Theory of Computation

UNIT 5

Chapter 1

Finite Automata and Regular Languages

LEARNING OBJECTIVES

- *Fundamentals*
- *Languages*
- *Operations*
- *Finite state machine*
- *NFA with ⍰-moves*
- *Conversion of NFA to DFA*
- *Minimization of DFA*
- *Equivalence between NFA and DFA*
- *Mealy and Moore machines*
- *Equivalence of Moore and Mealy machine*
- *Regular languages*
- *Constructing FA for given RE*
- *Pumping lemma for regular sets*
- *Closure properties of regular sets*
- *Regular grammar*

FUNDAMENTALS

Alphabet: An alphabet is a finite non-empty set of symbols.

Example: Portion of a calculator: {0, 1, 2, 3 … 9, ÷, =, –, +, ×, (,)}

Note: 1. At least one symbol is necessary.

2. 'Σ' denote Alphabet.

String: A string over an alphabet 'A' is a finite ordered sequence of symbols from 'A'. The length of string is number of symbols in string, with repetitions counted.

Example: If Σ = {0 – 9, ÷, =, –, +, × (,)} then Strings valid: 12 + 34, 90 × 10, (1 + 2) × (1 ÷ 3)

Strings Invalid: sin (45), log (10) etc. These strings are not valid because sin ( ), log ( ) are not defined over the alphabet set.

Note: Repetitions are allowed.

Length of |12 + 34| = 5(1, 2, +, 3, 4)

- The Empty string denoted by 'ε', is the (unique) string of length zero.

Note: Empty string, $\varepsilon \neq$ empty set, $\varnothing$.

- If S and T are sets of strings, then $ST = \{xy | x \in S \text{ and } y \in T\}$

  Given an alphabet A,

  $A^o = \{\varepsilon\}$

  $A^{n+1} = A.A^n$

  …

  $A^* = \bigcup_{n=0}^{\infty} A^n$

Languages

- A language 'L' over Σ is any finite or infinite set of strings over Σ.
- The elements in L are strings – finite sequences of symbols.
- A language which does not contain any elements is called 'empty language'.

Note: Empty language, { } ≠ {ε}, empty string because { } = $\varnothing \neq \varepsilon$ i.e., Empty language resembles empty set i.e., $\varnothing$.

- A language L over an alphabet A is subset of A^* i.e., $L \subset A^*$.

Example 1: Language (L) for strings that consists of only 0's or only 1's and have an odd length over alphabet {0, 1} is

(A) {0, 1, 00, 11, 000, 111 …}

(B) {00, 11, 01, 10 …}

(C) {000, 101, 110, 111 …}

(D) {0, 1, 000, 111, 11111, 00000 …}

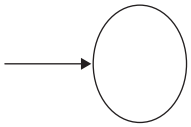
Solution: (D)

Only 0's → should have only 0's. It should not be combination of 0's and 1's.

Only 1's → should have only 1's. It should not be combination of 0's and 1's.

Odd length → only odd number of 0's or odd number of 1's i.e., length of string should be odd.

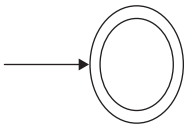
An Empty Languages An empty language is a language which does not accept any strings includinge. The Finite automata for empty language can be represented as



(i.e., One state, non-accepting and no transitions).

A language which only accepts (ε)

E: The language which only accepts 'ε' can be represented as



This machine accepts E – only.

Σ^*: The set of all strings over an alphabet Σ will be denoted by Σ^*.

Σ^+: This will denote the set $\Sigma^* - \{\varepsilon\}$.

Ex: If $\Sigma = \{0, 1\}$ then

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

$\Sigma^+ = \{0, 1, 00, 01, 10, 11, 000, 001,\}$

Operations

Operations on strings

1. **Concatenation:** Combines two strings by putting one after other.

Example 2: Two strings are defined as x = java, y = script. The concatenation ($x.y$) of two strings results in _____.

(A) scriptjava (B) javascript
(C) jascriptva (D) scrijavapt

Solution: (B)

$x.y$ = java.script = javascript

Note: Concatenation of empty string with any other string gives string itself.

i.e., $x.\varepsilon = \varepsilon.x = x$

2. **Substring:** If 'w' is a string, then 'v' is a substring of 'w' if there exists string x and y such that $w = xvy$.

'x' is called 'prefix' and y is called suffix of w.

Example 3: String, w = 'gymnastics' is defined with prefix, x = '*gym*' and suffix, y = '*cs*'. The substring of the given string is _____

(A) nasti (B) mnas
(C) gymnastics (D) ics

Solution: (A)

Because, $w = xvy$

$\Rightarrow$ gymnastics = gymvcs

$\therefore v$ = nasti

3. **Kleen star operation:** Let 'w' be a string, w^* is set of strings obtained by applying any number of concatenations of w with itself, including empty string.

Example: $a^* = \{\varepsilon, a, aa, aaa, \ldots\}$

4. **Reversal:** If 'w' is a string, then w^R is reversal of string spelled backwards.

Rules:

- $x = (x^R)^R$
- $(xz)^R = z^R \cdot x^R$

Example 4: A string, x is defined as, x = butter. Then $(x^R)^R$ is _____

(A) butter (B) rettub
(C) butret (D) retbut

Solution: (A)

$x \rightarrow$ butter

$x^R \rightarrow$ rettub

$(x^R)^R \rightarrow$ butter.

Operations on languages

1. **Union:** Given some alphabet Σ, for any two languages, L_1, L_2 over Σ, the union $L_1 \cup L_2$ of L_1 and L_2 is the language, $L_1 \cup L_2 = \{w \in \Sigma^* | w \in L_1 \text{ or } w \in L_2\}$
2. **Intersection:** Given some alphabet Σ, for any two languages L_1, L_2 over Σ, the intersection $L_1 \cap L_2$ of L_1 and L_2 is language, $L_1 \cap L_1 = \{w \in \Sigma^* | w \in L_1 \text{ and } w \in L_2\}$
3. **Difference:** Given some alphabet Σ, for any two languages L_1, L_2 over Σ, the difference $L_1 - L_2$ of L_1 and L_2 is language, $L_1 - L_2 = \{w \in \Sigma^* | w \in L_1 \text{ and } w \notin L_2\}$

Note: Difference is also called 'Relative Complement.'

A special case of difference is obtained when $L_1 = \Sigma^*$, in which case. Complement $\overline{L}$ of language, L is defined as, $\overline{L} = \{w \in \Sigma^* | w \notin L\}$

4. **Concatenation:** Given an alphabet Σ, for any two languages L_1, L_2 over Σ, the concatenation L_1 L_2 of L_1 and L_2 is language

$L_1L_2 = \{w \in \Sigma^* | \exists\ u \in L_1, \exists\ v \in L_2, w = uv\}$

Properties:

$L\varnothing = \varnothing = \varnothing L$

$L\ \{\varepsilon\} = L = \{\varepsilon\}\ L$

$(L_1 \cup \{\varepsilon\})L_2 = L_1L_2 \cup L_2$

$L_1\ (L_2 \cup \{\varepsilon\}) = L_1L_2 \cup L_1$

$L^n\ L = LL^n = L^{n+1}$

Note: $L_1L_2 \neq L_2L_1$

Example 5: Let $L_1 = \{00, 11\}$, $L_2 = \{01, 10\}$. Then $L_1 o L_2$ = _____

(A) {00, 11, 01, 10}
(B) {0001, 0010, 1101, 1110}
(C) {0001, 0010, 11, 01, 10}
(D) {00, 1101, 1110, 11, 10}

Solution: (B)

$L_1 o L_2$ = {00, 11} o {01, 10} = {00.01, 00.10, 11.01, 11.10} = {0001, 0010, 1101, 1110}

5. **Kleen * closure (L^*):** Given an alphabet Σ, for any language L over Σ, the * closure L^* of L is language, $L^* = U_{n \geq 0} L^n$

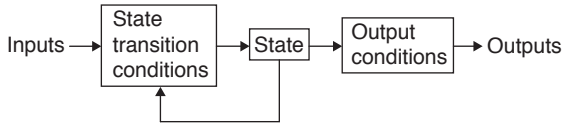
6. **Kleen + closure (L^+):** The kleen +closure, L^+ of L is the language, $L^+ = U_{n\geq 1} L^n$

$L^* = L^0 \cup L^1 \cup L^2 \cup \ldots L^n \cup \ldots$
$L^+ = L^1 \cup L^2 \cup L^3 \ldots \cup L^n \cup \ldots$

Properties:
$\varnothing^* = \{\varepsilon\}$
$L^+ = L^* L$
$(L^*)^* = L^*$
$L^* L^* = L^*$

Finite State Machine (FSM)

- FSM is simplest computational model of limited memory computers.
- FSM is designed to solve decision problems i.e., to decide whether given input satisfies certain conditions.
- The next state and output of a FSM is a function of input and of current state.

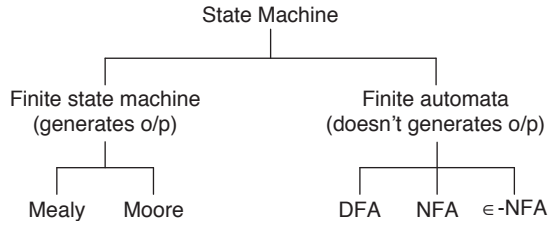


Types of FSM:

1. Melay machine.
2. Moore machine

Finite Automata (FA):

- FA is a state machine that comprehensively captures all possible states and transitions that a machine can take while responding to a stream (sequence) of input symbols.
- FA is recognizer of 'regular languages'.



Types of FA

1. Deterministic Finite Automata (DFA):

- DFA machine can exists in only one state at any given time.
- DFA is defined by 5-tuple: $\{Q, \Sigma, q_0, F, \delta\}$, where

$Q \rightarrow$ Finite number of states (elements)
$\Sigma \rightarrow$ Finite set of symbols (alphabets)
$q_o \rightarrow$ Start/Initial state
$F \rightarrow$ Set of final states.
$\delta \rightarrow$ Transition function, which is a mapping between

$$\delta: Q \times \Sigma \rightarrow Q.$$

How to use DFA:

Input: A word w in Σ^*
Question: Is w acceptable by DFA?

Steps:

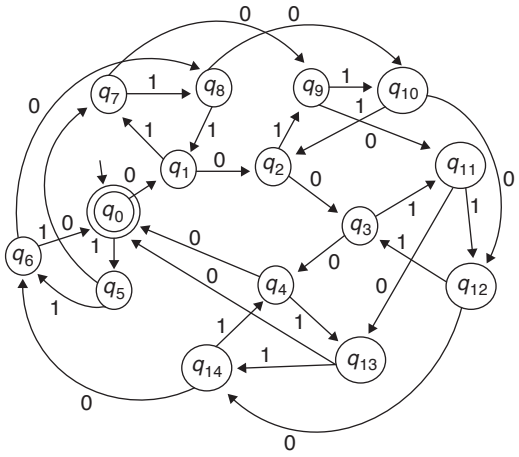
- Start at 'initial state', q_o.
- For every input symbol in sequence w, do.
- Compute the next state from current state, given the current input symbol in w and transition function.
- If after all symbols in 'w' are consumed, the current state is one of the final states (f) then accept 'w';
- Otherwise, reject w.

Transition diagram: State machines are represented by directed graphs called transition (state) diagrams.

- The vertices denoted by single circle represent the state and arcs labeled with input symbol correspond to transition.
- The final states are represented with double circles.

Transition Table: Transition function can be represented by tables.

Example 6: The following finite state machine accepts all those binary strings in which the numbers of 0's and 1's are respectively.



(A) Divisible by 3 and 2
(B) Odd and even
(C) Divisible by 5 and 3
(D) Divisible by 2 and 3

Solution: (C)
Number of 0's is divisible by 5.
Number of 1's is divisible by 3.

Table *Transition Table*

Current State	0	1
$\rightarrow (q_0)$	q_1	q_5
q_1	q_2	q_7
q_2	q_3	q_9
q_3	q_4	q_{11}
q_4	q_0	q_{13}
q_5	q_7	q_6
q_6	q_8	q_0
q_7	q_9	q_8
q_8	q_{10}	q_1

q_9	q_{11}	q_{10}
q_{10}	q_{12}	q_2
q_{11}	q_{13}	q_{12}
q_{12}	q_{14}	q_3
q_{13}	q_0	q_{14}
q_{14}	q_6	q_4

Note: Minimum number of states for k-divisibility is k-states.
In above example, $q_0 - q_{14} \rightarrow 15$ – states.
$\therefore \quad 5 \times 3 = 15$
The given binary strings have number of 0's divisible by 5 and number of 1's divisible by 3.

2. Non-deterministic finite Automata (NFA):

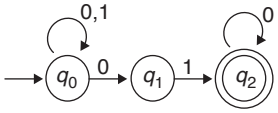
- The machine can exist in multiple states at the same time.
- Each transition function maps to a set of states.
- NFA is defined by 5-tuple: $\{Q, \Sigma, q_0, F, \delta\}$, where

$Q \rightarrow$ Finite number of states (elements)
$\Sigma \rightarrow$ Finite set of symbols. (Alphabets)
$q_o \rightarrow$ Start/Initial state
$F \rightarrow$ Set of final states.
$\delta \rightarrow$ Transition function which is a mapping between
$\delta = Q \times \Sigma \rightarrow 2^Q$

How to use NFA:
Input: a word w in Σ^*
Question: Is w accepted by NFA?
Steps:

- Start at 'start state' q_0.
- For every input symbol in the sequence, *w* does.
- Determine all possible next states from current state, given the current input symbol in *w* and transition function.
- If after all symbols in *w* are consumed, at least one of the current states is a final state then accept *w*.
- Otherwise, reject *w*.

Example 7: What is the language, *L* generated by the below NFA, given strings defined over alphabet, $\Sigma = \{0, 1\}$.



(A) Strings that end with '0'
(B) Strings that start with '0' and end with '0'
(C) Strings that contain '01' as substring
(D) Strings that contain '01' as substring and end with '0'

Solution: (D)

State	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	q_2
q_2	$\{q_2\}$	$\varnothing$

String: 0100100

$$q_0 \xrightarrow{0} \{q_0, q_1\}$$
$$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} \{q_0, q_1\}$$
$$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} \{q_0, q_1\}$$
$$\xrightarrow{q_0 1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} \{q_0, q_1\} \text{ (Non-deterministic)}$$
$$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} \{q_0, q_1\}$$
$$q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_2 \xrightarrow{0} q_2$$

Table 2 *Difference between NFA and DFA*

DFA	NFA
1. All transitions are deterministic i.e., each transition leads to exactly one state.	1. Transitions could be non-deterministic i.e., a transition could lead to a subset of states.
2. For each state, the transition on all possible symbols should be defined.	2. For each state, not all symbols necessarily have to be defined.
3. Accepts input if last state is in 'F'.	3. Accepts input if one of last states is in 'F'.
4. Practical implementation is feasible.	4. Practical implementation has to be deterministic (so needs conversion to DFA).

Relation between DFA and NFA

- A language '*L*' is accepted by a DFA if and only if it is accepted by a NFA.
- Every DFA is special case of a NFA.

Example 8: Let N_f and D_f denote the classes of languages accepted by non-deterministic finite automata and deterministic finite automata respectively. Which one of following is true?

(A) $D_f \subset N_f$ (B) $D_f \supset N_f$
(C) $D_f = N_f$ (D) $D_f \in N_f$

Solution: (C)
According to 'subset construction', every language accepted by NFA is also accepted by some DFA.
$\therefore \quad D_f = N_f$

NFA WITH ∈-MOVES

- ∈-transitions in finite automata allows a state to jump to another state without consuming any input symbol.

Conversion and Equivalence:

∈-NFA → NFA → DFA

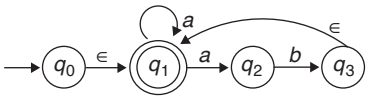
NFA without ∈-moves:

- Two FA, N_ϵ and *N* are said to be equivalent, if $L(N_\epsilon) = L(N)$ i.e., any language described by some N_ϵ, there is an *N* that accepts the same language.
- For $N_\epsilon = (Q, Z, \delta, q_0, F)$ and $N = (Q, \Sigma', \delta', q_0, F')$, Find
- $\delta'(q, a) = \epsilon\text{-closure}\,(\delta(\epsilon\text{-closure}(q), a))$

- $F' = \{F \cup \{q_0\}\}$, if $\in$-closure (q_0) contains a member of $F = F$, otherwise.

Note: When transforming $N_\in$ to N, only transitions are required to be changed and states remains same.

Example 9: Consider following NFA with $\in$-moves.



If given NFA is converted to NFA without $\in$-moves, which of following denotes set of final states?

(A) $\{q_0, q_1\}$ (B) $\{q_1, q_2\}$
(C) $\{q_1, q_2, q_3\}$ (D) $\{q_1\}$

Solution: Let $N = (Q, \Sigma^1, \delta^1, q_0, F^1)$
$F^1 = F \cup \{q_0\}$
ε-closure $(q_0) = \{q_0, q_1\}$
$\therefore F^1 = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$
Conversion $N_\in \rightarrow N$:
To compute, δ^1
$\in$-closure $(q_0) = \{q_0, q_1\}$, $\in$-closure $(q_3) = \{q_3, q_1\}$
$\delta^1(q_0, a) = \{q_1, q_2\}$, $\delta^1(q_0, b) = \varnothing$, $\delta^1(q_2, a) = \varnothing$.
$\delta^1(q_1, a) = \{q_1, q_2\}$, $\delta^1(q_1, b) = \varnothing$, $\delta^1(q_2, b) = \{q_1, q_3\}$
$\delta^1(q_3, a) = \{q_1, q_2\}$, $\delta^1(q_3, b) = \varnothing$

Table 3 *Transition Table*

State \ Input	a	b
$\rightarrow (q_0)$	$\{q_1, q_2\}$	$\varnothing$
(q_1)	$\{q_1, q_2\}$	$\varnothing$
q_2	$\varnothing$	$\{q_1, q_3\}$
q_3	$\{q_1, q_2\}$	$\varnothing$

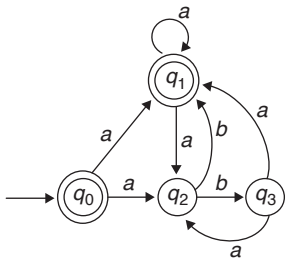


Figure 1 Transition diagram

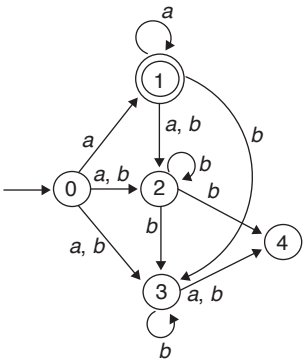
Conversion of NFA to DFA

Let a NFA be defined as, $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$
The equivalent DFA, $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ where:
Step I: $Q_D = 2^{Q_N}$; i.e., Q_D is set of all subsets of Q_N i.e., it is power set of Q_N.
Step II: F_D is set of subsets S of Q_N such that $S \cap F_N \neq \varnothing$. i.e., F_D is all sets of N's states that include atleast one accepting state of N.
Step III: For each set, $S \leq Q_N$ and for each input symbol a in Σ: $\delta_D(S, a) = \cup_{P \in S} \delta_N(P, a)$

That is, to compute $\delta_D(S, a)$, look at all states P in S, see what states N goes to starting from P on input a, and take the union of all those states.

Note: For any NFA, N with 'n' states, the corresponding DFA, D can have 2^n states.

Example 10: What is the number of final states in DFA constructed from the given NFA?



(A) 1 (B) 2
(C) 3 (D) 4

Solution:

Table 4 *Transition Table of NFA*

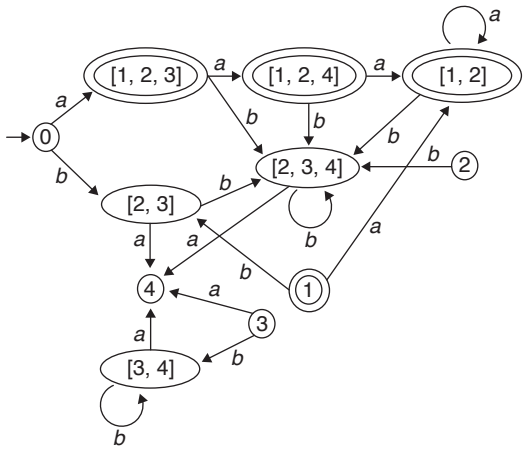
State \ Input	a	b
$\rightarrow 0$	{1, 2, 3}	{2, 3}
(1)	{1, 2}	{2, 3}
2	$\varnothing$	{2, 3, 4}
3	{4}	{3, 4}
4	$\varnothing$	$\varnothing$

Table 5 *Transition Table of DFA*

State \ Input	a	b
$\rightarrow 0$	[1, 2, 3]	[2, 3]
(1)	[1, 2]	[2, 3]
2	$\varnothing$	[2, 3, 4]
3	4	[3, 4]
4	$\varnothing$	$\varnothing$
([1, 2])	[1, 2]	[2, 3, 4]
[2, 3]	[4]	[2, 3, 4]
[3, 4]	[4]	[3, 4]
([1, 2, 3])	[1, 2, 4]	[2, 3, 4]
([1, 2, 4])	[1, 2]	[2, 3, 4]
[2, 3, 4]	[4]	[2, 3, 4]

Hence final states in obtained DFA is '4'.

DFA is: Choice (D)

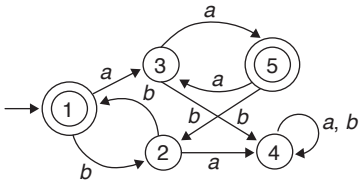


Minimization of DFA

Given a DFA, $M = (Q, \Sigma, \delta, q_0, F)$, we construct a reduced DFA, $M' = (Q', \Sigma', \delta', q_0', F')$ as follows

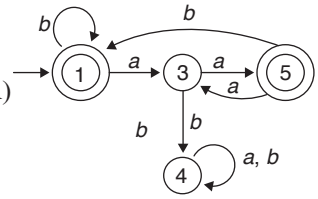
1. Remove all inaccessible states. All states that are unreachable from the initial state are removed.
2. Consider all pairs of states (p, q), If $p \in F$ and $q \in F$ or vice versa mark the pair (p, q) as distinguishable.
3. Repeat until no previously unmarked pairs are marked. For all pairs (p, q) and all $a \in \Sigma$, compute $\delta(p, a) = p_a$ and $\delta(p, q) = q_a$. If the pair (p_a, q_a) is marked as distinguishable mark (p, q) as distinguishable.
4. Find the sets of all indistinguishable states, say $\{q_i, q_j, \cdots q_k\}$, $\{q_\ell, q_m, \cdots q_n\}$, etc. For each set $\{q_i, q_j, \ldots q_k\}$ of such indistinguishable states, create a state labelled $ij \ldots k$ for M.
5. For each transition rule of M of the from $\delta(q_r, q) = q_p$, find the sets to which q_r and q_p belong. If $q_r \in \{q_i, q_j, \ldots q_k\}$ and $q_p \varepsilon \{q_\ell, q_m, \cdots q_n\}$, add a rule to δ: $\delta'(ij \ldots k, a) = \ell m \ldots n$.

Example 11: A DFA with alphabet $\Sigma = \{a, b\}$ is given below:

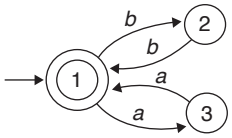


Which of the following is valid minimal DFA which accepts same language as given DFA?

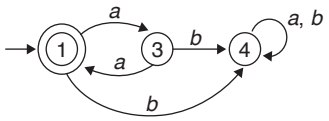
(A)



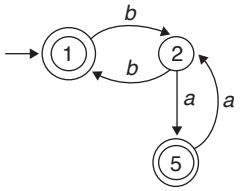
(B)



(C)



(D)



Solution: (B)

Initially, $\{1, 5\}, \{2, 3, 4\}$

Depending on next states and inputs, the partitions of states can be as: $\{\{1, 5\}, \{2\}, \{3\}$, and $\{4\}\}$

Since, 1 to 5 have same transition, unite $\{1, 5\}$

State 4 is dead state $\rightarrow$ It has transition only to itself.

Since, $\{2\}, \{3\}$ are singletons, they exist.

$\therefore$ States in minimized DFA are $\{1, 2,$ and $3\}$

$\{1\} \rightarrow \{1, 5\}$

For transitions, since $1 \xrightarrow{a} 3$, $1 \xrightarrow{b} 2$ in given DFA, in minimized DFA, transitions are added from $1 \xrightarrow{a} 3$, $1 \xrightarrow{b} 2$. Also, since $2 \xrightarrow{b} 1$, $3 \xrightarrow{a} 1$ in given DFA, the minimized DFA, transitions are added from $2 \xrightarrow{b} 1$, $3 \xrightarrow{a} 1$.

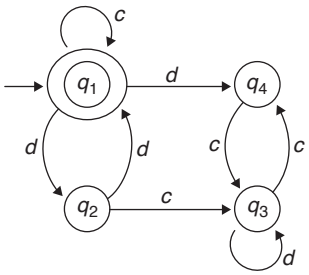
Equivalence Between NFA and DFA

There is a DFA_D for any NFA_N i.e., $L(D) = L(N)$.

