+aa)^*b\ )^*$

Substituting the values

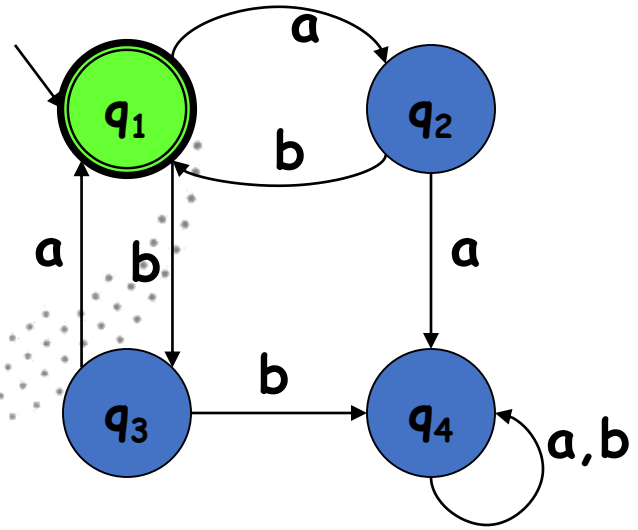
q2 = (a+a(b+aa)*b)*a(b+aa)*

q3 = (a+a(b+aa)*b)*a(b+aa)*a

So the Resultant regular expression that describes the language accepted by the given FA is:

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

Example : 2



Initial Equations

$q_1 = q_2b + q_3a + \lambda$

$q_2 = q_1a$

$q_3 = q_1b$

$q_4 = q_2a + q_3b + q_4a + q_4b$

Since q1 is the only final state and it involves only q2 and q3 in its equation, the equation q4 is redundant for us.

So we will consider only q2 and q3 to solve q1.

$q_1 = q_1ab + q_1ba + \lambda$

$= q_1(ab + ba) + \lambda$

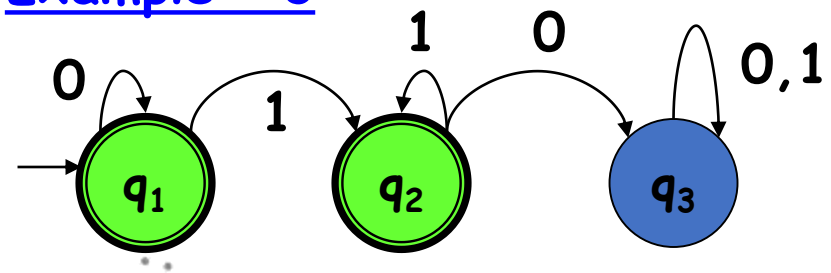
By Arden's Theorem

$q_1 = \lambda\ (ab + ba)^*$

So the Resultant Regular Exp is:

$(ab + ba)^*$

Example : 3



Initial Equations

$q_1 = q_1 0 + \lambda$

$q_2 = q_1 1 + q_2 1$

$q_3 = q_2 0 + q_3(0 + 1)$

$q_1 = \lambda\, 0^* = 0^*$

$q_2 = q_1\, 1 + q_2\, 1$

$q_2 = 0^*\, 1 + q_2\, 1$

$q2 = (0^*1)1^*$

As the Final States are only q_1 and q_2, we need not solve for q_3

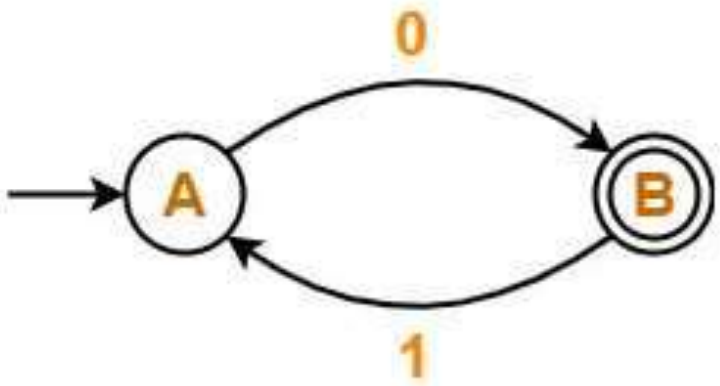
Therefore the Regular Expression will be:

$q1 + q2 = 0^* + 0^*(11^*)$

$= 0^* (\lambda + 11^*)$

$= 0^*1^*$ (by theorem)

Arden's Theorem – Example 1



Step-01:

Form a equation for each state-

- A = ∈ + B.1 .....(1)
- B = A.0 .....(2)

Step-02:

Bring final state in the form R = Q + RP.

