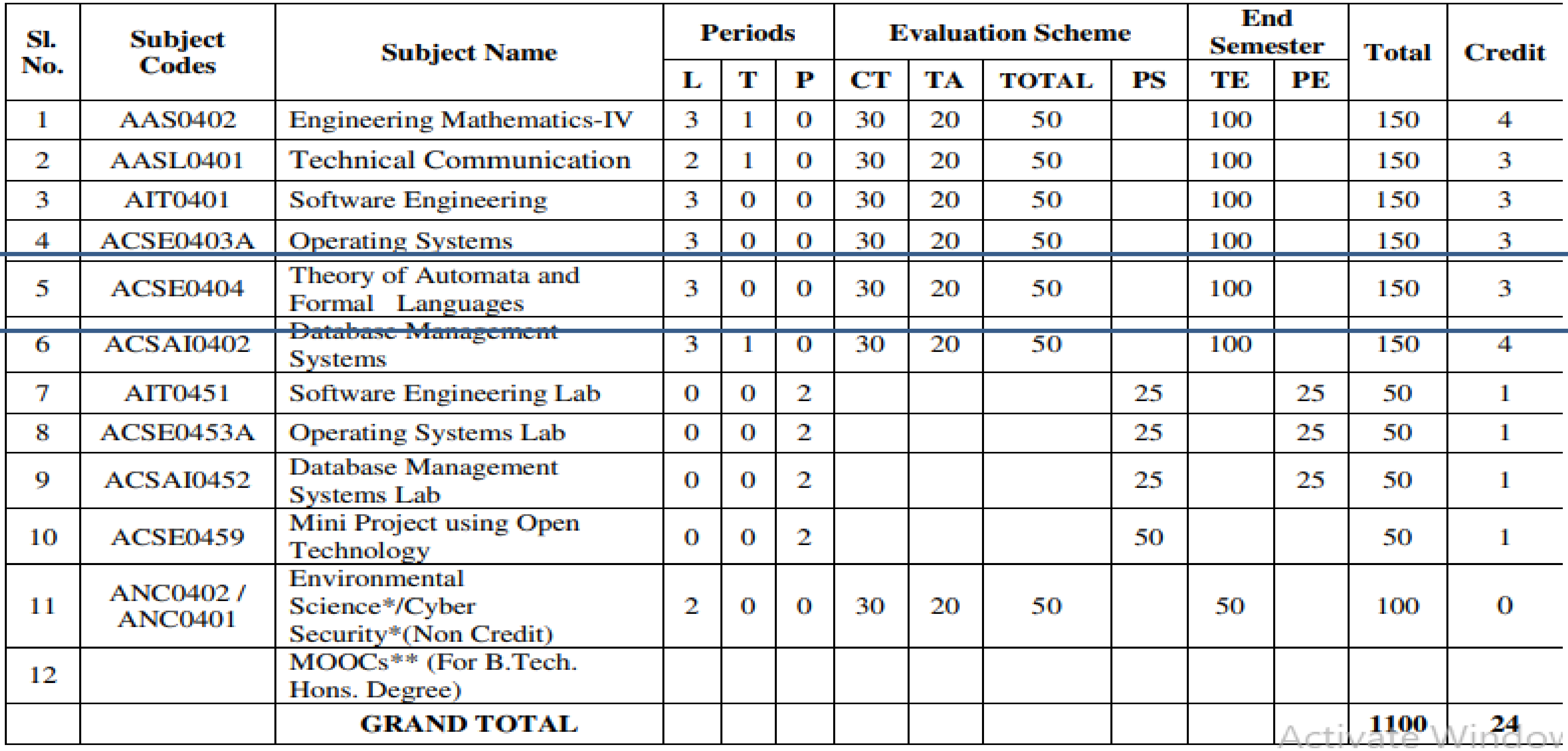
Theory of Automata and Formal Languages (ACSE0404)

Unit: 2

Theory of Automata and Formal Languages

B Tech 4th Sem

Evaluation Scheme



Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Scheme				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
1	AAS0402	Engineering Mathematics-IV	3	1	0	30	20	50		100		150	4
2	AASL0401	Technical Communication	2	1	0	30	20	50		100		150	3
3	AIT0401	Software Engineering	3	0	0	30	20	50		100		150	3
4	ACSE0403A	Operating Systems	3	0	0	30	20	50		100		150	3
5	ACSE0404	Theory of Automata and Formal Languages	3	0	0	30	20	50		100		150	3
6	ACSAI0402	Database Management Systems	3	1	0	30	20	50		100		150	4
7	AIT0451	Software Engineering Lab	0	0	2				25		25	50	1
8	ACSE0453A	Operating Systems Lab	0	0	2				25		25	50	1
9	ACSAI0452	Database Management Systems Lab	0	0	2				25		25	50	1
10	ACSE0459	Mini Project using Open Technology	0	0	2				50			50	1
11	ANC0402 / ANC0401	Environmental Science*/Cyber Security*(Non Credit)	2	0	0	30	20	50		50		100	0
12		MOOCs** (For B.Tech. Hons. Degree)											
		GRAND TOTAL										**1100**	**24**

Subject Syllabus

B. TECH. SECOND YEAR

Course Code	**ACSE0404**	**L T P**	**Credits**
Course Title	**Theory of Automata and Formal Languages**	**3 0 0**	**3**

Course objective:

To teach mathematical foundations o

f computation including automata theory, provide the design concepts of abstract computation model of finite automata, push down automata and Turing Machine and familiarize the notions of algorithm, decidability, complexity, and computability.

Pre-requisites:

- Discrete Mathematics
- Fundamental of Computer System

Course Contents / Syllabus

UNIT-I	**Basic Concepts of Formal Language and Automata Theory**	**8 Hours**

Introduction to Theory of Computation- Alphabet, Symbol, String, Formal Languages, Grammar, Derivation and Language generation by Grammar, Chomsky Hierarchy, Finite Automata, Deterministic Finite Automaton (DFA)- Definition, Representation, Acceptability of a String and Language, Non-Deterministic Finite Automaton (NFA), Equivalence of DFA and NFA, NFA with ∊- Transition, Equivalence of NFA's with and without ∊-Transition, Finite Automata with output- Moore Machine, Mealy Machine, Equivalence of Moore and Mealy Machine, Minimization of Finite Automata, Myhill-Nerode Theorem, Simulation of DFA and NFA.

Subject Syllabus

UNIT II	**Regular Language and Finite Automata**	**8 Hours**
Regular Expressions, Transition Graph, Kleen's Theorem, Finite Automata and Regular Expression- Arden's theorem, Algebraic Method Using Arden's Theorem, Regular Grammars-Right Linear and Left Linear grammars, Conversion of FA into Regular grammar and Regular grammar into FA, Regular and Non-Regular Languages- Closure properties of Regular Languages, Pigeonhole Principle, Pumping Lemma, Application of Pumping Lemma. Decidability- Decision properties, Finite Automata and Regular Languages, Simulation of Transition Graph and Regular language.		
UNIT-III	**Context Free Language and Grammar**	**8 Hours**
Context Free Grammar (CFG)-Definition, Derivations, Languages, Derivation Trees and Ambiguity, Simplification of CFG, Normal Forms- Chomsky Normal Form (CNF), Greibach Normal Form (GNF), Pumping Lemma for CFL, Closure properties of CFL, Decision Properties of CFL		
UNIT-IV	**Push Down Automata**	**8 Hours**
Pushdown Automata- Definition, Representation, Instantaneous Description (ID), Acceptance by PDA, Nondeterministic Pushdown Automata (NPDA)- Definition, Moves, Pushdown Automata and Context Free Language, Pushdown Automata and Context Free Grammar, Two stack Pushdown Automata.		

Subject Syllabus

UNIT-V	Turing Machine and Undecidability	8 Hours
Turing Machine Model, Representation of Turing Machines, Language Acceptability of Turing Machines, Techniques for Turing Machine Construction, Variations of Turing Machine, Turing Machine as Computer of Integer Functions, Universal Turing machine, Linear Bounded Automata, Church's Thesis, Recursive and Recursively Enumerable language, Closure Properties of Recursive and Recursively Enumerable Languages, Non-Recursively Enumerable and Non-Recursive Languages, Undecidability, Halting Problem, Undecidability of Halting Problem, Post's Correspondence Problem.		

Syllabus

- **UNIT II**
- **Regular Language and Finite Automata**

- Regular Expressions, Transition Graph, Kleen’s Theorem, Finite Automata and Regular Expression- Arden’s theorem, Algebraic Method Using Arden’s Theorem, Regular Grammars-Right Linear and Left Linear grammars, Conversion of FA into Regular grammar and Regular grammar into FA, Regular and Non-Regular Languages- Closure properties of Regular Languages, Pigeonhole Principle, Pumping Lemma, Application of Pumping Lemma.
- Decidability- Decision properties, Finite Automata and Regular Languages, Simulation of Transition
- Graph and Regular language.

Branch Wise Applications

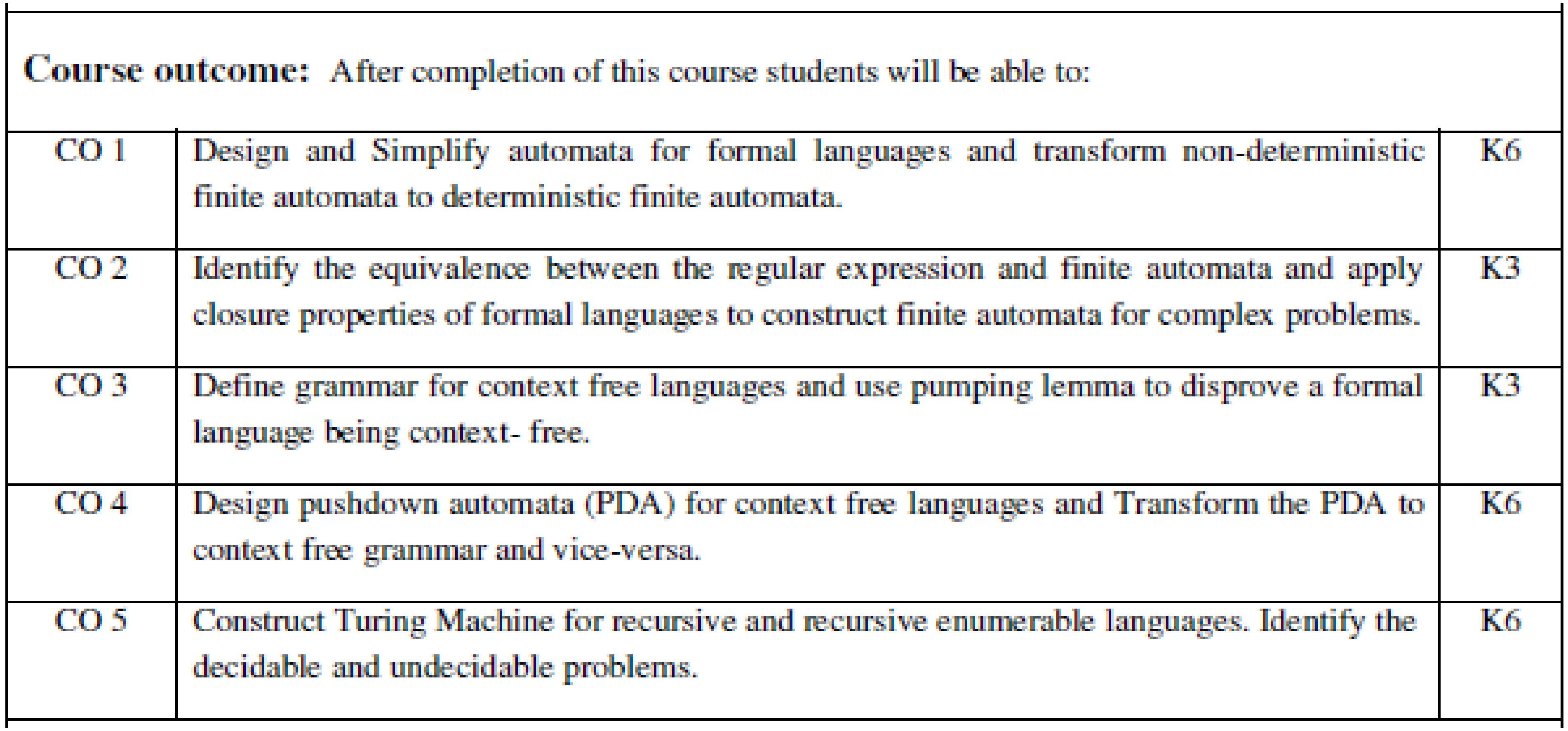
- It can compute man-made problems as well as natural phenomena.

- Automata theory has a lot of applications in real life as well, such that: Lambda calculus, Combinatory logic, Markov algorithm, and Register, Natural Language Processing ,Compiler Design and Lexical analysis and Semantic analysis. Also, the concept of Formal languages and automata theory can overlap with other subjects as well.

Course Objective

- Introduce concepts in automata theory and theory of computation
- Identify different formal language classes and their relationships and PDAs.
- Applying the Concept of Turing Machine and LBAs.
- Prove or disprove theorems in automata theory using its properties
- Determine the decidability and intractability of computational problems and complexity theory.

Course Outcome



Course outcome: After completion of this course students will be able to:

CO 1	Design and Simplify automata for formal languages and transform non-deterministic finite automata to deterministic finite automata.	K6
CO 2	Identify the equivalence between the regular expression and finite automata and apply closure properties of formal languages to construct finite automata for complex problems.	K3
CO 3	Define grammar for context free languages and use pumping lemma to disprove a formal language being context- free.	K3
CO 4	Design pushdown automata (PDA) for context free languages and Transform the PDA to context free grammar and vice-versa.	K6
CO 5	Construct Turing Machine for recursive and recursive enumerable languages. Identify the decidable and undecidable problems.	K6

Program Outcomes (POs)

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

Program Outcomes (POs)

Contd..

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

Program Outcomes (POs)

Contd..

