UNIT 1

INTRODUCTION TO FINITE AUTOMATA

PART-A

SHORT QUESTIONS WITH SOLUTIONS

Q1. Write a short note on power of an alphabet.

Answer :

Model Paper-I, Q1(a)

Power of an Alphabet

Let Σ denotes an alphabet, the set of strings of any length formed using these alphabet can be expressed by an exponential notation Σ^* which is called as the power of alphabet.

Where n represents length of string over the alphabet Σ.

Example

Let $\Sigma = \{a, b\}$ then,

$\Sigma^1 = \{a, b\}$

$\Sigma^2 = \{aa, ab, ba, bb\}$

$\Sigma^3 = \{aaa, bbb, aba, abb, aab, baa, bab, bba\}$

Q2. What is a string? Write about concatenation of two strings.

Answer :

String

A finite sequence of symbols drawn from an alphabet is known as string 'S'. For example, if the alphabet is $\Sigma = \{0, 1\}$ then string S drawn from this alphabet is 001100.

Concatenation of Two Strings

It can be defined as the process of combining two given strings adjacently. For instance, if a and b are two strings then the concatenation of a and b is ab.

Suppose, $a = x_1x_2 \ldots x_i$ is a string with 'i' number of symbols and $b = y_1y_2 \ldots y_j$ is another string with 'j' number of symbols, then concatenation of a and b is,

$ab = x_1x_2 \ldots x_iy_1y_2 \ldots y_j$

The length of this string will be equal to $i + j$.

Example

Consider,

$a = 10011$

$b = 01100$

$ab = 1001101100$

Length of $ab = 10$.

Q3. Define transition table.

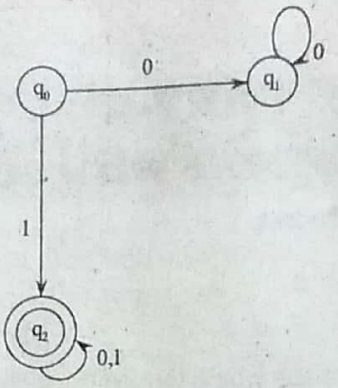
Answer :

Transition Table

It is the tabular representation of a given finite automata. It consists of rows and columns wherein rows correspond to the input state and the column corresponds to the input symbol of given input state. The transition table is used to define a transition function δ.

Example

Consider the finite automata,



The transition table for the given automata is,

Input States	Input Symbol	
	0	1
$\rightarrow q_0$	q_1	q_2
q_1	0	–
(q_2)	q_2	q_2

Table: Transition Table

Q4. What is the language of DFA?

Answer :

Model Paper-III, Q1(a)

Let L be a language and let DFA be $M = \{Q, \Sigma, \delta, q_0, F\}$. The languages accepted by DFA denoted by $L(M)$ is given by,

$L(M) = \{\omega \mid \hat{\delta}(q_0, \omega)$ reaches the final state $M\}$

A DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ accepts a set of language (L) if a string (W) starts from initial state and reaches the final state/ accepting state at the end.

Q5. What are the advantages of NFA over DFA?

Answer :

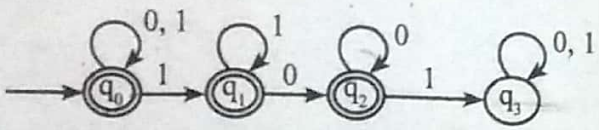
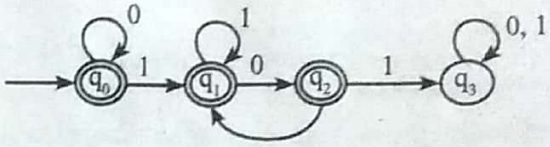
Model Paper-III, Q1(b)

The following are the advantages of NFA over DFA,

1. It is possible to define more than one transitions from each state on the same input symbol.
2. It accepts the input symbol ϵ i.e., empty input symbol.
3. It can be smaller than DFA.
4. It accepts a string if there is some sequence of possible moves that will put the machine in a final state at the string. It rejects string only if there is no possible sequence of moves by which a final state can be reached.

