KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

LESSON PLAN
DEPARTMENT OF MATHEMATICS

Name of the faculty : Y.Sangeetha

Class : II M.Sc Mathematics

Subject : Formal languages and Automata Theory

Subject Code : 17MMP305A

S.No	Lecture Duration	Topics to be covered	Support Materials
		Unit – I	
1.	1	Definition of an Automation with examples	T1: Ch 3: Pg: 71-72
2.	1	Description of Finite Automaton	T1: Ch 3: Pg: 73
3.	1	Transition systems	T1: Ch 3: Pg: 74
4.	1	Property of transition functions	T1: Ch 3: Pg: 75
5.	1	Finite Automaton	T1: Ch 3: Pg: 77
6.	1	Acceptability of a string by a finite Automaton	T1: Ch 3: Pg: 77
7.	1	Non deterministic finite automaton	R2: Ch 3: Pg: 147-148
8.	1	The equivalence of DFA	T1: Ch 3: Pg: 80
9.	1	The equivalence of NDFA	T1: Ch 3: Pg: 80
10.	1	Recapitulation and discussion of possible question	
Total No of Hours Planned For Unit 1– 10 hours			
		Unit – II	
1.	1	Introduction on Formal Languages	T1: Ch 4: Pg: 107
2.	1	Basic Definitions and examples	T1: Ch 4: Pg: 107
3.	1	Chomsky classification of Languages	T1: Ch 4: Pg: 120-122
4.	1	Languages and their relation	T1: Ch 4: Pg: 123
5.	1	Recursive and Recursively Enumerable sets	T1: Ch 4: Pg: 124
6.	1	Continuation on Recursive and Recursively Enumerable sets	T1: Ch 4: Pg: 125
7.	1	Operations on Languages.	T1: Ch 4: Pg: 126
8.	1	Continuation on Operations on Languages.	R2:Ch 3:119-120
9.	1	Recapitulation and discussion of possible question	

Total No of Hours Planned For Unit 1I – 9 hours			
Unit – III			
1.	1	Definitions and examples of Regular expressions	R1: Ch3: Pg: 83
2.	1	Identities for regular expressions	T1: Ch5: Pg: 126
3.	1	Finite Automata and Regular expressions.	R1: Ch3: Pg: 90
4.	1	Transition system containing $\wedge$ - moves	T1: Ch5: Pg: 140
5.	1	Conversion of Nondeterministic Systems to Deterministic Systems	T1: Ch 5: Pg: 146-147
6.	1	Algebraic method using Arden's theorem	T1: Ch 5: Pg: 148-152
7.	1	Constructions of finite automata	T1: Ch 5: Pg: 153-156
8.	1	Equivalence of two finite automata	T1: Ch 5: Pg: 157-159
9.	1	Equivalence of two regular expressions	T1: Ch 5: Pg: 160-161
10.	1	Recapitulation and discussion of possible question	
Total No of Hours Planned For Unit 1II – 10 hours			
Unit – IV			
1.	1	Basic Definitions and examples of Regular sets	T1: Ch 5: Pg: 161
2.	1	Pumping Lemma for Regular sets	T1: Ch 5: Pg: 162
3.	1	Applications of Pumping Lemma	T1: Ch 5: Pg: 163-164
4.	1	Closure Property of Regular sets	T1: Ch 5: Pg: 165-166
5.	1	Basic Definitions and examples of Regular grammars	T1: Ch 5: Pg: 167
6.	1	Continuation on Regular grammars	T1: Ch 5: Pg: 167
7.	1	Construction of a regular grammar	T1: Ch 5: Pg: 168
8.	1	Construction of a Transition system	T1: Ch 5: Pg: 169
9.	1	Examples of construction of a Transition system	T1: Ch 5: Pg: 170
10.	1	Recapitulation and discussion of possible question	
Total No of Hours Planned For Unit 1V – 10 hours			
Unit – V			
1.	1	Context free Languages	R1: Ch5: Pg: 171-172
2.	1	Derivation trees	T1: Ch 6: Pg: 181-185
3.	1	Context free grammars	R3: Ch 16: Pg: 427-430
4.	1	Ambiguity in Context free grammars	R3: Ch 16: Pg: 457-460
5.	1	Simplification of Context free grammars	T1: Ch 6: Pg: 189-192
6.	1	Recapitulation and discussion of possible questions	
7.	1	Discussion of previous ESE question papers	
8.	1	Discussion of previous ESE question papers	
9.	1	Discussion of previous ESE question papers	
Total No of Hours Planned For Unit V -9 hours			

SUGGESTED READINGS

TEXTBOOK

1. Mishra, K. L. P and Chandrasekaran, N.,(2008). Theory of Computer Science, Automata

Languages and Computation, Prentice Hall of India, New Delhi.

REFERENCES

1. John E. Hopcroft and J.D. Ullman, (2006). Introduction to Automata theory, Languages and Computation, Third Edition, Prentice Hall.ofIndia,New Delhi.

2. Aho A.V., and Ullman J.D., (2002). Principles of compiler design, Narosa Publishing Company, London.

3. Rakesh Duke, Adesh Pandey and RiTu Gupta, (2007). Discrete Structures and Automata theory. Narosa Publishing Company, New Delhi.

.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021. **Semester – III**

SYLLABUS **L T P C**
4 0 0 4

17MMP305A FORMAL LANGUAGES AND AUTOMATA THEORY

Scope: This course makes the students to understand various aspects of automata theory and Grammar, relationship between them those have wide applications in the field of computers.

Objectives: To understand the formulation of DFA and NDFA, Chomsky classification of Languages, regular expression, pumping Lemma and get familiar with context free grammars.

UNIT I

Definition of an Automation - Description of Finite Automaton – Transition systems - Property of transition functions - Acceptability of a string by a finite Automaton - Non deterministic finite automaton - The equivalence of DFA and NDFA.

UNIT II

Formal Languages - Basic Definitions and examples - Chomsky classification of Languages - Languages and their relation - Recursive and Recursively Enumerable sets- Operations on Languages.

UNIT III

Regular expressions - Finite Automata and Regular expressions.

UNIT IV

Pumping Lemma for Regular sets - Applications of Pumping Lemma - Closure Property of Regular sets - Regular sets and Regular grammars.

UNIT V

Context free Languages and Derivation trees - Ambiguity in Context free grammars - Simplification of Context free grammars (examples only).

SUGGESTED READINGS

TEXTBOOK

1. Mishra, K. L. P and Chandrasekaran, N.,(2008). Theory of Computer Science, Automata Languages and Computation, Prentice Hall of India, New Delhi.

REFERENCES

1. John E. Hopcroft and J.D. Ullman, (2006). Introduction to Automata theory, Languages and Computation, Third Edition, Prentice Hall.of India,New Delhi.

2. Aho A.V., and Ullman J.D., (2002). Principles of compiler design, Narosa Publishing Company, London.

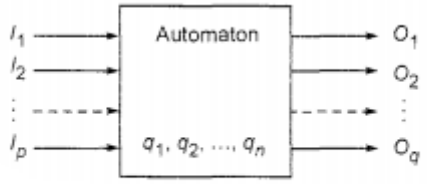
3. Rakesh Duke, Adesh Pandey and RiTu Gupta, (2007). Discrete Structures and Automata theory. Narosa Publishing Company, New Delhi.

UNIT-I

Definition of an Automation - Description of Finite Automaton – Transition systems - Property of transition functions - Acceptability of a string by a finite Automaton - Non deterministic finite automaton - The equivalence of DFA and NDFA.

DEFINITION OF AN AUTOMATON

We shall give the most general definition of an automaton and later modify it to computer applications. An automaton is defined as a system where energy, materials and information are transformed, transmitted and used for performing some functions without direct participation of man. Examples are automatic machine tools, automatic packing machines, and automatic photo printing machines.



Model of a discrete automaton.

The characteristics of automaton are now described.

(i) *Input.* At each of the discrete instants of time $t_1, t_2, \ldots, t_m$ the input values $I_1, I_2, \ldots, I_p$, each of which can take a finite number of fixed values from the input alphabet Σ, are applied to the input side of the model shown in Fig. 3.1.

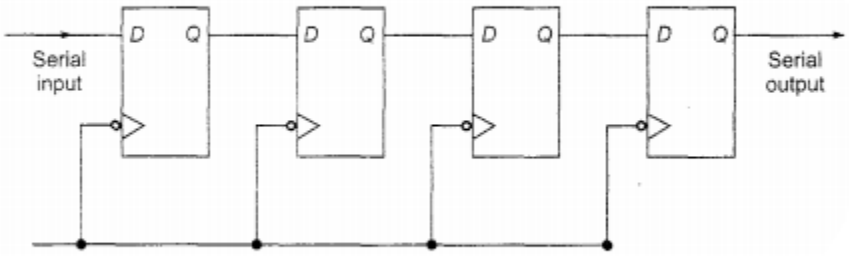
(ii) *Output.* $O_1, O_2, \ldots, O_q$ are the outputs of the model, each of which can take a finite number of fixed values from an output O.

(iii) *States.* At any instant of time the automaton can be in one of the states $q_1, q_2, \ldots, q_n$.

(iv) *State relation.* The next state of an automaton at any instant of time is determined by the present state and the present input.

(v) *Output relation.* The output is related to either state only or to both the input and the state. It should be noted that at any instant of time the automaton is in some state. On 'reading' an input symbol, the automaton moves to a next state which is given by the state relation.

Consider the simple shift register s
and study its operation.



A 4-bit serial shift register using D flip-flops.

DESCRIPTION OF A FINITE AUTOMATON

Definition Analytically, a finite automaton can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

(i) Q is a finite nonempty set of states.
(ii) Σ is a finite nonempty set of inputs called the *input alphabet*.
(iii) δ is a function which maps $Q \times \Sigma$ into Q and is usually called the *direct transition function*. This is the function which describes the change of states during the transition. This mapping is usually represented by a transition table or a transition diagram.
(iv) $q_0 \in Q$ is the initial state.
(v) $F \subseteq Q$ is the set of final states. It is assumed here that there may be more than one final state.

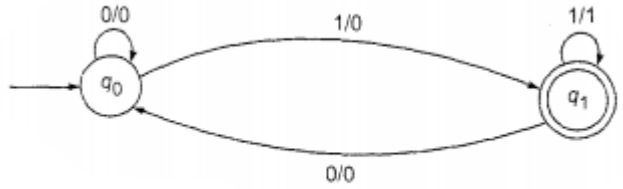
(i) *Input tape.* The input tape is divided into squares, each square containing a single symbol from the input alphabet Σ. The end squares of the tape contain the endmarker ¢ at the left end and the endmarker $ at the right end. The absence of endmarkers indicates that the tape is of infinite length. The left-to-right sequence of symbols between the two endmarkers is the input string to be processed.

(ii) *Reading head.* The head examines only one square at a time and can move one square either to the left or to the right. For further analysis, we restrict the movement of the R-head only to the right side.

(iii) *Finite control.* The input to the finite control will usually be the symbol under the R-head, say a, and the present state of the machine, say q, to give the following outputs: (a) A motion of R-head along the tape to the next square (in some a null move, i.e. the R-head remaining to the same square is permitted); (b) the next state of the finite state machine given by $\delta(q, a)$.

TRANSITION SYSTEMS

A transition graph or a transition system is a finite directed labelled graph in which each vertex (or node) represents a state and the directed edges indicate the transition of a state and the edges are labelled with input/output.



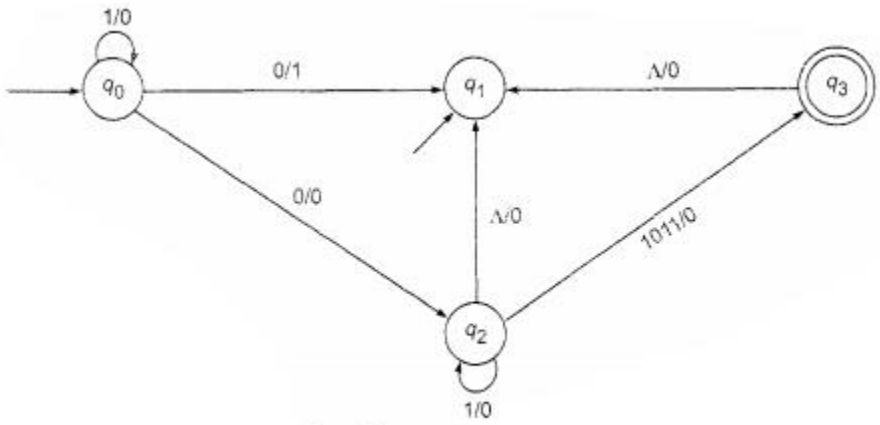
A transition system.

Definition A transition system is a 5-tuple $(Q, \Sigma, \delta, Q_0, F)$, where

(i) Q, Σ and F are the finite nonempty set of states, the input alphabet, and the set of final states, respectively, as in the case of finite automata;
(ii) $Q_0 \subseteq Q$, and Q_0 is nonempty; and
(iii) δ is a finite subset of $Q \times \Sigma^* \times Q$.

Definition A transition system accepts a string w in Σ^* if

(i) there exists a path which originates from some initial state, goes along the arrows, and terminates at some final state; and
(ii) the path value obtained by concatenation of all edge-labels of the path is equal to w.



Transition system

Determine the initial states, the final states, and the acceptability of 101011, 111010.

Solution

The initial states are q_0 and q_1. There is only one final state, namely q_3.

The path-value of $q_0q_0q_2q_3$ is 101011. As q_3 is the final state, 101011 is accepted by the transition system. But, 111010 is not accepted by the transition system as there is no path with path value 111010.

PROPERTIES OF TRANSITION FUNCTIONS

Property 1 $\delta(q, \Lambda) = q$ is a finite automaton. This means that the state of the system can be changed only by an input symbol.

Property 2 For all strings w and input symbols a,

$$\delta(q, aw) = \delta(\delta(q, a), w)$$
$$\delta(q, wa) = \delta(\delta(q, w), a)$$

This property gives the state after the automaton consumes or reads the first symbol of a string aw and the state after the automaton consumes a prefix of the string wa.

Prove that for any transition function δ and for any two input strings x and y,

$$\delta(q, xy) = \delta(\delta(q, x), y)$$

Proof By the method of induction on $|y|$, i.e. length of y.

Basis: When $|y| = 1$, $y = a \in \Sigma$

$$\text{L.H.S. of (3.1)} = \delta(q, xa)$$
$$= \delta(\delta(q, x), a) \quad \text{by Property 2}$$
$$= \text{R.H.S. of (3.1)}$$

Assume the result, for all strings x and strings y with $|y| = n$. Let y be a string of length $n + 1$. Write $y = y_1a$ where $|y_1| = n$.

$$\text{L.H.S. of (3.1)} = \delta(q, xy_1a) = \delta(q, x_1a), \quad x_1 = xy_1$$
$$= \delta(\delta(q, x_1), a) \quad \text{by Property 2}$$
$$= \delta(\delta(q, xy_1), a)$$
$$= \delta(\delta(\delta(q, x), y_1), a) \quad \text{by induction hypothesis}$$
$$\text{R.H.S. of (3.1)} = \delta(\delta(q, x), y_1a)$$
$$= \delta(\delta(\delta(q, x), y_1), a) \quad \text{by Property 2}$$

Hence, L.H.S. = R.H.S.

ACCEPTABILITY OF A STRING BY A FINITE AUTOMATON

Definition A string x is accepted by a finite automaton

$$M = (Q, \Sigma, \delta, q_0, F)$$

if $\delta(q_0, x) = q$ for some $q \in F$.

This is basically the acceptability of a string by the final state.

Consider the finite state machine whose transition function δ is given by Table in the form of a transition table. Here, $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$. Give the entire sequence of states for the input string 110001.

Transition Function

State	Input	
	0	1
→ q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Solution

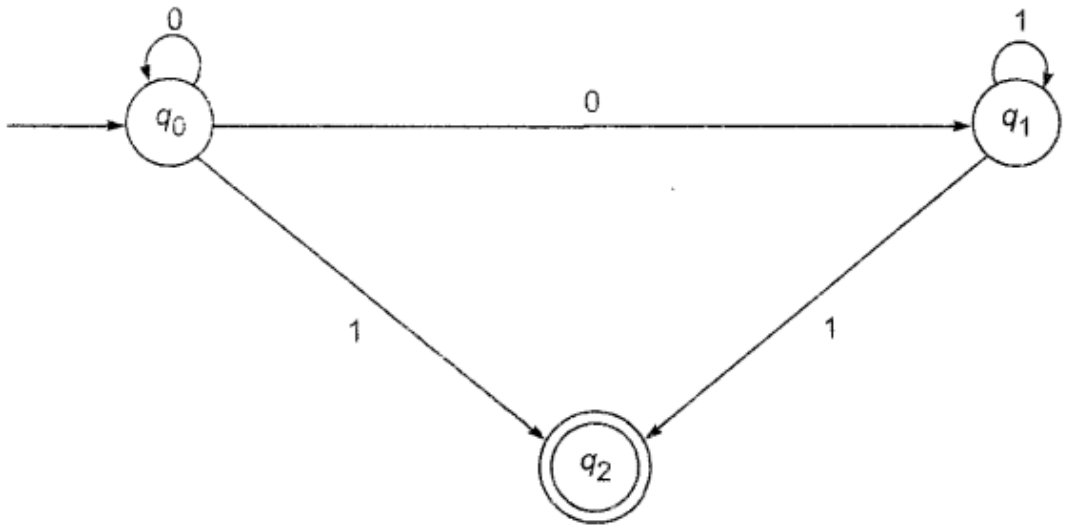
$$\begin{aligned}\delta(q_0, 110101) &= \delta(q_1, 10101)\\ &= \delta(q_0, 0101)\\ &= \delta(q_2, 101)\\ &= \delta(q_3, 01)\\ &= \delta(q_1, 1)\\ &= \delta(q_0, \Lambda)\\ &= q_0\end{aligned}$$

Hence,

$$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_1 \xrightarrow{1} q_0$$

The symbol ↓ indicates that the current input symbol is being processed by the machine.