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

CO-PO correlation matrix

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	3	3	2	2	-	-	2	1	-	3
CO2	**1**	**3**	**2**	**3**	**2**	**2**	-	**1**	**1**	**1**	**2**	**2**
CO3	2	2	3	2	2	2	-	2	2	1	2	3
CO4	2	2	2	3	2	2	-	2	2	1	1	3
CO5	3	2	2	2	2	2	-	2	1	1	1	2
Average	2	2.2	2.4	2.6	2	2	-	1.4	1.6	1	1.2	2.6

CO-PSO correlation matrix

CO	PSO			
	PSO1	PSO2	PSO3	PSO4
CO1	2	2	2	2
CO2	2	2	1	1
CO3	2	2	1	1
CO4	2	2	1	1
CO5	2	2	2	2
Average	2	2	1.4	1.4

Result Analysis

Subject Result:

Department Result:

Faculty-Wise Result:

End Semester Question Paper Template

B TECH

(SEM-V) THEORY EXAMINATION 20__-20__

Theory of Automata and Formal Languages

Time: 3 Hours **Total Marks: 100**

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. Attempt all questions in brief. 2 x 10 = 20

Q.No.	**Question**	**Marks**	**CO**
1		1	
2		1	
.		.	
10		1	

End Semester Question Paper Templates

SECTION B

2. Attempt any three of the following: 2X5=10

Q.No.	Question	Marks	CO
1		2	
2		2	
.		.	
5		2	

3. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		6	
2		6	

End Semester Question Paper Templates

4. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

5. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

6

Q.No.	Question	Marks	CO
1		10	
2		10	

End Semester Question Paper Templates

7. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

8. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

Prerequisite

- Discrete Mathematics
- Fundamental of Computer System

Links

UNIT 1-
https://www.youtube.com/watch?v=58N2N7zJGrQ&list=PLBlnK6fEyqRgp46KUv4ZY69yXmpwKOIev

UNIT 2-
https://www.youtube.com/watch?v=rjG5LwbqAp4

UNIT 3-
https://www.youtube.com/watch?v=SlSA9vEXCm4

UNIT 4-
https://www.youtube.com/watch?v=7lcwlNNCP1E

UNIT 5-
https://www.youtube.com/watch?v=LE_7krgRGt8

UNIT-2

Topics	**Duration (in Hours)**
Regular Expressions	4
Regular Languages	2
Pumping Lemma	2
Decision properties	1
Finite Automata and Regular Languages	1

Index

Objective of Unit

Objective of the course is to make students able to:

- Write regular expression.
- Correlate between regular expression and FA.
- Apply Arden’s Theorem.
- Use pumping lemma for regular languages.

CO-PO correlation matrix

CO-PO correlation matrix w.r.t. Unit-2

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO2	1	3	2	3	2	2	-	1	1	1	2	2

CO-PSO correlation matrix

CO-PSO correlation matrix w.r.t. Unit-2

CO	PSO			
	PSO1	PSO2	PSO3	PSO4
CO2	2	2	1	1

Prerequisite

- DFA and NFA are used to accept regular languages.
- Mealy and Moore machines are used to generate output.
- DFA and NFA have same power.
- It is easy to construct Ɛ-NFA.

Regular Expressions (CO2)

Objective of the Topic

The objective of the topic is to make the student able to:

- Understand application of regular expression.
- Write regular expression
- Realize the expressive power of RE and FA
- Convert FA to RE and vice-versa.

Regular Expression(RE) (CO2)

Topic mapping with Course Outcome

Topic	CO1	CO2	CO3	CO4	CO5
Regular Expression	-	3	-	-	-

Regular Expression(RE)

- We have already discussed that languages accepted by Finite Automata are called Regular Language.
- *Regular expressions* are an algebraic way to describe languages.
- They describe exactly the regular languages.
- If E is a regular expression, then L(E) is the language it defines.
- We'll describe RE's and their languages recursively.
- An expression that express the regular language is called the regular Expression
- RE is written for the Regular Language using **+, . , *** is known as Regular Expression.

Examples

- (a+b)
- (0+10*)
- (a+Ɛ)b*a
- (ab+ba)*

Regular Expression(RE)

Formally a regular expression is defined recursively as

1. Any terminal symbol (i.e. an element of ∑:), Ɛ and ϕ are regular expressions. When we view *a in ∑ as a regular expression, we* ***denote*** **it by a.**
2. The union of two regular expressions R1 and R2• written as **R1+ R2, is also a regular expression.**
3. The concatenation of two regular expressions R1 and R2, written as **R1 R2, is also a regular expression**.
4. The iteration (or closure) of a regular expression R written as **R*, is also a regular expression.**
5. *If R is a regular expression, then* ***(R) is also a regular expression****.*
6. The regular expressions over ∑ are precisely those obtained recursively by the application of the rules *1-5 once or several times.*

Regular Expression(RE)

Basis 1: If *a* is any symbol, then **a** is a RE, and L(**a**) = {a}.

Note: {a} is the language containing one string, and that string is of length 1.

Basis 2: ε is a RE, and L(ε) = {ε}.

Basis 3: ∅ is a RE, and L(∅) = ∅.

Induction 1: If E_1 and E_2 are regular expressions, then E_1+E_2 is a regular expression, and $L(E_1+E_2) = L(E_1) \cup L(E_2)$.

Induction 2: If E_1 and E_2 are regular expressions, then E_1E_2 is a regular expression, and $L(E_1E_2) = L(E_1)L(E_2)$.

Concatenation : the set of strings wx such that w
Is in $L(E_1)$ and x is in $L(E_2)$.

Regular Expression(RE)

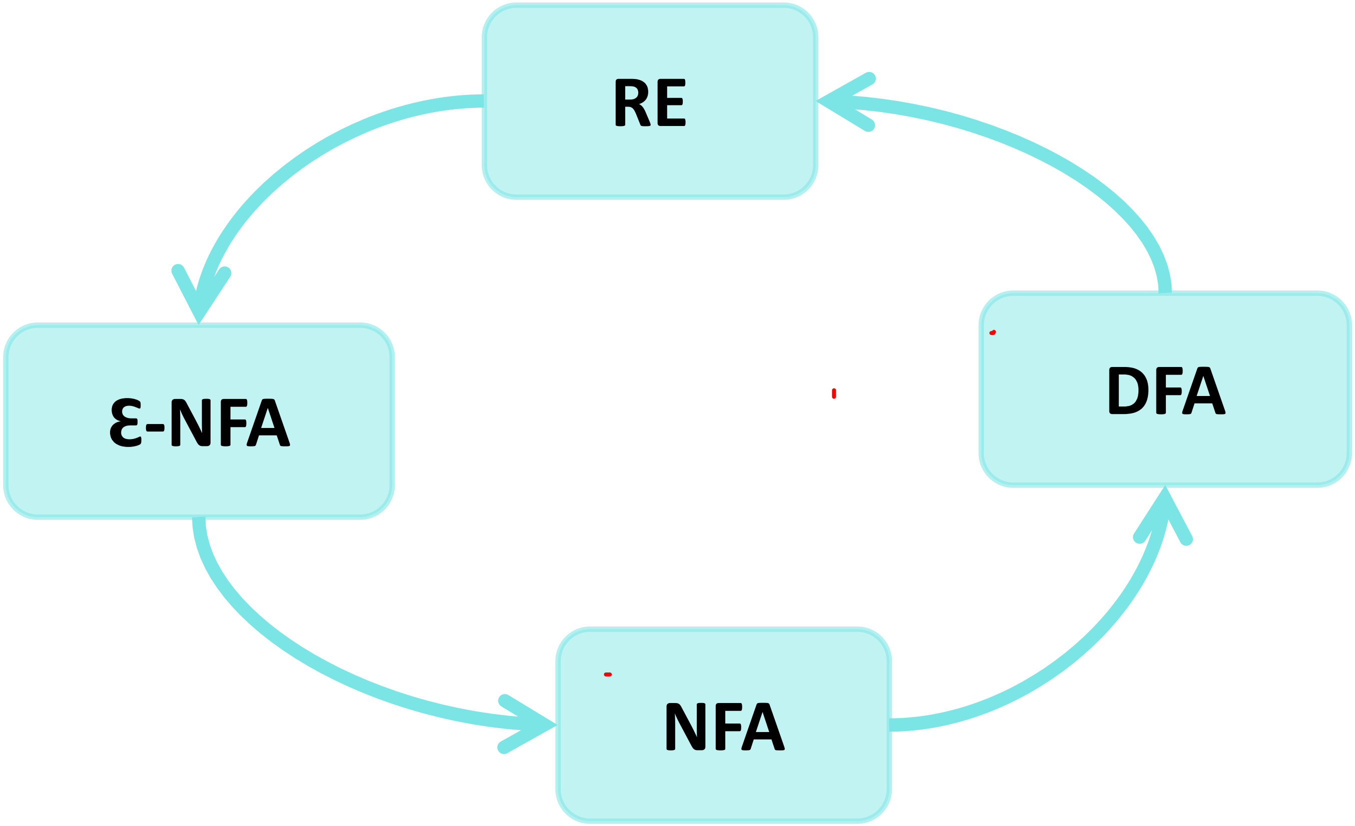
Induction 3: If E is a RE, then E* is a RE, and L(E*) = (L(E))*.

Closure, or “Kleene closure” = set of strings $w_1w_2\ldots w_n$, for some $n \geq 0$, where each w_i is in L(E).
Note: when n=0, the string is ε.

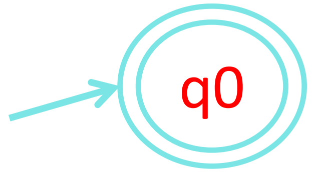
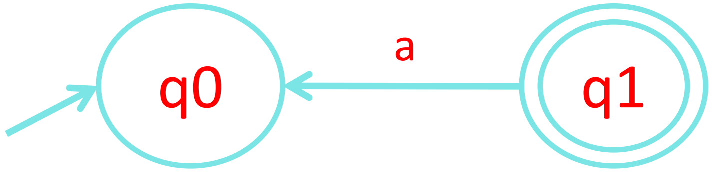
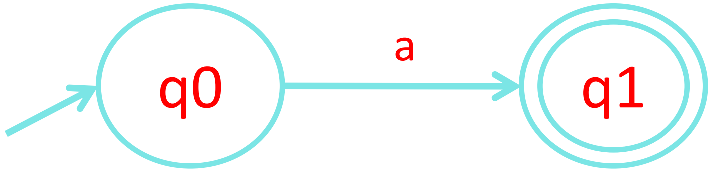
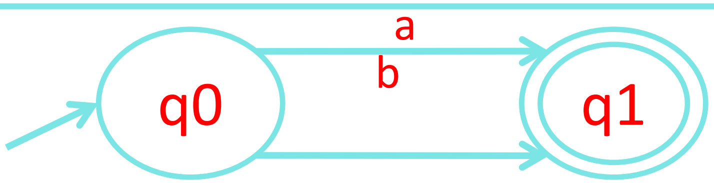
Precedence of Operators

Parentheses may be used wherever needed to influence the grouping of operators.
Order of precedence is * (highest), then concatenation, then + (lowest).

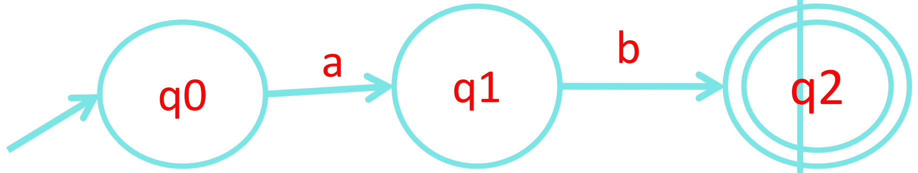
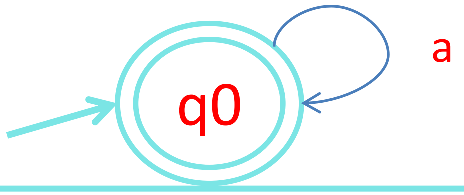
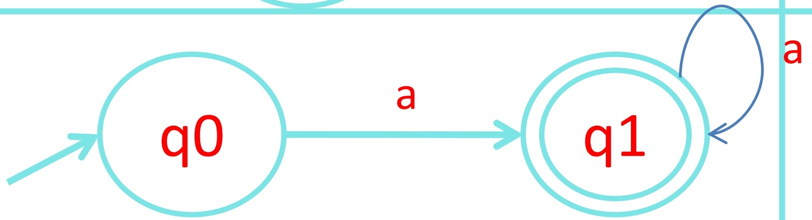
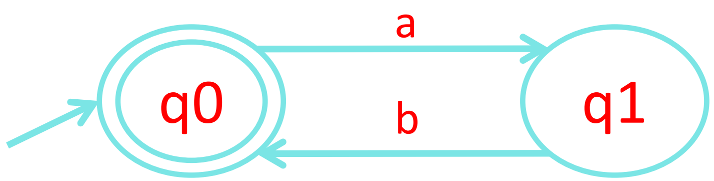
DFA, NFA, Ɛ-NFA and Regular Expression(RE)


$$\begin{tikzcd} & RE \arrow[dl, bend right] & \\ Ɛ\text{-NFA} \arrow[dr, bend right] & & DFA \arrow[ul, bend right] \\ & NFA \arrow[ur, bend right] & \end{tikzcd}$$

DFA, Regular Languages,Regular Expressions

Finite Automata	Regular Language	Regular Expression
	{Ɛ}	RE = Ɛ
	{ϕ}	RE = ϕ
	{a}	RE = a
	{a, b}	RE = a+b

DFA, Regular Languages,Regular Expressions

Finite Automata	Regular Language	Regular Expression
	{ab}	RE = a.b
	{Ɛ,a,aa,aaa,……}	RE = a*
	{a,aa,aaa,…….}	RE = a.a* = a^+
	{Ɛ, ab,abab,…….}	RE = (ab)*

Regular Expressions, Regular Languages

Regular Expressions	**Regular Languages**
(0 + 10*)	L = { 0, 1, 10, 100, 1000, 10000, … }
(0*10*)	L = {1, 01, 10, 010, 0010, …}
(0 + ε)(1 + ε)	L = {Ɛ, 0, 1, 01}
(a+b)*	L = {Ɛ, a, b, aa , ab , bb , ba, aaa…….}
(a+b)*abb	L = {abb, aabb, babb, aaabb, ababb, …………..} ending with abb
(11)*	L= {Ɛ, 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 = {Ɛ, aa, ab, ba, bb, aaab, aaba, …………..}

Identities of Regular Expression

I_1: $R + \emptyset = \emptyset + R = R$

I_2: $R. \emptyset = \emptyset$

I_3: $R.\varepsilon = \varepsilon.R = R$

I_4: $\varepsilon^* = \varepsilon$

I_5: $R+R=R$

I_6: $R^*R^* = R^*$

I_7: $RR^* = R^*R$

I_8: $(R^*)^* = R^*$

I_9: $\varepsilon + RR^* = \varepsilon + R^*R = R^*$

I_{10}: $(PQ)^*P = P(QP)^*$

I_{11}: $(P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^*$

I_{12}: $PQ+PR = P(Q+R)$ and $QP+RP = (Q+R)P$

Examples of Regular Expression

L(**01**) = {01}.