Q6. What are the differences between DFA and NFA with examples?

Answer :

	NFA		DFA
1.	It is possible to define more than one transition from each state on the same input symbol.	1.	It consist of only one transition from each state on any input symbol.
2.	It accept the input symbol $\in$ i.e., empty input symbol.	2.	It does not accept empty input symbol.
3.	It takes more time to recognize input string.	3.	It is faster than NFA.
4.	It can be smaller than DFA.	4.	It is much bigger than NFA.
5.	It accepts a string if there is some sequence of possible moves that will put the machine in a final state at the string. It rejects string only if there is no possible sequence of moves by which a final state can be reached.	5.	It accepts a string if the automaton is in one of its final states when the end of the string is reached. Otherwise it rejects the string.
6.	**Example** The NFA for the language of strings that does not contain 101 as substring is given as,  **Figure: NFA**	6.	**Example** The DFA for the language of strings that does not contain 101 as substring is given as,  **Figure: DFA**

Q7. Define δ in NFA with $\in$ (Epsilon) moves.

Answer :

(Model Paper-II, Q1(a) | May-15(R13), Q1(a))

A Non-deterministic Finite Automata (NFA) with $\in$-moves is defined by 5 tuples,

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

In NFA with $\in$ moves, δ is referred as a transition function that defines the rules for change of a state. It maps from $Q \times (\Sigma \cup \in)$ to 2^Q.

Where,

$Q \rightarrow$ Finite set of states i.e., q_0, q_1, q_2

$\Sigma \rightarrow$ Finite set of input symbols.

Q8. Discuss in brief about language of $\in$-NFA.

Answer :

Model Paper-I, Q1(b)

Languages Accepted by Finite Automata with $\in$-transitions

An NFA with $\in$ transitions defined by $M = \{Q, \Sigma, \delta, q_0, F\}$ accepts set of strings only if it reaches final state (q_F). This means that any string that starts from initial state (q_0) and ends at final state (q_F) is said to be acceptable by finite automata.

Let L be a language and W be the string of language then the acceptance of language by NFA is given by,

$L(M) = \{W / W \in \Sigma^*$ and transition (δ) on W starts from q_0 and ends at $q_F\}$.

Example

Consider the below NFA with $\in$-transitions that accepts a language consisting the string of any number of 0's followed by any number of 1's followed by any number of 2's.

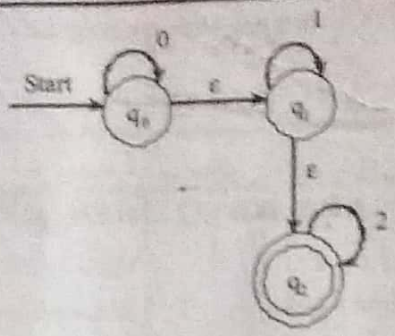


Figure: NFA

Q9. What is a state? What are different types of states?

Answer :

State

State defines the set of properties exchanged between the objects using communication channels. Any modifications done in the property is detected by the receiving object.

Types of States

The different types of states are,

(a) Start State

It is the initial state of FSM.

(b) Accepting State

FSM after finishing an input string changes its state from "start" to "accepting" wherein the string is accepted.

(c) Next State

It is the state which is immediately figured after the current state defined by the transition function.

(d) Universal State

It is the state in which FSM accepts only those moves that lead to acceptance.

(e) Dead/Trap State

It is the state in which the input symbol terminates on itself.

Q10. What is dead state? Give an example.

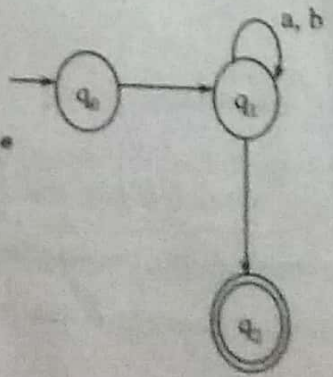
Answer :

Model Paper-II, Q10

Dead State

A transition wherein a state that once entered can not be left is called a dead or trap state. In a dead state, inputs cannot be processed further.