NONDETERMINISTIC FINITE STATE MACHINES



Transition system representing nondeterministic automaton.

A nondeterministic finite automaton (NDFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

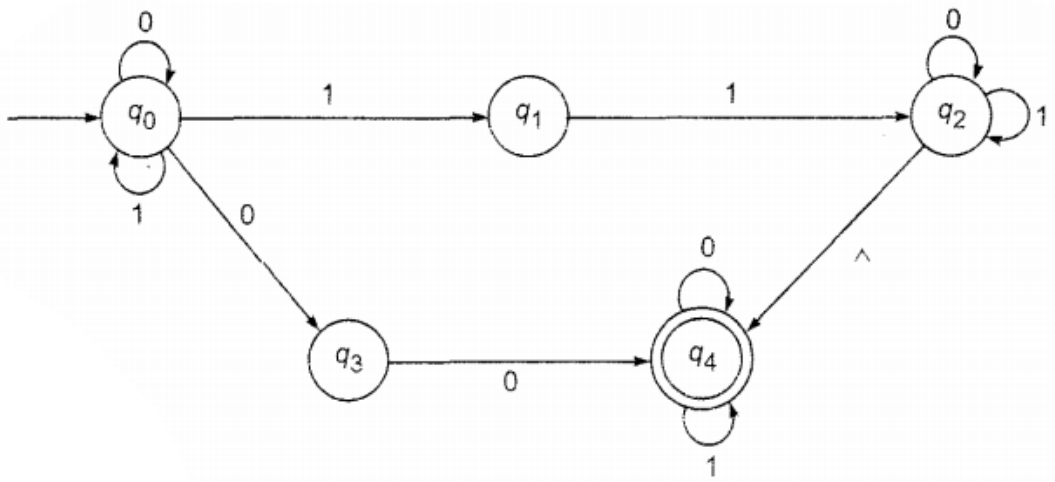
(i) Q is a finite nonempty set of states;

(ii) Σ is a finite nonempty set of inputs;

(iii) δ is the transition function mapping from $Q \times \Sigma$ into 2^Q which is the power set of Q, the set of all subsets of Q;

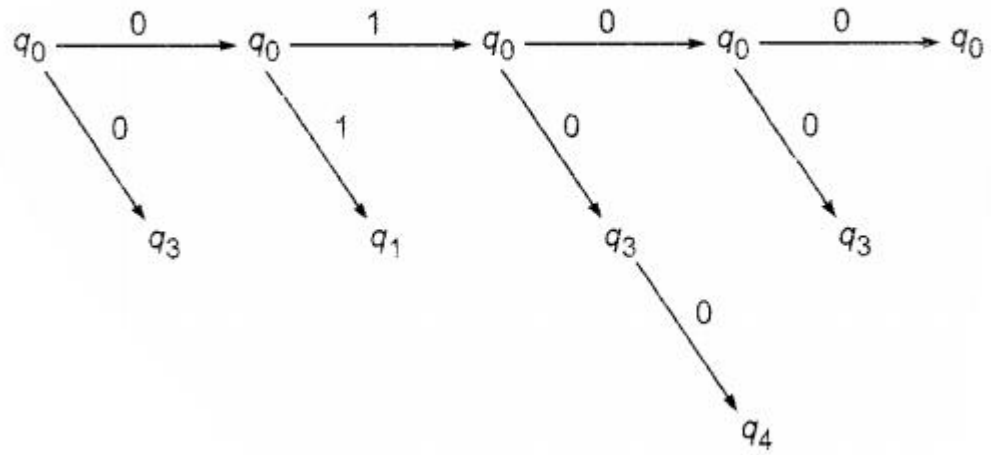
(iv) $q_0 \in Q$ is the initial state; and

(v) $F \subseteq Q$ is the set of final states.



Transition system for a nondeterministic automaton.

Transition system for a nondeterministic automaton.



States reached while processing 0100.

Definition A string $w \in \Sigma^*$ is accepted by NDFA M if $\delta(q_0, w)$ contains some final state.

Definition The set accepted by an automaton M (deterministic or nondeterministic) is the set of all input strings accepted by M. It is denoted by $T(M)$.

THE EQUIVALENCE OF DFA AND NDFA

We naturally try to find the relation between DFA and NDFA. Intuitively we now feel that:

(i) A DFA can simulate the behaviour of NDFA by increasing the number of states. (In other words, a DFA $(Q, \Sigma, \delta, q_0, F)$ can be viewed as an NDFA $(Q, \Sigma, \delta', q_0, F)$ by defining $\delta'(q, a) = \{\delta(q, a)\}$.)

(ii) Any NDFA is a more general machine without being more powerful.

Construct a deterministic automaton equivalent to

$$M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$$

TABLE State Table

State/Σ	0	1
→ (q_0)	q_0	q_1
q_1	q_1	q_0, q_1

Solution

For the deterministic automaton M_1,

(i) the states are subsets of $\{q_0, q_1\}$, i.e. $\emptyset$, $[q_0]$, $[q_0, q_1]$, $[q_1]$;
(ii) $[q_0]$ is the initial state;
(iii) $[q_0]$ and $[q_0, q_1]$ are the final states as these are the only states containing q_0; and
(iv) δ is defined by the state table

State Table of M_1

State/Σ	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$[q_0]$	$[q_0]$	$[q_1]$
$[q_1]$	$[q_1]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

Possible Questions

Part B(6 marks)

1. Explain characteristic of finite automaton.
2. Prove that for any transition function δ and for any two input strings x and y, δ(q, xy) = δ(δ(q, x), y).
3. Consider the finite state machine whose transition function δ is given by table in the form of a transition table. Here, Q = $\{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0,1\}$, F = $\{q_0\}$. Give the entire sequence of states for the input string 110001.

State	Input	
	0	1
→ q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

4. Construct a deterministic finite automaton equivalent toM =($\{q_0, q_1, q_2, q_3\}$, {0,1}), δ,q_0,{q_3})where δ is given by table

State table

State/Σ	a	b
→q_0	q_0, q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_3
(q_3)		q_2

5. Define DFA and acceptability of a string by it with example.
6. Construct a DFA accepting all strings over {a,b} ending in ab.

7. Write a note on deterministic and nondeterministic model with example.

8. Find a DFA equivalent to M=($\{q_0, q_1, q_2\}$,{a,b},δ,q_0,$\{q_2\}$) and δ is given by

δ (q_0, a)=$\{q_0, q_1, q_2\}$,δ (q_0, b)= $\{q_2\}$,δ (q_1, a)= $\{q_0\}$,δ (q_1, b)= $\{q_1\}$, δ (q_2, a)= $\emptyset$,

δ (q_2, b)= $\{q_0, q_1\}$.

9. Construct a DFA equivalent to an NDFA whose transition table is given below

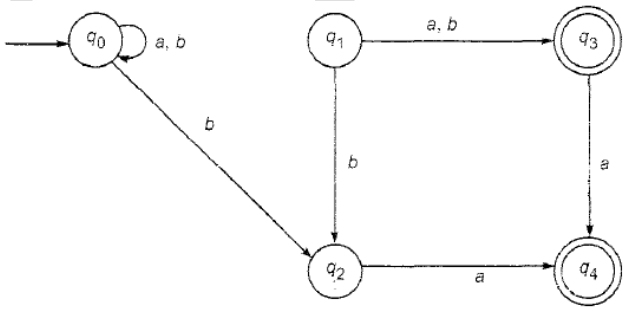
State	a	b
q_0	q_1, q_3	q_2, q_3
q_1	q_1	q_3
q_2	q_3	q_2
((q_3))	—	—

10.Find a deterministic acceptor equivalent to M=($\{q_0, q_1, q_2\}$,{a,b},δ,q_0,$\{q_2\}$) where δ is given by the table

State/Σ	a	b
→q_0	q_0, q_1	q_2
q_1	q_0	q_1
((q_2))		q_0, q_1

Part- C (Compulsory)

1.Construct a DFA equivalent to the NDFA M whose transition diagram is given below



KARPAGAM ACADEMY OF HIGHER EDUCATION
DEPARTMENT OF MATHEMATICS
FORMAL LANGUAGES AND AUTOMATA THEORY (17MMP305A)

Questions	choice 1	choice 2	choice 3	choice 4	Answer
UNIT - I					
Assume the R is a relation on a set A, aRb is partially ordered such that a and b are _____	reflexive	transitive	symmetric	and transitive	and transitive
A regular language over an alphabet ∑ is one that cannot be obtained from the basic languages using the operation _____	Union	Concatenatic	Kleene	All the above	All the above
Which of the following is a not a part of 5-tuple finite automata?	Input alphabe	Transition fu	Initial State	Output	Output
If an Infinite language is passed to Machine M, the subsidiary which gives a finite solution to the infinite input tape is _____	Compiler	Interpreter	Loader	Linkers	Compiler
For the following change of state in FA, which of the following codes is an incorrect option?	δ (m, 1) =n	δ (0, n) =m	δ (m,0) =ε	s: accept = false; cin >> char;if	δ (0, n) =m
Given: ∑= {a, b} L= {xϵ∑*|x is a string combination} ∑4 represents which among the following?	{aa, ab, ba, bb}	ε, abaa, aabb}	{aaa, aab, aba, bbb}	All the above	ε, abaa, aabb}
There are _____ tuples in finite state machine	4	5	6	unlimited	5
Transition function maps.	Σ * Q -> Σ	Q * Q -> Σ	Σ * Σ -> Q	Q * Σ -> Q	Q * Σ -> Q
δ*(q,ya) is equivalent to	δ((q,y),a)	δ(δ*(q,y),a)	δ(q,ya)	from δ notation	δ(δ*(q,y),a)
Languages of a automata is	If it is accepted by automata	If it halts	touch final state in its life time	All language are language of automata	If it is accepted by automata
Language of finite automata is.	Type 0	Type 1	Type 2	Type 3	Type 3
Finite automata requires minimum _____ number of stacks.	1	0	2	3	0
The basic limitation of finite automata is that _____	remember arbitrary large amount of information.	It sometimes recognize grammar that are not regular.	It sometimes fails to recognize regular grammar	All the above	remember arbitrary large amount of information.
NFA, in its name has 'non-deterministic' because of :	The result is undetermined	of path is non-deterministic	be transited next is non-deterministic	All the above	of path is non-deterministic
Which of the following option is correct?	NFA is slower to process and its representation uses more memory	DFA is faster to process and its representation uses less memory	NFA is slower to process and its representation uses less memory	DFA is slower to process and its representation uses less memory	NFA is slower to process and its representation uses less memory
What is the relation between DFA and NFA on the basis of computational power?	than DFA	than NFA	than DFA	than NFA	than DFA
The production of form non-terminal -> ε is called:	DFA > NFA	NFA > DFA	Equal	nonequal	Equal
Which of the following is a regular language?	Sigma Production	Null Producti	Epsilon Prod	All the above	Null Production
If L1 and L2 are regular languages, which among the following is an exception?	length is a sequence of prime numbers	String with substring ww^r in between	Palindrome s	String with even number of Zero's	String with even number of Zero's
Language for which no DFA exist is a_____	L1 U L2	L1 – L2	L1 ∩ L2	All the above	All the above
	Regular Language	Non-Regular Language	May be Regular	language	Non-Regular Language

UNIT-II

Formal Languages - Basic Definitions and examples - Chomsky classification of Languages - Languages and their relation - Recursive and Recursively Enumerable sets- Operations on Languages.

BASIC DEFINITIONS AND EXAMPLES

Definition A phrase-structure grammar (or simply a grammar) is (V_N, Σ, P, S), where

(i) V_N is a finite nonempty set whose elements are called variables,
(ii) Σ is a finite nonempty set whose elements are called terminals,
(iii) $V_N \cap \Sigma = \emptyset$,
(iv) S is a special variable (i.e. an element of V_N) called the start symbol. and
(v) P is a finite set whose elements are $\alpha \to \beta$, where α and β are strings on $V_N \cup \Sigma$. α has at least one symbol from V_N. The elements of P are called productions or production rules or rewriting rules.

$G = (V_N, \Sigma, P, S)$ is a grammar

where

$V_N = \{\langle\text{sentence}\rangle, \langle\text{noun}\rangle, \langle\text{verb}\rangle, \langle\text{adverb}\rangle\}$
$\Sigma = \{\text{Ram, Sam, ate, sang, well}\}$
$S = \langle\text{sentence}\rangle$

P consists of the following productions:

$\langle\text{sentence}\rangle \to \langle\text{noun}\rangle\ \langle\text{verb}\rangle$
$\langle\text{sentence}\rangle \to \langle\text{noun}\rangle\ \langle\text{verb}\rangle\ \langle\text{adverb}\rangle$
$\langle\text{noun}\rangle \to \text{Ram}$
$\langle\text{noun}\rangle \to \text{Sam}$
$\langle\text{verb}\rangle \to \text{ate}$
$\langle\text{verb}\rangle \to \text{sang}$
$\langle\text{adverb}\rangle \to \text{well}$

Definition If $\alpha \to \beta$ is a production in a grammar G and γ, δ are any two strings on $V_N \cup \Sigma$, then we say that $\gamma\alpha\delta$ directly derives $\gamma\beta\delta$ in G (we write this as $\gamma\alpha\delta \underset{G}{\Rightarrow} \gamma\beta\delta$). This process is called one-step derivation. In particular, if $\alpha \to \beta$ is a production, then $\alpha \underset{G}{\Rightarrow} \beta$.

Definition If α and β are strings on $V_N \cup \Sigma$, then we say that α derives β if $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$. Here $\overset{*}{\underset{G}{\Rightarrow}}$ denotes the reflexive-transitive closure of the relation $\underset{G}{\Rightarrow}$ in $(V_N \cup \Sigma)^*$

Definition The language generated by a grammar G (denoted by $L(G)$) is defined as $\{w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$. The elements of $L(G)$ are called *sentences*.

Definition G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$.

If $G = (\{S\}, \{0, 1\}, \{S \to 0S1, S \to \Lambda\}, S)$, find $L(G)$.

Solution

As $S \to \Lambda$ is a production, $S \underset{G}{\Rightarrow} \Lambda$. So Λ is in $L(G)$. Also, for all $n \geq 1$,

$$S \underset{G}{\Rightarrow} 0S1 \underset{G}{\Rightarrow} 0^2S1^2 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} 0^nS1^n \underset{G}{\Rightarrow} 0^n1^n$$

Therefore,

$$0^n1^n \in L(G) \text{ for } n \geq 0$$

(Note that in the above derivation, $S \to 0S1$ is applied at every step except in the last step. In the last step, we apply $S \to \Lambda$). Hence, $\{0^n1^n \mid n \geq 0\} \subseteq L(G)$.

To show that $L(G) \subseteq \{0^n1^n \mid n \geq 0\}$, we start with w in $L(G)$. The derivation of w starts with S. If $S \to \Lambda$ is applied first, we get Λ. In this case $w = \Lambda$. Otherwise the first production to be applied is $S \to 0S1$. At any stage if we apply $S \to \Lambda$, we get a terminal string. Also, the terminal string is obtained only by applying $S \to \Lambda$. Thus the derivation of w is of the form

$$S \overset{*}{\underset{G}{\Rightarrow}} 0^nS1^n \underset{G}{\Rightarrow} 0^n1^n \quad \text{for some } n \geq 1$$

$L(G) \subseteq \{0^n 1^n \mid n \geq 0\}$

Therefore,

$$L(G) = \{0^n 1^n \mid n \geq 0\}$$

If G is $S \to aS \mid bS \mid a \mid b$, find $L(G)$.