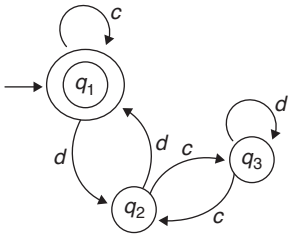
Construction:

- In DFA or NFA, whenever an arrow is followed, there is a set of possible states. This set of states is a subset of Q.
- Track the information about subsets of states that can be reached from initial state after following arrows.
- Consider each subset of states of NFA as a state of DFA and every subset of states containing a final state as a final state of DFA.

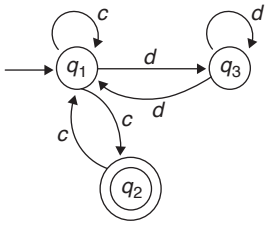
Example 12: Which of following is equivalent DFA for the NFA given below:



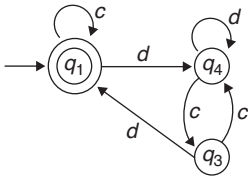
(A)



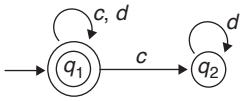
(B)



(C)



(D)



Solution: (A)

Table 6 *Transition Table of NFA*

δ	**c**	**d**
$\rightarrow(q_1)$	q_1	$\{q_2, q_4\}$
q_2	q_3	q_1
q_3	q_4	q_3
q_4	q_3	$\varnothing$

Table 7 *Transition Table of DFA*

δ	**c**	**d**
$\rightarrow(q_1)$	q_1	q_2
q_2	q_3	q_1
q_3	q_2	q_1

Table 8 *Common Table*

δ	**c**	**d**
(q_1, q_1)	(q_1, q_1)	(q_2, q_4, q_2)
(q_2, q_2)	(q_3, q_3)	(q_1, q_1)
(q_3, q_3)	(q_4, q_2)	(q_3, q_3)
(q_4)	q_3	$\varnothing$

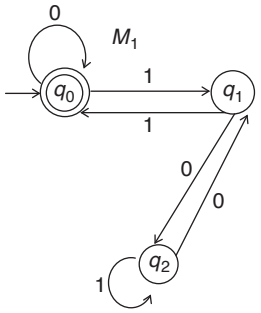
Equivalence of Finite Automatas:

- Two automatas A and B are said to be equivalent if both accept exactly the same set of input strings.
- If two automatas M_1 and M_2 are equivalent then

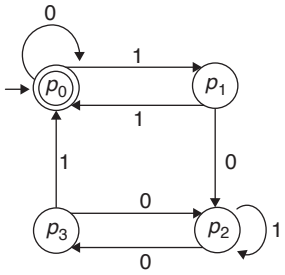
(i) If there is a path from the start state of M_1 to a final state of M_1 labeled $a_1a_2 \ldots a_k$ then there is a path from the start state of M_2 to the final state of M_2 labeled $a_1a_2 \ldots a_k$.

(ii) If there is a path from the start state of M_2 to a final state M_2 labeled $b_1b_2 \ldots b_i$ then there is a path from the start state of M_1 to the final state of M_1 labeled $b_1b_2 \ldots b_i$.

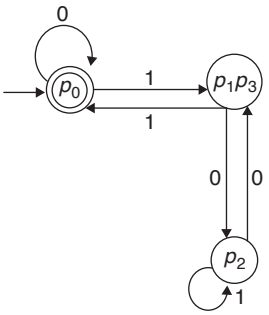
Example:



M_2:



In M_2, states p_1 and p_3 are equivalent (as both are reaching either final or non-final states with same input). After minimizing M_2, we will get



$\therefore$ M_1 and M_2 are equivalent.

Union: The union of two languages L and M is the set of strings that are in both L and M.
Ex: $L = \{0, 1\}$, $M = \{111\}$
$L \cup M = \{0, 1, 111\}$.

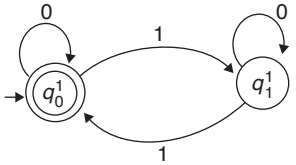
Concatenation: The concatenation of Languages L and M is the set of strings that can be formed by taking any string in L and concatenating it with any string in M.

Example: $L = \{0, 1\}$, $M = \{\varepsilon, 010\}$
$LM = \{0, 1, 0010, 1010\}$.

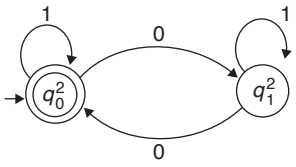
Closure, Star or Kleen star of a language L:
Kleen star is denoted as L^*. It represents the set of strings that can be formed by taking any number of strings from L with repetition and concatenating them. It is a Unary operator.
L^0 is the set; we can make selecting zero strings from L.
$L^0 = \{\varepsilon\}$
L^1 is the language consisting of selecting one string from L.
L^2 is the language consisting of concatenations selecting two strings from L.
…
L^* is the union of $L^0, L^1, \ldots L^\infty$.
Ex: $L = \{0,10\}$
$L^* = \{0,00,000,10,010, \ldots\}$

Intersection:
Let two DFAs M_1 and M_2 accept the languages L_1 and L_2.
$M_1 = (Q_1, \Sigma, \delta_1, q_0^1, .F_1)$
$M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$
The intersection of M_1 and M_2 can be given as
$M = (Q_1, \Sigma, \delta, q_0 F)$
Q = Pairs of states, one from M_1 and one from M_2 i.e.,
$Q = \{(q_1, q_2) \mid q_1 \in Q_1$ and $q_2 \in Q_2\}$
$Q = Q_1 \times Q_2$.
$q_0 = (q_0^1, q_0^2)$
$\delta\,(q_i^1\, q_j^2), x) = (\delta_1(q_1^1, x), \delta_2(q_j^2, x))$
$F = \{(q_1, q_2) \mid q_1, \in F_1$ and $q_2 \in F_2\}$

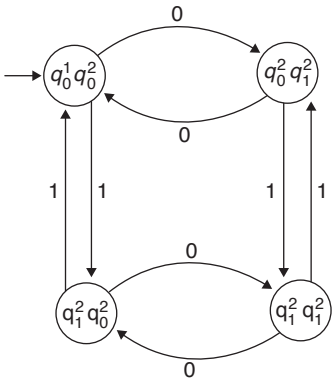
Example:
M_1: Strings with even number of 1's.



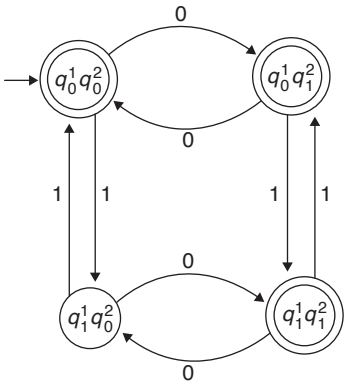
M_2: Strings with odd number of 0's.



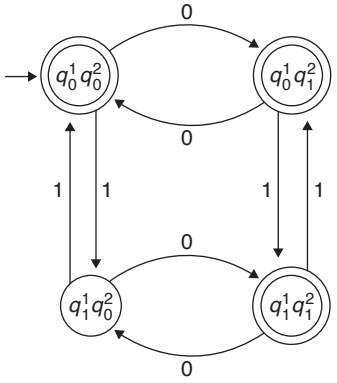
$M_1 \cap M_2$: Strings with even number of 1's and odd number of 0's.



Union of M_1 and M_2:



Difference: The difference of L_1 and L_2 can be given as $L_1 - L_2$ with $M = (Q, \Sigma, \delta, q_0, F)$.
$Q = Q_1 \times Q_2$
$q_0 = (q_0^1, q_0^2)$
$\delta((q_i^1, q_j^2), x) = (\delta_1\,(q_i^1, x), \delta_2(q_j^2, x))$
$F = \{(q_1, q_2) \mid q_1, \in F_1$ and $q_2 \notin F_2\}$



Reversing a DFA:

- M is a DFA which recognizes the language L.
- M^R will accept the language L^R.

To construct M^R:

- Reverse all transitions
- Turn the start state to final state
- Turn the final states to start state.
- Merge states and modify the FA,

such that the resultant contain a single start state.

Mealy and Moore Machines

Moore Machine

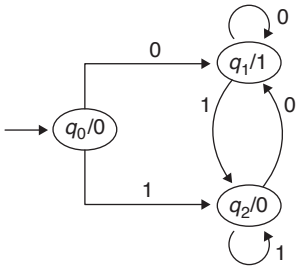
A moore machine is a finite state machine, where outputs are determined by current state alone.

A Moore machine associates an output symbol with each state and each time a state is entered, an output is obtained simultaneously. So, first output always occurs as soon as machine starts.

Moore machine is defined by 6-tuples:
$(Q, \Sigma, \delta, q_0, \Delta, \lambda)$, where
$Q \rightarrow$ Finite set of states
$\Sigma \rightarrow$ Finite set of input symbols
$\Delta \rightarrow$ It is an output alphabet
$\delta \rightarrow$ Transition function, $Q \times \Sigma \rightarrow Q$ (state function)
$\lambda \rightarrow$ Output function, $Q \rightarrow \Delta$ (machine function)
$q_0 \rightarrow$ Initial state of machine

Note: The output symbol at a given time depends only on present state of moore machine.

Example 13: The language generated by the following moore machine is:



(A) 2's complement of binary number.
(B) 1's complement of binary number.
(C) Has a substring 101.
(D) Has a substring 110.

Solution: (B)
Binary number: 1011
1's complement: 0100

$$q_0 \xrightarrow{1/0} q_2, \xrightarrow{0/1} q_1 \xrightarrow{1/0} q_2 \xrightarrow{1/0} q_2$$

$1 \rightarrow 0, 0 \rightarrow 1, 1 \rightarrow 0, 1 \rightarrow 0$

Mealy Machine

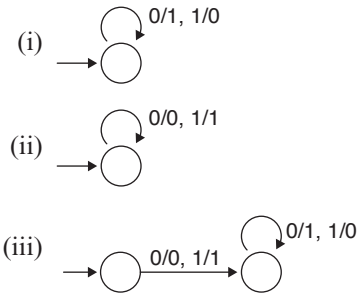
- A mealy machine is a FSM, where outputs are determined by current state and input.
- It associates an output symbol with each transition and the output depends on current input.
- Mealy machine is defined on 6-tuples: $(Q, \Sigma, \delta, q_0, \Delta, \lambda)$, where

Q – Finite set of states.
Σ – Finite set of input symbols.
δ – $(Q \times \Sigma \rightarrow Q)$ is transition function.
$q_0 \rightarrow q_0 \in Q$ is initial state.
$\Delta \rightarrow$ Finite set of output symbols.
$\lambda \rightarrow$ Output function, $\lambda(Q \rightarrow \Delta)$

Note: In Moore machine, for input string of length n, the output sequence consists of $(n + 1)$ symbols.

In Mealy machine, for input string of length n, the output sequence also consists of 'n' symbols.

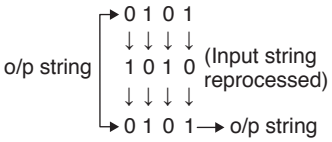
Example 14: Let $(Me)^2$ mean that given a Mealy machine, an input string is processed and then output string is immediately fed into the machine (as input) and reprocessed. Only this second resultant output is considered as the final output of $(Me)^2$. If final output string is same as original input string then $(Me)^2$ has an identity property. Consider following machines.



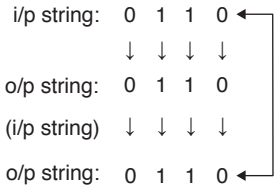
Which of above machines have identity property?
(A) (i) only
(B) (i) and (ii) but not (iii)
(C) (i) and (iii) but not (ii)
(D) All have identity property

Solution: (D)

(i) Consider i/p string



(ii)



(iii)

Equivalence of Moore and Mealy machine

(a) Mealy machine equivalent to Moore machine:
If $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ is a Moore machine, then there is a Mealy machine M_2 equivalent to M_1.

Proof: Let $M_2 = (Q, \Sigma, \Delta, \delta, \lambda^1, q_0)$ and define $\lambda^1(q, a)$ to be $\lambda(\delta(q, a))$ for all states q and input symbol 'a'.

Then M_1 and M_2 enter the same sequence of states on the same input, and with each transition M_2 emits the o/p that M_1 associates with the state entered.
Let us consider Mealy Machine

Present State	Next State			
	Input State	$a = 0$ Output	Input State	$a = 1$ Output
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

To convert the Mealy machine to Moore machine,

- We look into the next state column for any state, say q_i and determine the number of different outputs associated with q_i in next column.
- Split q_i into several different states, the number of such states being equal to the number of different outputs associated with q_i.

Present State	Next State			
	Input State	$a = 0$ Output	Input State	$a = 1$ Output
$\rightarrow q_1$	q_3	0	q_{20}	0
q_{20}	q_1	1	q_{40}	0
q_{21}	q_1	1	q_{40}	0
q_3	q_{21}	1	q_1	1
q_{40}	q_{41}	1	q_3	0
q_{41}	q_{41}	1	q_3	0

- The pair of states and outputs in the next state column can be rearranged as:

Present state	Next State		output
	$a = 0$	$a = 1$	
$\rightarrow q_1$	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Moore machine equivalent to Mealy machine

Let $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ be a Mealy machine. Then there is a machine M_2 equivalent to M_1

Proof: Let $M_2 = (QX\Delta, \Sigma, \Delta, \delta^1, \lambda^1, [q_0, b_0])$, where b_0 is an arbitrary selected member of Δ.

That is, the states of M_2 are pairs $[q, b]$ consisting of a state of M_1 and output symbol, Define $\delta^1([q, b], a) = [\delta(q, a), \lambda, (q, a)]$ and $\lambda^1([q, b]) = b$.

The second component of a state $[q, b]$ of M_2 is the output made by M_1 on some transition into state q.

Only the first components of M_2's states determine the moves made by M_2.

Every induction on 'n' shows that if M_1 enters states $q_0, q_1 \ldots q_n$ on inputs $a_1, a_2 \ldots a_n$ and emits output $b_1, b_2, \ldots b_n$ then M_2 enters states $[q_0, b_0], [q_1, b_1] \ldots [q_n, b_n]$ and emits outputs $b_0, b_1 \ldots b_n$.

Let us consider the Moore machine

Present State	Next State		Output
	$a = 0$	$a = 1$	
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

- To convert Moore into Mealy machine, we must follow the reverse procedure of converting Mealy machine into Moore machine.
- For every input symbol we form, the pair consisting of the next state and the corresponding output and reconstruct the table for Mealy machine.
- For example, the state q_3 and q_1 in the next state column should be associated with outputs 0 and 1, respectively.

The Transition table for Mealy machine is:

Present state	Next State			
	$a = 0$ state	output	$a = 1$ state	output
$\rightarrow q_0$	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

Regular Languages

The set of regular languages over an alphabet Σ is defined recursively as below. Any language belonging to this set is a regular language over Σ.

Definition of set of regular languages

- Basis clause: $\varnothing$, $\{\varepsilon\}$, $\{a\}$ for any symbol $a \in \Sigma$, are regular languages.
- Inductive clause: If L_r and L_s are regular languages, then $L_r \cup L_s$, $L_r \cdot L_s$, L_r^* are regular languages.
- External clause: Nothing is a regular language, unless it is obtained from above two clauses.

Regular language: Any language represented by regular expression(s) is called a regular language.

Ex: The regular expression a^* denotes a language which has $\{\varepsilon, a, aa, aaa, \ldots\}$

Regular expression

- Regular expressions are used to denote regular languages.
- The set of regular expressions over an alphabet Σ is defined recursively as below. Any element of that set is a regular expression.

- Basis clause: $\varnothing, \in$, a are regular expression corresponding to languages $\varnothing$, $\{\in\}$, $\{a\}$ respectively where a is an element of Σ.
- Inductive clause: If r and s are regular expression corresponding to languages L_r and L_s then $(r + s)$, (rs) and (r^*) are regular expressions corresponding to the languages $L_r \cup L_s$, $L_r \cdot L_s$ and L_r^* respectively.
- External clause: Nothing is a regular expression, unless it is obtained from above two clauses.

Closure property of regular expressions The iteration or closure of a regular expression R, written as R^* is also a regular expression.
Ex: $\Sigma = \{a\}$ then a^* denotes the closure of Σ.
$a^* = \{\varepsilon, a, aa, aaa, \ldots\}$

Conventions on regular expressions

1. The operation '*' has highest precedence over concatenation, which has precedence over union (+).
   i.e., $RE\ (a + (b(c^*))) = a + bc^*$
2. The concatenation of K r's, where r is a regular expression is written as r^k. The language corresponding to r^k is L_r^k. Where L_r is language corresponding to regular expression r i.e., $rr = r^2$
3. r^+ is a regular expression to represent L_r^+

Note: A regular expression is not unique for a language i.e., regular language corresponds to more than one regular expression.

Example 15: Give regular expression for set of strings which either have 'a' followed by some b's or all b's also containing 'ε'.
(A) $b^* + ab^*$ (B) $a^* + ba^*$
(C) $(\varepsilon) + (\varepsilon + a)\, b^+$ (D) $b^* + ab^* + \varepsilon$

Solution: (C)
The regular expression is, $r = ab^+ + b^+ + \varepsilon = b^+(a + \varepsilon) + \varepsilon$.

Identity rules for regular expressions:

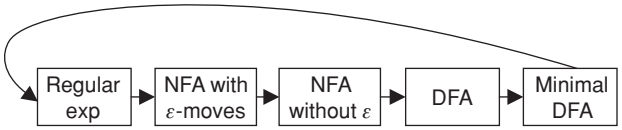
1. $\varnothing + R = R$
2. $\varnothing \cdot R = R\varnothing = \varnothing$
3. $\varepsilon R = R\varepsilon = R$
4. $\varnothing^* = \varepsilon$ and $\varepsilon^* = \varepsilon$
5. $R + R = R$
6. $RR^* = R^*R = R^+$
7. $\varepsilon + RR^* = R^*$ and $\varepsilon + R^*R = R^*$
8. $(R^*)^* = R^*$
9. $R^*R^* = R^*$
10. $\varepsilon + R^* = R^*$
11. $(R + \varepsilon)^* = R^*$
12. $R^*(\varepsilon + R)^* = (\varepsilon + R)^* R^* = R^*$
13. $R^*R + R = R^*R$
14. $(P + Q)R = PQ + QR$ and $R(P + Q) = RP + RQ$
15. $(P + Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$
16. $(PQ)^*P = P(QP)^*$
17. R is given as, $R = Q + RP$ has unique solution, $R = QP^*$. This is Arden's theorem.
18. $(P + Q)^* = (P^* + Q) = (P + Q^*)$

Example 16: If r_1 and r_2 are regular expressions denoting languages L_1 and L_2 respectively then which of following is false?
(A) $(r_1)|\,(r_2)$ is regular expression denoting $L_1 \cup L_2$.
(B) $(r_1)\,(r_2)$ is regular expression denoting $L_1 \cdot L_2$.
(C) $\varnothing$ is not a regular expression.
(D) $\{r_1\}^*$ is regular expression denoting L_1^*.

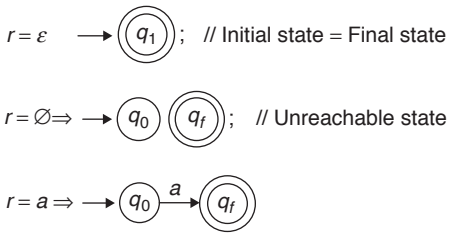
Solution: (C)

Constructing FA for Given RE

- Relationship between FA and RE.



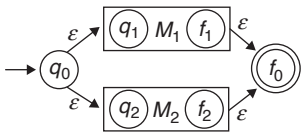
Identities:
Basis:



Induction:

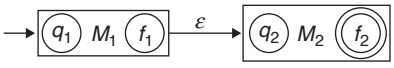
- **Union:** $L(r) = L(r_1) + L(r_2)$ i.e., $L(M) = L(M_1) \cup L(M_2)$

Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$, $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, \{f_2\})$ with $L(M_1) = L(r_1)$ and $L(M_2) = L(r_2)$, then $M = (Q_1 \cup Q_2 \cup \{q_0, f_0\}, \Sigma_1 \cup \Sigma_2, \delta, q_0, \{f_0\})$



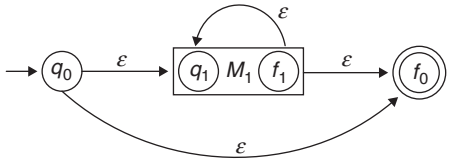
- **Concatenation:**

$L(r) = L(r_1) \cdot L(r_2)$ i.e., $L(M) = L(M_1) \cdot L(M_2)$

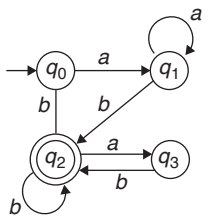


- **Closure:**

$L(r) = L(r)^*$ i.e., $L(M) = L(M_1)^*$
Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$ then $L(M) = (Q_1 \cup \{q_0, f_0\}, \Sigma_1, \delta, q_0, \{f_0\})$



Example 17: The regular expression generated by the given FA.



(A) $(a + ba^*)\, b^*$ (B) $(aa^*b + bb^*)\, b^*$
(C) $(b + ab^*)\, a^*$ (D) $(ab + ba)^*$

Solution: (B)
q_2 is final state which is obtained with input symbol only 'b'. So, (C) or (D) is not true.

In (A) $\rightarrow ba^*$ is not defined in given *FA*. Instead bb^* is defined.

Pumping Lemma for Regular Sets

Theorem Let 'L' be an arbitrary regular language. Then there exists a positive integer, P with following property:

Given an arbitrary member, w of L having length at least P (i.e., $|w| \geq P$), w can be divided into 3-parts, $w = xyz$ ∃

- $|y| \geq 1$ (the middle part is non-empty)
- $|xy| \leq P$ (the first two parts have length atmost P)
- For each, $i \geq 0$, $xy^iz \in L$ (removing or repeating the middle part produces member of L)

Proof Let L be an arbitrary regular language. Then there is a *FA*, say M that decides L.

Let P be the number of states of M.

Let w be an arbitrary member of L, having length 'n' with $n \geq P$.

Let $q_0, q_1, \ldots q_n$ be states that M on input w. That is, for each i, after reading the first i symbols of w, M is at q_i.

q_0 is initial state of M. Also, since $w \in L$, q_n is a final state of M.

Let $x = w_1 \ldots w_c$, $y = w_{c+4} \ldots w_d$, $z = w_{d+1} \ldots w_n$. Then:

- $|y| \geq 1$
- $|xy| \leq P$
- M transitions from q_0 to q_c on x.
- M transitions from q_c to q_c on y.
- M transitions from q_c to q_n on z.

Thus, for every $i \geq 0$, M transitions from q_0 to q_n on xy^iz and so, xy^iz is a member of L.

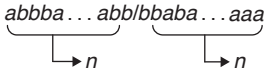
Note:

- Pumping lemma is used to verify that given language is not regular.
- Pumping lemma follows pigeon hole principle.

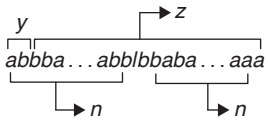
Example 18: The language, L is defined as:
$L = \{w_1w_2 : w_1, w_2 \in \{a, b\}^*, |w_1| = |w_2|\}$. Is the language regular?

(A) Regular
(B) Not regular
(C) Cannot be determined
(D) None of these

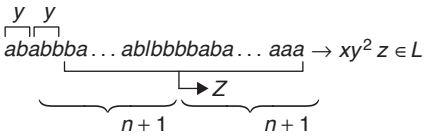
Solution: (A)
Fix pumping length, $K = 2$
For every proper strings in L, $(2n \geq 2)$



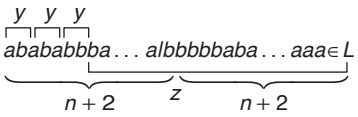
- Split in x, y, z with desired properties.



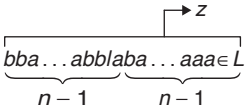
- Let $x = \varepsilon$, y = first two symbols, z = rest.



- xy^3z:



- $xy^oz \rightarrow$



∴ For every $i \geq 0$, $xy^iz \in L$. Hence given language is regular.

Closure Properties of Regular Sets

1. **Union:** If L and M are regular languages, LUM is regular language closed under union.
2. **Concatenation and Kleen closure:** If L and M are regular languages, L.M is regular language and L^* is also regular.
3. **Intersection:** $L \cap M$ is regular, if L and M are regular languages.
4. **Difference:** $L - M$ contains strings in 'L' but not M, where L and M are regular languages.
5. **Complementation:** The complement of language L is Σ^*–L.

Note: Since Σ^* is surely regular, the complement of a regular language is always regular. Where Σ^* is a universal language.

6. **Homomorphism:** If L is a regular language, h is homomorphism on its alphabet then $h(L) = \{h(w) \mid w$ is in $L\}$ is also a regular language.

Regular grammar

- **Grammar:** Generative description of a language.
- **Automaton:** Analytical description.
- A grammar is a 4-tuple, G = (V, Σ, R, S) where V: alphabet (variable) (non-terminals)

$\Sigma \subseteq V$ is set of terminal symbols.
$R \subseteq (V^+ \times V^*)$ is a finite set of production rules.
$S \in V - \Sigma$ is start symbol.