Example



In the above example, when q_0 receives input as 'a' it enters into 'q_1'. Then the machine remains in state q_1 when either of the input is given i.e., 'a' or 'b' then q_1 is a dead state.

UNIT 2 REGULAR EXPRESSIONS AND PROPERTIES OF REGULAR LANGUAGES

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. Define regular expression.

Answer :

Model Paper-I, Q1(c)

Regular Expression

A regular expression is a concise notation for denoting regular sets. Regular expressions describe the language accepted by a finite automata.

Let Σ be a finite alphabet. Then, the regular expressions over Σ denoting the regular sets are recursively defined as follows,

1. ϕ is a regular expression denoting the regular set $\{\phi\}$.
2. $\in$ is a regular expression denoting the regular set $\{\in\}$.
3. a is a regular expression denoting the regular set $\{a\}$.

If P and Q are regular sets of languages L_1 and L_2, then

- $(p + q)$ is a regular expression denoting the set $P \cup Q$
- pq is a regular expression denoting the set PQ
- P^* is a regular expression denoting the set P^*

Example

$0(0 + 1)^*1$ is a regular expression denoting the set of all strings over $\{0, 1\}$ starting with 0 and ending with 1.

Q2. Discuss about the identity rules for regular expressions.

Answer :

Two regular expressions P and Q are said to be equivalent (i.e., $P = Q$) if and only if both represent the same set of strings.

Let P, Q and R are regular expressions. The identity rules of regular expressions are given below. These rules are useful in simplifying regular expressions.

(i) $\phi + R = R$

(ii) $\phi R = R\phi = \phi$

(iii) $\in R = R\in = R$

(iv) $\phi^* = \in$ and $\in^* = \in$

(v) $R + R = R$

(vi) $RR^* = R^*R = R^+$

(vii) $\in + RR^* = R^*$ and $\in + R^* R = R^*$

(viii) $(R^*)^* = R^*$

(ix) $R^*R^* = R^*$

(x) $\in + R^* = R^*$

Q3. Write regular expression for the language over {0, 1}: the set of all strings that contain 1011.

Answer : Model Paper-II, Q1(c)

Given language is,

L = Set of strings that contain 1011

$\Sigma = \{0, 1\}$

The regular expression for given language L will be.

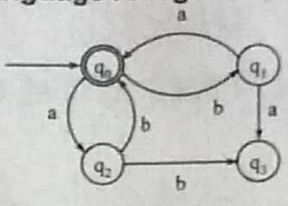
$(0 + 1)^* 1011 (0 + 1)^*$

$\therefore \quad R = (0 + 1)^* 1011 (0 + 1)^*$

The given language accepts strings like,

$W = \{010110, 110111, 010111, 110110, ...\}$

Q4. Find the regular expression for the language recognized by the following FA.



Answer :

The regular expressions for each state of the given FA are,

$q_0 = q_1 a + q_2 b + \in$ ... (1)

$q_1 = q_0 b$ ... (2)

$q_2 = q_0 a$ ... (3)

$q_3 = q_2 b + q_1 a$ ... (4)

Since q_0 is a final state, solving it ultimately gives the required regular expression. So we solve for q_0.

Substituting equations (2) and (3) in equation (1), we get,

$q_0 = q_0 ba + q_0 ab + \in$

$q_0 = q_0 (ba + ab) + \in \quad (\because R = Q + RP \Rightarrow QP^*)$

$q_0 = \in (ba + ab)^* \quad (\because \in . R = R)$

Since, q_0 is final state in the given FA. So , regular expression is given by q_0,

$\therefore \quad r = (ba + ab)^*$

Q5. What is a regular set? Give examples for it.