Solution

We show that $L(G) = \{a, b\}^+$. As we have only two terminals a, b, $L(G) \subseteq \{a, b\}^*$. All productions are S-productions, and so Λ can be in $L(G)$ only when $S \to \Lambda$ is a production in the grammar G. Thus,

$$L(G) \subseteq \{a, b\}^* - \{\Lambda\} = \{a, b\}^+$$

To show $\{a, b\}^+ \subseteq L(G)$, consider any string $a_1 a_2 \ldots a_n$, where each a_i is either a or b. The first production in the derivation of $a_1 a_2 \ldots a_n$ is $S \to aS$ or $S \to bS$ according as $a_1 = a$ or $a_1 = b$. The subsequent productions are obtained in a similar way. The last production is $S \to a$ or $S \to b$ according as $a_n = a$ or $a_n = b$. So $a_1 a_2 \ldots a_n \in L(G)$. Thus, we have $L(G) = \{a, b\}^+$.

Construct a grammar generating $L = \{wcw^T \mid w \in \{a, b\}^*\}$.

Solution

Let $G = (\{S\}, \{a, b, c\}, P, S)$, where P is defined as $S \to aSa \mid bSb \mid c$. It is easy to see the idea behind the construction. Any string in L is generated by recursion as follows: (i) $c \in L$; (ii) if $x \in L$, then $wxw^T \in L$. So, as in the earlier example, we have the productions $S \to aSa \mid bSb \mid c$.

Find a grammar generating $L = \{a^n b^n c^i \mid n \geq 1, i \geq 0\}$.

Solution

$$L = L_1 \cup L_2$$

$$L_1 = \{a^n b^n \mid n \geq 1\}$$

$$L_2 = \{a^n b^n c^i \mid n \geq 1, i \geq 1\}$$

We construct L_1 by recursion and L_2 by concatenating the elements of L_1 and c^i, $i \geq 1$. We define P as the set of the following productions:

$$S \to A, \qquad A \to ab, \qquad A \to aAb, \qquad S \to Sc$$

Let $G = (\{S, A\}, \{a, b, c\}, P, S)$. For $n \geq 1$, $i \to 0$, we have

$$S \overset{*}{\Rightarrow} Sc^i \Rightarrow Ac^i \overset{*}{\Rightarrow} a^{n-1}Ab^{n-1}c^i \Rightarrow a^{n-1}abb^{n-1}c^i = a^nb^nc^i$$

Thus,

$$\{a^nb^nc^i \mid n \geq 1,\ i \geq 0\} \subseteq L(G)$$

To prove the reverse inclusion, we note that the only S-productions are $S \to Sc$ and $S \to A$. If we start with $S \to A$, we have to apply

$$A \Rightarrow a^{n-1}Ab^{n-1} \overset{*}{\Rightarrow} a^nb^n, \text{ and so } a^nb^nc^0 \in L(G)$$

If we start with $S \to Sc$, we have to apply $S \to Sc$ repeatedly to get Sc^i. But to get a terminal string, we have to apply $S \to A$. As $A \overset{*}{\Rightarrow} a^nb^n$, the resulting terminal string is $a^nb^nc^i$. Thus, we have shown that

$$L(G) \subseteq \{a^nb^nc^i \mid n \geq 1,\ i \geq 0\}$$

Therefore,

$$L(G) = \{a^nb^nc^i \mid n \geq 1,\ i \geq 0\}$$

Problem:

Let $G = (\{S, A_1\}, \{0, 1, 2\}, P, S)$, where P consists of $S \to 0SA_12$, $S \to 012$, $2A_1 \to A_12$, $1A_1 \to 11$. Show that

$$L(G) = \{0^n1^n2^n \mid n \geq 1\}$$

Solution

As $S \to 012$ is a production, we have $S \Rightarrow 012$, i.e. $012 \in L(G)$.
Also,

$$\begin{aligned}
S &\overset{*}{\Rightarrow} 0^{n-1}S(A_12)^{n-1} && \text{by applying } S \to 0SA_12 \quad (n-1) \text{ times}\\
&\Rightarrow 0^n12(A_12)^{n-1} && \text{by applying } S \to 012\\
&\overset{*}{\Rightarrow} 0^n1A_1^{n-1}2^n && \text{by applying } 2A_1 \to A_12 \quad \text{several times}\\
&\overset{*}{\Rightarrow} 0^n1^n2^n && \text{by applying } 1A_1 \to 11 \quad (n-1) \text{ times}
\end{aligned}$$

Therefore.

$$0^n1^n2^n \in L(G) \qquad \text{for all } n \geq 1$$

Problem:

Let $G = (\{S, A_1, A_2\}, \{a, b\}, P, S)$, where P consists of

$$S \to aA_1A_2a,\ A_1 \to baA_1A_2b,\ A_2 \to A_1ab,\ aA_1 \to baa,\ bA_2b \to abab$$

Test whether $w = baabbabaaabbaba$

is in $L(G)$.

Solution

We have to start with an S-production. At every stage we apply a suitable production which is likely to derive w. In this example, we underline the substring to be replaced by the use of a production.

$$\begin{aligned} S &\Rightarrow \underline{aA_1}\,A_2a \\ &\Rightarrow baa\,\underline{A_2}\,a \\ &\Rightarrow baa\,\underline{A_1}\,aba \\ &\Rightarrow baab\,\underline{aA_1}\,A_2baba \\ &\Rightarrow baabbaa\,\underline{A_2}\,baba \\ &\Rightarrow baabba\,\underline{aA_1}\,abbaba \\ &\Rightarrow baabbabaaabbaba = w \end{aligned}$$

Therefore.

$$w \in L(G)$$

CHOMSKY CLASSIFICATION OF LANGUAGES

Definition A grammar is called type 1 or context-sensitive or context-dependent if all its productions are type 1 productions. The production $S \to \Lambda$ is also allowed in a type 1 grammar, but in this case S does not appear on the right-hand side of any production.

Definition The language generated by a type 1 grammar is called a type 1 or context-sensitive language.

Definition A grammar $G = (V_N, \Sigma, P, S)$ is monotonic (or length-increasing) if every production in P is of the form $\alpha \to \beta$ with $|\alpha| \leq |\beta|$ or $S \to \Lambda$. In the second case, S does not appear on the right-hand side of any production in P.

Definition A type 2 production is a production of the form $A \to \alpha$, where $A \in V_N$ and $\alpha \in (V_N \cup \Sigma)^*$. In other words, the L.H.S. has no left context or right context. For example, $S \to Aa$, $A \to a$, $B \to abc$, $A \to \Lambda$ are type 2 productions.

Definition A grammar is called a type 2 grammar if it contains only type 2 productions. It is also called a context-free grammar (as A can be replaced by α in any context). A language generated by a context-free grammar is called a type 2 language or a context-free language.

Definition A production of the form $A \to a$ or $A \to aB$, where $A, B \in V_N$ and $a \in \Sigma$, is called a type 3 production.

Definition A grammar is called a type 3 or regular grammar if all its productions are type 3 productions. A production $S \to \Lambda$ is allowed in type 3 grammar, but in this case S does not appear on the right-hand side of any production.

Example:

Find the highest type number which can be applied to the following productions:

(a) $S \to Aa, \quad A \to c \mid Ba, \quad B \to abc$
(b) $S \to ASB \mid d, \quad A \to aA$
(c) $S \to aS \mid ab$

Solution

(a) $S \to Aa$. $A \to Ba$, $B \to abc$ are type 2 and $A \to c$ is type 3. So the highest type number is 2.
(b) $S \to ASB$ is type 2, $S \to d$, $A \to aA$ are type 3. Therefore, the highest type number is 2.
(c) $S \to aS$ is type 3 and $S \to ab$ is type 2. Hence the highest type number is 2.

LANGUAGES AND THEIR RELATION

Property 1 From the definition, it follows that $\mathcal{L}_{rl} \subseteq \mathcal{L}_{cfl}$, $\mathcal{L}_{csl} \subseteq \mathcal{L}_0$, $\mathcal{L}_{cfl} \subseteq \mathcal{L}_0$.

Property 2 $\mathcal{L}_{cfl} \subseteq \mathcal{L}_{csl}$.

Property 3 $\mathcal{L}_{rl} \subseteq \mathcal{L}_{cfl} \subseteq \mathcal{L}_{csl} \subseteq \mathcal{L}_0$. This follows from properties 1 and 2.

Property 4 $\mathcal{L}_{rl} \subset_{\neq} \mathcal{L}_{cfl} \subset_{\neq} \mathcal{L}_{csl} \subset_{\neq} \mathcal{L}_0$.

RECURSIVE AND RECURSIVELY ENUMERABLE SETS

Definition A set X is recursive if we have an algorithm to determine whether a given element belongs to X or not.

Definition A recursively enumerable set is a set X for which we have a procedure to determine whether a given element belongs to X or not.

Example:

Consider the grammar G given by $S \to 0SA_1 2$, $S \to 012$, $2A_1 \to A_1 2$, $1A_1 \to 11$. Test whether (a) $00112 \in L(G)$ and (b) $001122 \in L(G)$.

Solution

(a) To test whether $w = 00112 \in L(G)$, we construct the sets W_0, W_1, W_2 etc. $|w| = 5$.

$$W_0 = \{S\}$$
$$W_1 = \{012,\ S,\ 0SA_1 2\}$$
$$W_2 = \{012,\ S,\ 0SA_1 2\}$$

As $W_2 = W_1$, we terminate. (Although $0SA_1 2 \Rightarrow 0012A_1 2$, we cannot include $0012A_1 2$ in W_1 as its length is > 5.) Then $00112 \notin W_1$. Hence, $00112 \notin L(G)$.

(b) To test whether $w = 001122 \in L(G)$. Here, $|w| = 6$. We construct W_0, W_1, W_2, etc.

$$W_0 = \{S\}$$
$$W_1 = \{012,\ S,\ 0SA_1 2\}$$
$$W_2 = \{012,\ S,\ 0SA_1 2,\ 0012A_1 2\}$$
$$W_3 = \{012,\ S,\ 0SA_1 2,\ 0012A_1 2,\ 001A_1 22\}$$
$$W_4 = \{012,\ S,\ 0SA_1 2,\ 0012A_1 2,\ 001A_1 22,\ 001122\}$$
$$W_5 = \{012,\ S,\ 0SA_1 2,\ 0012A_1 2,\ 001A_1 22,\ 001122\}$$

; $W_5 = W_4$, we terminate. Then $001122 \in W_4$. Thus, $001122 \in L(G)$.

OPERATIONS ON LANGUAGES

Theorem Each of the classes $\mathcal{L}_0$, $\mathcal{L}_{cs1}$, $\mathcal{L}_{cf1}$, $\mathcal{L}_{r1}$ is closed under union.

Proof Let L_1 and L_2 be two languages of the same type i. We can apply Theorem 4.1 to get grammars

$$G_1 = (V'_N, \Sigma_1, P_1, S_1) \quad \text{and} \quad G_2 = (V''_N, \Sigma_2, P_2, S_2)$$

of type i generating L_1 and L_2, respectively. So any production in G_1 or G_2 is either $\alpha \to \beta$, where α, β contain only variables or $A \to a$, where $A \in V_N$, $a \in \Sigma$.

We can further assume that $V'_N \cap V''_N = \emptyset$. (This is achieved by renaming the variables of V''_N if they occur in V'_N.)

Define a new grammar G_u as follows:

$$G_u = (V'_N \cup V''_N \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_u, S)$$

where S is a new symbol, i.e. $S \notin V'_N \cup V''_N$

$$P_u = P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}$$

We prove $L(G_u) = L_1 \cup L_2$ as follows: If $w \in L_1 \cup L_2$, then $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w$ or $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w$. Therefore,

$$S \underset{G_u}{\Rightarrow} S_1 \overset{*}{\underset{G_u}{\Rightarrow}} w \quad \text{or} \quad S \underset{G_u}{\Rightarrow} S_2 \overset{*}{\underset{G_u}{\Rightarrow}} w, \text{ i.e. } w \in L(G_u)$$

Thus, $L_1 \cup L_2 \subseteq L(G_u)$.

To prove that $L(G_u) \subseteq L_1 \cup L_2$, consider a derivation of w. The first step should be $S \Rightarrow S_1$ or $S \Rightarrow S_2$. If $S \Rightarrow S_1$ is the first step, in the subsequent steps S_1 is changed. As $V'_N \cap V''_N \neq \emptyset$, these steps should involve only the variables of V'_N and the productions we apply are in P_1. So $S \overset{*}{\underset{G_1}{\Rightarrow}} w$. Similarly, if the first step is $S \Rightarrow S_2$, then $S \underset{G_2}{\Rightarrow} S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w$. Thus, $L(G_u) = L_1 \cup L_2$. Also, $L(G_u)$ is of type 0 or type 2 according as L_1 and L_2 are of type 0 or type 2. If Λ is not in $L_1 \cup L_2$, then $L(G_u)$ is of type 3 or type 1 according as L_1 and L_2 are of type 3 or type 1.

Suppose $\Lambda \in L_1$. In this case, define

$$G_u = (V'_N \cup V''_N \cup \{S, S'\}, \Sigma_1 \cup \Sigma_2, P_u, S')$$

where (i) S' is a new symbol, i.e. $S' \notin V'_N \cup V''_N \cup \{S\}$, and (ii) $P_u = P_1 \cup P_2 \cup \{S' \to S, S \to S_1, S \to S_2\}$. So, $L(G_u)$ is of type 1 or type 3 according as L_1 and L_2 are of type 1 or type 3. When $\Lambda \in L_2$, the proof is similar. ∎

Possible Questions

Part-B(6 mark)

1. If G = ({S}, {0,1},{S → 0S1, S → Λ}. S), find L(G).
2. Let G = ({S, C} , {a, b} , P, S) where P consists of S →aCa, C →aCa |b. Find L(G).
3. If G is S →aS | bS | a | b, find L(G).
4. Let G = ({S, A_1} , {0, 1,2} , P, S) where P consists of S → 0S$A_1$2, S → 012, 2A_1 →$A_1$2. 1A_1 → 11. Show that L(G) = {0"1"2" | n ≥ 1}.
5. Let G = ({S, A_1,A_2} , {a, b} , P, S) where P consists of S → a$A_1$$A_2$a, A_1 →ba$A_1$$A_2$b, A_2 → A_1 ab, aA_1→baa, bA_2b →abab. Test whether w = baabbabaaabbaba is in G.
6. Consider the grammar G given by S → 0S$A_1$2. S → 012, 2A_1→ $A_1$2, 1A_1→ 11. Test whether (a) 00112 ϵ L(G) and (b) 001122 ϵ L(G).
7. Show that there exists a recursive set which is not a context- sensitive language over {0,1}.
8. Show that {a^ n^2 | n ≥ 1} is generated by the grammar S → a, S → $A_3$$A_4$, A_3→ $A_1$$A_3$$A_2$,
   A_3→ $A_1$$A_2$, $A_1$$A_2$→a$A_2$A1, A_1a→aA_1 , A_2a→aA_2, $A_1$$A_4$→$A_4$a, $A_2$$A_5$ →A_5a, A_5 → a.
9. Explain about the Chomsky classification of languages and their properties.
10. Show that each of the classes L_0, L_{cs1}, L_{cf1}, L_{r1} is closed under concatenation.

PART C(10 mark)

1. Show that each of the classes L_0, L_{cs1}, L_{cf1}, L_{r1} is closed under union.

KARPAGAM ACADEMY OF HIGHER EDUCATION
DEPARTMENT OF MATHEMATICS
FORMAL LANGUAGES AND AUTOMATA THEORY (17MMP305A)

Questions	choice 1	choice 2	choice 3	choice 4	Answer
		UNIT - II			
Concatenation Operation refers to which of the following set operations:	Union	Dot	Kleene	Two of the options are correct	Dot
Concatenation of R with Φ outputs:	R	Φ	R.Φ	r	Φ
RR* can be expressed in which of the forms:	R+	R-	R+ U R-	R	R+
If L1 and L2 are context free languages, which of the following is context free?	L1*	L2UL1	L1.L2	All the above	All the above
A grammar G=(V, T, P, S) is _____ if every production taken one of the two forms: B->aC B->a	Ambiguous	Regular	Non Regular	set	Regular
Which of the following languages are most suitable for implement context free languages ?	C	Perl	Assembly Languag	language	C
Regular sets are closed under union,concatenation and kleene closure.	True	FALSE	Depends on regular set	set	True
Complement of a DFA can be obtained by	making starting state as final state.	no trival method.	making final states non-final and non-final to final.	make final as a starting state.	making final states non-final and non-final to final.
Complement of regular sets are _____	Regular	CFG	CSG	RE	Regular
If L1 and L2 are regular sets then intersection of these two will be	Regular	Non Regular	Recursive	Non Recursive	Regular
If L1 is regular L2 is unknown but L1-L2 is regular ,then L2 must be	Empty set	CFG	Decidable	Regular	Regular
Reverse of a DFA can be formed by	using PDA	making final state as non-final	making final as starting state and starting state as final state	final	making final as starting state and starting state as final state
Reverse of (0+1)* will be _____	Phi	Null	(0+1)*	(0+1)	(0+1)*
(a ^ 5b ^ 5)* is example of _____	Type 0 language	Type 1 language	Type 2 language	Type 3 language	Type 3 language
Which of the following is type 3 language ?	Strings of 0's whose length is perfect square	Palindromes string	Strings of 0's having length prime number	String of odd number of 0's	String of odd number of 0's
a ^ nb ^ n where (n+m) is even .	Type 0	Type 1	Type 2	Type 3	Type 3
Complement of a ^ nb ^ m where n >= 4 and m	Type 0	Type 1	Type 2	Type 3	Type 3
Complement of (a + b)* will be	phi	null	a	b	phi
Which of the following strings do not belong the given regular expression? (a)*(a+cba)	aa	aaa	acba	acbacba	acbacba

UNIT-III

Regular expressions - Finite Automata and Regular expressions.

