THEORY OF COMPUTATION

For
COMPUTER SCIENCE

THEORY OF COMPUTATION

Syllabus

Regular expressions and finite automata. Context-free grammars and push-down automata. Regular and contex-free languages, pumping lemma. Turing machines and undecidability.

ANALYSIS OF GATE PAPERS

Exam Year	1 Mark Ques.	2 Mark Ques.	Total
2003	3	6	15
2004	1	4	9
2005	-	7	14
2006	2	5	12
2007	2	5	12
2008	3	5	13
2009	4	3	11
2010	1	3	7
2011	3	3	9
2012	4	1	6
2013	2	3	8
2014 Set-1	2	2	6
2014 Set-2	2	2	6
2014 Set-3	2	2	6
2015 Set-1	1	2	5
2015 Set-2	1	3	7
2015 Set-3	1	1	3
2016 Set-1	3	3	9
2016 Set-2	3	4	11
2017 Set-1	2	4	10
2017 Set-2	3	4	11
2018	1	2	5

CONTENTS

1 FINITE AUTOMATA AND REGULAR EXPRESSION

1.1 ALPHABETS, STRINGS & LANGUAGES

The notion of natural languages like English, Hindi, etc. is familiar to us.
Informally, language can be defined as a system suitable for expression of certain ideas, facts, or concepts, which includes a set of symbols and rules to manipulate these.
The languages we consider for our discussion is an abstraction of natural languages. That is, our focus here is on formal languages that need precise and formal definitions. Programming languages belong to this category.

Symbols:

Symbols are indivisible objects or entity that cannot be defined. (Symbols are the atoms of the world of languages)
A symbol is any single object such as ♠, a, 0, 1, #, begin, or do.

Alphabets:

An alphabet is a finite, nonempty set of symbols. The alphabet of a language is normally denoted by Σ. When more than one alphabets are considered for discussion, then subscripts may be used (e.g. Σ_1, $\Sigma_2$etc) or sometimes other symbol like G may also be introduced.

Strings or Words over Alphabet

A string or word over an alphabet Σ is a finite sequence of concatenated symbols ofΣ.

Example
0110, 11, 001 are three strings over the binary alphabet {0, 1}. aab,abcb, b, cc are four strings over the alphabet { a, b, c }.

It is not the case that a string over some alphabet should contain all the symbols from the alphabet. For example, the string cc over the alphabet { a, b, c } does not contain the symbols a and b. Hence, it is true that a string over an alphabet is also a string over any superset of that alphabet.

Length of A String:
The number of symbols in a string w is called its length, denoted by |w|.

Example:
|011|=4, |11|=2, |b|=1
It is convenient to introduce a notation ε for the empty string, which contains no symbols at all. The length of the empty string ε is zero, i.e., |ε| = 0.

Some String Operations

Let $x = a_1a_2a_3 \in a_n$ and $y = b_1b_2b_3 \in b_m$ be two strings.
Concatenation: The concatenation of x and y denoted by xy, is the string $a_1a_2a_3b_1b_2b_3$ That is, the concatenation of x and y denoted by xy is the string that has a copy of x followed by a copy of y without any intervening space between them.

Example
Concatenation of the strings 0110 and 11 is 011011 and concatenation of the strings good and boy is good boy.

Note:
For any string w, wε=εw = w.
It is also obvious that if | x | = n and | y |= m, Then | x + y | = n + m.
Prefix: u is a prefix of v if v =ux for some string x.
Suffix: u is a suffix of v if v =xu for some string x.

Substring: u is a substring of v if v =xuy for some strings x and y.

Example
Consider the string 011 over the binary alphabet. All the prefixes, suffixes and substrings of this string are listed below.
Prefixes: ε, 0, 01, 011.
Suffixes: ε, 1, 11, 011.
Substrings: ε, 0, 1, 01, 11, 011.
Note that x is a prefix (suffix or substring) to x, for any string x and ε is a prefix (Suffix or substring) to any string.
A string x is a proper prefix (suffix) of string y if x is a prefix (suffix) of y and x≠y.
In the above example, all prefixes(suffixes) except 011 are proper prefixes(Suffixes).

Powers of Strings
For any string x and integer n ≥ 0, we use x^n to denote the string formed by sequentially concatenating n copies of x. We can also give an inductive definition of x^n as follows: x^n = ε, if n = 0 ; otherwise $x^n = xx^{n-1}$

Example
If x = 011, then x^3= 011011011, x^1= 011 and $x^0 = \varepsilon$

Powers of Alphabets
We write$\sum^k$(for some integer k) to denote the set of strings of length k with symbols from ∑. In other words,$\sum^k$={w| w is a string over ∑ and |w|=k}. Hence, for any alphabet, $\sum^0$denotes the set of all strings of length zero. That is, $\sum^0$= {ε}. For the binary alphabet {0, 1} we have the following.
$\sum^0 = \{\varepsilon\}$
$\sum^1 = \{0,1\}$
$\sum^2 = \{00, 01, 10, 11\}$
$\sum^3 = \{000,001,010,011, 100, 101, 110, 111\}$
The set of all strings over an alphabet ∑ is denoted by $\sum^*$. That is,
$\sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \cup \text{---} \sum^n \cup \text{---} = \cup \sum^k$ (K is 0 to ∞)

The set $\sum^*$ contains all the strings that can be generated by iteratively concatenating symbols from ∑ any number of times.

Example
If ∑ = { a, b }, then $\sum^*$ = {ε, a, b,aa,ab,ba, bb,aaa,aab, aba,abb, baa, ...}.
Please note that if ∑ = Ø, then $\sum^*$ that is Ø*={ε}. It may lookodd that one can proceed from the empty set to a non-empty set by iterated concatenation. But there is a reason for this and we accept this convention. The set of all nonempty strings over an alphabet ∑ is denoted by $\sum^+$. That is,

$\sum^+ = \sum^1 \cup \sum^2 \cup \text{---} \sum^* \cup \text{---} = \cup \sum^k$ (k is 1 to ∞)

Note that k is infinite. It contains no infinite strings but strings of arbitrary lengths.

Reversal
For any string w = $a_1a_2a_3\cdots a_n$the reversal of the string is $w^R = a_n a_{n-1}\cdots a_3\, a_2\, a_1$.
For example if the string is x = abb then reverse of x = bba.

Languages
A language over an alphabet is a set of strings over that alphabet. Therefore, a language L is any subset of $\sum^*$. That is, any L ⊆ $\sum^*$ is a language.

Example
1. ϕ is the empty language.
2. $\sum^*$is a language for any∑.
3. {ε } is a language for any ∑.
   Note that,ϕ ≠ { ε}. Because the language F does not contain any string but {ε } contains one string of length zero.
4. The set of all strings over {0, 1} containing equal number of 0's and 1's.
5. The set of all strings over {a,b,c} that starts with a.

1.2 OPERATIONS ON LANGUAGES

Since languages are set of strings we can apply set operations to languages. Here are some simple examples (though there is nothing new in it).

Union
A string $x \in L_1 \cup L_2$ iff $x \in L_1$ or $x \in L_2$

Example
$\{ 0, 11, 01, 011 \} \cup \{ 1, 01, 110 \} = \{ 0, 1, 11, 01, 011, 110 \}$

Intersection
A string $x \in L_1 \cap L_2$ iff $x \in L_1$ and $x \in L_2$

Example
$\{ 0, 11, 01, 011 \} \cap \{ 1, 01, 110 \} = \{ 01 \}$

Complement
Usually, $\sum^*$ is the universe that a complement is taken with respect to. Thus for a language L, thecomplement is L(bar) = $\{ x \in \sum^* \mid x \notin L \}$.

Example
Let L = {x | |x| is even}. Then its complement is the language $\{ x \in \sum^* \mid |x|$ is odd}. Similarly we can define other usual set operations on languages like relative complement, symmetric difference, etc.

Reversal of A Language
The reversal of a language L, denoted as L^R, is defined as: $L^R = \{ W^R \mid W \in L \}$.

Example
1. Let L = { 0, 11, 01, 011 }. Then L^R = { 0, 11, 10, 110 }.
2. Let L = { 1^n0^n| n is an integer }. Then L^R = {0^n1^n | n is an integer }.

Language concatenation
The concatenation of languages L_1 and $L_2$is defined as $L_1 L_2 = \{ xy \mid x \in L_1$and $y \in L_2 \}$.

Example
{ a, ab }{ b, ba } = { ab, aba, abb, abba }.
Note that,
1. $L_1 L_2 \neq L_2 L_1$ in general.
2. $L\Phi = \Phi$ and $\Phi L = \Phi$
3. $L\{ \varepsilon \} = L$ and $\{ \varepsilon \}L = L$

Since we can concatenate two languages, we also repeat this to concatenate any number of languages. Or we can concatenate a language with itself any number of times. The operation L^n denotes the concatenation of L with itself n times. This is defined formally as follows:
$L^0 = \{ \varepsilon \}$
$L^n = LL^{n-1}$

Example
Let L = {a,ab}.
Then according to the definition, we have
$L^0 = \{\varepsilon\}$
$L^1 = L\{\varepsilon\} = L = \{ a, ab \}$
$L^2 = LL_1 = \{a, ab \} \{a, ab\} = \{ aa, aab, aba, abab \}$
$L^3 = LL_2 = \{a, ab\} \{aa, aab, aba, abab\}$
= { aaa, aaab, aaba, aabab, abaa, abaab, ababa,
ababab } and so on.

Kleene's Star operation
The Kleene star operation on a language L, denoted as $L^*$is defined as follows:
$L^* = ($ Union in N and n>=0$) L^n$
$= L^0 \cup L^1 \cup L^2 \cup \cdots$
= { x | x is the concatenation of zero or more strings from L }
Thus L^* is the set of all strings derivable by any number of concatenations of strings in L.

It is also useful to define $L^+ = LL^*$, i.e., all strings derivable by one or more concatenations of strings in L.
That is $L^+ = ($Union n in N and n>0$) L^n$
$= L^1 \cup L^2 \cup L^3 \cup \cdots$

Example
Let L = { a, ab }. Then we have,
$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots$
$= \{ \varepsilon \} \cup \{a, ab\} \cup \{aa, aab, aba, abab\} \cup \ldots$
$L^+ = L^1 \cup L^2 \cup L^3 \cup \cdots$
$= \{a, ab\} \cup \{aa, aab, aba, abab\} \cup \ldots$

1.3 AUTOMATA AND GRAMMARS

Automata

An automata is an abstract computing device (or machine). There are different verities of such abstract machines (also called models of computation) which can be defined mathematically. Some of them are

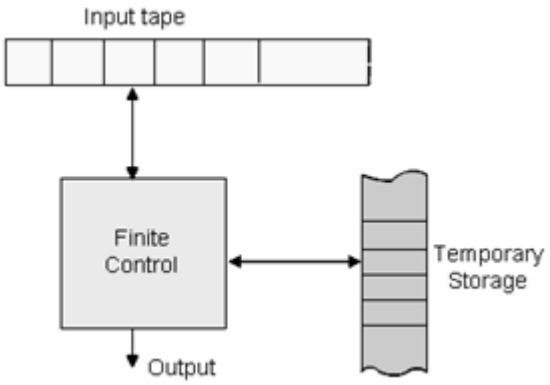
as powerful in principle as today's real computers, while the simpler ones are less powerful. (Some models are considered even more powerful than any real computers as they have infinite memory and are not subject to physical constraints on memory unlike in real computers). Studying the simpler machines are still worth as it is easier to introduce some formalisms used in theory.

- Every automaton consists of some essential features as in real computers. It has a mechanism for reading input. The input is assumed to be a sequence of symbols over a given alphabet and is placed on an input tape(or written on an input file). The simpler automata can only read the input one symbol at a time from left to right but not change. Powerful versions can both read (from left to right or right to left) and change the input.

- The automaton can produce output of some form. If the output in response to an input string is binary (say, accept or reject), then it is called an accepter. If it produces an output sequence in response to an input sequence, then it is called a transducer(or automaton with output).

- The automaton may have a temporary storage, consisting of an unlimited number of tape cells, each capable of holding a symbol from an alphabet (which may be different from the input alphabet). The automaton can both read and change the contents of the storage cells in the temporary storage. The accusing capability of this storage varies depending on the type of the storage.

- The most important feature of the automaton is its control unit, which can be in any one of a finite number of interval states at any point. It can change state in some defined manner determined by a transition function



Diagrammatic representation of a generic automation.

Operation of the automation is defined as follows.

- At any point of time the automaton is in some integral state and is reading a particular symbol from the input tape by using the mechanism for reading input. In the next time step the automaton then moves to some other integral (or remain in the same state) as defined by the transition function. The transition function is based on the current state, input symbol read, and the content of the temporary storage. At the same time the content of the storage may be changed and the input read may be modified. The automation may also produce some output during this transition. The internal state, input and the content of storage at any point defines the configuration of the automaton at that point. The transition from one configuration to the next (as defined by the transition function) is called a move. Finite state machine or Finite Automation is the simplest type of abstract machine we consider. Any system that is at any point of time in one of a finite number of interval state and moves among these states in a defined manner in response to some input, can be modeled by a finite automaton. It does not have any temporary storage

and hence a restricted model of computation.

Grammar

A grammar is a mechanism used for describing languages. This is one of the most simple but yet powerful mechanism. There are other notions to do the same, of course.
In everyday language, like English, we have a set of symbols (alphabet), a set of words constructed from these symbols, and a set of rules using which we can group the words to construct meaningful sentences. The grammar for English tells us what are the words in it and the rules to construct sentences. It also tells us whether a particular sentence is well-formed (as per the grammar) or not. But even if one follows the rules of the English grammar it may lead to some sentences which are not meaningful at all, because of impreciseness and ambiguities involved in the language. In English grammar we use many other higher level constructs like noun-phrase, verb-phrase, article, noun, predicate, verb etc.
A typical rule can be defined as < sentence >→< noun-phrase ><predicate> meaning that "a sentence can be constructed using a 'noun-phrase' followed by apredicate".
Some more rules are as follows:

< noun-phrase >→< article >< noun >
<predicate>→< verb >

with similar kind of interpretation given above.
If we take {a, an, the} to be <article>; cow, bird, boy, Ram, pen to be examples of <noun>; and eats, runs, swims, walks, are associated with <verb>, then we can construct the sentence- a cow runs, the boy eats, an pen walks- using the above rules. Even though all sentences are well-formed, the last one is not meaningful. We observe that we start with the higher level construct <sentence> and then reduce it to <noun-phrase>, <article>, <noun>, <verb> successively, eventually leading to a group of words associated with these constructs.

These concepts are generalized in formal language leading to formal grammars. The word 'formal' here refers to the fact that the specified rules for the language are explicitly stated in terms of what strings or symbols can occur. There can be no ambiguity in it.

Formal definitions of a Grammar

A grammar G is defined as a quadruple.
$G = (N, \sum , P, S)$
N is a non-empty finite set of non-terminals or variables, $\sum$ is a non-empty finite set of terminal symbols such that $N \cap \sum = \emptyset$.
$S \in N$, is a special non-terminal (or variable) called the start symbol, and
$P:(N \cup \sum)^{+} X (N \cup \sum)^{*}$ is a finite set of production rules.
The binary relation defined by the set of production rules is denoted by →, i.e. $\alpha \rightarrow \beta$ if
$\alpha, \beta \in P$. In other words, P is a finite set of production rules of the form $\alpha \rightarrow \beta$, where
$\alpha \in (N \cup \sum)^{+}$, α must contain at least one non-terminal and $\beta \in (N \cup \sum)^{*}$

The production rules specify how the grammar transforms one string to another. Given a string δαγ, we say that the production rule $\alpha \rightarrow \beta$ is applicable to this string, since it is possible to use the rule $\alpha \rightarrow \beta$ to rewrite the α (in δαγ) to β obtaining a new string δβγ We say that δαγ derives δβγ and is denoted as $\delta\alpha\gamma \Rightarrow \delta\beta\gamma$
Successive strings are derived by applying the productions rules of the grammar in any arbitrary order. A particular rule can be used if it is applicable, and it can be applied as many times as described.
We write $\alpha \Rightarrow^{*} \beta$ if the string β can be derived from the string α in zero or more steps; $\alpha \Rightarrow^{+} \beta$ if β can be derived from α in one or more steps.
By applying the production rules in arbitrary order, any given grammar can generate many strings of terminal symbols starting with the special start symbol, S, of the grammar. The set of all such terminal

strings is called the language generated (or defined) by the grammar.
Formally, for a given grammar G = (N, ∑, P, S) the language generated by G is

$$L(G) = \left\{ w \varepsilon \sum^{*} \mid S \xrightarrow{*} w \right\}$$

That is W ∈ L(G) if S ⇒ G. If,W ∈ L(G) we must have for some n ≥ 0,
S = $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \cdots \Rightarrow \alpha_n$, denoted as a derivation sequence of w, The strings
S = $\alpha_1, \alpha_2, \alpha_3 \cdots, \alpha_n = w$ are denoted as sentential forms of the derivation

Example
Consider the grammar G = (N, ∑, P, S) , where N = {S}, ∑ ={a, b} and P is the set of the following production rules { S →ab, S → aSb}
Some terminal strings generated by this grammar together with their derivation is given below.
S ⇒ ab
S ⇒ aSb⇒ aabb
S ⇒ aSb⇒ aaSbb⇒ aaabbb
It is easy to prove that the language generated by this grammar is

$$L(G) = \left\{ a^i S b^i, i \geq 1 \right\}$$

By using the first production, it generates the string ab(for i =1).
To generate any other string, it needs to start with the production S → aSb and then the non-terminal S in the RHS can be replaced either by ab(in which we get the string aabb) or the same production S →aSb can be used one or more times. Every time it adds an 'a' to the left and a 'b'
to the right of S, thus giving the sentential form $a^i S b^i, i \geq 1$. When the non-terminal is replaced by ab (which is then only possibility for generating a terminal string) we get a terminal string of the form $a^i S b^i, i \geq 1$.
There is no general rule for finding a grammar for a given language. For many languages we can devise grammars and there are many languages for which we cannot find any grammar.

Chomsky Hierarchy
The famous linguistic Noam Chomsky attempted to formalize the notion of grammar and languages in the 1950s. This effort, due to Chomsky, resulted in the definition of the "Chomsky Hierarchy", a hierarchy of language classes defined by gradually increasing the restrictions on the form of the productions. Chomsky numbered the four families of grammars (and languages) that make up the hierarchy and are defined as below.
Let G = (N , Σ, P, S) be a grammar

1. G is called a Type-0 or unrestricted, or semi-there or phrase-structure grammar if
   all productions are of the form α → β , where α ∈ (N ∪ ∑)⁺and β ∈ (N ∪ ∑)*

2. G is a Type-1 or context-sensitive grammar if each production α → β in P satisfies |α|≤|β| such that α ∈ (N ∪ ∑)⁺ α must contain at least one non-terminal and β ∈ (N ∪ ∑)*. Type-1 grammar, by specification it is allowed to have the production S → ∈ (esplin), provided S does not appear on the right-hand side of any production.

3. G is a Type-2 or context-free grammar if each production α → β in P, α ∈ N, |α| = 1 satisfies |α| i.e. α is a single non terminal and β ∈ (N ∪ ∑)*.

4. G is a Type-3 or regular grammar if each production has one of the
   Following forms: N→N∑*, N→∑* or N→∑*N, N→∑* where N is the set of Non-Terminals and ∑ is the set of Terminal symbols.

The language generated by a Type-i grammar is called a Type-i language, i = 0,1,2,3. A Type-0 language is also called unrestricted language. A Type-1 language is also called a context-sensitive language (CSL). We have already observed that a Type-2 language is also called a Context-

Free Language (CFL) and a Type-3 language is also called a regular language. Each class of language in the Chomsky hierarchy is characterized as the language generated by a type of automata. These relationships have been summarized in the following table for convenience.

Grammars	Languages	Automata
Type-0 , phrase-struct , semi-true,Unrestricted grammars	Recursively enumerable language	Turing Machine
Type-1 , Context Senstive Grammar	Context-sensitive language	Linear-bounded automata
Type-2, Context-Free Grammars	Context-free language	Pushdown Automata
Type-3, Regular, right-linear,left-linear grammar	Regular Language	Finite Automata

Example

Find a grammar for the language

$L = \{a^n b^{n+1} | n \geq 1\}$. It is possible to find a grammar for L by modifying the previous grammar since we need to generate an extra b at the end of the string $a^n b^n$, $n \geq 1$. We can do this by adding a production $S \rightarrow Bb$ where the non-terminal B generates $a^i S b^i, i \geq 1$ as given in the previous example. Using the above concept we devise the following grammar for L.

$G = (N, \sum, P, S)$ where $N = \{ S, B \}$

$P = \{ S \rightarrow Bb, B \rightarrow ab, B \rightarrow aBb \}$

1.4 FINITE AUTOMATA

Automata (singular: automation) are a particularly simple, but useful, model of computation. They were initially proposed as a simple model for the behavior of neurons. The concept of a finite automaton appears to have arisen in the 1943 paper "A logical calculus of the ideas immanent in nervous activity", by Warren McCullock and Walter Pitts. In 1951 Kleene introduced regular expressions to describe the behavior of finite automata. He also proved the important theorem saying that regular expressions exactly capture the behaviors of finite automata. In 1959, Dana Scott and Michael Rabin introduced non-deterministic automata and showed the surprising theorem that they are equivalent to deterministic automata. We will study these fundamental results. Since those early years, the study of automata has continued to grow, showing that they are indeed a fundamental idea in computing.

States, Transitions And Finite-State Transition System

Let us first give some intuitive idea about a state of a system and state transitions before describing finite automata.

Informally, a state of a system is an instantaneous description of that system which gives all relevant information necessary to determine how the system can evolve from that point on. Transitions are changes of states that can occur spontaneously or in response to inputs to the states. Though transitions usually take time, we assume that state transitions are instantaneous (which is an abstraction). Some examples of state transition systems are: digital systems, vending machines, etc.

A system containing only a finite number of states and transitions among them is called a finite-state transition system. Finite-state transition systems can be modeled abstractly by a mathematical model called finite automation. We said that automata are a model of computation. That means that they are a simplified abstraction of `the real thing'. So what gets abstracted away?

One thing that disappears is any notion of hardware or software. We merely deal with states and transitions between states. The distinction between program and machine executing it disappears. One could say that an automaton is the machine and the program. This makes automata relatively easy to implement in either hardware or software. From the point of view of resource consumption, the essence of a finite

automaton is that it is a strictly finite model of computation. Everything in it is of a fixed, finite size and cannot be modified in the course of the computation.

1.4.1 Deterministic Finite (State) Automata (DFA)

Informally, a DFA (Deterministic Finite State Automaton) is a simple machine that reads an input string -- one symbol at a time -- and then, after the input has been completely read, decides whether to accept or reject the input. As the symbols are read from the tape, the automaton can change its state, to reflect how it reacts to what it has seen so far. Thus, a DFA conceptually consists of 3 parts:

1. A tape to hold the input string. The tape is divided into a finite number of cells. Each cell holds a symbol from $\sum$.
2. A tape head for reading symbols from the tape
3. A control, which itself consists of 3 things:
   - Finite number of states that the machine is allowed to be in (zero or more states are designated as accept or final states),
   - A current state, initially set to a start state,
   - A state transition function for changing the current state.

An automaton processes a string on the tape by repeating the following actions until the tape head has traversed the entire string:

1. The tape head reads the current tape cell and sends the symbol s found there the control. Then the tape head moves to the next cell.
2. The control takes s and the current state and consults the state transition function to get the next state, which becomes the new current state. Once the entire string has been processed, the state in which the automation enters is examined. If it is an accept state , the input string is accepted ; otherwise, the string is rejected. Summarizing all the above we can formulate the following formal definition:

Formal Definition of DFA

A Deterministic Finite State Automaton (DFA) is a 5-tuple $M = (Q,\sum,\delta,q_0,F)$

- Q is a finite set of states.
- $\sum$ is a finite set of input symbols or alphabet.
- $\delta : Q \times \sum \rightarrow Q$ is the "next state" transition function (which is total). Intuitively, δ is a function that tells which state to move to in response to an input, i.e., if M is in state q and sees input a, it moves to state $\delta(q,a)$
- $q_o \in Q$ is the start state.
- $F \subseteq Q$is the set of accept or final states.

Acceptance of Strings

A DFA accepts a string $w = a_1a_2\cdots a_n$ if there is a sequence of states $q_0, q_1, \cdots, q_n$in Q such that

1. q_0 is the start state.
2. $\delta(q_i,a_{i+1}) = q_{i+1}$ for all $0 < i < n$.
3. $q_n \epsilon F$.

Language Accepted or Recognized by a DFA

The language accepted or recognized by a DFA M is the set of all strings accepted by M , and is denoted by L(M)i.e.
$L(M) = \{ w \in \sum^* \mid M \text{ accepts a } \}$ The notion of acceptance can also be made more precise by extending the transition function δ.

Extended transition function

Extend $\delta : Q \times \sum \rightarrow Q$ (which is function on symbols) to a function on strings, i.e.
$\delta^* : Q \times \sum^* \rightarrow Q$. That is, $\delta^*(q,w)$ is the state the automation reaches when it starts from the state q and finish processing the string w.

Formally, we can give an inductive definition as follows: The language of the DFA M is the set of strings that can take the start state to one of the accepting states i.e.
$L(M) = \{ w \in \sum^* \mid M \text{ accepts } w \}$
$= \{ w \in \sum^* \mid \delta^*(q_0,w) \in F \}$

Example of DFA
$M = (Q, \sum, \delta, q_0, F)$
$Q = \{ q_0 , q_1 \}$
q0 is the start state
$F=\{q_1\}$
$\delta(q_0,0)= q_0 \quad \delta(q_1,0)= q_1$
$\delta(q_0,1)= q_1 \quad \delta(q_1,1)= q_1$

It is a formal description of a DFA. But it is hard to comprehend. For ex. The language of the DFA is L = {any string over { 0, 1} having at least one 1}. We can describe the same DFA by transition table or state transition diagram as following: It is easy to comprehend the transition diagram.

Transition table
It is basically a tabular representation of the transition function that takes two arguments (a state and a symbol) and returns a value (the "next state").

- Rows correspond to states,
- Columns correspond to input symbols,
- Entries correspond to next states
- The start state is marked with an arrow
- The accept states are marked with a star (*).

A transition table for the language L is shown below.

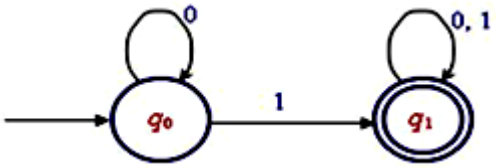
	0	1
→ q_0	q_0	q_1
* q_1	q_1	q_1

(State) Transition diagram:

A state transition diagram or simply a transition diagram is a directed graph which can be constructed as follows:

1. For each state in Q there is a node.
2. There is a directed edge from node q to node plabelled a if$\delta(q, a)=p$. (If there are several input symbols that cause a transition, the edge is labelled by the list of these symbols.)
3. There is an arrow with no source into thestart state.
4. Accepting states are indicated by double circle.

A (State) Transition diagram for the language L is shown below.

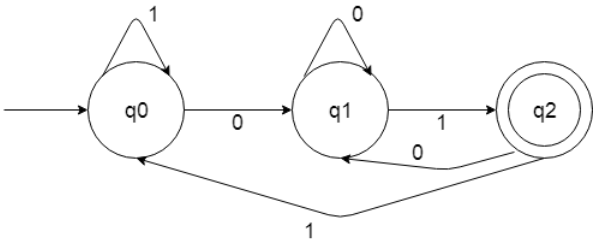


Explanation
We cannot reach final state until 1 appears in the i/p string. There can be any no. of 0's at the beginning. (The self-loop at q_0 on label 0 indicates it). Once the one 1 appeared in the input there can be any no. of 0's & 1's in any order at the end of the string.

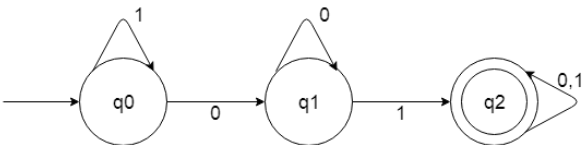
Informal description how a DFA operates:
An input to a DFA can be any string $w \in \sum^*$. Put a pointer to the start state q. Read the input string w from left to right, one symbol at a time, moving the pointer according to the transition function, δ. If the next symbol of w is a and the pointer is on state p, move the pointer to δ (p,a). When the end of the input string w is encountered, the pointer is on some state, r. The string is said to be **accepted** by the DFA if $r \in F$ and **rejected** if $r \notin F$.
A language $L \in \sum^*$ is said to be regular if L = L(M) for some DFA M. i.e. There must exist at least one DFA which can simulate the strings of language L.

Few More Examples on DFA

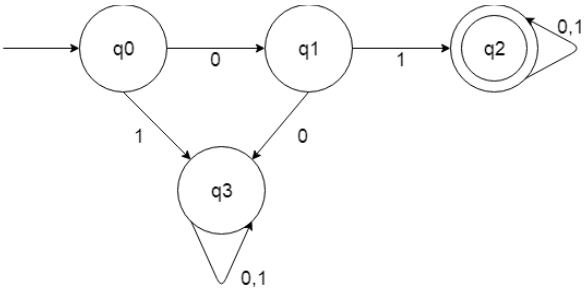
L1 = {Set of strings that ends with 01}



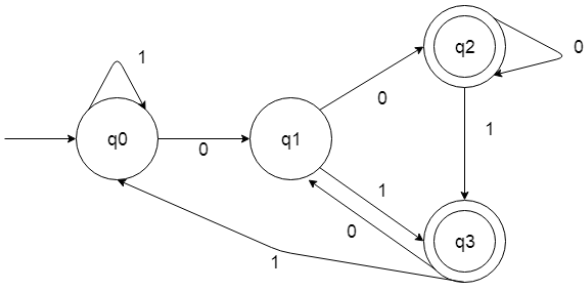
L2 = {Set of strings that contains 01}



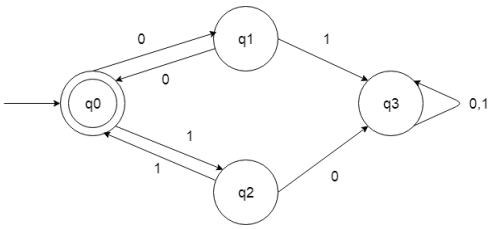
L3 = {Set of strings that starts with 01}



L4 = {Strings having second last symbol 0}



L5 = (00+11)*



1.4.2 Nondeterministic Finite Automata (NFA)

Non determinism is an important abstraction in computer science. Importance of non-determinism is found in the design of algorithms. For examples, there are many problems with efficient nondeterministic solutions but no known efficient deterministic solutions. (Travelling salesman, Hamiltonian cycle, clique, etc). Behavior of a process is in a distributed system is also a good example of nondeterministic situation. Because the behavior of a process might depend on some messages from other processes that might arrive at arbitrary times with arbitrary contents.

It is easy to construct and comprehend an NFA than DFA for a given regular language. The concept of NFA can also be used in proving many theorems and results. Hence, it plays an important role in this subject.

In the context of FA non determinism can be incorporated naturally. That is, an NFA is defined in the same way as the DFA but with the following two exceptions:

1. Multiple or No next state.
2. ∈ - Transitions.

∈ - Transitions

In an ∈-transition, the tape head doesn't do anything- it does not read and it does not move. However, the state of the automata can be changed - that is can go to zero, one or more states. This is written formally as $\delta(q,\in) = \{q_1, q_2, \cdots, q_k\}$ implying that the next state could by any one of $q_1, q_2, \cdots q_k$ w/o consuming the next input symbol.}

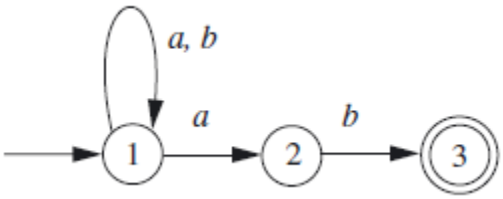
Acceptance

Informally, an NFA is said to accept its input w if it is possible to start in some start state and process w, moving according to the transition rules and making choices along the way whenever the next state is not uniquely defined, such that when w is completely processed (i.e. end of w is reached), the automata is in an accept state. There may be several possible paths

through the automation in response to an input w since the start state is not determined and there are choices along the way because of multiple next states. Some of these paths may lead to accept states while others may not. The automation is said to accept w if at least one computation path on input w starting from at least one start state leads to an accept state- otherwise, the automation rejects input w. Alternatively, we can say that, w is accepted if there exists a path with label w from some start state to some accept state. Since there is no mechanism for determining which state to start in or which of the possible next moves to take (including the w - transitions) in response to an input symbol we can think that the automation is having some "guessing" power to choose the correct one in case the input is accepted.

Example

Consider the language L = {w∈{a,b}*| Strings ending in ab}. The following NFA accepts L.



The m/c is not deterministic since there are two transitions from state 1 on input a, no transition (zero transition) from state 2 for symbol a and no transition from state 3 for symbols a & b. For any string w which ends with ab, there exists a sequence of legal transitions leading from the start state 1, to the accept state 3. But for any string w which does not end with ab, there is no possible sequence of legal transitions leading from 1 to 3. Hence m/c accepts L.

If you observe the NFA, you will find that for string aab it ends in state 1 as well as state 3 but one of the state from 1 and 3 is accepting state, hence the string is said to be accepted by NFA.

Note: In case of NFA, for string w there can be many paths from the starting state. If any of the path is ending in accepting state, the string is said to be accepted by NFA. If none of the path ends in accepting state the string is said to be rejected by NFA.

The Extended Transition function δ^*,

To describe acceptance by an NFA formally, it is necessary to extend the transition function, denoted as δ^*, takes a state $q \in Q$ and a string $w\sum^*$, and returns the set of states, $S \subseteq Q$, that the NFA is in after processing the string w if it starts in state q. Formally, $\hat{\delta}$is defined as follows:

1. $\delta^*(q,\in)=\{q\}$ that is, without rending any input symbol, an NFA does not change state.
2. Let y = xa some∞, $x\epsilon\sum^*$ and $a\epsilon\sum$. $\delta^*(q,y) = \delta^*(q,xa) = U\{\delta(r,a)$, where $r \epsilon \delta^*(q,x)\}$

.

Formal definition of NFA

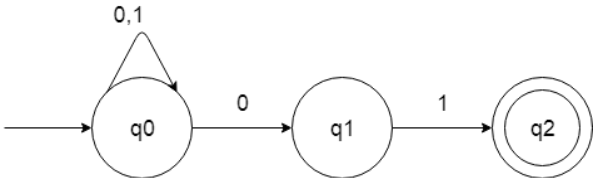
Formally, an NFA is a quintuple
M = (Q, ∑, δ , q_0, F) where Q, ∑, q_0, and F bear the same meaning as for a DFA, but δ, the transition function is redefined as follows:
δ : Q x (∑ ∪ {∈}) → P (Q) where P(Q) is the power set of Q i.e.2^Q.

The Language of an NFA:

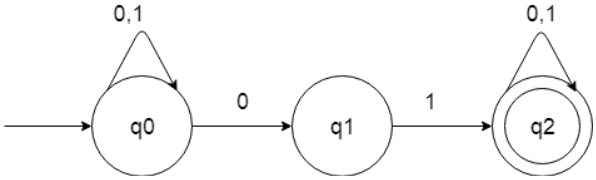
From the discussion of the acceptance by an NFA, we can give the formal definition of a language accepted by an NFA as follows :
If N = (Q, ∑, δ, q_0, F) is an NFA, then the language accepted by N is written as L(N) given by **$L(N) = \{w \mid \delta^*(q,w) \cap F \neq \Phi\}$.**
That is, L(N) is the set of all strings w in $\sum^*$ such that $\delta^*(q,w)$contains at least one accepting state.

Few More Examples on NFA

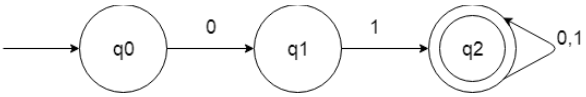
L1 = {Set of strings that ends with 01}



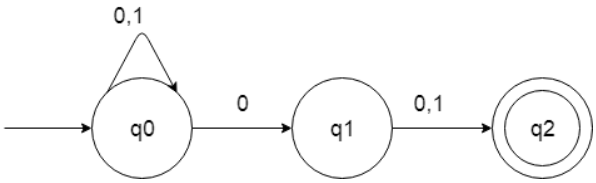
L2 = {Set of strings that contains 01}



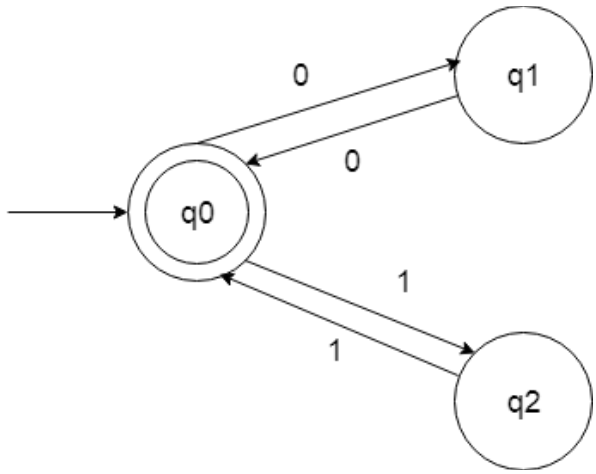
L3 = {Set of strings that starts with 01}



L4 = {Strings having second last symbol 0}



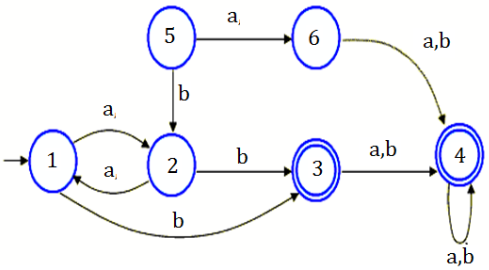
L5 = (00+11)*



1.5 MINIMIZATION OF DETERMINISTIC FINITE AUTOMATA (DFA)

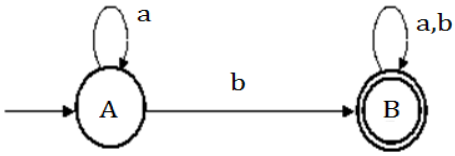
For any regular language L it may be possible to design different DFAs to accept L. Given two DFAs accepting the same language L, it is now natural to ask - which one is more simple? In this case, obviously, the one with less number of states would be simpler than the other. So, given a DFA accepting a language, we might wonder whether the DFA could further be simplified i.e. can we reduce the number of states accepting the same language?

Consider the following DFA M_1,



A minute observation will reveal that it accepts the language of the regular expression

a*b(a + b)* The same language is accepted by the following simpler DFA M_2 as well.



It is a fact that, for any regular language L there is a unique minimal state DFA (the uniqueness is up to isomorphism to be defined next).

For any given DFA M accepting L we can construct the minimal state DFA accepting L by using an algorithm which uses following generic steps.

- First, remove all the states (of the given DFA M) which are not accessible from the start state i.e. states P for which there is no string $x \in \Sigma^*$ s.t. $\hat{\delta}(q_0, x) = p$. Removing these states, clearly, will not change the language accepted by the DFA.
- Second, remove all the trap states, i.e. all states P from which there is no transition out of it.
- Finally, merge all states which are "equivalent" or "indistinguishable". We need to define formally what is meant by equivalent or indistinguishable states;

but at this point we assume that merging these states would not change the accepted language. Inaccessible states can easily be found out by using a simple research e.g. depth first search. Removing trap states are also simple. In the example, states 5 and 6 are inaccessible and hence can be removed; states 1 and 2 are equivalent and can be merged. Similarly states 3 & 4 are also equivalent and can be merged together to have the minimal DFA M_2as produced above.

To construct the minimal DFA we need to see how to find out indistinguishable or equivalent states for merging. we start with a definition and then proceed to find method to construct minimal state DFAs.

Note: For any regular language L there is a unique DFA that has a minimum number of states. In fact, the minimum DFA has as many states as the equivalence classes of L. (as defined in the context of Mayhill-Nerada Theorem).

Definition of Indistinguishability:

States p and q are indistinguishable if $\forall x \in \sum^* \hat{\delta}(p,x) \in F$ if $f\,\hat{\delta}(q,x) \in F$, and is denoted as $p \equiv q$. It is easy to see that indicting usability is an equivalence relation.

In other words we say that sates p and q are "distinguishable" if $\exists x \in \sum^*$ s.t $\hat{\delta}(p,x) \in F$ and $\hat{\delta}(q,x) \notin F$ is denoted as $p \neq q$.

We say that, states p and q of a DFA M accepting a language L can be merged safely (i.e. without changing the accepted language L) if $p \equiv q$ i.e. if p and q are indistinguishable. We can prove this by showing that when p and q are merged. Then they correspond to the same state in M_L Formally, $p \equiv q$ iff

$\forall x, y \in \sum^*, \hat{\delta}(q_0, x) = p$ and x

$f\,\hat{\delta}(q_0, y) = q \Rightarrow x \equiv_L y$.

A Minimization Algorithm

We now produce an algorithm to construct the minimal state DFA from any given DFA accepting L by merging states inductively.

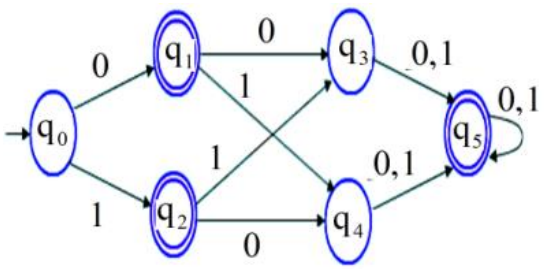
The algorithm assume that all states are reachable from the start state i.e. there is no inaccessible states. The algorithm keeps on marking pairs of states (p, q) as soon as it determines that p and q are distinguishable i.e. $p \not\equiv q$. The pairs are, of course, unordered i.e. pairs (p, q) and (q ,p) are considered to be identical. The steps of the algorithm are given below.

1. For every $p, q \in Q$, initially unmark all pairs (p, q).
2. If $p \in F$ and $q \notin F$ (or vice versa) then mark (p, q).
3. Repeat the following step until no more changes occur**:** If there exists an unmarked pair (p, q) such that $(\delta(p,a), \delta(q,a))$ is marked for some $a \in \sum$, then mark (p, q).
4. $p \equiv q$ iff (p, q) is unmarked.

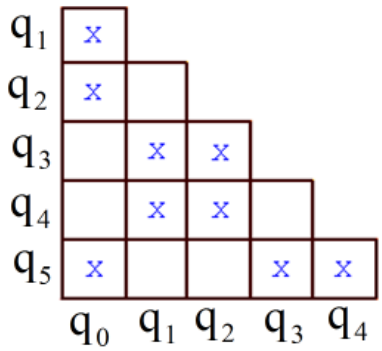
The algorithm correctly computes all pairs of states that can be distinguished i.e. unmarked. It is easy to show (by induction) that thepair (p, q) is marked by the above algorithm if $\exists x \in \sum^*$ s.t. $\hat{\delta}(p,x) \in F$ and $\hat{\delta}(q,x) \notin F$ (or vice versa) i.e. if $p \not\equiv q$

Example:

Consider the following DFA which we need to minimise:



Initially, all cells are unmarked. (i.e. at step 1 of the algorithm). After step 2, all cells representing pairs of states of which one is accepting and the other is non-accepting are marked by putting an X. The table below shows the status after this step.

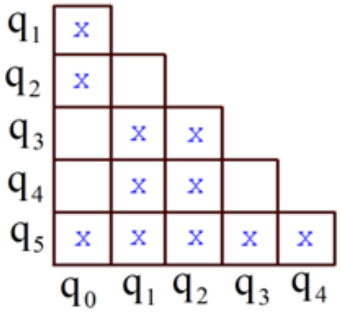


In step 3, we consider all unmarked pairs one by one. Considering the unmarked pair (q_0, q_3), we find that q_0 & q_3 go to q_1 and q_5, respectively, on input 0. we use the notation $(q_0, q_3) \xrightarrow{0} (q_1, q_5)$ to indicate this. Since the pair (q_1, q_5) is not marked, (q_0, q_3) cannot be marked at this point. Again, we see that, $(q_0, q_3) \xrightarrow{1} (q_2, q_5)$ and (q_2, q_5) is unmarked.

Hence, we cannot mark (q_0, q_3) and since we have considered all input symbols (0 & 1) we need to examine other unmarked pairs. The observations and actions are shown below.