Using (1) in (2), we get-

B = (∈ + B.1).0

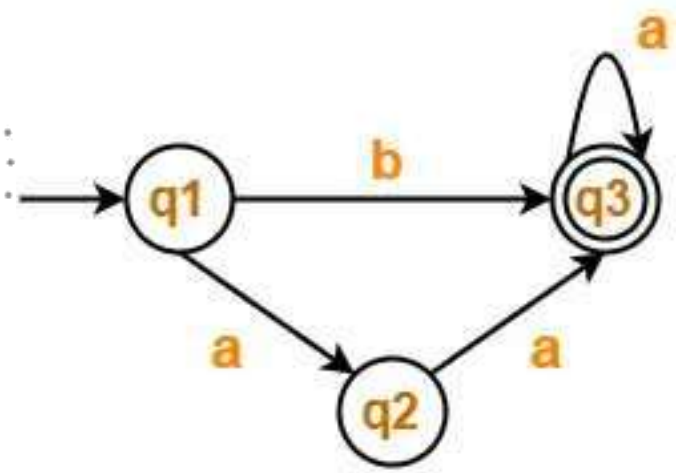
B = ∈.0 + B.1.0

B = 0 + B.(1.0) .....(3)

Using Arden's Theorem in (3), we get-

B = 0.(1.0)*

Arden's Theorem – Example 2



Step-01:

Form a equation for each state-

- $q1 = \epsilon$ .....(1)
- $q2 = q1.a$ .....(2)
- $q3 = q1.b + q2.a + q3.a$ .....(3)

Step-02:

Bring final state in the form R = Q + RP.

Using (1) in (2), we get-

$q2 = \epsilon.a$

$q2 = a$ .....(4)

Using (1) and (4) in (3), we get-

$q3 = q1.b + q2.a + q3.a$

$q3 = \epsilon.b + a.a + q3.a$

$q3 = (b + a.a) + q3.a$ .....(5)

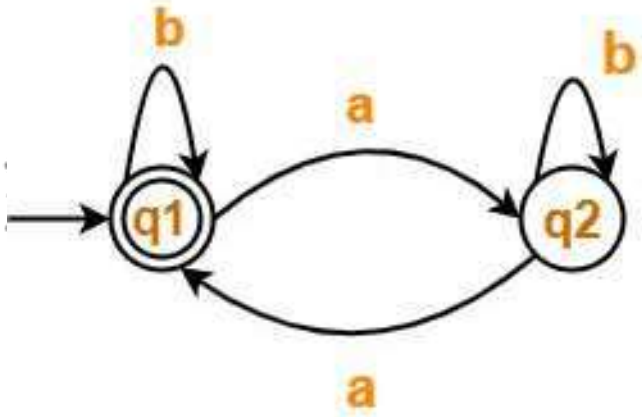
Arden's Theorem – Example 2

Using Arden's Theorem in (5), we get-

$q3 = (b + a.a)a^*$

Thus, Regular Expression for the given DFA = $(b + aa)a^*$

Arden's Theorem – Example 3



Step-01:

Form a equation for each state-

- $q1 = \epsilon + q1.b + q2.a$ .....(1)
- $q2 = q1.a + q2.b$ .....(2)

Thus, Regular Expression for the given DFA = $(b + a.b^*.a)^*$

Step-02:

Bring final state in the form R = Q + RP.

Using Arden's Theorem in (2), we get-

$q2 = q1.a.b^*$ .....(3)