L(**01**+**0**) = {01, 0}.

L(**0**(**1**+**0**)) = {01, 00}.

Note order of precedence of operators.

L(**0***) = {ε, 0, 00, 000,… }.

L((**0**+**10**)*(ε+**1**)) = all strings of 0’s and 1’s without two consecutive 1’s.

Examples of Regular Expression

Q1: Write the regular expression for the language accepting all combinations of a's, over the set ∑ = {a}

Solution:

All combinations of a's means a may be zero, single, double and so on. If a is appearing zero times, that means a null string. That is we expect the set of {ε, a, aa, aaa, ...}. So we give a regular expression for this as:

R = a*

That is Kleen closure of a.

Examples of Regular Expression

Q2: Write the regular expression for the language accepting all combinations of a's except the null string, over the set ∑ = {a}

Solution:

The regular expression has to be built for the language

L = {a, aa, aaa, ...}

This set indicates that there is no null string. So we can denote regular expression as: R = a^+

Q3: Write the regular expression for the language accepting all the string containing any number of a's and b's.

Solution:

The regular expression will be:

r.e. = (a + b)*

This will give the set as L = {ε, a, aa, b, bb, ab, ba, aba, bab, ...}, any combination of a and b.

The (a + b)* shows any combination with a and b even a null string.

Examples of Regular Expression

Q4: Write the regular expression for the language accepting all the string which are starting with 1 and ending with 0, over ∑ = {0, 1}.

Solution:

In a regular expression, the first symbol should be 1, and the last symbol should be 0. The r.e. is as follows:

R = 1 (0+1)* 0

Q5: Write the regular expression for the language starting and ending with a and having any having any combination of b's in between.

Solution:

The regular expression will be:

R = a b* b

Examples of Regular Expression

Q6:
Write the regular expression for the language starting with a but not having consecutive b's.
Solution: The regular expression has to be built for the language:
L = {a, aba, aab, aba, aaa, abab, ...}
The regular expression for the above language is:
R = {a + ab}*
Q7:
Write the regular expression for the language accepting all the string in which any number of a's is followed by any number of b's is followed by any number of c's.
Solution: As we know, any number of a's means a* any number of b's means b*, any number of c's means c*. Since as given in problem statement, b's appear after a's and c's appear after b's. So the regular expression could be:
R = a* b* c*

Examples of Regular Expression

Q8:Write the regular expression for the language over $\sum = \{0\}$ having even length of the string.

Solution:

The regular expression has to be built for the language:

L = {ε, 00, 0000, 000000, ...}

The regular expression for the above language is:

R = (00)*

Q9:

Write the regular expression for the language having a string which should have atleast one 0 and alteast one 1.

Solution:

The regular expression will be:

$R = [(0 + 1)^* \, 0 \, (0 + 1)^* \, 1 \, (0 + 1)^*] + [(0 + 1)^* \, 1 \, (0 + 1)^* \, 0 \, (0 + 1)^*]$

Examples of Regular Expression

Q10: Describe the language denoted by following regular expression

r.e. = (b* (aaa)* b*)*

Solution:

The language can be predicted from the regular expression by finding the meaning of it.

We will first split the regular expression as:

r.e. = (any combination of b's) (aaa)* (any combination of b's)

L = {The language consists of the string in which a's appear triples, there is no restriction on the number of b's}

Q11:

Write the regular expression for the language L over ∑ = {0, 1} such that all the string do not contain the substring 01.

Solution:

The Language is as follows:

L = {ε, 0, 1, 00, 11, 10, 100, ...}

The regular expression for the above language is as follows:

R = (1* 0*)

Examples of Regular Expression

Q12:

Write the regular expression for the language containing the string over {0, 1} in which there are at least two occurrences of 1's between any two occurrences of 1's between any two occurrences of 0's.

Solution: At least two 1's between two occurrences of 0's can be denoted by (0111*0)*.

Similarly, if there is no occurrence of 0's, then any number of 1's are also allowed. Hence the r.e. for required language is:

R = (1 + (0111*0))*

Q13:

Write the regular expression for the language containing the string in which every 0 is immediately followed by 11.

Solution:

The regular expectation will be:

R = (011 + 1)*

Equivalence of RE's and Automata

- We need to show that for every RE, there is an automaton that accepts the same language.
  - Pick the most powerful automaton type: the ε-NFA.
- And we need to show that for every automaton, there is a RE defining its language.
  - Pick the most restrictive type: the DFA.

Kleen’s Theorem

- Kleene's theorem: The set of regular languages, the set of NFA-recognizable languages, and the set of DFA-recognizable languages are all the same.
- Proof: We must be able to translate between NFAs, DFAs, and regular expressions. We have covered the following algorithms to do these translations:

Kleen’s Theorem

Tranlations overview; DFA to NFA, NFA to regular expression (via generalized NFA), regular expression to DFA (via ε-NFA)

$$\begin{tikzcd}
\text{NFA} \arrow[rrrr, bend left, "\text{power set}"] \arrow[dr, swap, "\text{trivial}"] & & & & \text{DFA} \arrow[llll, "\text{trivial}"'] \\
& \text{generalized NFA} \arrow[dr, swap, "\text{remove nodes}"] & & \varepsilon\text{-NFA} \arrow[ur, swap, "\text{powerset}"] & \\
& & \text{regular expression} \arrow[ur, swap, "\text{struct. ind.}"] & &
\end{tikzcd}$$

Arden’s Theorem

Let **P** and **Q** be two regular expressions. If P does not contain Ɛ, then

R = Q + RP has a unique solution that is

R = QP*

Proof

R=Q+RP

= Q+QP*P

= Q(Ɛ + P*P)

= QP*

So, R = QP* is a solution.

R = Q+RP

= Q + (Q + RP)P

= Q + QP + RP^2

= Q + QP + (Q + RP)PP

= Q + QP + QP^2 + RP^3

=…

=…

= Q + QP + QP^2 +_ _ _ _ _ + QP^n + RP^{n+1}

= Q + QP + QP^2 +_ _ _ _ _ + QP^n + QP*P^{n+1}

=Q (Ɛ+ P + P2 + _ _ _ _ _ + P^n + P*P^{n+1})

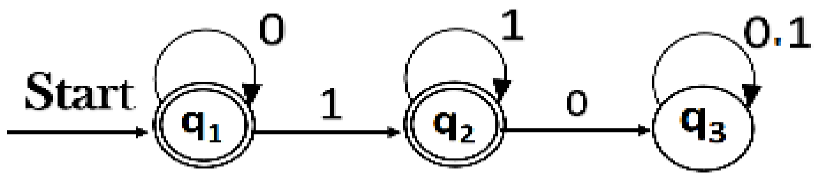
=QP*

Hence, R=QP* is a Unique Solution.

Example of Arden’s theorem

Q1: Construct the regular expression for the given DFA

Solution:



Let us write down the equations

q1 = q1 0 + ε

Since q1 is the start state, so ε will be added, and the input 0 is coming to q1 from q1 hence we write

State = source state of input × input coming to it

Similarly,

q2 = q1 1 + q2 1 q3 = q2 0 + q3 (0+1)

Example of Arden’s theorem

Since the final states are q1 and q2, we are interested in solving q1 and q2 only. Let us see q1 first

$q1 = q1\ 0 + \varepsilon$

We can re-write it as

$q1 = \varepsilon + q1\ 0$

Which is similar to $R = Q + RP$, and gets reduced to $R = OP^*$.

Assuming $R = q1$, $Q = \varepsilon$, $P = 0$

We get

$q1 = \varepsilon.(0)^*\ q1 = 0^*\ (\varepsilon.R^* = R^*)$

Substituting the value into q2, we will get

$q2 = 0^*\ 1 + q2\ 1\ q2 = 0^*\ 1\ (1)^*\ (R = Q + RP \rightarrow Q\ P^*)$

The regular expression is given by

$r = q1 + q2 = 0^* + 0^*\ 1.1^*\ r = 0^* + 0^*\ 1^+\ (1.1^* = 1^+)$

Example of Arden’s theorem

Q2:Find regular expression for the following DFA using Arden’s Theorem-

Solution-

Step-01:

Form a equation for each state-

q1 = ∈ ……(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 = ∈.a

q2 = a …….(4)

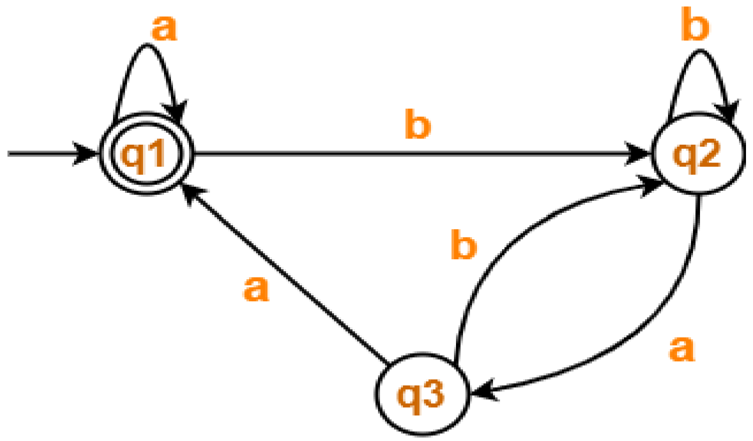
Example of Arden's theorem

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

q3 = q1.b + q2.a + q3.a
q3 = ∈.b + a.a + q3.a
q3 = (b + a.a) + q3.a …….(5)

Using Arden's Theorem in (5), we get-
q3 = (b + a.a)a*
Thus, Regular Expression for the given DFA = (b + aa)a*

Q3: Find regular expression for the following DFA using Arden's Theorem-



Example of Arden’s theorem

Step-01:

Form a equation for each state-

$q1 = \in + q1.a + q3.a$ ……(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 (3) in (2), we get-

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

$q2 = q1.b + q2.(b + a.b)$ …….(4)

Example of Arden’s theorem

Using (5) in (3), we get-
q3 = q1.b.(b + a.b)*.a …….(6)

Using (6) in (1), we get-
q1 = ∈ + q1.a + q1.b.(b + a.b)*.a.a
q1 = ∈ + q1.(a + b.(b + a.b)*.a.a) …….(7)

Using Arden’s Theorem in (7), we get-
q1 = ∈.(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)*

Algebraic method using 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*."

R = Q + RP ......(i)
Now, replacing R by R = QP*, we get,
R = Q + QP*P
Taking Q as common,
R = Q(+ P*P) R = 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. Let me take this equation again:
R = Q + RP
Now, replace R by R = Q + RP,
R = Q + (Q + RP)P = Q + QP + R

Algebraic method using Arden’s theorem

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

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

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

R = Q + QP + Q   + .. + Q   + QP*

Taking Q as common,

R = Q(+ P +   + .. +   + P*   ) = QP* [As   + P +   + .. +   + P*   represent the closure of P]

Hence proved.

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

Regular grammar Right linear

In this type of regular grammar, all the non-terminals on the right-hand side exist at the rightmost place, i.e.; right ends.

Examples :
A ⇢ a, A ⇢ a**B**, A ⇢ ∈ where, A and B are non-terminals,
a is terminal, and ∈ is empty string
S ⇢ 00**B** | 11**S** B ⇢ 0**B** | 1**B** | 0 | 1
where, S and B are non-terminals, and 0 and 1 are terminals

Regular grammar Left linear

In this type of regular grammar, all the non-terminals on the left-hand side exist at the leftmost place, i.e.; left ends.

Examples :

A ⇢ a, A ⇢ **B**a, A ⇢ ∈ where, A and B are non-terminals,
a is terminal, and ∈ is empty string
S ⇢ **B**00 | **S**11 B ⇢ **B**0 | **B**1 | 0 | 1 where S and B are non-terminals, and 0 and 1 are terminals

Conversion of regular FA into Regular Grammar

Algorithm for conversion

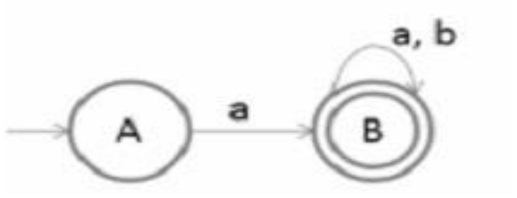
The algorithm to convert the finite automata (FA) to the right linear grammar is as follows −

Step 1 − Begin the process from the start state.
Step 2 − Repeat the process for each state.
Step 3 − Write the production as the output followed by the state on which the transition is going.
Step 4 − And at last, add € (epsilon) to end the derivation.

Conversion of regular FA into Regular Grammar

Example:

Let’s consider a Finite automaton (FA) as given below –



Pick the start state A and output is on symbol ‘a’ going to state B

$A \rightarrow aB$

Now we will pick state B and then we will go on each output

i.e $B \rightarrow aB$

$B \rightarrow bB$

$B \rightarrow \varepsilon$

Therefore,

Final right linear grammar is as follows –

$A \rightarrow aB$

$B \rightarrow aB/bB/\varepsilon$

Conversion of regular Grammar into FA

Algorithm

The algorithm to convert the finite automata (FA) to the right linear grammar is as follows −

Step 1 − Begin the process from the start state.

Step 2 − Repeat the process for each state.

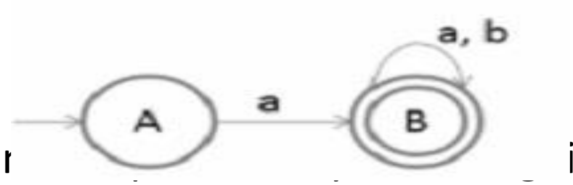
Step 3 − Write the production as the output followed by the state on which the transition is going.