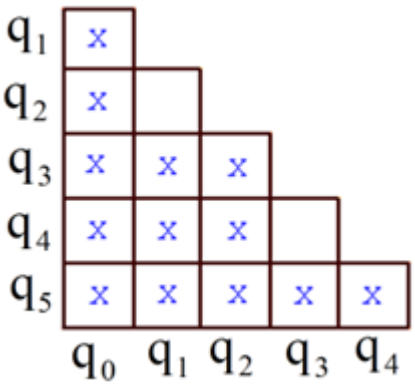
- $(q_0, q_4) \xrightarrow{0} (q_1, q_5)$
- $(q_0, q_4) \xrightarrow{1} (q_2, q_5)$, cannot mark (q_0, q_4) since (q_1, q_5) & (q_2, q_5) are unmarked.
- $(q_1, q_2) \xrightarrow{0} (q_3, q_4)$
- $(q_1, q_2) \xrightarrow{1} (q_3, q_4)$, cannot mark
- (q_1, q_2) since (q_3, q_4) is unmarked.
- $(q_1, q_5) \xrightarrow{0} (q_3, q_5)$, (q_1, q_5) is marked since (q_3, q_5) is already marked.
- $(q_2, q_5) \xrightarrow{0} (q_4, q_5)$, (q_2, q_5) is marked since (q_3, q_5) is already marked.
- $(q_3, q_4) \xrightarrow{0} (q_5, q_5)$, (q_5, q_5) is never marked since it is not in the table & hence (q_3, q_4) is not marked.
- $(q_3, q_4) \xrightarrow{1} (q_5, q_5)$

The resulting table after this pass is given below.



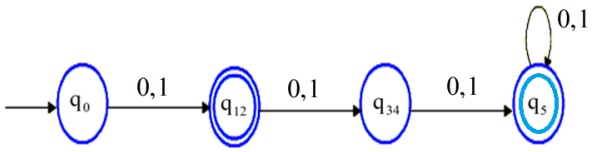
In the next pass we find that $(q_0, q_3) \xrightarrow{0} (q_1, q_5)$ and (q_1, q_5) is marked in the previous pass .Hence, (q_0, q_3) can be marked now.

Similarly, $(q_0, q_4) \xrightarrow{1} (q_2, q_5)$ and hence (q_0, q_4) can be marked since (q_2, q_5) has been marked in the previous pass. Other pairs cannot be marked and the resulting table is shown below. By executing step 3 again we observe that no more pairs can be marked and hence the algorithm stops with this table as the final result.



The unmarked pairs left in the table after execution of the algorithm are (q_1, q_2) and (q_3, q_4) implying $q_1 \equiv q_2$ and $q_3 \equiv q_4$. Now, we merge q_1 & q_2 and q_3 & q_4 to have new states q_{12} & q_{34}, respectively.

Transitions are adjusted appropriately to obtain the following minimal DFA.



q_{12} is a final state, since both q_1 & q_2 were final states. Similarly q_{34} is a non-final state. q_0 Goes to q_{12} on input 0 and 1, since q_0 go to q_1 and q_2 respectively on 0 and .Similar, justifications suffice for other adjusted transitions.

1.6 REGULAR EXPRESSION: FORMAL DEFINITION

We construct REs from primitive constituents (basic elements) by repeatedly applying certain recursive rules as given below. (In the definition)

Definition

Let Σ be an alphabet. The regular expressions are defined recursively as follows.

BASIS

i) ϕ is a RE

ii) $\in$ is a RE

iii) $\forall a \in \sum$, a is RE.

These are called primitive regular expression i.e. Primitive Constituents

RECURSIVE STEP

If r_1 and r_2 are REs over Σ^*, then so are

i) $r_1 + r_2$ **ii)** $r_1 r_2$

iii) r_1^* **iv)** (r_1)

CLOSURE

r is RE over Σ^* only if it can be obtained from the basic elements (Primitive REs) by a finite no of applications of the recursive step (given in 2.1.2).

Example

Let Σ = {0,1,2}. Then (0+21)*(1+ ∅) is a RE, because we can construct this expression by applying the above rules as given in the following step.

Steps	RE Constructed	Rule Used
1	1	Rule 1(iii)
2	∅	Rule 1(i)
3	1+∅	Rule 2(i) & Results of Step 1, 2
4	(1+∅)	Rule 2(iv) & Step 3
5	2	1(iii)
6	1	1(iii)
7	21	2(ii), 5, 6
8	0	1(iii)
9	0+21	2(i), 7, 8
10	(0+21)	2(iv), 9
11	(0+21)*	2(iii), 10
12	(0+21)*	2(ii), 4, 11

1.7 LANGUAGE DESCRIBED BY RES

Each describes a language (or a language is associated with every RE). We will see later that REs are used to attribute regular languages.

Notation

If r is a RE over some alphabet then L(r) is the language associated with r. We can define the language L(r) associated with (or described by) a REs as follows.

1. ∅ is the RE describing the empty language i.e. L(∅) =∅.
2. ∈ is a RE describing the language {∈} i.e. L(∈) = {∈} .
3. $\forall a \in \sum$, a is a RE denoting the language {a} i.e . L(a) = {a} .
4. If r_1 and r_2 are REs denoting language $L(r_1)$ and $L(r_2)$ respectively, then
   - **i)** $r_1 + r_2$ is a regular expression denoting the language $L(r_1 + r_2) = L(r_1) + L(r_2)$
   - **ii)** $r_1 r_2$ is a regular expression denoting the language $L(r_1 r_2) = L(r_1)\, L(r_2)$
   - **iii)** r_1^* is a regular expression denoting the language $L(r_1^*) = (L(r_1))^*$
   - **iv)** (r_1) is a regular expression denoting the language $L((r_1)) = L(r_1)$

Example

Consider the RE (0*(0+1)). Thus the language denoted by the RE is

L(0*(0+1)) = L(0*) L(0+1) ……..by 4(ii)

= L(0)*{L(0) + L(1)}

= { ∈, 0,00,000,.....} {0} ∪{1}

= { ∈, 0,00,000,.....} {0,1}

={0,00,000,0000,.....,1,01,001, 0001,.....}

Precedence Rule

Consider the RE ab + c. The language described by the RE can be thought of either L(a)L(b+c) or L(ab)∪ L(c) as provided by the rules (of languages described by REs) given already. But these two represents two different languages lending to ambiguity. To remove this ambiguity we can either

1) Use fully parenthesized expression-(cumbersome) or

2) Use a set of precedence rules to evaluate the options of REs in some order. Like other algebras mod in mathematics .For REs, the order of precedence for the operators is as follows:

i) The star operator precedes concatenation and concatenation precedes union (+) operator.

ii) It is also important to note that concatenation & union (+) operators are associative and union operation is commutative. Using these precedence rule, we find that the RE ab+c represents the language L(ab)∪L(c) i.e. it should be grouped as ((ab)+c).

We can, of course change the order of precedence by using parentheses. For example, the language represented by the RE a(b+c) is L(a)L(b+c).

Example

The RE ab*+b is grouped as ((a(b*))+b) which describes the language L(a)(L(b))*∪ L(b)

Example

The RE (ab)*+b represents the language (L(a)L(b))* ∪L(b).

Example

It is easy to see that the RE (0+1)*(0+11) represents the language of all strings over {0,1} which are either ended with 0 or 11.

Example

The regular expression r =(00)*(11)*1 denotes the set of all strings with an even number of 0's followed by an odd number of 1's i.e. $L(r) = \{0^{2n}1^{2m+1} \mid n \geq 0, m \geq 0\}$

Note:

The notation r^+ is used to represent the RE rr*. Similarly, r^2 represents the RE rr, r^3 denotes rr^2, and so on. An arbitrary string over ∑= {0,1} is denoted as (0+1)*.

Exercise:

Give a RE r over {0,1} s.t. L(r)={ x∈{0,1}*|x has at least one pair of consecutive 1's}

Solution

Every string in L(r) must contain 11 somewhere, but what comes before and what goes before is completely arbitrary. Considering these observations we can write the REs as (0+1)*11(0+1)*.

Example

Considering the above example it becomes clear that the RE (0 + 1) * 11 (0 + 1) * + (0 + 1)*00(0+1)* represents the set of string over {0,1} that contains the substring 11 or 00.

Example

Consider the RE 0*10*10*. It is not difficult to see that this RE describes the set of strings over {0,1} that contains exactly two 1's. The presence of two 1's in the RE and any no of 0's before, between and after the 1's ensure it.

Example

Consider the language of strings over {0,1} containing two or more 1's.

Solution
There must be at least two 1's in the RE somewhere and what comes before, between, and after is completely arbitrary. Hence we can write the RE as (0+1)*1(0+1)*1(0+1)*. But following two REs also represent the same language, each ensuring presence of least two 1's somewhere in the string
i) 0*10*1(0+1)*
ii) (0+1)*10*10*

Example
Consider a RE r over {0,1} such that L(r) = {x∈{0,1}*|x has no pair of consecutive 1's}

Solution
Though it looks similar to ex ……., it is harder to construct to construct. We observer that, whenever a 1 occurs, it must be immediately followed by a 0. This substring may be preceded & followed by any no of 0's. So the final RE must be a repetition of strings of the form: 00…0100….00 i.e. 0*100*. So it looks like the RE is (0*100*)*. But in this case the strings ending in 1 or consisting of all 0's are not accounted for. Taking these observations into consideration, the final RE is r = (0*100*)(1+∈)+0*(1+∈).

Alternative Solution:
The language can be viewed as repetitions of the strings 0 and 01. Hence get the RE as r = (0+10)*(1+∈). This is a shorter expression but represents the same language.

Few more examples of Regular Expressions: Consider ∑ = {0,1}
L = {Strings ending in 01}
RE = (0+1)*01

L = {Strings starting with 01}
RE = 01(0+1)*

L = {Strings starting or ending with 00}
RE = 00(0+1)* + (0+1)*00

L = {Strings starting and ending with 00}
RE = 00(0+1)*00 + 00 + 000

L = {Strings starting with 00 or 11}
RE = 00(0+1)* + 11(0+1)*
= (00+11) (0+1)*

L = {Strings starting with 00 and 11}
RE = ∅

L = {Strings containing 00 or 101}
RE = (0+1)*00(0+1)* + (0+1)*101(0+1)*
= (0+1)*(00+101)(0+1)*

L = {Strings containing 00 and 11}
RE = (0+1)*00(0+1)*11(0+1)* + (0+1)*11(0+1)*00(0+1)*
= (0+1)*[00(0+1)*11+11(0+1)*00] (0+1)*

L = {Strings having second last symbol 0}
RE = (0+1)*0(0+1)

L = {Strings having even length}
RE = ((0+1) (0+1))* or (00+11+01+10)*
L = {Strings having odd length}
RE = ((0+1)(0+1))*(0+1) or
(00+11+01+10)* (0+1)

If we take ∑ = {0,1,2} then:
L = {Strings ending in 01}
RE = (0+1+2)*01

L = {Strings starting with 01}
RE = 01(0+1+2)*

L = {Strings having second last symbol 0}
RE = (0+1+2)*0(0+1+2)

L = {Strings having even length}
RE = ((0+1+2) (0+1+2))* or
(00+01+02+10+11+12+20+21+22)*

L = {Strings having odd length}
RE = ((0+1+2)(0+1+2))*(0+1+2) or
(00+01+02+10+11+12+20+21+22)*
(0+1+2)

Properties of Regular Expression:
Let r, r1, r2, p, and q are the regular expressions:

- $\emptyset^* = \in$
- $\in^* = \in$
- $r \in = \in r = r$
- $r \emptyset = \emptyset r = \emptyset$
- $\emptyset + r = r$
- $r1 + r2 = r2 + r1$
- $r1\ r2 \neq r2\ r1$
- $r + r = r$
- $r^* + r^* = r^*$
- $r^* r^* = r^*$
- $r\, r^* = r^* r = r^+$
- $r^+ + \in = r^*$
- $p(qr) = (pq)r$
- $p+(q+r) = (p+q) + r$
- $p\,(q+r) = pq + pr$
- $(q+r)\,p = qp + rp$
- $(pq)^*\, p = p\,(qp)^*$
- $(p+q)^* = (p^* + q^*)^* = (p^*q^*)^* = (q^*p^*)^* = (p^* + q^*)^+ = (p^+ + q^+)^* = (p^* + q^+)^* = (p^+ + q^*)^* = (p^+ + q^*)^+ = (p^* + q^+)^+$

1.8 REGULAR EXPRESSION & REGULAR LANGUAGE:

Equivalence (of REs) with FA :

Recall that, language that is accepted by some FAs are known as Regular language. The two concepts : REs and Regular language are essentially same i.e. (for) every regular language can be developed by (there is) a RE, and for every RE there is a Regular Language. This fact is rather surprising, because RE approach to describing language is fundamentally different from the FA approach. But REs and FA are equivalent in their descriptive power. We can put this fact in the focus of the following Theorem.

Theorem

A language is regular if some RE describes it. This Theorem has two directions, and are stated & proved below as a separate lemma

RE to FA

REs denote regular languages:

Lemma

If L(r) is a language described by the RE r, then it is regular i.e. there is a FA such that $L(M) \cong L(r)$.

Proof

To prove the lemma, we apply structured index on the expression r. First, we show how to construct FA for the basic elements:∅, ∈ and for any $\alpha \in \sum$. Then we show how to combine these Finite Automata into Complex Automata that accept the Union, Concatenation, Kleen Closure of the languages accepted by the original smaller automata.

Use of NFAs is helpful in the case i.e. we construct NFAs for every REs which are represented by transition diagram only.

Basis

- **Case (i) :**

  r=∅. Then L(r)=∅. Then L(r)=∅ and the following NFA N recognizes L(r). Formally $N = (Q, \{q\}, \sum, \delta, F, \phi)$ where Q = {q} and $\delta(q, a) = \phi \forall \alpha \in S, F = \phi$.

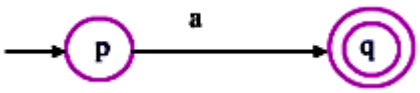
- **Case(ii) :**

  $r = \in$. $L(r) = \{\in\}$, and the following NFAN accepts L(r). Formally $N = (\{q\}, \sum, \delta, q, \{q\})$ where $\delta(q, \alpha) = \phi \forall \alpha \in \sum$.

  Since the start state is also the accept step, and there is no any transition defined, it will accept the only string ∈ and nothing else.

- **Case (iii) :**

  r = a for some $\alpha \in \sum$. Then L(r) = {a}, and the following NFA N accepts L(r).



  Formally, $N = (\{p, q\}, \sum, \delta, p, \{q\})$ where $\delta(p, q) = \{q\}, \delta(s, b) = \{\phi\}$ for s≠p or b≠a.

Induction

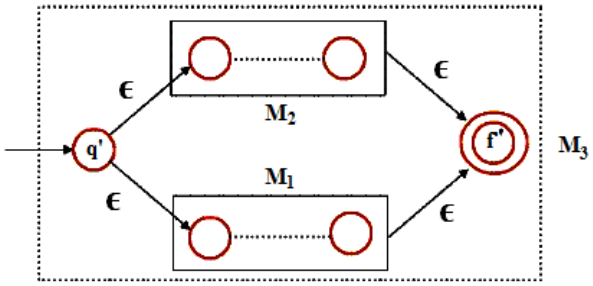
Assume that the start of the theorem is true for REs r_1 and r_2. Hence we can assume that we have automata M_1 and M_2 that accepts languages denoted by REs $r_1$and r_2, respectively i.e.

$L(M_1) = L(r_1)$ and $L(M_2) = L(r_2)$.

The FAs are represented schematically as shown below. Each has an initial state and a final state. There are four cases to consider.

Case (i) : Union

- Consider the RE $r_3 = r_1 + r_2$ denoting the language $L(r_1) \cup L(r_2)$.



- We construct FA M_3, from M_1 and M_2 to accept the language denoted by RE r_3 as follows :
- Create a new (initial) start state q' and give ∈ - transition to the initial state of M_1 and M_2.This is the initial state of M_3.
- Create a final state f' and give ∈-transition from the two final state of $M_1$and M_2. F'is the only final state of M_3 and final state of M_1 and M_2 will be ordinary states in M_3.
- All the state of M_1 and M_2 are also state of M_3.
- All the moves of M_1 and M_2 are also moves of M_3. [Formal Construction]

  It is easy to prove that $L(M_3) = L(r_3)$

  Proof: To show that $L(M_3) = L(r_3)$ we must show that

  $= L(r_1) \cup L(r_2)$

  $= L(M_1) = L(M_2)$ by following transition of M_3.Starts at initial state q' and enters the start state of either M_1 or M_2 following the transition i.e. without consuming any input. WLOG, assume that, it enters the start state of M_1. From this point onward it has to follow only the transition of M_1 to enter the final state of M_1, because this is the only way to enter the final state of M by following the e-transition.(Which is the last transition & no input is taken at the transition). Hence the whole input w is considered while traversing from the start state of M_1 to the final state of M_1. Therefore M_1 must accept w_3.

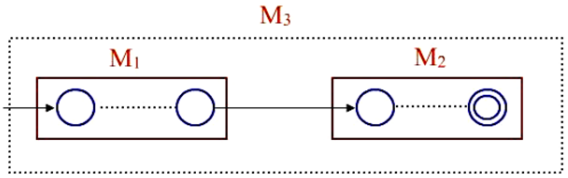
  Say $w \in L(M_1)$or $w \in L(M_2)$.

  WLOG, say $w \in L(M_1)$

  Therefore when M_1 process the string w, it starts at the initial state and enters the final state when w consumed totally, by following its transition. Then M_3 also accepts w, by starting at state q' and taking ∈ -transition enters the start state of M_1 -follows the moves of M_1 to enter the final state of M_1 consuming input w thus takes ∈-transition tof' . Hence proved.

Case(ii): Concatenation

- Consider the RE $r_3 = r_1 r_2$ denoting the language $L(r_1)L(r_2)$.

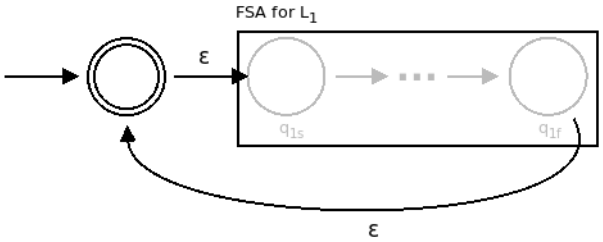


We construct FA M_3 from M_1 & M_2 to accept $L(r_3)$ as follows :

- Draw ∈ - transitions from the accepting states of M1 to the starting states of M2.
- Convert the accepting states of M1 to non-accepting states.
- Accepting states of M3 will be the accepting states of M2 only.
- Starting state of M3 will be starting state of M1 only.

Case(iii): Kleene Star

- Consider the RE r3 =(r1)* denoting the language L(r3) = L(r1)*



We can construct FA for L(r) as follows:

- Create a new state. Make it starting state. Make it accepting state. Let us call this state as q.
- Draw ∈ - transition from q to the starting state of M1.
- Draw ∈ - transition from the accepting state of M1 to q.
- Convert the accepting state of M1 to Non-accepting states.

$$L(M_3) = (L(M_1))^* = (L(r)_1)^* = r_1^*.$$

Case (iv): Parenthesis

Let $r_3 = (r_1)$. Then the FA M_1 is also the FA for (r_1), since the use of parentheses does not change the language denoted by the expression.

1.9 REGULAR GRAMMARS

A grammar is said to be regular grammar if it is either left linear or right linear.

Right linear grammar:

A grammar G=(V, ∑,S,P)is right-linear if each production has one of the following forms:

V -> T*V or V->T*

where V is the set of variables and T ∈ ∑.

Example:

G1 = < V, { a, b }, S, P > , where V = {S} ,

P = { S -> aS, S -> bS, S -> Λ } is a right linear grammar.

Hence G1 is regular grammar.

Left linear grammar:

A grammar G is left-linear if each production has one of the following forms.

V -> VT* or V->T*

where V is the set of variables and T ∈ ∑.

Example:

G2 = < V, { a, b }, S, P > , where V = {S} ,

P = { S -> Sa, S -> Sb, S -> Λ } is a left linear grammar.

Hence G2 is regular grammar.

Example:

G3 = < V, { a, b }, S, P > , where V = {S} ,

P = { S -> aSb, S -> Λ } is neither right linear nor left linear.

Hence G3 is not regular grammar.

1.10 REGULAR LANGUAGE

Definition:

A language L is said to be regular if :

- It has a regular grammar. Or
- Its regular expression exist. Or
- Its DFA exist. Or
- Its NFA exist.

So Regular grammar, Regular expression, DFA and NFA are all equivalent in the sense that any language which can be defined by any of these can also be defined by others.

For the example in the previous topic we can say that the language((a+b)*) represented by G1 and G2 is regular but the language ($(a^n b^n | n>=0)$) represented by G3 is not.

All languages we have seen in topic 2.2 are regular.

Closure properties of Regular Languages:

Regular languages are closed under:

Union, Intersection, Concatenation, Kleene Closure, Difference, Complement, Reversal, Homomorphism, Inverse Homomorphism

i) Union:

If L and M are regular languages, so is L ∪ M.

- Let L and M be the languages of regular expressions R and S, respectively.
- Then R+S is a regular expression whose language is L ∪ M.

Example:

L = (a+b)* and M = a*

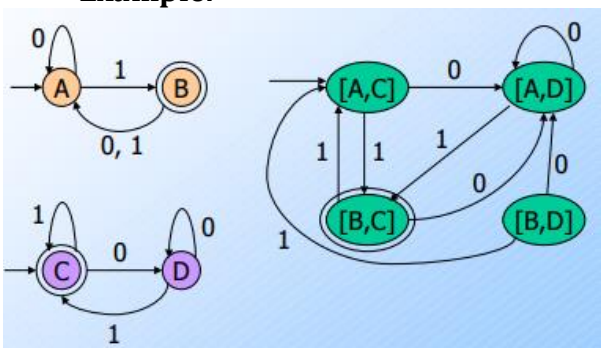
L U M = (a+b)* + a* = (a+b)*

ii) Intersection:

If L and M are regular languages, then so is L ∩ M.

- Let D1 and D2 be DFA's whose languages are L and M, respectively.
- Construct D3, the product automaton of D1 and D2.
- Make the final states of D3 be the pairs consisting of final states of both D1 and D2.

Example:



iii) Concatenation:

If L and M are regular languages, so is L M.

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

Example:

L = (a+b)* and M = a*

L M = (a+b)*a*

iv) Kleene Star:

If L is a regular language then L* will also be a regular language.

- Let the regular expression for the language L is r then the regular expression of language L* will be r*.

Example:

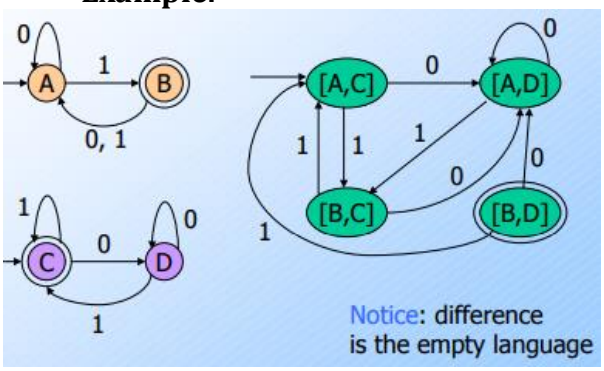
Let L = (a+b) then L* = (a+b)*

v) Difference:

If L and M are regular languages, then so is L – M = strings in L but not M.

- Let D1 and D2 be DFA's whose languages are L and M, respectively.
- Construct D3, the product automaton of A and B.
- Make the final states of C be the pairs where A-state is final but B-state is not.

Example:



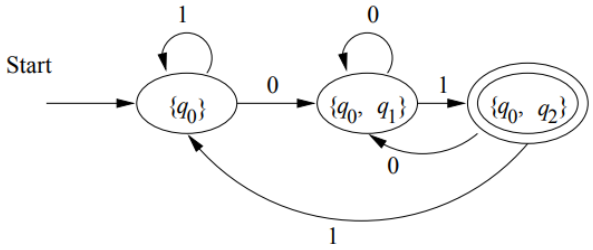
vi) Complement:

Let L is the regular language then L' will also be a regular language.

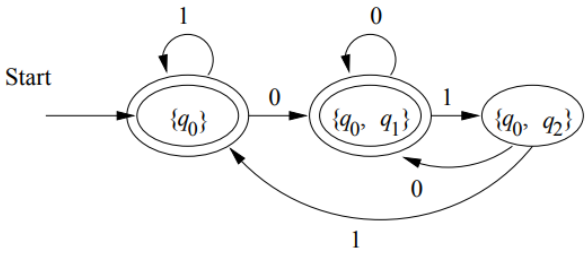
- Let D is the DFA for language L.

- Convert accepting states of D to non-accepting states and non-accepting states to accepting states to get the DFA for language L'.

Example:
Consider the DFA D for language L:



The DFA D' for language L' will be:



vii) Reversal:
If L is regular then L^R will also be regular language.

- Let D be the NFA for language L.
- Convert starting state of D to accepting state and accepting state of D to starting state.
- If D has more than one accepting state then create new state, make it starting state and draw null transitions to the old accepting states.
- Reverse the direction of transtitions.

Example:
Let L = 01* + 10*then L^R = 1*0 + 0*1

viii) Homomorphism:
If L is a regular language, and h is a homomorphism on its alphabet, then h(L) = {h(w) | w is in L} is also a regular language.

- A homomorphism on an alphabet is a function that gives a string for each symbol in that alphabet.
- Let E be a regular expression for L.
- Apply h to each symbol in E.
- Language of resulting RE is h(L)

Example:
Let h(0) = ab; h(1) = ε and L = 01* + 10* then h(L) will be abε* + ε(ab)* = (ab)*

ix) Inverse Homomorphism:
Let h be a homomorphism and L a language whose alphabet is the output language of h. $h^{-1}(L)$ = {w | h(w) is in L}.

Example:
Let h(0) = ab; h(1) = ε. Let L = {abab, baba}.
$h^{-1}(L)$ = the language with two 0's and any number of 1's = L(1*01*01*)

Here, no string maps to baba; any string with exactly two 0's maps to abab.

1.11 NON-REGULAR LANGUAGES:

We have learned regular languages, their properties and their usefulness for describing various systems. There are, however, languages that are not regular and therefore require devices other than finite automata to recognize them.

In this section we are going to study some of the methods for testing given languages for regularity and see some of

the languages that are not regular. The main idea behind these test methods is that finite automata have only finite amount of memory in the form of states and that they can not distinguish infinitely many strings.

For example to recognize the language $\{a^nb^n \mid n \text{ is a natural number}\}$, a finite automaton must remember how many a's it has read when it starts reading b's. Thus it must be in different states when it has read different number of a's and starts reading the first b. But any finite automaton has only finite number of states.

Thus there is no way for a finite automaton to remember how many a's it has read for all possible strings a^nb^n. That is the main limitation of finite automata. Since a regular language must be recognized by a finite automaton, we can conclude that $\{a^nb^n \mid n \text{ is a natural number}\}$ is not regular.

This is the basis of two of the regularity test methods we are going to study below: Myhill-Nerode Theorem and Pumping Lemma.

Myhill-Nerode's theorem

Indistinguishability of strings : Strings x and y in $\sum^*$ are **indistinguishable with respect to a language L** if and only if for every string z in $\sum^*$, either xz and yz are both in L or they are both not in L.

For example, a and aa are indistinguishable with respect to the language a^n over alphabet { a }, where n is a positive integer, because aa^k and aaa^k are in the language a^n for any positive integer k. However, with respect to the language a^nb^n, a and aa are not indistinguishable (hence distinguishable), because ab is in the language a^nb^n while aab is not in the language.

Using this concept of indistinguishability, the following theorem by Myhill and Nerod gives a criterion for (non)regularity of a language.

Theorem : A language L over alphabet ∑ is nonregular if and only if there is an infinite subset of $\sum^*$, whose strings are pairwise distinguishable with respect to L.

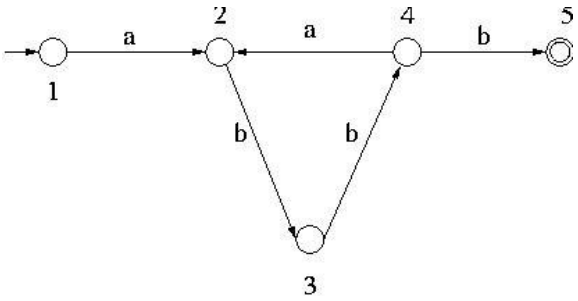
Example 1: $L_1 = \{ a^nb^n \mid n \text{ is a positive integer } \}$ over alphabet { a , b } can be shown to be nonregular using Myhill-Nerode as follows:

Consider the set of strings $S_1 = \{ a^n \mid n \text{ is a positive integer } \}$. S_1 is over alphabet { a , b } and it is infinite. We are going to show that its strings are pairwise distinguishable with respect to L_1. Let a^k and a^m be arbitrary two different members of the set S_1, where k and m are positive integers and $k \neq m$. Select b^m as a string to be appended to a^k and a^m. Then a^kb^m is not in L_1 while a^mb^m is in L_1. Hence a^k and a^m are distinguishable with respect to L_1. Since a^k and a^m are arbitrary strings of S_1, S_1 satisfies the conditions of Myhill-Nerode theorem. Hence L_1 is nonregular.

Example 2: $L_2 = \{ ww \mid w \in \{a, b\}^* \}$ is nonregular.
Consider the set of strings S_2 which is the same as S_1 of Example 1 above. It can be shown to be pairwise distinguishable with respect to L_2 as follows. Let a^k and a^m be arbitrary two different members of the set, where k and m are positive integers and $k \neq m$. Select ba^kb as a string to be appended to a^k and a^m. Then a^kba^kb is in L_2 while a^mba^kb is not in L_2. Hence a^k and a^m are distinguishable with respect to L_2. Since a^k and a^m are arbitrary strings of S_2, S_2 satisfies the

conditions of Myhill-Nerode theorem. Hence L_2 is nonregular.

Pumping Lemma:
Let us consider the NFA given below.



NFA accepting a(bba)*bbb

This NFA accepts among others some strings of length greater than 5 such as abbabbb, abbabbabbb etc. Those strings which are accepted by this NFA and whose length is greater than 5 have a substring which can be repeated any number of times without being rejected by the NFA.

For example the string abbabbb is accepted by the NFA and if one of its substrings bba is repeated any number of times in abbabbb, the resultant strings such as abbb (bba repeated 0 times), abbabbabbb, abbabbabbabbb etc. are also accepted by the NFA.
In general if a string w (such as abbabbb in the example above) is accepted by an NFA with *n* states and if its length is longer than *n*, then there must be a cycle in the NFA along some path from the initial state to some accepting state (such as the cycle 2-3-4-2 in the above example).

Then the substring representing that cycle (bba in the example) can be repeated any number of times within the string w without being rejected by the NFA.

The following theorem which is called Pumping Lemma is based on this observation. It states that if a language is regular, then any long enough string of the language has a substring which can be repeated any number of times with the resultant strings still in the language. It is stated without a proof here.

Pumping Lemma :
Suppose that a language L is regular. Then there is an FA that accepts L. Let n be the number of states of that FA. Then for any string x in L with $|x| \geq n$, there are strings u, v and w which satisfy the following relationships:

$x = uvw$
$|uv| \leq n$
$|v| > 0$ and
for every integer $m \geq 0$,
$uv^m w \in L$.

Note that Pumping Lemma gives a necessity for regular languages and that it is not a sufficiency, that is, even if there is an integer n that satisfies the conditions of Pumping Lemma, the language is not necessarily regular. Thus Pumping Lemma can not be used to prove the regularity of a language. It can only show that a language is nonregular.

Example 4: As an example to illustrate how Pumping Lemma might be used to prove that a language is nonregular, let us prove that the language $L = a^k b^k$ is nonregular, where k is a natural number.

Suppose that L is regular and let n be the number of states of an FA that accepts L. Consider a string $x = a^n b^n$ for that n. Then there must be strings u, v, and w such that

$x = uvw$,
$|uv| \leq n$
$|v| > 0$, and
for every $m \geq 0$, $uv^m w \in L$.

Since $|v| > 0$, v has at least one symbol. Also since $|uv| \leq n$, $v = a^p$, for some $p > 0$, Let us now consider the string uv^mw for $m = 2$. Then $uv^2w = a^{n-p}a^{2p}b^n = a^{n+p}b^n$. Since $p > 0$, $n + p \neq n$. Hence $a^{n+p}b^n$ can not be in the language L represented by a^kb^k . This violates the condition that for every $m \geq 0$, $uv^mw \in L$. Hence L is not a regular language.

Few examples of Non-Regular Languages:

$L = \{ w \in \{a,b\}^* : w = a^ib^i, \text{ for some } i \geq 0 \}$

$L = \{ w \in \{a,b\}^* : w = a^mb^n, \text{ for } m > n \}$

$L = \{ w \in \{a,b\}^* : w = a^mb^n, \text{ for } m < n \}$

$L = \{ w \in \{a,b\}^* : \#_a(w) = \#_b(w) \}$

$L = \{ ww^R : w \in \{a,b\}^* \}$

L = {Even length palindrome strings}

L = {Odd length palindrome strings}

$L = \{ ww : w \in \{a,b\}^* \}$

$L = \{ w \in \{a,b\}^* : w = a^mb^nc^p, m=n+p \}$

$L = \{ w \in \{a,b\}^* : w = a^mb^nc^p, m=n \}$

$L = \{ w \in \{a,b\}^* : w = a^mb^nc^p, m=4n \text{ or } n=5p \}$

1.12 POWER OF DFA AND NFA IN TERMS OF LANGUAGE ACCEPTANCE

For any language if DFA can be built then NFA can be and vice versa.

For any language if DFA can't be built then NFA can't be and vice versa.

Hence, DFA and NFA are equally powerful.

1.13 THINGS TO REMEMBER:

- Every finite language is regular.
- A regular language may or may not be finite.
- Every non-regular language is infinite.
- An infinite language may or may not be finite.
- Every finite subset of regular set is regular.
- Every subset of regular set need not be regular.
- Subset of non regular set need not be non regular.
- Every finite language has regular subset.
- Every finite language except Ø has regular proper subset.
- If the language is defined over single symbol then it will be regular only if the length of strings of the language is in arithmetic progression.
- Let RL = Regular language and NRL = Non-Regular language then:
  - RL U NRL = RL/NRL
  - RL ∩ NRP = RL/NRL
  - NRL U NRL = RL/NRL
  - RL ∩ NRL = RL/NRL

GATE QUESTIONS

Q.1 Consider a DFA over $\sum = \{a, b\}$ accepting all strings which have number of a's divisible by 6 and number of b's divisible By 8. What is the minimum number of states that the DFA will have?

a) 8 b) 14
c) 15 d) 48

[GATE-2001]

Q.2 Given An arbitrary Non-deterministic Finite Automaton (NFA) with N states, the maximum number of states in an equivalent minimized DFA is at least

a) N^2 b) 2^N
c) 2N d) N!

[GATE-2001]

Q.3 Which of the following languages is/are regular?

$L_1 = \{ww \mid w \in \{a, b\}^*\}$

$L_2 = \{ww^R \mid w \in \{a, b\}^*, w^R \text{is the reverse of } w\}$

$L_3 = \{0^{2i} \mid i \text{ is an integer}\}$

$L_4 = \{0^{i^2} \mid i \text{ is an integer}\}$

a) L1 and L2
b) All
c) L3 and L4
d) L3 only

[GATE-2001]

Q.4 Consider the following two statements:
S1- $\{0^{2n} \mid n \geq 1\}$ is regular language.
S2 - $\{0^m 1^n 0^{m+n} \mid m \geq 1 \text{ and } n \geq 1\}$ is a regular language.
Which of the following statements is correct?

a) Only S_1 is correct
b) Only S_2 is correct
c) Both S_1 and S_2 are correct
d) None of these

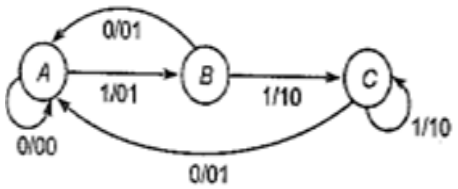
[GATE-2001]

Q.5 The smallest finite automaton which accepts the language {x | length of x is divisible by 3} has

a) 2 states b) 3 states
c) 4 states d) 5 states

[GATE-2002]

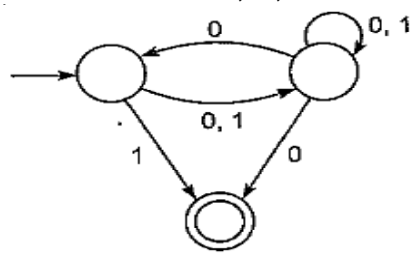
Q.6 The finite state machine described by the following state diagram with A as starting state, where an arc label is $\frac{x}{y}$ and stands for 1 -bit input and y stands for 2 bit output



a) Outputs the sum of the present and the previous bits of the input
b) Outputs 01 whenever the input sequence contains 11
c) Outputs 00 whenever the input sequence contains 10
d) None of the above

[GATE-2002]

Q.7 Consider the NFA, M, shown below:

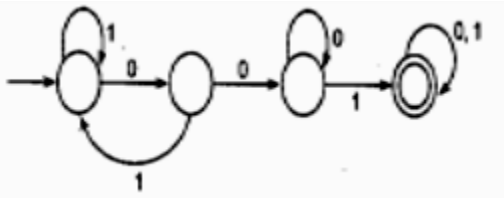


Let the language accepted by M be L. Let L_1 be the language accepted by the NFA M_1 obtained by changing the accepting state of M to a non-

accepting state and by changing the non-accepting states of M to accepting states.
Which of the following statements is true?
a) $L_1 = \{0, 1\}^* - L$ b) $L_1 = \{0, 1\}^*$
c) $L_1 \subseteq 1$ d) $L_1 = L$

[GATE-2003]

Q.8 Consider the following deterministic finite state automaton M



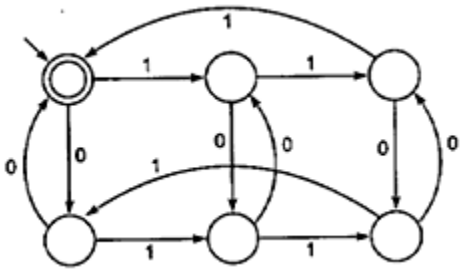
Let S denotes the set of seven bit binary strings in which the first, the fourth and the last bits are 1. The number of strings in S that are accepted by M is
a) 1 b) 5
c) 7 d) 8

[GATE-2003]

Q.9 The regular expression 0*(10*)* denotes the same set as
a) (1*0)*1*
b) 0 + (0 + 10)*
c) (0+1) *10 (0+1) *
d) None of these

[GATE-2003]

Q.10 The following finite state machine accepts all those binary strings in which the number of 1's and 0's are respectively



a) divisible by 3 and 2
b) odd and even
c) even and odd
d) divisible by 2 and 3

[GATE-2004]

Q.11 Which one of the following regular expressions is NOT equivalent to the regular expression (a + b + c)*?
a) $(a^*+b^*+c^*)^*$ b) $(a^*b^*c^*)^*$
c) $((ab)^*+c^*)^*$ d) $(a^*b^*+c^*)^*$

[GATE-IT-2004]

Q.12 Let $M=(K,\Sigma,\delta,s,F)$ be a finite state automation, where $K=\{A,B\}$, $\Sigma=\{a,b\}$, $s=A, F=\{B\}$, $\delta(A,a)=A$, $\delta(A,b)=B$, $\delta(B,a)=$ Band $\delta(B,b)=A$
A grammar to generate the language accepted by M can be specified as $G=(V,\Sigma,R,S)$, Where V=K∪Σ, S = A
Which one of the following setoff rules will make L(G)=L(M)?
a) $\{A\to aB, A\to bA, B\to bA, B\to aA, B\to\varepsilon\}$
b) $\{A\to aA, A\to bB, B\to aB, B\to bA, B\to\varepsilon\}$
c) $\{A\to bB, A\to aB, B\to aA, B\to bA, B\to\varepsilon)\}$
d) $\{A\to aA, A\to bA, B\to aB, B\to bA, A\to\varepsilon\}$

[GATE-IT-2004]

Q.13 Which of the following statements is TRUE about the regular expression 01*0?
a) It represents a finite set of finite strings.
b) It represents an infinite set of finite strings.
c) It represents a finite set of infinite strings.
d) It represents an infinite set of infinite strings.

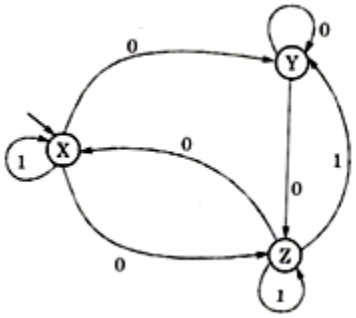
[GATE-IT-2005]

Q.14 The language $\{o^n1^n2^n \mid 1\le n\le 10^6\}$ is
a) Regular
b) Context-free but not regular.
c) Context-free but its complement is not context-free.

d) Not context-free.

[GATE-IT-2005]

Q.15 Consider the non-deterministic finite automaton (NFA) shown in the figure



State X is the starting state of the automaton. Let the language accepted by the NFA with Y as the only accepting state be L1. Similarly, let the language accepted by the NFA with Z as the only accepting state b e L2. Which of the following statements about L1 and L2 is TRUE?

a) L1 = L2 b) L1 ⊂ L2

c) L2 ⊂ L1 d) None of these

[GATE-IT-2005]

Q.16 Consider the regular grammar:

$S \rightarrow Xa \mid Ya$

$X \rightarrow Za$

$Z \rightarrow Sa \mid \varepsilon$

$Y \rightarrow Wa$

$W \rightarrow Sa$

Where S is the starting symbol, the set of terminals is {a} and the set off non-terminals is {S,W,X,Y,Z}. We wish to construct a deterministic finite automaton (DFA) to recognize the same language. What is the minimum number of states required for the DFA?

a) 2 b) 3

c) 4 d) 5

[GATE-IT-2005]

Q.17 A language L satisfies the Pumping Lemma for regular languages, and also the Pumping Lemma for con t ext-free languages. Which of the following statements about L is TRUE?

a) L is necessarily a regular language.

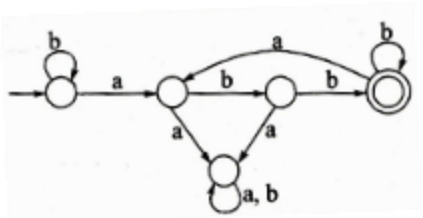
b) L is necessarily a context-free language, but not necessarily a regular language.

c) L is necessarily a non-regular language.

d) None of the above

[GATE-IT-2005]

Q.18 Consider the machine M:



The language recognized by M is

a) {W ∈ {a,b}* | every a in W is followed by exactly two b' s}

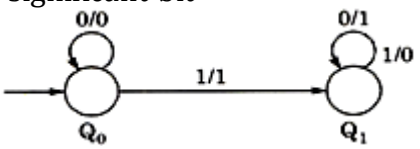
b) {W ∈ {a,b}* | every a in W is followed by at least two b' s}

c) {W ∈ {a,b}* | W contains the substring abb}

d) {W ∈ {a, b}* | W does not contain aa as a substring}

[GATE-2005]

Q.19 The following diagram represents a finite state machine which takes as input a binary number from the least significant bit



Which one off the following is TRUE?

a) It computes 1's complement of the input number

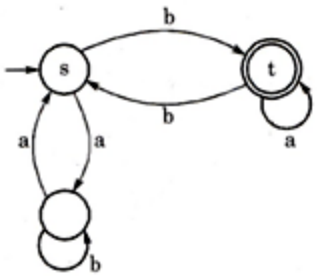
b) It computes 2's complement of the input number

c) It increments the input number

d) It decrements the input number

[GATE-2005]

Q.20 In the automaton below, s is the start state and t is the only final state

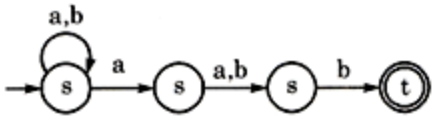


Consider the string u=abbaba, v= bab, and w=aabb. Which of the following statements is true?

a) The automaton accepts u and v but not w
b) The automaton accepts each of u, v, and w
c) The automaton rejects each of u, v, and w
d) The automaton accepts u but rejects v and w

[GATE-IT-2006]

Q.21 Which regular expression best describe the language accepted by the non-deterministic automaton below?