Using (3) in (1), we get-

$q1 = \epsilon + q1.b + q1.a.b^*.a$

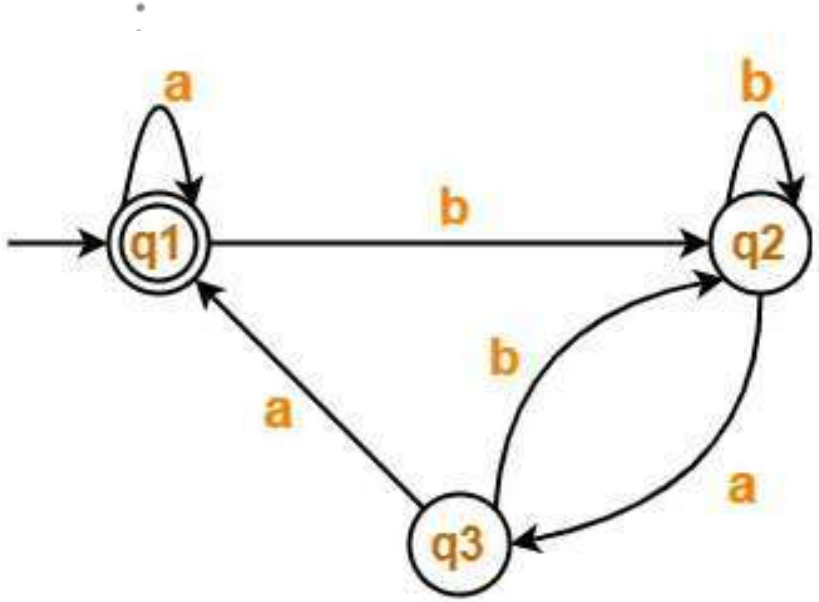
$q1 = \epsilon + q1.(b + a.b^*.a)$ .....(4)

Using Arden's Theorem in (4), we get-

$q1 = \epsilon.(b + a.b^*.a)^*$

$q1 = (b + a.b^*.a)^*$

Arden's Theorem – Example 4



Step-01:

Form a equation for each state-

- q1 = ∈ + q1.a + q3.a ……(1)
- q2 = q1.b + q2.b + q3.b ……(2)
- q3 = q2.a ……(3)

Step-02:

Bring final state in the form R = Q + RP.

Using (3) in (2), we get-

q2 = q1.b + q2.b + q2.a.b

q2 = q1.b + q2.(b + a.b) ……(4)

Using Arden's Theorem in (4), we get-

q2 = q1.b.(b + a.b)* ……(5)

Using (5) in (3), we get-

q3 = q1.b.(b + a.b)*.a ……(6)

Arden's Theorem – Example 4

Using (6) in (1), we get-

$q1 = \epsilon + q1.a + q1.b.(b + a.b)^*.a.a$

$q1 = \epsilon + q1.(a + b.(b + a.b)^*.a.a)$ .....(7)

Using Arden's Theorem in (7), we get-

$q1 = \epsilon.(a + b.(b + a.b)^*.a.a)^*$

$q1 = (a + b.(b + a.b)^*.a.a)^*$

Thus, Regular Expression for the given DFA = $(a + b(b + ab)^*aa)^*$

SUMMARY

Regular Expressions represent the Regular Languages.
RE can be converted to Finite Automata using Thomson's Method
FA can be converted to RE using Arden's Method

SELF-ASSESSMENT QUESTIONS

1. Which of the following does not represents the given language?
   Language: {0,01}

(a) 0+01

(b) {0} U {01}

(c) {0} U {0}{1}

(d) {0} ^ {01}

2. Which among the following are incorrect regular identities?

(a) εR=R

(b) ε*=ε

(c) Φ*=ε

(d) RΦ=R

SELF-ASSESSMENT QUESTIONS

3. Conversion of a regular expression into its corresponding NFA :

(a) Thomson's Construction Algorithm
(b) Powerset Construction
(c) Kleene's algorithm
(d) None of the mentioned

4. The appropriate precedence order of operations over a Regular Language is

(a) Kleene, Union, Concatenate
(b) Kleene, Star, Union
(c) Kleene, Dot, Union
(d) Star, Union, Dot

TERMINAL QUESTIONS

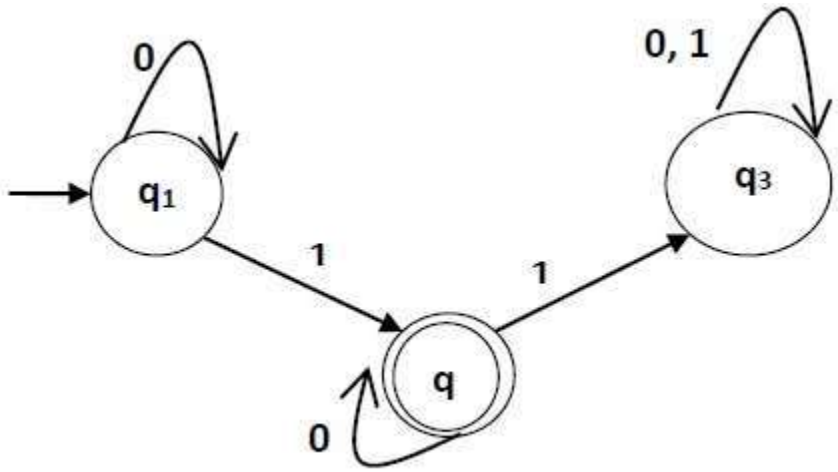
1. Construct a regular expression for the language L ={ w | w Є{0,1}* and 'w' does not contain two consecutives 1} (4.5 marks)