Notation

- Elements of $V - \Sigma$: $A, B, \ldots$
- Elements of Σ: $a, b \ldots$
- Rules $(\alpha, \beta) \in R$: $\alpha \to \beta$ or $\alpha \xrightarrow[G]{} \beta$
- Start symbol is written as S.
- Empty word: ε

Example 19: The regular expression that describe the language generated by grammar, $G = (\{S, A, B\}, \{a, b\}, S, \{S \to Aab, A \to Aab|B, B \to a\}$

(A) $(ab)^* a$ (B) $aab(ab)^*$
(C) $ab^* aa$ (D) $(a + ba)^*$

Solution: (B)
$S \to Aab \to Aab\ ab \to A\ ab\ abab \to Bababab$
$\to aababab \to aab(ab)^*$

Union of two Regular languages:
If L_1 and L_2 are two languages then
$L_1 \cup L_2 = \{w/w \in L_1$ or $w \in L_2\}$
The union of two regular languages is also a regular language.
Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, f_1)$
$M_2 = (Q_2, \Sigma, \delta_2, q_2, f_2)$
$M = M_1 U M_2$ can be given as
$M = (Q, \Sigma, \delta, q_0, f)$.
Where $Q = \{(r_1, r_2) \mid r_1 \in Q_1$ and $r_2 \in Q_2\}$
i.e., Q is the Cartesian product of sets Q_1 and Q_2.
Σ is the alphabet, is the same in M_1 and M_2.
$\Sigma = \Sigma_1 \cup \Sigma_2$.
δ is the transition function given as:
$\delta(r_1, r_2), a = (\delta_1(r_1, a)\ \delta_2(r_2, a))$.
q_0 is the pair (q_1, q_2).
F is the set of pairs in which either member is an accept state of M_1 or M_2.
$F = \{(r_1, r_2) \mid r_1 \in F_1$ or $r_2 \in F_2\}$

Types of Grammars

- Type 0: Unrestricted, recursively enumerable languages.
- Type 1: Context-sensitive grammar.
- Type 2: Context free grammar.
- Type 3: Regular grammar.

Type 0: Recursively enumerable grammar: (Turing Machine) (TM):

Every production rule is of form: $\alpha \to \beta$, where α and β are in $(V \cup T)^*$, i.e., there can be any strings of terminals and non-terminals (no-restriction).

Type 1: Context-sensitive Grammar: (Linear bounded automaton) (LBA):

Every production rule is of form, $\alpha \to \beta$ are in $(V \cup T)^*$ and $\alpha \neq \varepsilon$ and $|\beta| \geq |\alpha|$ i.e., any strings of terminals and non-terminals and length of string that can appear on RHS of production must be greater than or equal to length of string that can appear on LHS of production.

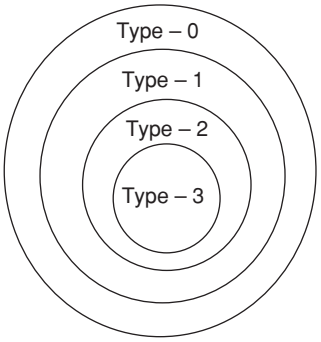
Type 2: Context-free grammar: (Push down automaton) (PDA):

Every production rule is of form, $A \to \alpha$ where α is in $(V \cup T)^*$ i.e., LHS of rule is single non-terminal and RHS can be any string of terminals and non-terminals.

Type 3: Regular grammar: (Finite automaton) (*FA*):

Every production is of form, $A \to aB$ or $A \to a$ where A and $B \in V$ and $a \in T$. That is, LHS of rule is non-terminal and RHS can be terminal (or) terminal followed by non-terminal.

Relationship between types of grammar:



- Regular sets are properly contained in CFL (Context Free Languages).
- The CFL's not containing empty string ε, are properly contained in CSL. (Context sensitive language).
- The CSL's are properly contained in Recursively enumerable languages.
- RG ⊂ CFG ⊂ CSL ⊂ REG

Left-linear Grammar:
All productions have form: $A \to Bx$ or $A \to x$

Right-linear Grammar:
All productions have the form: $A \to xB$ or $A \to x$.

Note:
- The regular grammars characterize the regular sets i.e., a language is regular if and only if it has a left-linear grammar or if and only if it has a right-linear grammar.
- If L has a regular grammar, then L is a regular set.
- If L is a regular set, then L is generated by some left-linear grammar and by some right-linear grammar.

Arden's theorem: Let P and Q be two regular expressions over Σ. If P does not contain 'ε' then the following equation in R, namely $R = Q + RP$ has a unique solution given by $R = QP^*$.

Arden's Theorem to obtain regular expression from given transition diagram: The following steps are used to find the RE recognized by transition system.

The following assumptions are made regarding the transition system.

(i) The transition graph does not have ε-moves
(ii) It has only one initial state, q_o.
(iii) The states in the transition diagram are $q_o, q_1, q_2, \ldots q_n$.
(iv) Q_i, the regular expression represents the set of strings accepted by a system even though q_i is the final state.
(v) ${}^a ij$ denotes the regular expression representing the set of labels of edges from q_i to q_j. When there is no such edge ${}^a ij = \phi$.

We will get the following set of equations.

$$Q_1 = Q_1 \alpha_{11} + Q_2 \alpha_{12} + \ldots Q_n \alpha_{n1} + \varepsilon$$
$$Q_2 = Q_1 \alpha_{12} + Q_2 \alpha_{22} + \ldots Q_n \alpha_{n2}$$
$$\vdots$$
$$Q_n = Q_1 \alpha 1n + Q_2 \alpha_{2n} + \ldots Q_n \alpha_{nn}.$$

By Repeatedly applying substitutions and Arden's theorem, we can express Q_i in terms of α_{ij}'s.

For getting the set of strings recognized by the transition system, we have to take the union of all Q_i's corresponding to final states.

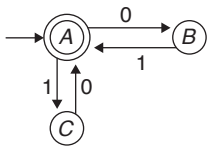
Construction of Regular Grammar from FA

Step I: Associate suitable variables like A, B, C … with states of automata.

Step II: Obtain the productions of the grammar as:
If $\delta(A, a) = B$ then add production $A \to aB$ to list of productions of grammar, if B is a final state, then add either $A \to a$ or $B \to \varepsilon$, to list of productions of grammar.

Step III: The variable associated with initial state of automata is start symbol of grammar.

Example 20: Regular grammar generating language accepted by below automata is



(A) $A \to 0B|1C|\varepsilon$
$B \to 1A$
$C \to 0A$

(B) $A \to 1B|0C|\varepsilon$
$B \to 1A$
$C \to 0A$

(C) $A \to B|C|\varepsilon$
$B \to 1$
$C \to 0$

(D) $A \to 0A|1B|\varepsilon$
$B \to 1C$
$C \to 0A$

Solution: (A)
$A \to 0B, A \to 1C, B \to 1A, C \to 0A$
$\therefore A$ is final state, $A \to \varepsilon$

$\therefore A \to 0B|1C|\varepsilon$
$B \to 1A$
$C \to 0A$

(or)

$A \to 0B|1C$
$B \to 1A|1$
$C \to 0A|0$

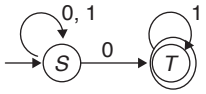
Construction of FA from given regular grammar

Given a regular grammar, G; a regular expression specifying $L(G)$ can be obtained directly as follows:

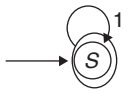
- Replace the '→' symbol in productions of grammar by '=' symbol, to get set of equations.
- Solve the set of equations obtained above to get the value of variable, S, where S is start symbol of grammar, result is regular expression specifying $L(G)$.

Example 21: The Regular grammar and *FA* for given regular expression $\phi^* 1^* U (0\phi)^*$ is ___

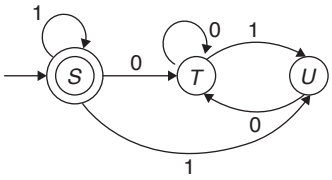
(A) $S \to 0S|1S|0$
$T \to 1T|\varepsilon$



(B) $S \to 1S|\varepsilon$

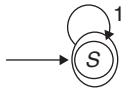


(C) $S \to 0T|1S|\varepsilon$
$T \to 0T|1U|\varepsilon$
$U \to 0T|1S$



(D) Cannot be determined

Solution: (B)
$\varnothing^* 1^* \cup (0\varnothing)^* = \varnothing^* \cdot 1^* \cup \varnothing^* = \varepsilon \cdot 1^* \cup \varepsilon = 1^*$.



EXERCISES

Practice Problems I

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. Find a regular expression for

$L = \{uvu: u, v \in \{a, b\}^*, |u| = 2\}$

(A) $(ab)^*a(ab)^*$

(B) $(aa)^*ab(aa)^*$

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

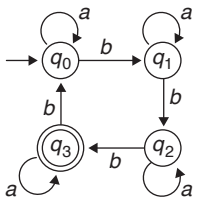
(D) $aa(a + b)^*aa + ab(a + b)^* ab + ba\ (a + b)^*\ ba + bb\ (a + b)^*bb$

2. Consider the regular expression, $R = 10 + (0 + 11)0^*\ 1$. The minimum number of states in any DFA accepting this regular expression is:

(A) 5 (B) 4

(C) 3 (D) 6

3. The following DFA accepts the set of all strings over $\{a, b\}$ that



(A) Contains number of b's divisible by 3.

(B) Contain number of a's and b's divisible by 3

(C) Contain number of b's congruent to 3 modulo 4.

(D) Contain any number of a's and b's

4. Consider the grammar, $S \rightarrow SS/a$. To get string of n terminals, the number of productions to be used is

(A) n^2 (B) n

(C) 2^{n+1} (D) $2n–1$

5. The language L is defined as, $L = \{a^i\ b^j\ c^{2j} |\ i \geq 0, j \geq 0\}$. Is this language L regular?

(A) Yes (B) No

(C) Cant be determined (D) None of these

6. The language, L is defined by set of strings over $\{a, b\}^*$ in which number of a's is a perfect cube. What is the nature of language, L?

(A) Regular (B) Non-regular

(C) Cant be determined (D) None of these

7. The language, L is defined over $\Sigma = \{0 – 7\}$. The string include 7, 16, 43, 61, 223, … The language generated is:

(A) Alternate odd and even numbers

(B) Octal representation of a number

(C) Divisible by 7.

(D) Octal representation of a number divisible by 7.

8. The language L, is defined as set of strings that start and end with equal number of a's and contain any number of b's. The grammar $L(G)$ for language L is defined with productions as:

(A) $S \rightarrow aBa$
$B \rightarrow \varepsilon | bB$

(B) $S \rightarrow aB$
$B \rightarrow a | bB$

(C) $S \rightarrow aT | bS$
$T \rightarrow aT | bT | a | b$

(D) $S \rightarrow B | aSa$
$B \rightarrow \varepsilon | bB$

9. If the regular set A is represented by $A = ((01)^*1^*)^*$. And the regular set B is represented by $B = (01 + 1)^*$, which of the following is true?

(A) $A \subset B$

(B) $B \subset A$

(C) $A = B$

(D) A and B are incomparable

10. The language, L that is generated over $\Sigma = \{0, 1\}$ for regular expression $L(r) = (0 + 10)^*\ 1\ (1 + 10)^*$

(A) Any string whose number of 1's length is greater than or equal to 3.

(B) Any string that has no substring 110.

(C) Any string that has no substring 00 after first 11.

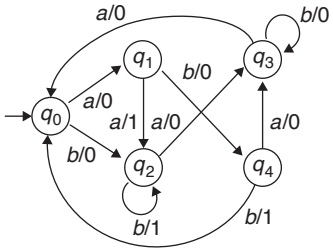
(D) Any string that has only one occurrence of substring 010.

11. The R.E $L(r) = (a^+b^*)\ U\ \varepsilon$. Is the grammar with productions generated over non-terminals $\{S, A\}$ ambiguous?

(A) Yes (B) No

(C) Can't be determined (D) None

12. The number of states in the obtained Moore machine while converting the given mealy to Moore are:



(A) 5 (B) 6

(C) 4 (D) 7

13. The language L is defined as $L = \{0^i 1^j / i \neq j\}$ over $\{0, 1, 2\}$, $A = \{0^i\ 1^j / i \geq 0, j \geq 0\}$ and $B = \{0^i\ 1^j / i = j\}$. For language, L to be non-regular. What should be relation between A, B, L?

(A) $B = (A \cup L)^c$ (B) $B = A \cup L$

(C) $B = A \cap \overline{L}$ (D) $B = A^c$

14. Which of following grammars are unambiguous?

(A) $S \rightarrow (S)\ S | [S]\ S | \varepsilon$ (B) $S \rightarrow S(S)S | \varepsilon$

(C) $S \rightarrow aS | Sa | a$ (D) $S \rightarrow a | Sa | bSS | Ssb | SbS$

15. What will be number of final states obtained in DFA for language $L = \{w/w$ contains at least two 0's and atmost one 1$\}$ over $\Sigma = \{0, 1\}$.

(A) 2 (B) 1
(C) 3 (D) 4

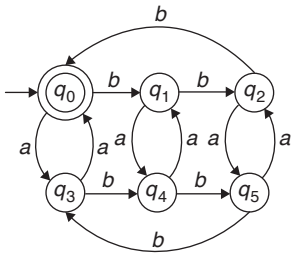
Practice Problems 2

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. Which of the following Regular expression is equal to given regular expression: $(b + aa^* b) + (b + aa^*b)\ (a + ba^* b)^*\ (a + ba^* b)$

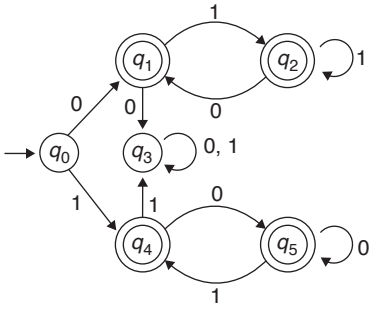
(A) $Ab\ (b + baa^*)$ (B) $a^*b\ (a + ba^* b)$
(C) $a^*\ b\ (a + ba^* b)^*$ (D) $ab\ (b + aa^*b)^*$

2. The following DFA accepts set of all strings over $\{a, b\}$ that contain



(A) Number of a's even and number of b's odd.
(B) Consecutive a's and b's
(C) Contain bbb as substring
(D) Number of a's even and number of b's divisible by three.

3. The regular language $L(r)$ for the given FSM is:

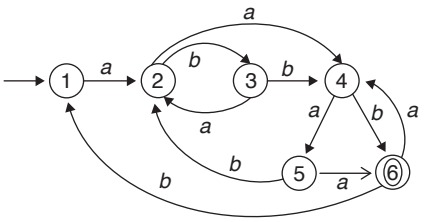


(A) It can start with zero followed by any number of 1's but no two consecutive 0's.
(B) It can start with 1, followed by any number of 0's but no two consecutive 1's.
(C) It is a combination of 0's and 1's but no two consecutive 0's or 1's.
(D) Both (A) and (B).

4. The language, L is defined as a set of non-palindromes over $\{a, b\}$. Is L regular?

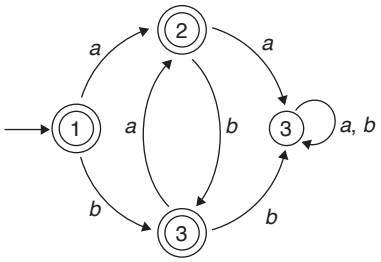
(A) Yes (B) No
(C) Cannot be determined (D) None of above

5. The DFA, for language, L over $\Sigma = \{a, b\}$ is given below. What will be number of states in minimized DFA.



(A) 4 (B) 6
(C) 2 (D) 3

6. The minimal DFA given below is defined for language, $L = \{w \in \{a, b\}^*\}$ over $\Sigma = \{a, b\}$. The 'L' is:



(A) Strings that contain equal number of a's and b's that have adjacent characters same.
(B) Contains adjacent characters same
(C) No two adjacent characters are same
(D) Starts and ends with same character that have adjacent character same.

7. The regular grammar $L(G)$ contains productions, P for language, $L = \{w \in \{a, b\}^*/$ there is at least one a$\}$ are:

(A) $S \rightarrow aS|bS|a|aT$
$T \rightarrow aT|bT|a|b$
(B) $S \rightarrow aS|bS|\varepsilon$
(C) $S \rightarrow aBb|bB$
$B \rightarrow a|b$
(D) $S \rightarrow bB$
$B \rightarrow b|\varepsilon$

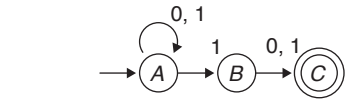
8. The regular expression for a language is defined as $((a^* b)^* (bc^*)^*)$. The total number of final states obtained in both NFA and DFA are respectively:

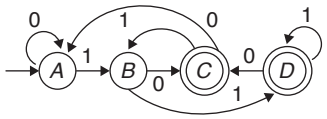
(A) 4, 2 (B) 1, 3
(C) 1, 5 (D) 2,3

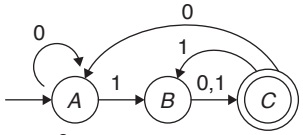
9. The language, L is defined as $\{w/w$ has n occurrences of 0's where n mod 5 is 3$\}$ over $\Sigma = \{0, 1\}$. The number of final states obtained in the DFA for L is:

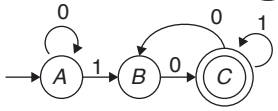
(A) 4 (B) 5
(C) 1 (D) 2

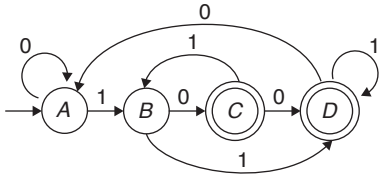
10. Which of the following is an equivalent DFA for the following NFA?



(A)


(B)


(C)


(D)


11. A regular grammar over alphabet $\Sigma = \{a, b, c, d\}$ whose language, is set of strings that contain exactly two b's is:

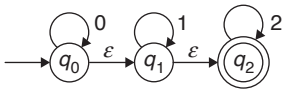
(A) $S \to aS|bS|cS|dA$
$A \to aA|bA|cA|dA|\varepsilon$

(B) $S \to aS|cS|dS|bB$
$B \to aB|cB|dB|bC$,
$C \to aC|cC|dC|\varepsilon$

(C) $S \to aS|bS|cS|dA$
$A \to aA|bB|cC$
$B \to b$
$C \to c$

(D) None of above

12. The following NFA contains ε-moves with 5, transitions. If this NFA with ε-moves is converted to NFA without ε-moves, what will be total number of transitions in obtained NFA?



(A) 5 (B) 4
(C) 6 (D) 3

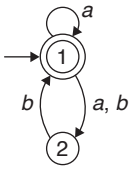
13. The regular expression, $r = (a + b)^*$. One more regular expression which represents same regular expression 'r' is:

(A) $a^* + b^*$ (B) $a^* \cdot b^*$
(C) $a^*(ba^*)^*$ (D) $(a + b)^* (a + b)$

14. The Regular grammar, $L(G)$ is defined for L with productions as $S \to Aab$, $A \to Aab|aB$, $B \to a$. What is Language generated by $L(G)$?

(A) Containing alternative a's and b's
(B) Containing alternative a's and b's, begins with an 'a' and ends with a 'b'.
(C) 'aa' followed by at least one set of alternating ab's.
(D) Consecutive aa's followed by 'b'.

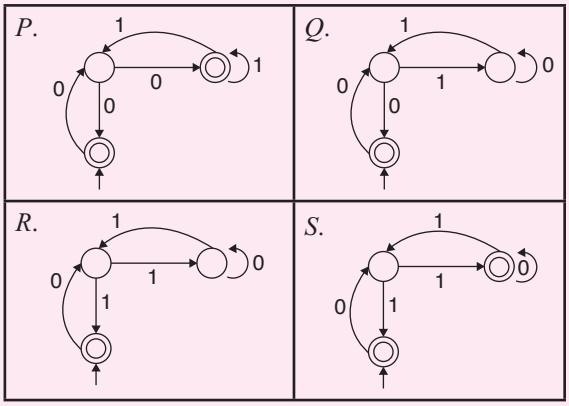
15. The number of final states in DFA after converting the NFA given below is:



(A) 4 (B) 2
(C) 3 (D) 1

Previous Years' Questions

1. Match the following NFAs with the regular expressions they correspond to **[2008]**



1. $\in + 0(01^*1 + 00)^*01^*$
2. $\in + 0(10^*1 + 00)^*0$
3. $\in + 0(10^*1 + 10)^*1$
4. $\in + 0(10^*1 + 10)^*10^*$

(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

2. Which of the following are regular sets?

I. $\{a^n b^{2m} \mid n \geq 0, m \geq 0\}$
II. $\{a^n b^m \mid n = 2m\}$
III. $\{a^n b^m \mid n \neq m\}$
IV. $\{xcy \mid x, y \in \{a, b\}^*\}$ **[2008]**

(A) I and IV only (B) I and III only
(C) I only (D) IV only

3. Which one of the following languages over the alphabet {0, 1} is described by the regular expression: $(0 + 1)^*0(0 + 1)^*0(0 + 1)^*$? **[2009]**

(A) The set of all strings containing the substring 00.
(B) The set of all strings containing atmost two 0's.
(C) The set of all strings containing at least two 0's.

(D) The set of all strings that begin and end with either 0 or 1.

4. Which one of the following is FALSE? **[2009]**

(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 non-deterministic PDA can be converted to an equivalent deterministic PDA.

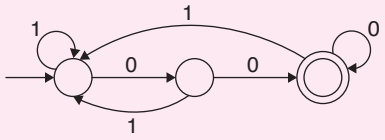
5. Match all items in Group 1 with correct options from those given in Group 2. **[2009]**

	Group 1		Group 2
P.	Regular expression	1.	Syntax analysis
Q.	Pushdown 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

6.



The above DFA accepts the set of all strings over {0, 1} that **[2009]**

(A) Begin either with 0 or 1

(B) End with 0

(C) End with 00

(D) Contain the substring 00.

7. Let $L = \{w \in (0 + 1)^* \mid w$ has even number of 1's$\}$, 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? **[2010]**

(A) $(0^*10^*1)^*$ (B) $0^*(10^*10^*)^*$

(C) $0^*(10^*1^*)^*0^*$ (D) $0^*1(10^*1)^*10^*$

8. Consider the languages $L_1 = \{0^i1^j \mid i \neq j\}$. $L_2 = \{0^i1^j \mid i = j\}$, $L_3 = \{0^i1^j \mid i = 2j + 1\}$. $L_4 = \{0^i1^j \mid i \neq 2j\}$. Which one of the following statements is true? **[2010]**

(A) Only L_2 is context free

(B) Only L_2 and L_3 are context free

(C) Only L_1 and L_2 are context free

(D) All are context free

9. Let w be any string of length n in $\{0, 1\}^*$. Let L be the set of all substrings of w. What is the minimum number of states in a non-deterministic finite automaton that accepts L? **[2010]**

(A) n-1 (B) n

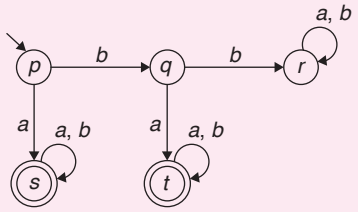
(C) n+1 (D) 2^{n-1}

10. Let P be a regular language and Q be a context-free language such that $Q \subseteq P$ (For example let P be the language represented by the regular expression p^*q^* and Q be $\{p^nq^n\}$ $n \in N\}$, Then which of the following is ALWAYS regular? **[2011]**

(A) $P \cap Q$ (B) $P - Q$

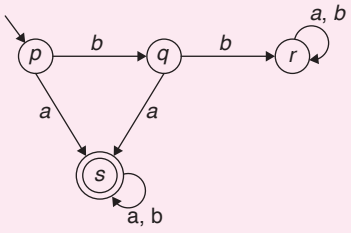
(C) $\Sigma^* - P$ (D) $\Sigma^* - Q$

11. A deterministic finite automaton (DFA) D with alphabet $\Sigma = \{a, b\}$ is given below.

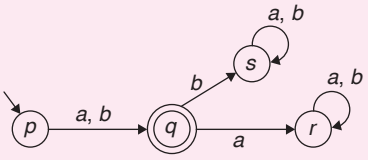


Which of the following finite state machines is a valid minimal DFA which accepts the same language as D? **[2011]**

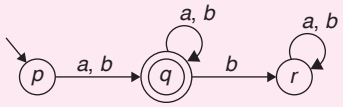
(A)



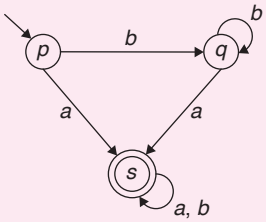
(B)



(C)



(D)



12. Given the language $L = \{ab, aa, baa\}$, which of the following strings are in L^*? **[2012]**

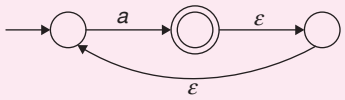
(1) *abaabaaabaa* (2) *aaaabaaaa*

(3) *baaaaabaaaab* (4) *baaaaabaa*

(A) 1, 2 and 3 (B) 2, 3 and 4

(C) 1, 2 and 4 (D) 1, 3 and 4

13. What is the complement of the language accepted by the NFA shown below?

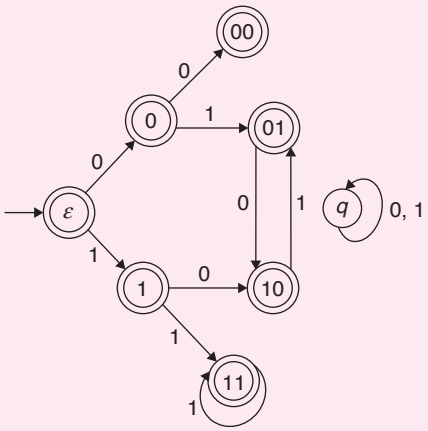


Assume $\Sigma = \{a\}$ and ε is the empty string. **[2012]**

(A) $\varnothing$ (B) $\{\varepsilon\}$

(C) a^* (D) $\{a, \varepsilon\}$

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



The missing arcs in the DFA are **[2012]**

(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)

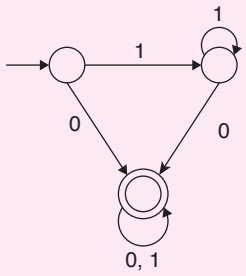
	00	01	10	11	q
00		1			0
01				1	
10	0				
11			0		

15. Consider the languages $L_1 = \Phi$ and $L_2 = \{a\}$. Which one of the following represents $L_1 L_2^* \cup L_1^*$? **[2013]**

(A) $\{\in\}$ (B) Φ

(C) a^* (D) $\{\in, a\}$

16. Consider the DFA A given below



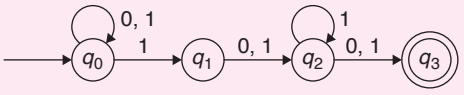
Which of the following are FALSE?