Step 4 − And at last, add € (epsilon) to end the derivation

Conversion of regular Grammar into FA

Example 1

Let’s consider a Finite automaton (FA) as given below –



Pick the start state A ar ing to state B

A→aB

Now we will pick state B and then we will go on each output

i.e B→aB

B→bB

B→ε

Therefore,

Final right linear grammar is as follows –

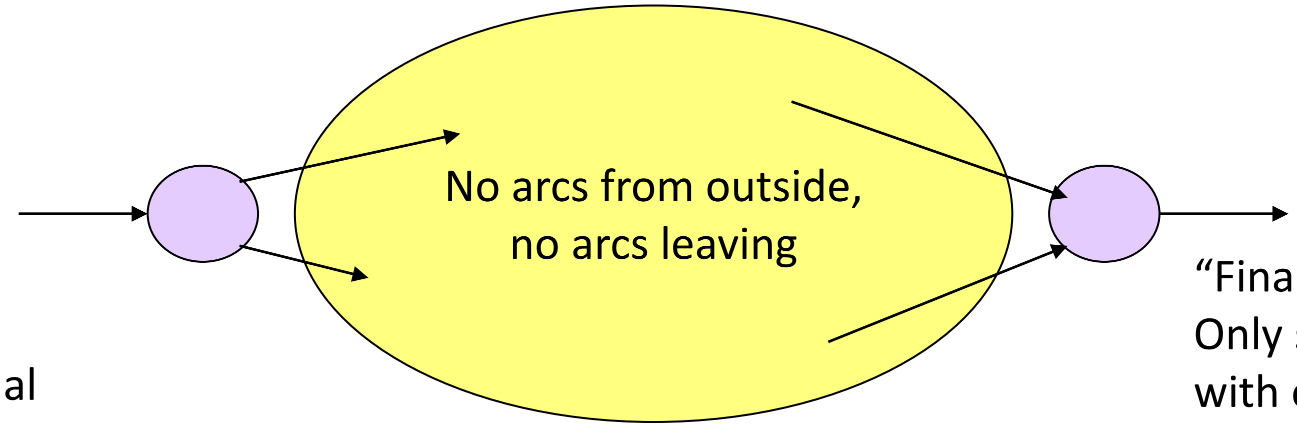
A→aB

B→aB/bB/ε

Converting a RE to an ϵ-NFA

- Proof is an induction on the number of operators (+, concatenation, *) in the RE.
- We always construct an automaton of a special form (next slide).

Form of ϵ-NFA's Constructed

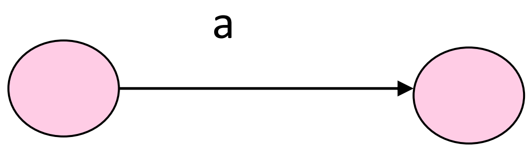


Start state:
Only state
with external
predecessors

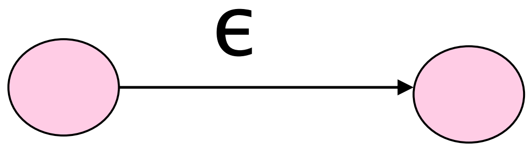
"Final" state:
Only state
with external
successors

RE to ϵ-NFA: Basis

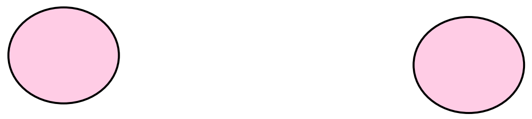
- Symbol **a**:



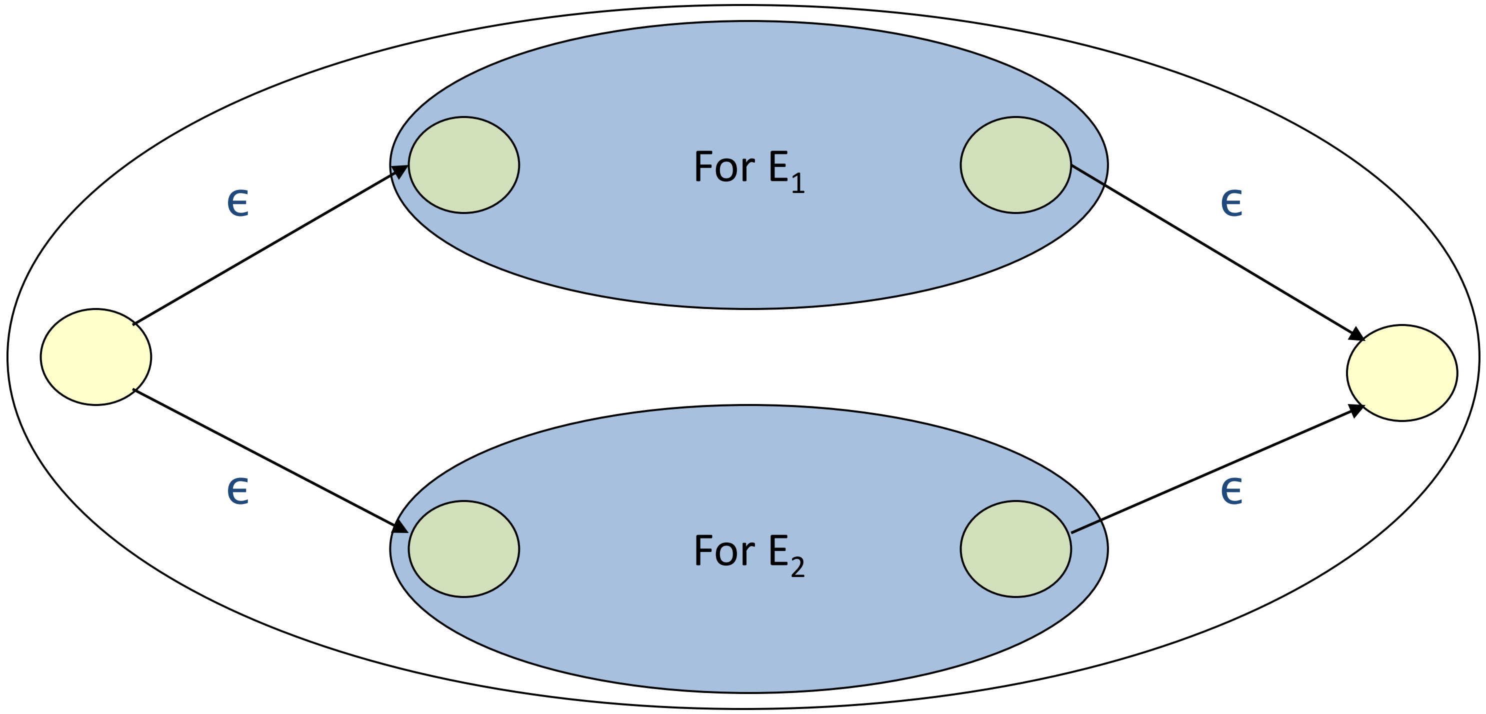
- ϵ:



- ∅:

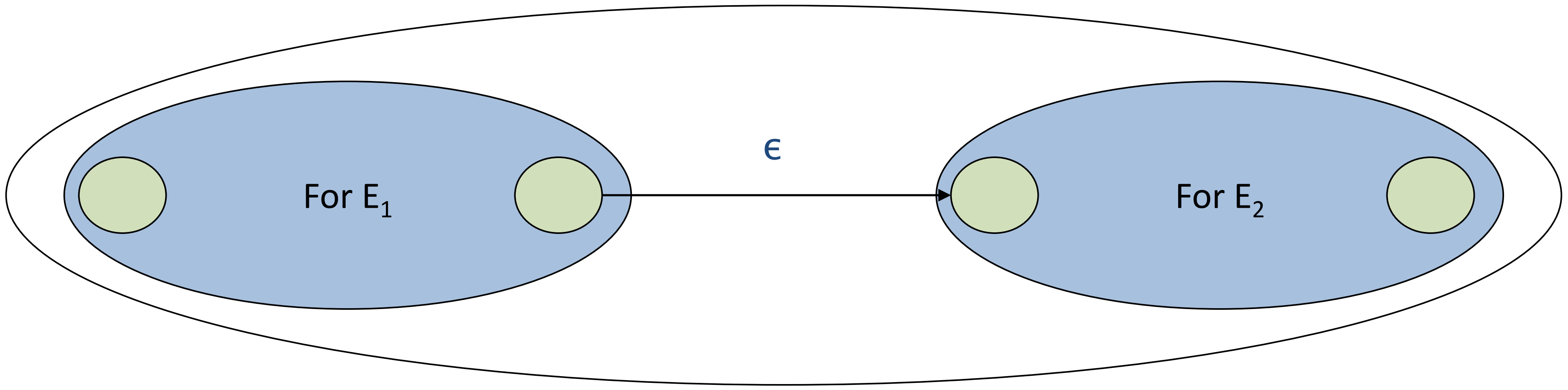


RE to ϵ-NFA: Induction 1 – Union



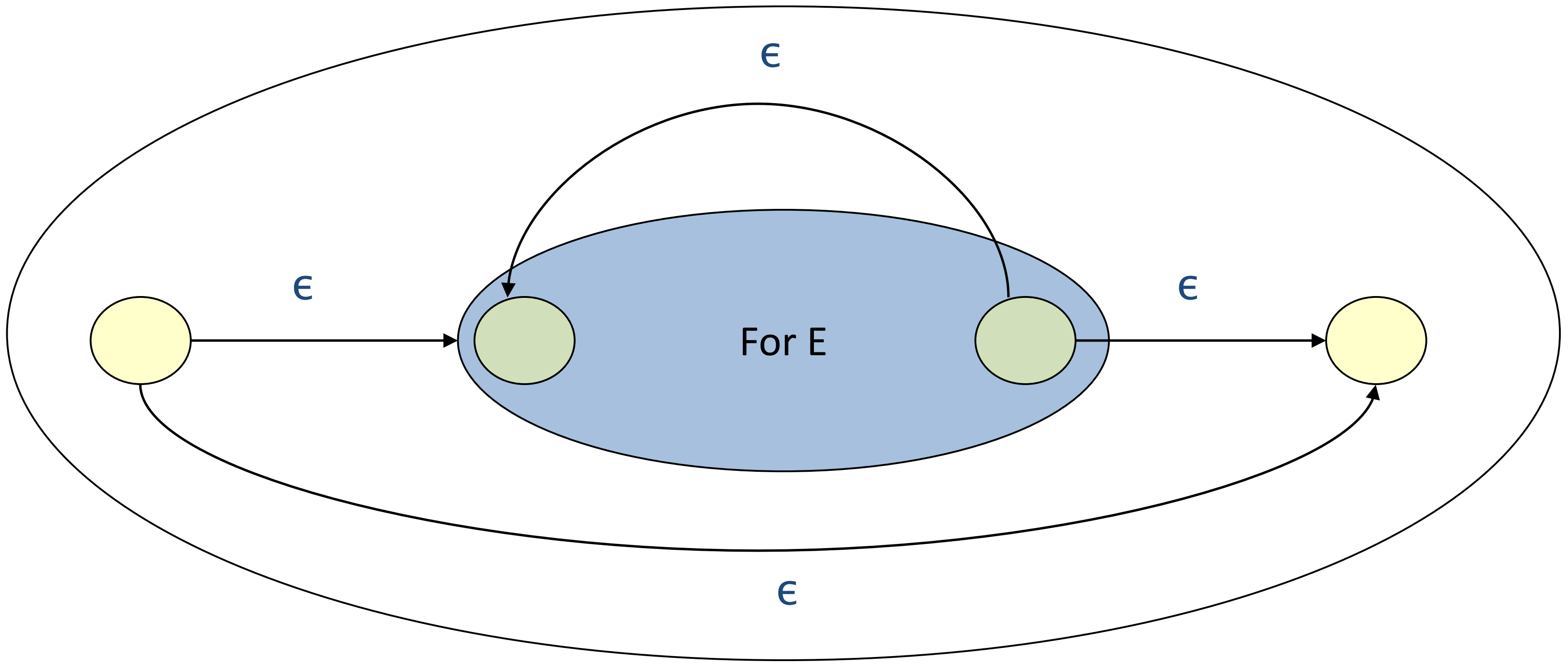
For $E_1 \cup E_2$

RE to ϵ-NFA: Induction 2 – Concatenation



For E_1E_2

RE to ϵ-NFA: Induction 3 – Closure



For E*

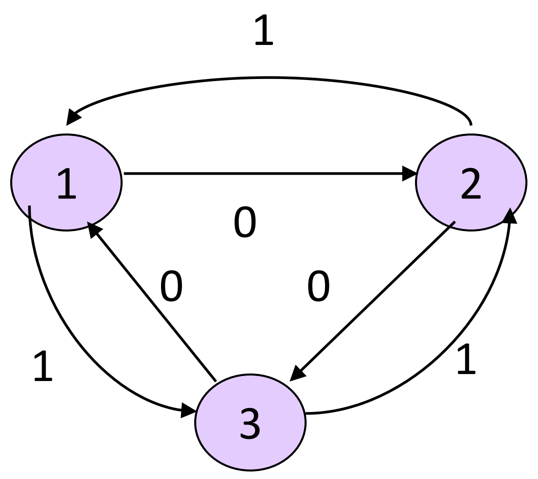
DFA-to-RE

- A strange sort of induction.
- States of the DFA are assumed to be 1,2,…,n.
- We construct RE’s for the labels of restricted sets of paths.
  - Basis: single arcs or no arc at all.
  - Induction: paths that are allowed to traverse next state in order.

k-Paths

- A k-path is a path through the graph of the DFA that goes though no state numbered higher than k.
- Endpoints are not restricted; they can be any state.

Example: k-Paths



0-paths from 2 to 3:
RE for labels = **0**.

1-paths from 2 to 3:
RE for labels = **0**+**11**.