Answer : Model Paper-III, Q1(c)

Regular Sets

Let Σ be a finite alphabet. Then a class containing sets of strings over Σ, called regular sets are recursively defined as follows,

1. The empty set is a regular set over Σ.
2. Σ, the string of length 0 is a regular set over Σ.
3. Each input symbol say a in Σ is a regular set over Σ.
4. If P and Q are two regular sets over Σ, then
   - $P \cup Q$ is a regular set over Σ.
   - PQ is a regular set over Σ.
   - P^* is a regular set over Σ.

Examples

1. Let $\Sigma = \{0\}$, then the set of strings $\{0, 00, 000, ...\}$ is a regular set.
2. Let $\Sigma = \{0, 1\}$, then the set of strings $\{01, 10\}$ is a regular set.
3. Let $\Sigma = \{a, b\}$, then the set of strings starting with a and ending with b is a regular set.

Q6. Explain and prove "if L_1 and L_2 are two languages then $L_1 \cup L_2$ is regular".

Model Paper-III, Q1(d)

Answer :

The regular sets are closed under union, concatenation and Kleene closure. That is, if L_1 and L_2 are two regular languages then $L_1 \cup L_2$, L_1L_2 and L_1^* are also regular.

Proof

If L_1 and L_2 are regular sets then there exist regular expressions r_1 and r_2 denoting L_1 and L_2 respectively. That is,

$L_1 = L(r_1)$ and $L_2 = L(r_2)$.

By the definition, $r_1 + r_2$ is a regular expression denoting the language $L_1 \cup L_2$ i.e.,

$L_1 \cup L_2 = L(r_1 + r_2)$. Thus, $L_1 \cup L_2$ is also regular.

Similarly, the concatenation of regular sets and Kleene closure of a regular set is also regular.

Q7. Explain and prove "The complement of regular language is regular".

Model Paper-II, Q1(d)

Answer :

The regular sets are closed under complementation. That is, if L is a regular set such that $L \subseteq \Sigma^*$, then $\Sigma^* \to L$ is regular.

Proof

We need to prove that if L is a regular set then its complement $\bar{L}$ is also regular.

Let, L is accepted by a DFA $M = (Q, \Sigma, \delta, q_0, F)$ and $L \subseteq \Sigma^*$.

To accept $\bar{L}$, i.e., $(\Sigma^* - L)$ machine must complement the final states of M. So the DFA accepting $\bar{L}$ is given by $M' = (Q, \Sigma, \delta, q_0, \{Q - F\})$. That is in M' all the non-final states of M become as final states. This means the sets of strings accepted by M are rejected by M' and vice-versa.

Thus $\bar{L}$ accepted by M' is a regular set.

Q8. Let r_1 be the regular expression representing the language L_1, r_2 be the regular expression representing the language L_2, what is the language represented by the regular expression $r_2 + r_1$.

May/June-15(R13), Q1(d)

Answer :

Given that,

r_1 is the regular expression of language L_1.

r_2 is the regular expression of language L_2.

The language represented by regular expression $r_2 + r_1$ contains the set containing all strings of L_1 and L_2.

Q9. Describe the following sets by regular expressions,

(a) {101}

(b) {abba}

(c) {01, 10}

(d) {a, ab}.

Answer :

(a) {101}

Regular expression of the set {101}

Clearly, the language denoted by the required regular expression accepts only 101.

So, regular expression is 101.

(b) {abba}

Regular expression of the set {*abba*}

Clearly, the language denoted by the required regular expression accepts only *abba*.

So, regular expression is *abba*.

(c) {01, 10}

Regular expression for the set {01, 10}

It is clear that only strings 01 and 10 are accepted.

∴ The regular expression is 01, 10.

(d) {a, ab}

Regular expression for the set {*a*, *ab*}

It is clear that only strings *a* and *ab* are accepted.

∴ The regular expression is *a*, *ab*.

Q10. Describe in the English language, the sets represented by the following regular expressions,

(a) a(a+b)*ab

(b) a*b + b*a.