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

b) $(abb)^*$

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

d) $(a+b)^*$

[GATE-IT-2006]

Q.22 Consider the regular grammar below

$S \rightarrow bS|aA|\varepsilon$

$A \rightarrow aS|bA$

The My hill-Nerode equivalence classes for the language generated by the grammar are

a) $\{w \in (a+b)^* | \#_a(w) \text{ is even}\}$ and $\{w \in (a+b)^* | \#_a(w) \text{ is odd}\}$

b) $\{w \in (a+b)^* | \#_b(w) \text{ is even}\}$ and $\{\{w \in (a+b)^* | \#_b(w) \text{ is odd}\}$

c) $\{w \in (a+b)^* | \#_a(w) = \#_b(W)\}$ and $\{w \in (a+b)^* | \#_a(w) \neq \#_b(w)\}$

d) $\{\varepsilon\}, \{wa | w \in (a+b)^*\}$ and $\{wb | w \in (a+b)^*\}$

[GATE-IT-2006]

Q.23 Which of the following statements about regular languages is NOT true?

a) Every language has a regular superset
b) Every language has a regular subset
c) Every subset of a regular language is regular
d) Every subset of a finite language is regular

[GATE-IT-2006]

Directions for Q.24 to Q.25:

Let L be a regular language. Consider the constructions on L below:

I. repeat (L) = {ww | w ∈ L)

II. prefix (L) = $\{u | \exists v : uv \in L\}$

III. suffix (L) = $(v | \exists u : uv \in L)$

IV. half (L) = $\{u | v : |\exists v| = |u| \text{ and } uv \in L\}$

Q.24 Which of the constructions could lead to a non-regular language?

a) Both I and I V b) Only I
c) Only IV d) Both II and III

[GATE-IT-2006]

Q.25 Which choice of L is best suited to support your answer above?

a) (a+b)* b) {ε,a, ab, bab}
c) (ab)* d) $\{a^n b^n | n \geq 0\}$

[GATE-IT-2006]

Q.26 Consider the regular language L= (111+11111)*. The minimum number of states in any DFA accepting this language is

a) 3 b) 5
c) 8 d) 9

[GATE-2006]

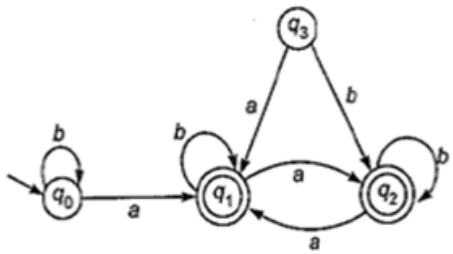
Q.27 If s is a string over (0+1)* then let $n_0(s)$ denotes the number of 0's in s and n_1 (s) the number of 1's in s. Which one of the following languages is not regular?

a) L={s∈(0+1)*|$n_0(s)$ is a 3-digit prime}
b) L= {s ∈ (0 + 1)* | for every prefix s' of s,| $n_0(s') - n_1(s')$ |≤ 2}
c) L = {s ∈ (0 +1)*| $n_0(s)-n_1(s)$ | ≤ 4
d) L = {s ∈ (0 + 1)* | $n_0(s)$ mod 7= $n_1(s)$ mod 5=0}

[GATE-2006]

Common Data for Questions 28 and 29

Consider the following Finite State Automation:



Q.28 The language accepted by this automaton is given by the regular expression

a) b* ab* ab* ab* b) (a+b)*
c) b* a(a + b)* d) b* ab* ab*

[GATE-2007]

Q.29 The minimum state automaton equivalent to the above FSA has the following number of states

a) 1 b) 2
c) 3 d) 4

[GATE-2007]

Q.30 Which of the following languages is regular?

a) $\{WW^R \mid W \in \{0,1\}^+\}$
b) $(WW^RX \mid X, W \in \{0,1\}^*$
c) $\{WXW^R \mid X, W\{0,1\}^*\}$
d) $\{XWW^R \mid X, W \in \{0,1\}^*\}$

[GATE-2007]

Q.31 A minimum state deterministic finite automaton accepting the language L={W|W∈{0,1}*, number of 0's and 1's in W are divisible by 3 and 5, respectively} has

a) 15 states b) 11 states
c) 10 states d) 9 states

[GATE-2007]

Q.32 Which of the following is true?

a) Every subset of a regular set is regular
b) Every finite subset of a non-regular set is regular
c) The union of two non-regular sets is not regular
d) Infinite union of finite sets is regular

[GATE-2007]

Q.33 The two grammars given below generate a language over the alphabet {x, y, z}

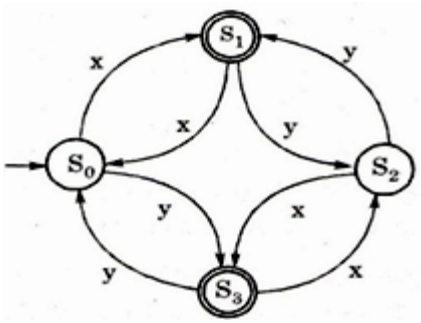
G1: S → x|z|xS|zS|yB
B→ y|z|yB|zB
G2: S→ y|z|yS|zS|xB
B→y|yS

Which one of the following choices describes the properties satisfied by the string in these languages?

a) G1 : No y appears before any x
G2 : Every x is followed by at least one y
b) G1 : No y appears before any x
G2 : No x appears before any y
c) G1 : No y appears after any x
G2 : Every x is followed by at least one y
d) G1 : No y appears after any x
G2 : Every y is followed by at least one x

[GATE-IT-2007]

Q.34 Consider the following DFA in which s_0 is the start state and s_1, s_3 are the final states.



What language does this DFA recognize?

a) All strings of x and y

b) All strings of x and y which have either even number of x and even number of y or odd number or x and odd number of y

c) All strings of x and y which have equal number of x and y

d) All strings of x and y with either even number of x and odd number of y or odd number of x and even number of y

[GATE-IT-2007]

Q.35 Consider the grammar given below

S → xB|yA

A → x|xS|yAA

B →y|yS|yBB

Consider the following strings.

(i)xxyyv (ii)xxyyxv

(iii)xyxy (iv)yxxy

(v)yxx (vi)xyx

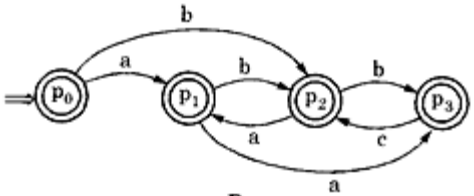
Which of the above strings are generated by the grammar?

a) (i), (ii) & (iii) b) (ii),(v)& (vi)

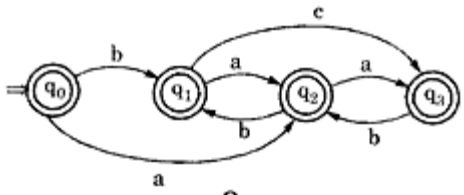
c) (iii) & (iv) d)(i),(iii) &(iv)

[GATE-IT-2007]

Q.36 Consider the following finite automata P and Q over the alphabet {a,b,c}. the start states are indicated by a double arrow and final states are indicated by a double circle. Let the languages recognized by them be denoted by L(P) and L(Q) respectively.



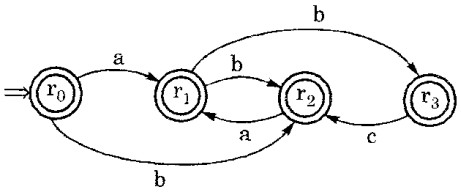
P



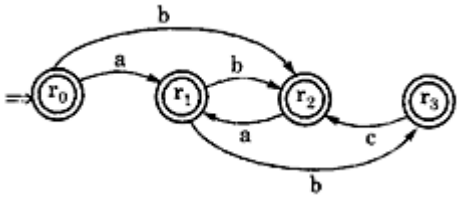
Q

The automation which recognize the language L(P)∩L(Q) is:

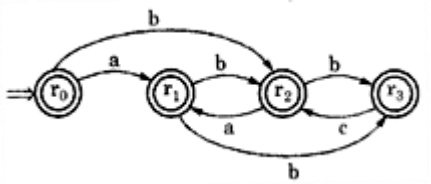
a)



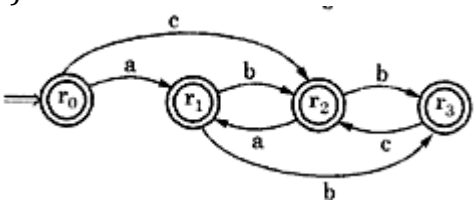
b)



c)



d)



[GATE-IT-2007]

Common Data for Q.37 to Q.38:

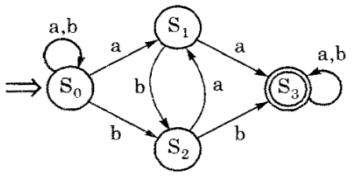
Consider the regular expression:

$R = (a+b)^*(aa+bb)(a+b)^*$

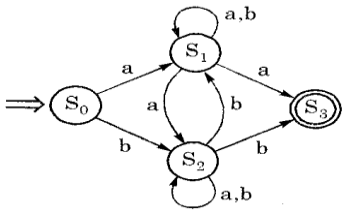
Q.37 Which of the following non-deterministic finite automata recognizes the language defined by the regular expression R? Edges

labelled λ denote transitions on the empty string.

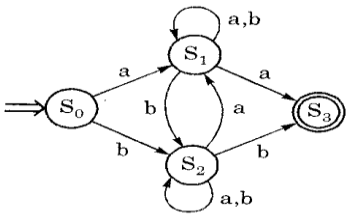
a)



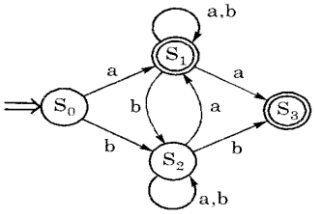
b)



c)



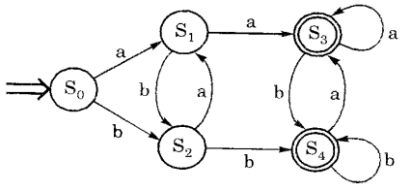
d)



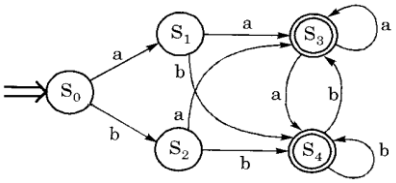
[GATE-IT-2007]

Q.38 Which deterministic finite automaton accepts the language represented by the regular expression R?

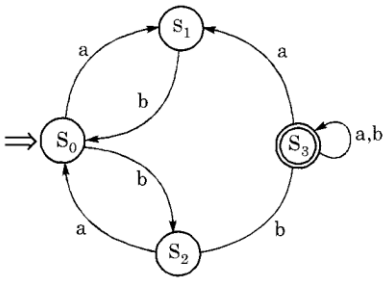
a)



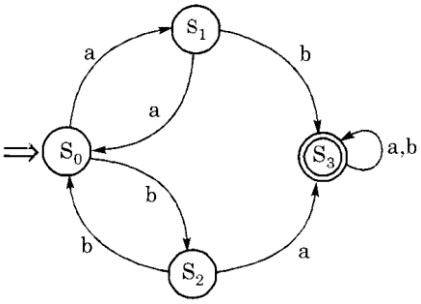
b)



c)



d)



[GATE-IT-2007]

Q.39 Which one of the regular expressions given below defines the same language as defined by the regular expression R?

a) $(a(ba)^*+b(ab)^*)(a+b)^+$

b) $(a(ba)^*+b(ab)^*)^*(a+b)^*$

c) $(a(ba)^*(a+bb)+b(ab)^*(b+aa))(a+b)^*$

d) $(a(ba)^*(a+bb)+b(ab)^*(b+aa))(a+b)^+$

[GATE-IT-2007]

Q.40 Which of the following regular expressions describes the language over {0,1} consisting of strings that contain exactly two 1's?

a) $(0+1)^*11(0+1)^*$

b) 0^*110^*

c) $0^*10^*10^*$

d) $(0+1)^*1(0+1)^*1(0+1)^*$

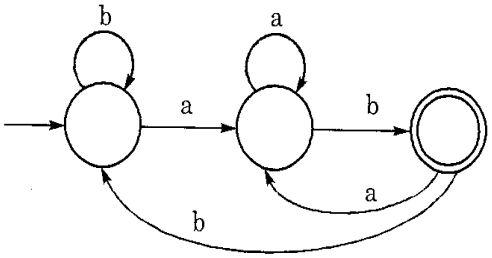
[GATE-IT-2008]

Q.41 Let N be an NFA with n states and let M be the minimized DFA with m states recognizing the same language. Which of the following in NECESSARILY true?

a) $m \le 2^n$
b) $n \le m$
c) M has one accept state
d) $m = 2^n$

[GATE-IT-2008]

Q.42 If the final states and non-final states in the DFA below are interchanged, then which of the following languages over the alphabet {a, b} will be accepted by the new DFA?



a) Set of all strings that do not end with ab
b) Set of all strings that begin with either an a or ab
c) Set of all strings that do not contain t eh substring ab,
d) The set described by the regular expression b*aa*(ba)*b*

[GATE-IT-2008]

Q.43 Which of the following languages is (are) non-regular?

$L_1 = \{0^m 1^n \mid 0 \le m \le n \le 10000\}$

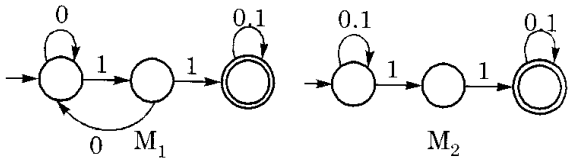
$L_2 = \{w \mid w$ reads the same forward and backward$\}$

$L_3 = \{w \in \{0,1\}^* \mid w$ contains an even number of 0's and an even number of 1's$\}$

a) L_2 and L_3 only b) L_1 and L_2 only
c) L_3 only d) L_2 only

[GATE-IT-2008]

Q.44 Consider the following two finite automata. M_1 accepts L_1 and M_2 accepts L_2. Which one of the following is TRUE?



a) $L_1 = L_2$ b) $L_1 \cap L_2 = \phi$
c) $L_1 \cap \overline{L_2} = \phi$ d) $L_1 \cup L_2 \ne L_1$

[GATE-IT-2008]

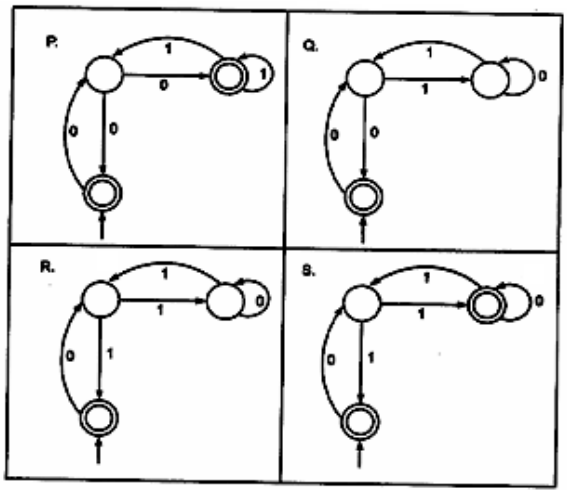
Q.45 Which of the following are regular sets?

1. $\{a^n b^{2m} \mid n \ge 0, m \ge 0\}$
2. $\{a^n b^m \mid n = 2m\}$
3. $\{a^n b^m \mid n \ne m\}$
4. $\{xcy \mid x, y, \in \{a,b\}^*\}$

a) 1 and 4 b) 1 and 3
c) 1 only d) 4 only

[GATE-2008]

Q.46 Match the following NFAs with the regular expressions they correspond to



1. ε + 0 (01*1+00)*01*
2. ε + 0(10* 1 + 00)*0
3. ε + 0(10* 1+10)* 1
4. ε + 0 (10* 1 + 10)* 10*

a) P-2, Q-1, R-3, S-4
b) P-1, Q-3, R-2, S-4
c) P-1, Q-2, R-3, S-4
d) P-3, Q-2, R-1, S-4

[GATE-2008]

Q.47 Given below are two finite state automata (→indicates the start state and F indicates a final state)

Y:

	a	b
→1	1	2
2(F)	2	1

Z:

	a	b
→1	2	2
2(F)	1	1

Which of the following represents the product automation Z × Y?

a)

	a	b
→P	S	R
Q	R	S
R(F)	Q	P
S	P	Q

b)

	a	b
→P	S	Q
Q	R	S
R(F)	Q	P
S	P	Q

c)

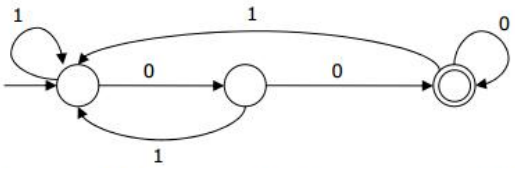
	a	b
→P	Q	S
Q	R	S
R(F)	Q	P
S	Q	P

d)

	a	b
→P	S	Q
Q	S	R
R(F)	Q	P
S	Q	P

[GATE-2008]

Q.48



The above DFA accept the set of all strings over (0, 1) that

a) begin either with 0 or 1

b) end with 0

c) end with 00

d) contain the substrings 00

[GATE-2009]

Q.49 Given the following state table of an FSM with two states A and B, one input and one output.If the initial state is A=0, B=0, what is the minimum length of an input string, which will take the machine to the state A =0, B =1 with output =1?

Present state A	Present state B	input	Next state A	Next state B	Output
0	0	0	0	0	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	1	0	0
0	0	1	0	1	0
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	0	0	1

a) 3 b) 4

c) 5 d) 6

[GATE-2009]

Q.50 Which one of .the following languages over the alphabet {0,1} is described by the regular expression (0+1)* 0(0+1)* 0(0+1) ?

a) The set of all strings containing the substring 00

b) The set of all strings containing at most two O's

c) The set of all strings containing at least two O's

d) The set of all strings that begin and end with either 0 or 1

[GATE-2009]

Q.51 Let W be any string of length n in (0,1)*. Let L be the set of all sub-strings of W. What is the minimum

number of states in a non-deterministic finite automation that accepts L?

a) n - 1 b) n

c) n + 1 d) 2^{n-1}

[GATE-2010]

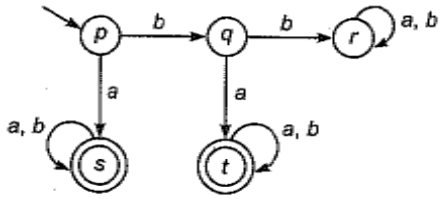
Q.52 Let L = {W ∈ (0+1)* | W has even number of 1s }, i.e., L is the set of all bit strings with even number of 1's. Which one of the regular expressions below represents L?

a) (0* 10* 1)* b) 0* (10* 10*)*

c) 0* (10* 1*)*0* d)0*1(10*1)* 10*

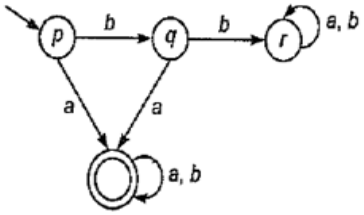
[GATE-2010]

Q.53 A deterministic finite automation (DFA) D with alphabet {a, b} is given below

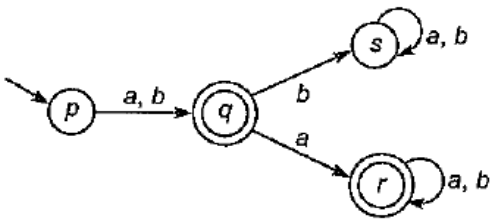


Which of the following finite state machines is a valid minimal DFA which accepts the same language as D?

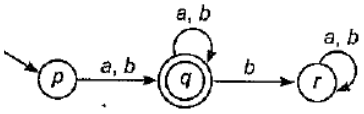
a)



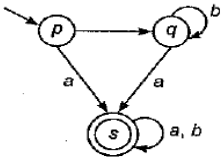
b)



c)



d)



[GATE-2011]

Q.54 Definition of a language L with alphabet {a} is given as following L = { a^{nk} | k > 0, and n is a positive integer constant} What is the minimum number of states needed in a DFA to recognize L?

a) k + 1 b) n + 1

c) 2a + 1 d) 2k + 1

[GATE-2011]

Q.55 Which of the following pairs have different expressive power?

a) Deterministic Finite Automata (DFA) and Mon-deterministic Finite Automata (NPFA)

b) Deterministic Push Down Automata (DPDA) and Non-deterministic Push Down Automata (NPDA)

c) Deterministic single-tape turing machine and non-deterministic single tape turing machine

d) Single-tape Wring machine and multi-tape turing machine

[GATE-2011]

Q.56 Let P be a regular language and Q be a context free language such that Q ⊆ P. For example, let P be the language represented by the regular expression p* q* and Q be {$p^n q^n$ | n ∈ N} Then which of the following is always regular?

a) P∩ Q b) p - Q

c) Σ*- P d) Σ*- Q

[GATE-2011]

Q.57 The lexical analysis of a modern computer language such as Java needs the power of which one of the following machine models in a necessary and sufficient sense?

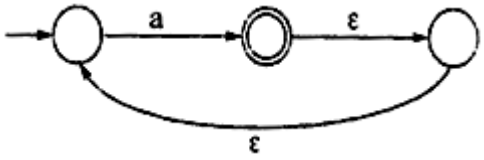
a) Finite state automata

b) Deterministic pushdown automata

c) Non-detreministic pushdown automata
d) Turing machine

[GATE-2011]

Q.58 What is the complement of the language accepted by the NFA shown below? Assume Σ = {a} and ε is the empty string.



a) $\varnothing$ b) $\{\varepsilon\}$

c) a* d) $\{a, \varepsilon\}$

[GATE-2012]

Q.59 Given the language $L = \{ab, aa, baa\}$, which of the following strings are in L*?

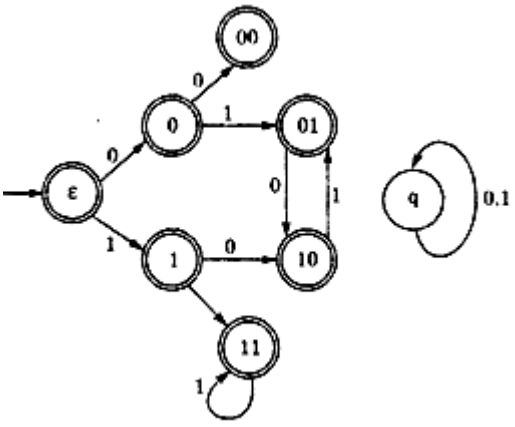
1. abaabaabaa 2. aaaabaaaa
3. baaaaabaaaab 4. Baaaaabaa

a) 1, 2 and 3 b) 2, 3 and 4
c) 1, 2and 4 d) 1, 3 and 4

[GATE-2012]

Q.60 Consider the set of strings on {0, 1} in which, every substring of 3 symbols has at most two zeros. For example, 001110 and 011001 are in the language, but 100010 is not. All strings of length less than 3 are also in the language. A partially complete DFA that accepts this language is shown below.

The missing arcs i n the DFA are



a)

	00	01	10	11	q
00	1	0			
01				1	
10	0				
11			0		

b)

	00	01	10	11	q
00		0			1
01		1			
10			0		
11		0			

c)

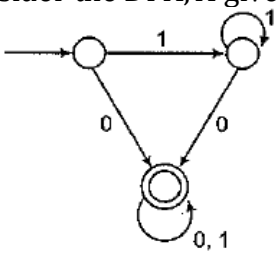
	00	01	10	11	Q
00		1			0
01		1			
10			0		
11		0			

d)

	00	01	10	11	Q
00		1			0
01				1	
10	0				
11			0		

[GATE-2012]

Q.61 Consider the DFA, A given below.



Which of the Following are false?

1. Complement of L(A) is context-free.
2. L(A)=L $(11^*0+0)(0+1)^*0^*1^*)$
3. For the language accepted by A, A is the minimal DFA.
4. A accepts all strings over (0,1} of length at least 2.

a) 1 and 3 b) 2 and 4
c) 2 and 3 d) 3 and 4

[GATE-2013]

Q.62 Consider the languages $L_1=\phi$ and L_2 = {a}. Which one or the following represents $L_1 L^*_2 \cup L^*_1$?

a) {ε} b)ø
c) a d) {ε,α}

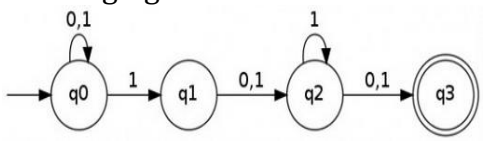
[GATE-2013]

Q.63 Which one of the following is TRUE?

a) The language $L=\{a^nb^n|n>=0\}$ is regular
b) The language $L=\{a^n|n$ is prime} is regular
c) The language L ={w| w has 3k+1 b's for some k∈N with Σ={a,b}} is regular
d) The language L={ww| w ∈ Σ* with Σ={0,1}} is regular

[GATE-2014]

Q.64 Consider the finite automaton in the following figure:

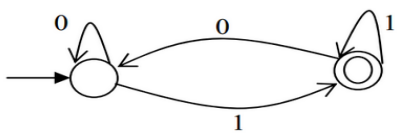


What is the set of reachable states for the input string 0011?

a) $\{q_0,q_1,q_2\}$ b) $\{q_0,q_1\}$
c) $\{q_0,q_1,q_2,q_3\}$ d) $\{q_3\}$

[GATE-2014]

Q.65 Which of the regular expressions given below represent the following DFA



I) 0*1(1+00*1)*
II) 0*1*1+11*0*1
III) (0+1)*1

a) I and II only b) I and III only
c) II and III only d) I, II and III

[GATE-2014]

Q.66 Let $L_1=\{a^n|n>=0\}$ and $L_2=\{b^n|n>=0\}$. Consider
I) $L_1.L_2$ is a regular language
II) $L_1.L_2 = \{a^nb^n|n>=0\}$
Which of the following is correct?

a) only I b) only II
c) Both I and II d) Neither I nor II

[GATE-2014]

Q.67 Let L1={w∈{0,1}* | w has at least as many occurrences of (110)'s as (011)'s}. Let L2 = {w∈{0,1}* | w has at least as many occurrences of (000)'s as (111)'s}. Which one of the following is TRUE?

a) L1 is regular but not L2
b) L2 is regular but not L1
c) Both L2 and L1 are regular
d) Neither L1 nor L2 are regular

[GATE-2014]

Q.68 The length of shortest string NOT in the language (over Σ={a,b}) of the following regular expression a*b*(ba)*a* is _____

[GATE-2014]

Q.69 The number of states in the minimal deterministic finite automaton corresponding to the regular expression (0 + 1) * (10) is _____.

[GATE-2015]

Q.70 Consider the alphabet ∑ = {0,1}, the null/empty string λ and the sets of strings X0, X1, and X2 generated by the corresponding non-terminals of a regular grammar X0, X1, and X2 are related as follows.
X0 = 1 X1
X1 = 0 X1 + 1 X2
X2 = 0 X1 + {λ}
Which one of the following choices precisely represents the strings in X0?

a) 10(0*+(10)*)1
b) 10(0*+(10)*)*1
c) 1(0*+10)*1
d) 10(0+10)*1+110(0+10)*1

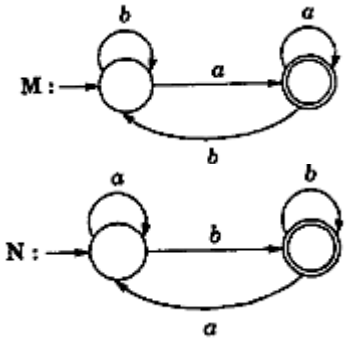
[GATE-2015]

Q.71 Let T be the language represented by the regular expression Σ*0011Σ* where Σ = {0, 1}. What is the minimum number of states in a DFA that recognizes L' (complement of L)?

a) 4 b) 5
c) 6 d) 8

[GATE-2015]

Q.72



Consider the DFAs M and N given above. The number of states in a minimal DFA that accepts the language L(M)∩L(N) is ___

[GATE-2015]

Q.73 Which of the following language is/are regular?

$L_a : \{wxw^R \mid w, x \in \{a,b\}^* \text{ and } |w|, |x| > 0\} w^R$ is the reverse of string w

$L_2 : \{a^n b^m \mid m \neq n \text{ and } m, n \geq 0\}$

$L_3 : \{a^p b^q c^r \mid p, q, r \geq 0\}$

a) L_1 and L_3 only b) L_2 only
c) L_2 and L_3 only d) L_3 only

[GATE-2015]

Q.74 Which of the following languages is generated by the given grammar?

$S \rightarrow aS \mid bS \mid \varepsilon$

a) $\{a^n b^m \mid n, m \geq 0\}$
b) $\{w \in \{a,b\}^* \mid w$ has equal number of a's and b's$\}$
c) $\{a^n \mid n \geq 0\} \cup \{b^n \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$
d) $\{a,b\}^*$

[GATE-2016]

Q.75 Which of the following decision problems are undecidable?

I. Given NFAs N_1 & N_2, is $L(N_1) \cap L(N_2) = \Phi$?

II. Given a CFG G = (N, Σ, P, S) and a string $x \in \Sigma^*$, does $x \in L(G)$?

III. Given CFGs G_1 and G_2, is $L(G_1) = L(G_2)$?

IV. Given a TM M, is $L(M) = \Phi$

a) I and IV only b) II and III only
c) III and IV only d) II and IV only

[GATE-2016]

Q.76 Which one of the following regular expressions represents the language: the set of all binary strings having two consecutive 0s and two consecutive 1s?

a)(0+1)*0011(0+1)*+(0+1)* 1100(0+1)*
b)(0+1)*(00(0+1)*11+11(0+1)* 00)(0+1)*
c)(0+1)* 00(0+1)* +(0+1)*11(0+1)*
d) 00(0+1)* 11+11(0+1)* 00

[GATE-2016]

Q.77 The number of states in the minimum sized DFA that accepts the language defined by the regular expression (0+1)*(0+1)(0+1)* is _____.

[GATE-2016]

Q.78 Language L1 is defined by the grammar: $S_1 \rightarrow aS_1b \mid \varepsilon$
Language L2 is defined by the grammar: $S_2 \rightarrow abS_2 \mid \varepsilon$
Consider the following statements:
P: L1 is regular
Q: L2 is regular

Which one of the following is TRUE?
a) Both P and Q are true
b) P is true and Q is false
c) P is false and Q is true
d) Both P and Q are false

[GATE-2016]

Q.79 Consider the following two statements:
I. If all states of an NFA are accepting states then the language accepted by the NFA is Σ*.
II. There exists a regular language A such that for all languages B, A∩B is regular.
Which one of the following is correct?
a) Only I is true
b) Only II is true
c) Both I and II are true
d) Both I and II are false

[GATE-2016]

Q.80 Consider the language L given by the regular expression (a+b)*b(a +b) over the alphabet {a, b}. The smallest number of states needed in a deterministic finite-state automaton (DFA) accepting L is _____.

[GATE-2017]

Q.81 Consider the following context-free grammar over the alphabet ∑ = {a, b, c} with S as the start symbol:
S → abScT | abcT
T → bT | b

Which of the following represents the language generated by the above grammar?
a) $\{(ab)^n(cb)^n \mid n >= 1\}$
b) $\{(ab)^n cb^{m_1} cb^{m_2} \ldots cb^{m_n} \mid n, m_1, m_2, \ldots, m_n >= 1\}$
c) $\{(ab)^n(cb^m)^n \mid n >= 1\}$
d) $\{(ab)^n(cb^n)^m \mid m, n >= 1\}$

[GATE-2017]

Q.82 Consider the following grammar over the alphabet {a,b,c} given below, S and T are non-terminals.
G1: S-->aSb|T
T--> cT|∈

G2: S-->bSa|T
T--> cT|∈
The language L1(G1) ∩ L2(G2).
a) Finite
b) Non-finite but Regular
c) Context Free but not Regular
d) Recursive but not Context-Free

[GATE-2017]

Q.83 If G is grammar with productions S → SaS | aSb | bSa | SS | ∈ where S is the start variable, then which one of the following is not generated by G?
a) abab
b) aaab
c) abbaa
d) babba

[GATE-2017]

ANSWER KEY:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
(d)	(b)	(d)	(a)	(c)	(a)	(b)	(c)	(a)	(a)	(c)	(b)	(b)	(a)	(a)
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
(b)	(d)	(b)	(b)	(d)	(a)	(a)	(c)	(b)	(a)	(d)	(c)	(c)	(b)	(c)
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
(a)	(b)	(a)	(d)	(c)	(b)	(a)	(a)	(c)	(c)	(a)	(a)	(d)	(a)	(a)
46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
(c)	(a)	(c)	(a)	(c)	(c)	(b)	(a)	(b)	(b)	(c)	(a)	(b)	(c)	(d)
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
(d)	(a)	(c)	(a)	(b)	(a)	(a)	3	3	(c)	(b)	1	(a)	(d)	(c)
76	77	78	79	80	81	82	83							
(b)	2	(c)	(b)	4	(b)	(b)	(d)							

EXPLANATIONS

Q.1 **(d)**

The minimum number of states in DFA that accepts strings which have number of a's divisible by m and number of b's divisible by n is m*n.

So minimum number of states= 6*8=48

Q.2 **(b)**

If the NFA has n states, the resulting DFA may have up to 2^n states.

Q.3 **(d)**

$L_1 = \{ww | w \in \{a,b\}^*\}$

is context sensitive language (CSL) (since these is infinite string matching in straight order).

$L_2 = \{wwR | w \in (a,b)^*, w^R$ is the reverse of w} is context free language (since there is infinite string matching in reverse order).

$L_3 = \{0^{2i} | i \text{ is an integer}\} = (00)^*$

is regular language which contains all string having even number of 0's.

$L_4 = \{0^{i^2} | i \text{ is an integer}\}$

Is context sensitive language (CSL) (since the power is infinite and non linear).

Q.4 **(a)**

Let's consider both the statements separately to find the correct option.

$S_1 : \{0^{2n} | n \geq 1|\}$

Applying the values of n, $\rightarrow S1 = 00, 0000, 000000, ...$

The behavior shown by the output is regular and hence, the language is a regular language.

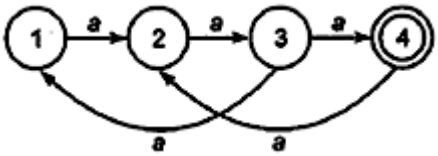
$S_2 : \{0^m 1^n 0^{m+n} | m \geq 1 \text{ and } n \geq 1|\}$

Applying the values of m and n, S_2=0100,00110000, 000111000000, Here the values of m and n are kept same so they are showing the output in symmetry but if we use the different values of m and n then the output will display a behaviour which is not regular. Therefore, confirmed is that S1 is a regular language.

Q.5 **(c)**

Since, X is divisible by 3 and we know that 3 does not have any factor apart itself and 1, so the number of states would be 3+1(F)= 4.

It can be further represented as



Q.6 **(a)**

As per the given diagram

State	Input	Output
A to A	10	00
B to A	10	00
C to A	10	00

Therefore, the finite state machine outputs the sum of the present and previous bits of the input.

Q.7 **(b)**

Even after changing non-final states to final state and final state to non-final state in NFA, still it accepts all the strings with 0's and 1's.

Q.8 **(c)**

The given bit pattern can be represented as 1 _ _ 1 _ _ 1

The four blanks can be filled in 2^4 =16 ways.

Therefore, there are 16 such strings in this pattern.

Not all of these are accepted by the machine.
The strings and its acceptance is as follows Accepted

1	0	0	1	0	0	1	√
1	0	0	1	0	1	1	√
1	0	0	1	1	0	1	√
1	0	0	1	1	1	1	√
1	0	1	1	0	0	1	√
1	1	0	1	0	0	1	√
1	1	1	1	0	0	1	√

Only these seven strings given above are accepted. All other strings in this pattern are rejected.

Q.9 (a)
Given regular expression is 0*(10*)*
A: (1*0)*1*
All strings that can be generated from given regular expression can also be generated from this.
B: 0 + (0 + 10)* and
C: (0 + 1)* 10(0 + 1)*
We can generate 11 from given regular expression which is not possible with B and C.
C: (0 + 1)* 10(0 + 1)*
Not possible as we can produce {epsilon} from the given regular expression but not from C.

Q.10 (a)
The given finite state machine accepts any string W∈(0,1)* in which the number of 1's is multiple of 3 and the number of 0's is multiple of 2.

Q.11 (c)
In choices (a), (b) and (d), inside t he parenthesis we can generate "a" and "b" and "c" separately and hence all three are same as (a + b + c)*.

In choice (c) the strings "a" and "b" cannot be generated separately since "ab" is always together.
So, choice (c) is not same as (a + b + c)*.

Q.12 (b)
$\delta(A,a) = A = A \to aA$
$\delta(A,b) = B = A \to bB$
$\delta(B,a) = B = B \to aB$
$\delta(B,b) = A = B \to bA$
Since B is final state, so we need to put
$B \to \varepsilon$. So the correct grammar is choice (b) which is
$\{A \to aA,\ A \to bB,\ B \to aB,$
$B \to bA, B \to \varepsilon\}.$

Q.13 (b)
Given regular expression is infinite set (because of *) of finite strings. A regular expression cannot generate any infinite string (Since string is always finite in length by definition).

Q.14 (a)
Given language is finite. Hence it is regular language.

Q.15 (a)
Writing Y and Z interms of incoming arrows (Arden's method), we get
$Y = X0 + Y0 + Z1$
$Z = X0 + Z1 + Y0$
Clearly, Y =Z

Q.16 (b)
The given grammar after substitution of X and Y becomes
$S \to Zaa \mid waa$
$Z \to Sa \mid \in$
$w \to Sa$
Which after substituting Z and Wis equivalent to S → Saaa |aa.
So, $L(G) = (aaa)^*aa$

So the language generated by the grammar is the set of strings with a's such that number of a mod 3 is 2. So the number of states required should be 3 to maintain the count of number of a's mod 3.

Q.17 (d)

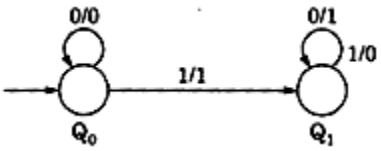
If L is regular⟹L satisfies the pumping lemma for regular languages. If L is CFL ⟹ L satisfies the pumping lemma for CFLs.

By satisfying pumping lemma, we can never say that a language is regular or CFL. It can only be used to prove that a certain language is not regular or not CFL in case the language violates the corresponding pumping lemma. So, both regular and non-regular languages can satisfy pumping lemma for regular language. Similarly, both CFLs and non-CFLs can satisfy pumping lemma for CFLs. So satisfying pumping lemma doesn't prove anything about the type of language.

Q.18 (b)

a is followed by two or more than 2 b's so the language recognized by M is {W∈{a, b}*| every a followed by atleast 2 b's}.

Q.19 (b)

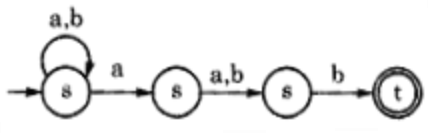


The given machine, executes the algorithm for 2's complement when input is given from LSB.

Q.20 (d)

u = abbaba : Accepted by automata.
v = bab: Not accepted by automata.
w = aabb; Not accepted by automata.

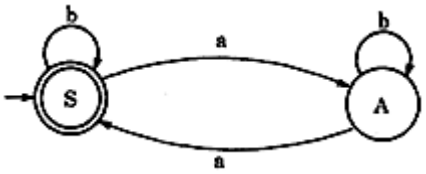
Q.21 (a)



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

Q.22 (a)

The given right-linear grammar can be converted to the following DFA.



The machine accepts all strings over the alphabet {a,b} which have an even number of a's . It is a minimal DFA. So, Myhill-Nerode equivalence classes for the language is nothing but the set of strings reaching S and A respectively.

$$S=\{w\in(a+b)^*|\#_a(w)\text{is even}\}$$

$$A=\{w\in(a+b)^*|\#_a(w)\text{is odd}\}$$

Q.23 (c)

a) Every language has a regular superset: True. Σ^* is such a superset.

b) Every language has a regular subset:
True. Φ is such a subset.

c) Every subset of a regular language is regular. False $a^nb^n \subseteq \Sigma^*$, but a^nb^n is not Regular.

d) Every subset of a finite language is regular. True. Every subset of a finite set must be finite by definition. Every finite set is regular. Hence, every subset of a finite language is regular.

Q.24 (b)

Prefix (L), suffix (L) and Half (L) are regular languages.
Repeat (L) is not a regular language but a CSL.

Q.25 (a)

Option (A): If $L=(a+b)^*$, then repeat (L) = $\{ww|w\in(a+b)^*\}$ is clearly not regular. So option (a) is best suited to show that repeat (L) need not be regular.

Option (b): IF $L=\{\varepsilon, a, ab, bab\}$, then repeat $(L)=\{ww|w\in L\}$ becomes finite and hence regular. So option (b) is not suited to show that repeat (L) need not be regular.

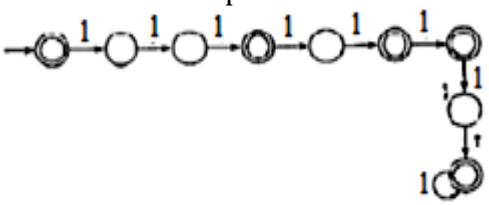
Option(c): If $L=(ab)^*$, then repeat $L=\{ww|w\in(ab)^*\}=(ab)^*$ which is regular.

So option (c) is not suited to show that repeat (L) need not be regular.

Option (d): $L=\{a^nb^n|n\geq 0\}$, is not suited since it is not regular.

Q.26 (d)

Given language L= (111 + 11111)*. Here, we can see that more than eight 1's can be generated by the combination of 111 and 11111. Therefore, number of states =8+1=9. These can be represented as



Q.27 (c)

Option (a)

$L=\{s\in(0+1)^*|\, n_0(s)$ is a 3- digit prime$\}$

It is a regular language.

Option (b)

$L=\{s\in(0+1)^*|$ for every prefix s' of $s\,|\,n_0(S')-n_1(S')|\leq 2\}$

It is a regular language.

Option (c)

$L=\{s\in(0+1)^*|\,n_0(s)-n_1(s)|\leq 4\}$

It is not a regular language.

Option (d)

$L=\{s\in(0+1)^*|\,n_0(s) \bmod 7=n_1(s) \bmod 5=0\}$

It is a regular language.

Q.28 (c)

The behavior as per the given PDA is as seen below.

q_0 to $q_1 \rightarrow b^*a$

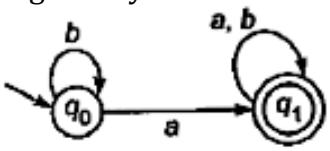
q_1 to $q_2 \rightarrow (a+b)^*$

Regular expression =b* a(a + b)*

Q.29 (b)

The FSA as obtained in the previous question is b * a(a + b) *.

The minimum number of states is thus given by



q_0 and q_1 are the states that are required at most and hence the minimum number of states is 2(q_0 & q_1).

Q.30 (c)

Let's study the regular languages.

Conventions on regular expressions:

1. Bold face is not used for regular expressions when the expression is hot confusing. So, for example, (r +s) is used instead of (r + s).
2. The operation * has precedence over concatenation, which further has precedence over union (+). Thus, the regular expression (a + (b(c*))) is written as a + be*.
3. The concatenation of k r's, where r is a regular expression, is written as r^k. Thus, for example $rr=r^2$. The language corresponding to r^k is L_r^k, Where L_r is the language corresponding to the regular expression r. For a recursive definition of L_r^k.
4. The (r^+) is used as a regular expression to represent L_r^*. Since,

language Lean be expressed as $r=[0(0+1)^*0]+[1(0+1)^*1]$ and follows the above convention, therefore is regular.

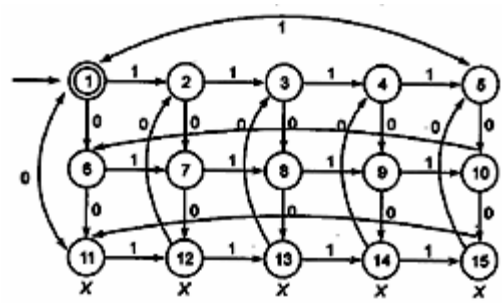
Q.31 (a)

It is given that the 0's and 1's are divisible by 3 and 5 and we know that 3 and 5 do not have any factor other than themselves or 1 i.e., these cannot be further breakdown.

Therefore, number of states

=3×5 =15

The schematic representation is as follows:



Q.32 (b)

Finite subset of non-regular set is regular as we know that Pumping Lemma can be applied on all the finite sets that state that all finite sets are regular.

Infinite union of finite set is not regular because regular sets can never be closed under infinite union.

Q.33 (a)

G1: $S\to x|z|xS|zS|yB$

$B\to y|z|yB|zB$

$B\to(y+z)^+$

Substitute in s to get

$S\to x|z|xS|zS|y(y+z)^+$

Now solution o f S is

$S\to(x+z)^*S$

$S\to(x+z)^*\left(x+z+y(y+z)^+\right)$

So $L(G1)=S=(x+z)^++$

$(x+z)^*y(y+z)^+$

G1 generates every string in which " no y appears before any x".

$G2: S\to y|z|yS|zS|xB$

$B\to y|yS$

Substitute B in S to get

$S\to y|z|yS|zS|xy|xyS$

Now solution of S is

$S\to(y+z+xy)^*S$

$S\to(y+z+xy)^*(y+z+xy)$

So $L(G2) = S = (y + z + xy)^+$

G2 generates every string in which "every x followed by atleast one y" .

Q.34 (d)

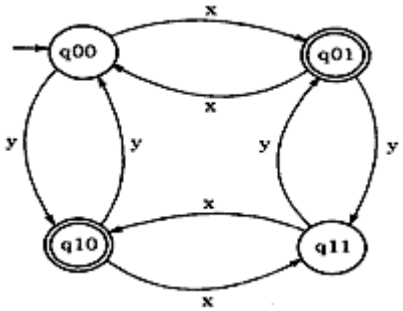
Given DFA can be redesigned as S0 asq00, S1 as q10, S2 as q11, S3 as q01.