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. **[2013]**

(A) 1 and 3 only (B) 2 and 4 only

(C) 2 and 3 only (D) 3 and 4 only

17. Consider the finite automaton in the following figure. **[2014]**



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) $\{q_3\}$

18. If $L_1 = \{a^n | n \geq 0\}$ and $L_2 = \{b^n | n \geq 0\}$, consider the statements **[2014]**

(I) $L_1 \cdot L_2$ is a regular language

(II) $L_1 \cdot L_2 = \{a^n b^n | n \geq 0\}$

Which one of the following is CORRECT?

(A) Only (I) (B) Only (II)

(C) Both (I) and (II) (D) Neither (I) nor (II)

19. Let $L_1 = \{w \in \{0, 1\}^* | w$ has at least as many occurrences of (110)'s as (011)'s$\}$. Let $L_2 = \{w \in \{0, 1\}^* | w$ has at least

as many occurrences of (000)'s as (111)'s}. Which one of the following is TRUE? **[2014]**

(A) L_1 is regular but not L_2
(B) L_2 is regular but not L_1
(C) Both L_1 and L_2 are regular
(D) Neither L_1 nor L_2 are regular

20. The length of the shortest string NOT in the language (over $\Sigma = \{a, b\}$) of the following regular expression is _____. **[2014]**

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

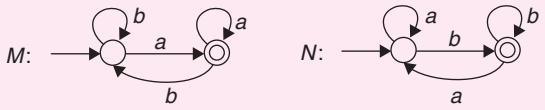
21. Let Σ be finite non-empty alphabet and let 2^{Σ^*} be the power set of Σ^{Σ^*}. Which one of the following is TRUE? **[2014]**

(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

1. $\varepsilon + 0\,(01^*\,1 + 00)^*\,01^*$
2. $\varepsilon + 0\,(10^*\,1 + 00)^*\,0$
3. $\varepsilon + 0\,(10^*\,1 + 10)^*\,1$
4. $\varepsilon + 0\,(10^*\,1 + 10)^*\,10^*$

(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) $P-3, Q-2, R-1, S-4$

22. Consider the DFAs M and N given above. The number of states in a minimal DFA that accepts the language $L(M) \cap L(N)$ is _____ **[2015]**



23. The number of states in the minimal deterministic finite automaton corresponding to the regular expression $(0 + 1)^*(10)$ is _____ **[2015]**

24. Which of the following languages is/are regular? **[2015]**

L_1: $\{wxw^R | w, x \in \{a, b\}^*$ and $|w|, |x| > 0\}$, w^R is the reverse of string w

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

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

(A) L_1 and L_3 only (B) L_2 only
(C) L_2 and L_3 only (D) L_3 only

25. Consider the alphabet $\Sigma = \{0, 1\}$, the null/empty string λ and the sets of strings X_0, X_1 and X_2 generated by the corresponding non-terminals of a regular grammar. X_0, X_1 and X_2 are related as follows

$X_0 = 1X_1$
$X_1 = 0X_1 + 1X_2$
$X_2 = 0X_1 + \{\lambda\}$

Which one of the following choices precisely represents the strings in X_0? **[2015]**

(A) $10(0^* + (10)^*)1$
(B) $10(0^* + (10)^*)^*1$
(C) $1(0 + 10)^*1$
(D) $10(0 + 10)^*1 + 110(0 + 10)^*1$

26. Let L be the language represented by the regular expression Σ^* 0011 Σ^* where $\Sigma = \{0, 1\}$. What is the minimum number of states in a DFA that recognizes $\overline{L}$ (complement of L)? **[2015]**

(A) 4 (B) 5
(C) 6 (D) 8

27. Which of the following languages is generated by the given grammar? **[2016]**

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

(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) $\{a^n \mid n \geq 0\}\ U\ \{b^n \mid n \geq 0\}\ U\ \{a^n b^n \mid n \geq 0\}$
(D) $\{a, b\}^*$

28. Which of the following decision problems are undecidable? **[2016]**

I. Given NFAs N_1 and N_2, is $L(N_1) \cap L(N_2) = \Phi$?
II. Given a CFG $G = (N, \Sigma, P, S)$ 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

29. Which one of the following regular expressions represents the language: *the set of all binary strings having two consecutive 0's and two consecutive 1s?* **[2016]**

(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

30. 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 _____ . **[2016]**

31. Language L_1 is defined by the grammar: $S_1 \rightarrow aS_1b \mid \in$
Language L_2 is defined by the grammar: $S_2 \rightarrow abS_2 \mid \in$

Consider the following statements:
P: L_1 is regular
Q: L_2 is regular
Which one of the following is TRUE? **[2016]**
(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

32. 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 $\cap$ B is regular.

Which one of the following is **CORRECT**? **[2016]**
(A) Only **I** is true
(B) Only **II** is true
(C) Both **I** and **II** are true
(D) Both **I** and **II** are false

33. 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* _____. **[2017]**

34. The minimum possible number of states of a deterministic finite automaton that accepts the regular language $L = \{w_1 a w_2 |\ w_1, w_2 \in \{a, b\}^*, |w_1| = 2, |w_2| \geq 3\}$ is _____. **[2017]**

35. Let δ denote the transition function and $\hat{\delta}$ denote the extended transition function of the $\in$-NFA whose transition table is given below:

δ	$\in$	a	b
$\rightarrow q_0$	$\{q_2\}$	$(q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$	$\{q_3\}$
q_2	$\{q_0\}$	$\varnothing$	$\varnothing$
q_3	$\varnothing$	$\varnothing$	$(q_2\}$

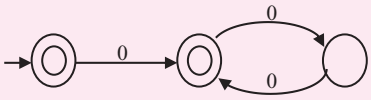
Then $\hat{\delta}(q_2, aba)$ is **[2017]**
(A) $\varnothing$ (B) $\{q_0, q_1, q_3\}$
(C) $\{q_0, q_1, q_2\}$ (D) $\{q_0, q_2, q_3\}$

36. Let N be an NFA with n states. Let k be the number of states of a minimal DFA which is equivalent to N. Which one of the following is necessarily true? **[2018]**
(A) $k \geq 2^n$ (B) $k \geq n$
(C) $k \leq n^2$ (D) $k \leq 2^n$

37. Given a language L, define L^i as follows:

$$L^0 = \{\varepsilon\}$$
$$L^i = L^{i-1}.\ L \text{ for all } i > 0$$

The order of a language L is defined as the smallest k such that $L^k = L^{k+1}$. Consider the language L_1 (over alphabet 0) accepted by the following automaton.



The order of L_1 is _____. **[2018]**

Answer Keys

Exercises

Practice Problems 1

1. D	**2.** B	**3.** C	**4.** D	**5.** B	**6.** B	**7.** D	**8.** D	**9.** C	**10.** C
11. A	**12.** D	**13.** C	**14.** A	**15.** A					

Practice Problems 2

1. C	**2.** D	**3.** D	**4.** B	**5.** B	**6.** C	**7.** A	**8.** C	**9.** C	**10.** A
11. B	**12.** C	**13.** C	**14.** C	**15.** B					

Previous Years' Questions

1. C	**2.** A	**3.** C	**4.** D	**5.** B	**6.** C	**7.** B	**8.** D	**9.** C	**10.** C
11. A	**12.** C	**13.** B	**14.** D	**15.** A	**16.** D	**17.** A	**18.** A	**19.** A	**20.** C
21. C	**22.** 1	**23.** 3	**24.** A	**25.** C	**26.** B	**27.** D	**28.** C	**29.** B	**30.** 2
31. C	**32.** B	**33.** 4	**34.** 8	**35.** C	**36.** D	**37.** 2			

Chapter 2

Context Free Languages and Push Down Automata

LEARNING OBJECTIVES

- *Context free grammar*
- *Context free language*
- *Ambiguity in context free grammars*
- *Removing ⍰-productions*
- *Removing unit productions*
- *Normal forms*
- *Chomsky's normal form*
- *Greiback normal form*
- *Closure properties of CFL's*
- *Push down automata*
- *PDAs accepting by final state and empty stack are equivalent*
- *Converting CFG to PDA*
- *Deterministic PDA*

Context Free Grammar

- A context free grammar (CFG) is a finite set of variables (non-terminals) each of which represents a language. The language represented by variables is described recursively in terms of each other. The primitive symbols are called terminals.
- The rules relating variables are called productions. A typical production states that the language associated with a given variable contains strings that are formed by concatenating strings from languages of certain other variables.
- CFG is a collection of three things;
  An alphabet Z of letters called terminals.
  A set of symbols called non-terminals, one of which is a start symbol, S.
  A finite set of productions of the form:
  One terminal $\rightarrow$ finite set of terminals and/or non-terminals.
- A CFG is defined as: $G = (V, T, P, S)$

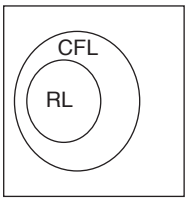
  Where

  $V \rightarrow$ Finite set of variables (non-terminals)
  $T \rightarrow$ Finite set of terminals (symbols)
  $P \rightarrow$ Finite set of productions, each, production is of the form, $A \rightarrow \alpha, A \in V, \alpha \in (V \cup T)^*$
  $S \rightarrow$ Start symbol

Context Free Language (CFL)

The language generated by CFG is a set of all strings of terminals that can be produced from start symbols, using the productions as substitutions. A language generated by a CFG is called context free language (CFL).

Note: Every regular grammar is context free, so a regular language (RL) is also context free.
Family of RL's is proper subset of CFL's.
i.e., RL $\subset$ CFL

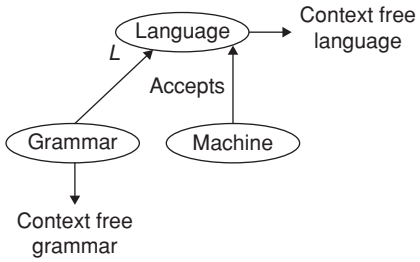


Solved Examples

Example 1: What is the language that is generated by CFG, $G = S \rightarrow AB|A \rightarrow +/-|B \rightarrow CB/C|C \rightarrow 0/1/2/ \ldots 9$.
(A) Set of all rational numbers
(B) Set of all integers
(C) Set of all natural numbers
(D) Set of all complex numbers

Solution: (B)
$S \rightarrow AB|A \rightarrow +/-|B \rightarrow CB/C|C \rightarrow 0/1/2/ \ldots 9$
Consider-18 (integer)

$S \rightarrow AB$
$\rightarrow -B$
$\rightarrow -CB$
$\rightarrow -1B$
$\rightarrow -18$



Ambiguity in Context Free Grammars

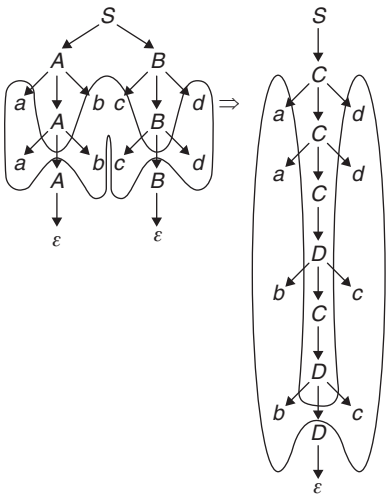
A CFG, G is called ambiguous if there is $w \in L(G)$ such that w has (at least) two different parse trees with respect to G.

Example 2: The language, $L = \{a^n\ b^n\ c^m\ d^m / n \geq 0,\ m \geq 0\} \cup \{a^n b^m c^m d^n / n \geq 0,\ m \geq 0\}$ is designed in CFG, G. The Grammar is
(A) Ambiguous
(B) Unambiguous
(C) Cannot be determined
(D) None of above

Solution: (A)
CFG G for given language L is:

$S \rightarrow AB|C$
$A \rightarrow aAb|\varepsilon$
$B \rightarrow cBd|\varepsilon$
$C \rightarrow aCd|$D
$D \rightarrow bDc|\varepsilon$

It's an inherently ambiguous grammar.
Consider string, aabbccdd



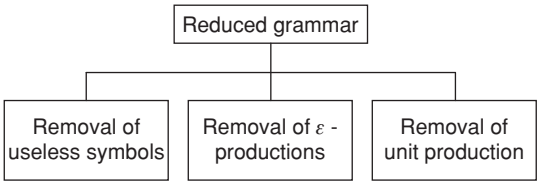
Note:
- A context free language with property that all grammars that generate it are ambiguous is inherently ambiguous.
- Inherently ambiguous grammars cannot convert to unambiguous grammars.

Minimization of Context Free Grammar

- Grammar may consist of some extra symbols (non-terminals). Having extra symbols unnecessarily increases the length of grammar.
- Simplification of grammar means reduction of grammar.

The properties of reduced grammar are:

1. Each variable (non-terminal) and each terminal of G appears in the derivation of some word in L.
2. There should not be any production as $X \rightarrow Y$ where X and Y are non-terminals.
3. If ε is not in language L, then there need not be production $X \rightarrow \varepsilon$.



Removal of Useless Symbols

- Any symbol is useful when it appears on right hand side, in the production rule and generates some terminal string. If no such derivation exists, then it is supposed to be a useless symbol.
- A symbol P is useful, if there exists some derivation

$$S^* \Rightarrow \alpha PB \text{ and } \alpha PB \overset{*}{\Rightarrow} W$$

Then P is said to be useful symbol.

Example 3: A grammar G', is generated by removing useless symbols from G defined below. The obtained G' contains productions:

$S \rightarrow aA|bB$
$A \rightarrow aA|a$
$B \rightarrow bB$
$D \rightarrow ab|Ea$
$E \rightarrow aC|d$

(A) $S \rightarrow aA$
$A \rightarrow aA|a$

(B) $S \rightarrow aS|bA|C$
$A \rightarrow a$
$C \rightarrow aCd$

(C) $S \rightarrow aA|bB$
$A \rightarrow aA|a$
$B \rightarrow bB$

(D) Cannot remove useless symbols

Solution:
$S \to aA \to aaA \to aaaA \to aaaa$ ✓
$B \to bB \to bbB \to bbbB \to bbbbB$ ..... (string cannot be generated)
$\therefore$ B is useless
D and E cannot be generated from 'S'. So, eliminate. Hence G' contains
$\therefore$ $S \to aA$
$A \to aA|a$

Removing ε-Productions

A production of the form $A \to \varepsilon$ is called an ε-production. If A is a non-terminal and $A \to^{(*)} \varepsilon$, then A is called a 'nullable non-terminal'. So eliminate such productions without changing meaning of grammar.

Example 4: The grammar, G is given below. The CFG generated after eliminating ε-production is:

$S \to ABC$
$A \to BC|a$
$B \to bAC|\in$
$C \to cAB|\in$

(A) $S \to ABC|AB|BC|CA$
$A \to BC|B|C$
$B \to bAC|bA|bC$
$C \to cAB|cA|cB$

(B) $S \to ABc$
$A \to BC$
$B \to bAC$
$C \to cAB$

(C) $S \to ABC|BC|AC|AB|A|B|C$
$A \to BC|B|C|a$
$B \to bAC|bA|bC|b$
$C \to cAB|cA|cB|c$

(D) None of these

Solution (C)
$B \to \in, C \to \in$
$\Rightarrow$ $A \to \in$
$\therefore$ Remove ε-productions and obtained CFG is Choice (C).

Removing Unit Productions

- A production of form $A \to B$, where A and B are both non-terminals, is called a 'unit production'.
- Presence of unit production in a grammar increases the cost of derivation.

Example 5: The total number of productions obtained by removing unit production from the Grammar,

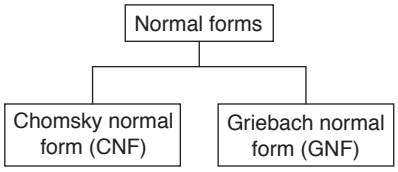
$A \to PQ$
$P \to 0$
$Q \to R|1$
$R \to S$
$S \to W|1R$
$W \to 2|P1$

(A) 9 (B) 2
(C) 3 (D) 5

Solution: (A)
$A \to PQ$
$Q \to R \to S \to W \to 2$
$\Rightarrow$ $Q, R, S \to$ Unit production
$A \to PQ$
$P \to 0$
$Q \to R|1$
$\downarrow$
$\Rightarrow$ $Q \to 2|P1|1R|1$ (substitute the production of R, S, W)
$\therefore$ $A \to PQ$
$P \to 0$
$Q \to 2|P1|1R|1$
$R \to 2|P1|1R$
$\therefore$ 9 – Productions

Normal Forms

- It is necessary to have a grammar in some specific form so, grammar normalization is needed.



That is, There should be fixed number of terminals and non-terminals, in CFG.

Chomsky's Normal Form (CNF)

- A context free grammar (CFG), $G = (V, \Sigma, R, S)$ is said to be in CNF, if and only if every rule in R is of one of the following forms
  1. $A \to a$, for some $A \in V$ and some $a \in \Sigma$
  2. $A \to BC$, for some $A \in V$ and $B, C \in V \cup \{S\}$
  3. $S \to \varepsilon$
- Every rule either replaces a variable by a single character or by a pair of variables except the start symbol and the only rule that can have the empty word as it's right hand side must have start symbol as it's left hand side.

Note: Every parse tree for a grammar in CNF must be a binary tree and the parse tree for any non-empty word cannot have any leaves labeled with ε in it.

Transforming of a grammar to CNF

- In order to construct the grammar G in CNF that is equivalent to a given grammar G, first identify how exactly G can violate the rules for a CNF. Since CNF only restricts the rules in G, see only at R. The 'bad' cases of rules are:
- $A \to uSv$ where $A \in V$ and $u, v \in (V \cup \Sigma)*$. The start symbol must not appear on the right-hand side of any rule. This is called 'start symbol rule'.

- **To remove 'start symbol rule',** add a new symbol, so make it the start symbol in new grammar G_1, and add the single rule $S_0 \to S$ to R to get the rules for G_1. Since S_0 does not appear in any rules, the new grammar has no start symbol rules.
  - $A \to \varepsilon$ where $A \in V \cup \{S\}|$. The only symbol that can be replaced by the word is start symbol. This is called 'ε-rules'.
- **To remove 'ε-rules',** identify all variables that can yield the empty string, either directly or indirectly.

  These variables are 'nullable'. Remove all direct rules $A \to \varepsilon$ from the grammar and fix up the grammar by removing all occurrences of nullable variables from the right hand sides of all rules.

  $A \to B$ where $A, B \in V$. The only rules involving variables on the right-hand side must have exactly two of them. This is called 'unit rules'.
- **To remove 'Unit rules',** identify a set of unit pairs. These are pairs of symbols (A, B), where $A \overset{*}{\Rightarrow} B$. Then remove all unit rules by copying right-hand sides. If there is a rule $A \to B$, (A, B) is a unit pair. Then, if there is a rule $B \to W$, derive W from A by $A \to B$, $B \to W$. To remove the unit rule and still generate an equivalent grammar, add the right-hand side W to the rules for A directly, $A \to W$.

  $A \to W$ where $A \in V$, $W \in (V \cup \Sigma)^*$ and W contains at least one character and at least one variable. The only rules where character appear on right-hand side must have exactly one character as right-hand side. This is called 'mixed rules'.
- **To remove 'mixed rules',** Let $A \to W \in R_3$ is a mixed rules. Then write W as $W = V_0\, C_1 V_1 \ldots V_{n-1} C_n V_n$, where $C_i \in \Sigma$ are occurrences of characters, and the $V_i \in V^*$ are strings of only variables. Then add a new symbol, C_i to V_4 for every character C_i and add the rules $C_i \to c_i$ to R_4. Finally define $W^1 := V_0\, C_1\, V_1 \ldots V_{n-1}\, C_n\, V_n \in V^*$ and add rules $A \to W'$ to R_4. If the rule $A \to W$ is part of the derivation for some word, replace that single rule by applying rule $A \to W'$ first and then replacing all C_i by c_i using their respective rules.

  $\boxed{A \to w}$ Where $A \in V$ and $W \in (V \cup \Sigma)^*$ with $|w| > 2$. Rules must have one symbol (character) or two variables (two variables as right hand side). These are called long rules.
- **To remove 'long rules',** Let $A \to B_1 \ldots B_n$ be a long rule, i.e., $n > 2$. B_i is all variables. Break up every single long rule, into several 'short' rules, by introducing new 'helper variables' and splitting right hand side from left to right: add new symbols $A_1, \ldots A_{n-2}$ to set of variables and add following rules to R_5: $A \to B_1 A_1, A_1 \to B_2 A_2, \ldots A_{n-2} \to B_{n-1} B_n$.

Example 6: Consider grammar, $G = S \to ASB$, $A \to aAS|a|\varepsilon$, $B \to SbS|A|bb$. The CNF generated contains ____ non-terminals.

(A) 5 (B) 6
(C) 9 (D) 11

Solution: (C)

Add new start state:

$S_0 \to S$
$S \to ASB$
$A \to aAS|a|\varepsilon$
$B \to SbS|A|bb$

Eliminate ε-rules

$A \to \varepsilon$:
$S_0 \to S$
$S \to ASB|SB$
$A \to aAS|a|aS$
$B \to SbS|A|bb|\varepsilon$

Eliminate $B \to \varepsilon$:

$S_0 \to S$
$S \to ASB|SB|S|AS$
$A \to aAS|a|aS$
$B \to SbS|A|bb$

Remove Unit rules:

$B \to A$:
$S_0 \to S$
$S \to ASB|SB|S|AS$
$A \to aAS|a|aS$
$B \to SbS|bb|aAS|a|aS$

$S \to S$:
$S_0 \to S$
$S_0 \to ASB|SB|AS$
$A \to aAS|a|aS$
$B \to SbS|bb|aAS|a|aS$

$S_0 \to S$:
$S_0 \to ASB|SB|AS$
$S \to ASB|SB|AS$
$A \to aAS|a|aS$
$B \to SbS|bb|aAS|a|aS$

Replace rules which have more than two symbols:

$S_0 \to ASB$: $S_0 \to AU_1$ and $U_1 \to SB$
$\therefore S_0 \to AU_1|SB|AS$
$S \to AU_1|SB|AS$
$A \to aAS|a|aS$
$B \to SbS|bb|aAS|a|aS$
$U_1 \to SB$
$A \to aAS \Rightarrow A \to aU_2$ and $U_2 \to AS$ and $B \to SbS$
$\Rightarrow B \to SU_3$ and $U_3 \to bS$
$\therefore S_0 \to AU_1|SB|AS$
$S \to AU_1|SB|AS$
$A \to aU_2|a|aS$
$B \to SU_3|bb|aU_2|a|aS$
$U_1 \to SB$
$U_2 \to AS$
$U_3 \to bS$

Eliminate rules which have terminals and variables or two terminals.

Let $V_1 \to a$, $V_2 \to b$
$\therefore S_0 \to AU_1|AS|SB$
$S \to AU_1|SB|AS$
$A \to V_1U_2|a|V_1S$

$B \rightarrow SU_3|V_2V_2|V_1U_2|a|V_1S$
$U_1 \rightarrow SB$
$U_2 \rightarrow AS$
$U_3 \rightarrow V_2S$
$V_1 \rightarrow a$
$V_2 \rightarrow b$
$\therefore$ Nine non-terminals.

Greiback Normal Form (GNF)

- A CFG, $G = (V, T, R, S)$ is said to be in GNF, if every production is of form $A \rightarrow a\alpha$ where $a \in T$, $\alpha \in V^*$, i.e., α is a string of zero or more variables.
- Left recursion in R can be eliminated by following schema: If $A \rightarrow A\alpha_1 |A\alpha_2| \ldots |A\alpha_r|\beta_1|\beta_2| \ldots |\beta_s$, then replace the above rules by
  (i) $A \rightarrow \beta_i|\beta_iZ, 1 \le i \le s$
  (ii) $Z \rightarrow \alpha_i| \alpha_iZ, 1 \le i \le r$
- If $G = (V, T, R, S)$ is a CFG, then another CFG, $G_1 = (V_1, T, R_1, S)$ can be constructed in GNF $\exists$ $L(G_1) = L(G) - \{\varepsilon\}$.