REGULAR EXPRESSIONS

The regular expressions are useful for representing certain sets of strings in an algebraic fashion. Actually these describe the languages accepted by finite state automata.

We give a formal recursive definition of regular expressions over Σ as follows:

1. Any terminal symbol (i.e. an element of Σ), Λ and $\emptyset$ are regular expressions. When we view a in Σ as a regular expression, we denote it by **a**.
2. The union of two regular expressions $\mathbf{R}_1$ and $\mathbf{R}_2$, written as $\mathbf{R}_1 + \mathbf{R}_2$, is also a regular expression.
3. The concatenation of two regular expressions $\mathbf{R}_1$ and $\mathbf{R}_2$, written as $\mathbf{R}_1\mathbf{R}_2$, is also a regular expression.
4. The iteration (or closure) of a regular expression **R**, written as $\mathbf{R}^*$, is also a regular expression.
5. If **R** is a regular expression, then (**R**) is also a regular expression.
6. The regular expressions over Σ are precisely those obtained recursively by the application of the rules 1–5 once or several times.

Definition Any set represented by a regular expression is called a *regular set.*

If, for example, $a, b \in \Sigma$, then (i) **a** denotes the set $\{a\}$, (ii) **a** + **b** denotes $\{a, b\}$, (iii) **ab** denotes $\{ab\}$, (iv) $\mathbf{a}^*$ denotes the set $\{\Lambda, a, aa, aaa, \ldots\}$ and (v) $(\mathbf{a} + \mathbf{b})^*$ denotes $\{a, b\}^*$.

The set represented by R is denoted by $L(\mathbf{R})$.

Example:

Describe the following sets by regular expressions: (a) {101}, (b) {*abba*}, (c) {01, 10}, (d) {Λ, *ab*}, (e) {*abb, a, b, bba*}, (f) {Λ, 0, 00, 000, . . .}, and (g) {1, 11, 111, . . .}.

Solution

(a) Now, {1}, {0} are represented by **1** and **0**, respectively. 101 is obtained by concatenating 1, 0 and 1. So, {101} is represented by **101**.

(b) **abba** represents {*abba*}.
(c) As {01, 10} is the union of {01} and {10}, we have {01, 10} represented by **01 + 10**.
(d) The set {Λ, *ab*} is represented by Λ + **ab**.
(e) The set {*abb, a, b, bba*} is represented by **abb + a + b + bba**.
(f) As {Λ, 0, 00, 000, . . .} is simply {0}*, it is represented by **0***.
(g) Any element in {1, 11, 111, . . .} can be obtained by concatenating 1 and any element of {1}*. Hence **1**(**1**)* represents {1, 11, 111, . . .}.

Example:

Describe the following sets by regular expressions:
(a) L_1 = the set of all strings of 0's and 1's ending in 00.
(b) L_2 = the set of all strings of 0's and 1's beginning with 0 and ending with 1.
(c) L_3 = {Λ, 11, 1111, 111111, . . .}.

Solution

(a) Any string in L_1 is obtained by concatenating any string over {0, 1} and the string 00. {0, 1} is represented by **0 + 1**. Hence L_1 is represented by (**0 + 1**)* **00**.
(b) As any element of L_2 is obtained by concatenating 0, any string over {0, 1} and 1, L_2 can be represented by **0**(**0 + 1**)* **1**.
(c) Any element of L_3 is either Λ or a string of even number of 1's, i.e. a string of the form $(11)^n$, $n \geq 0$. So L_3 can be represented by (**11**)*.

IDENTITIES FOR REGULAR EXPRESSIONS

Two regular expressions **P** and **Q** are equivalent (we write **P** = **Q**) if **P** and **Q** represent the same set of strings.

We now give the identities for regular expressions; these are useful for simplifying regular expressions.

I_1 $\emptyset + \mathbf{R} = \mathbf{R}$

I_2 $\emptyset\mathbf{R} = \mathbf{R}\emptyset = \emptyset$

I_3 $\Lambda\mathbf{R} = \mathbf{R}\Lambda = \mathbf{R}$

I_4 $\Lambda^* = \Lambda$ and $\emptyset^* = \Lambda$

I_5 $\mathbf{R} + \mathbf{R} = \mathbf{R}$

I_6 $\mathbf{R}^*\mathbf{R}^* = \mathbf{R}^*$

I_7 $\mathbf{R}\mathbf{R}^* = \mathbf{R}^*\mathbf{R}$

I_8 $(\mathbf{R}^*)^* = \mathbf{R}^*$

I_9 $\Lambda + \mathbf{R}\mathbf{R}^* = \mathbf{R}^* = \Lambda + \mathbf{R}^*\mathbf{R}$

I_{10} $(\mathbf{P}\mathbf{Q})^*\mathbf{P} = \mathbf{P}(\mathbf{Q}\mathbf{P})^*$

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

I_{12} $(\mathbf{P} + \mathbf{Q})\mathbf{R} = \mathbf{P}\mathbf{R} + \mathbf{Q}\mathbf{R}$ and $\mathbf{R}(\mathbf{P} + \mathbf{Q}) = \mathbf{R}\mathbf{P} + \mathbf{R}\mathbf{Q}$

Theorem (Arden's theorem) Let **P** and **Q** be two regular expressions over Σ. If **P** does not contain Λ, then the following equation in **R**, namely

$$\mathbf{R} = \mathbf{Q} + \mathbf{R}\mathbf{P}$$

has a unique solution (i.e. one and only one solution) given by $\mathbf{R} = \mathbf{Q}\mathbf{P}^*$.

Proof $\mathbf{Q} + (\mathbf{Q}\mathbf{P}^*)\mathbf{P} = \mathbf{Q}(\Lambda + \mathbf{P}^*\mathbf{P}) = \mathbf{Q}\mathbf{P}^*$ by I_9

To prove uniqueness. replacing **R** by **Q + RP** on the R.H.S., we get the equation

$$\begin{aligned}\mathbf{Q} + \mathbf{RP} &= \mathbf{Q} + (\mathbf{Q} + \mathbf{RP})\mathbf{P} \\ &= \mathbf{Q} + \mathbf{QP} + \mathbf{RPP} \\ &= \mathbf{Q} + \mathbf{QP} + \mathbf{RP}^2 \\ &= \mathbf{Q} + \mathbf{QP} + \mathbf{QP}^2 + \cdots + \mathbf{QP}^i + \mathbf{RP}^{i+1} \\ &= \mathbf{Q}(\Lambda + \mathbf{P} + \mathbf{P}^2 + \cdots + \mathbf{P}^i) + \mathbf{RP}^{i+1}\end{aligned}$$

$$\mathbf{R} = \mathbf{Q}(\Lambda + \mathbf{P} + \mathbf{P}^2 + \cdots + \mathbf{P}^i) + \mathbf{RP}^{i+1} \quad \text{for } i \geq 0$$

Let w be a string of length i in the set **R**. Then w belongs to the set $Q(\Lambda + \mathbf{P} + \mathbf{P}^2 + \ldots + \mathbf{P}^i) + \mathbf{RP}^{i+1}$. As **P** does not contain Λ, $\mathbf{RP}^{i+1}$ has no string of length less than $i + 1$ and so w is not in the set $\mathbf{RP}^{i+1}$. This means that w belongs to the set $Q(\Lambda + \mathbf{P} + \mathbf{P}^2 + \ldots + \mathbf{P}^i)$, and hence to $\mathbf{QP}^*$.

Consider a string w in the set $\mathbf{QP}^*$. Then w is in the set $\mathbf{QP}^k$ for some $k \geq 0$, and hence in $O(\Lambda + \mathbf{P} + \mathbf{P}^2 + \cdots + \mathbf{P}^k)$.

Thus **R** and $\mathbf{QP}^*$ represent the same set. This proves the uniqueness of the solution .

Example:

(a) Give an r.e. for representing the set L of strings in which every 0 is immediately followed by at least two 1's.

(b) Prove that the regular expression $\mathbf{R} = \Lambda + \mathbf{1}^*(\mathbf{011})^*(\mathbf{1}^* (\mathbf{011})^*)^*$ also describes the same set of strings.

Solution

(a) If w is in L, then either (a) w does not contain any 0, or (b) it contains a 0 preceded by 1 and followed by 11. So w can be written as $w_1 w_2 \ldots w_n$, where each w_i is either 1 or 011. So L is represented by the r.e. $\mathbf{(1 + 011)^*}$.

(b) $\mathbf{R} = \Lambda + \mathbf{P}_1\mathbf{P}_1^*$, where $\mathbf{P}_1 = \mathbf{1^*(011)^*}$

$= \mathbf{P}_1^*$ using I_9

$= \mathbf{(1^*(011)^*)^*}$

$= (\mathbf{P}_2^*\mathbf{P}_3^*)^*$ letting $\mathbf{P}_2 = \mathbf{1}$, $\mathbf{P}_3 = \mathbf{011}$

$= (\mathbf{P}_2 + \mathbf{P}_3)^*$ using I_{11}

$= \mathbf{(1 + 011)^*}$

Prove $\mathbf{(1 + 00^*1) + (1 + 00^*1)(0 + 10^*1)^* (0 + 10^*1) = 0^*1(0 + 10^*1)^*}$.

Solution

L.H.S. $= \mathbf{(1 + 00^*1)}\,(\Lambda + \mathbf{(0 + 10^*1)^*\ (0 + 10^*1)}\Lambda$ using I_{12}

$= \mathbf{(1 + 00^*1)\ (0 + 10^*1)^*}$ using I_9

$= (\Lambda + \mathbf{00^*})\mathbf{1\ (0 + 10^*1)^*}$ using I_{12} for $\mathbf{1 + 00^*1}$

$= \mathbf{0^*1(0 + 10^*1)^*}$ using I_9

$=$ R.H.S.

FINITE AUTOMATA AND REGULAR EXPRESSIONS

TRANSITION SYSTEM CONTAINING Λ-MOVES

Suppose we want to replace a Λ-move from vertex v_1 to vertex v_2. Then we proceed as follows:

Step 1 Find all the edges starting from v_2.

Step 2 Duplicate all these edges starting from v_1, without changing the edge

Step 3 If v_1 is an initial state, make v_2 also as initial state.

Step 4 If v_2 is a final state, make v_1 also as the final state.

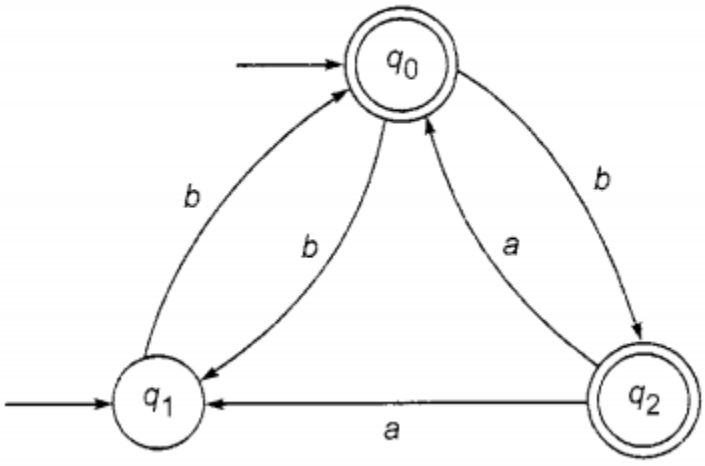
CONVERSION OF NONDETERMINISTIC SYSTEMS TO DETERMINISTIC SYSTEMS

Step 1 Convert the given transition system into state transition table where each state corresponds to a row and each input symbol corresponds to a column.

Step 2 Construct the successor table which lists the subsets of states reachable from the set of initial states. Denote this collection of subsets by Q'.

Step 3 The transition graph given by the successor table is the required deterministic system. The final states contain some final state of NDFA. If possible, reduce the number of states.

Obtain the deterministic graph (system) equivalent to the transition system



Solution

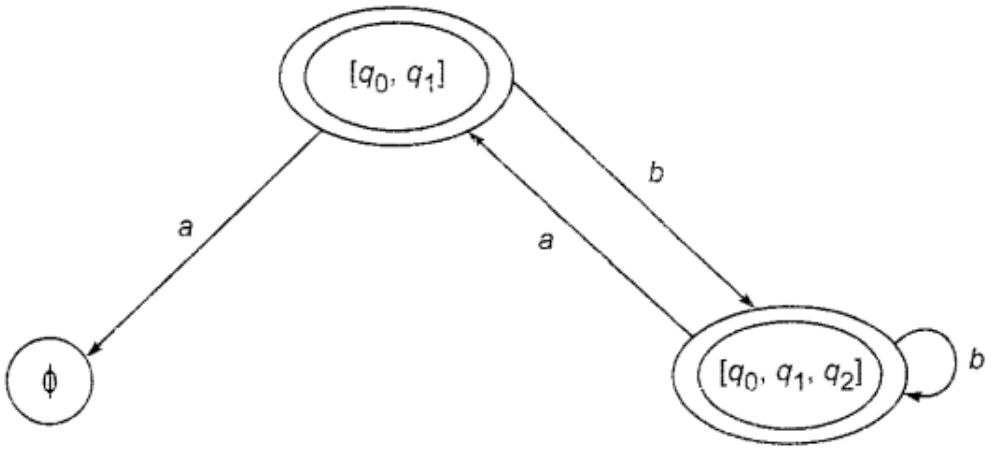
We construct the transition table corresponding to the given nondeterministic system

State/Σ	a	b
$\rightarrow(q_0)$		q_1, q_2
q_1		q_0
(q_2)	q_0, q_1	

We construct the successor table by starting with $[q_0, q_1]$.
we see that $[q_0, q_1, q_2]$ is reachable from $[q_0, q_1]$ by a b-path. There are no a-paths from $[q_0, q_1]$. Similarly, $[q_0, q_1]$ is reachable from $[q_0, q_1, q_2]$ by an a-path and $[q_0, q_1, q_2]$ is reachable from itself. We proceed with the construction for all the elements in Q'.

Deterministic Transition Table

Q	a	b
$[q_0, q_1]$	$\emptyset$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2]$	$[q_0, q_1]$	$[q_0, q_1, q_2]$
$\emptyset$	$\emptyset$	$\emptyset$



Deterministic transition system

as q_0 and q_2 are the final states of the nondeterministic system $[q_0, q_1]$ and $[q_0, q_1, q_2]$ are the final states of the deterministic system.

ALGEBRAIC METHOD USING ARDEN'S THEOREM

The following assumptions are made regarding the transition system:

(i) The transition graph does not have Λ-moves.
(ii) It has only one initial state, say v_1.
(iii) Its vertices are $v_1 \ldots v_n$.
(iv) $\mathbf{V}_i$ the r.e. represents the set of strings accepted by the system even though v_i is a final state.
(v) α_{ij} denotes the r.e. representing the set of labels of edges from v_i to v_j. When there is no such edge, $\alpha_{ij} = \emptyset$. Consequently, we can get the following set of equations in $\mathbf{V}_1 \ldots \mathbf{V}_n$:

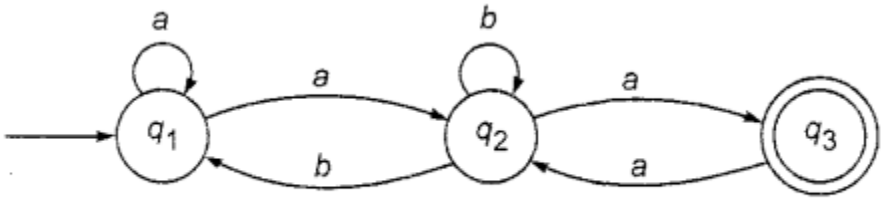
$$\mathbf{V}_1 = \mathbf{V}_1\alpha_{11} + \mathbf{V}_2\alpha_{21} + \cdots + \mathbf{V}_n\alpha_{n1} + \Lambda$$

$$\mathbf{V}_2 = \mathbf{V}_1\alpha_{12} + \mathbf{V}_2\alpha_{22} + \cdots + \mathbf{V}_n\alpha_{n2}$$

$$\vdots$$

$$\mathbf{V}_n = \mathbf{V}_1\alpha_{1n} + \mathbf{V}_2\alpha_{2n} + \cdots + \mathbf{V}_n\alpha_{nn}$$

Consider the transition system Prove that the strings recognized are $(a + a(b + aa)^*b)^* a(b + aa)^* a$.



Transition system

Solution

We can directly apply the above method since the graph does not contain any Λ-move and there is only one initial state.