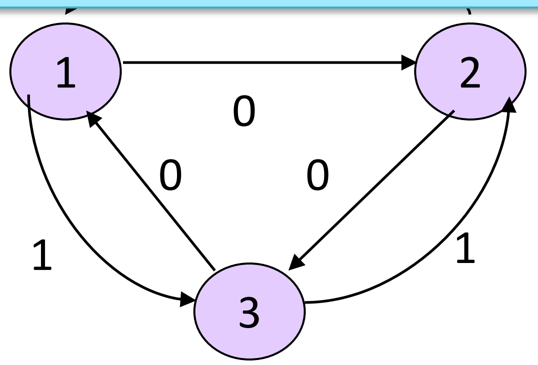
2-paths from 2 to 3:
RE for labels =
(**10**)***0**+**1**(**01**)***1**

3-paths from 2 to 3:
RE for labels = ??

k-Path Induction

- Let R_{ij}^{k} be the regular expression for the set of labels of k-paths from state i to state j.
- Basis: k=0. R_{ij}^{0} = sum of labels of arc from i to j.
  - $\varnothing$ if no such arc.
  - But add ϵ if i=j.

Example: Basis



- R_{12}^{0} = **0**.
- $R_{11}^{0} = \emptyset + \epsilon = \epsilon$.

k-Path Inductive Case

- A k-path from i to j either:
  1. Never goes through state k, or
  2. Goes through k one or more times.

$$R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^{*} R_{kj}^{k-1}.$$

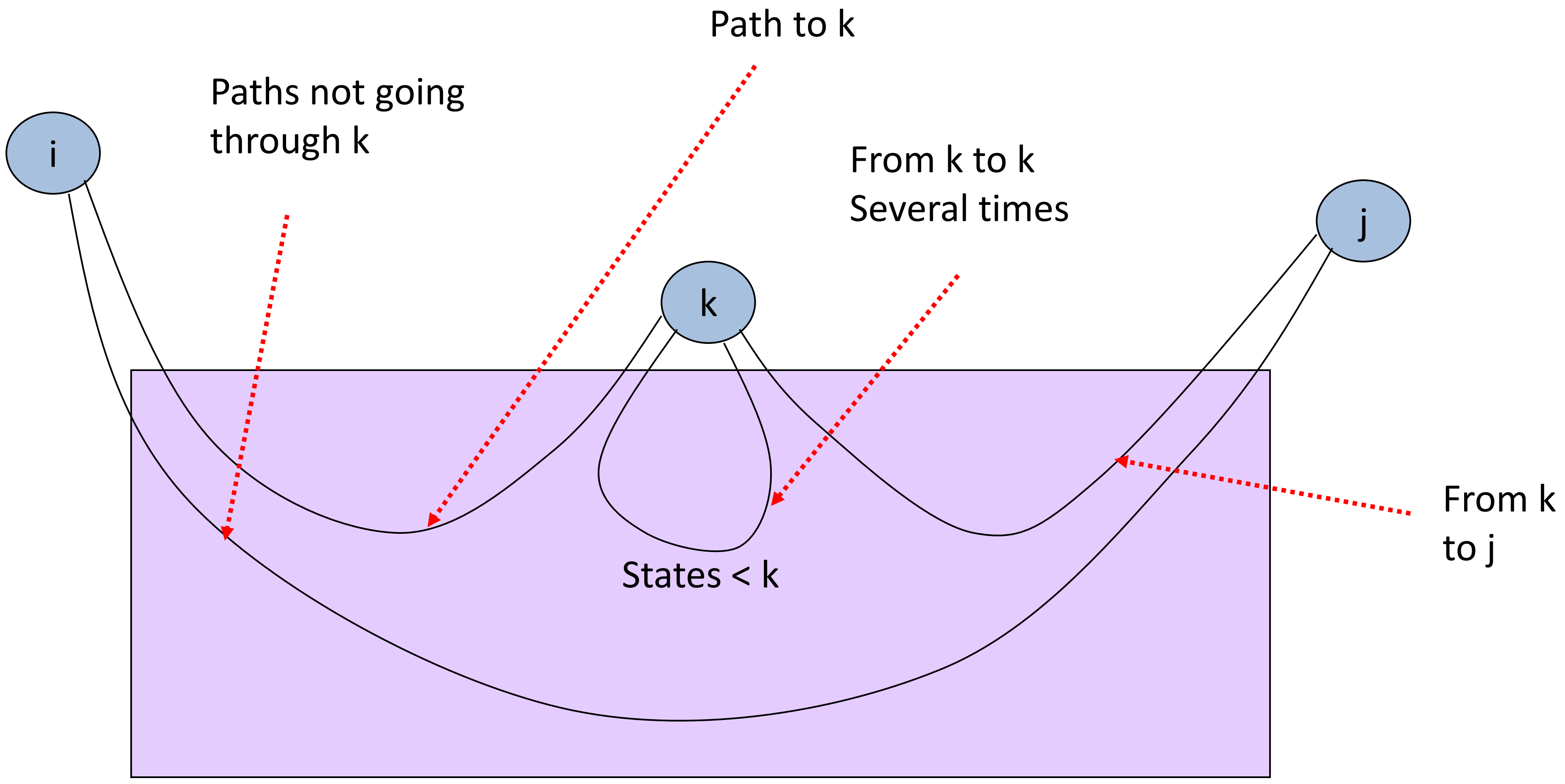
Doesn't go through k

Goes from i to k the first time

Zero or more times from k to k

Then, from k to j

Illustration of Induction

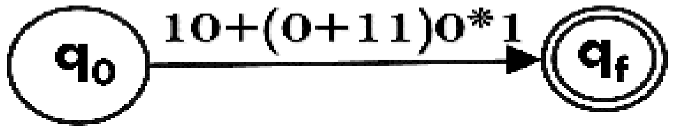


Example of RE to NFA

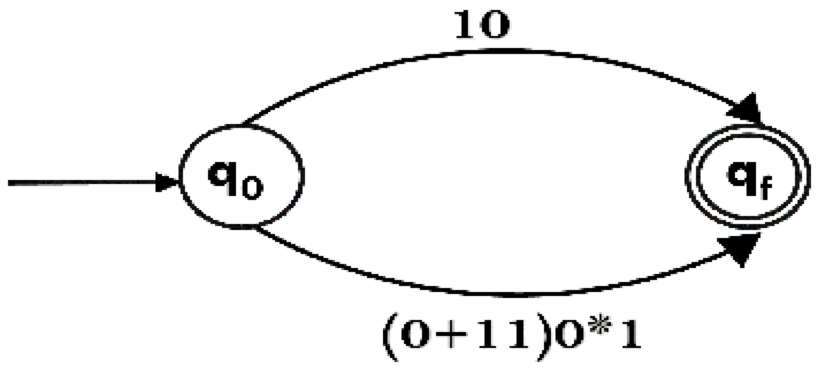
Design a FA from given regular expression 10 + (0 + 11)0* 1.

Solution: First we will construct the transition diagram for a given regular expression.

Step 1:



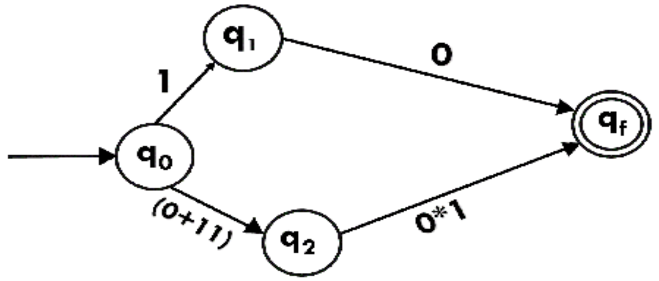
Step 2:



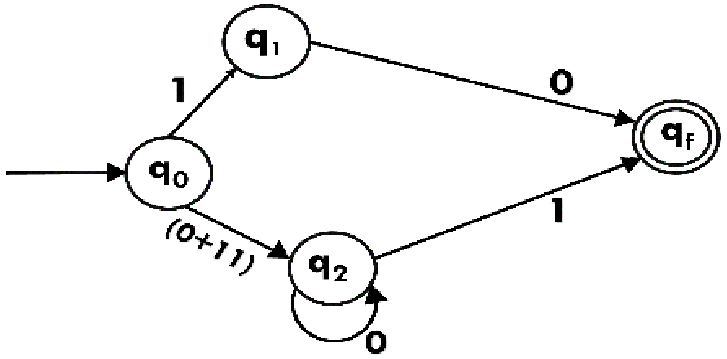
The equivalent DFA will be:

Example of RE to NFA

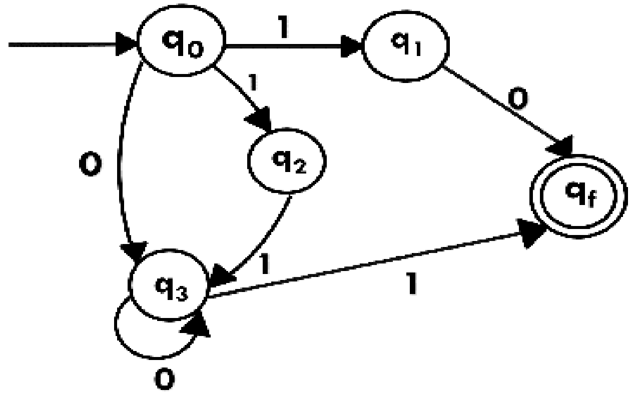
Step 3:



Step 4:



Step 5:



The equivalent DFA will be:

Example of RE to NFA

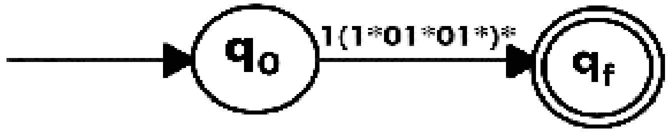
Now we have got NFA without ε. Now we will convert it into required DFA for that, we will first write a transition table for this NFA.

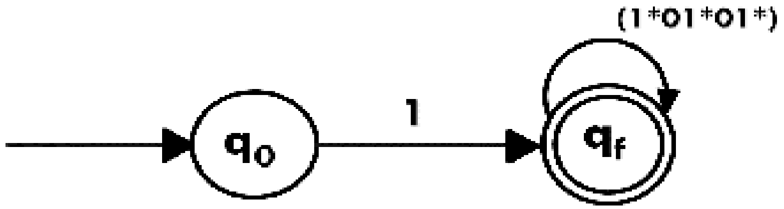
State	0	1
→q0	q3	{q1, q2}
q1	qf	ϕ
q2	ϕ	q3
q3	q3	qf
*qf	ϕ	ϕ

State	0	1
→[q0]	[q3]	[q1, q2]
[q1]	[qf]	ϕ
[q2]	ϕ	[q3]
[q3]	[q3]	[qf]
[q1, q2]	[qf]	[qf]
*[qf]	ϕ	ϕ

Example of RE to NFA

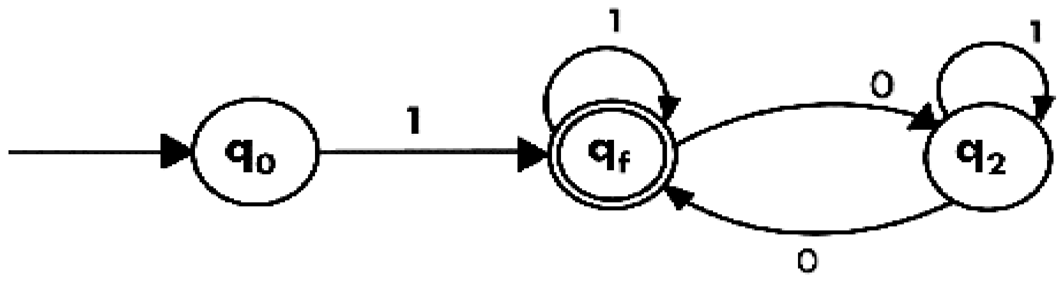
Design a NFA from given regular expression 1 (1* 01* 01*)*.
Solution: The NFA for the given regular expression is as follows:
Step 1:





Example of RE to NFA

Step 3:

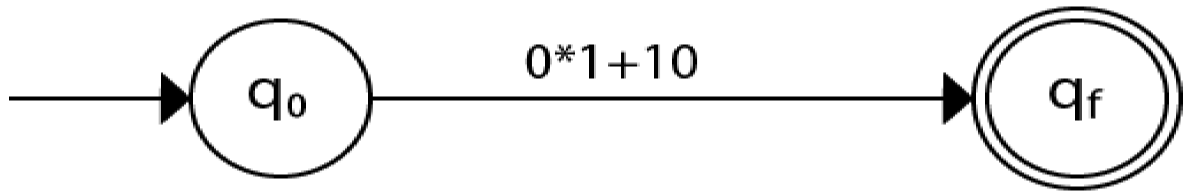


Construct the FA for regular expression 0*1 + 10.