2. Construct Finite Automata for the following regular expressions.

   (a) (01)*(10)*+00* (4marks)

   (b) (1+0)* 11 + 101* (4 marks)

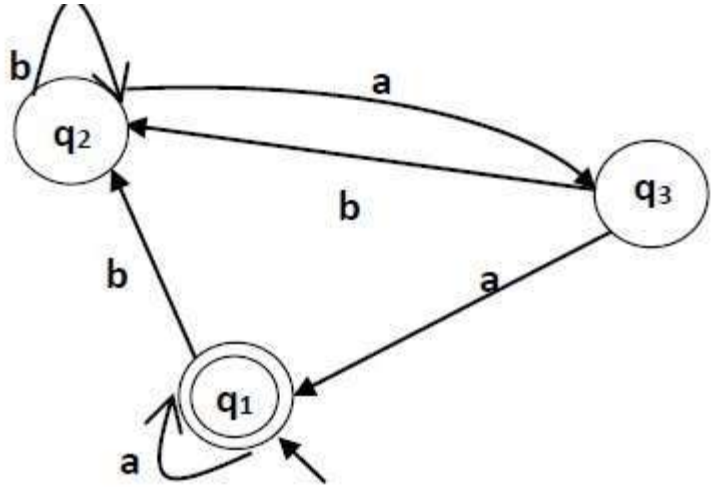
3. Construct regular expression for the following FA (12.5 marks)



TERMINAL QUESTIONS

4. (a) List out the identity rules of Regular Expression (5 marks)

(b) Convert the following FA to RE (10 marks)



MCQ

Question 1:

Which symbol is used to represent the Kleene star operation in regular expressions?

a) +

b) *

c) ?

d) .

Answer: b) *

Question 2:

What does the "+" operator represent in regular expressions?

a) Matches the preceding character or subexpression zero or one time.

b) Matches the preceding character or subexpression one or more times.

c) Matches the preceding character or subexpression zero or more times.

d) Matches the preceding character or subexpression exactly once.

Answer: b) Matches the preceding character or subexpression one or more times.

MCQ

Question 3:

Which of the following is the correct regular expression for matching any digit?

a) [0-9]

b) [a-zA-Z]

c) \d

d) \w

Answer: a) [0-9]

Question 4:

What does the "^" symbol represent in regular expressions?

a) Matches the start of a line or string.

b) Matches the end of a line or string.

c) Matches any single character.

d) Matches the preceding character or subexpression zero or one time.

Answer: a) Matches the start of a line or string.

Terminal question

1. What is the purpose of converting a regular expression to an NFA?
2. What are the steps involved in Thomson's construction method for converting a regular expression to an NFA?
3. How does the state reduction method contribute to the conversion of an NFA to a DFA?
4. What are the advantages of using NFAs and DFAs in language recognition?
5. Can any regular expression be converted into an equivalent DFA?
6. How does the conversion of regular expressions to NFAs and DFAs impact language recognition efficiency?
7. Are Thomson's construction method and state reduction method the only techniques for this conversion?
8. Can the resulting DFA after conversion recognize a language that the original regular expression could not?
9. What role do states and transitions play in the NFA and DFA representations?
10. How does the conversion process from regular expression to NFA to DFA contribute to the study of automata theory?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books:

1. John E. Hopcroft, Rajeev Motwani, and Jeffery D. Ullman, “Introduction to Automata Theory, Languages and Computation”, 3rd Edition, Pearson Education, 2008.
2. K. L. P. Mishra and N. Chandrasekaran, “Theory of Computer Science: Automata, Languages, and Computation”, 2nd edition, PHI.
3. Harry R Lewis, Christos H Papadimitriou, “Elements of the theory of computation”, 2nd Edition, PHI/Pearson Education, 1997.

THANK YOU

(DEEMED TO BE UNIVERSITY)

Team - TOC