Each state is

$q_{ab}\left[a=n_a \bmod 2,\ b=n_b \bmod 2\right].$

$q00$ as $n_a \bmod 2 = 0,\ n_b \bmod 2=0$

[Number of x is ever, number of y is even].



q01 is final state mean where number of x is even and number of y is odd. q10 if final state mean where number of x is odd and number of y is even.

Q.35 (c)

This grammar cannot generate any string starting with xx or ending with xx so (i), (ii) and (v) cannot be generated by the grammar. xyx also cannot be generated by the

grammar. The derivation for (iii) xyxy and (iv) yxxy is shown below.

$S \to xB \to xyS \to xyxB \to xyxy$

$S \to yA \to yxS \to yxxS \to yxxy$

So only (iii) and (iv) can we derived from this grammar.

Q.36 (b)

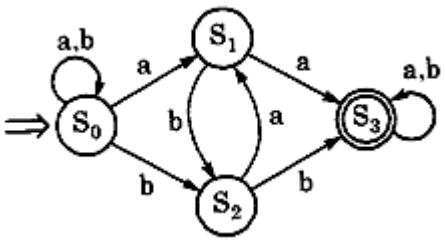
L(P)∩L(Q) must contain all strings common to P and Q. Note that `aa' is common to both P and Q and hence must be accepted by any FA accepting L(P)∩L(Q). However, all of the given machine reject 'aa'. Therefore option (e) i.e. none of these is the right answer.

Q.37 (a)

The language of the given regular expression R is 'containing the substring aa or bb'. Option (a) is the correct machine for this language.

Q.38 (a)

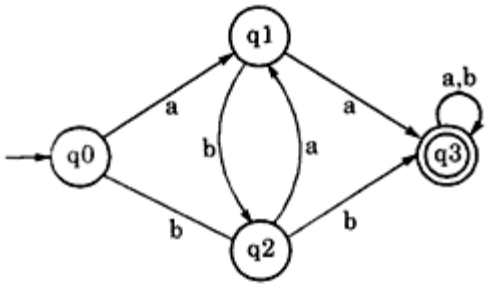
In option (a) S3 and S4 together act as a permanent accept and can therefore be collapsed into a single permanent accept state as shown below.



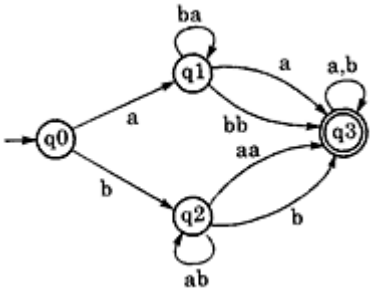
This machine clearly accepts all strings containing the substring `aa' or `bb', which is same as regular expression R,

Q.39 (c)

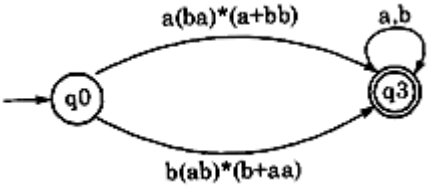
$R = (a+b)^*(aa+bb)(a+b)^*$ has following equivalent DFA



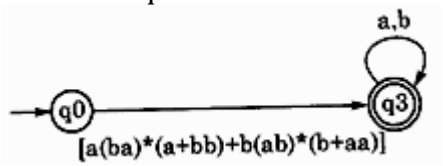
Which is equivalent Transition graph [by removing transition from q1 to q2 and q2 to q1 but does not effect on language]



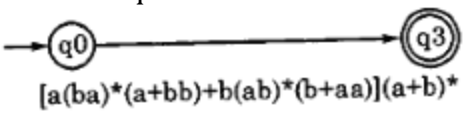
That is equivalent to:



Which is equivalent to:



Which is equivalent to:



Equivalent regular expression is

$\lfloor a(ba)^*(a+bb)+b(ab)^*(b+aa) \rfloor (a+b)^*$.

Q.40 (c)

a) with at least 2 consecutive 1's, any no of 0's and any no of 1's
b) exactly two consecutive 1's
c) exactly two 1's but need not be consecutive
d) Any no of 1's & 0's with at least two 1's

Q.41 (a)

A state in a DFA will be a subset of the set of states of the equivalent NFA.

So, the maximum number of states in the equivalent DFA of an NFA, will be 2^n, where n is the number of states in NFA, as a set with n elements has maximum 2^n subsets.

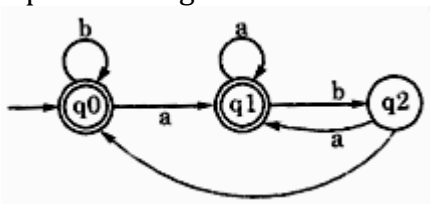
So, number of states in equivalent minimal DFA ≤ number of states in equivalent DFA ≤ 2^n .

Q.42 (a)

Interchanging final and non-final states of DFA is used for complementation.

Given D FA generates all strings end with ab. Complement of DFA accepts all strings do not end with ab.

Complement of given DFA is :



It accepts all strings that do no end with ab.

Q.43 (d)

L1 is regular, since 10000 is finite.

L3 is also regular (mod machine, since finite number of residue combinations). $L2 = \{w | w = w^R\}$ which is the Palindrome language which is known to be non regular.

Q.44 (a)

M1 is a DFA accepting all string containing 2 consecutive 1's (containing the substring `11').M2 is a NFA accepting the same language. Containing 2consecutive 1's.

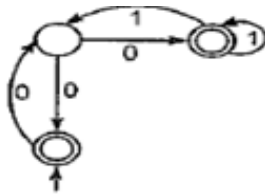
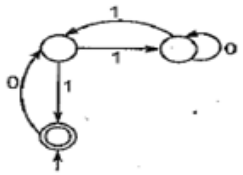
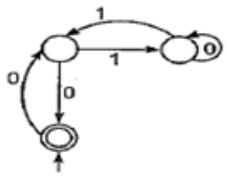
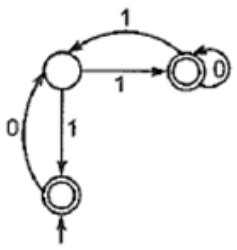
So, L1 = L2.

Q.45 (a)

1. There is no comparison between number of a's and b's. We must have even number of b's. This can be accepted by DFA. So it is regular.
2. There is a relation between number of a's and b's which can be solved by stack. So this is CFL.
3. There is a relation between number of a's and b's which can be solved by stack. So this is CFL.
4. The given language is set of strings having c in it. There exists DFA to accept all such strings. So it is regular.

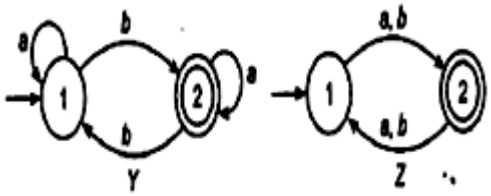
Q.46 (c)

The correct sequence is given as follows:

P. 	1. $\varepsilon+0(01*1+00)*01*$
R. 	2. $\varepsilon+0(10*1+00)*0$
Q. 	3. $\varepsilon+0(10*1+10)*1$
S. 	4. $\varepsilon+0(10*1+10)*10*$

Q.47 (a)

Given: Transition table for Y and Z
The number of states of Z = 2
The number of states of Y = 2
Z x Y



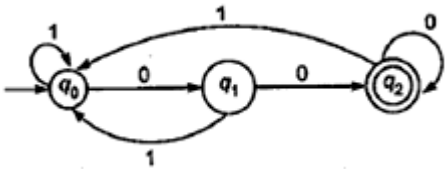
No. of states of the product of Z and Y = 2 x 2 = 4 Now, the states as per the given option are P, Q, R and S. The finite state automata is

	a	B
→P	S	R
Q	R	S
R(F)	Q	P
S	Q	P

Table for Z x Y is

	a	B
→(1,2)	(2,1)	(2,2)
(1,2)	(2,2)	(2,1)
(2,1)	(1,1)	(1,2)
F(2,2)	(1,2)	(1,1)

Q.48 (c)

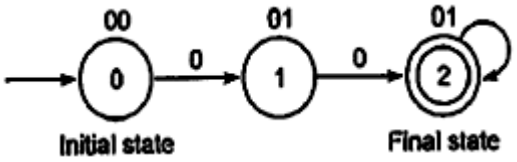


State	Input	Output
q_0 to q_0	1, 0	00
Q_1 to q_0	1, 0	00
Q_2 to q_0	1, 0	00

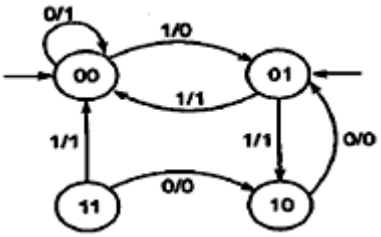
Therefore, the above DFA ends with 00.

Q.49 (a)

The initial state =00
Final state required =01
Let us construct the transition diagram for the given.



We get the total number of states to be 3 when getting the desired output. The transition diagram can further be elaborated as below.



There can be 4 states 00, 01, 10, 11. With this, the FSM can be designed as the desired output is obtained with the input string 101, however the concern is number of states which we found to be 3.

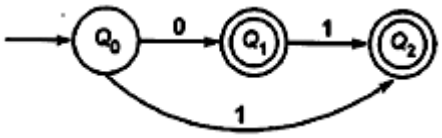
Q.50 (c)

We have given the relation
(0 + 1)* 0(0 + 1)* 0(0 + 1)*
Here, the accepting languages are
L={00, 000, 100, 001, 010, 0000, 0001, 1000, 1001, 0100, 1100, 0010, 0011, 0110, 0101, 1010,...}
The common feature in the accepting languages can be seen that they consist of atleast two 0's.

Q.51 (c)

The length of W is not defined. It is given as n so let us take an example.
Consider the length of W be 2.
Let W =01
The sub-strings thus formed are 0, 1 and 01.

The number of states required by NFA as compared to the string will give us the minimum number of states in the NFA accepting L.
The automaton for the above is



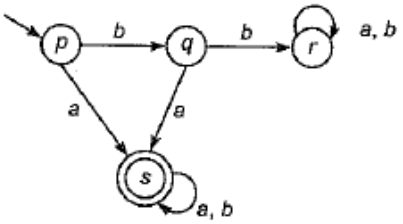
Here the minimum number of states required is 3 which is 1 greater than the string of length 2. Therefore, n + 1 are the desired solution.

Q.52 (b)

It is given that L is the set of all bit strings with even number of 1's, so the regular expression should exhibit the same.
Now, the min string should be s and the string should be e, 0, 11, 101...
The string obtained from such expression is 0* (10*10*)*

Q.53 (a)

As state (s) and (t) both are final states and accepting a* + b*, we can combine both states and we will get



Q.54 (b)

As n is constant at least n + 1 states will be required to design a^{nk}

Q.55 (b)

DPDA and NPDA because an NPDA cannot be converted into DPDA.

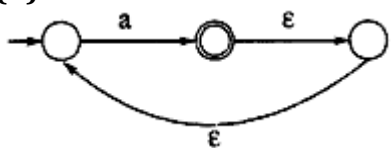
Q.56 (c)

Σ^*- P as Σ^*- P is the complement of P and complement of regular language is also regular.

Q.57 (a)

Lexical analysis only requires the power of FA.

Q.58 (b)



Language for NFA L= a^+

Now $\overline{L} = \Sigma^* - L$

$\Rightarrow \quad \overline{L} = a^* - a^+$

$\Rightarrow \quad \overline{L} = \{\varepsilon\}$

Q.59 (c)

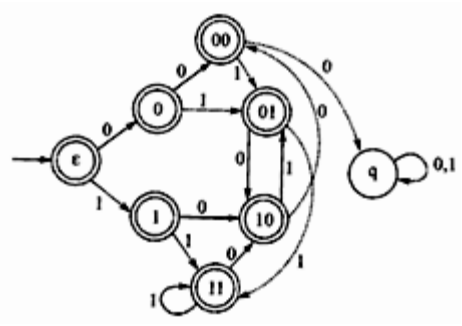
$L = \{ab, aa, baa\}$

The breakdown of the strings 1,2,4 in terms of ab, aa and baa is shown below:
1. a baa baa ab aa
2. aa aab aa aa
3. baa aa a baa

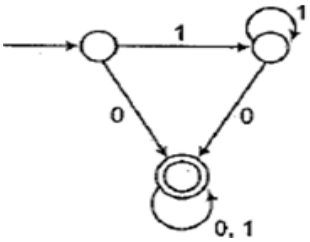
String no (3) has no breakdown in terms of strings in L and hence string (3) does not belong to L*. Only 1, 2 and 4 belongs to L*.

Q.60 (d)

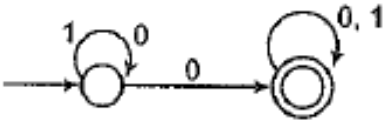
Notice that the state names are given based on ending bits of the string, which has been processed.
The are from 00 labeled "0" should go to trap state q (since at most 2 zeros are allowed in any substring).
Based on this fact, option (a) and (b) are incorrect Between option (c) and (d), if you look at arc labeled "I" from state 01, this arc should go to state 11 since the string at this point is ending with 11. So option (c) is wrong and option (d) is correct.
The dfa corresponding to correct option (d) is correct.
The dfa corresponding to correct option (d) is shown below with missing arrows shown in dotted lines.



Q.61 (d)



1. Element of L(A) is context free – true As there exist a DFA, L(A) is regular. Regular languages are closed under complement.
   ∴ Complement of L(A) is regular and therefore context free.
2. $L(A) = L((11^*0+0)(0+1)^*0^*1^*) - \text{true}$
   AS $L(A) = L((11^*0+0)(0+1)^*)$ is Equivalent to $L((11^*0+0)(0+1)^*0^*1^*)$ For the language accepted by A,
   A is minimal DFA – false
   Minimal DFA



3. A accepts all strings over {0, 1} of length at least 2 – false.
   The given DFA A accepts the string "0" Ans (d) 3 and 4 only false.

Q.62 (a)

Considering the languages $L_1 = \phi$ and $L_2 = \{a\}$ for all languages L it is known that Ø. L =Ø
Suppose, ∃α string $s \in \phi$. L ∃s' Such that s=s' s"
And $s' \in \phi s'' \in L$
But $s' \in \phi \because \phi$ is an empty language
$\therefore \phi, L = \phi$
$\{\varepsilon\} \subseteq L^*$ for all languages L.
As it indicates taking letters from the language and concatenating them '0' length string which is possible for all languages.
$\therefore \varepsilon$ is in ϕ^*
$L_1.L_2 = \phi$
And $L_1^* = \{\varepsilon\}$
$L_1.L_2{}^* \cup L_1^* = (L_1.L_2) \cup L_1^*$
$\therefore \phi \cup \{\varepsilon\} = \{\varepsilon\}$
Hence, the answer is option (a) = {**ε**}

Q.63 (c)

Only for C we can build the finite automata

Q.64 (a)

The given automaton is NFA and clearly the states q_0, q_1, q_2 can be reached with the input string 0011.

Q.65 (b)

I and III are the expressions for DFA. But II is not representing the given DFA as it accepts strings like 11011 which is not accepted by regular expression.

Q.66 (a)

Both L1 and L2 are regular. The concatenation of two regular languages is regular.
But L1.L2 ={$a^m b^n$ | m, n>=0}

Q.67 (a)

L1 is regular let us consider the string 011011011011 In this string, number of occurrences of 011 are 4 but when we see here 110 is also occurred and the number of occurrence of 110 is 3. Note that if i add a 0 at the last of string we can have same number of occurrences of 011 and 110 so this string is accepted. We can say if the string is

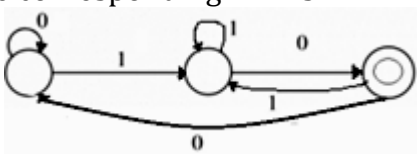
ending with 011 so by appending a 0 we can make 110 also. Now string2: 110110110110 in this number of occurrences of 110 is 4 and 011 is 3 which already satisfy the condition So we can observe here that whenever 110 will be there string will be accepted So with this idea we can build an automata for this. Therefore, it is regular.

Q.68 (3)

Length 0 string is present as it accepts epsilon, all length 1 strings are present (a,b) and also length two string aa, ab, ba, bb are present, But 'bab ' is not present. So length of string not accepted is 3.

Q.69 (3)

The corresponding DFA is



Q.70 (c)

The smallest possible string by given grammar is "11".

X0 = 1X1

= 11X2 [Replacing X1 with 1X2]

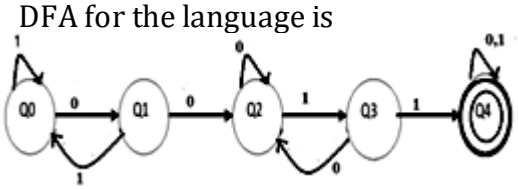
= 11 [Replacing X2 with λ]

The string "11" is only possible with option (c)

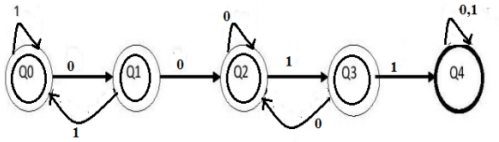
Q.71 (b)

In regular language, DFA containing sub-string of length n contains n+1 state. Complement of that DFA also contain n+1 state.

DFA for the language is



Following is the DFA for complement of L which again contains 5 states.



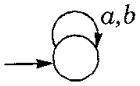
Q.72

$L(M) = (a+b)^*a$ (strings ending with a)

$L(N) = (a+b)^*b$ (strings ending with b)

$L(M) \cap L(N) = \phi$ (nothing in common)

∴ Number of states = 1



Minimal DFA which accepts empty language.

Q.73 (a)

$L_1 = \{wxw^R \mid w,\ x \in \{a,b\}^* \text{and} |w|, |x| > 0\}$

w cannot be put as ``ϵ" since $|w| > 0$.

So we put w as its smallest string which is `a' and `b'and get the regular expression:

$r = a(a+b)^+ a + b(a+b)^+ b$.

Now putting w as any other string like say "ab" will not add any new string to the expression r, since any such string so generated will be either already generated by either $a(a+b)^+ a$ or $b(a+b)^+ b$.

So the given language

$= a(a+b)^+ a + b(a+b)^+ b$ which is clearly regular.

∴ L_1 is regular.

$L_2 = \{a^n b^m \mid m \neq n\}$. This language has infinite comparisons between number of a's and b's . L_2 is not regular language.

$L_3 : \{a^p b^q c^r \mid p,q,r \geq 0\} = a^*b^*c^*$ is a regular language.

Q.74 (d)

G: $G : S \rightarrow aS|bS|\varepsilon$

$L(G) = \{a,b\}^*$.

Q.75 (c)

I. Disjointedness problem of regular = Decidable
II. Membership of CFL's = Decidable
III. Equivalence of CFL's = Undecidable
IV. Emptiness of RE language's = Undecidable

So, III and IV only is correct answer.

Q.76 (b)

Option A represents those strings which either have 0011 or 1100 as substring.

Option C represents those strings which either have 00 or 11 as substring.

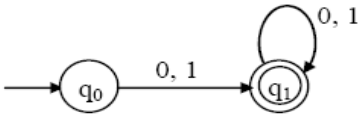
Option D represents those strings which start with 11 and end with 00 or start with 00 and end with 11.

Option B represents those strings having two consecutive 0s and two consecutive 1s.

Q.77 (2)

R.e (0+1)*(0+1)(0+1)*

The number of states in minimal DFA is 2



Q.78 (c)

L_1: $S_1 \rightarrow aS_1 b \mid \varepsilon$

L_2: $S_2 \rightarrow abS_2 \mid \varepsilon$

L_1: $\{a^n b^n \mid n > 0\} \rightarrow$ CFL

L_2: $(ab)^* \rightarrow$ RL

P is false and Q is true

Q.79 (b)

I is true because consider the empty lnguage which is regular. And its intersection with any language is regular. I is false because, if we do not define the transition for an input 'a' at a particular state but the next character in string is 'a', then it cannot be accepted by that NFA.

2 CONTEXT-FREE GRAMMARS AND PUSHDOWN AUTOMATA

As we have seen in previous chapter, many languages are not regular languages. As well as finite automata can't be constructed for few languages. So to solve few problems we need more powerful language and more powerful machine. Context-free grammars allows us to generate more interesting languages; much of the syntax of a high-level programming language. In this chapter we are going to learn the context free language and push down automata.

2.1 CONTEXT FREE GRAMMAR (CFG)

Definition:
A context-free grammar (CFG) is a 4-tuple G = (V, Σ, S, P), where V and Σ are disjoint finite sets, S ∈ V , and P is a finite set of formulas of the form A → α, where A ∈ V and α ∈ (V ∪ Σ)∗.
Elements of Σ are called terminal symbols, or terminals, and elements of V are variables, or nonterminals. S is the start variable, and elements of P are grammar rules, or productions.

Example of CFG
Consider the Grammar:
V = {S}, S = {S}, Σ = {id} and P is as follows:
S → (S)
S → S + S
S → S − S
S → S × S
S → S/S
S → id
Sometimes we also write as:
S → (S)| S + S | S − S | S × S | S/S | id
Here, is an example of how a string id × (id + id) can be generated from the above grammar.
S ⇒ S × S ⇒ S × (S) ⇒ id × (S) ⇒ id × (S + S) ⇒ id × (id + S) ⇒ id × (id + S) ⇒ id × (id + S) ⇒ id × (id + id)

Language and CFG
All the set of strings generated by CFG forms the language. Every CFG generates one language.

Examples:

1. L = {$a^n b^n$ | n>= 0}
   S → aSb | ^
2. L = {$a^n b^n$ | n> 0}
   S → aSb | ab
3. L = {Even length palindrome strings, Σ = {a,b}}
   S → aSa | bSb | ^
4. L = {Odd length palindrome strings, Σ = {a,b}
   S → aSa | bSb | a | b
5. L = {Every string of Σ*, Σ = {a,b}}
   S → aSa | bSb | aSb | bSa | ^ |a | b
   Or
   S → aS | bS | ^
6. L = {Strings having middle symbol a, Σ = {a,b}}
   S → aSa | bSb | aSb | bSa | a
7. L = {Strings having first, last and middle symbol same, Σ = {a,b}}
   S → aAa | bBb | a | b
   A → aAa | bAb | aAb | bAa | a
   B → aBa | bBb | aBb | bBa | b
8. L = (a+b)*ab
   S → Aab
   A → aA | bA | ^
9. L = {$a^i b^j c^k$ | j=i+k, Σ = {a,b,c}}
   S → AB
   A → aAb | ^
   B → bBc | ^

10. $L = \{x \mid n_a(x) = n_b(x), \Sigma = \{a,b\}\}$
    $S \rightarrow aSb \mid bSa \mid SS \mid \wedge$

2.2 DERIVATION, PARSE TREE AND AMBIGUITY

Derivation:
The process of deriving a string from the starting non-terminal is called a derivation.

Derivation can be of two types:
Left Most Derivation: In the process of derivation every time if left most non-terminal gets replaced by one of right side of its production then it is called left most derivation.

Example:
Consider the grammar E → E+E | E*E| id

Left most derivation for the string id+id*id is as follows:
E → E+E
→ id+E
→ id+E*E
→ id+id*E
→ id+id*id

The left most non-terminal (underline) gets replaced every time by one of right side of its production.

Here, E+E, id+E, id+E*E, id+id*E, id+id*id are called left sentential forms – in general sentential forms.

Right Most Derivation: In the process of derivation every time if right most non-terminal gets replaced by one of right side of its production then it is called left most derivation.

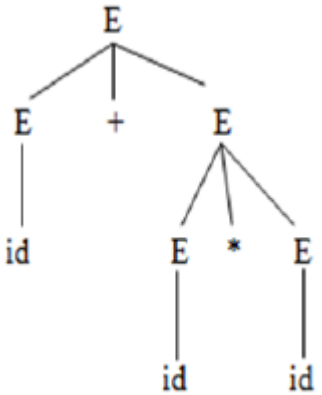
Example:
Consider the grammar E → E+E | E*E| id
Left most derivation for the string id+id*id is as follows:

E → E+E
→ E+E*E
→ E+E*id
→ E+id*id
→ id+id*id
The right most non-terminal (underline) gets replaced every time by one of right side of its production.

Here, E+E, E+E*E, E+E*id, E+id*id, id+id*id are called right sentential forms – in general sentential forms.

Parse Tree/ Derivation Tree:
Pictorial representation of derivation is called parse tree.
Root Node: Starting non-terminal
Internal Nodes: Non-Terminals
Leaf Nodes: Terminals

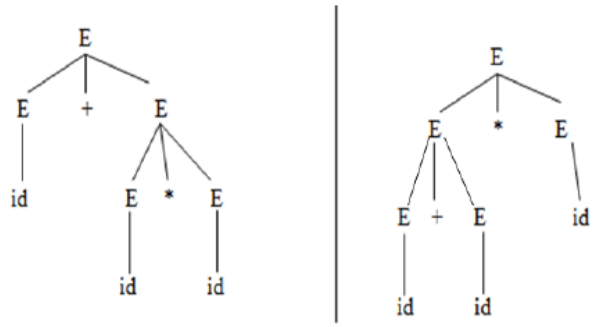
Example: Parse tree for id+id*id is as below.



Ambiguous Grammar:
A context-free grammar G is ambiguous if for at least one $x \in L(G)$, x has more than one derivation tree (or, equivalently, more than one leftmost derivation).

Example:
The grammar E → E+E | E*E | id is ambiguous.

Because there exist more than one different parse tree for the string id+id*id which are shown below.



2.3 LEFT RECURSION, LEFT FACTORING AND NORMAL FORMS

Left Recursion:
A grammar is left recursive if it has a non terminal (variable) A such that their is a derivation
A → Aα | β

where α ∈ (VUT)* and β ∈ (VUT)*

Due to the presence of left recursion some top down parsers enter into infinite loop so we have to eliminate left recursion.

Let the productions is of the form A → Aα1 | Aα2 | Aα3 | ….. | Aαm | β1 | β2 | …. | βn

Where no βi begins with an A then we replace the A-productions by

A → β1 A' | β2 A' | ….. | βn A'
A' → α1A' | α2A' | α3A'| ….. | αmA' | ε

The nonterminal A generates the same strings as before but is no longer left recursive.

Example:

E --> E + T | E-T | T
Here, A = E, α1 = +T, α2 = -T, β1 = T
E → T E'
E' → ε | +TE' | -TE'

Left Factoring:
A grammar is said to be left factored when it is of the form :
A → αβ1 | αβ2 | αβ3 | …… | αβn | γ i.e the productions start with the same terminal (or set of terminals).

For the grammar
A → αβ1 | αβ2 | αβ3 | …… | αβn | γ
The equivalent left factored grammar will be:
A → αA' | γ
A' → β1 | β2 | β3 | …… | βn

Example:
S → iEtS|iEtSeS|id
Here, α = iEtS, β1 = ε, β2 = eS, γ = id
The equivalent left factored grammar will be:
S → iEtSS'|id
S' → ε|eS

Removal of Null Production:
In a CFG, a non-terminal symbol 'A' is a nullable variable if there is a production A → ε or there is a derivation that starts at A and finally ends up with A ⇒* ε.

Removal Procedure:
Step 1 – Find out nullable non-terminal variables which derive ε.
Step 2 –Replace all combinations of nullable variables by ε.

Example:
S → ASA | aB | b, A → B, B → b | ∈
Here. A and B both are nullable.

S → ASA | aB | b | a | SA | AS | S, A → B, B → b

Removal of Unit Productions:
Any production rule in the form A → B where A, B are Non-terminal is called unit production..

Removal Procedure –
Step 1 – To remove A → B, add production A → x to the grammar rule whenever B → x

occurs in the grammar. [x is Terminal, x can be Null]
Step 2 – Delete A → B from the grammar.
Step 3 – Repeat from step 1 until all unit productions are removed.

Example:
S → aA, A → a | B, B → bb
After removing unit production:
S → aA, A → a | bb

CHOMSKY NORMAL FORM:
A CFG is in Chomsky Normal Form if the productions are in the following forms –
A → a
A → BC
S → ε
where A, B, and C are non-terminals and a is terminal.

Steps to Convert into Chomsky Normal Form:
Step 1 – If the start symbol S occurs on some right side, create a new start symbol S' and a new production S'→ S.

Step 2 – Remove Null productions.

Step 3 – Remove unit productions.

Step 4 – Replace each production A → B1…Bn where n > 2 with A → B1C where C → B2 …Bn. Repeat this step for all productions having two or more symbols in the right side.

Step 5 – If the right side of any production is in the form A → aB where a is a terminal and A, B are non-terminal, then the production is replaced by A → XB and X → a. Repeat this step for every production which is in the form A → aB.

Example:
Convert the following CFG into CNF

S → ASA | aB, A → B | S, B → b | ε

Solution
(1) Since S appears in R.H.S, we add a new state S0 and S0→S is added to the production set and it becomes –
S0→S, S→ ASA | aB, A → B | S, B → b | ε

(2) Now we will remove the null productions – B → ε and A → ε
After removing B → ε, the production set becomes –
S0→S, S→ ASA | aB | a, A → B | S | ε, B → b

After removing A → ε, the production set becomes –
S0→S, S→ ASA | aB | a | AS | SA | S, A → B | S, B → b

(3) Now we will remove the unit productions.
After removing S → S, the production set becomes –
S0→S, S→ ASA | aB | a | AS | SA, A → B | S, B → b

After removing S0→ S, the production set becomes –
S0→ ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → B | S, B → b

After removing A→ B, the production set becomes –
S0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA
A → S | b
B → b

After removing A→ S, the production set becomes –
S0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA
A → b |ASA | aB | a | AS | SA, B → b

(4) Now we will find out more than two variables in the R.H.S

Here, S0→ ASA, S → ASA, A→ ASA violates two Non-terminals in R.H.S.

Hence we will apply step 4 and step 5 to get the following final production set which is in CNF –

S0→ AX | aB | a | AS | SA
S→ AX | aB | a | AS | SA
A → b |AX | aB | a | AS | SA
B → b
X → SA

(5) We have to change the productions S0→ aB, S→ aB, A→ aB

And the final production set becomes –
S0→ AX | YB | a | AS | SA
S→ AX | YB | a | AS | SA
A → b A → b |AX | YB | a | AS | SA
B → b
X → SA
Y → a

Note: If G is a Context Free Grammar in the Chomsky Normal Form, then for any string w belongs L(G) of length n ≥ 1, it requires exactly 2n-1 steps to make any derivation of w.

GREIBACH NORMAL FORM:
A CFG is in Greibach Normal Form if the Productions are in the following forms –

A → b
A → bD1…Dn
S → ε

where A, D1,....,Dn are non-terminals and b is a terminal.

Steps to Convert a CFG into Greibach Normal Form:
Step 1 – If the start symbol S occurs on some right side, create a new start symbol S' and a new production S' → S.

Step 2 – Remove Null productions. (Using the Null production removal algorithm discussed earlier)

Step 3 – Remove unit productions. (Using the Unit production removal algorithm discussed earlier)

Step 4 – Remove all direct and indirect left-recursion.

Step 5 – Do proper substitutions of productions to convert it into the proper form of GNF.

Example:
S → XY | Xn | p
X → mX | m
Y → Xn | o

Solution
Here, S does not appear on the right side of any production and there are no unit or null productions in the production rule set. So, we can skip Step 1 to Step 3.

Step 4

Now after replacing X in S → XY | Xo | p with mX | m we obtain

S → mXY | mY | mXo | mo | p.

And after replacing X in Y → Xn | o with the right side of X → mX | m we obtain

Y → mXn | mn | o.

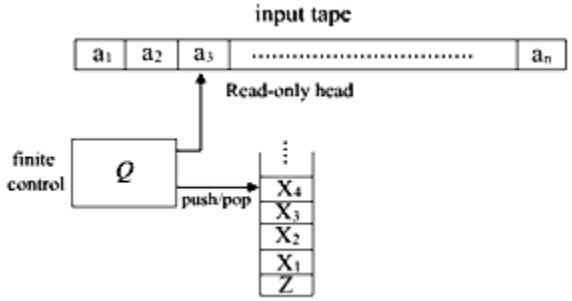
Two new productions O → o and P → p are added to the production set and then we came to the final GNF as the following –

S → mXY | mY | mXC | mC | p
X → mX | m
Y → mXD | mD | o
O → o
P → p

Note: If G is a in GNF, then for any string w belongs L(G) of length n ≥ 1, it requires exactly n steps to make any derivation of w.

2.4 PUSHDOWN AUTOMATA (PDA)

Regular language can be characterized as the language accepted by finite automata. Similarly, we can characterize the context-free language as the language accepted by a class of machines called "Pushdown Automata" (PDA). A pushdown automation is an extension of the NFA. It is observed that FA have limited capability. (in the sense that the class of languages accepted or characterized by them is small). This is due to the "finite memory" (number of states) and "no external memory" involved with them. A PDA is simply an NFA augmented with an "external stack memory". The addition of a stack provides the PDA with a last-in, first-out memory management capability. This "Stack" or "pushdown store" can be used to record a potentially unbounded information. It is due to this memory management capability with the help of the stack that a PDA can overcome the memory limitations that prevents a FA to accept many interesting languages like $\{a^n b^n \mid n \geq 0\}$. Although, a PDA can store an unbounded amount of information on the stack, its access to the information on the stack is limited. It can push an element onto the top of the stack and pop off an element from the top of the stack. To read down into the stack the top elements must be popped off and are lost. Due to this limited access to the information on the stack, a PDA still has some limitations and cannot accept some other interesting languages.



As shown in figure, a PDA has three components: an input tape with read only head, a finite control and a pushdown store. The input head is read-only and may only move from left to right, one symbol (or cell) at a time. In each step, the PDA pops the top symbol off the stack; based on this symbol, the input symbol it is currently reading, and its present state, it can push a sequence of symbols onto the stack, move its read-only head one cell (or symbol) to the right, and enter a new state, as defined by the transition rules of the PDA. PDA are nondeterministic, by default. That is, ∈-transitions are also allowed in which the PDA can pop and push, and change state without reading the next input symbol or moving its read-only head. Besides this, there may be multiple options for possible next moves.

Formal Definitions

Formally, a PDA *M* is a 7-tuple
$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$
where,

- Q is a finite set of states,
- Σ is a finite set of input symbols (input alphabets),
- Γ is a finite set of stack symbols (stack alphabets),
- δ is a transition function from $Q \times (\Sigma \cup \{\in\}) \times \Gamma$ to subset of $Q \times \Gamma$
- $q_0 \in Q$ is the start state
- $z_0 \in \Gamma$, is the initial stack symbol, and, F is the final or accept states.

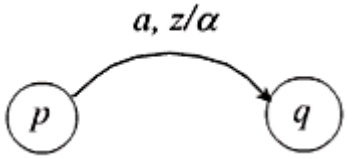
Explanation of the transition function, δ:

If, for any $a \in \Sigma$, $\delta(q,a,z) = \{(p_1,\beta_1),(p_2,\beta_2),....(p_k,\beta_k)\}$. This means institutively that whenever the PDA is in state *q* reading input symbol *a* and *z* on top of the stack, it can non deterministically for any *i*,
$1 \leq i \leq k$

- go to state p_1
- pop *z* off the stack
- push β_1 onto the stack (where $\beta_1 \in \Gamma^*$) (The usual convention is that if $\beta_1 = X_1X_2...X_k$, then X_1 will be at the top and X_k at the bottom.)
- move read head right one cell past the current symbol *a*. If *a* = $\in$, then $\delta(q, \in, z) = \{(p_1, \beta_1), (p_2, \beta_2), ...(p_k, \beta_k)\}$ means institutively that whenever the PDA is in state *q* with *z* on the top of the stack regardless of the current input symbol, it can non deterministically for any *i*, $1 \le i \le k$,
- go to state p_i
- pop *z* off the stack
- push β_i onto the stack, and
- leave its read-only head where it is.

State Transition Diagram

A PDA can also be depicted by a state transition diagram. The labels on the arcs indicate both the input and the stack operation. The transition $\delta(p, a, z) = \{(q, a)\}$ for $a \in \sum \cup \{\in\}, p, q \in Q, z \in \Gamma$ and $\alpha \epsilon \Gamma^*$ is depicted by



Final states are indicated by double circles and the start state is indicated by an arrow to it from nowhere.

Language accepted by a PDA *M*

There are two alternative definition of acceptance as given below.

1. Acceptance by final state

Consider the PDA $M = (Q, \sum, \Gamma, \delta, q_0, z_0, F)$. Informally, the PDA *M* is said to accept its input ∞ by final state if it enters any final state in zero or more moves after reading its entire input, starting in the start configuration on input x.

2. Acceptance by empty stack (or Null stack)

The PDA *M* accepts its input x by empty stack if starting in the start configuration on input x , it ever empties the stack w/o pushing anything back on after reading the entire input.

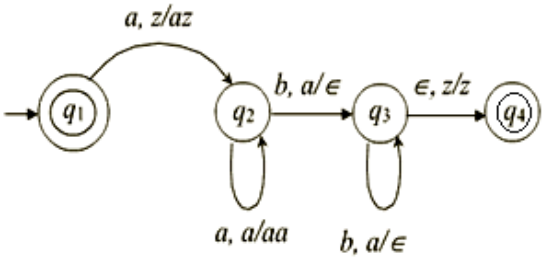
Note that the set of final states, *F* is irrelevant in this case and we usually let the *F* to be the empty set i.e. $F = \emptyset$.

Following is the PDA for language $L = \{a^n b^n \mid n>=0\}$

1. $\delta(q_1, a, z) = \{(q_2, az)\}$
2. $\delta(q_2, a, a) = \{(q_2, aa)\}$
3. $\delta(q_2, b, a) = \{(q_3, \in)\}$
4. $\delta(q_3, b, a) = \{(q_3, \in)\}$
5. $\delta(q_3, \in, z) = \{(q_4, z)\}$

Here, q_1 and q_4 are the accepting states.

The PDA can also be described by the adjacent transition diagram.



Informally, whenever the PDA *M* sees an input *a* in the start state q_1 with the start symbol *z* on the top of the stack it pushes *a* onto the stack and changes state to q_2. (to remember that it has seen the first '*a*'). On state q_2 if it sees anymore *a*, it simply pushes it onto the stack. Note that when *M* is on state q_2, the symbol on the top of the stack can only be *a*. On state q_2 if it sees the first *b* with *a* on the top of the stack, then it needs to start comparison of numbers of *a*'s and *b*'s, since all the *a*'s at the beginning of the input have already been pushed onto the stack. It start this process by popping off the

a from the top of the stack and enters in state q3 (to remember that the comparison process has begun). On state q_3 , it expects only b's in the input (if it sees any more a in the input thus the input will not be in the proper form of a^nb^n). Hence there is no more on input a when it is in state q_3. On state q_3 it pops off an a from the top of the stack for every b in the input. When it sees the last b on state q_3 (i.e. when the input is exhausted), then the last a from the stack will be popped off and the start symbol z is exposed. This is the only possible case when the input (i.e. on ∈ -input) the PDA *M* will move to state q_4 which is an accept state.

we can show the computation of the PDA on a given input using the IDs and next move relations. For example, following are the computation on two input strings.

Let the input be aabb. We start with the start configuration and proceed to the subsequent IDs using the transition function defined

$(q_1,aabb,z) \rightarrow (q_2,abb,az)$(using transition 1)

$\rightarrow \vdash (q_2, bb, aaz)$ (using transition 2)

$\rightarrow \vdash (q_3, b, az)$ (using transition 3)

$\rightarrow \vdash (q_3, \in, z)$ (using transition 4)

$\rightarrow \vdash (q_4, \in, z)$ (using transition 5)

q_4 is final state. Hence , accept. So the string *aabb* is rightly accepted by *M*. We can show the computation of the PDA on a given input using the IDs and next move relations. For example, following are the computation on two input strings.

Let the input be aabab.

$(q_1,aabab,z) \rightarrow (q_2,abab,az)$

$\rightarrow \vdash (q_2, bab, aaz)$

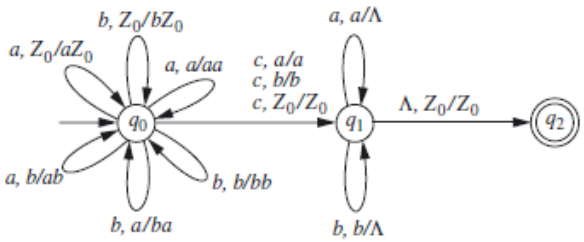
$\rightarrow \vdash (q_3, ab, az)$

No further move is defined at this point. Hence the PDA gets stuck and the string *aabab* is not accepted.

More Examples:

L1 = {wcw^r|w∈{a,b}*}

Sr	State	I/P	Stack	Move
1	q_o	a	z0	(q_o,az0)
2	q_o	b	z0	(q_o,bz0)
3	q_o	a	a	(q_o,aa)
4	q_o	b	a	(q_o,ba)
5	q_o	a	b	(q_o,ab)
6	q_o	b	b	(q_o,bb)
7	q_o	c	z0	(q_1,z0)
8	q_o	c	a	(q_1,a)
9	q_o	c	b	(q_1,b)
10	q_1	a	a	(q_1,^)
11	q_1	b	b	(q_1,^)
12	q_1	^	z0	(q_2,z0)



L2 = {x | x is Palindrome String over {a,b}*}

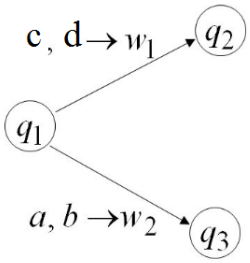
Sr	State	I/P	Stack	Move
1	q_o	a	z0	(q_o,az0) (q_1,z0)
2	q_o	b	z0	(q_o,bz0) (q_1,z0)
3	q_o	a	a	(q_o,aa) (q_1,a)
4	q_o	b	a	(q_o,ba) (q_1,a)
5	q_o	a	b	(q_o,ab) (q_1,b)
6	q_o	b	b	(q_o,bb) (q_1,b)
7	q_o	c	z0	(q_1,z0)
8	q_o	c	a	(q_1,a)
9	q_o	c	b	(q_1,b)
10	q_1	a	a	(q_1,^)
11	q_1	b	b	(q_1,^)
12	q_1	^	z0	(q_2,z0)

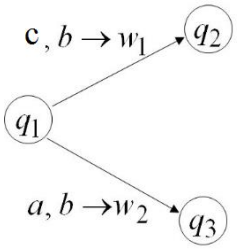
L3 = {String of balanced parenthesis over [}

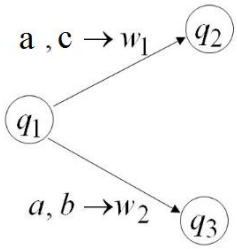
Sr	State	I/P	Stack	Move
1	q_0	[	z0	(q_1, [z0)
2	q_1	[	[	(q_1, [[)
3	q_1	]	[	(q_1, ^)
4	q_1	^	z0	(q_1, z0)

2.5 DETERMINISTIC AND NON-DETERMINISTIC PDA

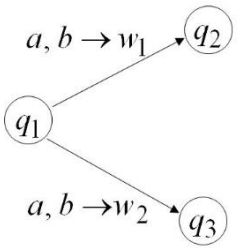
Deterministic Moves/Choices:

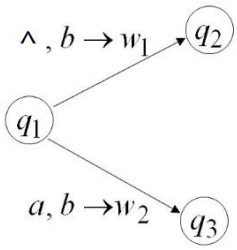






Non-Deterministic Moves/Choices





Deterministic PDA (DPDA):
If all moves in PDA are deterministic then it is called deterministic PDA.

Non-Deterministic PDA (NPDA):
If any move of PDA is non-deterministic then it is called non-deterministic PDA.

Example:
In the previous section the PDA for language L1 and L3 was deterministic but PDA for L2 was non-deterministic.

Power of DPDA and NPDA in terms of language acceptance:
For every CFL there exist a non-deterministic PDA or simply PDA. But for every CFL deterministic PDA does not exist. It means there are few CFL for which DPDA can't be constructed but NPDA can be.

For example:
L = {x | x is Palindrome String over {a,b}*}

For above language we have constructed NPDA in the previous section but its DPDA does not exist.

Note: For this reason NPDA is said to be more powerful than DPDA.

2.6 EQUIVALENCE OF PDAS AND CFGS

If P is a pushdown automaton, an equivalent context-free grammar G can be constructed where
L(G) = L(P)

Steps to find PDA corresponding to a given CFG:
Input – A CFG, G = (V, T, P, S)
Output – Equivalent PDA, P = (Q, ∑, S, δ, q0, I, F)
Step 1 – Convert the productions of the CFG into GNF.

Step 2 – The PDA will have only one state {q}.

Step 3 – The start symbol of CFG will be the start symbol in the PDA.

Step 4 – All non-terminals of the CFG will be the stack symbols of the PDA and all the terminals of the CFG will be the input symbols of the PDA.

Step 5 – For each production in the form A → aX where a is terminal and A, X are combination of terminal and non-terminals, make a transition δ (q, a, A).