Solution:

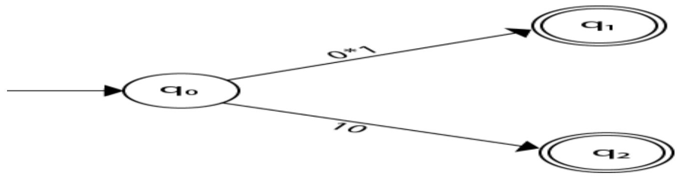
We will first construct FA for R = 0*1 + 10 as follows:

Step 1:

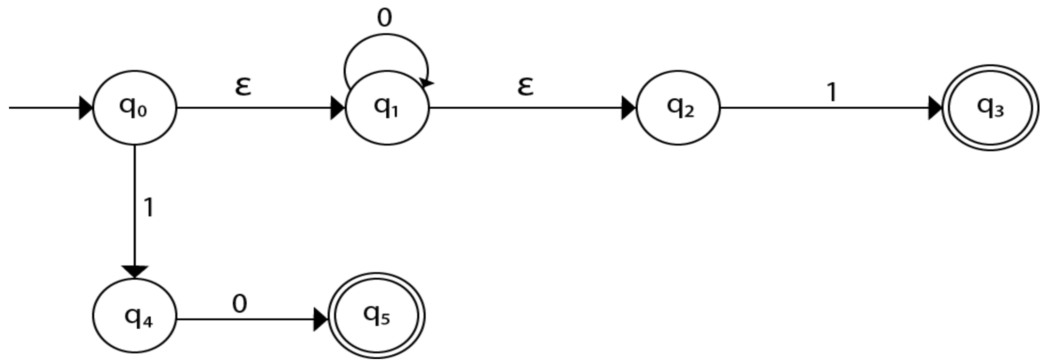


Example of RE to NFA

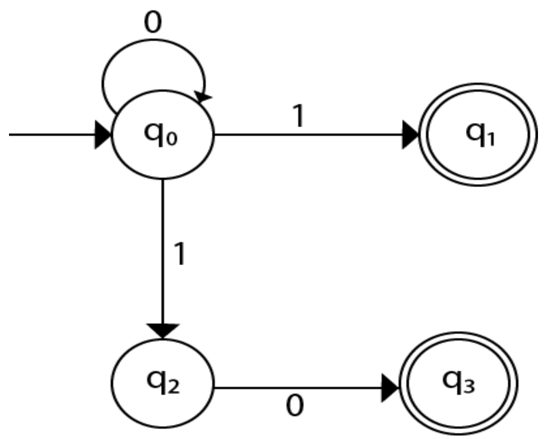
Step 2:



Step 3:

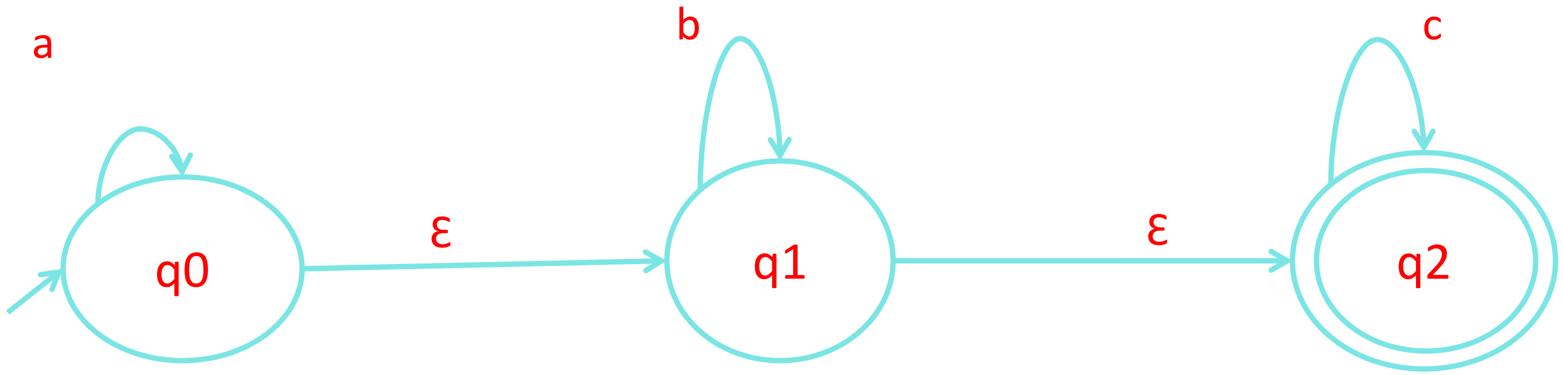


Step 4:



Regular Expression to NFA

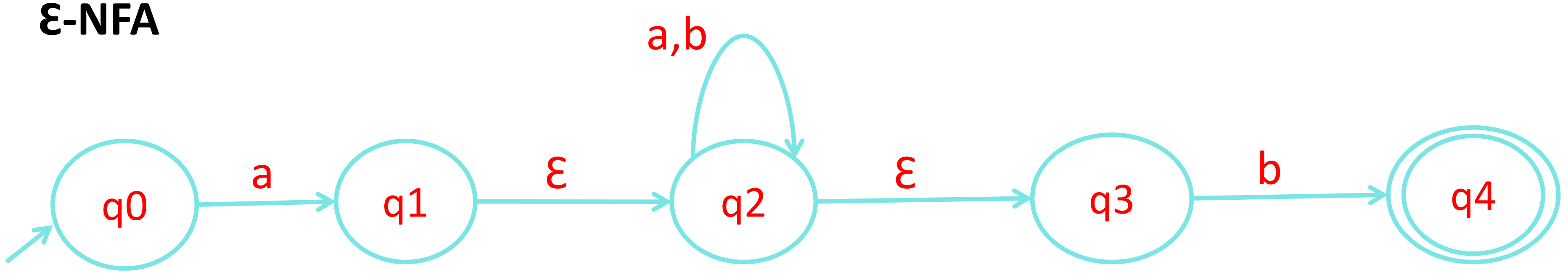
Construct an NFA for a*b*c*.



Regular Expression to NFA

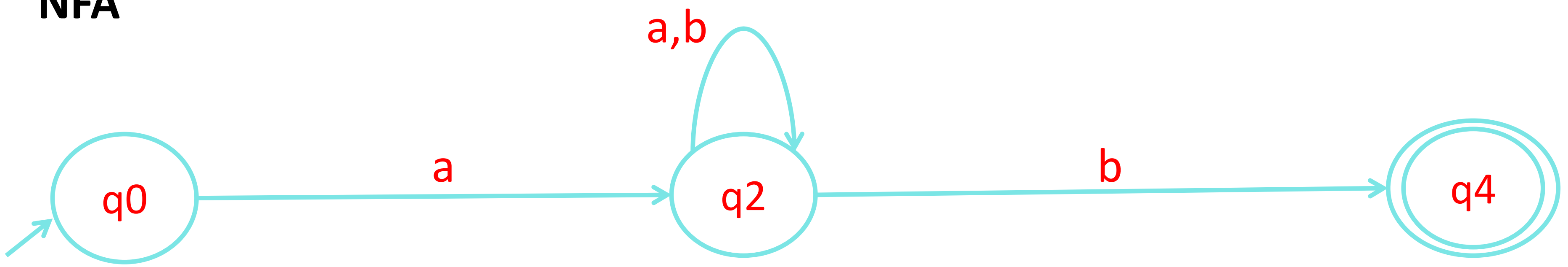
Construct an NFA for a(a+b)*b.

Ɛ-NFA



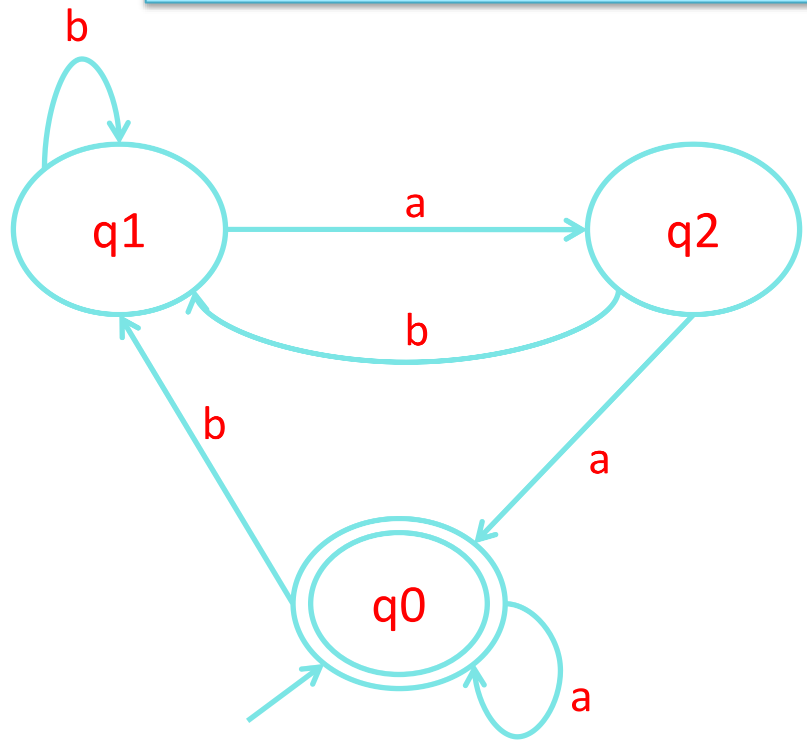
$$\begin{tikzcd} \arrow[r] & q0 \arrow[r, "a"] & q1 \arrow[r, "\varepsilon"] & q2 \arrow[loop above, "{a,b}"] \arrow[r, "\varepsilon"] & q3 \arrow[r, "b"] & \boxed{q4} \end{tikzcd}$$

NFA



$$\begin{tikzcd} \arrow[r] & q0 \arrow[r, "a"] & q2 \arrow[loop above, "{a,b}"] \arrow[r, "b"] & \boxed{q4} \end{tikzcd}$$

DFA to Regular Expression


$$\begin{tikzcd} q1 \arrow[loop above, "b"] \arrow[r, "a"] & q2 \arrow[l, bend left, "b"] \arrow[dl, "a"] \\ & q0 \arrow[ul, "b"] \arrow[loop right, "a"] \end{tikzcd}$$

$q_0 = q_0a + q_2a + ε$

$q_1 = q_0b + q_1b + q_2b$

$q_2 = q_1a$

$q_1 = q_0b + q_1b + q_2b$

$= q_0b + q_1b + (q_1a)b$ (Substituting value of q_2)

$= q_0b + q_1(b + ab)$

$= q_0b (b + ab)^*$ (Applying Arden's Theorem)

$q_0 = q_0a + q_2a + ε$

$= q_0a + q_1aa + ε$ (Substituting value of q_2)

$= q_0a + q_0b(b + ab^*)aa + ε$ (Substituting value of q_1)

$= q_0(a + b(b + ab)^*aa) + ε$

$= ε (a+ b(b + ab)^*aa)^*$

$= (a + b(b + ab)^*aa)^*$

Recap

We learnt:

- Regular Expressions,
- Transition Graph,
- Kleen’s Theorem,
- Finite Automata and Regular Expression- Arden’s theorem,
- Algebraic Method Using Arden’s Theorem,

Regular and Non-regular Languages

Objective of the Topic

The objective of the topic is to make the student able to:

- Differentiate between regular and non-regular languages.
- Generate language from FA/ RE
- Learn the closure properties of regular languages.
- Convert FA to RE and vice-versa.

Regular and Non-regular Languages

Topic mapping with Course Outcome

Topic	CO1	CO2	CO3	CO4	CO5
Regular and Non – regular Languages	-	3	-	-	-

Regular and Non-regular Languages

- Regular languages are accepted by Finite automata.
- They are generated by Regular grammar.
- Regular language can also be expressed in the form of regular expression.

Closure Properties of Regular Languages

- Regular languages are closed under:
  - Complement
  - Concatenation
  - Union
  - Intersection
  - Reverse
  - Difference
  - Kleene closure
  - Homomorphism
  - Inverse homomorphism

Closure Properties of Regular Languages

Complement: If L is a regular language then L’ is also a regular language.

Proof

Let D(L) = (Q, Σ, δ,q0, F) be a DFA that accepts L.

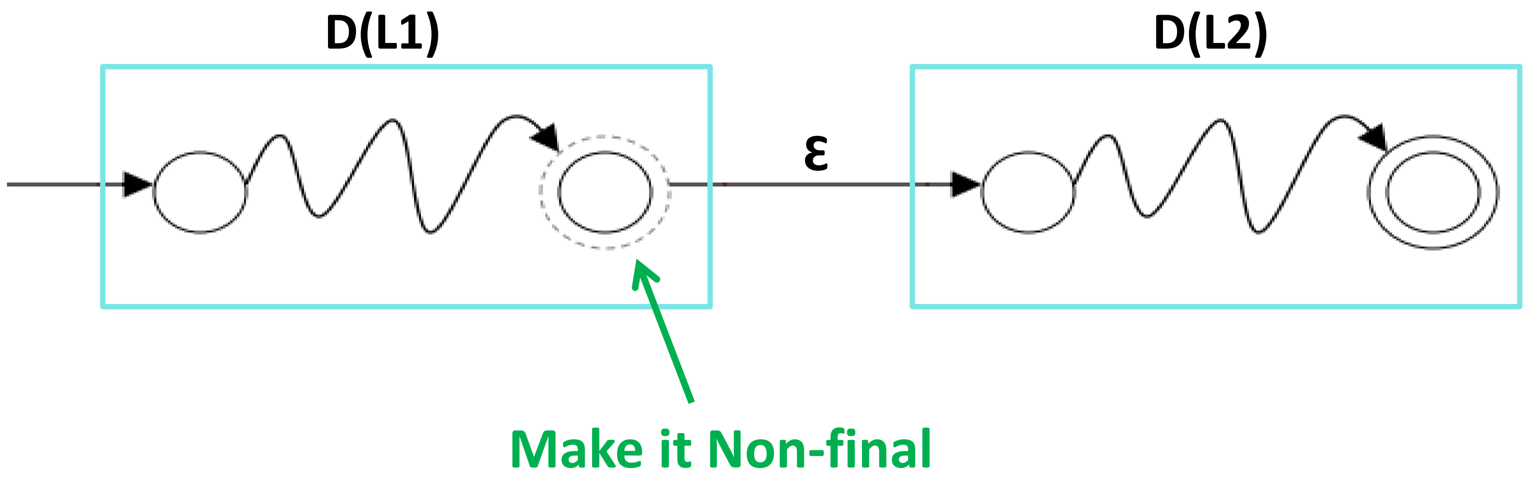
Then we can construct a DFA D(L’) = (Q, Σ, δ,q0, Q-F) that accepts L’.

Make final state non final and vice-versa.

Closure Properties of Regular Languages

Concatenation: If L1 and L2 are regular languages then L1·L2 is also a regular language.