The step wise algorithm is as follows:

1. Eliminate null production, unit productions and useless symbols from the grammar G and then construct a $G^1 = (V^1, T, R^1, S)$ in CNF generating the language $L(G^1) = L(G) - \{\varepsilon\}$.
2. Rename the variables like $A_1, A_2, \ldots A_n$ starting with $S = A_1$.
3. Modify the rules in R^1, so that if $A_i \rightarrow A_j\gamma \in R^1$ then $j > i$.
4. Starting with A_1 and proceeding to A_n, can be obtained as:
   (a) Assume that productions have been modified so that for $1 \le i \le k$, $A_i \rightarrow A_j\gamma \in R^1$ only if $j > i$
   (b) If $A_k \rightarrow A_j\gamma$ is a production with $j < k$, generate a new set of productions substituting for A_j, the body of each A_j production.
   (c) Repeating (b) atmost $k-1$ times, obtains rules of the form $A_k \rightarrow A_p\gamma, p \ge k$.
   (d) Replace rules $A_k \rightarrow A_k\gamma$ by removing left-recursion.
5. Modify the $A_i \rightarrow A_j\gamma$ for $i = n-1, n-2, \ldots 1$ in desired form at same time change z production rules.

Example 7: A grammar G is defined with rules $S \rightarrow XA|BB$, $B \rightarrow b|SB$, $X \rightarrow b$, $A \rightarrow a$. The normalized GNF of G contains ____ productions.
(A) 17 (B) 19
(C) 5 (D) 16

Solution: (B)
1. The Grammar, G is already in CNF.
2. Re-label with variables
   S with A_1
   X with A_2
   A with A_3
   B with A_4
   Grammar, G now is:
   $A_1 \rightarrow A_2A_3|A_4A_4$
   $A_4 \rightarrow b|A_1A_4$
   $A_2 \rightarrow b$
   $A_3 \rightarrow a$
3. Identify all productions which do not conform to any of the types listed below:
   $A_i \rightarrow A_j x_k \exists j > i$
   $Z_i \rightarrow A_j x_k \exists j \le n$
   $A_i \rightarrow a\, x_k \exists x_k \in V^*$ and $a \in T$
4. $A_4 \rightarrow A_1 A_4 \ldots$ identified
5. $A_4 \rightarrow A_1 A_4|b$
   To eliminate A_1, use substitution rule, $A_1 \rightarrow A_2 A_3|A_4A_4$
   $\therefore$ $A_4 \rightarrow A_2 A_3A_4|A_4A_4A_4|b$
   Substitute $A_2 \rightarrow b$
   $\therefore$ $A_4 \rightarrow b A_3A_4|A_4A_4A_4|b$
   $A_4 \rightarrow A_4A_4A_4$ is left recursive. So, remove left recursion i.e., $A_4 \rightarrow b A_3A_4|b|bA_3A_4Z|bZ$
   $Z \rightarrow A_4A_4|A_4A_4Z$
6. Now, $G = A_1 \rightarrow A_2 A_3|A_4A_4$
   $A_4 \rightarrow b A_3A_4|b|b A_3A_4 Z|b Z$
   $Z \rightarrow A_4 A_4|A_4A_4Z$
   $A_2 \rightarrow b$
   $A_3 \rightarrow a$
7. A_1, Z are not in GNF. So,
   For $A_1 \rightarrow A_2A_3|A_4A_4$:
   Substitute for A_2 and A_4 to convert it to GNF
   $A_1 \rightarrow b A_3| b A_3 A_4A_4|b A_4|b A_3 A_4Z A_4|b Z A_4$
   For $Z \rightarrow A_4 A_4|A_4A_4Z$
   substitute for A_4 to convert it to GNF
   $Z \rightarrow b A_3 A_4A_4|b A_4|b A_3 A_4Z A_4|b Z A_4|b A_3 A_4A_4$
   $Z|b A_4Z|b A_3A_4 Z A_4Z|b Z A_4 Z$
   $\therefore$ Final GNF is:
   $A_1 \rightarrow b A_3|b A_3A_4A_4|b A_4| b A_3A_4Z A_4|b Z A_4$
   $A_4 \rightarrow b A_3 A_4|b|b A_3 A_4Z|b Z$
   $A_2 \rightarrow b$
   $A_3 \rightarrow a$
   $Z \rightarrow b A_3 A_4A_4|b A_4|b A_3A_4 Z A_4|b Z A_4|b A_3 A_4 A_4$
   $Z|\ b A_4 Z|b A_3 A_4 Z A_4Z|b Z A_4 Z$
   $\therefore$ 19 productions.

Pumping Lemma for Context Free Languages

Let 'L' be context free language. There exists some integer, m $\exists$ $\forall$w in L, with $|w| \ge m$, $w = uvxyz$ with $|vxy| \le m$ and $|vy| \ge 1$ $\exists$ $u\, v^i\, x\, y^i\, z \in L$ $\forall$ $i = 0, 1, 2, 3, \ldots$

Note: Pumping lemma is used to show that a language is Not context free.

Example 8: The language $\{a^n b^m c^n d^{(n+m)}: m, n \ge 0\}$ is
(A) Regular
(B) Context free but not regular
(C) Neither context free nor regular
(D) Cannot be determined

Solution: (C)
$L = \{a^n b^m c^n d^{(n+m)}: m, n \geq 0\}$
Clearly, L is not regular because, number of a's and number of b's must be known to compute number of d's.
'L' is not context free because, Let $w = a^M b^M c^M d^{2M}$. Clearly neither v nor y can cross regions and include more than one letter, since if that happened; letters obtained will be out of order when pumped.

So, consider cases, where v and y fall within a single region. Consider 4-regions corresponding to a, b, c and d.
(1, 1) → change number of a's and they won't match c's any more.
(1, 2) → If v is not empty, change a's and they won't match with c's. If y is non-empty, number of b's changed won't have right number of d's.
(1, 3), (1, 4) → ruled out. $\because\ |v\,x\,y| \leq M$
(2, 2) → Change number of b's and they won't match right number of d's.
(2, 3) → If v is non-empty, change number of b's without changing number of d's. If y is not empty, change c's and they'll no longer match a's.
(2, 4) → ruled out $\because\ |v\,x\,y| \leq M$
(3, 3) → Change number of c's and they won't match a's.
(3, 4) → If v is not empty change c's and they won't match a's. If y is not empty, change d's without changing b's.
(4, 4) → change d's without changing a's or b's.
$\therefore$ L is not context free.

Closure Properties of CFL's

1. **CFL's are closed under union:** For CFL's L_1, L_2 with CFG's G_1, G_2 and start variables S_1, S_2. The grammar of Union $L_1 \cup L_2$ has new start symbol S and additional production $S \to S_1 | S_2$
2. **CFL's are closed under concatenation:** For CFL's L_1, L_2 with CFG's G_1, G_2 and start variables S_1, S_2. The grammar of concatenation L_1L_2 has new start variables S and additional production: $S \to S_1 S_2$
3. **CFL's are closed under star operation:** For CFL L, with CFG G and start variable S. The grammar of the start operation L^* has new start variable S_1 and additional production:
$$S_1 \to S\,S_1 | \varepsilon$$
4. **CFL's are not closed under intersection:** If L_1, L_2 are two context free languages, $L_1 \cap L_2$ not necessarily be context free.
5. **CFL's are not closed under complement:** If L is context free language, $\overline{L}$ not necessarily be context free.
6. **Intersection of CFL's and regular language: (regular closure):** If L_1 is a CFL and R_2 is a regular language then $L_1 \cap L_2$ is a CFL.

Example 9: The language, $L_1 = \{a^n b^n : n \geq 0\}$ and $L_2 = \{a^{100} b^{100}\}$. The relation $L_1 \cap \overline{L_2}$ is _____
(A) Regular
(B) Context free
(C) Regular but not context fee
(D) Cannot be determined

Solution: (B)
$L_1 = \{a^n b^n : n > 0\}$ is context free
$L_2 = \{a^{100}\ b^{100}\}$ is regular
$\overline{L_2} = \{(a+b)^*\} - \{a^{100} b^{100}\}$ is regular
$\{a^n b^n\}$ context free
$\overline{L_2} = \{(a+b)^*\} - \{a^{100} b^{100}\}$ is regular
$\{a^n b^n\} \cap \overline{L_2}$ → context free
$\{a^n b^n\} \cap \overline{L_2} = \{a^n b^n :\ n \neq 100,\ n \geq 0\} = L$ is context free:

Table 1 *Comparing Regular and Context free Languages:*

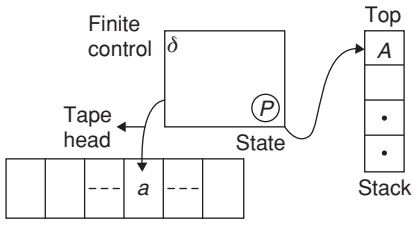
Regular Language	CFL
Regular expression or regular grammar	Context free grammar
Recognize the language	Parses the language
These are DFSA's	These are NDPDA's
Minimize FSA's	Find deterministic grammar.
Closed under: Concatenation Union Kleen star Complement Intersection	Closed under: Concatenation Union Kleen star

Push Down Automata (PDA)

A push down automata is merely a finite automata with a stack added to it.

PDA is used to generate context free language.

The stack allows for unbounded memorization.



Input tape: The tape is divided into finitely many cells. Each cell contains a symbol in an alphabet, Σ.
Stack: The stack head always scans the top symbol of the stack. It performs two basic operations.

- Push: Add a new symbol at the top
- Pop: Read and remove the top symbol

Tape head: The head scans at a cell on the tape and can read a symbol on the cell. In each move, the head can move to the right cell.

Finite control: The finite control has finitely many states which form a set Q. For each move, the state is changed according to the evaluation of transition function.

A PDA is defined as: $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$
Where Q: set of States
Σ: Input alphabet
Γ: Stack symbol
δ: Transition function
q_0: Start state
z_0: Initial stack top symbol
F: Final/accepting states
Transition functions δ: $Q \times \Gamma \times \Sigma \Rightarrow Q \times \Gamma$
Q: Old state
Γ: Stack top
Σ: Input symbol
Q: New state, Γ: New stack top

PDA's instantaneous description (IDs): A PDA has a configuration at any given instance: (q, w, y)

$q \rightarrow$ current state
$w \rightarrow$ remainder of input (i.e., unconsumed part)
$y \rightarrow$ current stack contents as a string from top to bottom of the stack.

If $\delta(q, a, x) = \{P, A\}$ is a transition, then following are also true:

- $(q, a, x) \vdash (P, \varepsilon, A)$
- $(q, aw, xB) \vdash (p, w, AB)$

Note: 1. $\rightarrow$: Turnstile notation and represents one move.
2. $\vdash^*$: represents sequence of moves.

Principles about IDs:

1. If for a PDA, $(q, x, A) \vdash^* (p, y, B)$, then for any string $w \in \Sigma^*$ and $\gamma \in \Gamma^*$, it is also true that: $(q, xw, A\gamma) \vdash^* (p, yw, B\gamma)$
2. If for a PDA, $(q, xw, A) \vdash^* (p, yw, B)$, then it is also true that: $(q, x, A) \vdash^* (p, y, B)$

Acceptance by PDA: There are two types of PDAs that one can design:

- Those that accept by final state or
- Those that accept by empty stack

PDAs that accept by final state: For a PDA, P, the language accepted by P, denoted by $L(P)$ by final state, is:

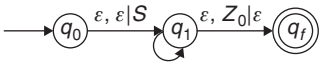
$$\{w | (q_0, w, z_0) \vdash^* (q, \in, A)\} \exists\, q \in F$$

PDAs that accept by empty stack: For a PDA P, the language accepted by P, denoted by $N(P)$ by empty stack, is:

$$\{w | (q_0, w, z_0) \vdash^* (q, \varepsilon, \varepsilon)\}, \text{ for any } q \in Q.$$

Example 10: Consider the grammar $S \rightarrow aTb | b$, $T \rightarrow Ta | \varepsilon$. The PDA constructed contains ____ states.
(A) 4 (B) 3 (C) 5 (D) 2

Solution: (B)
$S \rightarrow aTb | b$
$T \rightarrow Ta | \varepsilon$



$\varepsilon, S | aTb$
$\varepsilon, T | Ta$
$\varepsilon, S | b$
$\varepsilon, T | \varepsilon$
$a, a | \varepsilon$
$b, b | \varepsilon$

Let $S \rightarrow q_0$, $T \rightarrow q_1$
Consider string "aab" $S \rightarrow aTb \rightarrow aTab \rightarrow aab$
$\delta(q_0, aab, z_0) \vdash \delta(q_0, \varepsilon\, aab, z_0)$
$\vdash \delta(q_1, aab, q_0 z_0)$
$\vdash \delta(q_1, aab, aTb\, z_0)$
$\vdash \delta(q_1, ab, Tbz_0)$
$\vdash \delta(q_1, ab, aTbz_0)$
$\vdash \delta(q_1, b, Tbz_0)$
$\vdash \delta(q_1, b, \in bz_0)$
$\vdash \delta(q_1, b, bz_0)$
$\vdash \delta(q_1, \in, z_0)$
$\vdash \delta(q_f, \in) \rightarrow$ acceptance

PDAs accepting by final state and empty stack are equivalent:
$P_F \rightarrow$ PDA accepting by final state,
$P_F = (Q_F, \Sigma, \Gamma, \delta_F, q_0, z_0, F)$
$P_N \rightarrow$ PDA accepting by empty stack
$P_N = (Q_N, \Sigma, \Gamma, \delta_N, q_0, z_0)$

- For every P_N, $\exists\, P_F \exists\, L(P_F) = L(P_N)$
- For every P_F, $\exists\, P_N \exists\, L(P_N) = L(P_F)$

Converting CFG to PDA

The PDA simulates the left most derivation on a given w, and upon consuming it fully it either arrives at acceptance (by empty stack) or non-acceptance.

The steps to convert CFG to PDA are:

1. Push right hand side of the production on to stack, with left most symbol at the stack top.
2. If stack top is the left most variable, then replace it by all its productions (each possible substitution will represent a distinct path taken by non-deterministic PDA (NPDA).
3. If stack top has a terminal symbol and if it matches with the next symbol in the input string, then pop it. Follow from step-1 again to complete all productions.

Example 11: The CFG, G of a language L is $S \rightarrow AB$, $A \rightarrow aAb | \varepsilon$, $B \rightarrow cB | \varepsilon$. The PDA generated by G contains ____ states.
(A) 5 (B) 4 (C) 3 (D) 1

Solution: (C)

$S \to AB$
$A \to aAb|\varepsilon$
$B \to cB|\varepsilon$
$\Rightarrow \delta(q_0, w, S) = (q_1, AB)$
$\delta(q_1, w, A) = (q_1, aAb)$
$\delta(q_1, \varepsilon, A) = \delta(q_1, \varepsilon)$
$\delta(q_1, w, B) = (q_1, cB)$
$\delta(q_1, \varepsilon, B) = \delta(q_2, \varepsilon) \to$ accept
$\therefore$ $\{q_0, q_1, q_2\}$ 3-states.

Converting a PDA into a CFG

Given: $G = (V, T, P, S)$ Initial stack symbol (S) same as start variable in grammar
Output: $P_N = (\{q\}, T, V \cup T, \delta, q, S)$, where δ is

- If q_0 is start state in PDA and q_n is final state of PDA then $[q_0, z, q_n]$ becomes a start state of CFG. Here z represents stack symbol.
- The production rule for the ID of the form $\delta(q_i, a, z_0) = (q_{i+1}, z_1, z_2)$ can be obtained as:

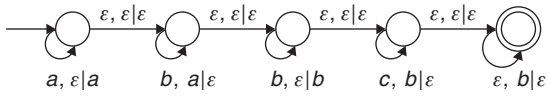
$$\delta(q_i, z_0, q_{i+k}) \to a(q_{i+1}, z_1, q_m)(q_m, z_2, q_{i+k})$$

Where q_{i+k}, q_m represents the intermediate staes, z_0, z_1, z_2 are stack symbols and a is input symbol.

- The production rule for the ID of the form $\delta(q_i, a, z_0) = (q_{i+1}, \varepsilon)$ can be converted as

$$(q_i, z_0, q_{i+1}) \to a$$

Example 12: The PDA, P for language L is generated as:



The CFG for P is:

(A) $S \to S_1S_2$
$S_1 \to aS_2b$
$S_2 \to c|\varepsilon$

(B) $S \to S_1bc$
$S_1 \to a|\varepsilon$

(C) $S \to S_1S_2$
$S_1 \to aS_1b|\varepsilon$
$S_2 \to bS_2|bS_2c|\varepsilon$

(D) $S \to aS_1c$
$S_1 \to b|\varepsilon$

Solution: (C)
The language, L generated by given PDA is

$$L = \{a^n b^n b^m c^p : m \geq p \text{ and } n, p \geq 0\}$$

It can be generated by following rules:

$$S \to S_1S_2$$

$S_1 \to aS_1b|\varepsilon \to S_1$ generates $a^n b^n$
$S_2 \to bS_2|bS_2c|\varepsilon \to S_2$ generates $b^m c^p$

Deterministic PDA (Deterministic CFL)

$$\left\{\begin{array}{c}\text{Deterministic}\\ \text{context free}\\ \text{languages}\\ \text{(DPDA)}\end{array}\right\} \subseteq \left\{\begin{array}{c}\text{Context-free}\\ \text{Languages}\\ \text{PDAs}\end{array}\right\}$$

- Every DPDA is also a PDA.
- A context free language 'L' accepted by PDA may or may not be accepted by DPDA.

A PDA, $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is deterministic if there is no configuration for which M has choice of more than one move. That is, it must satisfy the following conditions:

1. For any $q \in Q$, $a \in \Sigma\varepsilon$ and $s \in \Gamma\ \varepsilon$, the set $\delta(q, a, s)$ has almost one element. (Doesn't allow two or more transitions from same state).
2. For any $q \in Q$, and $s \in \Gamma\varepsilon$, if $\delta(q, \varepsilon, s) \neq \phi$, then $\delta(q, a, s) = \phi$ for every $a \in \Sigma$ and $\delta(q, a, \varepsilon) = \phi$ for all $a \in \Sigma\varepsilon$.
3. For any $q \in$ Q and $a \in \Sigma$, if $(q, a, \varepsilon) \neq \phi$, then $\delta(q, a, s) = \phi$ for all $s \in \Gamma$ and $\delta(q, \varepsilon, t) = \phi$ for all $t \in \Gamma_\varepsilon$.
4. For any $q \in Q$, if $\delta(q, \varepsilon, \varepsilon) \neq \phi$, then $\delta(q, a, t) = \phi$ for all $a \in \Sigma\varepsilon$ and $t \in \Gamma_\varepsilon$ (except when $a = \varepsilon$, $t = \varepsilon$).

Rule-2 says that if there is a transition from state q that reads character, s from stack but doesn't read other input, other transitions from q, that don't read stack are not allowed and other transitions from q that read s from the stack and read the input are not allowed either.

Rule-3 says that if there is a transition from state q that reads character a, but doesn't read stack, other transitions from q that don't read the input are not allowed and other transitions from q that read 'a' from input and read the stack are not allowed either.

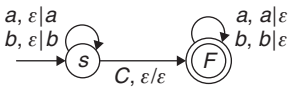
Rule-4 says that if there is a transition from q that doesn't read either input or stack, all other transitions from q are not allowed.

Example 13: A language, L is defined as: $L = \{wcw^R : w \in (a, b)^*\}$. What is Nature of language L?

(A) CFL and DCFL
(B) Only CFL
(C) Only DCFL
(D) None of these

Solution: (A)

$$L = \{x = wcw^R \text{ for } w \in (a, b)^*\}$$



Clearly, obtained PDA is also DPDA in sense; there is no choice in transitions.

$\therefore$ Hence L is CFL and DCFL.

EXERCISES

Practice Problems I

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. Consider the grammar, $G = (V, \Sigma, R, S)$ where $V = \{a, b, S, A\}$, $\Sigma = \{a, b\}$, $R = \{S \to AA, A \to AAA, A \to a, A \to bA, A \to Ab\}$ How many strings can be generated by $L(G)$ that can be produced by derivations of four or fewer steps?
(A) 5 (B) 10 (C) 14 (D) 8

2. Consider the following languages L_1, L_2 and L_3:
$L_1 = \{a^n b^m c^{n+m} | n, m \geq 0\}$
$L_2 = \{a^n b^{n+1} c^{n+2} | n \geq 0\}$
$L_3 = \{a^n b^n c^m | n, m \geq 0\}$
Which of following statement is true?
(A) L_1, L_2, L_3 are context free languages
(B) L_1, L_2 are context free but not L_3
(C) L_1, L_3 are context free but not L_2
(D) L_1, L_2, L_3 are not context free languages.

3. The language, $L = \{b_i \# b_{i+1} : b_i \text{ is } i \text{ in binary}, i \geq 1\}$ is:
(A) Regular
(B) Context free
(C) Regular and context free
(D) Neither context free nor Regular

4. The CFG, $G : A \to BAB|B|\varepsilon$, $B \to 00|\varepsilon$. The CFG is normalized using CNF. The obtained G', contains ___ rules.
(A) 11 (B) 14 (C) 12 (D) 13

5. The language $L = \{0^{2^i} : i \geq 1\}$ is:
(A) Context free
(B) DCFL
(C) Both CFL and DCFL
(D) Not context free language

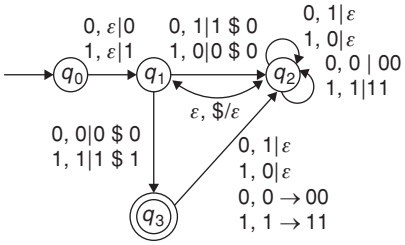
6. The context free grammar, G is defined with production rules $S \to EcC'|aAE|AU$, $A \to aA|\varepsilon$, $B \to bB|\varepsilon$, $C' \to cC'|\varepsilon$, $E \to aEc|F$, $F \to bFc|\varepsilon$, $U \to aUc|V$, $V \to bVc|bB$ What is the language generated by L?
(A) $L = \{a^n b^m c^k : k \neq n + m\}$
(B) $L = \{a^n b^m c^k : k = n + m\}$
(C) $L = \{a^n b^m c^k : k > n + m\}$
(D) $L = \{a^n b^m c^k : k < n + m\}$

7. Consider the grammar, $G \equiv S \to abScB|\varepsilon$, $B \to bB|b$. What language does it generate?
(A) $L(G) = \{(ab)^n (cb)^m | n = m\}$
(B) $L(G) = \{a^n b^n (cb)^m | n \neq m\}$
(C) $L(G) = \{(ab)^n (cb^m)^n | n \geq 0, m > 0\}$
(D) $L(G) = \{\{ab)^n (cb^m)^n | n \geq 0, m \geq 0\}$

8. The language, $L = \{0^i 1^j 2^k | i \neq j \text{ or } j \neq k\}$. The CFG, G generated by L contains ___ rules.
(A) 23 (B) 20
(C) 21 (D) 19

9. The DPDA constructed to accept language, L with property $L = L_1 \cup L_2$ where $L_1 = \{10^n 1^n | n > 0\}$, $L_2 = \{110^n 1^{2n} | n > 0\}$ contains ___ states.
(A) 4 (B) 5
(C) 6 (D) 7

10. The PDA is designed as:



What is the language generated by the above PDA?
(A) Binary strings that have same number of 0's and 1's.
(B) Binary strings that start with 00 and end with 11 and have same number of 0's and 1's.
(C) Binary strings that start and end with the same symbol and have same number of 0's and 1's.
(D) Binary strings that start with 11 and end with 00 and have same number of 0's and 1's.