Answer :

Model Paper-I, Q1(g)

(a) **a(a + b)*ab**

The regular expression represents the set of all start with '*a*' and ends with *ab*.

Example

aaab and *aabbab* are some valid strings.

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

The regular expression represents the set of all strings that have any number of *a*'s before *b* or any number of *b*'s before *a*.

Example

a, *b*, *aab*, *bba* are some valid strings.

UNIT 3

CONTEXT-FREE GRAMMARS, PUSHDOWN AUTOMATA

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. What are the applications of CFG?

Answer :

Model Paper-II, Q1(e)

There are many applications of CFG. Some of them are,

1. It is used by compilers to describe syntaxes of programming languages.
2. With the help of CFG compilers implements PDA to recognize syntax.
3. It can also be used for developing computation models.
4. It helps in generating parse trees.
5. It helps in describe up syntactic structure of natural languages.
6. It is used in markup language like HTML and XML.

Q2. Give an example for context free language.

Answer :

(Model Paper-II, Q1(f) | May-16(R13), Q1(e))

Example of context free language,

(i) $L = \{a^i b^j \mid i \geq 1 \text{ and } j \geq 1\}$

(ii) $L = \{a^i b^j \mid i \neq j\}$

(iii) $L = \{a^i b^j \mid i \neq j \text{ and } i \neq 2j\}$

Q3. Define derivation and its types.

Model Paper-I, Q1(e)

Answer :

Derivation

Derivation is a method of replacing the nonterminal symbol present on RHS of a production rule with a terminal input. This is the process of deriving a input string from a set of production rules is called as derivation.

Types of Derivation

There are two types of derivations

1. **Left-sentential Form**

   If $A \underset{lm}{\overset{*}{\Longrightarrow}} \beta$ using leftmost derivation then β is a left-sentential form of the grammar.

2. **Right-sentential Form**

   If $A \underset{lm}{\overset{*}{\Longrightarrow}} \beta$ using rightmost derivation (canonical derivation) and β may contain non-terminals, then β is a right-sentential form of the grammar.

Q4. Write a context free grammar for the language {0^n1^n|n>=1}.

Answer : May-16(R13), Q1(f)

The given language $L = \{0^n1^n | n \geq 1\}$ accepts sets of string where number of 0's are equal to number of 1's.

It accepts strings like,

$S = \{01, 0011, 000111, 010101, 0101...\}$

The context free grammar for the above language $L = \{0^n1^n \mid n \geq 1\}$ is,

$$P = \begin{Bmatrix} S \to 0S1 \\ S \to 01 \end{Bmatrix}$$

Where,

$G = [\{S\}, \{0, 1\}, P, \{S\}]$

Let us derive a string $\omega = 0011$ from the above grammar

$S \to 0S1$

$S \to 0011 \quad (\because S \to 01)$

$\therefore$ The generated grammar accepts strings or language $L = \{0^n1^n | n \geq 1\}$.

Q5. A variable A in a CFG is said to be reachable if S ==>* α A β. Develop a procedure for finding out reachable variable in a given CFG. Illustrate your procedure with an example.

Answer :

A procedure for finding reachable variable in a CFG is as follows,

1. Mark the start variable as reachable.
2. Mark a variable Y as reachable if there exists production $X \to w$ where 'X' is a variable previously marked as reachable and 'w' is string containing Y.
3. Repeat the above step till no further variables get marked "reachable".

Example

Consider grammar, G

$G : S \to BS/B$

$A \to aA/a$

$B \to b$

1. Mark the start symbol 'S' as reachable.
2. The variable 'A' is not reachable because it is not reachable from 'S'. Since, $S \to BS/B$
3. Mark the variable 'B' as reachable because it is reachable from 'S'.

$\therefore$ Reachable variables of 'G' = $\{S,B\}$.

Q6. Show that the language,

$L_1 = \{0^n1^m \mid n = m \text{ and } n \geq 1\}$

is deterministic context free language.