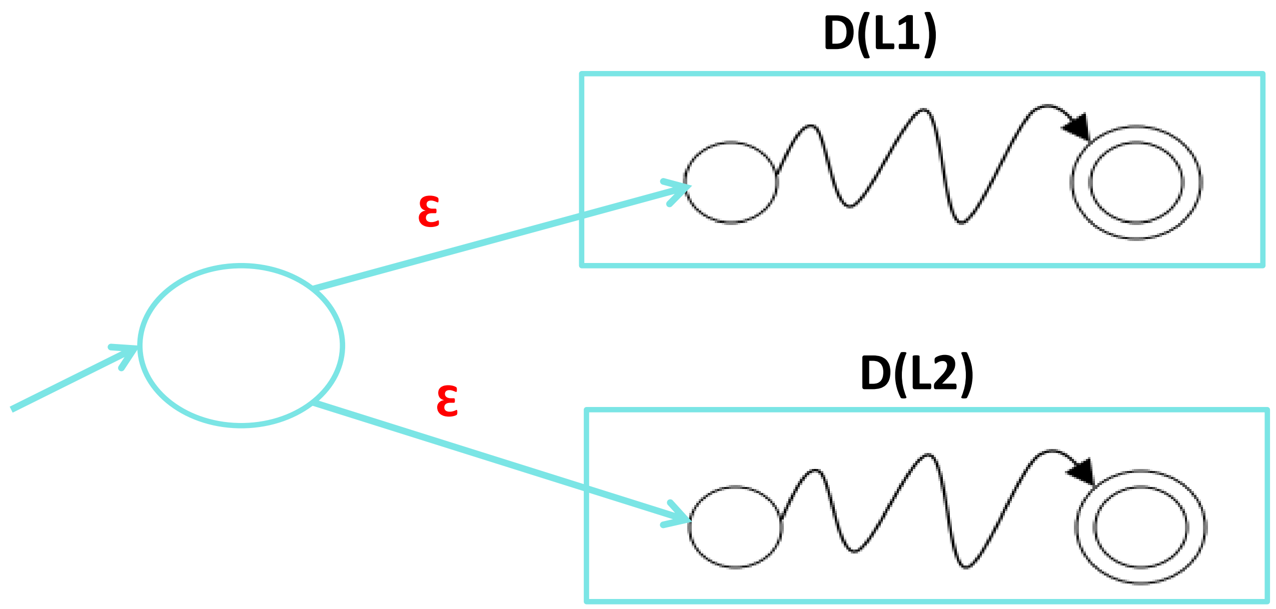
Proof: Construct NFA for L1.L2 as follows:



Closure Properties of Regular Languages

Union: If L1 and L2 are regular languages then L1 ∪ L2 is also a regular language.

Proof: Construct NFA for L1 ∪ L2 as follows:



Closure Properties of Regular Languages

Intersection: If L1 and L2 are regular languages then L1 $\cap$ L2 is also a regular language.

Proof: Using DeMorgan's law:

$$L1 \cap L2 = \overline{\overline{L1} \cup \overline{L2}}$$

L1 and L2 are Regular.

L1' and L2' are Regular.

L1' $\cup$ L2' is Regular.

$\overline{\overline{L1} \cup \overline{L2}}$ is Regular.

L1 $\cap$ L2 is Regular.

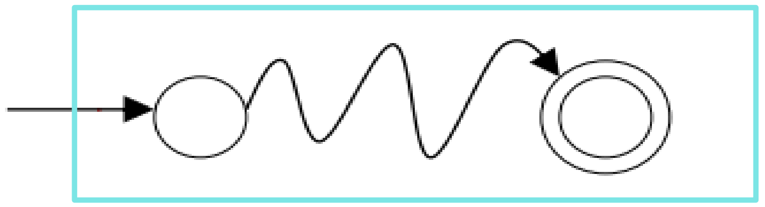
Closure Properties of Regular Languages

Reverse: If L is regular language then L^R is also a regular language.

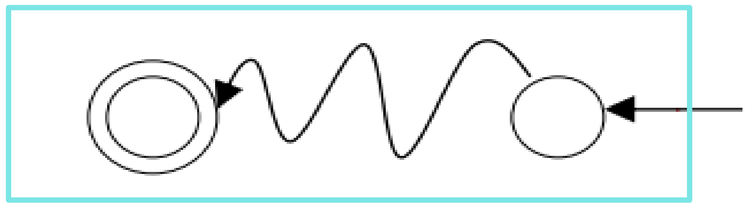
Proof: Construct NFA for L^R as follows:

- Reverse all transitions
- Make the initial state final and final state initial.

D(L)



D(L^R)



Closure Properties of Regular Languages

Difference: If L1 and L2 are regular languages then L1 - L2 is also a regular language.

Proof: $L1 - L2 = L1 \bigcap \overline{L2} = \overline{\overline{L1} \bigcup L2}$

L1 and L2 are Regular.

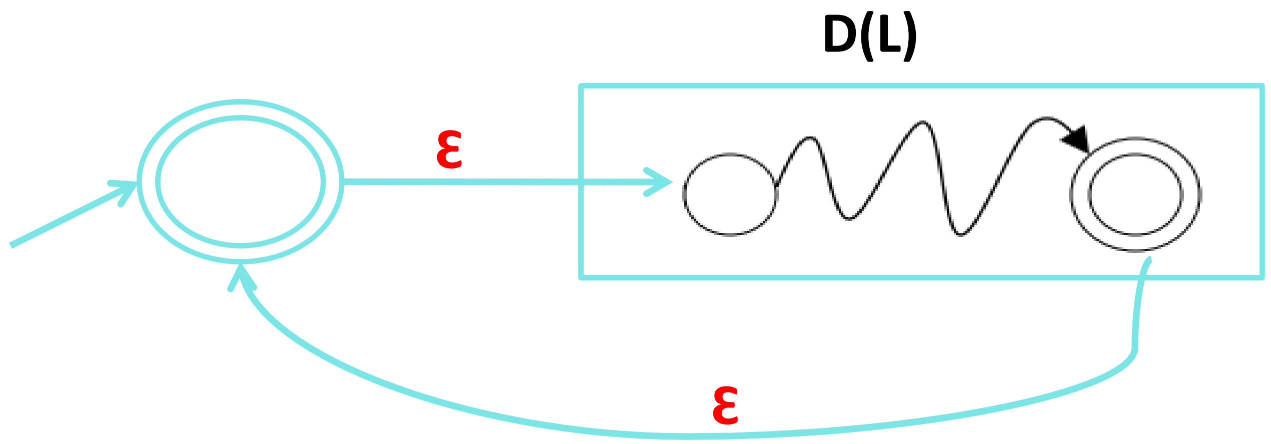
L1’ is Regular.

$\overline{\overline{L1} \bigcup L2}$ is Regular.

Closure Properties of Regular Languages

Kleene Closure: If L is regular language then L* is also a regular language.

Proof: Construct NFA for L* as follows:



Closure Properties of Regular Languages

Homomorphism:

A homomorphism is function h : Σ∗ → Δ∗ defined as follows:

- h(**ε**) = **ε** and for a ∈ Σ, h(a) is any string in Δ∗
- For a = a1a2 . . . an ∈ Σ∗ (n ≥ 2), h(a) = h(a1)h(a2). . . h(an).
- A homomorphism h maps a string a ∈ Σ∗ to a string in Δ∗ by mapping each character of a to a string h(a) ∈ Δ∗
- A homomorphism is a function from strings to strings that "respects" concatenation: for any x, y ∈ Σ ∗ , h(xy) = h(x)h(y). (Any such function is a homomorphism.)

Example. h : {0, 1} → {a, b}∗ where h(0) = ab and h(1) = ba. Then h(0011) = ababbaba

Closure Properties of Regular Languages

Homomorphism:

Regular languages are closed under homomorphism, i.e., if L is a regular language and h is a homomorphism, then h(L) is also regular.

Proof. We will use the representation of regular languages in terms of regular expressions to argue this.

- Define homomorphism as an operation on regular expressions
- Show that L(h(R)) = h(L(R))
- Let R be such that L = L(R). Let R0 = h(R). Then h(L) = L(R0).

Closure Properties of Regular Languages

Homomorphism:

Regular languages are closed under homomorphism, i.e., if L is a regular language and h is a homomorphism, then h(L) is also regular.

Proof. We will use the representation of regular languages in terms of regular expressions to argue this.

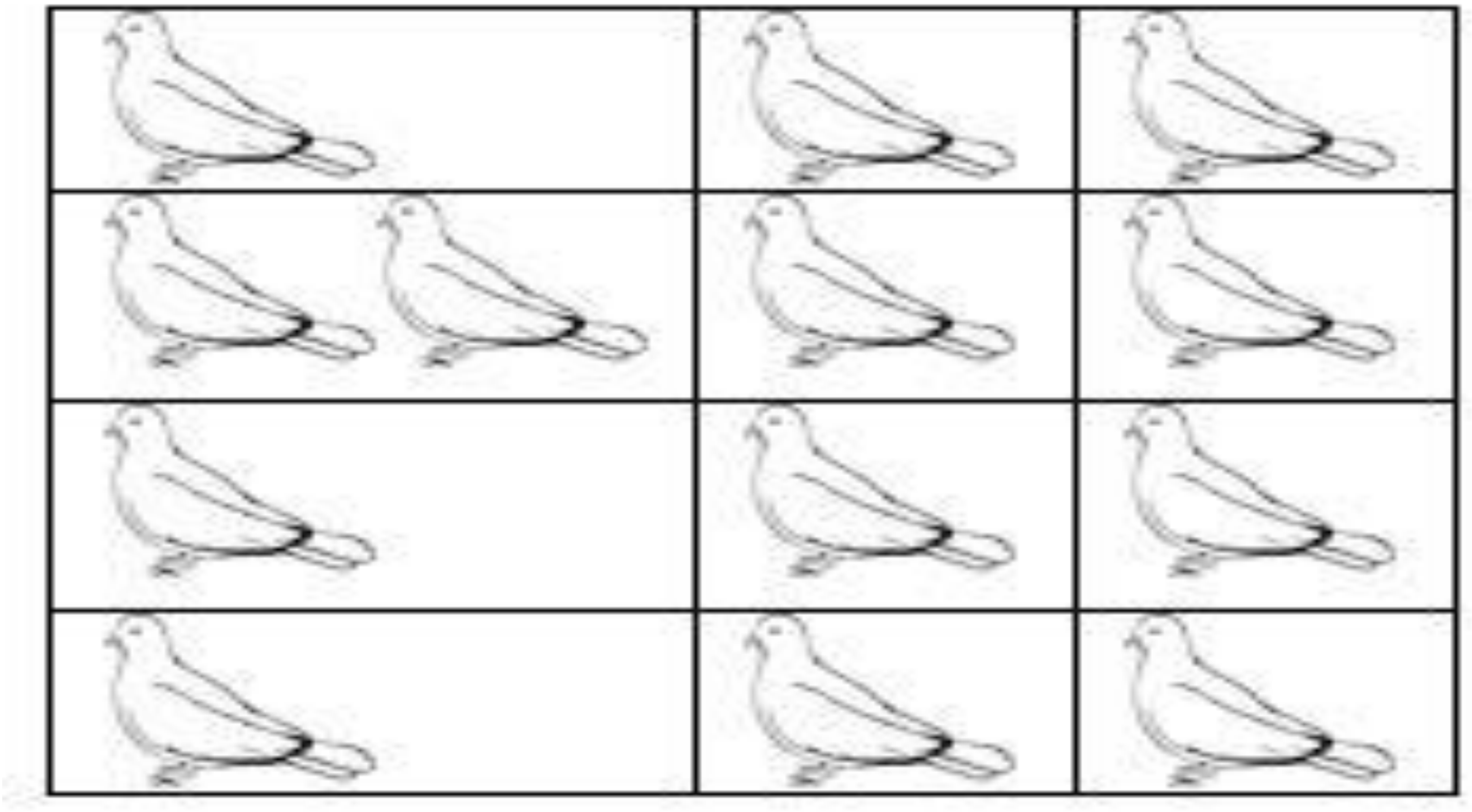
- Define homomorphism as an operation on regular expressions
- Show that L(h(R)) = h(L(R))
- Let R be such that L = L(R). Let R0 = h(R). Then h(L) = L(R0).

Pigeon Hole Priciple

the **pigeonhole principle** states that if *n* items are put into *m* containers, with $n > m$, then at least one container must contain more than one item.

The principle has several generalizations and can be stated in various ways. In a more quantified version: for natural numbers *k* and *m*, if $n = km + 1$ objects are distributed among *m* sets, then the pigeonhole principle asserts that at least one of the sets will contain at least $k + 1$ objects.[4] For arbitrary *n* and *m*, this generalizes to {\displaystyle k+1=\lfloor (n-1)/m\rfloor +1=\lceil n/m\rceil ,} where {\displaystyle \lfloor \cdots \rfloor } and {\displaystyle \lceil \cdots \rceil } denote the floor and ceiling functions, respectively.

Pigeon Hole Priciple



Pigeon Hole Priciple

Theorem –

I) If “A” is the average number of pigeons per hole, where A is not an integer then
At least one pigeon hole contains **ceil[A]** (smallest integer greater than or equal to A) pigeons
Remaining pigeon holes contains at most **floor[A]** (largest integer less than or equal to A) pigeons

II) We can say as, if n + 1 objects are put into n boxes, then at least one box contains two or more objects.
The abstract formulation of the principle: Let X and Y be finite sets and let be a function.
If X has more elements than Y, then f is not one-to-one.
If X and Y have the same number of elements and f is onto, then f is one-to-one.
If X and Y have the same number of elements and f is one-to-one, then f is onto.
Pigeonhole principle is one of the simplest but most useful ideas in mathematics. We will see more applications that proof of this theorem.

Pigeon Hole Priciple

Example – 2: A bag contains 10 red marbles, 10 white marbles, and 10 blue marbles. What is the minimum no. of marbles you have to choose randomly from the bag to ensure that we get 4 marbles of same color?**Solution:** Apply pigeonhole principle.

No. of colors (pigeonholes) n = 3

No. of marbles (pigeons) K+1 = 4

Therefore the minimum no. of marbles required = Kn+1

By simplifying we get Kn+1 = 10.

Verification: ceil[Average] is [Kn+1/n] = 4

[Kn+1/3] = 4

Kn+1 = 10

i.e., 3 red + 3 white + 3 blue + 1(red or white or blue) = 10

Pumping Lemma for Regular Languages

Objective of the Topic

The objective of the topic is to make the student able to:

- Apply pumping lemma.
- Prove that a language is not regular

Pumping Lemma for Regular Languages

Topic mapping with Course Outcome

Topic	CO1	CO2	CO3	CO4	CO5
Pumping Lemma for Regular Languages	-	3	-	-	-

Pumping Lemma for Regular Languages

- L1 = {$a^m b^n$, m,n >=0} is a regular language.
- L2 = {$a^n b^n$, n >=0} is not a regular language.
- L3 = {w|w has an equal number of 0s and 1s} is not a regular language.
- Pumping Lemma is used to prove that a language is not regular.
- It can not be used to prove that a language is regular.

Pumping Lemma

- If L is a Regular Language, then there is a number m (the pumping length) where if w is any string in L of length at least m, then w may be divided into 3 pieces, **w = xyz**, satisfying the following conditions:

  a. **$xy^iz \in L$**, For each $i \geq 0$

  b. **$|y| > 0$**, and

  c. **$|xy| \leq m$**.