11. The language, $L = (ba^{m_1} ba^{m_2} b \cdots ba^{m_n} : n \geq 2, m_1, \ldots m_n \geq 0$ and $m_i \neq m_j$ for some $i, j)$. What is nature of 'L'?
(A) Regular
(B) Context free but not regular
(C) Regular but not context free
(D) Neither context free nor regular

12. Two languages L_1, L_2 are defined as:
$L_1 = \{a^i b^j c^k : i, j, k \geq 0, i = j\}$
$L_2 = \{a^i b^j c^k : i, j, k \geq 0, j = k\}$ which of following statements are true?
(i) $L_1 \cap L_2$ is context free
(ii) $L_1 \cap L_2 = \{a^n b^n c^n | n \geq 0\}$
(iii) L_1, L_2 are context free
(iv) Only L_1 is context free
(A) All are true (B) (i), (ii) are true
(C) (iii), (iv) are true (D) (ii), (iii) are true

13. The language generated by grammar:
$S \to Te|Ue$, $T \to cTd|cT|\varepsilon$, $U \to cUd|Ud|dd$. is
(A) $L = \{c^n d^m e : m \geq n\}$
(B) $L = \{c^n d^m e : m = n\}$
(C) $L = \{c^m d^n e^m : m \geq n + 2\}$
(D) None of these

14. Remove null productions, useless symbols from the following grammar result in:
$S \to ABC$
$A \to aBC$
$B \to C|\varepsilon$
$C \to cd|DCF$
$D \to dD|\varepsilon$

$E \to eFE$
$F \to eC$

(A) $S \to ABC|AC$
$A \to aBC|aC$
$B \to C$
$C \to cd|DCF|CF$
$D \to dD|d$
$F \to eC$

(B) $S \to aBCc$
$A \to aBC$
$B \to cD|dDEF|dEF$
$C \to cD|dDEF|dEF$
$F \to eB$
$D \to dD|d$
$E \to eFE|e$

(C) $S \to aBCBC|aBC$
$B \to cD|dDEF|dEF$
$F \to eB$
$C \to dD|d$
$D \to e$
$F \to CD|dDEF$

(D) None of these

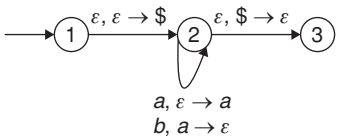
15. Let the language L_1, L_2 are defined as:
L_1: $\{a^i\, b^{2i}\, c^j |\ i, j \geq 0\}$, $L_2 = \{a^i\, b^{2i}\, a^i |\ i \geq 0\}$. Which of following is true?
(A) L_1, L_2 are context free
(B) Only L_1 is context free
(C) Only L_2 is context free
(D) Neither L_1 nor L_2 is context free

Practice Problems 2

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. Consider the alphabet $\Sigma = \{a, b, c, (,), \cup, ^*, \phi\}$. Then context free grammar that generates all strings in Σ^* that are regular expressions over $\{a, b\}$ is:
(A) $S \to S^*|a|b|SS$
(B) $S \to \phi|a|b|S$
(C) $S \to \phi|a \cup b|S^*$
(D) $S \to \phi|S^*|a|b|(S)|S \cup S|SS$

2. The PDA for language, L is designed below. The CFG generated contains ____ productions.



(A) 5 (B) 4
(C) 3 (D) 6

3. The language, L generated by the following grammar, $S \to SS|\ AAA|\varepsilon$, $A \to aA|\ Aa|\ b$ is
(A) $(a^*\, b^*)^*$ (B) $(a^*\, b^*\, b^*\, a^*)^*$
(C) $a^*\, b^*\, a^*$ (D) $(a^*\, b\, a^*\, b\, a^*\, b\, a^*)^*$

4. The grammar, G is defined with rules $S \to S_1|S_2$, $S_1 \to S_1 b|Ab|\varepsilon$, $A \to aAb|\ ab$, $S_2 \to S_2 a|\ Ba|\varepsilon$, $B \to bBa|\ ba$. The CNF is applied on G. The obtained grammar, G' contains ___ rules.
(A) 24 (B) 23
(C) 21 (D) 20

5. The language, $L = \{b^{n^2} :\ n \geq 1\}$ is:
(A) CFL but not DCFL
(B) DCFL but not CFL
(C) Only DCFL
(D) Not CFL

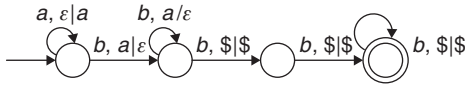
6. Consider the grammar, $G = S \to aSc|B$, $B \to bBc|\varepsilon$ The language, L generated by G is
(A) $L = \{a^n\, b^m\, c^k : k = n + m\}$
(B) $L = \{a^n\, b^m\, c^k : k \neq n + m\}$
(C) $L = \{a^n\, b^m\, c^k : k > n + m\}$
(D) $L = \{a^n\, b^m\, c^k : k < n + m\}$

7. The grammar, G is defined with productions:
$S \to 0A|1B$, $A \to 0AA|1S|1$, $B \to 1BB|0S|0$
The grammar, G_2 is defined with productions:
$S \to AB|aaB$, $A \to a|Aa$, $B \to b$
Which grammar is/are ambiguous?
(A) Only G_1
(B) Only G_2
(C) Both G_1 and G_2
(D) Both G_1 and G_2 are unambiguous

8. The language, $L_1 = \{0^n\, 1^n |\ n > 0\}$ and $L_2 = \{0^n\, 1^{2n} |\ n > 0\}$. The CFG generated for $L_1 \cup L_2$ is:
(A) $S \to 0\ A\ 1|0\ A\ 1\ 1$
$A \to 0|1|\varepsilon$
(B) $S \to 0\ A\ 1\ 1$
$A \to 0|1|\varepsilon$
(C) $S \to 0\ A\ 1|0\ B\ 1\ 1$
$A \to 0\ A\ 1|\varepsilon$
$B \to 0\ B\ 1\ 1|\varepsilon$
(D) $S \to 0\ A\ 1\ 1|0\ 1\ 1$
$A \to 0|1|\varepsilon$

9. The NPDA constructed to accept language, L with property, $L = L_1 \cup L_2$, where $L_1 = \{1^n\, 0^n |\ n > 0\}$, $L_2 = \{0^n\, 1^{2n} |\ n \geq 0\}$ contains ____ final states.
(A) 3 (B) 1
(C) 2 (D) 4

10. The DPDA for language, L is designed below. What is the language generated?



(A) $L = \{a^n b^m : m = n\}$
(B) $L = \{a^n b^m : m = n + 2\}$
(C) $L = \{a^n b^m : m \geq n + 2\}$
(D) $L = \{a^n b^m : m \leq n + 2\}$

11. The CFG, G is defined with rules:
$S \to AB|CD, A \to A00|\varepsilon, B \to B11|1, C \to C00|0, D \to D11|\varepsilon$. The language generated by G is
(A) $L = \{0^n 1^n | n \geq 0\}$
(B) $L = \{0\ 0^n\ 1\ 1^n | n > 0\}$
(C) $L = \{0^n 1^m | n + m \text{ is odd}\}$
(D) $L = \{0^n 1^m | n + m \text{ is even}\}$

12. The languages, L_1, L_2, L_3 are defined as:
$L_1 = \{a^n b^m c^{n+m} | n, m \geq 0\}$, $L_2 = \{a^n b^n c^m | n, m \geq 0\}$, $L_3 = \{a^n b^n c^{2n} | n \geq 0\}$, Which of the following statements are true?
(i) L_1, L_2 are context free
(ii) L_1, L_3 are context free
(iii) $L_3 = L_1 \cap L_2$
(iv) L_1, L_3 are context free but not L_2
(A) (i), (ii) (B) (i), (iii)
(C) (ii), (iii) (D) (iii), (iv)

13. The language, L_1 and L_2 are defined as $L_1 = \{a^n b^n : n \geq 0$ and n is not a multiple of 5$\}$ and $L_2 = \{0^n \# 0^{2n} \# 0^{3n} | n \geq 0\}$. Which of following is true?
(A) L_1 and L_2 are context free
(B) Only L_1 is context free
(C) Only L_2 is context free
(D) Neither L_1 nor L_2 is context free

14. The language, L_1 and L_2 are defined as $\overline{L_1} = \{0^n 1^n)^m | m, n > 0\}$, $L_2 = \{0^n\ 1^n\ 0^n\ 1^n | n \geq 0\}$ which of following is true?
(A) L_1 and L_2 are context free
(B) Only $\overline{L_1}$ is context free
(C) Only L_2 is context free
(D) Neither L_1 nor L_2 is context free

15. The language L_1, L_2 are defined as $L_1 = \{0^i\ 1^i\ 0^j\ 1^i | i, j > 0\}$, $L_2 = \{1^k\ 0^i\ 1^i\ 0^j\ 1^j\ 0^k | i, j, k > 0\}$. Which of following is true?
(A) L_1 and L_2 are context free
(B) Only L_1 is context free
(C) Only L_2 is context free
(D) Neither L_1 nor L_2 is context free

Previous Years' Questions

1. Match the following: **[2008]**

E. Checking that identifiers are declared before their use	P. $L = \{a^n b^m c^n d^m \| n \geq 1, m \geq 1\}$
F. Number of formal parameters in the declaration of a function agrees with the number of actual parameters in use of that function	Q. $X \to XbX \| XcX \| dXf \| g$
G. Arithmetic expressions with matched pairs of parentheses	R. $L = \{wcw \| w \in (a\|b)^*\}$
H. Palindromes	S. $X \to bXb \| cXc \| \varepsilon$

(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

2. Consider the languages L_1, L_2 and L_3 as given below.
$L_1 = \{0^p\ 1^q | p, q \in N\}$,
$L_2 = \{\ 0^p\ 1^q | p, q \in N$ and $p = q\}$ and
$L_3 = \{0^p\ 1^q\ 0^r | p, q, r \in N$ and $p = q = r\}$. Which of the following statements is NOT TRUE? **[2011]**
(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.

3. Which of the following problems are decidable? **[2012]**
(1) Does a given program ever produce an output?
(2) If L is a context free language, then, is $\overline{L}$ also context free?
(3) If L is a regular language, then, is $\overline{L}$ also regular?
(4) If L is recursive language, then, is $\overline{L}$ also recursive?
(A) 1, 2, 3, 4 (B) 1, 2
(C) 2, 3, 4 (D) 3, 4

4. Consider the following languages.
$L_1 = \{0^p\ 1^q\ 0^r | p, q, r \geq 0\}$
$L_2 = \{0^p\ 1^q\ 0^r | p, q, r \geq 0, p \neq r\}$
Which one of the following statements is **FALSE**? **[2013]**
(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

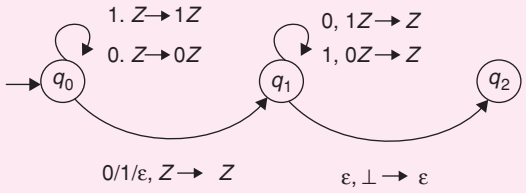
5. Which one of the following is **TRUE**? **[2014]**
(A) The language $L = \{a^n b^n | n \geq 0\}$ is regular
(B) The language $L = \{a^n | n$ is prime$\}$ is regular
(C) The language $L = \{w | w$ has $3k + 1b$'s for some k $\in N$ with $\Sigma = \{a, b\}\}$ is regular
(D) The language $L = \{ww | w \in \Sigma^*$ with $\Sigma = \{0, 1\}\}$ is regular.

6. Consider the following languages over the alphabet $\Sigma = \{0, 1, c\}$.
$L_1 = \{0^n\ 1^n | n \geq 0\}$
$L_2 = \{wcw^r | w \in \{0, 1\}^*\}$
$L_3 = \{ww^r | w \in \{0, 1\}^*\}$

Here w^r is reverse of the string w. Which of these languages are deterministic context-free languages? **[2014]**

(A) None of the languages
(B) Only L_1
(C) Only L_1 and L_2
(D) All the three languages

7. Consider the NPDA $<Q = \{q_0, q_1, q_2\}, \Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \perp\}, \delta, q_0, \perp, F = \{q_2\}>$, where (as per usual convention) Q is the set of states, Σ is the input alphabet, Γ is the stack alphabet, δ is the state transition function, q_0 is the initial state, $\perp$ 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 1011 00 so that the overall string is accepted by the automation? **[2015]**

(A) 10110 (B) 10010
(C) 01010 (D) 01001

8. Which of the following languages are context-free? **[2015]**

$L_1 = \{a^m b^n a^n b^m \mid m, n \geq 1\}$
$L_2 = \{a^m b^n a^m b^n \mid m, n \geq 1\}$
$L_3 = \{a^m b^n \mid m = 2n + 1\}$

(A) L_1 and L_2 only (B) L_1 and L_3 only
(C) L_2 and L_3 only (D) L_3 only

9. Consider the following context-free grammars:

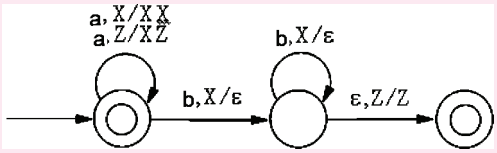
$G_1: S \to aS|B, B \to b|bB$

$G_2: S \to aA|bB, A \to aA|B|\varepsilon, B \to bB\varepsilon$

Which one of the following pairs of languages is generated by G_1 and G_2, respectively? **[2016]**

(A) $\{a^m b^n \mid m > 0$ or $n > 0\}$ and $\{a^m b^n \mid m > 0$ and $n > 0\}$
(B) $\{a^m b^n \mid m > 0$and $n > 0\}$ and $\{a^m b^n \mid m > 0$ or $n \geq 0\}$
(C) $\{a^m b^n \mid m \geq 0$ or $n > 0\}$ and $\{a^m b^n \mid m > 0$ and $n > 0\}$
(D) $\{a^m b^n \mid m \geq 0$ and $n > 0\}$ and $\{a^m b^n \mid m > 0$ or $n > 0\}$

10. Consider the transition diagram of a PDA given below with input alphabet $\sum = \{a, b\}$ and stack alphabet = $\{X,Z\}$. Z is the initial stack symbol. Let L denote the language accepted by the PDA.



Which one of the following is TRUE? **[2016]**

(A) $L = \{a^n b^n \mid n \geq 0\}$ and is not accepted by any finite automata.
(B) $L = \{a^n \mid n \geq 0) \cup \{a^n b^n \mid n \geq 0\}$ and is not accepted by any deterministic PDA.
(C) L is not accepted by any Turing machine that halts on every input.
(D) $L = \{a^n \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$ and is deterministic context-free.

11. Consider the following languages:

$L_1 = \{a^n b^m c^{n+m} : m, n \geq 1\}$
$L_2 = \{a^n b^n c^{2n} : n \geq 1\}$

Which one of the following is TRUE? **[2016]**

(A) Both L_1 and L_2 are context - free.
(B) L_1 is context - free while L_2 is not context - free
(C) L_2 is context - free while L_1 is not context - free.
(D) Neither L_1 nor L_2 is context - free.

12. Consider the following context-free grammar over the alphabet $\Sigma = \{a, b, c\}$ with S as the start symbol:
$S \to abScT \mid abcT$
$T \to bT \mid b$

Which one of the following represents the language generated by the above grammar? **[2017]**

(A) $\{(ab)^n (cb)^n \mid n \geq 1\}$
(B) $\{(ab)^n cb^{m_1} cb^{m_2} \ldots cb^{m_n} \mid n, m_1, m_2, \ldots, m_n \geq 1\}$
(C) $\{(ab)^n (cb^m)^n \mid m, n \geq 1\}$
(D) $\{(ab)^n (cb^n)^m \mid m, n \geq 1\}$

13. If G is a grammar with productions

$$S \to SaS \mid aSb \mid bSa \mid SS \mid \in$$

Where S is the start variable, then which one of the following strings is not generated by G? **[2017]**

(A) *abab* (B) *aaab*
(C) *abbaa* (D) *babba*

14. Consider the context-free rammers over the alphabet $\{a, b, c\}$ given below. S and T are non-terminals.

$$G_1 : S \to aSb|T, T \to cT|\in$$
$$G_2 : S \to bSa|T, T \to cT|\in$$

The language $L(G_1) \cap L(G_2)$ is **[2017]**

(A) Finite
(B) Not finite but regular
(C) Context-Free but not regular
(D) Recursive but not context-free.

15. Consider the following languages over the alphabet $\Sigma = \{a, b, c\}$.

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? **[2017]**

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

16. 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? **[2017]**

I. $L_1 \cup L_2$ is context-free.
II. $\bar{L}_1$ 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

17. Identify the language generated by the following grammar, where S is the start variable. **[2017]**

$$S \rightarrow XY$$
$$X \rightarrow aX|a$$
$$Y \rightarrow aYb|\in$$

(A) $\{a^m b^n \mid m \geq n, n > 0\}$ (B) $\{a^m b^n \mid m \geq n, n \geq 0\}$
(C) $\{a^m b^n \mid m > n, n \geq 0\}$ (D) $\{a^m b^n \mid m > n, n > 0\}$

18. Consider the following languages.

$L_t = \{a^p \mid p \text{ is a prime number}\}$
$L_2 = \{a^n b^m c^{2m} \mid n \geq 0, m \geq 0\}$
$L_3 = \{a^n b^n c^{2n} \mid n \geq 0\}$
$L_4 = \{a^n b^n \mid n \geq 1\}$

Winch of the following are CORRECT? **[2017]**

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

19. Consider the following languages:

I. $\{a^m b^n c^p d^q \mid m + p = n + q$, where $m, n, p, q \geq 0\}$
II. $\{a^m b^n c^p d^q \mid m = n$ and $p = q$, where $m, n, p, q \geq 0\}$
III. $\{a^m b^n c^p d^q \mid m = n = p$ and $p \neq q$, where $m, n, p, q \geq 0\}$
IV. $\{a^m b^n c^p d^q \mid mn = p + q$, where $m, n, p, q \geq 0\}$

Which of the languages above are context-free? **[2018]**

(A) I and IV only (B) I and II only
(C) II and III only (D) II and IV only

Answer Keys

Exercises

Practice Problems 1

1. D	**2.** C	**3.** D	**4.** B	**5.** D	**6.** A	**7.** C	**8.** B	**9.** D	**10.** C
11. B	**12.** D	**13.** D	**14.** A	**15.** B					

Practice Problems 2

1. D	**2.** C	**3.** D	**4.** A	**5.** D	**6.** A	**7.** C	**8.** C	**9.** C	**10.** C
11. C	**12.** B	**13.** B	**14.** B	**15.** C					

Previous Years' Questions

1. C	**2.** C	**3.** D	**4.** D	**5.** C	**6.** C	**7.** B	**8.** B	**9.** D	**10.** D
11. B	**12.** B	**13.** D	**14.** B	**15.** A	**16.** B	**17.** C	**18.** D	**19.** B	

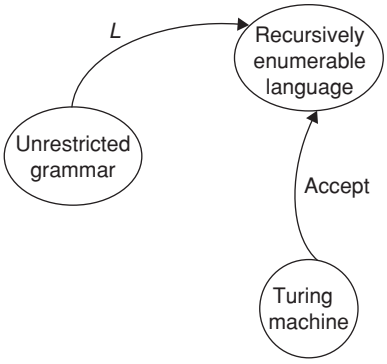
Chapter 3

Recursively Enumerable Sets and Turing Machines, Decidability

LEARNING OBJECTIVES

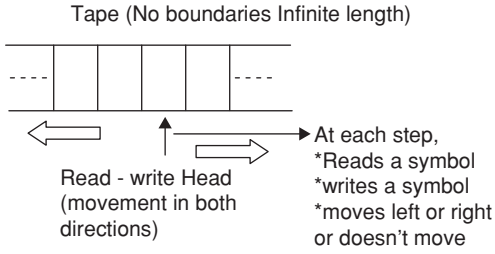
- *Turing machines*
- *Model of turing machine*
- *Types of turing machines*
- *Offline turing machine*
- *Universal turing machine*
- *Recursively enumerable languages*
- *Recursive language*
- *Undecidability*
- *Church's hypothesis*
- *Halting problem*
- *Post's correspondence problem*
- *7 P problems*
- *NP problems*
- *NP – complete problem*
- *NP – hard problem*
- *Closure properties of formal languages*

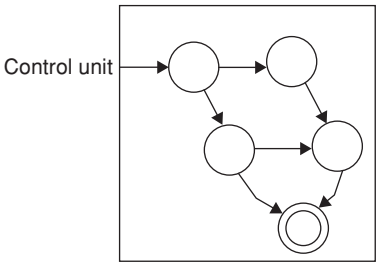
Turing Machines



A Turing machine is a kind of state machine, which is much more powerful in terms of languages it can recognize. At any time, the machine is in any one of the finite number of states. Instructions for a turing machine include the specification of conditions, under which the machine will make transitions from one state to other.

Model of Turing Machine

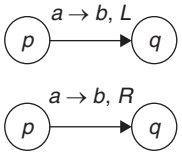




- A TM (turing machine) consists of Tape, Head, control unit.
- **Tape:** A tape is divided into a sequence of numbered cells, Each cell contains a symbol and cells that have not been written before are assumed to be filled with a blank symbol (B). The set of symbols of tape is denoted by Γ. The tape is assumed to be arbitrarily extensible to the left as well as to the right.
- **Head:** In a single step, a tape head reads the contents of a cell on the tape (reads a symbol), replaces it with some other characters (writes a symbol) and repositions itself to the next cell to the right or to the left of the one it has just read or does not move (moves left or right or does not move).
- **Control unit:** The reading from the tape or writing into the tape is determined by the control unit. It contains a finite set of states, Q. The states are:
  1. Initial state, q_0
  2. Halt state, h: This is state in which TM stops all further operations. There can be one or more halt states in a TM.
  3. Other states.

Note: A TM on entering the halt state stops making moves and whatever string is there on the tape, will be taken as the output, irrespective of whether the position of head is at the end or in the middle of the string on the tape.

Transition Diagram of TM



Reads a symbol
writes a symbol
Move Right (R)
Move Left (L)
No Move (N)

Specification of TM

5-Tuple specification:
TM = (state1, Read symbol, write symbol, $L/R/N$, state 2).
7 - Tuple specification of TM:
A TM, M is represented as a 7-tuple:
$M = (Q, \Sigma, \Gamma, \delta, q_0, B, h)$ where
$Q \rightarrow$ Finite set of states
$\Sigma \rightarrow$ Finite set of non-blank symbols
$\Gamma \rightarrow$ Set of tape characters
$q_o \rightarrow q_o \in Q$, initial state
$B \rightarrow$ Blank character
$h \rightarrow h \subseteq Q$, final state
$\delta \rightarrow$ Transition function, $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$

String classes in TM

Every TM, over the alphabet Σ, divides set of input string w into three classes:

1. **Accept (TM):** It is the set of all strings $w \in \Sigma^*$ ∋ if the tape initially contains w and the TM is then run, then TM ends in a halt state.
2. **LOOP (TM):** It is the set of all strings, $w \in \Sigma^*$ ∋ if the tape initially contains w and the TM is then run, then the TM loops forever (infinite loop).
3. **Reject (TM):** It is the set of all strings $w \in \Sigma^*$ ∋, any of the following 3-cases arise.

Case I: There may be a state and a symbol under the tape head, for which δ does not have a value.

Case II: If the head is reading the left most cell (i) containing the symbol x, the state of TM is say q, then δ (q, x) suggests a move to the left of the current cell. However as there is no cell to the left, no move is possible.

Case III: If TM enters an infinite loop or if a TM rejects a given string w, because of above two cases, TM crashes (terminates unsuccessfully).

Languages Accepted by a TM

- The language accepted by TM is the set of accepted strings $w \in \Sigma^*$.
- Formally, let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, h)$ be a TM. The language accepted by M denoted by $L(M)$ is defined as, $L(M) = \{w/w \in z^*$ and if $w = a_1 \ldots a_n$ then, $(q_o, \varepsilon, a_1, a_2, \ldots a_n)$ $(h, b, \ldots, b_{i-1}, b_j b_n)$ for some $b_1, b_2 \ldots b_n \in N^*$ ∋$\}$

  $L(M) = \{W: q_o w \vdash^* x_1 h x_2\}$
- There are three types of turing machine related languages:
  1. **Turing Acceptable language:** A language, L over some alphabet is said to be turing acceptable language if there exists a TM, M ∋ $L = L(M)$
  2. **Turing Decidable Language:** A language L over Σ i.e., $L \subseteq \Sigma^*$ is said to be turing decidable, if both languages, L and its complement $\Sigma^* - L$ are turing acceptable.
  3. **Recursively Enumerable Language:** A language L is recursively enumerable, if it is accepted by a TM.

Example 1: Let M be a turing machine has $M = (Q, \Gamma, \Sigma, \delta, S, B, F)$ with $Q = \{q_o, q_1, q_2, q_3, q_4\}$, $\Gamma = \{a, b, X, Y, \#\}$, $\Sigma = \{a, b\}$, $S = q_0$, $B = \#$, δ given by:

	a	*b*	*X*	*Y*	#
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	
q_1	(q_1, a, R)	(q_2, Y, L)	–	(q_1, Y, R)	
q_2	(q_2, a, L)	–	(q_0, X, R)	(q_2, Y, L)	
q_3	–	–	–	(q_3, Y, R)	$(q_4, \#, R)$
q_4	–	–	–	–	–

Which of following is true about M?
(A) M halts on L having 'baa' as substring
(B) M halts on L having 'bab' as substring
(C) M halts on $L = \{a^n b^n / n \geq 1\}$
(D) M halts on L not having 'bbaa' as substring.

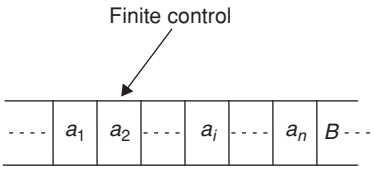
Solution: (C)
M accepts $a^n b^n$.

Example: *aaabbb*

$(q_0, \in, aaabbb) \rightarrow (q_1, XXXYY, b)$
$(q_1 X, aabbb) \rightarrow (q_2 XXXY, YY)$
$(q_1, Xa, abbb) \rightarrow {}^1(q_2, XXX, YYY)$
$(q_1, Xaa, bbb) \rightarrow (q_2, XY, XYYY)$
$(q_1, Xa, aYbb) \rightarrow (q_0, XXX, YYY)$
$(q_2, X, aaYbb) \rightarrow (q_3, XXXY, YY)$
$(q_2, \in, XaaYbb) \rightarrow (q_3, XXXYY, Y)$
$(q_o, X, aaYbb) \rightarrow (q_3, XXXYYY, \in)$
$(q_1, XX, aYbb) \rightarrow (q_4, XXXYYY\#, \in)$

Types of Turing Machines

Two-way infinite turing machine

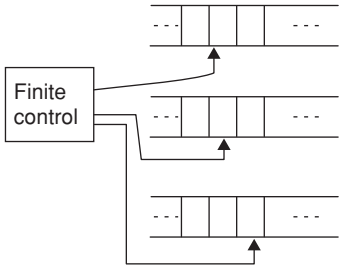


- A TM with a two-way infinite tape is denoted by $M = (Q, \Sigma, \Gamma, \delta, q_o B, F)$, as in original model.
- The tape is infinite to the left as well as to the right.