Example:
Construct a PDA from the following CFG.
G = ({S, X}, {a, b}, P, S) where the productions are –
S → XS | ε , A → aXb | Ab | ab

Solution
Let the equivalent PDA,
P = ({q}, {a, b}, {a, b, X, S}, δ, q, S)
where δ –
δ(q, ε , S) = {(q, XS), (q, ε)}
δ(q, ε , X) = {(q, aXb), (q, Xb), (q, ab)}
δ(q, a, a) = {(q, ε)}
δ(q, 1, 1) = {(q, ε)}

2.7 CONTEXT FREE LANGUAGES

A language is said to be context free if there exist a PDA which accepts it.

Informally, PDA can remember one condition/comparison but it can't remember more than one conditions simultaneously.

For example:
$L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ is context free but $L = \{a^i b^j c^k \mid i = j \text{ and } j = k\}$ is not context free.

Following is the list of context free languages:

- $\{x \in \{a, b\}* \mid na(x) < nb(x)\}$
- $\{x \in \{a, b\}* \mid na(x) \neq nb(x)\}$
- $\{x \in \{a, b\}* \mid na(x) = 2nb(x)\}$
- $\{a^n b^{n+m} a^m \mid n,m \geq 0\}$
- The set of even-length palindromes over {a, b}
- The set of odd-length palindromes over {a, b}
- $\{x \in \{a, b\}* \mid na(x) = nb(x)\}$
- $\{x \in \{0, 1\}* \mid na(x) = 2nb(x)\}$

Following is the list of non context free languages:

- $L = \{a^i b^j c^k \mid i < j < k\}$
- $L = \{x \in \{a, b\}* \mid nb(x) = na(x)^2$
- $L = \{a^n b^{2n} a^n \mid n \geq 0\}$
- $L = \{x \in \{a, b, c\}* \mid na(x) = \max\{nb(x), nc(x)\}\}$
- $L = \{x \in \{a, b, c\}* \mid na(x) = \min\{nb(x), nc(x)\}\}$
- $\{a^n b^m a^n b^{n+m} \mid m, n \geq 0\}$

Deterministic Context Free Language (DCFL)
A CFL is said to be DCFL if there exist DPDA which accepts it.

2.8 CLOSURE PROPERTIES OF CFL

Context-free languages are closed under union, concatenation, and Kleene star but not under intersection and complementation.

Suppose G1 = (V1, Σ1, R1, S1) and G2 = (V2, Σ2, R2, S2).

Example: For G1 we have $S_1 \rightarrow aS_1b$, $S_1 \rightarrow ^$.
For G2 we have $S_2 \rightarrow c\,S_2d$, $S_2 \rightarrow ^$.
Then $L(G1) = \{a^n b^n : n \geq 0\}$.
Also, $L(G2) = \{c^n d^n : n \geq 0\}$

Union:
G = (V1 ∪ V2 ∪ {S}, Σ1 ∪ Σ2, R, S) where
R = R1 ∪ R2 ∪ {S → S1, S → S2} and S is a new symbol.
Then L(G) = L(G1) ∪ L(G2).
Example:
$S_1 \rightarrow a\,S_1b$, $S_1 \rightarrow ^$. $S_2 \rightarrow c\,S_2d$ $S_2 \rightarrow ^$.
$S \rightarrow S_1 \mid S_2$
Then $L(G) = \{a^n\, b^n : n \geq 0\} \cup \{c^n d^n : n \geq 0\}$.

Concatenation
G = (V1 ∪ V2 ∪ {S}, Σ1 ∪ Σ2, R, S) where
R = R1 ∪ R2 ∪ {S → S1S2} and S is a new symbol. Example:
$S_1 \rightarrow aS_1b$, $S_1 \rightarrow ^$. $S_2 \rightarrow c\,S_2d$ $S_2 \rightarrow ^$.
$S \rightarrow S_1S_2$
Then $L(G) = \{a^m b^m c^n d^n : m, n \geq 0\}$.

Kleene Star
G = (V1 ∪ {S}, Σ1, R, S) where R = R1 ∪ {S → ^, S → SS1} and S is a new symbol.
Example:
$S_1 \rightarrow aS_1b$ $S1 \rightarrow ^$.
$S \rightarrow ^$, $S \rightarrow SS1$

Then L(G) = $\{a^n b^n : n \geq 0\}*$.

Not Closed under Intersection

Let L1 = $\{a^m b^m c^n : m, n \geq 0\}$ and
let L2 = $\{a^m b^n c^n : m, n \geq 0\}$.

- Then both L1 and L2 are context-free.
- However, their intersection L1 ∩ L2 is $\{a^n b^n c^n : n \geq 0\}$ which is not context-free.

Not Closed under Complement

Let L1 = $\{a^i b^j c^k \mid i \neq j$ or $j \neq k$ or $i \neq k\}$ U (a+b+c)*(ba+cb+ca+cba)(a+b+c)* Here, L1 is context-free.
But its complement
L1' = $\{a^n b^n c^n : n \geq 0\}$ is not Context Free.

2.9 THINGS TO REMEMBER:

- NPDA is more powerful than DPDA.
- Every DCFL is CFL.
- Every regular language is DCFL hence CFL.
- If the language is defined over single symbol then it is CFL if it is regular.

GATE QUESTIONS

Q.1 Which of the following statements is true?

a) If a language is context-free it can always be accepted by a deterministic push- down automaton
b) The union of two context-free languages is context-free
c) The intersection of two context-free languages is context-free
d) The complement of a context-free language is context-free

[GATE-2001]

Q.2 The language accepted by a push-down automaton is limited to 10 items is best described as

a) context-free
b) regular
c) deterministic context-free
d) recursive

[GATE-2002]

Q.3 Let $G = (\{S\}, \{a, b\}\ R, S)$ be a context-free grammar where the rule set R is $S \rightarrow aSb|SS|\varepsilon$

Which of the following statements is true?

a) G is not ambiguous
b) There exist $X, Y \in L(G)$ such that $xy \notin L(G)$
c) There is a deterministic push-down automaton that accepts $L(G)$
d) We can find a deterministic finite state automaton that accepts $L(G)$

[GATE-2003]

Q.4 Which one of the following statements is FALSE?

a) There exist context free languages such that all the context free grammars generating them are ambiguous
b) An unambiguous context-free grammar always has a unique parse tree for each string of the language generated by it
c) Both deterministic and non-deterministic pushdown automata always accept the same set of language
d) A finite set of strings from some alphabet is always a regular language

[GATE-2004 (IT)]

Q.5 Let $M = (K, \Sigma, \Gamma, \Delta, s, F)$ be a pushdown automaton, where

$K = (s, f)\ F = \{f\}, \Sigma = \{a, b\}, \Gamma = \{a\}$

and $\Delta = \{((s, a, \varepsilon),(s, a)), ((s, b, \varepsilon)$

$(s, a)), ((s, a, \varepsilon),(f, \varepsilon)), ((f, a, a),$

$(f, \varepsilon)), ((f, b, a), (f, \varepsilon))\}$

Which one of the following strings is not a member of L(M)?

a) aaa b) aabab
c)baaba d) bab

[GATE-2004 (IT)]

Q.6 Consider the following grammar G.

$S \rightarrow bS \mid aA \mid b$

$A \rightarrow bA \mid aB$

$B \rightarrow bB \mid aS \mid a$

Let N_a (W) and N_b (W) denote the number of a's and b's in a string W respectively.

The language $L(G) \subseteq \{a, b\}^+$ generated by G is

a) $\{W \mid N_a(W) > 3N_b(W)\}$
b) $\{W \mid N_b(W) > 3N_a(W)\}$
c) $\{W \mid N_a(W) = 3k,\ k(0, 1, 2, ..)\}$
d) $\{W \mid N_b(W) = 3k,\ k \in (0,1, 2, ..)\}$

[GATE-2004]

Q.7 The language $\{a^m b^n c^{m+n} \mid m, n \geq 1\}$ is

a) regular
b) context-free but not regular
c) Context-sensitive but not context-free
d) type-0 but not context-sensitive

[GATE-2004]

Q.8 Let L be a regular language and M be a context-free language, both over the alphabet Σ. Let L^c and M^c denote the complements of L and M respectively.
Which of the following statements about the language $L^c \cup M^c$ is TRUE?

a) It is necessarily regular but not necessarily context-free.
b) It is necessarily context-free.
c) It is necessarily non-regular.
d) None of the above

[GATE-2005 (IT)]

Q.9 Let P be a non-deterministic push-down automation (N PDA) with exactly one state, q, and exactly one symbol, Z, in its stack alphabet. State q is both the starting as well as the accepting state of the PDA. The stack is initialized with one Z before the start of the operation of the PDA. Let the input alphabet of the PDA be Σ.
Let $L(P)$ be the language accepted by the PDA by reading a string and reaching its accepting state. Let $N(P)$ be the language accepted by the PDA by reading a string and emptying its stack.
Which of the following statements is TRUE?

a) $L(P)$ is necessarily Σ^* but $N(P)$ is not necessarily Σ^*.
b) $N(P)$ is necessarily Σ^* but $L(P)$ is not necessarily Σ^*
c) Both $L(P)$ & $N(P)$ is necessarily Σ^*
d) Neither $L(P)$ nor $N(P)$ are necessarily Σ^*.

[GATE-2005 (IT)]

Q.10 Consider the languages :
$L_1 = \{WW^R \mid W \in (0, 1)^*\}$
$L2 = \{W \# W^R \mid W \in \{0,1)^*\}$ where # is a special symbol
$L_3 = \{WW \mid W \in \{0, 1\}^*\}$ Which one of the following is true?

a) L_1 is a deterministic CFL
b) L_2 is a deterministic CFL
c) L_3 is a CFI but not a deterministic CFL
d) L_3 is a deterministic CFL

[GATE-2005]

Q.11 Consider the languages :
$L_1 = \{a^n b^n c^m \mid n, m > 0\}$ and $L_2 = \{a^n b^m c^m \mid n, m > 0\}$
Which one of the following statements is false?

a) $L_1 \cap L_2$ is a context-free language
b) $L_1 \cup L_2$ is a context-free language
c) L_1 and L_2 arecontext-free languages
d) $L_1 \cap L_2$ is a context-sensitive language

[GATE-2005]

Q.12 Let N_f and N_p, denote the classes of languages accepted by non-deterministic finite automata and non - deterministic push - down automata, respectively. Let D_f and D_p denote the classes of languages accepted by deterministic finite automata and deterministic push-down automata respectively.
Which one of the following is true?

a) $D_f \subset N_f$ and $D_p \subset N_p$
b) $D_f \subset N_f$ and $D_p = N_p$
c) $D_f = N_f$ and $D_p = N_p$
d) $D_f = N_f$ and $D_p \subset N_p$

[GATE-2005]

Q.13 In the context-free grammar below, S is the start symbol, a and b are terminals, and ε denotes the empty string $S \rightarrow aSa|bSb|a|b|\varepsilon$

Which of the following string is NOT generated by the grammar?

a) aaaa b) baba

c) abba d) babaaabab

[GATE-2006 (IT)]

Q.14 Which of the following languages is accepted by a non-deterministic pushdown automaton (PDA) but NOT by deterministic PDA?

a) $\{a^n b^n c^n | n \geq 0\}$

b) $\{a^l b^m c^n | l \neq m \text{ or } m \neq n\}$

c) $\{a^n b^n | n \geq 0\}$

d) $\{a^m b^n | m, n \geq 0\}$

[GATE-2006 (IT)]

Q.15 Let L be a context-free language and M a regular language. Then the language $L \cap M$ is

a) always regular

b) never regular

c) always a deterministic context-free language

d) always a context-free language

[GATE-2006 (IT)]

Q.16 Consider the pushdown automaton (PDA) below which runs over the input alphabet (a,b,c). It has the stack alphabet $\{Z_0, X\}$ where Z_0 is the bottom –of-stack marker. The set of states of the PDA is {s, t, u, f} where s is the start state and f is the final state. The PDA accepts by final state. The transitions of the PDA given below are depicted in a standard manner. For example, the transition $(s, b, X) \rightarrow$ (t, XZ_0) means that if the PDA is in state s and the symbol on the top o f the stack is X, then it can read b from the input and move to state t after propping the top of stack and pushing the symbols Z_0 and X (in that order) on the stack.

$(s, a, Z_0) \rightarrow (s, XXZ_0)$

$(s, \varepsilon, Z_0) \rightarrow (f, \varepsilon)$

$(s, a, X) \rightarrow (s, XXX)$

$(s, b, X) \rightarrow (t, \varepsilon)$

$(t, b, X) \rightarrow (t, \varepsilon)$

$(t, c, X) \rightarrow (u, \varepsilon)$

$(u, c, X) \rightarrow (u, \varepsilon)$

$(u, \varepsilon, Z_0) \rightarrow (f, \varepsilon)$

The language accepted by the PDA is

a) $\{a^l b^m c^n | l = m = n\}$

b) $\{a^l b^m c^n | l = m\}$

c) $\{a^l b^m c^n | 2\,l = m + n\}$

d) $\{a^l b^m c^n | m = n\}$

[GATE-2006 (IT)]

Q.17 In the context-free grammar below, S is the start symbol, a and b are terminal, and ε denotes the empty string.

$S \rightarrow aSAb|\varepsilon$

$A \rightarrow bA|\varepsilon$

The grammar generates the language

a) $((a+b)^*b)^*$ b) $\{a^m b^n | m \leq n\}$

c) $\{a^m b^n | m = n\}$ d) a^*b^*

[GATE-2006 (IT)]

Q.18 Let $L_1 = \{0^{n+m}1^n 0^m | n, m \geq 0\}$,

$L_2 = \{0^{n+m}1^{n+m}0^m | n, m \geq 0\}$, and

$L_3 = \{0^{n+m}1^{n+m}0^{n+m} | n, m \geq 0\}$.

Which of these languages is/are not context?

a) L_1 only b) L_3 only

c) L_1 and L_2 d) L_2 and L_3

[GATE-2006]

Q.19 Consider the following statements about the context-free grammar

$G = \{S \rightarrow SS, S \rightarrow ab, S \rightarrow ba, S \rightarrow \varepsilon\}$

1. G is ambiguous.
2. G produces all strings with equal number of a's and b's
3. G can be accepted by a deterministic PDA.

Which combination below expresses all the true statements about G?

a) 1 only b) 1 and 3
c) 2 and 3 d) 1, 2 and 3

[GATE-2006]

Q.20 Consider an ambiguous grammar G and its disambiguated version D. Let the languages recognized by the two grammars be denoted b y L(G) and L(D)respectively. Which one of the following is true?

a) $L(D) \subset L(G)$ b) $L(D) \supset L(G)$
c) $L(D) = L(G)$ d) L(D) is empty

[GATE-2007]

Q.21 Consider the following grammars, Names representing terminals have been specified in capital letters.

G1: stmnt $\rightarrow$ WHILE (expr) stmnt
stmnt $\rightarrow$ OTHER
expr $\rightarrow$ ID

G2: stmnt $\rightarrow$ WHILE (expr) stmnt
stmnt $\rightarrow$ OTHER
expr $\rightarrow$ expr + expr
expr $\rightarrow$ exper * expr
expr $\rightarrow$ expr * expr
expr $\rightarrow$ ID

Which one of the following statements is true?

a) G_1 is context-free but not regular and G_2 is regular
b) G_2 is context-free b ut not regular and G_1
c) Both G_1 and G_2 are regular
d) Both G_1 and G_2 are context-free but neither of them is regular

[GATE-2007(IT)]

Q.22 The language $L = \{0^i 21^i \mid i \geq 0\}$ over the alphabet {0, 1, 2) s

a) not recursive
b) is recursive & is a deterministic CFL
c) is a regular language
d) is not a deterministic CFL but a CFL

[GATE-2007(IT)]

Q.23 Which of the following problems is undecidable?

a) Membership problem for CFGs
b) Ambiguity problem for CFGs
c) Finiteness problem for FSAs
d) Equivalence problem for FSAs

[GATE-2007]

Q.24 Consider the following languages.

$L_1 = \{a^i b^j c^k \mid i = j,\ k \geq 1\}$

$L_2 = \{a^i b^j \mid j = 2i,\ i \geq 0\}$

Which of the following if true?

a) L_1 is not a CFL but L_2 is
b) $L_1 \cap L_2 = \phi$ and L_1 is non-regular
c) $L_1 \cap L_2$ is not a CFL but L_2 is
d) There is a 4 state PDA that accepts L_1, but there is no DPDA that accepts L_2

[GATE-2008(IT)]

Q.25 Consider CFG with the following productions.

$S \rightarrow AA \mid B$

$A \rightarrow 0A \mid A0 \mid 1$

$B \rightarrow 0B00 \mid 1$

S is the start symbol, A and B are non-terminals and 0 and 1 are the terminals. The language generated by this grammar is

a) $\{0^n 10^{2n} \mid n \geq 1\}$

b) $\{0^i\ 10^j\ 10^k \mid i, j, k \geq 0\} \cup \{0^n 10^{2n} \mid n \geq 1\}$ c)

$\{0^i\ 10^j \mid i, j, \geq 0\} \cup \{0^n 10^{2n} \mid n \geq 1\}$

d) The set of all strings over {0, 1} containing at least two 0's

[GATE-2008(IT)]

Directions for Question Q.26 to Q.27:
A CFG G is given with the following productions where S is the start symbol, A is a non-terminal and a and b are terminals.

$S \rightarrow aS|A$

$A \rightarrow aAb|bAa|\varepsilon$

Q.26 Which of the following strings is generated by the grammar above?
a) aabbaba b) aabaaba
c) abababb d) aabbaab
[GATE-2008(IT)]

Q.27 For the correct answer in above Question, how many steps are required to derive the string and how many parse trees are there?
a) 6 and 1 b) 6 and 2
c) 7 and 2 d) 4 and 2
[GATE-2008(IT)]

Q.28 Which of the following are decidable?
1. Whether the intersection of two regular languages is infinite.
2. Whether a given context-free language is regular.
3. Whether two push-down automata accept the same language.
4. Whether a given grammar is context-free.

a) 1 and 2 b) 1 and 4
c) 2 and 3 d) 2 and 4
[GATE-2008]

Q.29 Which of the following is true for language $\{a^p \mid p \text{ is a prime}\}$?
a) it is not accepted by Turing machine
b) it is regular but not context-free
c) it is context-free but not regular
d) it is neither regular nor context-free, but accepted by a Turing machine
[GATE-2008]

Q.30 Match List I with List II and select the correct answer using the codes given below the lists

List I	List II
E. Checking that identifiers are declared before their use	P.L = {anbmcndm | n ≥ 1, m ≥ 1|}
F. number of formal parameters in the declaration of function agrees with the number of actual parameters in use of that function	Q. $X \rightarrow XbX \mid XcX \mid dXf \mid g$
G.Arithmetic expressions with matched pairs of parameters	R.L={WcW | W∈ (a|b)*}
H. Palindromes	S.X→bXb | cXc |ε

a) E-P, F-R, G-Q, H-S
b) E-R, F-P, G-S, H-Q
c) E-R, F-P, G-Q, H-S
d) E-P, F-R, G-S, H-Q
[GATE-2008]

Q.31 Which of the following statements are true?
1. Every left-recursive grammar can be converted to a right-recursive grammar and vice-versa.
2. All ε -productions can be removed from any context-free grammar by suitable transformations.
3. The language generated by a context-free grammar all of whose productions are of the form X→W or X→WY (where, W is a string of terminals and Y is a non-terminal), is always regular.
4. The derivation trees of strings generated by a context free grammar in Chomsky Normal Form are always binary trees.

a) 1, 2, 3 & 4 b) 2, 3 & 4
c) 1, 3 & 4 d) 1, 2 & 4

[GATE-2008]

Q.32 Which of the following statements is false?

a) Every NFA can be converted to an equivalent DFA
b) Every non-deterministic Turing machine can be converted to an equivalent deterministic Wring machine
c) Every regular language is also a context-free language
d) Every subset of a recursively enumerable set is recursive

[GATE-2008]

Q.33 Which one of the following is FALSE?

a) There is a unique minimal DFA for every regular language
b) Every NFA can be converted to an equivalent PDA
c) Complement of every context-free language is recursive
d) Every nondeterministic PDA can be converted to an equivalent deterministic PDA

[GATE-2009]

Q.34 Given the following state table of an FSM with two states A and B, one input and one output:

Present State A	Present State B	Input	Next State A	Next Stat B	Output
0	0	0	0	0	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	1	0	0
0	0	1	0	1	0
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	0	0	1

If the initial state is A = 0, B = 0, what is the minimum length of an input string which will take the machine to the state A =0, B =1 with Output =1?

a) 3 b) 4
c) 5 d) 6

[GATE-2009]

Q.35 Let $L_1 = L_1 \cap L_2$, where L_1 and L_2 are languages as defined below

$L_1 = \{a^m b^m c\, a^n b^n \mid m, n \geq 0\}$

$L_2 = \{a^i b^j c^k \mid i, j, k \geq 0\}$

Then L is

a) not recursive
b) regular
c) context-free but not regular
d) recursively enumerable but not context-free

[GATE-2009]

Q.36 Match all items in List I with correct options from those given in List II.

	List-I		List-II
P.	Regular expression	1.	Syntax analysis
Q.	Push-down automata	2.	Code generation
R.	Dataflow analysis	3.	Lexical analysis
S.	Register allocation	4.	Code optimization

a) P-4, Q-1, R-2, S-3
b) P-3, Q-1, R-4, S-2
c) P-3, Q-4, R-1, S-2
d) P-2, Q-1, R-4, S-3

[GATE-2009]

Q.37 Which one of the following is false?

a) There is unique minimal DFA for every regular language
b) Every NFA can be converted to an equivalent PDA.
c) Complement of every context-free language is recursive
d) Every non-deterministic PDA can be converted to an equivalent deterministic PDA

[GATE-2009]

Q.38 S→aSa|bSb|a|b; the language generated by the above grammar over the alphabet {a, b} is the set of

a) All palindromes
b) All odd length palindromes
c) Strings that begin and end with the same symbol
d) All even length palindromes

[GATE-2009]

Q.39 Consider the languages $L_1 = \{0^i1^j \mid i <> j\}$, $L_2 = \{0^i1^j \mid i = j\}$, $L_3 = \{0^i1^j \mid i = 2j+1\}$, $L_4 = \{0^i1^j \mid i \neq 2j\}$. Which one of the following statements is true?

a) Only L_2 is context-free
b) L_2 and L_3 are Context-free
c) L_1 and L_2 are context-free
d) All are context-free

[GATE-2010]

Q.40 Consider the languages L_1, L_2 and L_3 as given below

$L_1 = \{0^p1^q \mid p, q \in N\}$

$L_2 = \{0^p 1^q \mid p, q \in N \text{ and } p= q\}$ and

$L_3 = \{0^p1^q0^r \mid p, q\ r \in N \text{ and } p=q=r\}$

Which of the following statements is not true?

a) Push Down Automata (PDA) can be used to recognize L_1 and L_2
b) L_1 is a regular language
c) All the three languages are context-free
d) Turing machines can be used to recognize all the languages

[GATE-2011]

Q.41 Consider the following languages.

$L_1 = \{0^p1^q0^r \mid p,q,r \geq 0\}$

$L_2 = \{0^p1^q0^r \mid p,q,r >=0 \text{ and } p \neq r\}$

Which one of the following statements is false?

a) L_2 is context-free
b) $L_1 \cap L_2$ is context-free
c) Complement of L_2 is recursive
d) Complement of L_1 is context-free but not regular

[GATE-2013]

Q.42 Consider the following languages over the alphabet $\sum=\{0,1,c\}$

$L1 = \{0^n1^n \mid n \geq 0\}$

$L2 = \{wcw^r \mid w \in \{0,1\}^*\}$

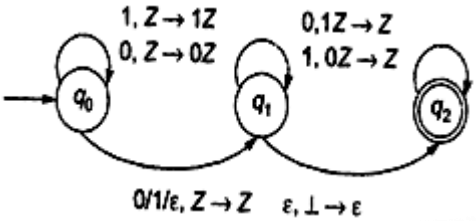
$L3 = \{ww^r \mid w \in \{0,1\}^*\}$

Here, w^r is the reverse of the string w. Which of these languages are deterministic context free languages?

a) None of the languages
b) Only L_1
c) Only L_1 and L_2
d) All the three languages

[GATE-2014]

Q.43 Consider the NPDA $\langle Q = \{q_0, q_1, q_2\}, \sum = \{0,1\}, \Gamma = \{0,1,\perp\}, \delta, q_0, \perp, F = \{q_2\}\rangle$, where (as per usual convention)Q is the set of states, Σ is the input alphabet, Γ is stack alphabet, δ is the state transition function, q_0 is the initial state, ⊥ is the initial stack symbol, and F is the set of accepting states, The state transition is as follows:



Which one of the following sequences must follow the string 101100 so that the overall string is accepted by the automaton?

(a) 10110 (b) 10010
(c) 01010 (d) 01001

[GATE-2015(1)]

Q.44 Which of the following languages are context-free?

$L_1 = \{a^mb^na^nb^m \mid m,n>=1\}$

$L_2 = \{a^mb^na^mb^n \mid m,n>=1\}$

$L_3 = \{a^mb^n \mid m = 2n+1\}$

a) L1 and L2 only
b) L1 and L3 only
c) L2 and L3 only
d) L3 only

[GATE-2015]

Q.45 Language L_1 is defined by the grammar:

$S_1 \rightarrow aS_1b|\varepsilon$

Language L_2 is defined by the grammar:

$S_2 \rightarrow abS_2|\varepsilon$

Consider the following statements:

P : L_1 is regular

Q : L_2 is regular

Which one of the following is TRUE?

a) Both P and Q are true

b) P is true and Q is false

c) P is false and Q is true

d) Both P and Q are false

[GATE-2016(2)]

Q.46 Which of the following languages is generated by the given grammar?

$S \rightarrow aS\ |bS|\ \varepsilon$

a) $\{a^nb^m\ |n,m \geq 0\}$

b) $\{w \in \{a,b\}^*\ |$ w has equal number of a's and b's$\}$

c) $\{a^n\ |\ n \geq 0\} \cup \{b^n\ |\ n \geq 0\} \cup \{a^nb^n\ |\ n \geq 0\}$

d) $\{a, b\}^*$

[GATE-2016]

Q.47 Consider the following context-free grammars:

G_1: $S \rightarrow aS|B,\ B \rightarrow b|bB$

G_2: $S \rightarrow aA|bB,\ A \rightarrow aA|B|\varepsilon,\ B \rightarrow bB|\varepsilon$

Which one of the following pairs of languages is generated by G_1 and G_2, respectively?

a) $\{a^mb^n|m>0$ or $n > 0\}$ and $\{a^mb^n|m > 0$ and $n > 0\}$

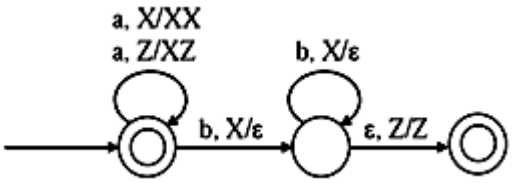
b) $\{a^mb^n|m>0$ and $n>0\}$ and $\{a^mb^n|m > 0$ or $n\geq 0\}$

c) $\{a^mb^n|m\geq 0$ or $n > 0\}$ and $\{a^mb^n|m > 0$ and $n > 0\}$

d) $\{a^mb^n|m\geq 0$ and $n>0\}$ and $\{a^mb^n|m>0$ or $n>0\}$

[GATE-2016]

Q.48 Consider the transition diagram of a PDA given below with input alphabet $\Sigma = \{a, b\}$ and stack alphabet $\Gamma = \{X,Z\}$. Z is the initial stack symbol. Let L denote the language accepted by the PDA.



Which one of the following is **TRUE**?

a) L $=\{a^nb^n\ |\ n \geq 0\}$ and is not accepted by any finite automata

b) L $\{a^n|n \geq 0\} \cup \{a^nb^n\ |n\geq 0\}$ and is not accepted by any deterministic PDA

c) L is not accepted by any Turning machine that halts on every input

d) L $\{a^n\ |\ n \geq 0\} \cup \{a^nb^n\ |\ n \geq 0\}$ and is deterministic context-free

[GATE-2016]

Q.49 Consider the following languages:

$L1 = \{a^nb^mc^{n+m} : m, n \geq 1\}$

$L2 = \{a^nb^nc^{2n} : n \geq 1\}$

Which one of the following is TRUE?

a) Both L1 and L2 are context-free.

b) L1 is context-free while L2 is not context-free.

c) L2 is context-free while L1 is not context-free.

d) Neither L1 nor L2 is context-free

[GATE-2016]

Q.50 Which one of the following grammars is free from left recursion?

a) $S \rightarrow AB$
$A \rightarrow Aa\ |\ b$
$B \rightarrow c$

b) $S \rightarrow Ab|\ Bb\ |c$
$A \rightarrow Bd\ |\ \varepsilon$
$B \rightarrow e$

c) $S \rightarrow Aa\ |\ B$
$A \rightarrow Bb\ |\ Sc\ |\ \varepsilon$
$B \rightarrow d$

d) $S \rightarrow Aa|Bb|c$
$A \rightarrow Bd\ |\ \varepsilon$
$B \rightarrow Ae\ |\ \varepsilon$

[GATE-2016]

Q.51 Identify the language generated by the following grammar, where S is the start variable.

$S \rightarrow XY$

$X \rightarrow aX|a$

$Y \rightarrow aYb|\in$

a) $\{a^mb^n\ |\ m \geq n, n>0\}$

b) $\{a^mb^n\ |\ m \geq n, n \geq 0\}$

c) $\{a^m b^n \mid m>n, n \geq 0\}$

d) $\{a^m b^n \mid m>n, n>0\}$

[GATE-2017(2)]

Q.52 Let L_1, L_2 be any two context-free languages and R be any regular language. Then which of the following is/are CORRECT ?

I. $L_1 \cup L_2$ is context-free .

II. $\overline{L_1}$ is context –free.

III. $L_1 - R$ is context –free.

IV. $L_1 \cap L_2$ is context –free.

a) I, II and IV only b) I and III only

c) II and IV only d) I only

[GATE-2017(2)]

Q.53 Consider the following languages.

$L_1 = \{a^p \mid p$ is a prime numder$\}$

$L_2 = \{a^n b^m c^{2m} \mid n \geq 0, m \geq 0\}$

$L_3 = \{a^n b^n c^{2n} \mid \geq 0\}$

$L_4 = \{a^n b^n \mid n \geq 1\}$

Which of the following are CORRECT ?

I. L_1 is context –free but not regular.

II. L_2 is not context –free.

III. L_3 is not context–free but recursive.

IV. L_4 is deterministic context –free.

a) I,II and IV only

b)II and III only

c) I and IV only

d) III and IV only

[GATE-2017(2)]

Q.54 Let δ denote the transition function and $\hat{\delta}$ denote the extended transition function of the ε-NFA whose transition table is given below:

δ	ϵ	a	b
→q_0	{q_2}	{q_1}	{q_0}
q_1	{q_2}	{q_2}	{q_3}
q_2	{q_0}	Φ	Φ
q_3	Φ	Φ	{q_3}

The $\hat{\delta}(q_2, aba)$ is

a) ϕ b) (q_0, q_1, q_3)

c) (q_0, q_1, q_2) d) (q_0, q_2, q_3)

[GATE-2017(2)]

Q.55 Consider the following context-free grammar over the alphabet $\sum = \{a, b, c\}$ with S as the start symbol:

S -> abScT | abcT

T -> bT|b

Which one of the following represents the language generated by the above grammar?

a) $\{(ab)^n (cb)^n \mid n >= 1\}$

b) $\{(ab^n cb^{m_1} cb^{m_2} \ldots cb^{m_n} \mid n, m_1, m_2, \ldots, m_n >= 1\}$

c) $\{(ab)^n (cb^m)^n \mid n >= 1\}$

d) $\{(ab)^n (cb^n)^m \mid m, n >= 1\}$

[GATE-2017(1)]

Q.56 Consider the context-free grammars over the alphabet {a,b,c} given below. S and T are non- terminals.

$G_1 : S \rightarrow aSb \mid T, T \rightarrow cT \mid \in$

$G_2 : S \rightarrow bSa \mid T, T \rightarrow cT \mid \in$

The language $L(G_1) \cap L(G_2)$ is

a) Finite

b) Not finite but regular

c) Context- Free but not regular

d) Recursive but not context - free

[GATE-2017(1)]

Q.57 If G is a grammar with productions

$S \rightarrow SaS \mid aSb \mid bSa \mid SS \mid \in$

Where S is the start variable , then which one of the following strings is not generated by G?

a) abab b) aaab

c) abbaa d) babba

[GATE-2017(1)]

Q.58 Consider the following languages over the alphabet $\sum' = \{a, b, c\}$. Let

$L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$ and

$L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$ Which of the

following are context – free languages?

I. $L_1 \cup L_2$

II. $L_1 \cap L_2$

a) I only b) II only

c) I and II d) Neither I nor II

[GATE-2017(1)]

Q.59 Consider the following languages:

I. $\{a^m b^n c^p d^q \mid m+p = n+q$, where $m,n,p,q >= 0\}$

II. $\{a^m b^n c^p d^q \mid m=n$ and $p=q$, where $m,n,p,q >= 0\}$

III. $\{a^m b^n c^p d^q \mid m=n=p$ and $p!=q$, where $m,n,p,q >= 0\}$

IV. $\{a^m b^n c^p d^q \mid mn = p+q$, where $m,n,p,q >= 0\}$

Which of the languages above are context-free?

a) I and IV only

b) I and II only

c) II and III only

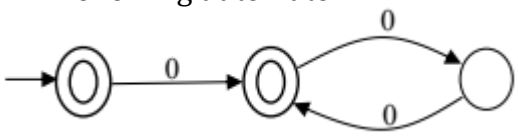
d) II and IV only

[GATE-2018]

Q.60 Given a language L, define L^i as follows:

$$L^0 = \{\varepsilon\}$$
$$L^i = L^{i-1} \cdot L \quad for\ all\ i > 0$$

The order of a language L is defined as the smallest k such that $L^k = L^{k+1}$. Consider the language L1 (over alphabet 0) accepted by the following automaton.



The order of L1 is ____.

[GATE-2018]

ANSWER KEY:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
(b)	(b)	(c)	(c)	(b)	(c)	(b)	(d)	(d)	(b)	(a)	(d)	(b)	(b)	(d)
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
(c)	(b)	(d)	(b)	(c)	(d)	(b)	(b)	(b)	(b)	(d)	(a)	(b)	(d)	(c)
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
(c)	(d)	(d)	(a)	(c)	(b)	(d)	(b)	(d)	(c)	(d)	(c)	(b)	(b)	(c)
46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
(d)	(d)	(d)	(b)	(b)	(c)	(b)	(d)	(c)	(c)	(b)	(d)	(a)	(b)	2

EXPLANATIONS

Q.1 (b)

Option (a) False: If the language is context-free then it cannot be said that the language will be accepted by the deterministic push-down automaton as there are some cases defined where the context-free language is not accepted by DPDA.

Option (b) True: The union of context-free languages always produces a context-free language but it's not the case with intersection of two context-free languages.

Option (c) False: As discussed in Option (b).

Option (d) False: Similar to intersection, complement of a context-free language is not always a context-free language.

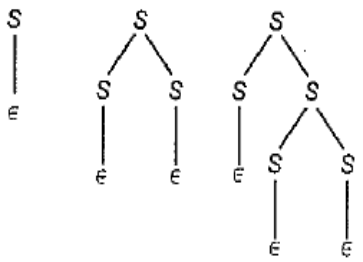
Q.2 (b)

It is given that the, stack is limited to 10 items. So the strings accepted by it are finite. All finite languages are regular.

Q.3 (c)

The grammar is S→ aSb |SS|ϵ

a) G is not ambiguous is false, since ϵ which belongs to L(G), has infinite number of derivation trees, which makes C ambiguous. Some derivation trees are



b) There exists x, yϵL(G) such that xyϵ L(G) is false, since S→SS, can be used derive xy, whenever xϵL(G) and yϵL(G).

C) It is true, since this language is L(G) ={W | $n_a(W) = n_b(W)$ and $n_a(V) \geq n_b(V)$ Where V is any prefix of W} This language happens to be deterministic context-free language.

∴ There exists a DPDA that accepts it.

d) It is false, as the given language is not regular.

∴ No DFA exists to accept it.

Q.4 (c)

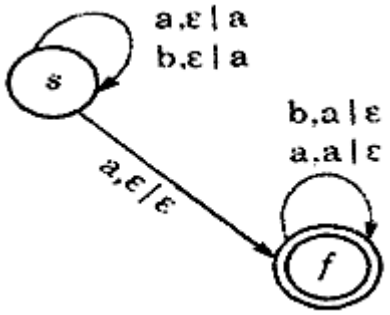
a) This is true, since some CFLs are inherently ambiguous.

b) Always correct, that's why it is called unambiguous.

c) The set of all languages accepted by all NPDAs is the set of CFLs, which is proper superset of the se t of all languages accepted b y all DPDAs, which is the set of DCFLs. Hence option (c) is false.

d) Finite language is always regular. So, true.

Q.5 (b)



This machine accepts the language

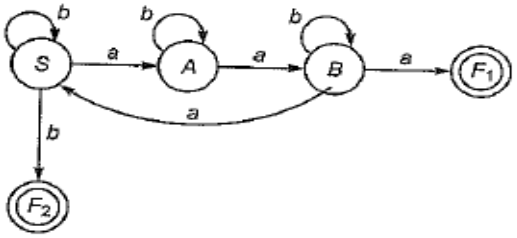
$$L = \{w_1 a w_2 \mid |w_1| = |w_2|\}$$

The language allows only strings whose centre bit is "a" and the strings on either side have equal length i.e., strings of odd length with centre bit "a".

The string "aabab" is odd length but centre bit is not "a".
So, this string is not a member of L(M)

Q.6 **(c)**

Here, we have
$S \rightarrow bS$
$S \rightarrow baA$ $(S \rightarrow aA)$
$S \rightarrow baaB$ $(A \rightarrow aB)$
$S \rightarrow baaa$ $(B \rightarrow a)$
Therefore, $|Na(w)| = 3$.
Also, if we use $A \rightarrow bA$ instead of $A \rightarrow aB$,
$S \rightarrow baA$
$S \rightarrow babA$
To terminate A,
we would have to use $A \rightarrow aB$ as only B terminates at a $(B \rightarrow a)$.
$S \rightarrow baA$
$S \rightarrow babA$
$S \rightarrow babaB$
$S \rightarrow babaa$



Thus, here also, $|Na(w)|=3$.
So, C is the correct choice.

Q.7 **(b)**

$L = \{a^m b^n c^{m+n} \mid m, n \geq\}$

A DPDA can accept this language. When a's and b's are in input, these are pushed into the stack and when c's appear in input the a's and b's are popped out. If after all c's are finished, if stack is empty, this means $n_c(W) = n_a\{W)+n_b(W)$ and the string is accepted. Else it is rejected. Now, since a DPDA exists, the language is context-free. Clearly the language is not regular, since we must count and compare c's with a's and b's which cannot be done by any FA.

Q.8 **(d)**

$L^C \cup M^C = (Reg)^C \cup (CFL)^C =$
$Reg \cup (CSL)^C = Reg \cup CSL = CSL$

A CSL may or may not be regular. So, options (a) and (c) are false. A CSL need not be a CFL. So, option (b) is false. So, answer is (d) none of these.

Q.9 **(d)**

In NPDA we may a dead configuration. This mean we may not give any transition to any alphabet from this state.
We say that a string is accepted if PDA is in final state after reading the final symbol in the string or after it has read `$ symbol denoting end of the string and it is in final state.
Question never says that we have transitions defined for all the alphabet symbols in the PDA. Neither L(P) nor N(P)are necessarily $\sum^*$.

Q.10 **(b)**

L_1 is CFL but not a DFCL, since accepting WW^R necessarily involves finding the middle of the string, which involves non-determinism. Therefore, (a) is false.
L_2 is a DFCL is true since # is a special symbol and middle string can now be surely found by using #, thereby eliminating the need for non-deterministic guessing. So, (b) is true.
L_3 is a CSL and not a CFL. So (c) is false.
L_3 is not a DCFL either. So, (d) is false.

Q.11 **(a)**

L_1 and L_2 are context-free languages and therefore, $L_1 \cap L_2$ may or may not be context-free, since CFLs are not closed under intersection.
Now, let us look at $L_1 \cap L_2$.

$L_1 \cap L_2 = \{a^n b^n c^n \mid n > 0\}$

Which is clearly not context-free but is context sensitive.

Q.12 (d)

We know that; the languages accepted by non-deterministic finite automata are also accepted by deterministic-finite automata. This may not be in the case of context-free languages.

Therefore, $D_f = N_f$ and $D_p \subset N_p$

Q.13 (b)

L(G) =Set of all palindrome strings.

"baba" is not a palindrome string.

Q.14 (b)

To be accepted by NPDA but not DPDA, the language must be a C FL but not DCFL.

a) is wrong, it is not context free.

b) $L = \{a^l b^m c^n \mid l \neq m \text{ or } m \neq n\}$

This language has double comparison on m with `or'. So the language is CFL but not DCFL.

c) In this language only one comparison is there and push and pop position are clear. This is a DCFL.

d) a*b* is regular. Every regular language is a DCFL. So this language is also a DCFL.

Q.15 (d)

L is CFL and M is Regular. Intersection of CFL and Regular is always a CFL, since all languages are closed under regular intersection.

Q.16 (c)

For every a, we put two X in stack [at state s]

For every b we pop out one X [reach to state (getting b after a)]

For every c we pop out one X [reach to state u (getting c after b)]

If all X are popped out then reached to final state f, means t hat,

Sum of number of b's and number of c's = twice of number of a's

i.e. $L = \{a^l b^m c^n \mid 2l = m+n\}$

Q.17 (b)

$A \to bA \mid \in$

$L(A) = b^*$

$S \to aSAb \mid \in$

Now, substituting A Qution into this gives

$S \to aSAb \mid \in$

No, substituting A Qution into this gives

$S \to aSb^*b \mid \in$

$S \to aSb \mid aSbb \mid aSbbb... \mid \in$

$L(S) = \{a^m b^n \mid m \leq n\}$

Q.18 (d)

The language is context free or not, this can be proved by finding out the grammar for all the languages.

$L = \{0^{n+m}1^n0^m \mid n, m \geq 0\}$

The production for L_1 as follows:

$S \to 0S_0 \mid 0A1 \mid \varepsilon$

$A \to 0A1 \mid \varepsilon$

Now, we need to apply these values of the production individually to generate

$0^m0^n1^n0^m \to$ to prove

This can be verified by using stack.

Hence, proved, So L_1 is context-free.

Going through the same procedure, we can see that two comparisons are made in the L_2 language, so, it is not context-free.

$L_3 \to$ Going through the same procedure again, we can see that two comparisons are made in the L_3 language, so it is not context-free.

Q.19 (b)

a) G would be ambiguous if replacement of 5 add equal a's and b's.

b) The grammar cannot generate string aabb though they have equal number of a's and b's.
c) The language exhibiting such nature is accepted by deterministic push- down automata.

Example of string abba
Now, S→SS→εS→εSS→SS abS→abba
Also, S→SS→abS→abba
Here, we can see two derivations of same string. So it is ambiguous.

Q.20 (c)
Grammar may change but language remain the same in the grammar and its disambiguated version.
$L(D) = L(G)$

Q.21 (d)
Using capital letters for variables and small letters for terminals, the given grammar G1 becomes
$S \to w(E)S$
$S \to o$
$E \to i$
Since LHS of every production is a single variable, G1 is context free grammar.
G2 becomes
$S \to w(E)S$
$S \to o$
$E \to E+E$
$E \to E*E$
$E \to i$
Since LHS of every production is a single variable, G2 is also a context free grammar. RHS of some of the productions in G1 and G2 has two non-terminals, hence it is not regular grammar.

Q.22 (b)
$\{0^i 21^i \mid i \geq 0\}$ is a DCFL since, a DPDA can accept this language. 0's are pushed into the stack and then when 2 appears in input, state is changed and immediately after that for e very 1, a zero is popped out of the stack. In the end, if stack has start stack symbol only, then the string is accepted. Else it is rejected. Since every DCFL is recursive, we can say that the language is recursive, and is a DCFL.

Q.23 (b)
Option (a) Decidable: Due to presence of CYK algorithm, the membership problem for CFGs becomes decidable.
Option (b) Undecidable: The ambiguity problem for CFGs cannot be decided.
Option (c) Decidable : Like pumping lemma is used to check whether the language is regular or not, similarly algorithms are employed and implemented to check for the regular language whether the regular languages are finite or not.
Option (d) Decidable: Equivalence problem for FSAs are always decidable.

Q.24 (b)
L1 is CFL [push a's onto stack, and pop a with each b]
L2 is CFL [push b's onto stack and pop b with each c]. So, option (a) is false.
In L1, every string has atleast one "c".
In L2, every string has no "c".
So, L1 ∩ L2 = ϕ. So option (b) us true.
Union of two CFLs is a CFL and so option (c) is false.
L2 is DCFL and hence acceptable b y a DPDA. So, option (d) is false.