Pumping Lemma for Regular Languages

Process to use Pumping lemma

- Assume that **L** is regular.
- So, the pumping lemma should hold for **L**.
- Use the pumping lemma to obtain a contradiction
  - Select **w** such that **$|w| \geq m$**
  - Divide w into x, y and z
    - Select **y** such that **$|y| \geq 1$**
    - Select **x** such that **$|xy| \leq m$**
    - Assign the remaining string to **z.**
  - Pump y such that **$xy^iz \notin L$.**
- By contradiction say L is not regular.

Pumping Lemma for Regular Languages

Prove that **$L = \{a^n b^n \mid n \geq 0\}$** is not regular.

Solution –

Assume that **L** is regular and m is the number of states.

Let $w = a^m b^m$. Thus $|w| = 2m \geq m$.

By pumping lemma, let $w = xyz$, where $|xy| \leq m$.

Let $x = a^p$, $y = a^q$, and $z = a^r b^m$, where $p + q + r = m$, $p \neq 0$, $q \neq 0$, $r \neq 0$. Thus $|y| \neq 0$.

Let $k = 2$. Then $xy^2z = a^p a^{2q} a^r b^m$.

Number of as = $(p + 2q + r) = (p + q + r) + q = m + q$

Hence, $xy^2z = a^{m+q} b^m$. Since $q \neq 0$, xy^2z is not of the form $a^n b^n$.

Thus, xy^2z is not in L. Hence L is not regular.

https://www.yumpu.com/en/document/read/31123078/visualizing-the-pumping-lemma-for-regular-

Decision Properties of Regular Languages

Following properties are decidable for Finite Automata.

- **Emptiness**

  To determine whether FA accepts any language or not?

- **Finiteness**

  To determine whether Language accepted by FA is finite or infinite?

- **Membership**

  To determine whether string w is accepted by FA or not?

- **Equality**

  To determine whether two FA are accepting same language.

Decision Properties of Regular Languages

Emptiness:

Step-1: Remove unreachable states.

Step 2: If the resulting machine contains at least one final states, then the finite automata accepts language.

Step 3: if there is no final state, then finite automata accepts empty language.

Finiteness:

Step-1: Remove unreachable states.

Step-2: Remove dead states.

Step-3: If the resulting machine contains loops or cycles then the FA accepts infinite language.

Step-4: If the resulting machine do not contain loops or cycles then the FA accepts finite language.

Decision Properties of Regular Languages

Membership:

Let M is a finite automata that accepts some strings over an alphabet, and let 'w' be any string defined over the alphabet, if there exist a transition path in M, which starts at initial state & ends in anyone of the final state, then string 'w' is a member of M, otherwise 'w' is not a member of M.

Equality:

Two finite state automata M1 & M2 is said to be equal if and only if, they accept the same language.

Minimise the finite state automata and the minimal DFA will be unique.

Limitations of FA

- FA can not be constructed for
  - Context Free Grammar
  - Context sensitive Grammar
  - Recursive Enumerable Grammar
- Calculation can not be done with FA.
- There is no storage in FA.
- FA can not modify the input (as Turing Machine does).

Video Links

My Video Channel:

https://www.youtube.com/playlist?list=PLKhyqvN-JcP5AF4UzRz9wxFVBm63igm7y

- NPTEL Video Links

  https://youtu.be/vQeROGcwQs4

  https://youtu.be/XxuH_K-wzpg

  https://youtu.be/C7LqbEunJ1Y

NIET Greater Noida
GET FUTURE READY

Daily Quiz

1. If L1 and L2 are regular languages is/are also regular language(s).

a) L1 + L2
b) L1L2
c) L1’
d) All of above

2. Every regular expression can be expressed as CFG but every CFG cannot be expressed as a regular expression. this statement is :

a) True
b) False
c) depends on language
d) None of the option

Daily Quiz

3. In pumping Lemma, If we select a string w such that w∈L, and w=xyz. Which of the following portions cannot be an empty string?

A. x

B. y

C. z

D. all of the mentioned

4.The total number of states required to automate the given regular expression(00)*(11)*

A. 3

B. 5

C. 6

D. 2

Daily Quiz

5. Moore machine produce the output string of length ---------- corresponding to n length input string

A. n

B. n+1

C. n-1

D. 2n

6. Which of the following does not represents the given language over {a,b} ?

A. a+ab

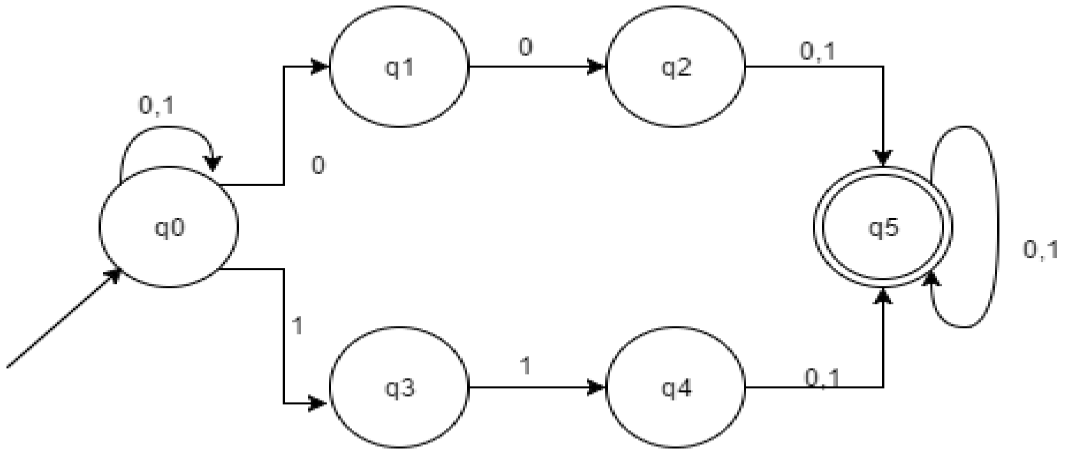
B. {a} U {ab}

C. {a} U {a}{b}

D. {a} ^ {ab}

Daily Quiz

7. The given NFA corresponds to which of the following Regular expressions?



A. $(0+1)^*(00+11)(0+1)^+$

B. $(0+1)^*(00+11)^*(0+1)^*$

C. $(0+1)^*(00+11)(0+1)$

D. $(0+1)(00+11)(0+1)^*$

Daily Quiz

8. Regular grammar is
   - a) context free grammar
   - b) non context free grammar
   - c) English grammar
   - d) none of the mentioned

9. Which of the following is true?
   - a) Every subset of a regular set is regular
   - b) Every finite subset of non-regular set is regular
   - c) The union of two non regular set is not regular
   - d) Infinite union of finite set is regular

10. Regular expressions are closed under
   - a) Union
   - b) Intersection
   - c) Kleen star
   - d) All of the mentioned

Weekly Assignment

1. Find a regular expression corresponding to each of the following subsets of {a. b}. **[CO2]**

- The set of all strings containing exactly 2a’s.
- The set of all strings containing at least 2a’s.
- The set of all strings containing at most 2a's.
- The set of a]] strings containing the substring aa.

2. Find the sets represented by the following regular expressions. **[CO2]**

- $(a + b)^*(aa + bb + ab + ba)^*$
- $(aa)^* + (aaa)^*$
- $(1 + 01 + 001)^*(\in + 0 + 00)$
- $a + b(a + b)^*$

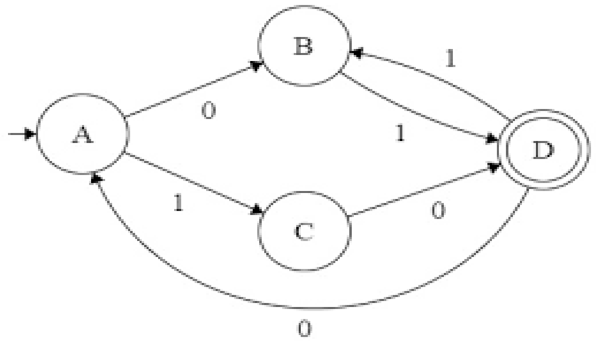
Weekly Assignment

3. Prove the identity: $(a^*ab + ba)^*a^* = (a + ab + ba)^*$
[CO2]

4. State Pumping Lemma for Non-Regular languages. Prove that the language L= $(a^q$ where q is a prime number$\}$ is a Non regular language.
[CO2]

5. Show that $L = \{aib^jc^k \mid k>i+j\}$ is not regular.
[CO2]

6. Find the regular expression corresponding to the automaton given in Fig.



Weekly Assignment

7. State and prove Arden's theorem. Give a regular expression for the Finite Automaton M=[{A ,B, C, D}, {a, b}, δ, A, {A}] where δ is given below: **[CO2]**

– δ (A, a) = B	δ (A, b) = C
– δ (B, a) = D	δ (B, b) = A
– δ (C, a} = A	δ (C, b) = D
– δ (D, a) = D	δ (D, b) = D

8. Explain the Chomsky hierarchy of formal languages. Define the relationship between languages and corresponding machine. **[CO2]**

Weekly Assignment

9. Design a Mealy Machine which reads input from {0, 1} and outputs are EVEN or ODD according to total number of 1's are even or odd. Also convert that Mealy Machine into Corresponding Moore Machine.
   [CO2]

10. Construct a finite automaton recognizing L(G), where G is the grammar $S \rightarrow a S \mid bA \mid b$ and $A \rightarrow aA \mid bS \mid a$.
    [CO2]

MCQ s

Which of the following are decision properties?

A. Emptiness

B. Infiniteness

C. Membership

D. All of the mentioned

In pumping lemma theorem (x y^n z) the range of n is

A. n=1, 2, 3, 4……….

B. n=0, 1, 2, 3, 4……….

C. n=…….-3,-2,-1, 0, 1, 2, 3, 4……

D. n=…….-3,-2,-1, 1, 2, 3, 4……

MCQ s

Let S and T be language over ={a,b} represented by the regular expressions (a+b*)* and (a+b)*, respectively. Which of the following is true?

A. S is a subset of T

B. T is a subset of S

C. S=T

D. S intersection T=Ø

The logic of pumping lemma is good example of

A. the pigeon hole principle

B. divide and conquer

C. recursion

D. iteration

MCQ s

Let a Grammar G is defined by the production S→aS | bS , then language generated by Grammar is defined as

A. Φ

B. ∈

C. {a,b}*

D. {a,b}+

Which of the following is true?

A. (01)*0 = 00(10)*

B. (0+1)*0(0+1)*1(0+1) = (0+1)*01(0+1)*

C. (0+1)*01(00+1)*+1*0* = (0+1)*

D. All of the mentioned

MCQ s

Which among the following looks similar to the given expression? ((0+1).(0+1))*

A. {xϵ {0,1} *|x is all binary number with even length}

B. {xϵ {0,1} |x is all binary number with even length}

C. {xϵ {0,1} *|x is all binary number with odd length}

D. {xϵ {0,1} |x is all binary number with odd length}

Which of the following represents a language which has no pair of consecutive 1's if ∑= {0,1}?

A. (0+10)*(1+ε)

B. (0+10)*(1+ε)*

C. (0+101)*(0+ε)

D. (1+010)*(1+ε)

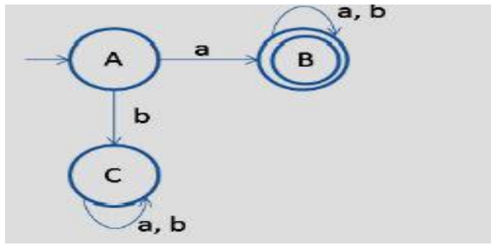
Old Question Papers

- https://drive.google.com/drive/folders/19Eia3VHCl3627foiH6V_j-p4X9ZkyyC7?usp=sharing

NIET Greater Noida GET FUTURE READY

Expected Questions for University Exam

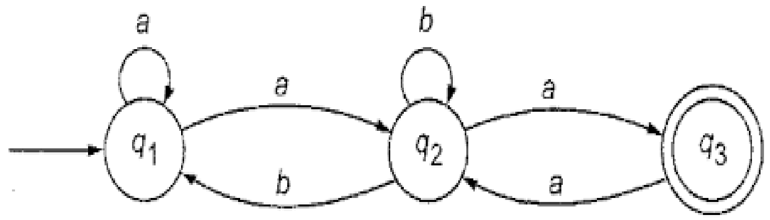
1. Find regular expression over ∑={a,b} for all strings containing either the sub string ‘aaa’ or ‘bbb’.
2. Write regular expression for set of all strings such that number of a’s divisible by 3 over Σ = {a,b}.
3. State the pumping lemma theorem for regular languages.
4. Convert the FA given below to left linear grammar.



5. Show that L = {$a^i b^j c^k$ | k > i + j} is not regular.

Expected Questions for University Exam

6. Find the regular expression for the given FA.


$$\begin{tikzcd} \arrow[r] & q_1 \arrow[loop above, "a"] \arrow[r, bend left, "a"] & q_2 \arrow[loop above, "b"] \arrow[l, bend left, "b"] \arrow[r, bend left, "a"] & \boxed{q_3} \arrow[l, bend left, "a"] \end{tikzcd}$$

7. For the two regular expressions :

r1 = a* + b*   r2 = ab* + ba* + b*a + (a*b)*

a. Find a string corresponding to r2 but not to r1 and

b. Find a string corresponds to both r1 and r2.

Summary

- Regular expressions are used for regular languages.
- Using Arden’s theorem a FA is converted into regular expression.
- Regular expression is converted into Ɛ-NFA.
- Pumping lemma is used to prove that a language is not regular.
- Regular languages are closed under union, intersection, complement, reverse, concatenation and kleene closure.

References

- Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, *32*(1), 60-65.
- Linz, P. (2006). *An introduction to formal languages and automata*. Jones & Bartlett Learning.
- Mishra, K. L. P., & Chandrasekaran, N. (2006). *Theory of Computer Science: Automata, Languages and Computation*. PHI Learning Pvt. Ltd..

References

- Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, *32*(1), 60-65.
- Linz, P. (2006). *An introduction to formal languages and automata*. Jones & Bartlett Learning.
- Mishra, K. L. P., & Chandrasekaran, N. (2006). *Theory of Computer Science: Automata, Languages and Computation*. PHI Learning Pvt. Ltd..