If $\delta(q, x) = (p, Y, L)$ then $q\ x\ \alpha \vdash_m pBY$. The tape, is infinte towards left.

If $\delta(q, x) = (p, B, R)$ then $q\ x\ \alpha \vdash_m p\alpha$ the is infinite towards right.

Multiple turing machines



- A multiple TM consists of a finite control with k tape heads and k-tapes, each tape is infinite in both directions, on a single move, depending on the state of the finite control and the symbol scanned by each of tape heads, the machine can,
  - change state
  - print new symbol on each of the cells scanned by its tape head
  - move each of its tape heads, independently, one cell to the left or right or keep it stationary.
- Initially, the input appears on the first tape and other tapes are blank.

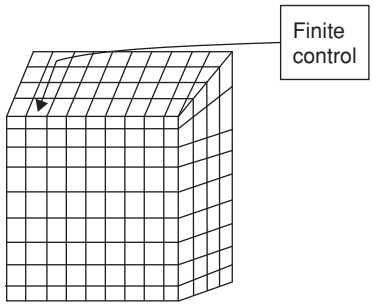
Non-deterministic turing machines

- A non-deterministic turing machine is a device with a finite control and a single one way infinite tape.
- For a given state and a tape symbol scanned by the tape head, the machine has a finite number of choices for next move.

Note: Non-deterministic TM is not permitted to make a move in which the next state is selected from one choice, and the symbol printed and direction of head motion are selected from other choices.

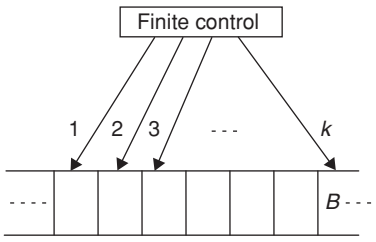
- The non-deterministic TM accepts its input if any sequence of choices of moves leads to an accepting state.

Multi-dimensional TM's



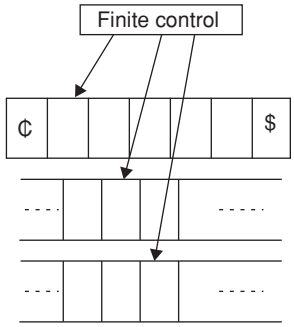
- The tape consists of a k-dimensional array of cells infinite in all 2k directions, for some fixed k.
- Depending on the state and the symbol scanned, the device changes it's state, prints a new symbol and moves its tape head in one of the $2k$ directions, either positively or negatively, along one of the k-axes.

Multihead TM



- A K-head TM has some fixed 'K' number of heads. The heads are numbered from 1 through k, and a move of the TM depends on the state and on the symbol scanned by each head.

Offline turing machine



- An offline TM is a multi tape TM, whose input tape is read only. The input is surrounded by end markers, ¢ on left and $ on right. The TM is not allowed to move the input tape head off the region between ¢ and $.

Multi stack machine

- A deterministic two stack machine is a deterministic TM with a read only input and two storage tapes.

Note:

- All these types of TM's does not add any language accepting power and all these are equivalent to the basic model.
- Any language accepted by a 2-PDA can be accepted by some TM and any language accepted by a TM can be accepted by some 2-PDA. Accepting power of a TM = accepting power of a computer.
- Any language accepted by a PDA with n stacks ($n \geq 2$), can also be accepted by some TM.

Example 2: Consider the following statement about L:

1. L is accepted by multi-tape turing machine M_1.
2. L is also accepted by single tape turing machine M_2.

Which of following statement is correct?

(A) Acceptance by M_2 is slower by $O(n^2)$
(B) Acceptance of M_2 is slower by $O(n)$
(C) Acceptance of M_2 is faster by $O(n)$
(D) Acceptance of M_2 is faster by $O(n^2)$

Solution: (A)
While simulating multi-tape TM on a single tape TM the head has to move at least $2k$ cells per move, where k is the number of tracks on single tape TM. Thus for k moves, $\sum_{i=1}^{k} 2i = 2k^2$.
Which means quadratic slow down?
Thus, acceptance of multi-tape is faster by $O(n^2)$.

Universal turing machine

A Universal turing machine is a turing machine that can simulate an arbitrary turing machine on arbitrary input.

- The machine consists of an input output relation to the machine computes.
- The input is given in binary form on the machine tape and the output consists of the contents of the tape when the machine halts.
- The contents of the tape will change based on the Finite State Machine (FSM) inside the TM.
- The problem with TM is that a different machine will be constructed for every new computation to be performed.
- A UTM can simulate any other machine.

Combining turing machines

If TM_1 and TM_2 are turing machines, then we can combine these machines and create a Turing machine which will first behave like TM_1 and TM_2.

To combine two turing machines follow below steps:

1. Change all states in TM_2, so that they do not conflict with the state names in TM_1.
2. Change all halts in TM_1's transition table to the new name of the start state of TM_2.
3. Append TM_2's transition table to the foot of TM_1's transition table.

- If TM_1 and TM_2 are combined in this way, we will write it as $TM_1 \rightarrow TM_2$.

So this new machine starts off in the initial state of TM_1, operates as per TM_1 until TM_1 would halt then it launches TM_2 and operates a TM_2, until TM_2 would halt.

Recursively Enumerable Languages

- A language L over the alphabet Σ is called 'recursively enumerable' if there is a TM, M that accept every word in L and either rejects or loops for every word in language L', the complement of L.
  Accept $(M) = L$
  Reject (M) + Loop $(M) = L'$.
- When TM, M is still running on some input of recursively enumerable languages, it is not decided that M will eventually accept, if let it run for long time or M will run forever (in loop).

Recursive language

- A language is said to be recursive, if there exists a TM which will halt and accept when presented with any input string $w \in \Sigma^*$, only if the string is in the language otherwise will halt and reject the string.
- Thus, for turing decidable language L, there is a TM which halts for a large number of inputs w belonging to L.
- A TM that always halts is known as a decider or a total turing machine and is said to decide the recursive language. The recursive language is also called as recursive set of decidable.
- A language accepted by a TM is said to be recursively enumerable language. The subclass of recursively enumerable sets are said to be recursive sets or recursive language.

Note:

- All recursive languages are also recursively enumerable.
- There may be languages which are recursively enumerable but not recursive.
- Set of all possible words over the alphabet of the recursive language is a recursive set.

- Set of all possible words, over the alphabet of the recursive enumerable language, is a recursively enumerable set.

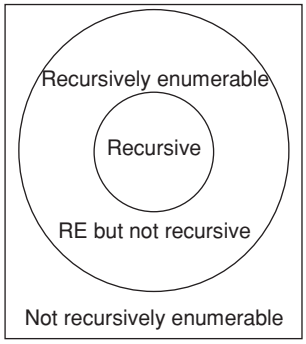


Figure 1 Relationship between the recursive, RE and non-RE languages.

Properties of Recursive and Recursively Enumerable Languages

- If a language L is recursive, then there is a TM T that accepts it and always halts.
- If L and L_1 are both recursively enumerable, then L and L_1 are recursive.
- Union of two recursive languages is recursive.
- Recursively enumerable languages are closed under union.
- If L, L_1 and L_2 are recursive languages, then so are $L_1 \cup L_2$, $L \cap L_2$, L_1L_2, L^*, $L_1 \cap L_2$ and $L_1 - L_2$.
- If L, L_1 and L_2 are recursively enumerable languages, then so are $L_1 \cup L_2$, L^*, $L_1 \cap L_2$, L_1L_2.
- If Σ is an alphabet, $L \subseteq \Sigma^*$, is a recursively enumerable language and $\Sigma^* - L$ is recursively enumerable, then L is recursive.

Example 3: If $\Sigma = \{0,1\}$, the canonical order is $\{\in,0,1,00,01,10,11,000,\ldots\}$ where w is the i^{th} word and M_j is TM whose code is the integer j, written in binary. The language generated is $L(M_j)$. The diagonalized language, L_d is a.

(A) Recursively enumerable language but not recursive
(B) Recursive language
(C) Non-recursively enumerable language
(D) Both (a) and (c)

Solution: (C)
Non-recursively enumerable language.

Non-recursively enumerable language

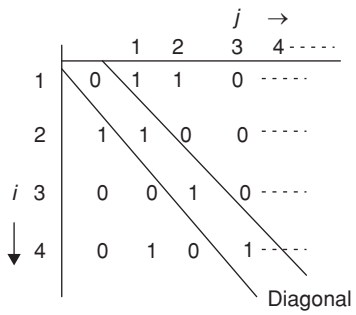
Non-Recursively Enumerable Language: A language which is not accepted by any turing machine is non-recursively enumerable.

Example: Power set of an infinite set.

- These languages cannot be defined by any effective procedure.

For any non-empty Σ, there exist languages that are not Recursively Enumerable.

Infinite table for all i and j is:



To guarantee that no TM accepts L_d:

w_i is in L_d if and only if the (i, i) entry is 0, that is, if M_i does not accept w_i.

Suppose that some TM M_j accepted L_d. Then it contradicts if w_j is in L_d, (j, j) entry is 0, implying that w_j is not in $L(M_i)$ and contradicting $L_d = L(M_i)$.

If w_i is not in L_d, then the (j, j) entry is 1, implying that w_i is in $L(M_j)$, which again contradicts $L_d = L(M_j)$, as w_j is either in or not in L_d, assumption, $L_d = L(M_j)$ is false.

Thus no TM in the list accepts L_d, Hence L_d is non-recursively enumerable language.

Decidable: A problem with two answers (Yes/No) is decidable if the corresponding language is recursive.

Example:

1. $A_{DFA} = \{(M, w)\ M$ accepts the input string $w\}$.

- A Language L is turing decidable, if there exists a TM M such that on input x, M accepts if $x \in L$ and M rejects otherwise. L is called undecidable if it is not decidable.
- Decidable Languages correspond to algorithmically solvable Decision problems.
- Undecidable language corresponds to algorithmically unsolvable decision problems.

Closure properties of decidable languages

- Decidable Languages are closed under complement, union, intersection, concatenation and star (closure) operations.

Note 1: A language is decidable if both the language and its complement are recognizable.

Note 2: Turing Decidable languages are Recursive languages.

Undecidability

There are problems that can be computed. There are also problems that cannot be computed. These problems which cannot be computed are called 'computationally undecidable problems'.

Church's Hypothesis

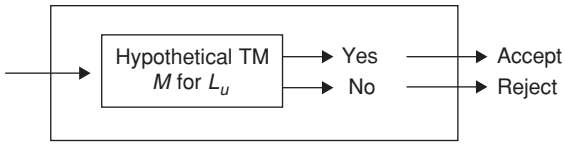
There is an assumption that the intuitive notion of computable functions can be identified with partial recursive functions.

However, this hypothesis cannot be proved. The computability of recursive function is based on following assumptions:

1. Each elementary function is computable.
2. Let 'f' be a computable function and 'g' be another function which can be obtained by applying an elementary operation to f, then g becomes a computable function.
3. Any function becomes computable, if it is obtained by rule (1) and (2).

Undecidability of the universal languages

- The universal language, L_u is a recursively enumerable language but not recursive.



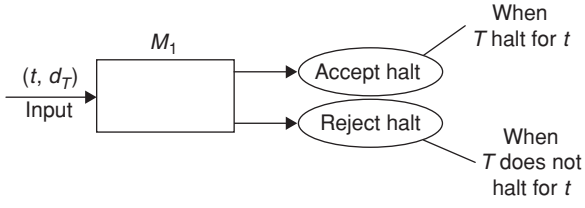
Halting Problem

The given configuration of TM is required to state halting problem. The output of TM can be:

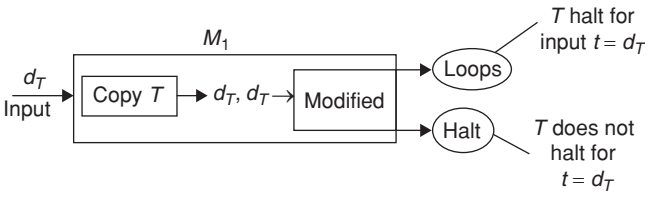
1. **Halt:** The machine starting at this configuration will halt after a finite number of states.
2. **No Halt:** The machine starting at this configuration never reaches a halt state, no matter how long it runs.

- The halting problem is unsolvable because, let, there exists a TM, M, which decides whether or not any computation by a TM, T will ever halt when a description d_T of T and tape t of T is given. That means the input to machine M, will be (machine, tape) pair. Then for every input (t, dT) to M_I if T halt for input t, M_I also halts which is called accept halt.

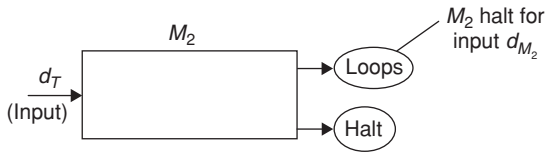
Similarly if T does not halt for input t then the M_1 will halt which is called reject halt.



- Consider another Turing Machine, M_2 which takes an input d_T. It first copies d_T on its tape and then this duplicated tape information is given as input to M_1. But M_1 is a modified machine.



Replace T by M_2 i.e., $M_2 = \mathrm{T}$



That's means, a machine M_1, which can tell whether any other TM will halt on particular input does not exist. Hence halting problem is unsolvable.

Post's Correspondence Problem (PCP)

The Undecidability of strings is determined with the help of Post's Correspondence Problem (PCP).

'The PCP consists of two lists of strings that are of equal length over the input Σ. The two lists are $A = w_1, w_2, w_3, \ldots w_n$ and $B = x_1, x_2, \ldots x_n$ then there exists a non-empty set of integers $i_1, i_2, \ldots i_n$ such that $w_1, w_2, \ldots w_n = x_1, x_2, \ldots x_n$'.

To solve PCP, try all the combinations of $i_1, i_2, \ldots i_n$ to find the $w_i = x_i$ then, PCP has a solution.

Example 4: What is the solution for the following system of post correspondence problem. $A = \{100, 0, 1\}$ $B = \{1, 100, 00\}$

(A) 1113322 (B) 1311322
(C) 2233111 (D) No solution

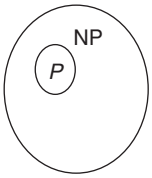
Solution: (B)
The string is:
A1A3A1A1A3A2A2 = 100 + 1 + 100 + 100 + 1 + 0 + 0 = 1001100100100,
B1B3B1B1B3B2B2 = 1 + 00 + 1 + 1 + 00 + 100 + 100 = 1001100100100.

Problems

- P stands for deterministic polynomial time. A deterministic machine at each time executes an instruction. Depending on instruction, it then goes to next state which is unique. Hence, time complexity of deterministic TM is the maximum number of moves made by M in processing an input string of length n, taken over all inputs of length n.
- A language, L is said to be in class P, if $\exists$ a (deterministic) TM, M is of time complexity $P(n)$ for some polynomial P and M accepts L.
- Class P consists of those problems that are solvable in polynomial time by a deterministic TM.

NP Problems

- NP stands for non-deterministic polynomial time.
- A language, L is in class NP, if there is a non-deterministic TM, M is of time complexity $P(n)$ for some polynomial P and M accepts L.
- Class NP consists of problems for which solutions are verified quickly. P consist of problems which can be solved quickly.



- NP languages are closed under union, Intersection, concatenation, Kleen star.
- NP problems are classified into two types:
  1. NP-complete
  2. NP-hard problems.

Example: Vertex (Graph) coloring problem, Travelling salesman problem, the vertex cover problem, the Hamiltonian circuit problem.

NP-Complete Problem

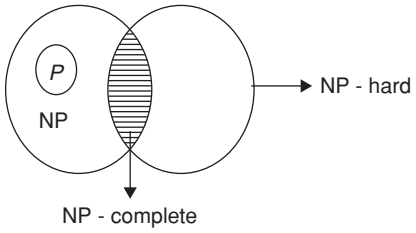
- A class of problems are known as NP-complete problems whose status is unknown. No polynomial time has yet been discovered for NP-complete problems nor has any one been able to prove that no polynomial time exists for any of them. These are hardest of NP-problems. The P and NP-complete problems are disjoint.

Example: (Cook's Theorem) SAT is NP-complete, Bin packing problem, Knapsack Problem.

- A language L is said to be NP-complete if $L \in NP$ and if every $L^I \in$ NP is polynomial-time reducible to L.

A language L_1 is said to be polynomial time reducible to some language L_2 if there exists a DTM by which any w_1 in the alphabet of L_1 can be transformed in polynomial time to a w_2 in the alphabet of L_2 in such a way that $w_1 \in L_1$ if $w_2 \in L_2$. It follows that if some L_1 is NP-complete and polynomial time reducible to L_2, then L_2 is also NP-complete.

NP-Hard Problem



- A problem that is NP-hard has a property that all problems that are in NP can be reduced in polynomial time to it.
- A language, L in NP-hard complete if and only if,

Condition 1: For every language, L^I in NP, there is a polynomial time reduction of L^I to L.

Condition 2: L is not necessarily in NP.

Table 1 *NP-Hard versus NP-complete problems:*

NP-Hard	NP-Complete
(1) A decision problem P_i is NP-hard if every problem in NP is polynomial time reducible to P_i.	(1) A Decision problem P_i is NP-complete if it is NP-hard and is also in class NP itself.
(2) In terms of symbols 'P_i' is NP-hard if for every $P_j \rightarrow$ NP	(2) In terms of symbols, 'P_i' is NP-complete, if P_i is NP-hard and $P_j \rightarrow$ NP
(3) P_i is 'as hard as' all the problem in NP	(3) P_i is one of the hardest problems in NP
(4) If any problem in NP is proved intractable, then P_i must also be intractable	(4) If any one ever shows that as NP-complete problem is also intractable, then every NP-complete problem is also intractable.

Example 5: Which of following is FALSE?

(A) $\{< x, y > \mid x$ and y are integers, gcd $(x, y) = 1\}$ is a NP class problem.
(B) CLIQUE is a NP class problem.
(C) Eulerian PATH is a P class problem
(D) Dijkstra's algorithm is a problem in P.

Solution: (A)
Choice (A) is a P class problem.
Consider the following table:

D – Decidable, U – Undecidable,? – Open question,T – Trivially Decidable Question	Regular Sets	DCFL's	CFL's	CSL's	Recursive Sets	Recursively Enumerable Sets
(1) Membership problem?	D	D	D	D	D	D
(2) Emptiness problem?	D	D	D	U	U	U
(3) Completeness problem is $L = \Sigma^*$?	D	D	D	U	U	U
(4) Equality problem?	D	?	U	U	U	U
(5) Subset problem is $L_1 \subseteq L_2$?	D	U	U	U	U	U
(6) Is L Regular?	T	D	U	U	U	U
(7) Is the intersection of two languages, a language, of the same type?	T	U	U	T	T	T
(8) Is the complement of a language, also a language of the same type?	T	T	U	?	T	U
(9) Is L is finite or infinite?	D	D	D	U	U	U

Table 2 *Closure properties of formal languages*

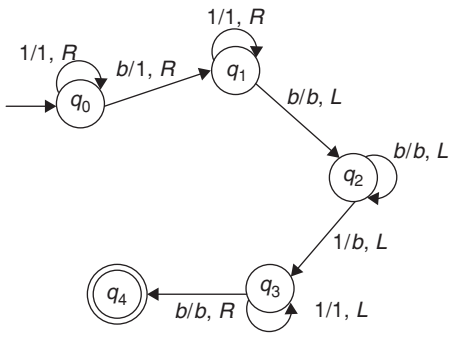
	Regular sets	DCFL'S	CFL'S	CSL'S	Recursive sets	Recursively enumerable sets
(1) Union	*Y*	*N*	*Y*	*Y*	*Y*	*Y*
(2) Concatenation	*Y*	*N*	*Y*	*Y*	*Y*	*Y*
(3) Kleen star	*Y*	*N*	*Y*	*Y*	*Y*	*Y*
(4) Intersection	*Y*	*N*	*N*	*Y*	*Y*	*Y*
(5) Complementation	*Y*	*Y*	*N*	*Y*	*Y*	*N*
(6) Homomorphism	*Y*	*N*	*Y*	*N*	*N*	*Y*
(7) Inverse Homomorphism	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*
(8) Reversal	*Y*	*N*	*Y*	*Y*	*Y*	*Y*
(9) Substitution	*Y*	*N*	*Y*	*Y*	*N*	*Y*
(10) Intersection with regular ets	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*

EXERCISES

Practice Problems I

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. The TM M over $\Sigma = \{1\}$ is given below



What does M generate?

(A) The output is total recursive multiplication function.
(B) The output is addition of two integers.
(C) The output is subtraction of two integers.
(D) The output should be w_1w_2 if input = (w_1w_2) a pair of words.

2. Consider language,

$A = \{<M>: M$ is a DFA which doesn't accept any string containing odd number 1's$\}$

Which of following is true about A?

(A) A is Trivially decidable (B) A is undecidable
(C) A is decidable (D) None of these

3. Consider $EQ_{CFG} = \{<G_1G_2>: G_1, G_2$ are CFGs and $L(G_1) = L(G_2)\}$. Which of following is true about EQ_{CFG}?

(A) Recognizable (B) Co-Recognizable
(C) Un-recognizable (D) None of the above.

4. A language is given as $INFINITE_{DFA} = \{<A>: A$ is a DFA and $L(A)$ is an infinite language$\}$. Which of following is true?

(A) Un-decidable (B) Decidable
(C) Trivially decidable (D) None of above.

5. A TM designed over an alphabet {0, 1, #}, where 0 indicates blank, which takes a non-null string of 1's and #'s and transfer's the right-most symbol to the left-most end contains-states. (Ex: 000#1#1#1000 … becomes 0001#1#1#000)

(A) 4 (B) 3
(C) 6 (D) 5.

6. Which of following statements are true?

(i) Let K, L be decidable languages. The concatenation of languages, K, L is also decidable language.

(ii) Let L be Turing recognizable language. Then the complement, L^1 is also Turing recognizable language.

(A) (i) and (ii) (B) Only (ii)
(C) Both are false (D) Only (i)

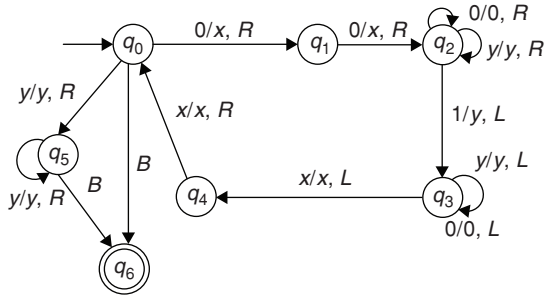
7. Let T_i: denote i th TM. Given, X determines whether $X \in S$, Where the set S is defined inductively as follows: If $u \in S$, then $u^2 + 1$, $3u + 2$ and $u!$ are all members of S. Which of following is true about the given decision problem?

(A) Decidable (B) Un-decidable
(C) Trivially decidable (D) No solution.

8. Fermat's last theorem asserts that there are no integer solution (x, y, z, n) to equation $x^n + y^n = z^n$ satisfying $x, y > 0$ and $n > 2$. Which of the following is true regarding the halting problem?

(A) Decidable
(C) Un-decidable
(C) Trivially decidable
(D) May or may not have solution.

9. The TM, T is designed as



Which of following is true?

(A) T halts on 0^n1^n, $n \geq 0$

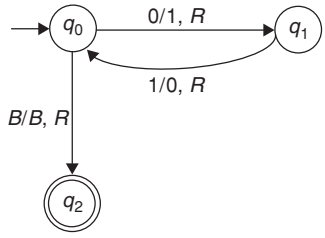
(B) T halts on (01) $(0^n 1^n)$, $n \geq 0$

(C) T halts on $0^{n^2}1^{n^2}$, $n \geq 0$

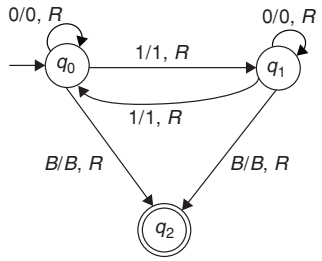
(D) T halts on $0^{2n}1^n$, $n \geq 0$

10. Design TM, which reads an input and starts inverting 0's to 1's till the first 1. The first 1 also inverted. After it has inverted first 1, it read the next symbols and keeps them as they are till the next 1. After encountering 1, it starts repeating the cycle by inverting the symbol till next 1. It halts when it encounters a blank symbol?

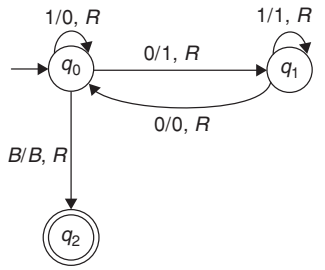
(A)



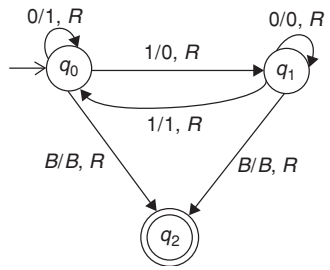
(B)



(C)



(D)



11. Consider three problems, P_1, P_2 and P_3. It is known that P_1 has polynomial time solution, P_2 is NP-complete and P_3 is in NP. Which one of the following is true?