The three equations for q_1, q_2 and q_3 can be written as

$$q_1 = q_1a + q_2b + \Lambda, \qquad q_2 = q_1a + q_2b + q_3a, \qquad q_3 = q_2a$$

$$\begin{aligned} q_2 &= q_1a + q_2b + q_2aa \\ &= q_1a + q_2(b + aa) \\ &= q_1a(b + aa)^* \end{aligned}$$

Substituting q_2 in q_1, we get

$$\begin{aligned} q_1 &= q_1a + q_1a(b + aa)^*b + \Lambda \\ &= q_1(a + a(b + aa)^*b) + \Lambda \end{aligned}$$

Hence,

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

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

$$q_3 = (a + a(b + aa)^*b)^* a(b + aa)^*a$$

Since q_3 is a final state, the set of strings recognized by the graph is given by

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

CONSTRUCTION OF FINITE AUTOMATA EQUIVALENT TO A REGULAR EXPRESSION

Step 1 Construct a transition graph (transition system) equivalent to the given regular expression using Λ-moves. This is done by using Theorem 5.2.

Step 2 Construct the transition table for the transition graph obtained in step 1. Using the method given in Section 5.2.3, construct the equivalent DFA. We reduce the number of states if possible.

EQUIVALENCE OF TWO REGULAR EXPRESSIONS

Suppose we are interested in testing the equivalence of two regular expressions, say **P** and **Q**. The regular expressions **P** and **Q** are equivalent iff they represent the same set. Also, **P** and **Q** are equivalent iff the corresponding finite automata are equivalent.

Theorem (Kleene's theorem) The class of regular sets over Σ is the smallest class $\mathcal{R}$ containing $\{a\}$ for every $a \in \Sigma$ and closed under union, concatenation and closure.

Proof The set $\{a\}$ is represented by the regular expression **a**. So $\{a\}$ is regular for every $a \in \Sigma$. As the class of regular sets is closed under union, concatenation, and closure, $\mathcal{R}$ is contained in the class of regular sets.

Let L be a regular set. Then $L = T(M)$ for some DFA, $M = (\{q_1, \ldots, q_m\}, \Sigma, \delta, q_0, F)$.

$$L = \bigcup_{j=1}^{n} P_{1f_j}^{m}$$

where $F = \{q_{f_1} \ldots q_{f_n}\}$ and $P_{1f_j}^{m}$ is obtained by applying union, concatenation and closure to singletons in Σ. Thus, L is in $\mathcal{R}$.

Possible Questions:

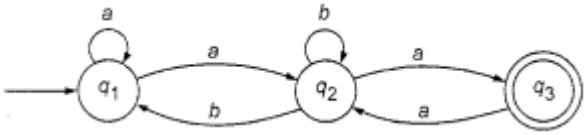
Part-B(6 mark)

1. Describe the following sets by regular expressions :
   i) L_1 = the set of all strings of 0's and 1's ending in 00.
   ii) L_2 = the set of all strings of 0's and 1's beginning with 0 and ending with 1.
   iii) L_3={ Λ, 11, 1111, 111111, …}.

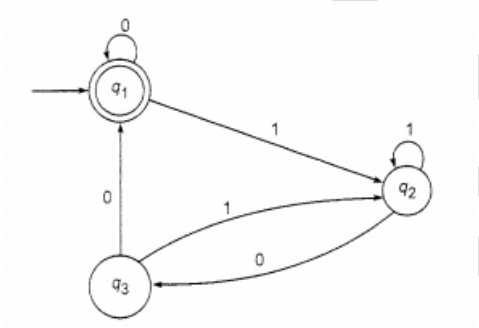
2. State and prove Arden's theorem

3. State and prove Kleen's theorem.

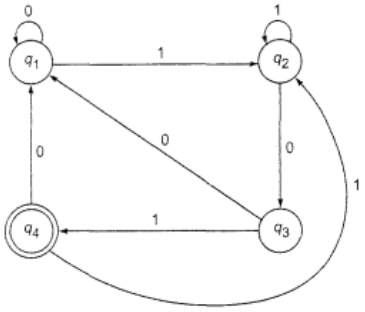
4. Prove that the strings recognized are (a + a(b + aa)*b)* a(b + aa) * a.



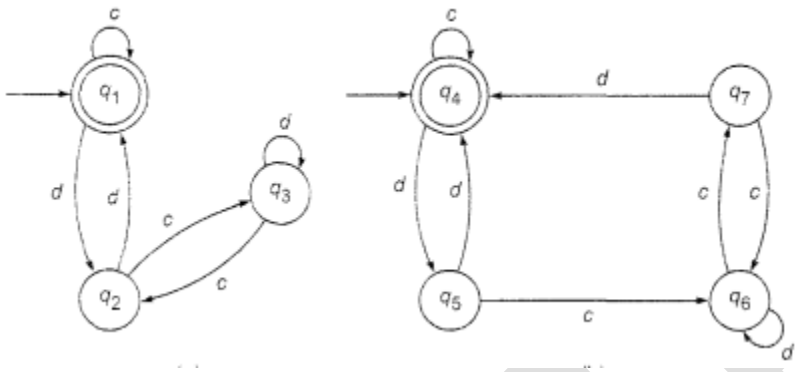
5. Construct a regular expression corresponding to the state diagram described by given below.



6. Find the regular expression corresponding to



7. Determine M and M' are equivalent. Automaton M and M' are given below.

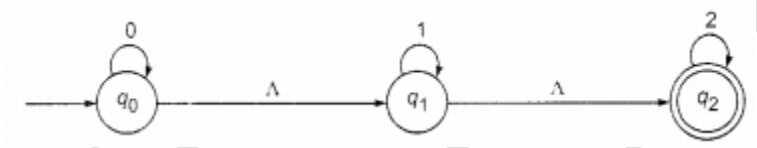


8.(i)Give an r.e for representing the set L of strings in which every 0 is immediately followed by at least two 1's.

(ii) Prove that the regular expression **R** = Λ + **1***(**011**)*(**1*** (**011**)*)* also describe the same set of strings.

9. Prove (**1 + 00*1**) + (**1 + 00*1**)(**0 + 10*1**)* (**0 + 10*1**) = **0*1**(**0 + 10*1**)*.

10. Find the equivalent automation without Λ-moves.



Part-B(10 mark)

1. Construct a DFA with reduced states equivalent to the r.e**10 + (0 + 11))0*1**

KARPAGAM ACADEMY OF HIGHER EDUCATION
DEPARTMENT OF MATHEMATICS
FORMAL LANGUAGES AND AUTOMATA THEORY (17MMP305A)

Questions	choice 1	choice 2	choice 3	choice 4	Answer
UNIT - III					
The following denotion belongs to which type of language: G=(V, T, P, S)	Regular grammar	Context free grammar	Context Sensitive ع	All the above	Context free grammar
The context free languages are closed under:	Intersection	Complement	Kleene	Arden	Kleene
Context free languages are not closed under:	Intersection	Intersection with Regular Language	Complement	All the above	All the above
If the start symbol is one of those symbols which produce no terminal through any sequence, the CFL is said to be	nullable	empty	eliminated	code	empty
Using the pumping constant n, If there is a string in the language of length between _____ and ____ then the language is infite else not.	n, 2n-1	2n, n	n+1, 3n+6	0, n+1	n, 2n-1
A_____ is context free grammar with atmost one non terminal in the right handside of the production.	linear grammar	linear bounded grammar	regular grammar	grammar	linear grammar
There is a linear grammar that generates a context free grammar	always	never	sometimes	struck	sometimes
Regular Expression R and the language it describes can be represented as:	R, R(L)	L(R), R(L)	R, L(R)	All the above	R, L(R)
Let for ∑= {0,1} R= (∑∑∑) *, the language of R would be	{w | w is a string of odd length}	{w | w is a string of length multiple of 3}	{w | w is a string of length 3}	All the above	{w | w is a string of length multiple of 3}
If ∑= {0,1}, then Φ* will result to:	ε	Φ	∑	1	ε
The given NFA represents which of the following NFA	(ab U a) *	(a*b* U a*)	(ab U a*)	(ab)* U a*	(ab U a) *
The finite automata accept the following languages:	Context Free Languages	Context Sensitive Languages	Regular Languages	All the above	Regular Languages
(a + b*c) most correctly represents:	(a +b) *c	(a)+((b)*.c)	(a + (b*)).c	a+ ((b*).c)	a+ ((b*).c)
According to the precedence rules, x-y-z is equivalent to which of the following?	(x-y)-z	x-(y-z)	Both (x-y)-z and x-(y-z)	x+y	(x-y)-z
Dot operator in regular expression resembles which of the following?	Expressions are juxtaposed	Expressions are multiplied	Cross operation	added	Expressions are juxtaposed
Which among the following is equivalent to the given regular expression? 01*+1	(01)*+1	0((1)*+1)	(0(1)*)+1	((0*1)1*)*	(0(1)*)+1
With reference to Automaton to Regular Expression Conversion, for each of the n rounds, where n is the number of states of DFA, we can _____ the size of the regular expression constructed.	double	triple	quadruple	single	quadruple
The conversion of NFA to DFA can be done in:	exponential time	linear time	logarithmic time	all the above	exponential time
NFA to DFA conversion is done via	Subset Construction method	Warshalls Algorithm	Ardens theorem	kleen's theorem	Subset Construction method

UNIT-IV

Pumping Lemma for Regular sets - Applications of Pumping Lemma - Closure Property of Regular sets - Regular sets and Regular grammars.

PUMPING LEMMA FOR REGULAR SETS

Theorem (Pumping Lemma) Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton with n states. Let L be the regular set accepted by M. Let $w \in L$ and $|w| \geq m$. If $m \geq n$, then there exists x, y, z such that $w = xyz$, $y \neq \Lambda$ and $xy^iz \in L$ for each $i \geq 0$.

Proof Let

$$w = a_1a_2 \ldots a_m, \qquad m \geq n$$

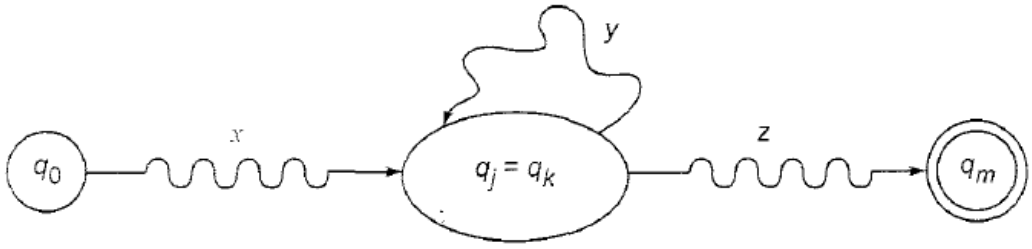
$$\delta(q_0, a_1a_2 \ldots a_i) = q_i \quad \text{for } i = 1, 2, \ldots, m; \qquad Q_1 = \{q_0, q_1, \ldots, q_m\}$$

That is, Q_1 is the sequence of states in the path with path value $w = a_1a_2 \ldots a_m$. As there are only n distinct states, at least two states in Q_1 must coincide. Among the various pairs of repeated states, we take the first pair. Let us take them as q_j and $q_k (q_j = q_k)$. Then j and k satisfy the condition $0 \leq j < k \leq n$.

The string w can be decomposed into three substrings $a_1a_2 \ldots a_j$, $a_{j+1} \ldots a_k$ and $a_{k+1} \ldots a_m$. Let x, y, z denote these strings $a_1a_2 \ldots a_j$, $a_{j+1} \ldots a_k$, $a_{k+1} \ldots a_m$, respectively. As $k \leq n$, $|xy| \leq n$ and $w = xyz$. The path with the path value w in the transition diagram of M

The string w can be decomposed into three substrings $a_1a_2 \ldots a_j$, $a_{j+1} \ldots a_k$ and $a_{k+1} \ldots a_m$. Let x, y, z denote these strings $a_1a_2 \ldots a_j$, $a_{j+1} \ldots a_k$, $a_{k+1} \ldots a_m$, respectively. As $k \leq n$, $|xy| \leq n$ and $w = xyz$. The path with the path value w in the transition diagram of M

The automaton M starts from the initial state q_0. On applying the string x, it reaches $q_j (= q_k)$. On applying the string y, it comes back to $q_j (= q_k)$. So after application of y^i for each $i \geq 0$, the automaton is in the same state q_j. On applying z, it reaches q_m, a final state. Hence, $xy^iz \in L$. As every state in Q_1 is obtained by applying an input symbol, $y \neq \Lambda$. ∎



APPLICATION OF PUMPING LEMMA

Step 1 Assume that L is regular. Let n be the number of states in the corresponding finite automaton.

Step 2 Choose a string w such that $|w| \geq n$. Use pumping lemma to write $w = xyz$, with $|xy| \leq n$ and $|y| > 0$.

Step 3 Find a suitable integer i such that $xy^iz \notin L$. This contradicts our assumption. Hence L is not regular.

Note: The crucial part of the procedure is to find i such that $xy^iz \notin L$. In some cases we prove $xy^iz \notin L$ by considering $|xy^iz|$. In some cases we may have to use the 'structure' of strings in L.

Show that the set $L = \{a^{i^2} \mid i \geq 1\}$ is not regular.

Solution

Step 1 Suppose L is regular. Let n be the number of states in the finite automaton accepting L.

Step 2 Let $w = a^{n^2}$. Then $|w| = n^2 > n$. By pumping lemma, we can write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$.

Step 3 Consider xy^2z. $|xy^2z| = |x| + 2|y| + |z| > |x| + |y| + |z|$ as $|y| > 0$. This means $n^2 = |xyz| = |x| + |y| + |z| < |xy^2z|$. As $|xy| \leq n$, we have $|y| \leq n$. Therefore,

$$|xy^2z| = |x| + 2|y| + |z| \leq n^2 + n$$

i.e.

$$n^2 < |xy^2z| \leq n^2 + n < n^2 + n + n + 1$$

Hence, $|xy^2z|$ strictly lies between n^2 and $(n + 1)^2$, but is not equal to any one of them. Thus $|xy^2z|$ is not a perfect square and so $xy^2z \notin L$. But by pumping lemma, $xy^2z \in L$. This is a contradiction.

Show that $L = \{a^p \mid p \text{ is a prime}\}$ is not regular.

Solution

Step 1 We suppose L is regular. Let n be the number of states in the finite automaton accepting L.

Step 2 Let p be a prime number greater than n. Let $w = a^p$. By pumping lemma, w can be written as $w = xyz$, with $|xy| \leq n$ and $|y| > 0$. x, y, z are simply strings of a's. So, $y = a^m$ for some $m \geq 1$ (and $\leq n$).

Step 3 Let $i = p + 1$. Then $|xy^iz| = |xyz| + |y^{i-1}| = p + (i - 1)m = p + pm$. By pumping lemma, $xy^iz \in L$. But $|xy^iz| = p + pm = p(1 + m)$, and $p(1 + m)$ is not a prime. So $xy^iz \notin L$. This is a contradiction. Thus L is not regular.

Show that $L = \{0^i1^i \mid i \geq 1\}$ is not regular.

Solution

Step 1 Suppose L is regular. Let n be the number of states in the finite automaton accepting L.

Step 2 Let $w = 0^n1^n$. Then $|w| = 2n > n$. By pumping lemma, we write $w = xyz$ with $|xy| \leq n$ and $|y| \neq 0$.

Step 3 We want to find i so that $xy^iz \notin L$ for getting a contradiction. The string y can be in any of the following forms:

Case 1 y has 0's, i.e. $y = 0^k$ for some $k \geq 1$.

Case 2 y has only 1's, i.e. $y = 1^l$ for some $l \geq 1$.

Case 3 y has both 0's and 1's, i.e. $y = 0^k1^j$ for some $k, j \geq 1$.

In Case 1, we can take $i = 0$. As $xyz = 0^n1^n$, $xz = 0^{n-k}1^n$. As $k \geq 1$, $n - k \neq n$. So, $xz \notin L$.

In Case 2, take $i = 0$. As before, xz is 0^n1^{n-l} and $n \neq n - l$. So, $xz \notin L$.

In Case 3, take $i = 2$. As $xyz = 0^{n-k}0^k1^j1^{n-j}$, $xy^2z = 0^{n-k}\,0^k1^j0^k1^j1^{n-j}$. As xy^2z is not of the form 0^i1^i, $xy^2z \notin L$.

Thus in all the cases we get a contradiction. Therefore, L is not regular.

Show that $L = \{ww \mid w \in \{a, b\}^*\}$ is not regular.

Solution

Step 1 Suppose L is regular. Let n be the number of states in the automaton M accepting L.

Step 2 Let us consider $ww = a^n b a^n b$ in L. $|ww| = 2(n + 1) > n$. We can apply pumping lemma to write $ww = xyz$ with $|y| \neq 0$, $|xy| \leq n$.

Step 3 We want to find i so that $xy^i z \notin L$ for getting a contradiction. The string y can be in only one of the following forms:

Case 1 y has no b's, i.e. $y = a^k$ for some $k \geq 1$.

Case 2 y has only one b.