Q.25 (b)
$S \to B$
$B \to 0B00 \mid 1$
So, S generates $\{0^n 10^{2n} \mid n \geq 1\}$
$S \to AA$
$A \to 0A \mid A0 \mid 1$
$A \to \{0^m 10^n \mid m, n \geq 0\}$

So, S generates $\{0^m10^n0^p10^q \mid m,n,p,q \geq 0\}$.

$= \{0^m10^x10^q \mid m,x,q \geq 0\}$

$= \{0^i10^j10^k \mid i,j,k \geq 0\}$

$L(S) = \{0^i10^j10^k \mid i,j,k \geq 0\} \cup \{0^n10^{2n} \mid n \geq 1\}$

Which is same as option (b).

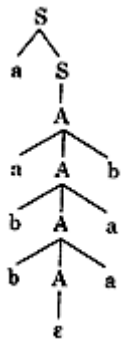
Q.26 (d)

$S \rightarrow aS$
$S \rightarrow aA$
$S \rightarrow aaAb$
$S \rightarrow aabAab$
$S \rightarrow aabbAaab$
$S \rightarrow aabbaab$

Q.27 (a)

$S \rightarrow aS$	1
$S \rightarrow aA$	2
$S \rightarrow aaAb$	3
$S \rightarrow aabAab$	4
$S \rightarrow aabbAaab$	5
$S \rightarrow aabbaab$	6

Thus 6 steps are needed and only one parse tree shown below.



Q.28 (b)

Statement 1 Decidable: The algorithm can be used to check the finiteness/infiniteness on the DFA and also the two given DFAs, a product DFA can be constructed.

Statement 2 Undecidable: It is not decidable since the language is context-free.

Statement 3 Undecidable : It is also undecidable that whether two push down automata accept the same language.

Statement 4 Decidable : If the LHS of each production has one and only one variable then it is a context-free grammar.

Q.29 (d)

We have to prove if the language is
1. regular
2. context-free
3. accepted by turing machine

Applying Pumping lemma, we get that the given language is neither regular nor context-free.

Q.30 (c)

The matched sequence is as given

E. Checking that identifiers are declared before their use	R. $L=\{WCW \mid W \in (a\|b)^*\}$
F. Number of formal parameters in the declaration of a function agrees with the number of actual parameters in the use of the function	P. $L=\{a^n b^m c^n d^m \mid n \geq 1, m \geq 1\}$
G. Arithmetic expressions with matched pairs of parentheses	Q. $X \rightarrow XbX \mid XcX \mid dXf \mid g$
H. Palindromes	S. $X \rightarrow bXb \mid cXc \mid \varepsilon$

Q.31 (c)

Statement 1 True: Left recursive and right recursive grammar can be converted into each other.

Statement 2 False: We can remove all epsilon productions only if grammar doesn't contain epsilon in the language.
Statement 3 True: It is the definition of regular grammar.
Statements 4 True: The derivation trees in CNF are binary trees.

Q.32 (d)

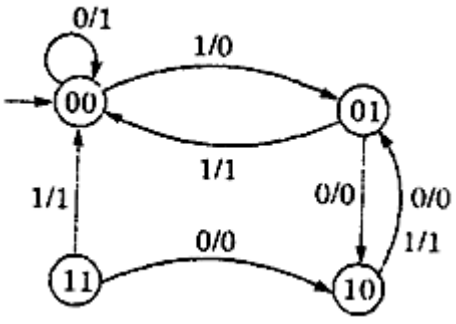
Option (a) True: Non-deterministic finite automata can be converted into the deterministic finite automata.
Option (b) True: With reference to option (a), same is with the non-deterministic turing machine.
Option (c) True: Regular language is always context-free but the reverse is not true.
Option (d) False: We know that a set is a subset of itself and hence, every subset of recursively enumerable set is not recursive.

Q.33 (d)

Since it is known that the set of DCFL' are a proper sub class of the set of CFL's NPDA's which corresponds in general to class of CFL's cannot in general be converted to equivalent DPDA's which correspond to the class of DCFL's

Q.34 (a)

The transition diagram for the given FSM is given below:



As can be seen from above diagram the minimum length of the input string that will take machine from 00 to 01 with output 1, is three.

In fact the minimum length input string is "101". Which will give an output of1, while reaching state 01.

Q.35 (c)

The language L1 accept strings {c, abc, abcab, aabbcab, aabbcaabb, ...} and L2 accept strings {a, b, c, ab, abc, aabc, aabbc, ... }. Intersection of these two languages is $L1 \cap L2 = \{a^k b^k c \mid k >= 0\}$ which is context free, but not regular.

Q.36 (b)

Regular expression A regular expression is a way for a computer programmer to express, how a computer program should look for a specified pattern in text and then what the program is to do when each pattern match is found. This is done using lexical analysis.
Push-down automata PDAs are finite automatons with a data structure that can be used to store an arbitrary number of symbols but which can be only.
Accessed in a Last-In-First-Out (LIFO) fashion. PDA follows syntax analysis.
Dataflow analysis It is the process of collecting information about the way the variables are used, defined in the program. The analysis is done at basic block granularity. Collected information is represented as a set of dataflow equations. The technique is useful for performing several optimizations such as constant propagation and copy propagation.

Q.37 (d)

There need not be an equivalent DPDA for each NPDA.

Q.38 (b)

The possible palindrome generated by above grammar can be of odd length only as there is no rule.

Example generated palindromes are aba, aaa, bab, ababa, aaaaa, ..

Q.39 (d)

A context-free language is $L=\{a^n b^n : n \geq 1\}$ that is the language of all non-empty even-length strings. It consists of

1. The entire first halves of which are a's.
2. The entire second halves of which are b's.

L is generated by the grammar S→aSb|ab, and is accepted by the push-down, automaton. ,

$M=(\{q_0,q_1,q_1\},\{a,b\},\{a,z\},\delta,q_0,\{q_1\})$

$M=(\{q_0,q_1,q_1\},\{a,b\},\{a,z\}\delta,q_0,\{q_1\})$

, where δ are defined as follows:

$\delta(q_0,a,z)=(q_0,a,z)$

$\delta(q_0,a,a)=(q_0,a)$

$\delta(q_0,b,a)=(q_1,x)$

$\delta(q_1,b,a)=(q_1,x)$

$\delta(q_1,\lambda,z)=(q_f,z)$

δ (state$_1$, read pop) = (state$_2$, push)

Where z is initial stack symbol and x means pop action.

The given languages follow all the rules of the context-free languages and also these languages are accepted by push-down automata, therefore all languages are context-free.

Q.40 (c)

$L_1=\{0^p1^q \mid p,q,\in N\}$ is regular language.

is context-free language

$L_2=\{0^p1^q \mid p,q,\in N \text{ and } p=q\}$

$L_3=\{0^p1^q0^r \mid p,q,\in N \text{ and } p=q=r\}$

is not context free.

Q.41 (d)

$L_1=\{0^p1^q0^r \mid p,q,r\geq 0\}$

$L_2=\{0^p1^q0^r \mid p,q,r\geq 0, p\neq r\}$

a) L_2 is context Free - true

We can accept or reject L_2 with single stack. Insert P 0's into stack skip q 1's.

For each 0 corresponding to r, remove 0 from stack.

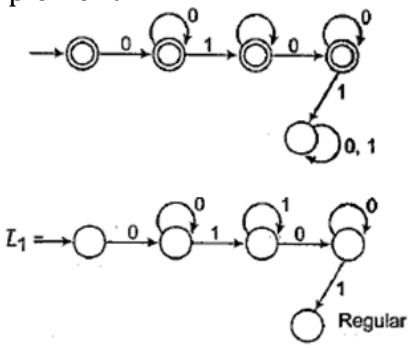
b) $L_1 \cap L_2$ is context Free — true

Here $L_1 \cap L_2 = L_2$ which is context Free.

c) Complement of L_2 is recursive — true L_2 is Context-Free language Complement of CFL may or may not be CFL. Complement by CFL is definitely recursive.

d) Complement of L_1 is Context Free, but not regular false.

$L_1=\{0^p1^q0^r|_{p,q,r\geq 0}\}$ is regular and regular languages are closed under complement.



Hence, the answer is (d).

Q.42 (c)

L3 is non-deterministic context free language.

Q.43 (b)

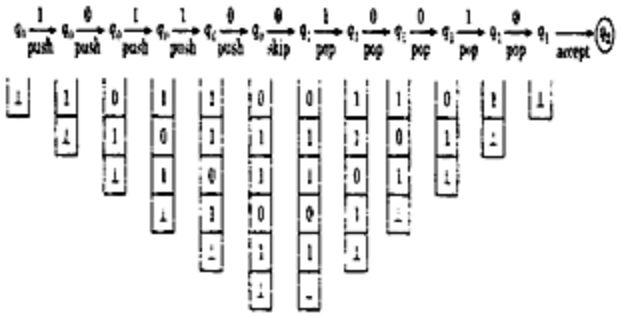
The language accepted is

$L(M)=\{wx\overline{w}^R \mid x\in\{0,1,\varepsilon\}$, where $\overline{w}$ is the 1's complement of w$\}$.

Only the string "10110010010" is in the language and hence will be accepted.

The following diagram shows how this string will be accepted by the PDA.

$\underset{\text{Given string}}{\underline{101100}} \quad \underset{\text{Option(b)string}}{\underline{10010}}$



Q.44 (b)

L1 is CFL because there is only one comparison at a time. First we will push 'a' into the stack m number of times and then push 'b', n number of times. Now when 'a' comes, don't push it into stack, pop b's for every a's. When b comes don't push then into the stack, pop a's for every b's.

L2 is not CFL. because here we are not able to check equal numbers of a's and equal number of b's.

L3 is CFL. We can rewrite it like $\{a^{2n+1}b^n\}=\{a^{2n}ab^n\}=\{aa^{2n}b^n\}$. This can be implemented using PDA.

Q.45 (c)

$G_1 : S_1 \rightarrow aS_1b|\varepsilon$

$G_2 : S_2 \rightarrow abS_2|\varepsilon$

$L_1 = L(G_1) = \{a^n b^n | n \geq 0\}$

$L_2 = L(G_2) = \{(ab)^n | n \geq 0\} = (ab)^*$

Clearly L_1 is regular is false of L_2 is regular is True. So correct answer is (c).

Q.46 (d)

$S \rightarrow aS | bS | \epsilon$

$S \rightarrow \epsilon$

$S \rightarrow aS$ generates all strings with RE a*

$S \rightarrow aS$ generates all strings with RE b*

$S \rightarrow aS | bS$ generates all strings with RE ab* and ba* and then any combination of a and b.

So given grammar represents RE {a,b}*

Q.47 (d)

G_1: $S \rightarrow aS | B$

$B \rightarrow b | b\ B$ which generates strings with RE b^+

The above grammar can be written as $S \rightarrow aS | b^+$ which represents the strings with RE $a^* b^+ = \{a^m b^n | m \geq 0, n > 0\}$

G_2: $S \rightarrow aA | bB$

$A \rightarrow aA | B | \epsilon$

$B \rightarrow bB | \epsilon$

The minimum strings obtained from starting symbol are a, b

$S \rightarrow aA$, $A \rightarrow aA | \epsilon$ generates strings a+

$S \rightarrow bB$, $B \rightarrow bB | \epsilon$ generates strings b+

Also, the combination of these productions generates strings start with at least one a and followed by any number of b's.

So it is the language $L = \{a^m b^n | m > 0, n \geq 0\} \cup \{b^n | n > 0\}$ which can also be written as $\{a^m b^n | m > 0$ **or** $n > 0\}$.

Q.48 (d)

$L = \{a^n | n \geq 0\} \cup \{a^n b^n | n \geq 0\}$ and L is DCFL.

Q.49 (b)

L1: $\{a^n b^m c^{n+m} | m, n >= 1\}$ – CFL

L2: $\{a^n b^n c^{2n} | n >= 1\}$ – CSL

L1 is CFL but L2 is Not CFL because in L2 comparison is needed twice which cannot be done using one stack.

Q.50 (b)

$S \rightarrow Ab | Bb | \varepsilon$

$A \rightarrow Bd | \varepsilon$

$B \rightarrow e$

This generates finite language and there is no left recursion.

C and D have indirect left recursion.

Q.51 (c)

$S \rightarrow XY$

$X \rightarrow aX|a \Rightarrow X \rightarrow \{a^m | m \geq 1\}$

$Y \rightarrow aYb|\in \Rightarrow Y \rightarrow \{a^n b^n | n \geq 0\}$

$S \rightarrow XY \Rightarrow S \rightarrow \{a^m b^n | m > n, n > 0\}$

m > n because at least 1 will be attached on left of $a^n b^n$.

Q.52 (b)

I. $L_1 \cup L_2$ is context –free = CFL ∪ CFL = CFL . So, True

II. $\overline{L_1}$ is context- free = $\overline{CFL}$ = CSL but not CSL but not CFL. So, false.

III. $L_1 - R$ is context –free CFL

$\cap \overline{\text{Regular}} = \text{CFL} = \text{CFL} = \text{CFL}$

.So, True

IV. $L_1 \cap L_2$ is context-free

$= \text{CFL} \cap \text{CFL} = \text{CSL}$. So, False

Q.53 (d)

$L_1 = \{a^p \mid p \text{ prime}\}$ is a CSL but not CFL(prime number checking involve division)

$L_2 = \{a^n b^m c^{2m} \mid n0, m0\}$ is CFL (one comparision)

$L_3 = \{a^n b^n c^{2n} \mid 0\}$ is CSL (two comparision)

$L_4 = \{a^n b^n \mid n1\}$ is a DCFL

So,

I. L_1 is CFL but not regular is false

II. L_2 is not CFL is false

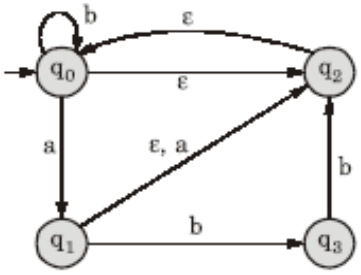
III. L_3 is not CFL but recursive is true since every CSL is recursive.

IV. L_4 is true DCLF is true.

So, only III and IV are true and correct.

Q.54 (c)

Covering the table to a state diagram, we get ,



$\hat{\delta}(q_2, abc)$ = All states reachable from q_2 by "aba" If aba is broken as ε.a. ε. ε.b.a.Then from q_2 we can reach q_1 and from there by null transition we can reach state q_2 as well as q_0.

$\hat{\delta}(q_2, aba) = \{q_0, q_1, q_2\}$

Q.55 (c)

$S \rightarrow abScT \mid abcT$

$T \rightarrow bT \mid b$

Qving $T \rightarrow bb^*$

Substitute in S to get

$S \rightarrow abScbb^* \mid abcbb^*$

So, Qution of S would be

$\{(ab)^n (cb^m)^n \mid m, n \geq 1\}$

So, option (c) is correct

Q.56 (b)

$G_1 : S \rightarrow aSb \mid T, T \rightarrow cT \mid \in$

$G_2 : S \rightarrow bSa \mid T, T \rightarrow cT \mid \in$

$L(G_1) = \{a^n c^m b^n \mid m, n, \geq 0\}$

$L(G_2) = \{b^n c^m a^n \mid m, n \geq 0\}$

$L(G_1) \cap L(G_2) = \{a^n c^m b^n\} \cap \{b^n c^m a^n\} = \{c^m \mid m \geq 0\} = C^*$

Since the only common strings will be those strings with only 'c', since in the first language all the other strings start with 'a' and in the second language all the other strings start with 'b' Clearly the intersection is not finite but regular.

Q.57 (d)

The given grammar is

$S \rightarrow SaS \mid aSb \mid bSa \mid SS \mid \in$

Now $S \rightarrow aSb \mid bSa \mid SS \mid \in$

Generated all s strings with equal number of 'a' and 'b'. Now $S \rightarrow SaS$ can only generated strings where 'a' is more than 'b' since on left and right of 'a' in SaS, S will have only strings with $n_b > n_a$ is not possible to generate by the given grammar

Q.58 (a)

L_1 is a CFL.

L_2 also a CFL.

Union of 2 CFL is always as CFL, but intersection may or may not be

So, I is clearly true.

Let us intersect and check II

$\{a^n b^n c^m\} \cap \{a^n b^m c^m\} = \{a^n b^n c^n\}$

Which is clearly not a CFL but CSL. So option (a) which is I only, is the answer.

Q. 59 (b)

I. $\{a^m b^n c^p d^q \mid m + p = n + q$, where m, n, p, $q \geq 0\}$ since, m+p = n+q can also be written as m-n=q-p. For this we can construct PDA in the following way: Push a on the stack. For every b pop a from the stack. When c comes push it. For every d pop one c from the stack. Do it until top of the stack becomes a or Z. Now there should be exactly same a on the stack as d in the input. For that pop one a for every d.

II. $\{a^m b^n c^p d^q \mid m = n$ and $p = q$, where m, n, p, $q \geq 0\}$ is also context free. Its PDA can easily be constructed.

III and IV are not context free.

Q. 60 2

L1* = ε + 0(00)

$L_0 = \varepsilon$

$L_1 = \varepsilon \,.\, (\varepsilon + 0(00)^*)$

$= \varepsilon + 0(00)^* = L1$

$L_2 = L_1 \,.\, L1 = L1 \,.\, L1$

$= (\varepsilon + 0(00)^*)\,(\varepsilon + 0(00)^*)$

$= \varepsilon + 0(00)^* + 0(00)^* + 0(00)^* \,.\, + 0(00)^*$

$= \varepsilon + 0(00)^* + 0(00)^*\,0(00)^* =$ **0***

Given automaton contains epsilon and even number of 0's and odd numbers of 0's.

$L_3 = L_2 \,.\, L1$

$= \{0^*\} \,.\, \{\varepsilon + 0(00)^*\} = 0^*\,0(00)^* = 0^*$

So, $L_2 = L_3 = L_{2+1}$

So, the smallest value of **k = 2**

3 TURING MACHINE, UNRESTRICTED LANGUAGES, COMPLEXITY

3.1 INFORMAL DESCRIPTION

We consider here a basic model of TM which is deterministic and have one-tape. There are many variations, all are equally powerful. The basic model of TM has a finite set of states, a semi-infinite tape that has a leftmost cell but is infinite to the right and a tape head that can move left and right over the tape, reading and writing symbols.

For any input w with |w|=n, initially it is written on the n leftmost (contiguous) tape cells. The infinitely many cells to the right of the input all contain a blank symbol, B which is a special tape symbol that is not an input symbol. The machine starts in its start state with its head scanning the leftmost symbol of the input w. depending upon the symbol scanned by the tape head and the current state the machine makes a move which consists of the following:

- Writes a new symbol on that tape cell, moves its head one cell either to the left or to the right and (Possibly) enters a new state.
- The action it takes in each step is determined by a transition functions. The machine continues computing (i.e. making moves) until it decides to "accept" its input by entering a special state called accept or final state or halts without accepting i.e. rejecting the input when there is no move defined.
- On some inputs the TM may keep on computing forever without ever accepting or rejecting the input, in which case it is said to "loop" on that input.

Formal Definition

Formally, a deterministic Turing machine (DTM) is a 7-tuple $M=(Q,\sum,\Gamma,\delta,q_0,B,F)$, where

- Q is a finite nonempty set of states.
- Γ is a finite non-empty set of tape symbols, called the tape alphabet of M.
- $\sum\subseteq\Gamma$ is a finite non-empty set of input symbols, called the input alphabet of M.
- $\delta: Q\times\Gamma\to Q\times\Gamma\times\{L\times R\}$ is the transition function of M,
- $q_0\in Q$ is the initial or start state.
- $B\in\Gamma\setminus\sum$ is the blank symbol
- $F\subseteq Q$ is the set of final state.

So, given the current state and tape symbol being read, the transition function describes the next state, symbol to be written on the tape, and the direction in which to move the tape head (L and R denote left and right, respectively).

Transition function: δ

- The heart of the TM is the transition function, δ because it tells us how the machine gets one step to the next.
- when the machine is in a certain state $q\in Q$ and the head is currently scanning the tape symbol $X\in\Gamma$, and if $\delta(q,x)=(p,Y,D)$, then the machine
  1. Replaces the symbol x by Y on the tape
  2. Goes to state p, and
  3. The tape had moves one cell (i.e. one tape symbol) to the left (or right) if D is L or R.

The ID (instantaneous description) of a TM capture what is going out at any moment i.e. it contains all the information to exactly capture the "current state of the computations".

It contains the following:

- The current state, q
- The position of the tape head,
- The constants of the tape up to the rightmost nonblank symbol or the symbol to the left of the head, whichever is rightmost.

Note that, although there is no limit on how far right the head may move and write nonblank symbols on the tape, at any finite TM, the TM has visited only a finite prefix of the infinite tape.

An ID (or configuration) of a TM M is denoted by $\alpha q \beta$

where $\alpha, \beta \in \Gamma^*$

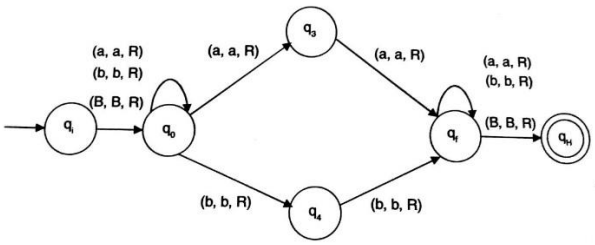
- α is the tape contents to the left of the head
- q is the current state.
- β is the tape contents at or to the right of the tape head.

That is, the tape head is currently scanning the leftmost tape symbol of β.

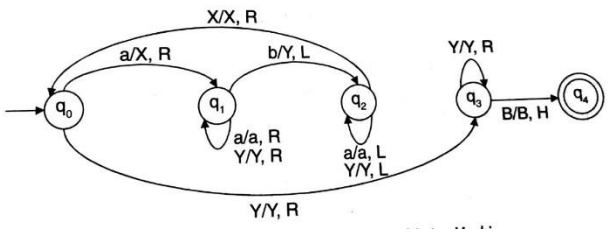
(Note that if $\beta = \in$, then the tape head is scanning a blank symbol), If q_0 is the start state and w is the input to a TM M then the starting or initial configuration of M is on piously denoted by q_0^w.

Examples:

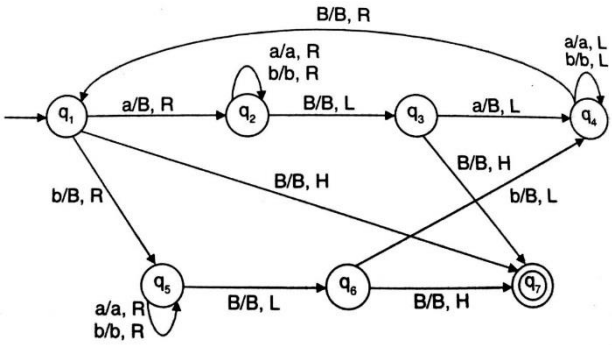
L1 = {Turing machine for
(a+b)*(aa+bb)(a+b)*



L2 = {$a^n b^n$|n>=0}



L3 = {Palindrome strings over {a,b}}



3.2 TYPES OF TURING MACHINE

There are a number of other types of Turing machines in addition to the one we have seen such as Turing machines with multiple tapes, ones having one tape but with multiple heads, ones with two dimensional tapes, nondeterministic Turing machines etc. It turns out that computationally all these Turing machines are equally powerful. That is, what one type can compute any other can also compute. However, the efficiency of computation, that is, how fast they can compute, may vary.

Turing Machines with Two Dimensional Tapes

This is a kind of Turing machines that have one finite control, one read-write head and one two dimensional tape. The tape has the top end and the left end but extends indefinitely to the right and down. It is divided into rows of small squares. For any Turing machine of this type there is a Turing machine with a one dimensional tape that is equally powerful, that is, the former can be simulated by the latter.

To simulate a two dimensional tape with a one dimensional tape, first we map the squares of the two dimensional tape to those of the one dimensional tape diagonally as shown in the following tables:

v	v	v	v	v	v	v	. . .	. . .
h	1	2	6	7	15	16	. . .	. . .
h	3	5	8	14	17	26	. . .	. . .
h	4	9	13	18	25	. . .	. . .	. . .
h	10	12	19	24	. . .	. . .	. . .	. . .
h	11	20	23	. . .	. . .	. . .	. . .	. . .
h	21	22	. . .	. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

Here the numbers indicate the correspondence of squares in the two tapes: square i of the two dimensional tape is mapped to square i of the one dimensional tape. h and v are symbols which are not in the tape alphabet and they are used to mark the left and the top end of the tape, respectively.

Turing Machines with Multiple Tapes:

This is a kind of Turing machines that have one finite control and more than one tapes each with its own read-write head. It is denoted by a 5-tuple < Q, Σ , Γ , q_0, δ >. Its transition function is a partial function δ : Q X ($\Gamma \cup \{\Delta\}$)^n -> (Q $\cup$ { h }) X ($\Gamma \cup \{\Delta\}$)^n X { R , L , S }^{n}. A configuration for this kind of Turing machine must show the current state the machine is in and the state of each tape. It can be proven that any language accepted by an n-tape Turing machine can be accepted by a one tape Turing machine and that any function computed by an n-tape Turing machine can be computed by a one tape Turing machine. Since the converses are obviously true, one can say that one tape Turing machines are as powerful as n-tape Turing machines.

Turing Machines with Multiple Heads:

This is a kind of Turing machines that have one finite control and one tape but more than one read-write heads. In each state only one of the heads is allowed to read and write. It is denoted by a 5-tuple < Q, Σ , Γ , q_0, δ >. The transition function is a partial function δ : Q X { H_1 , H_2 ... , H_n } X ($\Gamma \cup \{\Delta\}$) -> (Q $\cup$ { h }) X ($\Gamma \cup \{\Delta\}$ X { R , L , S } , where H_1 , H_2 ... , H_n denote the tape heads.

It can be easily seen that this type of Turing machines are as powerful as one tape Turing machines.

Turing Machines with Infinite Tape:

This is a kind of Turing machines that have one finite control and one tape which extends infinitely in both directions. It turns out that this type of Turing machines are only as powerful as one tape Turing machines whose tape has a left end.

Nondeterministic Turing Machines:

A nondeterministic Turing machine is a Turing machine which, like nondeterministic finite automata, at any state it is in and for the tape symbol it is reading, can take any action selecting from a set of specified actions rather than taking one definite predetermined action. Even in the same situation it may take different actions at different times. Here an action means the combination of writing a symbol on the tape, moving the tape head and going to a next state. For example let us consider the language L = {ww: w $\in$ {a, b}*}. Given a string x, a nondeterministic Turing machine that accepts this language L would first guess the midpoint of x that is the place where the second half of x starts. Then it would compare the first half of x with the second half by comparing the i-th symbol of the first half with the i-th symbol of the second half for i = 1, 2, ... A deterministic Turing machine, on the other hand, can not guess the midpoint of the string x. It must find the midpoint by for example pairing off symbols from either end of x. Formally a nondeterministic Turing machine is a Turing machine whose transition function takes values that are subsets of

$(Q \cup \{ h \}) \times (\Gamma \cup \{\Delta\} \times \{R,L,S\}$.

As in the case of NFA, it is understood that a nondeterministic Turing machine at any configuration selects one combination of next state, tape symbol and head movement out of the set of triples without following any specific predetermined rule.

It can be shown that a nondeterministic Turing machine is only as powerful as a deterministic Turing machine.

3.3 RECURSIVE AND RECURSIVELY ENUMERABLE LANGUAGES:

Output of turing machine can be one of the following three.

- It will halt and accept the string
- It will halt and reject the string
- It will go into an infinite loop

RECURSIVELY ENUMERABLE LANGUAGE: TURING RECOGNIZABLE

Recursively Enumerable (r.e) language: (or TM-recognizable language or semi-decidable language). Simply speaking, a language L is recursively enumerable if some Turing Machine accepts it. Formally, the class of r.e. languages is defined as

$\{L \mid L \subseteq \Sigma^*$ and $\exists$ TM M such that L= L(M)$\}$

- So, on input string $w \in L$, M enters an accepting ID and halts.
- But, on input strings $w \notin L$,M either halts entering a ID (i.e. without entering an accepting ID), or it never halts (i.e. it loops for ever).

RECURSIVE (OR DECIDABLE) LANGUAGES: TURING DECIDABLE

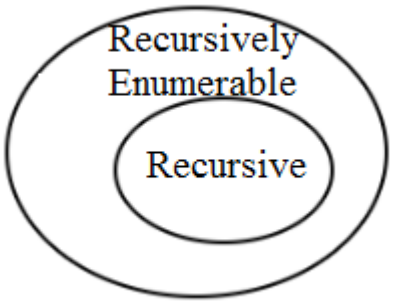
A language L is recursive if there is some TM M that halts on every input $w \in \Sigma^*$ and $L = L(M)$.

Formally, the class of recursive language is defined as $\{L \mid L \subseteq \Sigma^*$ and $\exists$TM M such that M halts $\forall\ w \in \Sigma^*$ and L =L(M)$\}$

- So, on any input strings $w \in L$, M enters an accepting ID and halts and
- On an input string $w \notin L$, M halts entering in a blocking ID (or entering in a reject state).

Informally, a language is said to be recursively enumerable if there exists a TM which accepts it and a language is said to be recursive if there exist a TM which halts on every input.

Every recursive language is recursively enumerable.



Closure properties of Recursive Languages:

Recursive languages are closed under Union, intersection, complement, concatenation, kleene closure.

Closure of Recursively Enumerable Languages:

Recursive languages are closed under Union, intersection, concatenation, kleene closure.

3.4 UNRESTRICTED GRAMMARS:

An unrestricted grammar is a 4-tuple G = (V, Σ, S, P), where V and Σ are disjoint sets of variables and terminals, respectively.

S is an element of V called the start symbol, and P is a set of productions of the form

$$\alpha \rightarrow \beta$$

where $\alpha, \beta \in (V \cup \Sigma)*$ and α contains at least one variable.

Example:

A Grammar Generating the language $L = \{a^{2^k} \mid k \in \mathcal{N}\}$

$S \rightarrow LaR\ L \rightarrow LD\ Da \rightarrow aaD\ DR \rightarrow R\ L \rightarrow \wedge\ R \rightarrow \wedge$

Derivation of string aaaa

$S \Rightarrow LaR \Rightarrow LDaR \Rightarrow LaaDR \Rightarrow LaaR \Rightarrow LDaaR$
$\Rightarrow LaaDaR \Rightarrow LaaaaDR \Rightarrow LaaaaR \Rightarrow aaaaR \Rightarrow aaaa$

Example:

A Grammar Generating $\{a^n b^n c^n \mid n \geq 1\}$

$S \rightarrow SABC \mid LABC$
$BA \rightarrow AB\ CB \rightarrow BC\ CA \rightarrow AC$
$LA \rightarrow a\ aA \rightarrow aa\ aB \rightarrow ab\ bB \rightarrow bb\ bC \rightarrow bc\ cC \rightarrow cc$

3.5 CONTEXT SENSITIVE GRAMMAR AND LINEAR BOUNDED AUTOMATA:

Context Sensitive Grammar:

A *context-sensitive grammar* (CSG) is an unrestricted grammar in which no production is length-decreasing. In other words, every production is of the form $\alpha \rightarrow \beta$, where $|\beta| \geq |\alpha|$.

A language is a context-sensitive language (CSL) if it can be generated by a context-sensitive grammar.

Linear Bounded Automata:

A *linear-bounded automaton* (LBA) is a 5-tuple $M = (Q, \Sigma, L, q0, \delta)$ that is identical to a nondeterministic Turing machine, with the following exception. There are two extra tape symbols, [and], assumed not to be elements of the tape alphabet L . The initial configuration of M corresponding to input x is $q0[x]$, with the symbol [in the leftmost square and the symbol] in the first square to the right of x. During its computation, M is not permitted to replace either of these brackets or to move its tape head to the left of the [or to the right of the].

3.6 CHOMSKY HIERARCHY:

Type	Languages (Grammars)	Form of Productions in Grammar	Accepting Device
3	Regular	$A \rightarrow aB,\ A \rightarrow \Lambda$ $(A, B \in V, a \in \Sigma)$	Finite automaton
2	Context-free	$A \rightarrow \alpha$ $(A \in V, \alpha \in (V \cup \Sigma)^*)$	Pushdown automaton
1	Context-sensitive	$\alpha \rightarrow \beta$ $(\alpha, \beta \in (V \cup \Sigma)^*, \lvert\beta\rvert \geq \lvert\alpha\rvert,$ α contains a variable)	Linear-bounded automaton
0	Recursively enumerable (unrestricted)	$\alpha \rightarrow \beta$ $(\alpha, \beta \in (V \cup \Sigma)^*,$ α contains a variable)	Turing machine

3.7 COMLEXITY THEORY:

Time complexity: is a measure of how long a computation takes to execute. For a Turing machine, this could be measured as the number of moves required to perform a computation. For a digital computer, it could be measured as the number of machine cycles required for the computation.

Space complexity: is a measure of how much storage is required for a computation. For a Turing machine, the obvious measure is the number of tape squared used; for a digital computer, the number of bytes used.

Both of these measures are functions of a single input parameter, the size of the input. Again, this can be defined in terms of squares or bytes.

For any given input size, different inputs typically require different amounts of space and time. Hence we can discuss for either the average case or for the worst case. Usually we are interested in worst-case complexity because

It may difficult or impossible to define an "average" case. For many problems, the notion of "average case" doesn't even make sense.

It is usually much easier to compute worst-case complexity.

In complexity theory we generally subject our equations to some extreme simplifications. For example, if a given algorithm takes exactly $5n^3+2n^2-n+1003$ machine cycles, where n is the size of the input, we will simplify this to $O(n^3)$ (read: Order n-cubed). This is called an order statistic. Specifically, we:

Drop all terms except the highest-ordered one, and

Drop the coefficient of the highest-ordered term.
Justifications for this procedure are:

For very large values of n, the effect of the highest-order term completely swamps the contribution of lower-ordered term. We are interested in large values of n because, from a strictly practical point of view, it is the large problems that give us trouble. Small problems are almost always feasible to compute.

Tweaking the code can improve the coefficients, but the order statistic is a function of the algorithm itself.

3.8 P, NP, NP-HARD AND NP-COMPLETE PROBLEMS:

In Computer Science, many problems are solved where the objective is to maximize or minimize some values, whereas in other problems we try to find whether there is a solution or not. Hence, the problems can be categorized as follows –

Optimization Problem
Optimization problems are those for which the objective is to maximize or minimize some values. For example,
Finding the minimum number of colors needed to color a given graph.

Finding the shortest path between two vertices in a graph.

Decision Problem
There are many problems for which the answer is a Yes or a No. These types of problems are known as decision problems. For example,

Whether a given graph can be colored by only 4-colors.

Finding Hamiltonian cycle in a graph is not a decision problem, whereas checking a graph is Hamiltonian or not is a decision problem.

What is Language?
Every decision problem can have only two answers, yes or no. Hence, a decision problem may belong to a language if it provides an answer 'yes' for a specific input. A language is the totality of inputs for which the answer is Yes. Most of the algorithms discussed in the previous chapters are polynomial time algorithms.

For input size n, if worst-case time complexity of an algorithm is $O(n^k)$, where k is a constant, the algorithm is a polynomial time algorithm.

Algorithms such as Matrix Chain Multiplication, Single Source Shortest Path, All Pair Shortest Path, Minimum Spanning Tree, etc. run in polynomial time. However there are many problems, such as traveling salesperson, optimal graph coloring, Hamiltonian cycles, finding the longest path in a graph, and satisfying a Boolean formula, for which no polynomial time algorithms is known. These problems belong to an interesting class of problems, called the NP-Complete problems, whose status is unknown.

In this context, we can categorize the problems as follows –

P-Class

The class P consists of those problems that are solvable in polynomial time, i.e. these problems can be solved in time $O(n^k)$ in worst-case, where k is constant.

These problems are called tractable, while others are called intractable or superpolynomial.

Formally, an algorithm is polynomial time algorithm, if there exists a polynomial p(n) such that the algorithm can solve any instance of size n in a time O(p(n)).

Problem requiring $\Omega(n^{50})$ time to solve are essentially intractable for large n. Most known polynomial time algorithm run in time $O(n^k)$ for fairly low value of k.

Regarding O(n log n) time, note that

- The base of the logarithms is irrelevant, since the difference is a constant factor, which we ignore; and
- Although n log n is not, strictly speaking, a polynomial, the size of n log n is bounded by n2, which is a polynomial.

Probably all the programming tasks you are familiar with have polynomial-time solutions. This is not because all practical problems have polynomial-time solutions. Rather, it is because your courses and your day-to-day work have avoided problems for which there is no known practical solution.

NP-Class

The class NP consists of those problems that are verifiable in polynomial time. NP is the class of decision problems for which it is easy to check the correctness of a claimed answer, with the aid of a little extra information. Hence, we aren't asking for a way to find a solution, but only to verify that an alleged solution really is correct.

Every problem in this class can be solved in exponential time using exhaustive search.

P versus NP

Every decision problem that is solvable by a deterministic polynomial time algorithm is also solvable by a polynomial time non-deterministic algorithm.

All problems in P can be solved with polynomial time algorithms, whereas all problems in NP - P are intractable.

It is not known whether P = NP. However, many problems are known in NP with the property that if they belong to P, then it can be proved that P = NP.

If P ≠ NP, there are problems in NP that are neither in P nor in NP-Complete.

The problem belongs to class P if it's easy to find a solution for the problem. The problem belongs to NP, if it's easy to check a solution that may have been very tedious to find.

Polynomial-Time Reducibility:

Let L1 and L2 be languages over alphabets $\Sigma 1$ and $\Sigma 2$, respectively. We say that L1 is polynomial-time reducible to L2 if and only if there is a function f from $\Sigma 1^*$ to $\Sigma 2^*$ such that for any string $x \in \Sigma 1^*$, $x \in L1$ if and only if $f(x) \in L2$ and f can be computed in polynomial time.

NP-Hard

A problem is said to be NP-Hard if every problem in NP are polynomially reducible to it.

NP-Complete

A problem is said to be NP-Complete if it is both NP and NP-Hard.

NP-complete Problems:

1. Satisfiability Problem for Propositional Logic
2. Graph Color Problem

3. Committee Meeting Schedule Problem. In fact most scheduling problems are NP-complete.
4. Traveling Salesman Problem: Given cities and traveling times between cities, a traveling salesman wants to know a shortest route to visit all cities exactly once and come back to where he/she started.
5. Bin Packing Problem: Given a set of objects, their sizes and a number of bins of the same size, find out whether or not the objects can be put into the bins.
6. Partition Problem: Given a set of integers, group them into two groups so that the sum of the numbers of one group is equal to that of the other group.
7. Subgraph Isomorphism Problem: Given two graphs, find out whether or not one is a subgraph of the other.
8. Set Cover Problem: Given a set S, a collection of subsets of S and an integer k, find out whether or not there are k or less subsets in the collection whose union is S.
9. Knapsack Problem: Given a knapsack of size S, a set of objects, their sizes, their values and an integer V, is it possible to select objects so that the sum of their sizes does not exceed S and the sum of their values is V or larger ?
10. 3-Dimensional Matching : Given three sets A, B and C of the same size, and a subset S of the Cartesian product A X B X C. Is there a subset T, called a matching, of S such that every element of A, B, and C appears exactly once in T ?

    For example, let A = {1,2}, B = {a,b}, and C = {x,y}, and S = {(1,b,x),(1,b,y), (2,a,x), (2,b,y)}.

    Then T = {(1,b,y), (2,a,x)} is a desired set satisfying all the requirements. Note that {(1,b,x),(2,a,x)} is not a matching.

GATE QUESTIONS

Q.1 The C language is
a) A context free language
b) A context sensitive language
c) A regular language
d) Parsable fully only by a Turing machine

[GATE-2002]

Q.2 Consider the following decision problems

$\mathbf{P_1}$: Does a given finite state machine accept a given string

$\mathbf{P_2}$: Does a given context free grammar generate an infinite number of strings.

Which of the following statement is true?

a) $\mathbf{P_1}$ and $\mathbf{P_2}$ are decidable

b) Neither $\mathbf{P_1}$ nor $\mathbf{P_2}$ are decidable

c) Only $\mathbf{P_1}$ is decidable

d) Only $\mathbf{P_2}$ is decidable

[GATE-2002]

Q.3 Consider the following Problem X. "Given a Turing machine M over the input alphabet Σ. Any state q of M and a word Σ^*, does the computation of M on w visit the state q" Which of the following statements about X is correct?
a) X is decidable
b) X is undecidable but partially decidable
c) X is undecidable and not even partially decidable
d) X is not a decision problem

[2002:2 Marks]

Q.4 Which of the following is true?
a) The complement of a recursive language is recursive.
b) The complement of a recursively enumerable language is recursively enumerable.
c) The complement of a recursive language is either recursive or recursively enumerable.
d) The complement of a context-free language is context-free.

[GATE-2002]

Q.5 A single tape Turing machine M has two states q_0 and q_1, of which q_0 is the starting state. The tape alphabet of M is $\{0, 1, \beta\}$ and its input alphabet is $\{0,1\}$. The symbol β is the blank symbol used to indicate end of an input string. The transition function of M is described in the following table.

	0	1	β
q_0	$q_1, 1, R$	$q_1, 1, R$	Halt
q_1	$q_1, 1, R$	$q_1, 1, L$	q_0, β, L

The table is interpreted as illustrated below.

The entry $(q_1, 1, R)$ in row q_0 and column 1 signifies that if M is in state q_0 and reads 1 on the current tape square, then it writes 1 on the same tape square, moves its tape head one position to the right and transitions to state q_1

Which of the following statements is true about M?

a) M does not halt on any string in $(0+1)^+$

b) M does not halt on any string in $(00+1)^+$

c) M halts on all strings ending in a 0

d) M halts on al l strings ending in a 1

[GATE-2003]

Q.6 If the strings of a language L can be effectively enumerated in lexicographic (i.e., alphabetic) order, which of the following statements is true?

a) L is necessarily finite
b) L is regular but not necessarily finite
c) L is context-free but not necessarily regular
d) L is recursive but not necessarily context-free

[GATE-2003]

Q.7 Nobody knows yet, if P=NP. Consider the language L de as follows:

$$L = \begin{cases} (0+1)^* \text{ if } P = NP \\ \phi \text{ otherwise} \end{cases}$$

Which of the following statement is true?

a) L is recursive
b) L is recursively enumerable but not recursive
c) L is not recursively enumerable
d) Whether L is recursive or not will be known after we find out if P = NP

[GATE-2003]

Q.8 Let L_1 be a recursive language, and let L_2 be a recursively enumerable but not a recursive language. Which one of the following is true?

a) $\overline{L_1}$ is recursive and $\overline{L_2}$ is recursively enumerable.
b) $\overline{L_1}$ is recursive and $\overline{L_2}$ is not recursively enumerable
c) $\overline{L_1}$ and $\overline{L_2}$ are recursively enumerable
d) $\overline{L_1}$ is recursively enumerable and $\overline{L_2}$ is recursive

[GATE-2005]

Q.9 Consider three decision problems P_1, P_2 and P_3 – it is known that P_1 is decidable and P_2 is undecidable. Which one of the following is true?

a) P_3 is decidable if P_1 is reducible to P_3
b) P_3 is undecidable if P_3 is reducible to P_2
c) P_3 is undecidable if P_2 is reducible to P_3
d) P_3 is decidable if P_3 is reducible to P_2's complement.

[GATE-2005]

Q.10 Let L_1 be regular language, L_2 be a deterministic context-free language and L_3 a recursively enumerable, but not recursive, language. Which one of the following statements is false?

a) $L_1 \cap L_2$ is a deterministic CFL
b) $L_3 \cap L_1$ is recursive
c) $L_1 \cup L_2$ is context free
d) $L_1 \cap L_2 \cap L_3$ is recursively enumerable

[2006:2 Marks]

Q.11 For $S \in (0+1)^*$ let $d(s)$ denotes the decimal value of s (e.g;d(101)= 5) Let $L = \{s \in (0+1)^* \mid d(s) \bmod 5 = 2$ and $d(s) \bmod 7 \neq 4\}$

Which one of the following statements is true?

a) L is recursively enumerable but not recursive
b) L is recursive but not context-free
c) L is context-free but not regular
d) L is regular

[GATE-2006]

Q.12 Which of the following problem is undecidable?

a) Membership problem for CFGs
b) Ambiguity problem for CFGs
c) Finiteness problem for FSAs
d) Equivalence problem for FSAs

[2007:1 Marks]

Q.13 Which of the following is true for the language $\{a^p \mid p \text{ is a prime}\}$?

a) It is not accepted by a Turning Machine
b) It is regular but not context-free
c) It is context -free but not regular
d) It is neither regular nor context free, but accepted by a Turning machine

[2008:1 Marks]

Q.14 Which of the following are decidable?