(A) P_3 has polynomial time solution if P_1 is polynomial time reducible to P_3.

(B) P_3 is NP-complete if P_3 is polynomial time reducible to P_2.

(C) P_3 is NP complete if P_2 is reducible to P_3

(D) P_3 has polynomial time complexity and P_3 is reducible to P_2.

12. Let FHAM be the problem of finding a Hamiltonian cycle in a graph G and DHAM be the problem of determining if a Hamiltonian cycle exists in a graph. Which one of the following is true?

(A) Both FHAM and DHAM are NP-hard.

(B) FHAM is NP-hard, but DHAM is not.

(C) DHAM is NP-hard but FHAM is not.

(D) Neither DHAM nor FHAM is NP-hard.

13. The solution for the system of post correspondence problem, $A = \{ba, abb, bab\}$, $B = \{bab, bb, abb\}$ is

(A) 1312212 (B) 15234434

(C) 1311322 (D) No solution.

14. A language, prefix_free REX = $\{R/R$ is a regular expression where $L(R)$ is prefix_free$\}$. Which of following is true about prefix _free REX?

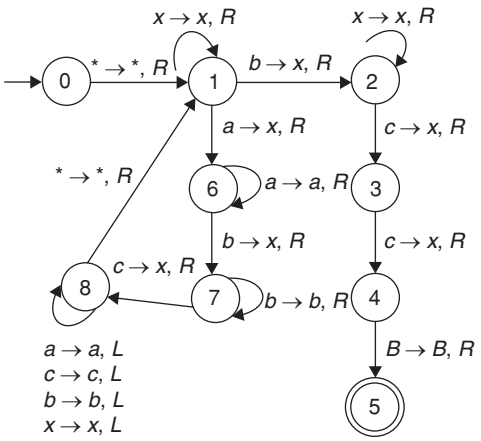
(A) Decidable

(B) Un-decidable

(C) Trivially decidable.

(D) Can't be determined.

15. The TM, M is designed as:



Which of following is true about M?

(A) M is designed for $a^n b^n c^n$, $n \geq 0$

(B) M is designed for $a^{n^2} b^{n^3} c^{n^4}$, $n \geq 0$

(C) M is designed for $a^n b^{n+1} c^{n+2}$, $n \geq 0$

(D) M is designed for $a^n b^n c^n$, $n > 0$

Practice Problems 2

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. Consider the language, $A\varepsilon_{\overline{CFG}} = \{<G>: G \text{ is a CFG that generates} \varepsilon\}$. Which of the following is true?
(A) Undecidable
(B) Decidable
(C) Trivially decidable.
(D) None of the above.

2. The TM is designed with input and output as binary form. (# represents blank). The turing machine TM (M) is

	0	1	#
$q0$	$(q1, 0, R)$	$(q, 1, R)$	ϕ
$q1$	$(q1, 0, R)$	$(q, 1, R)$	$(q2, \#, L)$
$q2$	$(q3, \#, L)$	$(q3, \#, L)$	ϕ
$q3$	$(q3, 0, L)$	$(q3, 1, L)$	$(q4, \#, L)$
$q4$	ϕ	ϕ	ϕ

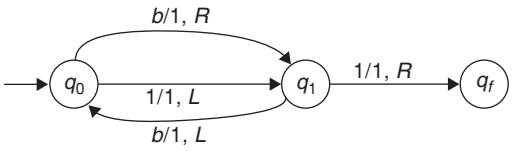
Which of following is true?
(A) M accepts $2n$
(B) M accepts n^2
(C) M replaces left most symbol with #
(D) M replaces right most symbol with #

3. The TM is designed with 3-characters 0, 1, # to compute function $f(n) = 2n$. Input and output are to be in binary form and string represented by 'n' is enclosed between two #'s on left and right of it. b is blank symbol. TM contains _____ states.
(A) 4 (B) 3
(C) 2 (D) 1

4. The language $\{1^n \mid n \text{ is a prime number}\}$ is
(A) Undecidable
(B) Decidable
(C) Trivially decidable
(D) None of the above

5. Which of following statement(s) are true?
(i) Let L be Turing decidable language. Then the complement $\overline{L}$ is also Turing decidable language.
(ii) Let K and L be two Turing recognizable languages. The intersection, $K \cap L$ is also Turing recognizable language.
(A) Both (i) and (ii)
(B) Only (i)
(C) Only (ii)
(D) Neither (i) nor (ii) are true.

6. For the following two-way infinite TM, the equivalent one-way TM contains _____ states.



(A) 7 (B) 6
(C) 5 (D) 4

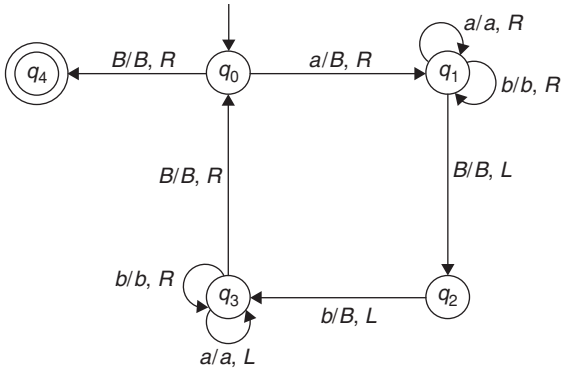
7. L contains at least two strings. Which of following is true?
(A) L has recursively enumerable sets and recursive.
(B) L is recursive.
(C) L has recursively enumerable sets but not recursive.
(D) L does not contain recursively enumerable sets and also is not recursive.

8. Consider the following TM:

Input / State	0	1	B
→ q_0	$(q0, 1, R)$	$(q0, 0, R)$	$(q1, B, R)$
$q1$	–	–	–

What does TM generates?
(A) It display's the negative of given binary number.
(B) It computes one's complement of a binary number.
(C) It computes two's complement of a binary number
(D) It generates double the 0's as 1's.

9. Consider the following TM, M:



Which of following is true?
(A) M halts on $a^{n+1}\, b^n$, $n \geq 0$.
(B) M halts on a^{n^2}, b^{n^3} $n \geq 0$.
(C) M halts on $(ab)\,(a^n)$, $n \geq 0$.
(D) M halts on $a^n\, b^n$, $n \geq 0$.

10. A TM, M is designed generates language $L = \{a^n b^m : n \geq 1 \text{ and } n \neq m\}$. The number of states used are _____
(A) 5 (B) 6
(C) 7 (D) 4

11. Consider three decision problems p_1, p_2 and p_3. It is known that p_1 is decidable, p_2 is undecidable. Which one of 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.

12. Which one of following is not decidable?
(A) Given a TM, M, a string S, and an integer K, M accepts S with in K-steps.
(B) Equivalence of two given Turing machines.
(C) Language accepted by a given DFSA is non-empty.
(D) Language accepted by a CFG is non-empty.

13. What is the solution for the correspondence system with two lists $x = \{b, bab^3, ba\}$ and $y = \{b^3, ba, a\}$
(A) 1312213 (B) 2113
(C) 3112 (D) No solution.

14. Given a Turing machine M, a state 'q'and a string 'w'. To determine whether M ever reaches state q when started with input w from its initial state is?
(A) Decidable
(B) Un-decidable
(C) Trivially decidable.
(D) Can not be determined.

15. Given a Turing machine, M to determine whether M ever moves its head to the left when started with input W is:
(A) Decidable
(B) Un-decidable
(C) Trivially decidable.
(D) Can not be determined.

Previous Years' Questions

1. For $s \in (0 + 1)^*$, let $d(s)$ denote the decimal value of s (e.g., $d(101) = 5$). **[2006]**

Let $L = \{s \in (0 + 1)^* | d(s) \bmod 5 = 2 \text{ and } d(s) \bmod 7 \neq 4\}$

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

2. Which of the following is true for the language $\{a^p \mid p$ is a prime$\}$? **[2008]**
(A) It is not accepted by a 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

3. If L and $\overline{L}$ are recursively enumerable then L is **[2008]**
(A) regular
(B) context-free
(C) context-sensitive
(D) recursive

4. Let $L = L_1 \cap L_2$, 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 **[2009]**
(A) Not recursive
(B) Regular
(C) Context free but not regular
(D) Recursively enumerable but not context free.

5. 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? **[2010]**
(A) $L_2 - L_1$ is recursively enumerable
(B) $L_1 - L_3$ is recursively enumerable
(C) $L_2 \cap L_1$ is recursively enumerable
(D) $L_2 \cup L_1$ is recursively enumerable

6. Which of the following statements is/are FALSE? **[2013]**
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 union and intersection.

(A) 1 and 4 only (B) 1 and 3 only
(C) 2 only (D) 3 only

7. Let L be a language and $\overline{L}$ be its complement. Which one of the following is NOT a viable possibility? **[2014]**
(A) Neither L nor $\overline{L}$ is recursively enumerable (*r. e*)
(B) One of L and $\overline{L}$ is r.e. but not recursive, the other is not r. e.
(C) Both L and $\overline{L}$ are r.e. but not recursive
(D) Both L and $\overline{L}$ are recursive

8. 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? **[2014]**
(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.

9. Let <M> be the encoding of a Turing machine as a string over $\Sigma = \{0, 1\}$. Let $L = \{<M>|M$ is a *Turing machine that accepts a string of length* 2014$\}$. Then, L is

(A) Decidable and recursively enumerable
(B) Undesirable but recursively enumerable
(C) Undesirable and not recursively enumerable
(D) Decidable but not recursively enumerable

10. For any two languages L_1 and L_2 such that L_1 is context-free and L_2 is recursively enumerable but not recursive, which of the following is/are necessarily true? **[2015]**

I. $\overline{L}_1$ (complement of L_1) is recursive
II. $\overline{L}_2$ (complement of L_2) is recursive
III. $\overline{L}_1$ is context-free
IV. $\overline{L}_1 \cup L_2$ is recursively enumerable

(A) I only (B) III only
(C) III and IV only (D) I and IV only

11. Consider the following statements.

I. The complement of every Turning 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? **[2015]**

(A) Only II (B) Only III
(C) Only I and II (D) Only I and III

12. 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 $\overline{y}$ reduces to W, and Z reduces to $\overline{x}$ (reduction means the standard many-one reduction). Which one of the following statements is **TRUE?** **[2016]**

(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.

13. Consider the following types of languages: L_1: Regular, L_2: Context - free, L_3: Recursive, L_4: Recursively enumerable. Which of the following is / are TRUE? **[2016]**

I. $\overline{L}_3 \cup L_4$ is recursively enumerable
II. $\overline{L}_2 \cup L_3$ is recursive
III. $L^*_1 \cap L_2$ is context - free
IV. $L_1 \cup \overline{L}_2$ is context - free

(A) I only (B) I and III only
(C) I and IV only (D) I, II and III only

14. Consider the following languages. **[2016]**

$L_1 = \{<M> \mid M$ takes at least 2016 steps on some input$\}$,

$L_2 = \{<M> \mid M$ takes at least 2016 steps on all inputs$\}$ and

$L_1 = \{<M> \mid M$ accepts $\varepsilon\}$

where for each Turing machine M, <M> 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 L_3 is not recursive
(D) L_1, L_2, L_3 are recursive

15. Let A and B be 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 turning machine M which given an input x in A^*, always halts with $f(x)$ on its tape. Let L_f denote the language $\{x \# f(x) \mid x \in A^*\}$. Which of the following statements is true: **[2017]**

(A) f is computable if and 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.

16. 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 Turing machine M. Which of the following decision problems are undecidable? **[2017]**

I. Given a regular expression R and a string w, is $w \in L(R)$?
II. Given a context-free grammar G, is $L(G) = \emptyset$?
III. Given a context-free grammar G, is $L(G) = \Sigma^*$ for some alphabet Σ ?
IV. Given a Turing 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 only (D) III and IV only

17. The set of all recursively enumerable languages is: **[2018]**

(A) Closed under complementation.
(B) Closed under intersection.
(C) A subset of the set of all recursive languages.
(D) An uncountable set.

18. Consider the following problems. $L(G)$ denotes the language generated by a grammar G. $L(M)$ denotes the language accepted by a machine M.

(I) For an unrestricted grammar G and a string w, whether $w \in L(G)$

(II) Given a Turing machine M, whether $L(M)$ is regular

(III) Given two grammars G_1 and G_2, whether $L(G_1) = L(G_2)$

(IV) Given and NFA N, whether there is a deterministic PDA P such that N and P accept the same language.

Which one of the following statements is correct? **[2018]**

(A) Only I and II are undecidable
(B) Only III is undecidable
(C) Only II and IV are undecidable
(D) Only I, II and III are undecidable

Answer Keys

Exercises

Practice Problems 1

1. D	**2.** C	**3.** B	**4.** B	**5.** D	**6.** D	**7.** A	**8.** D	**9.** D	**10.** D
11. C	**12.** A	**13.** D	**14.** A	**15.** C					

Practice Problems 2

1. B	**2.** D	**3.** B	**4.** B	**5.** A	**6.** B	**7.** C	**8.** B	**9.** D	**10.** B
11. C	**12.** B	**13.** B	**14.** B	**15.** A					

Previous Years' Questions

1. D	**2.** D	**3.** D	**4.** C	**5.** B	**6.** C	**7.** C	**8.** D	**9.** B	**10.** D
11. D	**12.** C	**13.** D	**14.** C	**15.** A	**16.** D	**17.** B	**18.** D		

TEST

THEORY OF COMPUTATION

Time: 60 min.

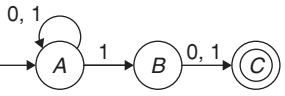
Directions for questions 1 to 30: Select the correct alternative from the given choices.

1. Phrase structure languages, context-sensitive languages, context-free languages and regular languages are commonly referred to as languages of type-0, 1, 2, and 3 respectively. Then, Chomsky's Hierarchy states that
(A) type-0 $\supseteq$ type-1 $\supseteq$ type-2 $\supseteq$ type-3
(B) type-0 $\supset$ type-1 $\supset$ type-2 $\supset$ type-3
(C) type-0 $\subset$ type-1 $\subset$ type-2 $\subset$ type-3
(D) type-0 $\subseteq$ type-1 $\subseteq$ type-2 $\subseteq$ type-3

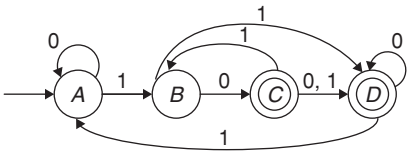
2. Let L be a language recognizable by a finite automaton. The language Reverse $(L) = \{x$ such that x is the reverse of y where $y \in L\}$ is a
(A) Regular language
(B) Context-sensitive language
(C) Context-free language
(D) Phrase-structure language

3. Which of the following statement is true?
(A) It is possible to construct an NFA with more number of states than its equivalent minimum DFA.
(B) There can be a DFA with more than one start state.
(C) Both (A) and (B)
(D) None of these

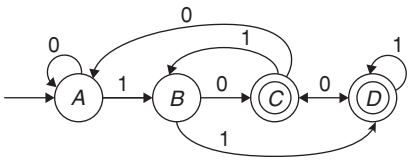
4. Which of the following is an equivalent DFA for the NFA shown below:



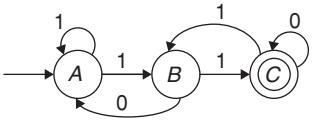
(A)



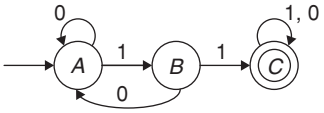
(B)



(C)

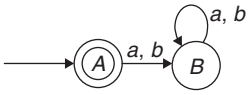


(D)



5. Which one of the following regular expressions over $\{0, 1\}$ denotes the set of strings not containing 100 as a substring?
(A) 0*(1*0)* (B) 0*1*01*
(C) (0*(10 + 1)*)* (D) 0*1010*

6. The following transition diagram of a finite automaton accepts



(A) All word over sigma (a, b) such that symbol a and b alternate.
(B) Only empty string.
(C) Only the λ_1 meaning this automaton accepts no string of length greater than zero.
(D) All words over sigma (a, b) except λ.

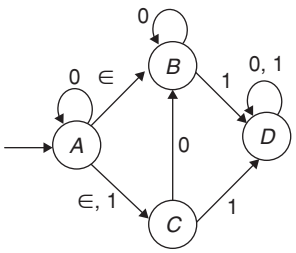
7. Sentence that can be generated from the following production grammar is
$S \rightarrow aS/bA$
$A \rightarrow d/ccA$
(A) *aabccccd* (B) *ababccccd*
(C) *bccddd* (D) *aaccdb*

8. Pumping lemma is generally used for proving
(A) A given grammar is regular
(B) A given grammar is non-regular
(C) Whether two given regular expression are equivalent or not
(D) Both (A) and (C)

9. Finite state machine can recognize
(A) Only context-free grammar
(B) Only regular grammar
(C) Any unambiguous grammar
(D) Any grammar

10. Which of the following is false?
(A) Regular sets are closed under reversal.
(B) Regular sets are closed under substitution.
(C) Regular sets are closed under intersection.
(D) None of these

11. For the DFA shown below $\hat{\delta}(A, 01)$ will be



(A) $\{B, D\}$ (B) $\{C, D\}$
(C) $\{A, B, C\}$ (D) $\{A, B, C, D\}$

12. 'NFA can be simulated by a DFA'. The statement is
(A) True (B) False
(C) Depends on NFA (D) Depends on DFA

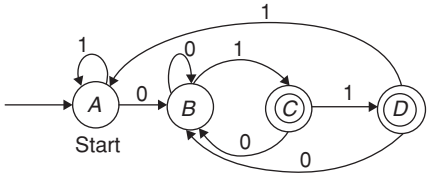
13. Given an arbitrary non-deterministic finite automaton (NFA) with N states, the maximum number of states in an equivalent minimized DFA is at least
(A) N^2 (B) 2^N
(C) $2N$ (D) $N!$

14. Let $M = (K, \Sigma, \delta, S, F)$ be a finite state automaton, Where
$K = \{A, B\}$
$\Sigma = \{a, b\}$
$S = A$
$F = \{B\}$,
$\delta(A, a) = A$
$\delta(A, b) = B$
$\delta(B, a) = B$ and
$\delta(B, b) = A$.

A grammar to generate the language accepted by M can be specified as G = (V, Σ, R, S), where

$V = K \cup \Sigma$, and $S = A$. Which one of the following set of rules will make $L(G) = L(M)$?
(A) $\{A \to aB, A \to bA, B \to bA, B \to aA, B \to \in\}$
(B) $\{A \to aA, A \to bB, B \to aB, B \to bA, B \to \in\}$
(C) $\{A \to bB, A \to aB, B \to aA, B \to bA, B \to \in\}$
(D) $\{A \to aA, A \to bA, B \to aB, B \to bA, A \to \in\}$

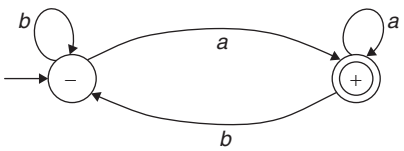
15 A deterministic finite automaton M shown below has a start state A and accepting state D. Which of the following regular expression denotes the set of all words accepted by M?



(A) 0 0 1 (B) 1 0* 1* 10
(C) 1* 0* 0 0 1 (D) (0/1)* 0 1 1

16. Which of the following regular expression is/are true?
(A) $(x^*)^* = x^*$ (B) $(x + y)^* = x^* + y^*$
(C) $x^*y^* = x^* + y^*$ (D) All of these

17. Consider the FA shown in the figure given below, where '–' is the start sate and '+' is the ending state. The language accepted by the FA is



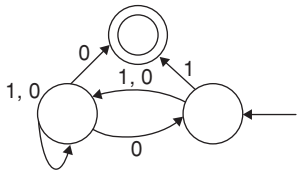
(A) $(a + b)^*b$ (B) $(a + b)^*a$
(C) a^*b (D) a^*b^*

18. Which of the following statement is false?
(A) The family of regular language is closed under the complementary operation.
(B) If L is a regular language, $L_1 = \{UV: U \in L, |V| = 2\}$ is also regular.
(C) If L is a regular language, $L_1 = \{UV: U \in L, V \in L^R\}$ is also regular.
(D) None of these

19. Which of the following is false?
(A) $L = \{0^i\ 1^m 2^m: i \geq 1, m \geq 1\}$ over $\Sigma = \{0, 1, 2\}$ is regular.
(B) $L = \{a^n b^l a^k, k \geq n + I\}$ is not regular.
(C) $L = \{UWW^2V: U, V, W \in \{a, b\}^+\}$ is regular.
(D) $L = \{a_n b_k : n > k\} \cup \{a_n b_k : n \neq k - 1\}$ is not regular.

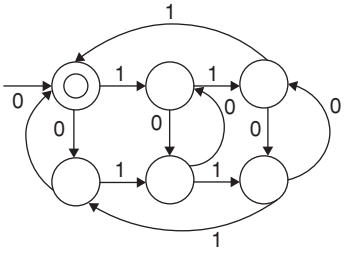
20. Consider a DFA over $\Sigma = \{a, b\}$ accepting all strings which have number of a's divisible by 6 and number of b's divisible by 8. What is the minimum number of states that the DFA will have?
(A) 16 (B) 15
(C) 48 (D) 8

21. For the NFA M given 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 statement is true?



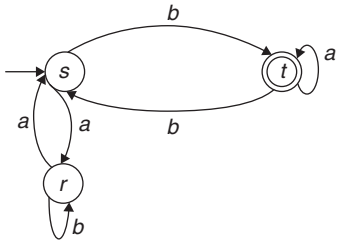
(A) $L_1 = A$ (B) $L_1 \subseteq L$
(C) $L_1 = \{0, 1\}^*$ (D) $L_1 = (0, 1)^* - L$

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



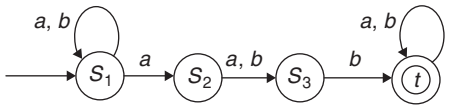
(A) Divisible by 3 and 2
(B) Odd and even
(C) Even and odd
(D) Divisible by 2 and 5

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



Consider the strings
$U = a\ b\ b\ a\ b\ a$
$V = b\ a\ b$ and
$W = a\ a\ b\ b$
Which of the following statement 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 U, V and W.
(D) The automaton accepts U but rejects V and W.

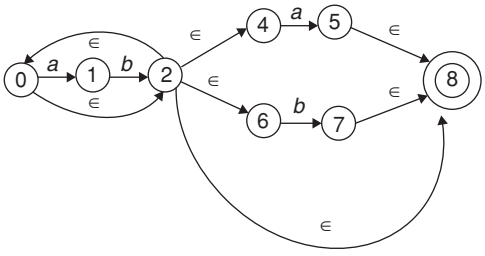
24. Which regular expression best describe the language accepted by the non-deterministic automaton below?



(A) $(a + b)^*a(a + b)b$
(B) $(a + b)^*a(a + b)b(a + b)^*$
(C) $(abb)^*$
(D) $(a + b)^*$

25. Which of the following strings are accepted by the regular expression:(0/1)* 0(0/1) (0/1)
(A) 000 or 001 (B) 001 or 010
(C) 010 or 011 (D) All the above

26.

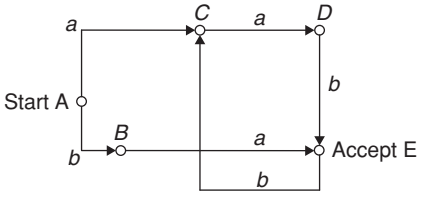


The above diagram represents NFA of regular expression.
(A) $(ab)^*\ (a/b/\in)$. (B) $(ab)^*\ (a/b)$.
(C) $(ab)^*\ (a/\in)$. (D) $(ab)^*\ (b/\in)$.

27. If 'a' is a terminal and S, A, B are three (3) non-terminals, then which of the following is regular grammar.
(A) $A \rightarrow a\ B/a\ A$ (B) $A \rightarrow B\ a/B\ a\ a$
(C) $A \rightarrow a\ B$, $B \rightarrow bA$ (D) $S \rightarrow \in$, $A \rightarrow a\ S/b$

28. Consider the grammar
$S \rightarrow ABc/Abc$
$BA \rightarrow AB$
$Ab \rightarrow ab$
$Aa \rightarrow aa$
Which of he following sentences can be derived by this grammar?
(A) aab (B) $abcc$
(C) $abab$ (D) abc

29.



The language recognized by the following finite automation is
(A) $aabb^* + bab^*$
(B) $(aab\ (bab^*))^*$
(C) $(aab + ba)\ (bab)^*$
(D) $(aab^* + bab^*)^*$.

30. From the following regular expressions over an alphabet $\{a, b\}$ given below, which can yield all the possible strings over $\sum (a, b)$?
(i) (a^*b^*)
(ii) $(a + b)^*$
(A) Only (i) (B) Only (ii)
(C) Both (A) and (B) (D) None of these

Answers Keys

1. B	**2.** A	**3.** A	**4.** B	**5.** B	**6.** B	**7.** A	**8.** B	**9.** B	**10.** D
11. B	**12.** A	**13.** B	**14.** B	**15.** D	**16.** A	**17.** B	**18.** D	**19.** A	**20.** B
21. C	**22.** A	**23.** D	**24.** B	**25.** D	**26.** A	**27.** D	**28.** D	**29.** C	**30.** B