We may note that y cannot have two b's. If so, $|y| \geq n + 2$. But $|y| \leq |xy| \leq n$. In Case 1, we can take $i = 0$. Then $xy^0z = xz$ is of the form $a^m b a^n b$, where $m = n - k < n$ (or $a^n b a^m b$). We cannot write xz in the form uu with $u \in \{a, b\}^*$, and so $xz \notin L$. In Case 2 too, we can take $i = 0$. Then $xy^0z = xz$ has only one b (as one b is removed from xyz, b being in y). So $xz \notin L$ as any element in L should have an even number of a's and an even number of b's.

Thus in both the cases we get a contradiction. Therefore, L is not regular.

CLOSURE PROPERTIES OF REGULAR SETS

If L is regular then L^T is also regular.

We construct a transition system M' by starting with the state diagram of M, and reversing the direction of the directed edges. The set of initial states of M' is defined as the set F, and q_0 is defined as the (only) final state of M', i.e. $M' = (Q, \Sigma, \delta', F, \{q_0\})$.

If $w \in T(M)$, we have a path from q_0 to some final state in F with path value w. By 'reversing the edges', we get a path in M' from some final state in F to q_0. Its path value is w^T. So $w^T \in T(M')$. In a similar way, we can see that if $w_1 \in T(M')$, then $w_1^T \in T(M)$. Thus from the state diagram it is easy to see that $T(M') = T(M)^T$. We can prove rigorously that $w \in T(M)$ iff $w^T \in T(M')$ by induction on $|w|$. So $T(M)^T = T(M')$. By (viii) of Section 5.2.7, $T(M')$ is regular, i.e. $T(M)^T$ is regular. ∎

If L is a regular set over Σ, then $\Sigma^* - L$ is also regular over Σ.

We construct another DFA $M' = (Q, \Sigma, \delta, q_0, F')$ by defining $F' = Q - F$, i.e. M and M' differ only in their final states. A final state of M' is a nonfinal state of M and vice versa. The state diagrams of M and M' are the same except for the final states.

$w \in T(M')$ if and only if $\delta(q_0, w) \in F' = Q - F$, i.e. iff $w \notin L$. This proves $T(M') = \Sigma^* - X$. ∎

Theorem If X and Y are regular sets over Σ, then $X \cap Y$ is also regular over Σ.

Proof By DeMorgan's law for sets, $X \cap Y = \Sigma^* - ((\Sigma^* - X) \cup (\Sigma^* - Y))$. By ... , $\Sigma^* - X$ and $\Sigma^* - Y$ are regular. So, $(\Sigma^* - X) \cup (\Sigma^* - Y)$ is also regular. By applying Theorem 5.7, once again $\Sigma^* - ((\Sigma^* - X) \cup (\Sigma^* - Y))$ is regular, i.e. $X \cap Y$ is regular. ∎

REGULAR SETS AND REGULAR GRAMMARS

CONSTRUCTION OF A REGULAR GRAMMAR GENERATING $T(M)$ FOR A GIVEN DFA M

Let $M = (\{q_0, \ldots, q_n\}, \Sigma, \delta, q_0, F)$. If w is in $T(M)$, then it is obtained by concatenating the labels corresponding to several transitions, the first from q_0 and the last terminating at some final state. So for the grammar G to be constructed, productions should correspond to transitions. Also, there should be provision for terminating the derivation once a transition terminating at some final state is encountered. With these ideas in mind, we construct G as

$$G = (\{A_0, A_1, \ldots, A_n\}, \Sigma, P, A_0)$$

where P is defined by the following rules:

(i) $A_i \to aA_j$ is included in P if $\delta(q_i, a) = q_j \notin F$.
(ii) $A_i \to aA_j$ and $A_i \to a$ are included in P if $\delta(q_i, a) = q_j \in F$.

We can show that $L(G) = T(M)$ by using the construction of P. Such a construction gives

$$A_i \Rightarrow aA_j \quad \text{iff } \delta(q_i, a) = q_j$$
$$A_i \Rightarrow a \quad \text{iff } \delta(q_i, a) \in F$$

So.

$$A_0 \Rightarrow a_1A_1 \Rightarrow a_1a_2A_2 \Rightarrow \ldots \Rightarrow a_1 \ldots a_{k-1}A_k \Rightarrow a_1a_2 \ldots a_k$$

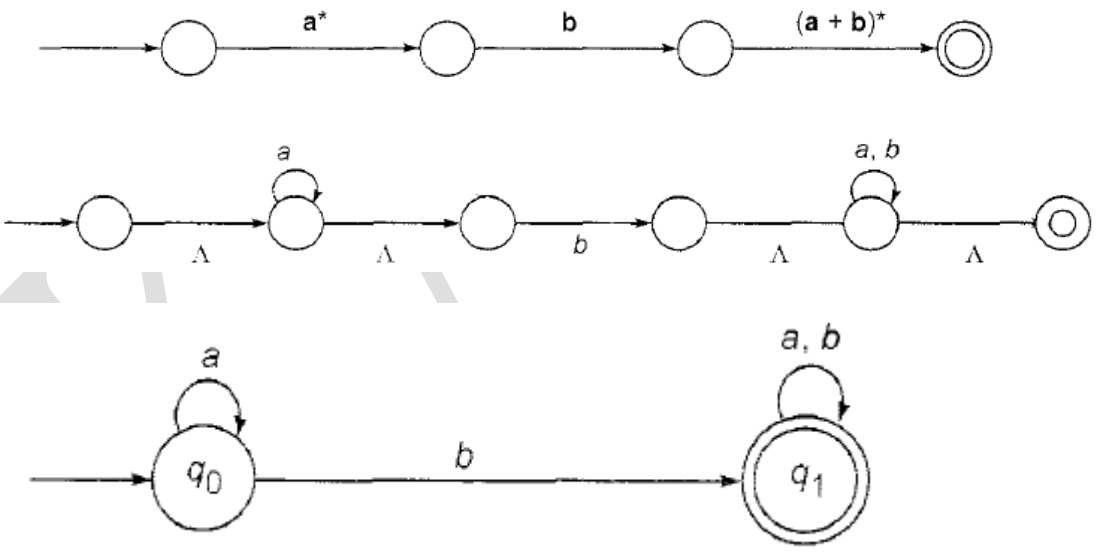
iff $\delta(q_0, a_1) = q_1, \quad \delta(q_1, a_2) = q_2, \ldots, \quad \delta(q_k, a_k) \in F$

This proves that $w = a_1 \ldots a_k \in L(G)$ iff $\delta(q_0, a_1 \ldots a_k) \in F$, i.e. iff $w \in T(M)$.

Construct a regular grammar G generating the regular set represented by $\mathbf{P = a^*b(a + b)^*}$.

Solution

We construct the DFA corresponding to **P** using the construction



Let $G = (\{A0, A_1\}, \{a, b\}, P, A_0)$. where P is given by

$$A_0 \to aA_0, \quad A_0 \to bA_1, \quad A_0 \to b$$
$$A_1 \to aA_1, \quad A_1 \to bA_1, \quad A_1 \to a, \quad A_1 \to b$$

G is the required regular grammar.

CONSTRUCTION OF A TRANSITION SYSTEM M ACCEPTING $L(G)$ FOR A GIVEN REGULAR GRAMMAR G

Let $G = (\{A_0, A_1, \ldots, A_n\}, \Sigma, P, A_0)$. We construct a transition system M whose (i) states correspond to variables, (ii) initial state corresponds to A_0, and (iii) transitions in M correspond to productions in P. As the last production applied in any derivation is of the form $A_i \to a$, the corresponding transition terminates at a new state, and this is the unique final state.

We define M as $(\{q_0, \ldots, q_n, q_f\}, \Sigma, \delta, q_0, \{q_f\})$ where δ is defined as follows:

(i) Each production $A_i \to aA_j$ induces a transition from q_i to q_j with label a.
(ii) Each production $A_k \to a$ induces a transition from q_k to q_f with label a.

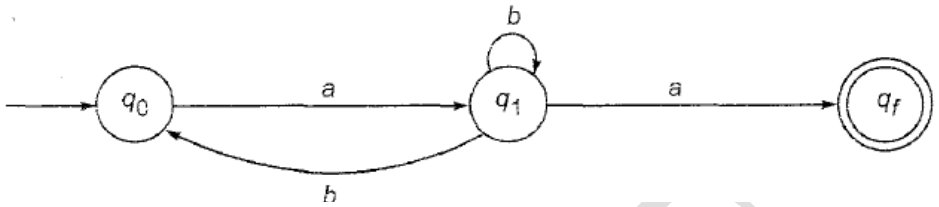
From the construction it is easy to see that $A_0 \Rightarrow a_1A_1 \Rightarrow a_1a_2A_2 \Rightarrow \ldots \Rightarrow a_1 \ldots a_{n-1}A_{n-1} \Rightarrow a_1 \ldots a_n$ is a derivation of $a_1a_2 \ldots a_n$ iff there is a path in M starting from q_0 and terminating in q_f with path value $a_1a_2 \ldots a_n$. Therefore, $L(G) = T(M)$.

Let $G = (\{A_0, A_1\}, \{a, b\}, P, A_0)$, where P consists of $A_0 \to aA_1$, $A_1 \to bA_1$, $A_1 \to a$, $A_1 \to bA_0$. Construct a transition system M accepting $L(G)$.

Solution

Let $M = (\{q_0, q_1, q_f\}, \{a, b\}, \delta, q_0, \{q_f\})$, where q_0 and q_1 correspond to A_0 and A_1, respectively and q_f is the new (final) state introduced. $A_0 \to aA_1$ induces a transition from q_0 to q_1 with label a. Similarly, $A_1 \to bA_1$ and $A_1 \to bA_0$ induce transitions from q_1 to q_1 with label b and from q_1 to q_0 with

label b, respectively. $A_1 \to a$ induces a transition from q_1 to q_f with label a. M is given in Fig. 5.33.

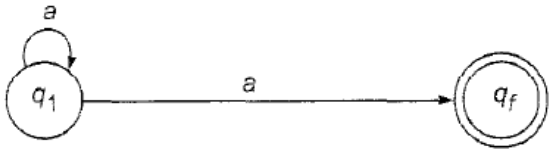


If a regular grammar G is given by $S \to aS \mid a$, find M accepting $L(G)$.

Solution

Let q_0 correspond to S and q_f be the new (final) state. M is given in Fig. 5.34. Symbolically,

$$M = (\{q_0, q_f\}, \{a\}, \delta, q_0, \{q_f\})$$



Find the regular expression representing the set of all strings of the form

(a) $a^m b^n c^p$ where $m, n, p \geq 1$
(b) $a^m b^{2n} c^{3p}$ where $m, n, p \geq 1$
(c) $a^n b a^{2m} b^2$ where $m \geq 0, n \geq 1$

Solution

(a) **aa*bb*cc***
(b) **aa*(bb)(bb)*ccc(ccc)***
(c) **aa*b(aa)*bb**

Find the sets represented by the following regular expressions.

(a) $(\mathbf{a} + \mathbf{b})^*(\mathbf{aa} + \mathbf{bb} + \mathbf{ab} + \mathbf{ba})^*$
(b) $(\mathbf{aa})^* + (\mathbf{aaa})^*$
(c) $(\mathbf{1} + \mathbf{01} + \mathbf{001})^*(\Lambda + \mathbf{0} + \mathbf{00})$
(d) $\mathbf{a} + \mathbf{b}(\mathbf{a} + \mathbf{b})^*$

Solution

(a) The set of all strings having an odd number of symbols from $\{a, b\}^*$
(b) $\{x \in \{a\}^* \mid |x| \text{ is divisible by 2 or 3}\}$
(c) The set of all strings over $\{0, 1\}$ having no substring of more than two adjacent 0's.
(d) $\{a, b, ba, bb, baa, bab, bba, bbb, \ldots\}$

Show that $\{w \in \{a, b\}^* \mid w$ contains an equal number of a's and b's$\}$ is not regular.

Solution

We prove this by contradiction. Assume that $L = T(M)$ for some DFA M with n states. Let $w = a^n b^n \in L$ and $|w| = 2^n$. Using the pumping lemma, we write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$. As $xyz = a^n b^n$, $xy = a^i$ where $i \leq n$ and hence $y = a^j$ for some j, $1 \leq j \leq n$. Consider xy^2z. Now xyz has an equal number of a's and b's. But xy^2z has $(n + j)$ a's and n b's. As $n + j \neq n$, $xy^2z \notin L$. This contradiction proves that L is not regular.

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

Solution

We prove this by contradiction. Assume $L = T(M)$ for some DFA with n states. Choose $w = a^n b^n c^{3n}$ in L. Using the pumping lemma, we write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$. As $w = a^n b^n c^{3n}$, $xy = a^i$ for some $i \leq n$. This means that $y = a^j$ for some j, $1 \leq j \leq n$. Then $xy^{k+1}z = a^{n+jk} b^n c^{3n}$. Choosing k large enough so that $n + jk > 2n$, we can make $n + jk + n > 3n$. So, $xy^{k+1}z \notin L$. Hence L is not regular.

Prove that $\mathbf{P + PQ^*Q = a^*bQ^*}$ where $\mathbf{P = b + aa^*b}$ and $\mathbf{Q}$ is any regular expression.

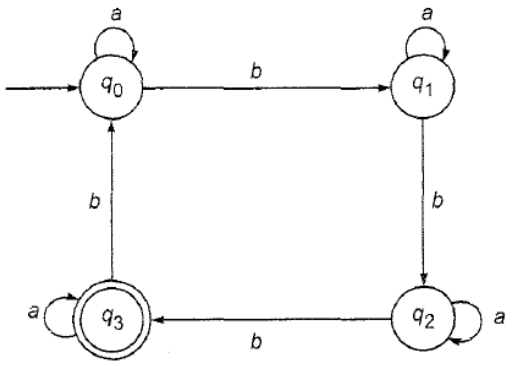
Proof

$$\begin{aligned}
\text{L.H.S.} &= \mathbf{PA + PQ^*Q} && \text{by } I_3\\
&= \mathbf{P(\Lambda + Q^*Q)} && \text{by } I_{12}\\
&= \mathbf{PQ^*} && \text{by } I_9\\
&= \mathbf{(b + aa^*b)Q^*} && \text{by definition of } \mathbf{P}\\
&= \mathbf{(\Lambda b + aa^*b)Q^*} && \text{by } I_3\\
&= \mathbf{(\Lambda + aa^*)bQ^*} && \text{by } I_{12}\\
&= \mathbf{a^*bQ^*} && \text{by } I_9\\
&= \text{R.H.S.}
\end{aligned}$$

Construct a regular grammar accepting $L = \{w \in \{a, b\}^* \mid w \text{ is a string over } \{a, b\} \text{ such that the number of } b\text{'s is 3 mod 4}\}$.

Solution

We construct a DFA M accepting L directly. The symbol a can occur in any place in w and b has to occur in $4k + 3$ places, where $k \geq 0$. So we can have states q_i, $i = 0, 1, 2, 3$, for remembering that the string processed so far has $4k$, $4k + 1$, $4k + 2$ and $4k + 3$ b's ($k \geq 0$). q_3 is the only final state. Also M does not change state on reading a's. The state diagram representing M is



$G = (\{A_0, A_1, A_2, A_3\}, \{a, b\}, P, A_0)$ where P consists of $A_0 \to aA_0$, $A_0 \to bA_1$, $A_1 \to bA_1$, $A_1 \to bA_2$, $A_2 \to aA_2$, $A_2 \to bA_3$, $A_2 \to b$, $A_3 \to aA_3$, $A_3 \to aA_0$.

Possible Questions:

Part-A(6 mark)

1. State and prove pumping lemma for regular sets.
2. Show that the set $L = \{ a\hat{} i^2\} \mid i \geq 1\}$ is not regular.
3. Show that $L = \{a^p \mid p \text{ is prime } \}$ is not regular.
4. State and prove Kleen's theorem.
5. Show that $L = \{ww \mid w \in \{a, b\}^*\}$ is not regular.
6. Prove that If L is regular then L^T is also regular.
7. Prove that If L is a regular set over Σ. Then $\Sigma^* - L$ is also regular over Σ.
8. Construct a regular grammar G generating the regular set represented by **P** = **a*b**(**a + b**)*.
9. Let $G = (\{A_0, A_1\}, \{a, b\}, P, A_0)$, where P consists of $A_0 \rightarrow a\ A_1$, $A_1 \rightarrow b\ A_1$, $A_1 \rightarrow a$, $A_1 \rightarrow b\ A_0$. Construct a transition system M accepting L(G).
10. Prove that **P + PQ*Q = a*bQ*** where **P = b + aa*b** and **Q** is any regular expression.