1. Whether the intersection of two regular languages in infinite
2. Whether a given context-free language is regular
3. Whether two push-down automata accept the same language
4. Whether a given grammar is context-free

a) 1 and 2 b) 1 and 4
c) 2 and 3 d) 2 and 4

[2008:1 Marks]

Q.15 If L and L' are recursively enumerable, then L is

a) regular b) context-free
c) context-sensitive d) recursive

[GATE-2008]

Q.16 Let L_1 be a recursive language Let L_2 and L_3 be languages that are recursively Enumerable but not recursive. Which of the following statements is not necessarily true?

a) L2 – L1 is recursively enumerable.
b) L1 – L3 is recursively enumerable
c) L2 ∩ L1 is recursively enumerable
d) L2 ∪ L1 is recursively enumerable

[GATE-2010]

Q.17 Which of the following pairs have DIFFERENT expressive power?

a) Diterministic finite automata (DFA) and non-deterministic finite automata (NFA)
b) Deterministic push down automata (DPDA) and non-deterministic push down automata (NPDA)
c) Deterministic single-tape Turing machine and Non-deterministic single-tape Turing machine
d) Single-tape Turing machine and multi-tape Turing machine

[2011:1 Marks]

Q.18 Which of the following problem are decidable?

1. Does a given program ever produce an output
2. If L is a context-free language, then is $\bar{L}$ also context-free?
3. If L is a regular language, then is $\bar{L}$ also regular
4. If L is a recursive language, then is $\bar{L}$ also recursive?

a) 1,2,3,4 b) 1,2
c) 2,3,4 d) 3,4

[2012:1 Marks]

Q.19 Which of the following is/are undecidable?

1. G is CFG. Is $L(G) = \varnothing$?
2. G is a CFG. Is $L(G) = \Sigma^*$?
3. M is a Turing machine. Is L(M) regular?
4. A is a DFA and N is an NFA. Is L(A)=L(N)?

a) 3 only b) 3 and 4 only
c) 1, 2 and 3 only d) 2 and 3 only

[2013:1 Marks]

Q.20 Which of the following statements Is/are false?

1. For every-non-deterministic Turing machine, there exists an equivalent deterministic Turing machine.
2. Turing recognizable languages are closed under union and complementation.

3. Turing decidable languages are closed under intersection and complementation.
4. Turing recognizable languages are closed under anion and intersection.

a) 1 and 4 b) 1 and 3
c) Only 2 d) Only 3

[GATE-2013]

Q.21 Let L be a language and $\bar{L}$ be its complement. Which one of the following is NOT a viable possibility?

a) Neither L nor $\bar{L}$ is recursively enumerable (r.e.)
b) One of L and $\bar{L}$ is r.e. but not recursive; the other is not r.e.
c) Both L and $\bar{L}$ are r.e. but not recursive.
d) Both L and $\bar{L}$ are recursive

[GATE-2014]

Q.22 Let $<M>$ be the encoding of a Turing machine as a string over $\{0, 1\}$.
Let $L = \{<M> | M$ is a Turing machine that accepts a string of length 2014$\}$. Then, L is

a) decidable and recursively enumerable
b) undecidable but recursively enumerable
c) undecidable and not recursively enumerable
d) decidable but not recursively enumerable

[GATE-2014]

Q.23 Let $A \leq_m B$ denotes that language A is mapping reducible (also known as many-to-one reducible) to language B. Which one of the following is FALSE?

a) If $A \leq_m B$ and B is recursive then A is recursive.
b) If $A \leq_m B$ and A is undecidable then B is undecidable.
c) If $A \leq_m B$ and B is recursively enumerable then A is recursively enumerable.
d) If $A \leq_m B$ and B is not recursively enumerable then A is not recursively enumerable.

[GATE-2014]

Q.24 Let Σ be a finite non-empty alphabet and let 2^{Σ^*} be the power set of Σ^*. Which one of the following is TRUE?

a) Both 2^{Σ^*} and Σ^* are countable
b) 2^{Σ^*} is countable and Σ^* is uncountable
c) 2^{Σ^*} is uncountable and Σ^* is countable
d) Both 2^{Σ^*} and Σ^* are uncountable

[GATE-2014]

Q.25 Which one of the following problems is undecidable?

a) Deciding if a given context-free grammar is ambiguous.
b) Deciding if a given string is generated by a given context-free grammar.
c) Deciding if the language generated by a given context-free grammar is empty.
d) Deciding if the language generated by a given context-free grammar is finite.

[GATE-2014]

Q.26 Consider the following statements.

I. The complement of every Turing decidable language is Turing decidable
II. There exists some language which is in NP but is not Turing decidable
III. If L is a language in NP, L is Turing decidable

Which of the above statements is/are true?

a) Only II b) Only III
c) Only I and II d) Only I and III

[GATE-2015]

Q.27 For any two languages L1 and L2 such that L1 is context-free and L2 is recursively enumerable but not recursive, which of the following is/are necessarily true?

I. $\overline{L_1}$ (complement of L1) is recursive

II. $\overline{L_2}$ (complement of L2) is recursive

III. $\overline{L_1}$ is context € free

IV. $\overline{L_1} \cup L2$ is recursively enumerable

a) I only b) III only
c) III and iv only d) I and IV only

[GATE-2015]

Q.28 Consider the following languages.

- $L_1 = \{\langle M\rangle \mid M \text{ takes at least 2016 steps on some input}\}$
- $L_2 = \{\langle M\rangle \mid M \text{ takes at least 2016 steps on all inputs}\}$ and
- $= L_3\{\langle M\rangle M \text{ accepts } \varepsilon\}$

Where for each turning machine M, $\langle M\rangle$ denotes a specific encoding of M.

Which one of the following is TRUE?

a) L_1 is recursive and L_2, L_3 are not recursive

b) L_2 is recursive and L_1, L_3 are not recursive

c) L_1, L_2 are recursive and L_3 is not recursive

d) L_1, L_2, L_3 are recursive

[GATE-2016(2)]

Q.29 Let X be a recursive language and Y be a recursively enumerable but not recursive language. Let W and Z be two languages such that Y reduces to W, and Z reduces to X (reduction means the standard many-one reduction). Which one of the following statements is TRUE?

a) W can be recursively enumerable and Z is recursive.

b) W can be recursive and Z is recursively enumerable.

c) W is not recursively enumerable and Z is recursive.

d) W is not recursively enumerable and Z is not recursive.

[GATE-2016]

Q.30 Consider the following types of languages:

L1: Regular,
L2:Context-free,
L3:Recursive
L4: Recursively enumerable.

Which of the following is/are TRUE?

I. $L_3' \cup L_4$ is recursively enumerable

II. $L_2 \cup L_3$ is recursive

III. $L_1^* \cap L_2$ is context-free

IV. $L_1 \cup L_2'$ is context-free

a) I only b) I and III only
c) I and IV only d) I, II and III only

[GATE-2016]

Q.31 Let L(R) be the language represented by regular expression R. Let L(G) be the language generated by a context free grammar G. Let L(M) be the language accepted by a Turning machine M.

Which of the following decision problems are undecidable ?

I. Given a regular expression R and a string w, is $w \in L(R)$?

II. Given a context-free grammar G, is $L(G) = \Phi$?

III. Given a context-free grammar G, is $L(G) = \sum^*$ for some alphabet $\sum$?

IV. Given a Turning machine M and a string w, is $w \in L(M)$?

a) I and IV only b) II and III only
c) II,III and IV d)III and IV only

[GATE-2017(2)]

Q.32 Let A and B finite alphabets and let # be a symbol outside both A and B. Let f be a total function from A*to B*.

We say f is computable if there exists a Turing machine M which given an input x in A*, always halts with f(x) on its tape. Let L_f denote the language {x#(f(x))|x ∈ A*}.

Which of the following statements is true:

a) f is computable if an only if L_f is recursive
b) f is computable if and only if L_f is recursively enumerable.
c) If f is computable then L_f is recursive, but not conversely.
d) If f is computable then L_f is recursively enumerable, but not conversely.

[GATE-2017(2)]

Q.33 The set of all recursively enumerable languages is

a) closed under complementation.
b) closed under intersection.
c) a subset of the set of all recursive languages.
d) an uncountable set.

[GATE-2018]

ANSWER KEY:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
(b)	(a)	(b)	(a)	(a)	(d)	(a)	(b)	(c)	(b)	(d)	(b)	(d)	(b)	(d)
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
(b)	(b)	(d)	(d)	(c)	(c)	(b)	(d)	(c)	(a)	(d)	(d)	(c)	(c)	(d)
31	32	33												
(d)	(a)	(b)												

EXPLANATIONS

Q.1 (b)

The C language is context sensitive language. All the feature of C language comes under CFL except following two which makes the C language as CSL:

a) Declaration of variable before use
b) The matching of actual and formal parameters.

Q.2 (a)

P_1 is membership of FSM i.e., regular language. P_2 is finite/infiniteness of context free grammar. Membership of FSM and finiteness of CFG, both are decidable problems.

So P_1 and P_2 both are decidable.

Q.3 (b)

Problem X is state entry problem. Halting problem of turing machine can be reduced to state entry problem. Since halting problem of turing machine is undecidable , the state-entry problem must also be undecidable. The language corresponding to state-entry problem is RE but not REC. So X is undecidable but partially decidable.

Q.4 (a)

Option (a) True: The complement of recursive language is always recursive.

Option (b) False: The recursively enumerable language is the language, when taken its complement; lose its recursively enumerable nature.

Option (c) False: The complement of recursive language is always recursive and never recursively enumerable.

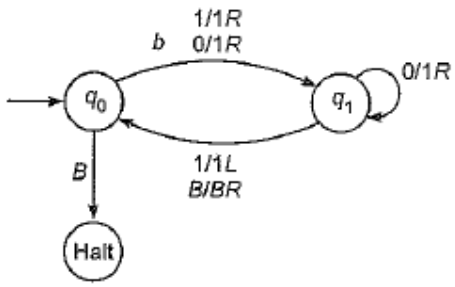
Option (d) False: The complement of context free language is never context free.

Q.5 (c)

Transition function of M

	0	1	B
q_0	$q_1, 1, R$	$q_1, 1, R$	Halt
q_1	$q_1, 1, R$	$q_1, 1, R$	q_1, B, L

The graphical representation of M is as



The original configuration of tape of M is as follows

$BBBq_0000$ ……. 111 ……. BBB

$BBB1q_100$ …….. 111 ……. BBB

$\lfloor S(q_0, 0 = q_1, 1, R \rfloor$

$BBB1q_100$ ……. 111 …….. BBB

$BBB11q_100$ …….. 111 ……. BBB

$\lfloor S(q_0, 0) = q_1, 1, R \rfloor$

If tape is finite, then after K right move all 0's will be changed to 1's and M remains in state q_1.

BBB 111 …….$q_1$111 ……. BBB

BBB 111 …….$q_0$1111 ……. BBB

$\lfloor S(q_1, 1) = q_0, 1, L \rfloor$

BBB 111 …….1q_1 11111 ……. BBB

$\lfloor S(q_0, 1) = q_1, 1, R \rfloor$

BBB 111 …….$q_0$1111 ……. BBB

$\lfloor S(q_1,1)=q_0,1,L \rfloor$

So, machine M moves one step right and one step left and goes in infinite loop

So M cannot reach a configuration $\delta(q_0,B)$ and M does not halt on any string in $(0+1)^+$.

Q.6 (d)

The strings of a language L can be effectively enumerated means a Turing machine exists for language L which will enumerate all valid strings of the language. If the string is in lexicographic order then TM will accept the string and halt in the final state. But, if the string is not lexicographic order then TM will reject the string and halt in non-final state.

Thus, L is recursive language.

We cannot construct PDA for language L. So, the given language is not context free. Thus, option (d) is correct.

Q.7 (a)

Consider the case where $L=(0+1)^*$

Here the answer of any Turing machine would be Yes since some output is displayed.

Consider the case where $L=\varnothing$

Here, the answer of any Turing machine would be No, since no output is displayed.

This type of behaviour is displayed by the recursive language and therefore, the language L is recursive.

Q.8 (b)

Since, L_1 is REC, clearly $\overline{L}_1$ is also REC.

Since, L_1 is RE but not REC, its complement has to be RE.

∴ But (a) claims that $\overline{L}_2$ is RE. Therefore, (a) is false.

Consider (b) $\overline{L_1}$ as REC & $\overline{L}_2$ is not RE:

If L_1 is REC, then $\overline{L_1}$ is REC.

Also, if L_2 is RE but not REC, then $\overline{L_2}$ is not RE.

Therefore (b) is true.Clearly, (c) and (d) false since $\overline{L_2}$ is not RE.

Q.9 (c)

According to the option, P2 is reducible to P3. Also it gives that P2 is undecidable. From (a) and (b), P_3 is undecidable.

Q.10 (b)

Given L_1 be regular, L_2 is DCFL and L_3 is RE but not Rec.

1. "$L_1 \cap L_2$ is a DCFL" is true since DCFL'sw are closed under regular intersection.
2. "$L_3 \cap L_1$ is recursive" is false since L_3 is RE but nt REC and therefore $L_3 \cap L_1$ is surely. RE according to closure of RE under regular intersection, but we cannot be sure that $L_3 \cap L_1$ is REC.
3. "$L_1 \cup L_2$ is CFL" is true, since L_2 is DCFL and hence a CFL and L_1 is regular. Therefore, by closure of CFL's on regular union, we can say that $L_1 \bigcup L_2$ is surely CFL.
4. "$L_1 \cap L_2 \cap L_3$ is RE" is true since all three are surely RE and RE languages are closed under intersection.

Q.11 (d)

Given that,

$d(s) \bmod 5 = 2$ and $d\{s) \bmod 7 \neq 4$

Both the languages are regular languages, since there exists deterministic finite automata for both of them. We also have an algorithm to check the regular

nature of the language therefore L is regular.

Q.12 (b)

a) Membership problem for CFG's is decidable (CYK algorithm exists).
b) Ambiguity problem of CFG's is undecidable.
c) Finiteness problem of FSA's is decidable, since there exist an algorithm to check if a given regular language L is finite or infinite.
d) Equivalence problem for FSA's is decidable, since there exist an algorithm to check where $L(M_1) = L(M_2)$ or not.

Q.13 (d)

Let $L = \{a^P \mid p \text{ is a prime}\}$. There is no DFA which recognizes L, since to check whether a number is prime or not requires division to be performed, which neither FA nor PDA can do. It is a CSL and can be accepted by a turing machine.

Q.14 (b)

1. "Intersection of two regular languages is infinite" is decidable, since we can construct a product DFA of the 2 given DFAS and then, using the algorithm to check finiteness or infiniteness on this DFA, we can Qve the problem.
2. "Whether a given CFL is regular" is undecidable.
3. "Whether two PDA's accept the same language" is undecidable, since equivalence of CFL's is undecidable.
4. "Whether a given grammar is context-free" is decidablesince we can easily check using a TM, whether the LHS of every production has a single variable and, then it is a CFG, else it is not a CFG.

∴ 1 and 4 are decidable.

Q.15 (d)

As per the theorem, a language is recursive if that language and its complement are recursively enumerable. Therefore, L is recursive.

Q.16 (b)

(a)$L2-L1=L2\cap L1'$

Recursive languages are closed under complement, and so L1' is recursive and hence recursively enumerable also. So, $L2\cap L1'$ is recursively enumerable is always TRUE.

(b)$L1-L3=L1\cap L3'$

Recursively enumerable languages are not closed under complement. So, L3' may or may not be recursively enumerable and hence we can't say anything if $L1\cap L3'$ is recursively enumerable or not.

Option (c) Intersection of two recursive enumerable is recursive enumerable.

Option (d) Union of two recursive enumerable is recursive enumerable.

This implies that option (b) is incorrect

Q.17 (b)

$DFA \equiv NFA$
$DPDA \neq NPDA$
$DTM \equiv NTM$
Single tape TM ≡ Multi-tape TM

Q.18 (d)

Statement (1) is undecidable (since it is the halting problem of TM).

Complement of a context free language, may or may not be context free. So statement number (2) is not only a non trival problem but is also undecidable..

If L is regular, $\overline{L}$ is surely regular.
So statement number (3) is (trivially) decidable. Also if L is recursive, $\overline{L}$ is also surely recursive. So statement number (4) is also (trivially) decidable.
So, correct option is (d) i.e. only 3 and 4 are decidable.

Q.19 (d)

1. G is CFG, is $L(G)=\varnothing$?
2. This is the emptiness problem of CFLs which is decidable.
3. G is CFG, is $L(G)=\Sigma^*$.
   This is the Kleene closeness problem of CFL's which is undecidable.
4. M is turning machine, Is $L(M)$ regular. This is the regularity problem of REs which is undecidable.
5. A is a DFA and N is NFA, Is $L(A)=L(N)$?.

This is the equivalence problem of regular languages which is decidable.
So, only 2 and 3 are undecidable.

Q.20 (c)

1. Non - deterministic Turing Machine can be simulated by a deterministic Turing Machine with exponential time true.
2. Turing recognizable language are "not" closed under complementation. For any Turing recognizable language the Turing Machine T recognizing 'L' may not terminate on inputs $\chi \notin L$ – False
3. Turing decidable languages are CLOSED under union and complementation. It is easy to determine if Turing machine is decidable – True

So, answer is option (c) only 2.

Q.21 (c)

If both L and its complement are Re then both are REC.

Q.22 (b)

There are finite numbers of strings of length '2014'. So, a turing machine will take the input string of length '2014' and test it. If, input string is present in the language then turing machine will halt in final state. But, if turing machine is unable to accept the input string then it will halt in non-final state or go in an infinite loop and never halt.
Thus, 'L' is undecidable and recursively enumerable

Q.23 (d)

$A \leq_m B$ means language A is mapping reducible to language B. Thus, A cannot be harder than B. Since, A can be reduced to B, instead of deciding A we can now decide B. So, the first three options are correct.
As B is not recursively enumerable, it doesn't guarantee A is not recursively enumerable. Thus, if $A \leq_m B$ and B is not recursively enumerable then A is not recursively enumerable. Therefore, answer is D is correct.

Q.24 (c)

2^{Σ^*} is uncountable and Σ^* is countable

Q.25 (a)

Ambiguity is not an operation and hence we can never say that CFG is closed under ambiguity. Thus, deciding ambiguity of CFG is undecidable.

Q.26 (d)

I and III are true. The definition of NP itself says Qvable in polynomial time

using non-deterministic Turing machine.

Q.27 (d)

1. L1' (complement of L1) is recursive is true L1 is context free. Every context free language is also recursive and recursive languages are closed under complement.
4. L1' ∪ L2 is recursively enumerable is true Since L1' is recursive, it is also recursively enumerable and recursively enumerable languages are closed under union.
3. L1' is context-free: Context-free languages are not closed under complement, intersection, or difference.
2. L2' (complement of L2) is recursive is false: Recursively enumerable languages are not closed under set difference or complementation

Q.28 (c)

Consider L1 = {<M>| M halts in < 2017 steps on some input}. Then L' = {<M>|M takes >= 2017 steps for all inputs}. Now whether or not a Turing Machine makes 2017th move depends on the state it is in after the 2016th move and the content of the first 2016 cells of the input tape(as the Turing machine could have processed the atmost n cells after n moves).

However the state that a Turing machine is after the 2016th move depends on its state after the 2015th move and the contents of the first 2015 cells.

Reasoning like this all the way back to before the first move was made we can conclude that whether the Turing machine will make 2017th move depends on its initial state and the contents of the first 2016 cells of the tape.

Now the first 2016 cells can have |Σ + 1|^2016 (+1 for the blank symbol) different combinations. Out of these only some combinations will be valid(eg. if the first square is blank then any subsiquent square can't have an input symbol).

So we can simulate the machine on all these valid combinations and if it does not make 2017th move in any of those combinations then we accept <M> in L1' and reject it otherwise. Hence L1' and thus L1 is Recursive. Similarly, we can prove that L2' is Recursive. L3 is not Recursive(Rice Theorem).

Q.29 (c)

$X \rightarrow$ recursive

$Y \rightarrow$ REL but not recursive

Y reduces to W

Z reduces to X

By the following theorem:

Theorem: If P1 reduces to P2 then,

- If P1 is Non-Recursively enumerable then so is P2.
- If P2 is Non-Recursively enumerable then so is P1.
- If P1 is Recursive then so is P2.
- If P2 is Recursive then so is P1.

Z is recursive and W is REL but not recursive.

Q.30 (d)

L1: Regular

L2: CFL

L3: Recursive language

L4: REL

(i) L_3' is Recursive

L_3' ∪ L4 is REL

(ii) L_2' is recursive

L_3' is recursive

L_2' $\cup$ L_3 is also recursive

(iii) L_1* is regular

L_2 is CFL

L_1* $\cap$ L_2 is CFL

Since CFL is closed under intersection with RL.

RL $\cap$ CFL is CFL

(iv) L_1 is RL & L_2' is recursive

$L_1 \cup L_2$' is CFL is not possible

So (i), (ii), & (iii) are correct

Q.31 (d)

I. Membership of regular language (Decidable)

II. Emptiness of CFL (Decidable)

III. L=$\sum$* problem of CFL(Undecidable)

IV. Membership of RE language (Undecidable)

So ,only III and IV are undecidable .So, correct answer is (d).

Q.32 (a)

If f is computable and let M be the Turing machine that computes it then when given a string of the form x#y we can calculate f(x) using M and accept it if y = f(x) or else reject it. Hence if f is computable then L is recursive. Now assume that L is recursive. Then we can list all of it strings in lexciographical order. Then when we are given x we can compare it with the left part of each string(the part before #). Once we find a match we return the right hand portion as f(x). Hence if L is recursive then f is computable. Hence f is computable iff L is recursive.

ASSIGNMENT QUESTIONS

Q.1 The word "formal" in formal languages
means
a) the symbols used have well-defined meaning
b) they are unnecessary, in reality
c) only the form of the string of symbols is significant
d) none of the above

Q.2 Let A = {0, 1} the number of possible strings of length "n" that can be formed by the elements of the set A is
a) n! b) n^2
c) n^n d) 2^n

Q.3 Choose the correct statements
a) Moore and Mealy machines are FSM's with output capability.
b) Any given Moore machine has an equivalent Mealy machine.
c) Any given Mealy machine has an equivalent Moore machine.
d) Moore machine is not an FSM.

Q.4 The major difference between a Moore and a Mealy machine is that
a) The output of the former depends on the present state and the current input
b) The output of the former depends only on the present state
c) The output of the former depends only on the current input
d) None of the above

Q.5 Choose the correct statements.
a) A Mealy machine generates no language as such
b) A Moore machine generates no language as such
c) A Mealy machine has no terminal state.
d) For a given input string, length of the output string generated by a Moore machine is one more than the length of the output string generated by that of a Mealy machine.

Q.6 The recognizing capability of NDFSM and DFSM
a) may be different
b) must be different
c) must be the same
d) none of the above

Q.7 FSM can recognize
a) any grammar
b) only CFG
c) any unambiguous grammar
d) only regular grammar

Q.8) Pumping lemma is generally used for proving
a) a given grammar is regular
b) a given grammar is not regular
c) whether two given regular expressions are equivalent
d) none of the above

Q.9 Which of the following are not regular?
a) string of 0's whose length is a perfect square
b) set of all palindromes made up of 0's and 1's
c) strings of 0's, whose length is a prime number
d) string of odd number of zeroes

Q.10 Which of the following pairs of regular expressions are equivalent?
a) 1(01) * and (10) * 1
b) x(xx) * and (xx) * x
c) (ab) * and a*b
d) x^+ and x^*x^+

Q.11 Choose the correct statements.
a) A = {$a^n b^n$|n=0, 1, 2, 3,.....} is a regular language.
b) The set B, consisting of all strings made up to only a's and b's having equal number of a's and b's defines a regular language.
c) L(a*b*) ∩ B gives the set A.
d) none of the above

Q.12 Pick the correct statements.
The logic of pumping lemma is a good example of
a) The Pigeon-hole principle
b) The divide & conquer technique
c) Recursion
d) iteration

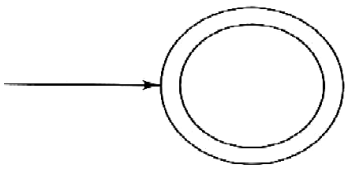
Q.13 The basic limitation of an FSM is that
a) it can't remember arbitrary large amount of information
b) it sometimes recognizes grammars that are not regular
c) it sometimes fails to recognize grammars that are regular
d) all of the above

Q.14 Palindromes can't be recognize by any FSM because
a) an FSM can't remember arbitrarily large amount of information
b) an FSM can't deterministically fix the mid-point
c) even if the mid-point is known, an FSM can't find whether the second half of the string matches the first half
d) None of the above

Q.15 An FSM can be consider a TM
a) of finite tape length, rewinding capability and unidirectional tape movement
b) of finite tape length, without rewinding capability and unidirectional tape movement
c) of finite tape length, without rewinding capability and bidirectional tape movement
d) of finite tape length, rewinding capability and bidirectional tape movement

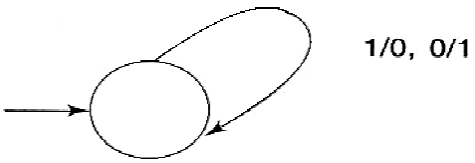
Q.16 TM is more powerful than FSM because
a) the tape movement is confined to one direction
b) it has no finite state control
c) it has the capability to remember arbitrary long sequence of input symbols.
d) none of the above

Q.17 The FSM pictured in below figure recognizes



a) all strings b) no string
c) ε- alone d) none of above

Q.18 The FSM pictured in below shown fig is a



a) Mealy machine
b) Moore machine
c) Keene machine
d) none of the above

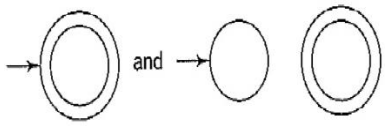
Q.19 The above machine
a) complements a given bit pattern
b) generates all strings of 0's and 1's
c) adds 1 to a given bit pattern
d) none of the above

Q.20 The language of all words (made up of a's and b;s) with at least two a's can be described by the regular expression
a) (a+b)*a(a+b)*a(a+b)*
b) (a+b)*ab*a(a+b)*

c) b* ab* a(a+b)*
d) a(a+b)* a(a+b) * a(a+b)*

Q.21 Which of the following pairs of regular expression are not equivalent?
a) (ab) * a and a(ba) *
b) (a+b)* and (a+b*)*
c) (a* +b)* and (a+b)*
d) none of the above

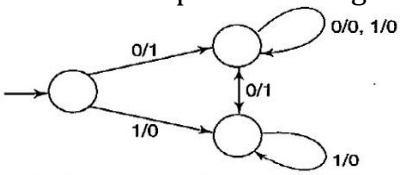
Q.22 Consider the two FSM's in fig.



Pick the correct statement
a) Both are equivalent
b) The second FSM accepts only ε
c) The first FSM accepts nothing
d) None of the above

Q.23 Set of regular languages over a given alphabet set, is not closed under
a) union b) complementation
c) intersection d) none of above

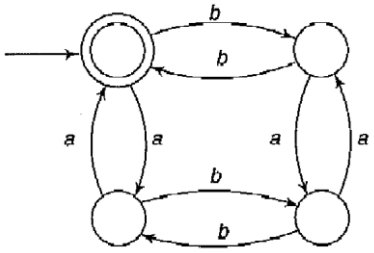
Q.24 The machine pictured in fig.



a) Complements a given bit pattern
b) Finds 2's complement of a given
c) Increments a given bit pattern by 1
d) Change the sign bit

Q.25 For which of the following application regular expressions can't be used?
a) Designing compliers
b) Developing text editors
c) Simulating sequential circuits
d) Designing computers

Q.26 The FSM pictured in fig. recognizes



a) Any string of odd number of a's
b) Any string of add number of a's and even number of b's
c) Any string of even number of a's and even number of b's
d) Any string of even number of a's and odd number of b's

Q.27 Any given transition graph has an equivalent
a) regular expression
b) DFSM
c) NDFSM
d) none of the above

Q.28 The following CFG S →aS|bS|a|b
Is equivalent to the regular expression
a) (a* + b)* b) (a + b)+
c) (a + b)(a + b)* d) (a + b)* (a + b)

Q.29 Any string of terminals that can be generated by the following CFG
S →XY
X →aX |bX|a
Y →Ya |Yb|a
a) Has at least one b
b) should end in an 'a'
c) Has no consecutive a's or b's
d) has at least two a's

Q.30 The following CFG
S→aB|bA
A →a|aS|bAA
B→b|bS|aBB
Generates strings of terminals that have
a) Equal number of a's and b's
b) Odd number of a's and odd number b's
c) Even number of a's and even number of b's

d) Odd number a's and even number of a's

Q.31 Let L(G) denote the language generated by the grammar G. to prove set A=L(G)
a) it is enough to prove that an arbitrary member of A can be generated by grammar G
a) It is enough to prove that an arbitrary string generated by G, belongs to set A
b) Both the above comments (a) and (b) are to be prove
c) Either of the above comments (a) or (b) is to be proved

Q.32 The set $\{a^nb^n \mid n = 1, 2, 3.....\}$ can be generated by the CFG
a) S→ ab | aSb
b) S→aaSbb | ab
c) S →ab | aSb |ε
d) S→aaSbb | ab | aabb

Q.33 Choose the correct statements.
a) All languages can be generated by CFG
b) Any regular language has an equivalent CFG
c) Some non-regular languages can't be generated by any CFG
d) Some regular languages can't be generated by any CFG

Q.34 Which of the following CFG's can't be simulated by an FSM?
a) $S \rightarrow Sa \mid a$
$S \rightarrow abX$
b) $X \rightarrow cY$
$Y \rightarrow d \mid aX$
c) $S \rightarrow aSb \mid ab$
d) None of the above

Q.35 CFG is not closed under
a) union
b) Kleene star
c) complementation
d) concatenation

Q.36 The set $A = \{a^nb^na^n \mid n = 1, 2, 3, ...\}$ is an example of grammar that is
a) regular
b) context free
c) not context free
d) none of the above

Q.37 Let $L1 = \{a^nb^na^m \mid m, n = 1, 2, 3, ...\}$
$L2 = \{a^nb^ma^m \mid m, n = 1, 2, 3, ...\}$
$L3 = \{a^nb^na^n \mid n = 1, 2, 3, ...\}$
Choose the correct statements,
a) $L3 = L1 \cap L2$
b) L1 and L2 are CFL but L3 is not a CFL
c) L1 and L2 are not CFL but L3 is a CFL
d) L1 is a subset of L3

Q.38 $L = \{a^nb^na^n \mid n = 1,2,3,.....\}$ is an example of a language that is
a) context free
b) not context free
c) not context free but whose complement is CF
d) context free but whose complement is not CF

Q.39 The intersection of a CFL and a regular language
a) need not be regular
b) need not be context free
c) is always regular
d) is always CF

Q.40 A PDM behaves like an FSM when the number of auxiliary memory it has is
a) 0 b) 1
b) 2 d) none of above

Q.41 A PDM behaves like a TM when the number of auxiliary memory it has is
a) 0 b) 1 or more
c) 2 or more d) none of the above

Q.42 Choose the correct statements.
a) The power of DFSM and NDFSM are the same

b) The power of DFSM and NDFSM are different
c) The power of DPDM and NDPDM are different
d) The power of DPDM and NDPDM are the same

Q.43 Which of the following is accepted by an NDPDM, but not by a DPDM?
a) All strings in which a given symbol is present at least twice.
b) Even palindromes
c) Strings ending with a particular terminal.
d) none of the above

Q.44 CSG can be recognized by a
a) FSM
b) DPDM
c) NDPDM
d) linearly bounded memory machine

Q.45 Choose the correct statements.
a) An FSM with 1 stack is more powerful than an FSM with no stack
b) An FSM with 2 stack is more powerful than an FSM with 1 stack
c) An FSM with 3 stack is more powerful than an FSM with 2 stack
d) All of these.

Q.46 Choose the correct statements
a) An FSM with 2 stacks is as powerful as a TM
b) DFSM and NDFSM have the same power
c) A DFSM with 1 stack and an NDFSM with 1 stack have the same power
d) A DFSM with 2 stack and an NDFSM with 2 stacks have the same power

Q.47 Bounded minimization is a technique for
a) proving whether a primitive recursive function is turning computable
b) proving whether a primitive recursive function is a total function
c) generating primitive recursive functions
d) generating partial recursive functions

Q.48 Which of the following is not primitive recursive but computable?
a) Carnot function
b) Riemann function
c) Bounded function
d) Ackermann function

Q.49 Which is the following is not primitive recursive but partially recursive?
a) Carnot function
b) Riemann function
c) Bounded function
d) Ackermann function

Q.50 Choose the correct statements.
a) A total recursive function is also a partial recursive function
b) A partial recursive function is also a total recursive function
c) A partial recursive function is also a primitive recursive function
d) A primitive recursive function is also a partial recursive function

Q.51 A language L for which there exists a TM, T that accepts every word in L and either rejects or loops for every word that is not in L is said to be
a) recursive b) recursively enumerable
c) NP-HARD d) none of the above

Q.52 Choose the correct statements
a) $L = \{ a^n b^n a^n \mid n = 1, 2, 3, \ldots\ldots\}$ is recursively enumerable
b) recursive languages are closed under union

c) every recursive language is recursively enumerable
d) recursive languages are closed under intersection

Q.53 Choose the correct statements
a) Set of recursively enumerable languages is closed under union
b) if a language and its complement are both regular then language must be recursive
c) recursive languages are closed under complementation
d) none of the above

Q.54 pick the correct answers. Universal TM influenced the concept of
a) Stored-program computers
b) Interpretive implementation of programming languages
c) Computability
d) None of the above

Q.55 The number of internal states of a UTM should be at least
a) 1 b) 2
c) 3 d) 4

Q.56 The number of symbols necessary to simulate a TM with m symbols and n states is
a) m + n b) 8mm + 4m
c) mn d) 4mn + m

Q.57 Any TM with m symbols and n states can be simulated by another TM with just 2 symbols and less than
a) 8mn states b) 4mn + 8 states
b) 8mn + 4 states d) mn states

Q.58 The statements – "A TM can't solve halting problem" is
a) true
b) false
c) still an open question
d) none of the above

Q.59 If there exists a TM which when applied to any problem in the class, terminates if the correct answer is yes and may or may not terminate otherwise is said to be
a) stable b) unsolved
c) partially solved d) unstable

Q.60 The number of states of the FSM required to simulate the behavior of a computer, with a memory capable of storing 'm' words each of length 'n' bits is
a) $M \times 2^n$ b) 2^{mn}
c) 2^{m+n} d) none of the above

Q.61 The vernacular language English, if considered a formal language is a
a) regular language
b) context free language
c) context sensitive language
d) none of the above

Q.62 Let P, Q and R be three languages. If P and R are regular and if PQ = R then
a) Q has to be regular
b) Q cannot be regular
c) Q need not be regular
d) Q has to be a CFL

Q.63 Consider the grammar
$S \longrightarrow PQ \mid SQ \mid PS$
$P \longrightarrow x$
$Q \longrightarrow y$
To get a string of n terminals the number of production to be used is
a) n^2 b) n + 1
c) 2n d) 2n – 1

Q.64 Choose the correct statements.
A class of languages that is closed under
a) Union and complementation has to be closed under intersection
b) Intersection and complementation has to be closed under union
c) Union and intersection has to be closed under complementation
d) Al of the above

Q.65 The following grammar is
S→ a α b|b α c|aB
S→ aS|b
S→ α bb| ab
bα→bdb| b
a) Context free b) regular
c) context sensitive d) LR (k)

Q.66 Which of the following definitions generates the same language as L, where
L = $\{x^n y^n, n \geq 1\}$?
I. E →xEy | xy
II. xy | x^+xyy^+
III. x^+y^+
a) I only b) I and II
c) II and III d) II only

Q.67 A finite states machine with the following state table has a single input X and a single output Z

Present State	Next State, Z	
	x=1	x=0
A	D,0	B,0
B	B,1	C,1
C	B,0	D,1
D	B,1	C,0

If the initial state is unknown then the shortest input sequence to reach the final state C is
a) 01 b) 10
c) 101 d) 110

Q.68 Let A = {0,1} and L =A*. let R = $\{0^n 1^n, n>0\}$. The language L∪R and R are respectively
a) regular, regular
b) not regular, regular
c) regular, not regular
d) not regular, not regular

Q.69 Which of the following conversion is not possible algorithmically?
a) regular grammar to context free grammar
b) non-deterministic FSA to deterministic FSA
c) non-deterministic PDA to deterministic PDA
d) non-deterministic Turing machine to deterministic Turing machine

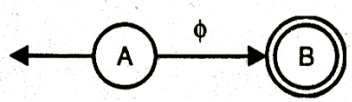
Q.70 An FSM can be used to add two given integers. This remark is
a) true b) false
c) may be true d) none of the above

Q.71 A CFG is said to be in Chomsky normal form(CNF), if all the productions are of the form A→BC or A→a. let G be a CFG in CNF. To derive a string of terminals of length x, the number of production to be said is
a) 2x – 1 b) 2x
c) 2x + 1 d) 2^x

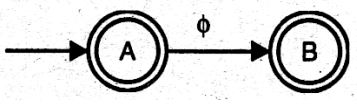
Q.72 Determine the regular expression over ∑ = {0, 1}
a) 1* 2* b) $0^+ 1^-$
c) 0 – 1 d) None of these.

Q.73 The r.e. (a + b).(a + b) represents the set
a) ∑ b) {(a + b).(a + b)}
c) {ab, ca,cb,bb} d) None of these.

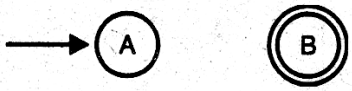
Q.74 The NFA for the r.e. ϕ is
a)



b)



c)



d) None of these.

Q.75 The regular expression for the set {ζ, ζ, ζ, ζ ζ ζ,....} is
a) 1 (1)* b) ζ *
c) ζ(ζ)* d) None of these.

Q.76 Obtain the regular expression for $L = \{a^{2n}b^{2m+1} \mid n \geq 0, m \geq 0\}$

a) $(a)^{2n}(b)^{2m+1}$
b) (aa)* (bb)* b
c) No regular expression exists for the language
d) None of these.

Q.77 Describe the set represented by the regular express (00 + 1)* (11 + 0)*.

a) The set of all strings having 0's and 1's occuring in pairs
b) The set of all strings having 0's in pairs
c) The set of all strings having 1's in pairs
d) None of these.

Q.78 Given r.e. for strings over {0, 1, 2} containing at 7 single 1.

a) (0 + 2)* 1 b) 1 (0+ 2)*
c) (0 +1 + 2)* d) None of these.

Q.79 Find the regular expression for the language
L = {w : w I mod 2 = 0}, wϵ {a, b, c}*.
a) aa+ab+ac+ba + bb+bc+ ca+cb +cc
b)(a + b + c) (a + b + c)
c) Both a) and b) above
d) None of these.

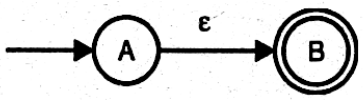
Q.80 Given P and Q two regular expression. Which of following identity hold ?

a) A* = A
b) (PQ)* P = P (QP)*
c) Both a) and b) above
d) None of these.

Q.81 Find the incorrect identity of regular expression f: the following :

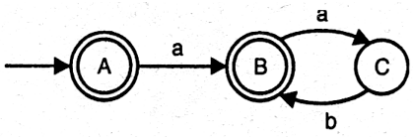
a) (R + S)* = R* + S*
b) ϕ*= ε
c) Both a) and b) above are wrong
d) None of these.

Q.82 Which of the following equation will be generated using Arden's Method to the automata ?



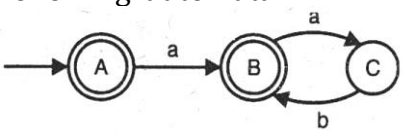
a) A= ε
b) B= A ε
c) Both a) and b) above
d) None of these.

Q.83 Which of the following equation will be generated using Arden's Method to the automata?



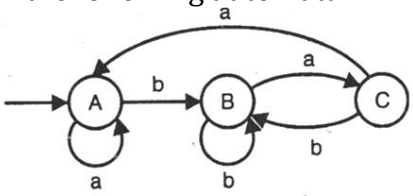
a) B = aA
b) C = aB
c) Both a) and b) above
d) None of these.

Q.84 What are the sub expression to be joined by + operator of the regular expression accepted by the following automata?



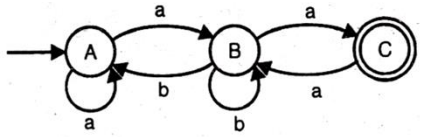
a) a b) ab
c) aab d) None of these.

Q.85 In the following automata



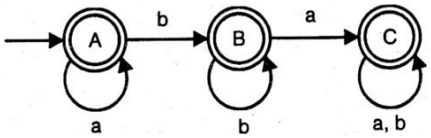
a) the r.e corresponding to the states A and B are same
b) the r.c. corresponding to the states B and C are same
c) the r.c. corresponding to the states C and A same
d) None of these

Q.86 Find the regular expression for the automata



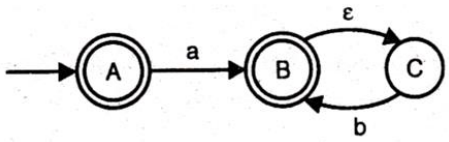
a) (a+a(b+aa)*b)*a(b+aa)*a
b) (a+a(b+aa)*b)*(b+aa)*a
c) (a+a(b+aa)*b)*a(b+aa)*
d) None of these

Q.87 Describe in English the set accepted by the following FA:

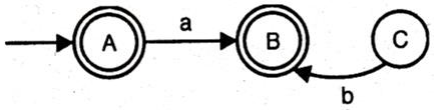


a) The strings of '0's followed by strings of 1s'
b) The strings of 0's (possibly no zero) followed by string of 1's
c) The strings of '0's followed by string of 1's (possibly no one.)
d) None of these

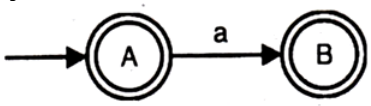
Q.88 Find the automata after removal of ε-move



a)



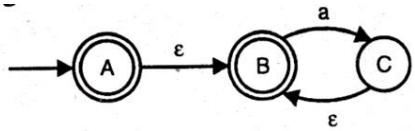
b)



c)

d) None of these

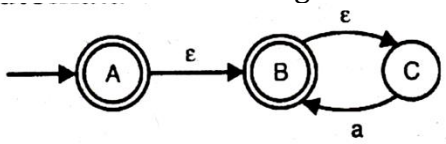
Q.89 Which ε-move



Should be removed first from the following automata?

a) A →B b) B→C
c) C→B d)Noneof these

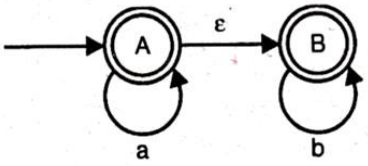
Q.90 Which ε-move should be removed first from the following automata>?



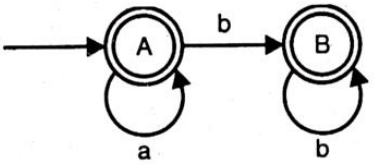
a) A →B b) B→C
c) C→B d) None of these

Q.91 Which of the following accept the language L (a*b*)?

a)

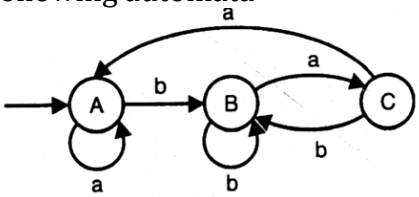


b)



c)Both a) and b) above
d) None of these

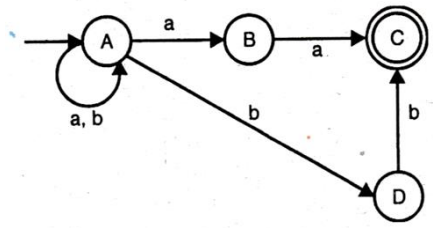
Q.92 Construct the regular expression corresponding to the state A in the following automata



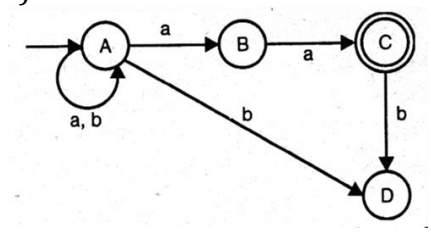
a) (a+b(b+ab)*aa)*
b) (0+1(1+01)*00)*
c) Both a) and b)
d) None of these

Q.93 Construct the FA equivalent to the regular expression (a+b)*(aa+bb).

a)

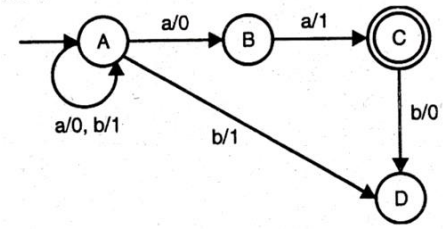


b)



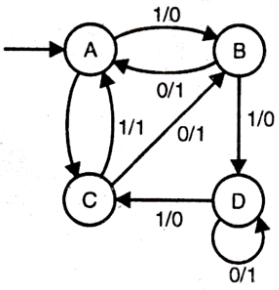
c)Both a) and b) above
d) None of these

Q.94 How many states at most will be there in the Moore machine converted from the following Mealy machine ?

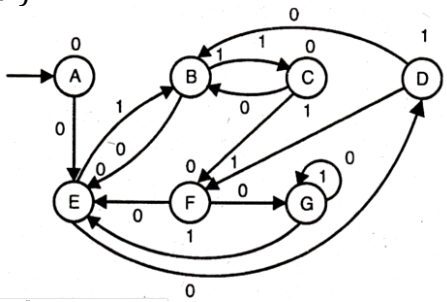


a)5 b) 6
c) 7 d) None of these

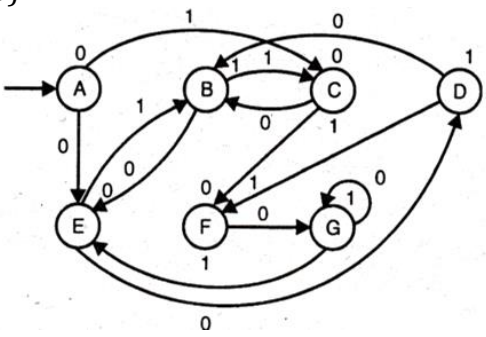
Q.95 Moore machine corresponding to the following Mealy machine is



a)



b)

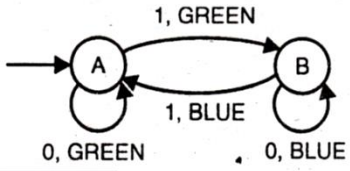


c)Both a) and b)above
d) None of these

Q.96 We may have increased number of states when
a) we convert Moore Machine to Mealy machine
b) we convert Mealy Machine to Moore Machine
c)Both a) and b) above
d) None of these

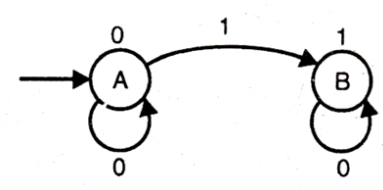
Q.97 The length of output string is … in equivalent Moore and Mealy Machines for the same input string
(A) same b) different
c) zero d) None of these

Q.98 Determine the output of the following Mealy machine .

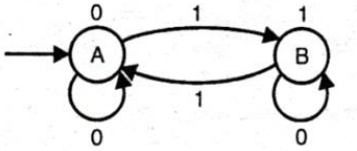


a) Green /Blue if total number of 1's encountered is odd/ even
b) Blue/Green if total number of 1's encountered is odd/ even
c) Green / Blue if total number of 0's encountered is odd/ even
d) None of these

Q.99 Determine the Moore Machine that determine residue mod 2 for each binary string treated as a binary integer.
a)

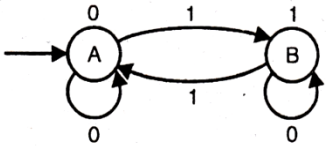


b)

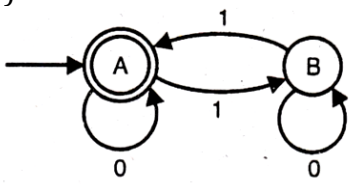


c) Both a) and b) above
d) None of these

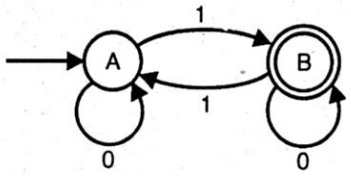
Q.100 Convert the following Moore machine to a DFA



a)

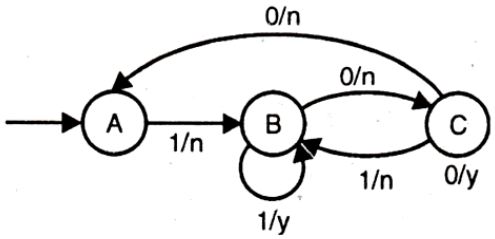


b)



c)Both a) and b) above
d) None of these

Q.101 When will you get y a output in the following Mealy Machine?



a) a pair of 0's occur
b)a pair of 1's occur
c)Both a) and b) above
d) None of these

Q.102 In a finite automata of n states , a str0ng of length more than
a) will always be accepted
b) will result in revist of a set of states
c) Both a) and b) above
d) None of these

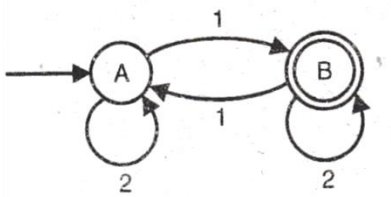
Q.103 Which of the following in a regular language?
a) $L = \{a^n b : n \geq 0\}$
b) $L = \{a^n b : n > 0\}$
c) Both a) and b) above
d) None of these

Q.104 Which of the following is a regular language?
a) $L = \{a^n b^n : n \geq 0\}$
b) $L = \{a^n b^n : n > 0\}$
c) Both a) and b) above
d) None of these

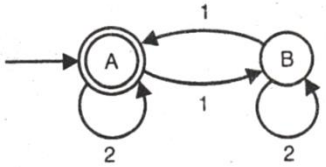
Q.105 Which of the following is a regular language?
a) L ={w:wϵ{a,b}*}
b) L ={ww:wϵ{a,b}*}
c) L ={w^3:wϵ{a,b}*}
d) None of these

Q.106 Design the automata for the language $L = \{1^n 2^n : n \geq 0\}$

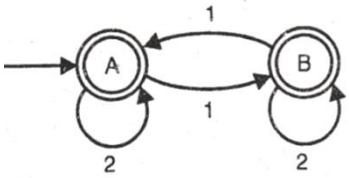
a)



b)



c)



d) None of these.

Q.107 Let L' be complement of a language. Find the language L for which corresponding L' is regular.

a) $L=\{w^2 : w \in \{a,b\}^*\}$

b) $L=\{w : w \in \{a,b\}^*\}$

c) Both a) and b) above

d) None of these.

Q.108 Is $L=\{0^{2n} : n \geq 1\}$ regular?

a) Yes b) No

c) Can't be known d) None of these.

Q.109 Is the set of prime numbers regular?

a) Yes

b) No, as it is infinite set

c) Can't be known

d) None of these.

Q.110 Let L be a regular set, Is the set $\{a1\ a3\ a5...a2n-1 : a1\ a2\ a3..a2n \text{ are in } L\}$

a) Yes b)No

c) Can't be known d) None of these.

Q.111 Is the set strings not containing two consecutive zeros regular?

a) Yes b) No

c) Can't be known d) None of these

Q.112 Given $L_1((a+b)^*a)$ and $L_2((a+b)^*b)$. Is $L_1 \cup L_2$ regular?

a) Yes b) No

c) Can't be known d) None of these

Q.113 Given $L_1((a+b)^*a) \& L_2((a+b)^*b)$ is $L_1 \cap L_2$ regular?

a) Yes b) No

c) Can't be known d) None of these

Q.114 Let $L(\varepsilon + a(a+b)^*)$ be regular language over $\Sigma=\{a,b\}$ Then L'is

a) L'(ϕ) b) L'((a+b)*)

c) L'((a+b)*b) d) None of these

Q.115 The intersection of the two regular Language $L_1((a+b)^*a)$ & $L_2(b(a+b)^*b)$ is

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

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

c) Both a) and b) above

d) None of these.

Q.116 Let a language $L(0^*(0+1)1^*)$ and a mapping f given by f(0) =a & f(1) =b*.

Find f (L).

a) $(ab)^*(ba)^*3^*$

b) $(ab)^*(ba)^*$

c) Both a) and b) above

d) None of these.

Q.117 Let L is language of regular expression 10*1, and homomorphism defined by $h(0) = ab$ and $h(1) = \varepsilon$ then h(L) is

a) h(L)(ϕ) b) h(L)(ab)

c) h(L)((ab)*) d) None of these

Q.118 Find $h^{-1}(L)$, where language is $L(a(a+b)^*)$ and homomorphism h is defined by h(0) = ab and h (1) = a.

a) $h^{-1}(L)((0+1))$

b) $h^{-1}(L)((0+1)^*)$

c) $h^{-1}(L)((0+1)(0+1))$

d) None of these.

Q.119 Let L_1/L_2 denote the Quotient of the languages L_1 and L_2. Find it for $L_1(0^*0^*)$ and $L_2(10^*1)$

a) ϕ

b) ε

c) Both a) and b) above

d) None of these

Q.120 Given $\frac{1}{2}(L)=\{x|$ for some y such that $|x|=|y|, xy$ is in $L\}$, find such L.

a) ϕ
b) ε
c) Both a) and b) above.
d) None of these.

Q.121 Find L/a for $L=\{a, aabb, baa\}$

a) $\{\varepsilon\}$ b) $\{ba\}$
c) $\{\varepsilon, ba\}$ d) None of these.

Q.122 Let $a/L=\{w: aw\ \varepsilon\ L\}$. Find a/L for $L=\{a, aab\ baa\}$

a) $\{\varepsilon\}$ b) $\{ab,\ ba\}$
c) $\{\varepsilon,\ ba\}$ d) None of these

Q.123 Which of the following identities are true?

a) $(L/a)a=L$
b) $(La)/a=L$
c) Both a) & b) above
d) None of these.

Q.124 True or False $(L/a)a=L$

a) True
b) False
c) Depends on the language L
d) None of these.

Q.125 Let half (L) = $\{w: wx\ \varepsilon\ L, |x|=|w|$ for some $x\}$.

Which of the following is correct always?

a) only odd length strings of L as in half (L)
b) only even length strings of L as in half (L)
c) Both a) and b) above
d) None of these.

Q.126 Let $L=\{w:|w|>10, w\in\{a,b\}^*\}$, then L may be accepted by an automata of at least...states

a) 8 b) 9
c) 10 d) None of these

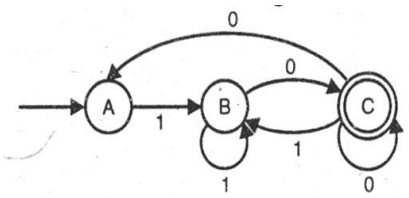
Q.127 Given a language L and a relation R_L such that xR_Ly iff for each z, either both or neither xz and yz are in L, where x ,y ϵ L. The relation R_L is.

a) irreflexive b) asymmetric
c) non-transitive d) None of these

Q.128 Let a language L(0)* and are a relation R_L as defined in above equation. Find the number of equivalence classes of L.

a) 2 b) 3
c) 1 d) None of these

Q.129 Let $M=(Q,\Sigma,\delta,qo,F)$ be a DFA and a relation RM such that xR_My if f $\delta(q0,x)=\delta(q0y)$. Find the index of the relation R_M for the following automata



a) 1 b) 2
c) 3 d) None of these.

Q.130 Let $M=(Q,\Sigma,\delta,qo,F)$ bean automat accepting the language L, which of the following relations are right invariant?

a) xRy such that ∀z either both or neither xz and yz are in L
b) xRy such that $\delta(q0,x)=\delta(q0y)$.
c) Both a) and b)above
d) None of these.

Q.131 Determine the type of the grammar $G=(V,T,P,S)$ where $V=\{v0\}$,

$T = \{a\}$ and

$P = \{v0 \to aav0, v0 \to aa\}$.

a) type 0 b) type 1
c) Type 2 d) None of these

Q.132 Which of the following is true for a type 1 `grammars having the production of the form $\beta \to \alpha$?

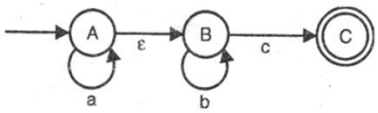
a) $|\beta| > |\alpha|$ b) $|\beta| > |\alpha|$ always
c) $|\alpha| < |\beta|$ d) none of these.

Q.133 Determine language of the grammar $S \to xS, S \to yA, A \to yA, A \to z$.

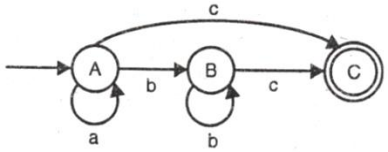
a) $L(x^*y^*z)$ b) $L(xy^*z)$
c) $L(xyz)$ d) None of these

Q.134 Determine the automata for the grammar $S \to aS, S \to bS, S \to c$.

a)



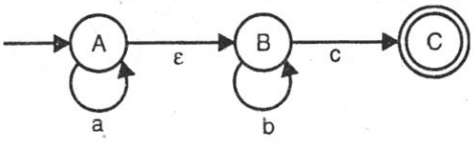
b)



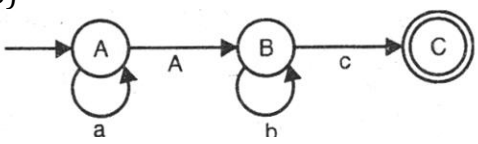
c) Botha) and b) above
d) None of these.

Q.135 Determine the automata for the grammar $S \to aSA, S \to bS, A \to c$.

a)



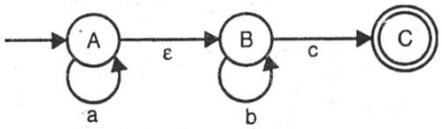
b)



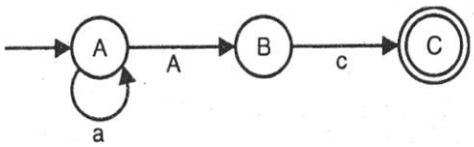
c) Both a) and b) above
d) None of these.

Q.136 Determine the automata for the grammar
$S \to aSA, S \to bS, A \to c, S \to \varepsilon$

a)



b)



c) Both a) and b) above
d) None of these.

Q.137 Let a grammar be $S \to (S), S \to a + A,$
$A \to a + B, B \to a + C, C \to a$. Is the corresponding language finite?

a) Yes
b) No
c) Can't be known as the grammar definition is not complete
d) None of these.

Q.138 Let a grammar G be $S \to (S), S \to a + A,$
$A \to a + B, B \to a + C, C \to a$
Is $(a + (a + a)) \varepsilon L(G)$?

a) Yes
b) No
c) Can't be known as the grammar definition is not complete
d) None of these.

Q.139 Let a grammar G be $S \to (S)$, $S \to a + A$, $A \to a + B$, $B \to a + C$, $C \to a$.
Find $w \varepsilon L(G)$ having the minimum length.

a) (a + a + a) b) (a + a + a +a)
c) a + a + a + a d) None of these.

Q.140 Let a grammar G be $S \to a + A$, $A \to a + B$, $B \to a + C$, $C \to a + D$, $D \to a +$ and set of terminals be $T = \{a+\}$.

Construct the automata corresponding to G.

a) Automata cannot be constructed as the grammar is of type 1
b) Automata cannot be constructed as the grammar is of type 2
c) Automata cannot be constructed as the grammar definition is incomplete
d) None of these.

Q.141 Determine the left and right context in a production $AC \rightarrow A$ of a type1 grammar.

a) A, A
b) A, A
c) Both a) and b) above
d) None of these.

Q.142 In the type 1production $\phi A \psi = \phi \alpha \Psi$

a) $\alpha \neq A$
b) erasing of A is not permitted
c) Both a) and b) above
d) None of these.

Q.143 Is the grammar S⟶ε. aSb⟶aabb, S⟶(S) of type 1?

a) Yes b) No
c) Can't be known d) None of these.

Q.144 Which of the following is a monotonic grammar?

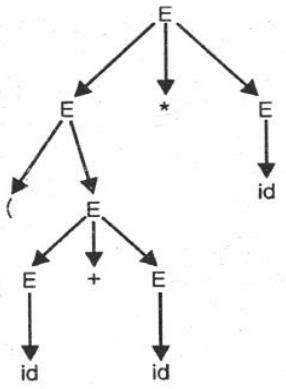
a) $aSb \rightarrow aAb, bAc \rightarrow bc$
b) $aSb \rightarrow aAb|\Lambda$, A⟶S, bAc⟶bcc
c) Both a) and b) above
d) None of these.

Q.145 Find the highest type number which can be applied to the grammar $S \rightarrow ASB|d, A \rightarrow aA$

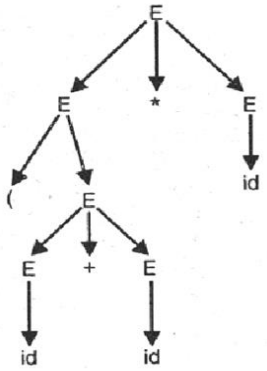
a) 0 b) 1
c) 2 d) None of these.

Q.146 Find the parse tree of $(id+id)*id$ for thee grammar $E \rightarrow E+E|E*E|(E)id.$

a)

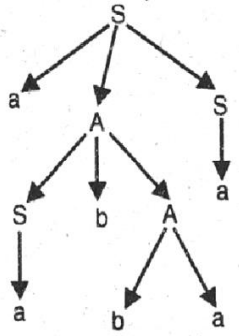


b)



c) Both a) and b) above
d) None of these.

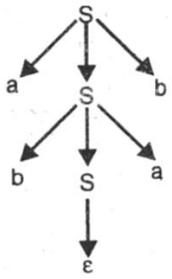
Q.147 Determine the yield of the derivation tree for the grammar $S \rightarrow aAS|a, A \rightarrow SbA|SS|ba$



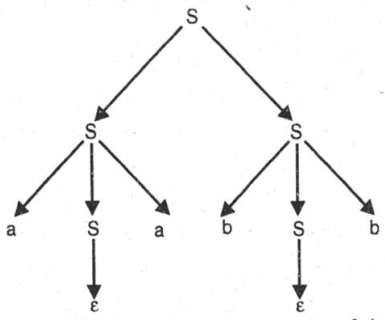
a) 2a2b2a
b) aabaa
c)Both a) and b) above
d)None of these.

Q.148 Determine the derivation tree of the string abab for the grammar $S \rightarrow aSb|bSa|SS|\varepsilon.$

a)



b)

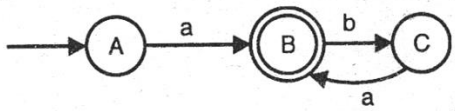


c) Both a) and b) above
d) None of these.

Q.149 Is the grammar $S \to S|S,\ S \to a$ ambiguous?
a) Yes b) No
c) Can't b known d) None of these.

Q.150 Find the sentential form of the grammar
$S \to aAS|a,\ A \to SbA|SS|ba.$
a) aabAS
b) aabbaS
c) Both a) and b)above
d) None of these.

Q.151 Construct the right-linear grammar from the following automata

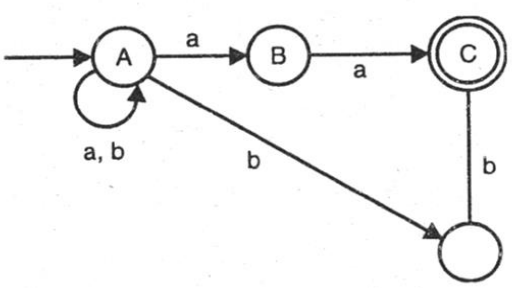


a) $S \to aA, A \to baA|\varepsilon$
b) $s \to S10|0|\varepsilon$
c) Both a) and b) above
d) None of these.

Q.152 Which variable of the grammar
$A \to 0B|1D|0, B \to 0D|1C$
$C \to 0B|1D|0, D \to 0D|1D$
Will correspond to a dead state in the automata of the grammar?
a) A b) B
c) C d) None of these.

Q.153 Find the corresponding left-linear grammar of the right-linear grammar
$S \to 0A, A \to 10A|\varepsilon.$
a) $S \to S10|\varepsilon$ b) $S \to 10\ S|\varepsilon$
c) $S \to 01\ S|0$ d) None of these.

Q.154 Construct the left-linear grammar for the automata



a) $S \to Aaa|Abb, A \to Aa|Ab|a|b$
b) $C \to Ba|Db, B \to Aa|a, D \to Ab|b,$
$A \to Aa|Ab|a|b$
c) Both a) and b) above
d) None of these.

Q.155 Construct the language generated by the left – linear grammar
$S \to Abbb \mid Sb, A \to Aa \mid aa$
a) $L\{a^n b^m : n \geq 2, m \geq 3\}$
b) $L\{a^n b^m : n \geq 2, m > 3\}$
c) Both (A) and b) above
d) None of these

Q.156 Let a grammar G be S→aSb|Λ. Find L(G).
a) $\{a^n b^n : n > 0\}$
b) $\{0^n 1^n : n \geq 0\}$
c) Both (A) and b) above
d) None of these

Q.157 The $L(G)\{a^n ba^n\}$ for G given b $S \to aAa,\ A \to aAa \mid b.$ Find the condition on n.
a) n >0
b) n ≥ 1
c) Both a) and b) above
d) None of these

Q.158 True or False L(G) = Φ
a) True b) False
c) Can't be known d) None of these

Q.159 Construct the grammar generating $L\{xax^r : w \in \{0.1\}^*\}$ Where x^r is reverse of string x
a) $A \to 0A0 \mid 1A1 \mid 0$
b) $A \to 0A0 \mid 1A1 \mid 1$
c) $A \to 0A0 \mid 1A1 \mid a$
d) None of these

Q.160 Construct the grammar of palindromes of odd length generated over {0,1}.
a) $A \to \varepsilon \mid a \mid b \mid aAa \mid bAb$
b) $A \to a \mid b \mid aAa \mid bAb$
c) $A \to \varepsilon \mid aAa \mid bAb$
d) None of these

Q.161 A regular language is also
a) a CFL b) a CSL
c) Both a) and b) d) None of these

Q.162 Given the CFG equivalent to the regular expression (011+1)*(01)*
a) $S \to BC, B \to AB \mid \varepsilon, A \to 011$
$C \to DC \mid \varepsilon, D \to 01$
b) $B \to AB \mid \varepsilon, A \to 011 \mid 1,$
$D \to DC \mid \varepsilon, D \to 01$
c) $S \to BC, B \to AB \mid \varepsilon,$
$A \to 011 \mid 1, C \to DC \mid \varepsilon$
d) None of these

Q.163 Find the grammar for {abc,bca,cab} Whose terminals are in {a,b,c}
a) $S \to acb \mid bac \mid cba$
b) $S \to aS \mid bS \mid cS \mid \varepsilon$
c) Both a) and b) above
d) None of these

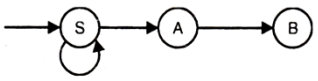
Q.164 Given a grammar G with productions $S \to aaB, A \to bBb \mid \lambda Aa$ then
a) aabbabba ∉ L(G)
b) aabbabab ∈ L(G)
c) aabbabbb ∈L(G)
d) None of these

Q.165 True or False: A regular grammar can be ambiguous
a) True b) False
c) Can't be know d) None of these

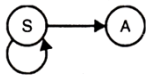
Q.166 Which variable does not drive a terminal string in the grammar
$S \to AB, A \to a, B \to b, B \to C, E \to C$
a)A b) B
c) S d) None of these

Q.167 Modify the grammar $S \to AB,$ $A \to a,$ $B \to b,$ $E \to C$ such that $\forall x \in V \cup T,\ \exists \alpha \in T^*$ and $S \overset{*}{\Rightarrow} \alpha$
a) $S \to AB, A \to a, B \to bc$
b) $S \to AB, A \to a, B \to b$
c) Both a) and b) above
d) None of these

Q.168 Determine the dependency graph of the reduced grammar for
$S \to aS \mid A \mid C, A \to a, B \to aa, C \to aCb$
a)



b)



c) Both a) and b) above
d) None of these

Q.169 Find the grammar with non-null productions from the grammar $S \to aAb, A \to aAb \mid \lambda$
a) $S \to aAb \mid ab, A \to aAb$

b) $S \to ab, A \to aAb \mid ab$

c) $S \to ab, A \to aAb$

d) None of these $S \to ab, A \to aAb$

Q.170 Determine the nullable variable in the grammar given by
$S \to ABaC, A \to BC, B \to b \mid \lambda, C \to D \mid \lambda, D \to d$

a) A b) B

c) Both a) and b) d) None of these

Q.171 Find the equivalent grammar of $S \to Aa \mid B, B \to A \mid bb, A \to a \mid bc \mid B$ free from unit – productions.

a) $S \to a \mid bc \mid bb \mid Aa, A \to a \mid bb \mid bc,$ $B \to a \mid bb \mid bc$

b) $S \to a \mid bc \mid bb \mid, A \to a \mid bb \mid bc,$ $B \to a \mid bb \mid bc$

c)Unit-free productions can't obtained

d) None of these

Q.172 Which of the following productions are in Chomsky Normal From?

a) A→a|b b) S → AB|c

c) Both a) and b) d) None of these

Q.173 Find the CNF of the grammar $S \to AB, B \to C, B \to b, C \to \lambda$

a) $S \to AB, B \to b, C \to \lambda$

b) $S \to AB, B \to C, C \to \lambda$

c) $S \to A \mid b$

d) None of these

Q.174 How many minimum new variables **will be added to convert the** grammar $S \to ABa, A \to aab, B \to Ac$ into CNF.

a) 1 b) 2

c) 3 d) None of these

Q.175 Convert the grammar $S \to ABa, A \to aab, B \to Ac$ into CNF:

a) $S \to AB, B \to CD, A \to DF, D \to a,$ $F \to DE, E \to b, C \to AG, G \to c$

b) $S \to AB, B \to CD, A \to DF, D \to a,$ $F \to DE, E \to b, C \to A$

c) Both (A) and b) above

d) None of these

Q.176 Which of the following productions are not in Greibach Normal Form?

a) $A \to AAa$ b) $A \to aAaA$

c) Both a) and b) d) None of these

ANSWER KEY:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
(c)	(d)	(a)	(b)	(a)	(c)	(d)	(B)	(a)	(a)	(c)	(a)	(a)	(a)	(b)
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
(c)	(c)	(a)	(a)	(a)	(d)	(d)	(d)	(c)	(a)	(c)	(a)	(b)	(d)	(a)
31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**
(c)	(a)	(b)	(c)	(c)	(c)	(a)	(b)	(c)	(a)	(c)	(a)	(b)	(d)	(a)
46	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**
(a)	(c)	(d)	(d)	(a)	(b)	(a)	(a)	(a)	(b)	(d)	(a)	(a)	(c)	(b)
61	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**
(b)	(c)	(d)	(a)	(c)	(a)	(b)	(c)	(c)	(b)	(a)	(d)	(d)	(c)	(c)
76	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**
(b)	(d)	(d)	(c)	(c)	(a)	(d)	(d)	(d)	(d)	(a)	(d)	(d)	(d)	(b)
91	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**
(c)	(a)	(a)	(c)	(d)	(b)	(b)	(a)	(b)	(c)	(d)	(b)	(c)	(d)	(a)
106	**107**	**108**	**109**	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**
(d)	(b)	(a)	(d)	(a)	(a)	(a)	(a)	(d)	(d)	(c)	(c)	(d)	(a)	(b)
121	**122**	**123**	**124**	**125**	**126**	**127**	**128**	**129**	**130**	**131**	**132**	**133**	**134**	**135**
(d)	(d)	(c)	(c)	(d)	(d)	(d)	(c)	(c)	(b)	(c)	(d)	(d)	(c)	(d)
136	**137**	**138**	**139**	**140**	**141**	**142**	**143**	**144**	**145**	**146**	**147**	**148**	**149**	**150**
(d)	(b)	(b)	(c)	(d)	(d)	(c)	(b)	(d)	(c)	(d)	(d)	(c)	(a)	(c)
151	**152**	**153**	**154**	**155**	**156**	**157**	**158**	**159**	**160**	**161**	**162**	**163**	**164**	**165**
(a)	(d)	(d)	(c)	(a)	(d)	(c)	(a)	(c)	(b)	(c)	(d)	(d)	(a)	(a)
166	**167**	**168**	**169**	**170**	**171**	**172**	**173**	**174**	**175**	**176**				
(d)	(b)	(b)	(b)	(c)	(c)	(c)	(d)	(c)	(a)	(c)				

EXPLANATIONS

Q.6 (c)
DFSM is a special case of NDFSM. Corresponding to any given NDFSM, one can construct an equivalent DFSM. Corresponding to any given DFSM, one can construct an equivalent NDFSM. So they are equally powerful.

Q.9 (a, b, c)
Strings of odd number of zeroes can be generated by the regular expression
(0 0)*0. Pumping lemma can be used to prove the non-regularity of the other

Q.10 (a)
Two regular expression R1 and R2 are equivalent if any string that can be generated by R1 can be generated by R2 and vice-versa. In (c), (ab)* will generate abab, which is not of the form $a^n b^n$ (because a's and b's should come together). All other s are correct (check it out)

Q.12 (a)
Pigeon-hole principle is that if 'n' balls are put in 'm' boxes, then at least one box will have more than one ball if n>m. Though this is obvious, still powerful.

Q.13 (a)
That's why it can't recognize strings of equal number of a's and b's, well-formed ness of nested parenthesis etc.

Q.17 (c)
Here the final state and start state are one and the same. No transition is there. But by definition, there is an (implicit) Ɛ-transition from any state to itself. So, the only string that could be accepted is Ɛ.

Q.22 (d)
In the second diagram, the final state is unreachable from the state. So not even Ɛ could be accepted.

Q.24 (c)
Let 011011 be the input to the FSM and let it be fed from the right (i.g., Least significant digit first). If we add 1 to 011011 we should get 011100. But did we obtain it? Whenever we add 1 to an 1, we make it 0 and carry 1 to the next stage (state) and repeat the process. If we add 1 to a 0, then first make it 1 and all the more significant digits will remain the same, i.g., a 0 will be 0 and an 1 will be 1.
That's what the given machine does. Hence the answer is (c).

Q.26 (c)
Here the initial and the final state are one and same. If you carefully examine the transition diagram, to move right you have to consume a 'b', to move left a 'b', to
go up an 'a' and to go down an 'a'. Whenever we move right, we have to move left at some stage or the other, to get back to the initial-cum-final state. This implies, a 'b' essentially has an associated another 'b'. Same is the case with 'a' (since any up (down) has a corresponding down (up)). So even number of a's and b's have to be present.

Q.29 (d)
S is the start state. X→a, Y→a are the only production that could terminate a string derivable from X and Y respectively. So at least two a's have to come anyway. Hence the answer is (d).

Q.30 (a)
We have S→aB → aaBB → aabB → aabb.
So (b) is wrong.
A careful observation of the productions will reveal a similarity. Change A to B, B to A, a to b and b to a. The new set of productions will be the same as the original set. So (d) is false and (a) is the correct answer.

Q.32 (a)
(b) is wrong because it can't generate aabb (in fact any even power). (c) is wrong since it generates Ɛ also. Both (a) and (d) are correct

Q.34 (c)
(c) generates the set $\{a^n b^n, n = 1, 2, 3,....\}$ which is not regular. (a) and (b) being left linear and right linear respectively. Should have equivalent regular expressions.

Q.60 (b)
Totally there are mn bits. Each bit will be in one of the two possible states- 1 or 0. So the entire memory made up of mn bits will be in one of the possible 2^{mn} states.

Q.62 (c)
For example, p = a*; Q = $a^n b^n b^*$; R = PQ = a*b*

Q. 64 (a)
The first two positions can be proved to be correct using De Morgan's laws. (c) can be disproved by the following counter-example. Let the universal set ∪ be {a, b, c, d}. Let A = {{a}, {d}, {a, d}, {b, d}, {a, b, d}, {}}. A is closed under union and intersection but is not closed under complementation. For example complement of {a, d} is {b, c}, which is not a member of A.

Q. 66 (a)
II generates strings like XXYYY, which are not supposed to be III generates strings like XYY, which are not supposed to be. I can be verified to generate all the strings in L and only those.

Q. 67 (b)
Draw the transition diagram and verify that the using 1 0 from A, leads to C.

Q. 68 (c)
L is the set of all possible strings made up of 0's and 1's (including the null string).
So L ∪ R is L, which can be generated by the regular expression (a + b)*, and hence a regular language. R is not a regular expression. This can be proved by using Pumping Lemma or simply by the fact that finite state automata, that recognizes regular expressions, has no memory to record the number of 0's or 1's it has scanned. Without this information $0^n 1^n$ cannot be recognized.

Q. 69 (c)
In general, a language (or equivalently the machine that recognize it) cannot be converted to another language that us less powerful

Q. 70 **(b)**
FSM is basically a language acceptor. As such, it does not have any output capability. So it cannot add and output the result.

Q. 71 **(a)**
This can be proved using induction

Q.72 **(d)**
The regular expression must contain only members of input alphabet set and predefined operators.

Q.73 **(d)**
The expression represents the set {ab, bb, ac, bc}.

Q.74 **(c)**
Follows from definition.

Q.75 **(c)**
The members of input alphabets set can be anything, even the symbol ∑ itself.

Q.76 **(b)**
Language L contains the strings having any number of pairs of a's followed by at least one but odd number of b's.

Q.77 **(d)**
The set of all strings of the xy, where 0's are in pairs in x and l's are in pairs in y.

Q.78 **(d)**
The regular expression is (0+2)*+(0 +2)* 1(0+2)*.

Q.79 **(c)**
The **(b)** when expanded will result in **(a)**, and the strings are of length exactly two.

Q.80 **(c)**
Follows from the definitions of regular expressions.

Q.81 **(a)**
The correct identity is (R+S)*=(R*+S*)*.

Q.82 **(d)**
An important assumption in the method is non-existence of e moves.

Q.83 **(d)**
A term on the right hand side of the equation should have a state followed by a input alphabet.

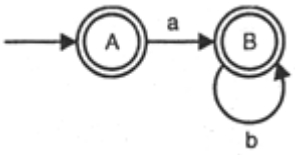
Q.84 **(d)**
The sub expression are a and a(ab)*.

Q.85 **(d)**
None of the states are equivalent, so their corresponding r.e.'s cannot be same.

Q.86 **(a)**
On solving the equations A = Aa + Bb + A, B = Aa + Bb + Ca and C = Ba using Arden's theorem, the expression will be obtained.

Q.87 **(d)**
The strings of any number of 0's (possibly A) followed by string of any number of l's (possibly A).

Q.88 **(d)**
The automata is :



Q.89 **(d)**
Here either of the two can be removed.

Q.90 (b)
The moves should be removed first, which does not have ε-moves from the destination state.

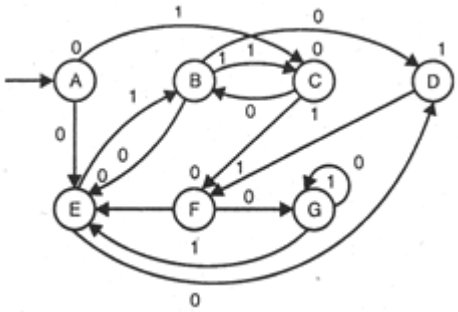
Q.91 (c)
As **(b)** above is obtained from **(a)** by the removal of ε-moves.

Q.92 (a)
Solve the equation A = Aa + Ca + A, B =Ab+Bb+Cb and C=Ba using Arden's theorem.

Q.93 (a)
We may consume any number of a's and b's after that a pair of either a's or b's must occur, so as to reach the final =state.

Q.94 (c)
If we have m-output, n-state Mealy machine, the corresponding in-output Moore machine has no more than in n+1 states. Here stages B and D need not be splitted.

Q.95 (d)
The corresponding Mealy machine is :



Q.96 (b)
Because we split the states which correspond to different outputs in this conversion.

Q.97. (b)
The length of output string in Moore Machine is one more than the string in corresponding Mealy Machine for the same input string.

Q.98 (a)
Every time we reach to state A even number of total I's are encountered and for state B they must be odd.

Q.99 (b)
The odd 1 in the input string will take us to state B and even 1 back to state A, as they result in residue 1 and 0 respectively.

Q.100 (c)
A state q can be designated as accepting/rejecting state
iff λ(q)=0/1.

Q.101 (d)
Either two consecutive 1's or two consecutive 0's must occur.

Q.102 (b)
In a finite automata with n states, the states must be joined by least n - 1 arcs. Whenever n^{th} input alphabet is encountered, any one state will be revisited, and is true for all the following input alphabets. This also holds if the number of arcs are increased.

Q.103 (c)
In fact **(a)** and **(b)** above are a *b and a^+ b respectively, which are obviously regular.

Q.104 (d)
The automata corresponding to a regular expression could not remember anything.

Q.105 (a)
The language in **(a)** above contains all strings over { a, b }

Q.106 (d)
The language is not regular and no automata can be designed for such languages.

Q.107 (b)
For **(b)** above L' = ϕ, which is regular.

Q.108 (a)
For n≥ 1, we can write 0^{2n} as 0 $(0^2)^i$ 0, where i ≥0, hence the language is $\{(0^2)^i : i \geq 0\}$, which is clearly regular.

Q.109 (d)
Use pumping lemma to prove the set to be non regular.

Q.110 (a)
In the automata for L, add ε-moves from δ(A, a_i) to δ (B, a_i +1), where i=1 (2) m, m is a natural number.

Q.111 (a)
As the set of strings containing two zeros regular and its complement set is a regular set.

Q.112 (a)
Union of regular sets is regular. Hence $L_1 \cup L_2$ ((a + b)*) is regular.

Q.113 (a)
Here $L_1 \cap L_2$ (ϕ), and is regular.

Q.114 (d)
The language is L' (b (a + b)*)

Q.115 (d)
$L_1 \cap L_2$ (ϕ)

Q.116 (c)
a* (a + b*) (b*)*
= a* a(b*)*+a* b*(b*)*
= a* ab*+aT b* b*
= a*ab* + a*b* = a*b*

Q.117 (c)
Here h drops the 1's, since they are replaced by ε, and turns each 0 into ab.

Q.118 (d)
The language is given by the regular expression (0 + 1) (0 + 1)*.

Q.119 (a)
Since every y in L_2 has two 1's and every string xy is in L_1 can have only one 1, there is no x such that xy is in L_1 and y is in L_2.

Q.120 (b)
As|ε|= 0 and ε.ε = ε.

Q.121 (d)
L/ a = { ε, ba}

Q.122 (d)
a/L = {E, ab}.

Q.123 (c)
Follows from the definition of quotient languages.

Q.124 (c)
Let L={a,aab,&la}, then (L/a)**(a)**={ε,a,ba.a)≠L. We could have taken L={a,baa} as other cases.

Q.125 (d)
The fact is odd-length strings of L do not contribute to half (L).

Q.126 (d)
The number of states must be greater than 10.

Q.127 (d)
The relation is an equivalence relation.

Q.128 (c)
The relation RL is independent of the choice of z for this L.

Q.129 (c)
The index will always be equal to number of states in the automata.

Q.130 (b)
Every finite automata induces such right invariant relation.

Q.131 (c)
All productions contain a variable on the left hand side and one or more variables or terminals on the right hand side.

Q.132 (d)
For the grammar $|\beta| \leq |\alpha|$

Q.133 (d)
The language is L (x* yy*z).

Q.134 (c)
The automata in **(b)** above is obtained from automata in **(a)** above by removing ε-move.

Q.135 (d)
The definition of the grammar is not complete.

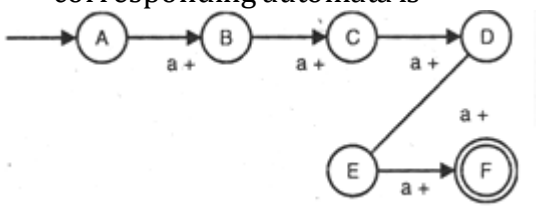
Q.136 (d)
The grammar of type 2, and for such grammars automata does not exist.

Q.137 (b)
We can introduce infinite number of pairs of braces in the strings of the language by the first production, hence the language is infinite.

Q.138 (d)
Braces can be introduced only in the beginning and the end.

Q.139 (c)
Braces are optional, and the production of A, B, C introduces at one a each.

Q.140 (d)
Set of terminals contains only one element denoted by 'a+'. The corresponding automata is



Q.141 (d)
For a production $\phi A \psi \rightarrow \phi \alpha \psi$, left and right contexts are ϕ and ψ respectively. Hence here ϕ=A, ψ=A and α = A.

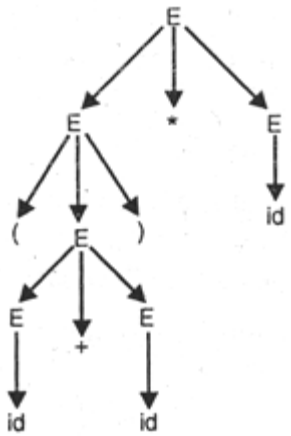
Q.142 (c)
Follows from the Chomsky Classifications.

Q.143 (b)
In type 1 grammars the production S→A can be included provided S does not appear on right hand side of any of the production of the grammar.

Q.144 (d)
For the grammars all productions are of the form $\alpha \rightarrow \beta$ with $|\alpha| \leq |\beta|$ or $S \rightarrow A$, in the second case S does not appear on the right-hand side of any production of the grammar.

Q.145 (c)
S→ASB is type 2, S→d, A→aA are type. Therefore, the highest type number is 2.

Q.146 (d)
The parse tree is:



Q.147 (d)

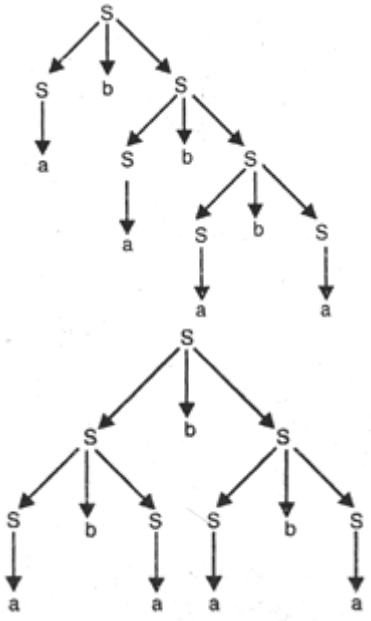
The yield is aabbaa and is obtained by the taking leaf nodes from left to right of the derivation tree.

Q.148 (c)

Both the trees has the same yield abab.

Q.149 (a)

The string abababa has two derivation trees as shown below:



Q.150 (c)

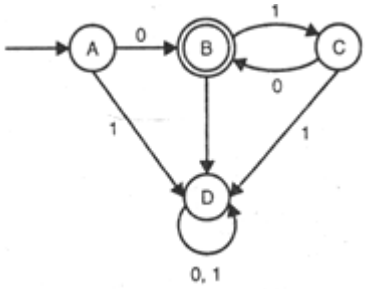
The sentential forms are obtained in the derivation of the string aabbaa.

Q.151 (a)

Grammar in **(a)** is obtained from the automata directly and the variables have been renamed.

Q.152 (d)

The corresponding automata is shown below and the dead state is D:



Q.153 (d)

The corresponding left linear grammar is S →S10|0.

Q.154 (c)

The grammar is **(b)** above is directly obtained from the automata and in **(a)** it is further simplified.

Q.155 (a)

As production S→Abbb introduces at least three b's and production A→4 aa introduces at least two a's in the derived strings.

Q.156 (d)

The L (G) = $\{a^n b^n: n \geq 0\}$, as the production S →A is present.

Q.157 (c)

After the first production if we apply A→b, we get aba, otherwise equal number of a's are added on both side of b.

Q.158 (a)

Consider the grammar S→SS having no terminal on the right hand side, which will generate language ϕ.

Q.159 (c)
The strings in L are such that nth symbol from the start and end are same.

Q.160 (b)
In a palindrome nth symbol from either end is same, and in the end of the derivation the variable must be replaced by a terminal, to get the palindrome of odd length.

Q.161 (c)
Let $\mathcal{L}_0$, $\mathcal{L}_{csl}$, $\mathcal{L}_{efl}$ and $\mathcal{L}_{rl}$ denote the family of type 0 languages, context sensitive languages, context free languages and regular languages, then $\mathcal{L}_0 \supseteq \mathcal{L}_{csl} \supseteq \mathcal{L}_{efl} \supseteq \mathcal{L}_{rl}$

Q.162 (d)
The grammar is S→BC, B→AB|ε, A→011|1, C→DC|ε, D → 01.

Q.163 (d)
The grammar will have the productions S→AB|bcC|CaD, A→a, B→bc, C→a, D→b.

Q.164 (a)
The grammar cannot generated strings having aa followed by bb in the beginning.

Q.165 (a)
Let the grammar be S→aA|aB, A→a, B→b, then for the string aa we have two derivation trees.

Q.166 (d)
The variables are C and E as C does not have a occur on left hand side of any production and also we cannot have E in any sentential form.

Q.167 (b)
The symbol E & c does not occur in any sentential form.

Q.168 (b)
The corresponding reduced grammar is S →aS| A, A →a.

Q.169 (b)
The productions are obtained by substituting for A wherever it occurs on the right hand side.

Q.170 (c)
The nullable variables in the grammar are B and C as we have productions B→A, C→A which further makes A as nullable as we have another production A -3 BC.

Q.171 (c)
For each unit production A B add production A→y1|y2|....|yn to the new grammar, where B→y1|y2|....|yn are the all productions with B on the left.

Q.172 (c)
The productions are in CNF if they are of the form A→BC or A →a.

Q.173 (d)
Only a reduced CFG without λ and unit productions can be converted to CNF.

Q.174 (c)
The new variable added will be one each for a terminal and we have three terminals.

Q.175 (a)
Follows from the CNF construction.

Q.176 (c)
All productions in GNF are of the form A→a α, where a ∈ T and a is ϕ or ∈ V*.