Part C(10 mark)

1. Show that $L = \{0^i\ 1^i \mid i \geq 1\}$ is not regular.

KARPAGAM ACADEMY OF HIGHER EDUCATION
DEPARTMENT OF MATHEMATICS
FORMAL LANGUAGES AND AUTOMATA THEORY (17MMP305A)

Questions	choice 1	choice 2	choice 3	choice 4	Answer
		UNIT - IV			
Which of the following regular expressions represents the set of strings which do not contain a substring 'rt' if ∑= {r, t}	(rt)*	(tr)*	(r*t*)	(t*r*)	**(t*r*)**
Arden's theorem is true for:	More than one ini	Null transitions	Non-null transitions	state	Non-null transitions
(0+ε) (1+ε) represents	{0, 1, 01, ε}	{0, 1, ε}	{0, 1, 01 ,11, 00, 10, ε}	{0, 1}	{0, 1, 01, ε}
Regular Expression denote precisely the _____ of Regular Language.	Class	Power Set	Super Set	set	Class
Which of the following is not a negative property of Context free languages?	Intersection	Complement	Both (a) and (b)	plus	Both (a) and (b)
The intersection of context free language and regular language is _____	regular language	context free language	context sensitive l	language	context free language
Which of the following is regular?	$a^{100}b^{100}$	$(a+b)^*-\{a^{100}b^{100}\}$	Both (a) and (b)	ab	Both (a) and (b)
Which of the following can be used to prove a language is not context free?	Ardens theorem	Power Construction method	Regular Closure	kleen	Regular Closure
Which of the following can be used to prove a language is not context free?	Arden's theorem	Power Construction method	Regular Closure	kleen's theorem	Regular Closure
The most suitable data structure used to represent the derivations in compiler:	Queue	Linked List	Tree	Hash Tables	Tree
n which order are the children of any node ordered?	From the left	From the right	Arbitrarily	empty	From the left
Which among the following is the root of the parse tree?	Production P	Terminal T	Variable V	Starting Variable S	Starting Variable S
For the expression E*(E) where * and brackets are the operation, number of nodes in the respective parse tree are:	6	**7**	5	2	**7**
The number of leaves in a parse tree with expression E*(E) where * and () are operators	5	2	4	3	5
A grammar with more than one parse tree is called:	Unambiguous	Ambiguous	Regular	set	Unambiguous
_____ is the acyclic graphical representation of a grammar.	Binary tree	Oct tree	Parse tree	tree	Parse tree
Grammar is checked by which component of compiler	Scanner	Parser	Semantic Analyzer	accepter	Parser
Which of the following are distinct to parse trees?	abstract parse trees	sentence diagram	both abstract parse trees and sentence diagrams	trees	both abstract parse trees and sentence diagrams
ε- closure of q1 in the given transition graph:	{q1	{q0, q2}	{q1, q2}	{q0, q1, q2}	{q1, q2}
Predict the total number of final states after removing the ε-moves from the given NFA?	1	2	3	0	3

UNIT-V

Context free Languages and Derivation trees - Ambiguity in Context free grammars - Simplification of Context free grammars (examples only).

CONTEXT-FREE LANGUAGES AND DERIVATION TREES

Context-free languages are applied in parser design. They are also useful for describing block structures in programming languages. It is easy to visualize derivations in context-free languages as we can represent derivations using tree structures.

Construct a context-free grammar G generating all integers (with sign).

Solution

Let

$$G = (V_N, \Sigma, P, S)$$

where

$$V_N = \{S, \text{(sign)}, \text{(digit)}, \text{(Integer)}\}$$

$$\Sigma = \{0, 1, 2, 3, \ldots, 9, +, -\}$$

P consists of $S \to \langle \text{sign} \rangle \langle \text{integer} \rangle$, $\langle \text{sign} \rangle \to + \mid -$,

$$\langle \text{integer} \rangle \to \langle \text{digit} \rangle \langle \text{integer} \rangle \mid \langle \text{digit} \rangle$$

$$\langle \text{digit} \rangle \to 0 \mid 1 \mid 2 \mid \ldots \mid 9$$

$L(G)$ = the set of all integers. For example, the derivation of -17 can be obtained as follows:

$$S \Rightarrow \langle \text{sign} \rangle \langle \text{integer} \rangle \Rightarrow - \langle \text{integer} \rangle$$

$$\Rightarrow - \langle \text{digit} \rangle \langle \text{integer} \rangle \Rightarrow - 1 \langle \text{integer} \rangle \Rightarrow - 1 \langle \text{digit} \rangle$$

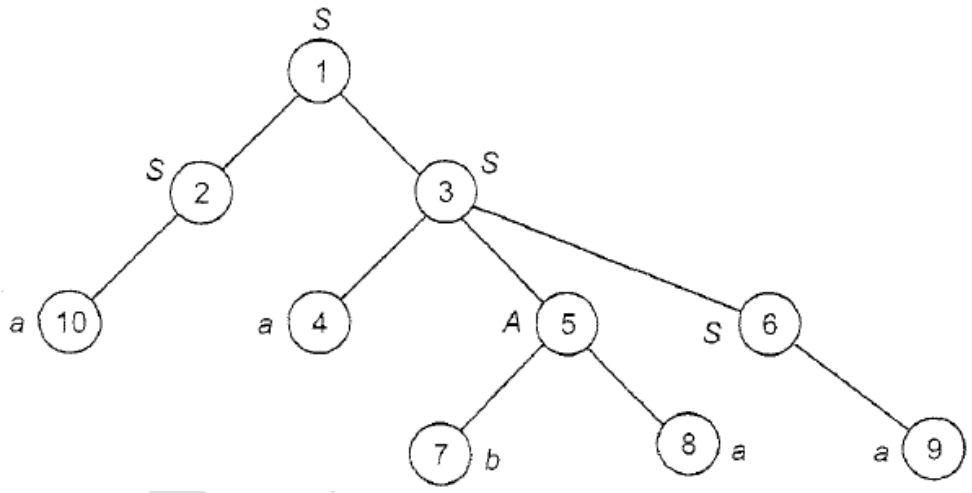
$$\Rightarrow - 17$$

DERIVATION TREES

Definition . A derivation tree (also called a parse tree) for a CFG $G = (V_N, \Sigma, P, S)$ is a tree satisfying the following conditions:

(i) Every vertex has a label which is a variable or terminal or Λ.
(ii) The root has label S.
(iii) The label of an internal vertex is a variable.
(iv) If the vertices $n_1, n_2, \ldots, n_k$ written with labels $X_1, X_2, \ldots, X_k$ are the sons of vertex n with label A, then $A \to X_1X_2 \ldots X_k$ is a production in P.
(v) A vertex n is a leaf if its label is $a \in \Sigma$ or Λ; n is the only son of its father if its label is Λ.

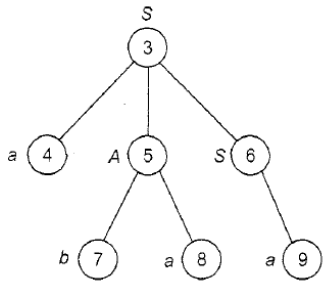
For example, let $G = (\{S, A\}, \{a, b\}, P, S)$, where P consists of $S \to aAS \mid a \mid SS$, $A \to SbA \mid ba$.



Ordering of Leaves from the Left

Definition The yield of a derivation tree is the concatenation of the labels of the leaves without repetition in the left-to-right ordering.

Definition A subtree of a derivation tree T is a tree (i) whose root is some vertex v of T. (ii) whose vertices are the descendants of v together with their labels, and (iii) whose edges are those connecting the descendants of v.

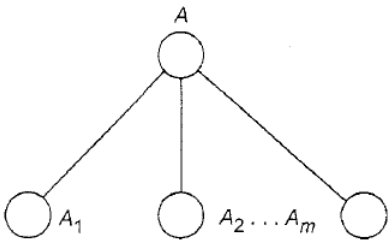


A subtree

Theorem Let $G = (V_N, \Sigma, P, S)$ be a CFG. Then $S \overset{*}{\Rightarrow} \alpha$ if and only if there is a derivation tree for G with yield α.

Proof We prove that $A \overset{*}{\Rightarrow} \alpha$ if and only if there is an A-tree with yield α. Once this is proved, the theorem follows by assuming that $A = S$.

Let α be the yield of an A-tree T. We prove that $A \overset{*}{\Rightarrow} \alpha$ by induction on the number of internal vertices in T.



By condition (iv) of Definition $A \to A_1A_2 \ldots A_m = \alpha$ is a production in G, i.e. $A \Rightarrow \alpha$. Thus there is basis for induction. Now assume the result for all trees with at most $k - 1$ internal vertices ($k > 1$).

Let T be an A-tree with k internal vertices ($k \geq 2$). Let $v_1, v_2, \ldots, v_m$ be the sons of the root in the left-to-right ordering. Let their labels be $X_1, X_2, \ldots, X_m$. By condition (iv) of Definition , $A \to X_1X_2 \ldots X_m$ is in P, and so

$$A \Rightarrow X_1X_2 \ldots X_m$$

As $k \geq 2$, at least one of the sons is an internal vertex. By the left-to-right ordering of leaves, α can be written as $\alpha_1\alpha_2 \ldots \alpha_m$, where α_i is obtained by the concatenation of the labels of the leaves which are descendants of vertex v_i. If v_i is an internal vertex, consider the subtree of T with v_i as its root. The number of internal vertices of the subtree is less than k (as there are k internal vertices in T and at least one of them, viz. its root, is not in the subtree). So by induction hypothesis applied to the subtree, $X_i \overset{*}{\Rightarrow} \alpha_i$. If v_i is not an internal vertex. i.e. a leaf, then $X_i = \alpha_i$.

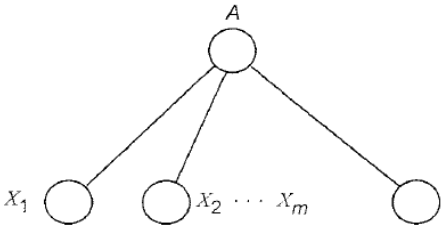
we get

$$A \Rightarrow X_1X_2 \ldots X_m \overset{*}{\Rightarrow} \alpha_1X_2X_3 \ldots X_m \ldots \overset{*}{\Rightarrow} \alpha_1\alpha_2 \ldots \alpha_m = \alpha,$$

i.e. $A \overset{*}{\Rightarrow} \alpha$. By the principle of induction, $A \overset{*}{\Rightarrow} \alpha$ whenever α is the yield of an A-tree.

To prove the 'only if' part, let us assume that $A \overset{*}{\Rightarrow} \alpha$. We have to construct an A-tree whose yield is α. We do this by induction on the number of steps in $A \overset{*}{\Rightarrow} \alpha$.

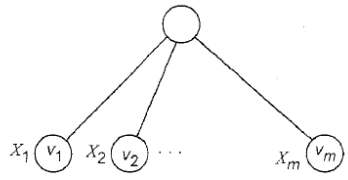
for induction. Assume the result for derivations in at most k steps. Let $A \overset{k}{\Rightarrow} \alpha$: we can split this as $A \Rightarrow X_1 \ldots X_m \overset{k-1}{\Rightarrow} \alpha$. Now, $A \Rightarrow X_1 \ldots X_m$ implies $A \to X_1X_2 \ldots X_m$ is a production in P. In the derivation $X_1X_2 \ldots X_m \overset{k-1}{\Rightarrow} \alpha$, either (i) X_i is not changed throughout the derivation, or (ii) X_i is changed in some subsequent step. Let α_i be the substring of α derived from X_i. Then $X_i \overset{*}{\Rightarrow} \alpha_i$ in (ii) and $X_i = \alpha_i$ in (i). As G is context-free, in every step of the derivation $X_1X_2 \ldots X_m \overset{*}{\Rightarrow} \alpha$, we replace a single variable by a string. As $\alpha_1, \alpha_2, \ldots, \alpha_m$, account for all the symbols in α, we have $\alpha = \alpha_1\alpha_2 \ldots \alpha_m$.

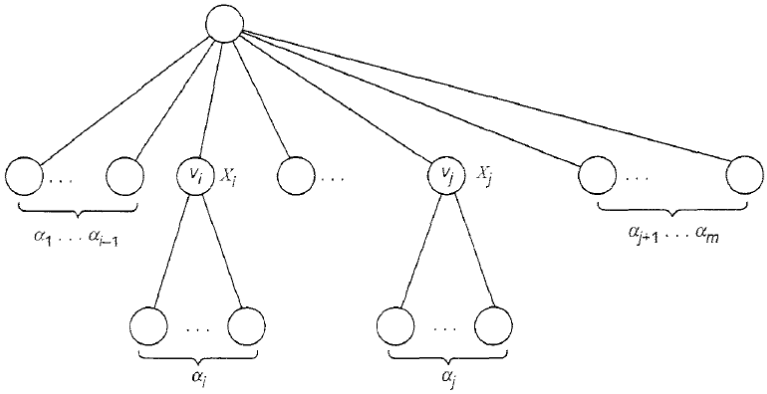


We construct the derivation tree with yield α as follows: As $A \to X_1 \ldots X_m$ is in P, we construct a tree with m leaves whose labels are $X_1, \ldots, X_m$ in the left-to-right ordering.

we leave the vertex v_i as it is. In (ii). $X_i \overset{*}{\Rightarrow} \alpha_i$ is less than k steps (as $X_1 \ldots X_m \overset{n-1}{\Rightarrow} a$). By induction hypothesis there exists an X_i-tree T_i with yield α_i. We attach the tree T_i at the vertex v_i (i.e. v_i is the root of T_i).

let i and j be the first and the last indexes such that X_i and X_j satisfy (ii). So, $\alpha_1 \ldots \alpha_{i-1}$ are the labels of leaves at level 1 in T. α_i is the yield of the X_i-tree T_i, etc.





Thus we get a derivation tree with yield α. By the principle of induction we can get the result for any derivation. This completes the proof of 'only if' part. ∎

Remark If A derives a terminal string w and if the first step in the derivation is $A \Rightarrow A_1A_2 \ldots A_n$, then we can write w as $w_1w_2 \ldots w_n$ so that $A_i \overset{*}{\Rightarrow} w_i$. (Actually, in the derivation tree for w, the ith son of the root has the label A_i, and w_i is the yield of the subtree whose root is the ith son.)

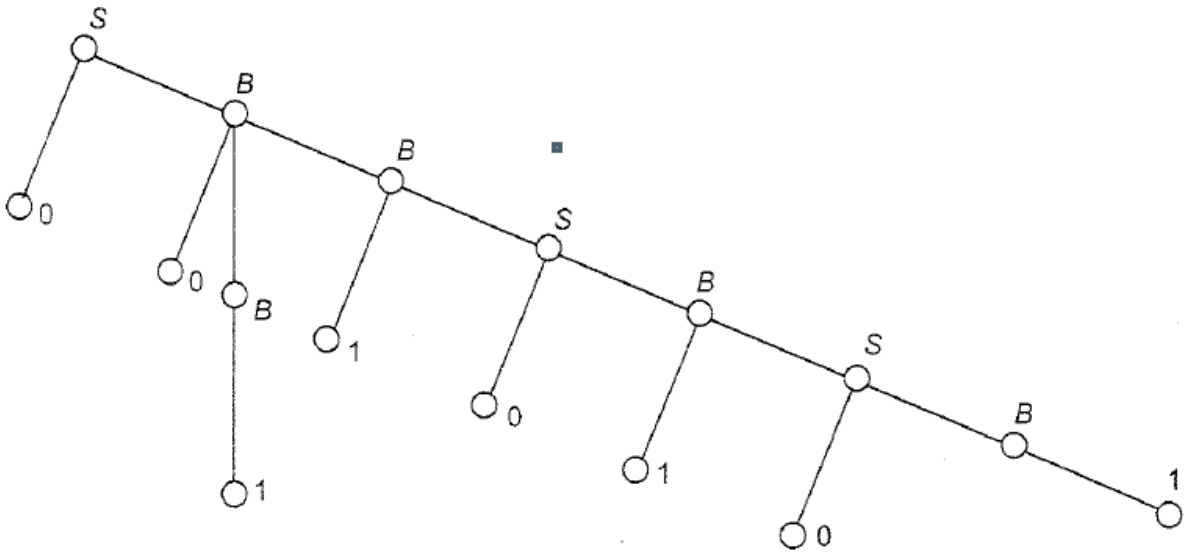
Let G be the grammar $S \to 0B \mid 1A$, $A \to 0 \mid 0S \mid 1AA$, $B \to 1 \mid 1S \mid 0BB$. For the string 00110101, find (a) the leftmost derivation, (b) the rightmost derivation, and (c) the derivation tree.

Solution

(a) $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 001B \Rightarrow 0011S$
$\Rightarrow 0^21^20B \Rightarrow 0^21^201S \Rightarrow 0^21^2010B \Rightarrow 0^21^20101$

(b) $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 00B1S \Rightarrow 00B10B$
$\Rightarrow 0^2B101S \Rightarrow 0^2B1010B \Rightarrow 0^2B10101 \Rightarrow 0^2110101.$

(c) The derivation tree is



AMBIGUITY IN CONTEXT-FREE GRAMMARS

Definition A terminal string $w \in L(G)$ is ambiguous if there exist two or more derivation trees for w (or there exist two or more leftmost derivations of w).

Definition A context-free grammar G is ambiguous if there exists some $w \in L(G)$, which is ambiguous.

SIMPLIFICATION OF CONTEXT-FREE GRAMMARS

In a CFG G, it may not be necessary to use all the symbols in $V_N \cup \Sigma$, or all the productions in P for deriving sentences. So when we study a context-free language $L(G)$, we try to eliminate those symbols and productions in G which are not useful for the derivation of sentences.

Consider, for example,

$$G = (\{S, A, B, C, E\}, \{a, b, c\}, P, S)$$

where

$$P = \{S \to AB, A \to a, B \to b, B \to C, E \to c|\Lambda\}$$

It is easy to see that $L(G) = \{ab\}$. Let $G' = (\{S, A, B\}, \{a, b\}, P', S)$, where P' consists of $S \to AB$, $A \to a$, $B \to b$. $L(G) = L(G')$. We have eliminated the symbols C, E and c and the productions $B \to C$, $E \to c|\Lambda$. We note the following points regarding the symbols and productions which are eliminated:

(i) C does not derive any terminal string.
(ii) E and c do not appear in any sentential form.
(iii) $E \to \Lambda$ is a null production.
(iv) $B \to C$ simply replaces B by C.

In this section, we give the construction to eliminate (i) variables not deriving terminal strings, (ii) symbols not appearing in any sentential form, (iii) null productions, and (iv) productions of the form $A \to B$.

CONSTRUCTION OF REDUCED GRAMMARS

Let $G = (V_N, \Sigma, P, S)$ be given by the productions $S \to AB$, $A \to a$, $B \to b$, $B \to C$, $E \to c$. Find G' such that every variable in G' derives some terminal string.

Solution

(a) *Construction of V'_N:*

$W_1 = \{A, B, E\}$ since $A \to a$, $B \to b$, $E \to c$ are productions with a terminal string on the R.H.S.

$$W_2 = W_1 \cup \{A_1 \in V_N | A_1 \to \alpha \text{ for some } \alpha \in (\Sigma \cup \{A, B, E\})^*\}$$
$$= W_1 \cup \{S\} = \{A, B, E, S\}$$
$$W_3 = W_2 \cup \{A_1 \in V_N | A_1 \to \alpha \text{ for some } \alpha \in (\Sigma \cup \{S, A, B, E\})^*\}$$
$$= W_2 \cup \emptyset = W_2$$

Therefore,

$$V'_N = \{S, A, B, F\}$$

(b) *Construction of P':*

$$P' = \{A_1 \to \alpha | A_1, \alpha \in (V'_N \cup \Sigma)^*\}$$

$$= \{S \to AB, A \to a, B \to b, E \to c\}$$

Therefore,

$$G' = (\{S, A, B, E\}, \{a, b, c\}, P', S)$$

Now we prove:

(i) If each $A \in V'_N$, then $A \overset{*}{\underset{G'}{\Rightarrow}} w$ for some $w \in \Sigma^*$; conversely, if $A \overset{*}{\underset{G}{\Rightarrow}} w$, then $A \in V'_N$.
(ii) $L(G') = L(G)$.

To prove (i) we note that $W_k = W_1 \cup W_2 \ldots \cup W_k$. We prove by induction on i that for $i = 1, 2, \ldots, k$, $A \in W_i$ implies $A \overset{*}{\underset{G'}{\Rightarrow}} w$ for some $w \in \Sigma^*$. If $A \in W_1$, then $A \underset{G}{\Rightarrow} w$. So the production $A \to w$ is in P'. Therefore, $A \overset{*}{\underset{G'}{\Rightarrow}} w$. Thus there is basis for induction. Let us assume the result for i. Let $A \in W_{i+1}$. Then either $A \in W_i$, in which case, $A \overset{*}{\underset{G'}{\Rightarrow}} w$ for some $w \in \Sigma^*$ by induction hypothesis. Or, there exists a production $A \to \alpha$ with $\alpha \in (\Sigma \cup w_i)^*$. By definition of P', $A \to \alpha$ is in P'. We can write $\alpha = X_1 X_2 \ldots X_m$, where $X_j \in \Sigma \cup W_i$. If $X_j \in W_i$ by induction hypothesis, $X_j \overset{*}{\underset{G'}{\Rightarrow}} w_j$ for some $w_j \in \Sigma^*$. So, $A \overset{*}{\underset{G'}{\Rightarrow}} w_1 w_2 \ldots w_m \in \Sigma^*$ (when X_j is a terminal, $w_j = X_j$). By induction the result is true for $i = 1, 2, \ldots, k$.

The converse part can be proved in a similar way by induction on the number of steps in the derivation $A \overset{*}{\underset{G}{\Rightarrow}} w$. We see immediately that $L(G') \subseteq L(G)$ as $V'_N \subseteq V_N$ and $P' \subseteq P$. To prove $L(G) \subseteq L(G')$, we need an auxiliary result

$$A \overset{*}{\underset{G'}{\Rightarrow}} w \qquad \text{if } A \overset{*}{\underset{G}{\Rightarrow}} w \text{ for some } w \in \Sigma^*$$

We prove by induction on the number of steps in the derivation $A \overset{*}{\underset{G}{\Rightarrow}} w$.

If $A \underset{G}{\Rightarrow} w$, then $A \to w$ is in P and $A \in W_1 \subseteq V'_N$. As $A \in V'_N$ and $w \in \Sigma^*$, $A \to w$ is in P'. So $A \underset{G'}{\Rightarrow} w$, and there is basis for induction.

Let $A \overset{k+1}{\underset{G}{\Rightarrow}} w$.

we can split this as $A \underset{G}{\Rightarrow} X_1X_2 \ldots X_m \overset{*}{\underset{G}{\Rightarrow}} w_1w_2 \ldots w_m$ such that $X_j \overset{*}{\underset{G}{\Rightarrow}} w_j$. If $X_j \in \Sigma$, then $w_i = X_j$.

If $X_j \in V_N$ then by (i), $X_j \in V'_N$. As $X_j \overset{*}{\underset{G}{\Rightarrow}} w_j$ in at most k steps, $X_j \overset{*}{\underset{G'}{\Rightarrow}} w_j$. Also, $X_1, X_2, X_m \in (\Sigma \cup V'_N)^*$ implies that $A \to X_1X_2 \ldots X_m$ is in P'. Thus, $A \underset{G'}{\Rightarrow} X_1X_2 \ldots X_m \overset{*}{\underset{G'}{\Rightarrow}} w_1w_2 \ldots w_m$. Hence by induction, is true for all derivations. In particular, $S \overset{*}{\underset{G}{\Rightarrow}} w$ implies $S \overset{*}{\underset{G'}{\Rightarrow}} w$. This proves that $L(G) \subseteq L(G')$, and (ii) is completely proved.

Find a reduced grammar equivalent to the grammar G whose productions are

$$S \to AB \mid CA, \quad B \to BC \mid AB, \quad A \to a, \quad C \to aB \mid b$$

Solution

Step 1 $W_1 = \{A, C\}$ as $A \to a$ and $C \to b$ are productions with a terminal string on R.H.S.

$W_2 = \{A, C\} \cup \{A_1 \mid A_1 \to \alpha$ with $\alpha \in (\Sigma \cup \{A, C\})^*\}$

$= \{A, C\} \cup \{S\}$ as we have $S \to CA$

$W_3 = \{A, C, S\} \cup \{A_1 \mid A_1 \to \alpha$ with $\alpha \in (\Sigma \cup \{S, A, C\})^*\}$

$= \{A, C, S\} \cup \emptyset$

As $W_3 = W_2$,

$$V'_N = W_2 = \{S, A, C\}$$
$$P' = \{A_1 \to \alpha \mid A_1, \alpha \in (V'_N \cup \Sigma)^*\}$$
$$= \{S \to CA, A \to a, C \to b\}$$

Thus,

$$G_1 = (\{S, A, C\}, \{a, b\}, \{S \to CA, A \to a, C \to b\}, S)$$

Step 2 We have to apply Theorem 6.4 to G_1. Thus,

$$W_1 = \{S\}$$

As we have production $S \to CA$ and $S \in W_1$, $W_2 = \{S\} \cup \{A, C\}$
As $A \to a$ and $C \to b$ are productions with $A, C \in W_2$, $W_3 = \{S, A, C, a, b\}$

$$\text{As } W_3 = V'_N \cup \Sigma,\ P'' = \{S \to a \mid A_1 \in W_3\} = P'$$

Therefore,

$$G' = (\{S, A, C\}, \{a, b\}, \{S \to CA, A \to a, C \to b\}, S)$$

is the reduced grammar.

ELIMINATION OF NULL PRODUCTIONS

A context-free grammar may have productions of the form $A \to \Lambda$. The production $A \to \Lambda$ is just used to erase A. So a production of the form $A \to \Lambda$, where A is a variable, is called a *null production.*

Definition A variable A in a context-free grammar is nullable if $A \overset{*}{\Rightarrow} \Lambda$.

ELIMINATION OF UNIT PRODUCTIONS

Definition A unit production (or a chain rule) in a context-free grammar G is a production of the form $A \to B$, where A and B are variables in G.

Let G be $S \to AB$, $A \to a$, $B \to C \mid b$, $C \to D$, $D \to E$ and $E \to a$. Eliminate unit productions and get an equivalent grammar.

Solution

Step 1 $W_0(S) = \{S\}$, $W_1(S) = W_0(S) \cup \emptyset$

Hence $W(S) = \{S\}$. Similarly,

$W(A) = \{A\}$, $W(E) = \{E\}$

$W_0(B) = \{B\}$, $W_1(B) = \{B\} \cup \{C\} = \{B, C\}$

$W_2(B) = \{B, C\} \cup \{D\}$, $W_3(B) = \{B, C, D\} \cup \{E\}$, $W_4(B) = W_3(B)$

Therefore,

$$W(B) = \{B, C, D, E\}$$

Similarly,

$W_0(C) = \{C\}$, $W_1(C) = \{C, D\}$. $W_2\{C\} = \{C, D, E\} = W_3(C)$

Therefore,

$$W(C) = \{C, D, E\}, \quad W_0(D) = \{D\}$$

Hence,

$$W_1(D) = \{D, E\} = W_2(D)$$

Thus,

$$W(D) = \{D, E\}$$

Step 2 The productions in G_1 are

$S \to AB$, $A \to a$, $E \to a$

$B \to b \mid a$, $C \to a$, $D \to a$

By construction, G_1 has no unit productions.
To complete the proof we have to show that $L(G') = L(G_1)$.

Step 3 $L(G') = L(G)$. If $A \to \alpha$ is in $P_1 - P$, then it is induced by $B \to \alpha$ in P with $B \in W(A)$, $\alpha \notin V_N$. $B \in W(A)$ implies $A \overset{*}{\underset{G'}{\Rightarrow}} B$. Hence, $A \overset{*}{\underset{G'}{\Rightarrow}} B \underset{G'}{\Rightarrow} \alpha$. So, if $A \underset{G_1}{\Rightarrow} \alpha$, then $A \overset{*}{\underset{G'}{\Rightarrow}} \alpha$. This proves $L(G_1) \subseteq L(G')$.

To prove the reverse inclusion, we start with a leftmost derivation

$$S \underset{G}{\Rightarrow} \alpha_1 \underset{G}{\Rightarrow} \alpha_2 \ldots \underset{G}{\Rightarrow} \alpha_n = w$$

in G'.

Let i be the smallest index such that $\alpha_i \overset{*}{\underset{G'}{\Rightarrow}} \alpha_{i+1}$ is obtained by a unit production and j be the smallest index greater than i such that $\alpha_j \underset{G}{\Rightarrow} \alpha_{j+1}$ is obtained by a nonunit production. So, $S \underset{G_1}{\Rightarrow} \alpha_i$, and $\alpha_j \overset{*}{\underset{G'}{\Rightarrow}} \alpha_{j+1}$ can be written as

$$\alpha_i = w_i A_i \beta_i \Rightarrow w_i A_{i+1} \beta_i \Rightarrow \ldots \Rightarrow w_i A_j \beta_i \Rightarrow w_i \gamma \beta_i = \alpha_{j+1}$$

$A_j \in W(A_i)$ and $A_j \to \gamma$ is a nonunit production. Therefore, $A_j \to \gamma$ is a production in P_1. Hence, $\alpha_j \overset{*}{\underset{G_1}{\Rightarrow}} \alpha_{j+1}$. Thus, we have $S \overset{*}{\underset{G_1}{\Rightarrow}} \alpha_{j+1}$.

Repeating the argument whenever some unit production occurs in the remaining part of the derivation, we can prove that $S \overset{*}{\underset{G_1}{\Rightarrow}} \alpha_n = w$. This proves $L(G') \subseteq L(G)$. ∎

Possible Questions:

Part-A(6 mark)

1. Consider G whose productions are $S \rightarrow aAS \mid a$, $A \rightarrow SbA \mid SS \mid ba$. Show that S aabbaa and construct a derivation tree whose yield is aabbaa.
2. Prove that if A w in G, then there is leftmost derivation of w.
3. Let G be the grammar $S \rightarrow 0B \mid 1A$, $A \rightarrow 0 \mid 0S \mid 1AA$, $B \rightarrow 1 \mid 1S \mid 0BB$. For the string 00110101, find (a) the leftmost derivation, (b) the rightmost derivation and (c) the derivation tree.
4. If G is the grammar $S \rightarrow SbS \mid a$, show that G is ambiguous.
5. If G is a CFG such that $L(G) \neq \phi$. Find an equivalent grammar G' such that each variable in G' derives some terminal string.
6. Let $G = (V_N, \Sigma, P, S)$ be given by the productions $S \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$, $B \rightarrow C$, $E \rightarrow c$. Find G such that every variable in G' derives some terminal string.
7. Prove that for every CFG G there exists a reduced grammar G' which is equivalent to G.
8. Find a reduced grammar equivalent to the grammar G whose productions are $S \rightarrow AB \mid CA$, $S \rightarrow BC \mid AB$, $A \rightarrow a$, $C \rightarrow aB \mid b$.
9. Construct a reduced grammar equivalent to the grammar $S \rightarrow aAa$, $A \rightarrow Sb \mid bCC \mid DaA$, $C \rightarrow abb \mid DD$, $E \rightarrow aC$, $D \rightarrow aDA$.
10. Consider the grammar G whose productions are $S \rightarrow aS \mid AB$, $A \rightarrow \Lambda$, $B \rightarrow \Lambda$, $D \rightarrow b$. construct a grammar G_1 without null productions generating $L(G) - \{\Lambda\}$.

Part C(10 mark)

1. Let G be $S \rightarrow AB$, $A \rightarrow a$, $B \rightarrow C \mid b$, $C \rightarrow D$, $D \rightarrow E$ and $E \rightarrow a$. Eliminate unit productions and get an equivalent grammar.

KARPAGAM ACADEMY OF HIGHER EDUCATION
DEPARTMENT OF MATHEMATICS
FORMAL LANGUAGES AND AUTOMATA THEORY (17MMP305A)

Questions	choice 1	choice 2	choice 3	choice 4	Answer
UNIT - V					
Which among the following is the format of unit production?	A->B	A->b	B->Aa	A	A->B
Given Grammar G:S->aA, A->a| A,B->B The number of productions to be removed immediately as Unit productions:	0	1	2	3	2
Given grammar:S->aA ,A->a,A->B,B->A,B->bb Which of the following is the production of B after simplification by removal of unit productions?	A	bb	aA	A| bb	bb
If grammar G is unambiguous, G' produced after the removal of Unit production will be:	ambiguous	unambiguous	finite	nonfinite	unambiguous
A can be A-> derivable if and only if _____	A-> A is actually a production	A->B, B-> A exists	Both (a) and (b)	B	A-> A is actually a production
Which of the following variables in the given grammar is called live variable? S->AB , A->a	S	A	B	c	A
CFGs are more powerful than:	DFA	NDFA	Mealy Machine	All the above	All the above
Which of the following are context free language?	L={a^ib^i|i>=0}	L={ww^r| w is a string and r represents reverse}	Both (a) and (b)	null	L={a^ib^i|i>=0}
e-transitions are _____	conditional	unconditional	input dependent	output	unconditional
The _____ of a set of states, P, of an NFA is defined as the set of states reachable from any state in P following e-transitions.	e-closure	e-pack	Q in the tuple	tuple	e-closure
If d is not defined on the current state and the current tape symbol, then the machine _____	does not halts	halts	goes into loop fc	stuck	halts
RASP stands for:	Random access storage program	Random access stored program	Randomly accessed stored program	Random access storage programming	Random access stored program
Enumerator is a turing machine with _____	an output printe	5 input tapes	a stack	1 tape	an output printer
For the following language, an enumerator will print: L={anbn|n>=0}	a^nb^n	{ab, a^2b^2, a^3b^3, …}	{e, ab, a^2b^2, a^3b^3, …}	ab	{ab, a^2b^2, a^3b^3, …}
L is a regular Language if and only If the set of _____ classes of IL is finite.	Equivalence	Reflexive	Myhill	Nerode	Equivalence
While applying Pumping lemma over a language, we consider a string w that belong to L and fragment it into _____ parts.	2	5	3	6	3