Answer :

The given language $L_1 = 0^n1^n$ generates the strings containing any number of 0's followed by any number of 1's which is equal to the number of 0's.

$L_1 = \{0^n 1^m | n = m \text{ and } n \geq 1\}$

If there is a DPDA for a language, then the language is said to be a deterministic context free language.

Let DPDA be $M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{z, a\}, \delta, q_0, z, \{q_2, q_3\})\delta$ is given below.

$\delta(q_0, 0, z) = (q_3, az)$

$\delta(q_3, 0, a) = (q_1, aa)$

$\delta(q_1, 0, a) = (q_1, aa)$

$\delta(q_1, 1, a) = (q_1, \in)$

$\delta(q_1, \in, z) = (q_2, z)$

Since there is a DPDA for the given language, the language is deterministic context free language.

Q7. Define ambiguous grammar with example.

Answer :

Ambiguous Grammar

Model Paper-I, Q1(f)

A grammar is said to be ambiguous, if it produces more than one derivation tree for some input string generated by it. In other words, if there are more than one leftmost derivation or more than one rightmost derivation for a given string. Then such a grammar is said to be ambigious.

An ambiguous grammar contains productions of the form,

$S \rightarrow S \alpha S$

Where, α may be string of terminals or non-terminals i.e., the same non-terminal appears twice on R.H.S of a production.

Example

Consider the following grammar,

$S \rightarrow SS \mid a \mid b$

Let us derive a string from this grammar,

$S \Rightarrow SS$

$\Rightarrow SSS$

$\Rightarrow aSS$

$\Rightarrow abS$

$\Rightarrow aba$

Q8. Show that the grammar is ambiguous,

S → a|Sa|bSS|SSb|SbS.

Answer :

Given grammar, G is

$S \rightarrow a \mid Sa \mid bSS \mid SSb \mid SbS$

The above grammar is ambiguous because it contains production of the form $S \rightarrow S \propto S$.

Let us prove it by constructing two distinct derivation trees for a string in $L(G)$.

Consider string $\omega = babaaa$ which is generated by G.

Let us first derive a string from this grammar,

$S \Rightarrow bSS$

$\Rightarrow bSaS$

$\Rightarrow bSbSaS$

$\Rightarrow babSaS$

$\Rightarrow babaaS$

$\Rightarrow babaaa$

Now construct derivation trees for the string *bababa*.

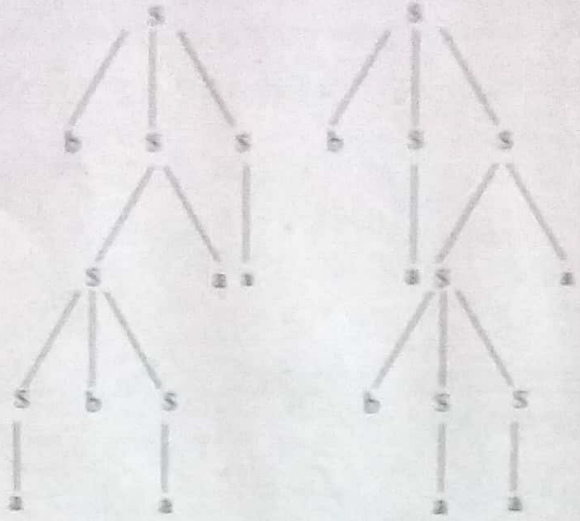


Figure: Parse Trees of bababa

Since, there are two distinct parse trees for the string *bababa*, the given grammar is ambiguous.

Q9. Give the formal definition of a PDA.

Answer :

Model Paper-III, Q1(e)

A finite automata with a stack memory can be viewed as pushdown automata.

Formally, a push down automata is defined as a seven tuple machine, $M = (Q, \Sigma, \tau, \delta, q_0, z_0, F)$

Where,

Q= Set of finite states

Σ= Set of input symbols

τ = Set of symbols push on top of stack

δ = Transition function mapping from

$$Q \times (\Sigma \cup \{\Sigma\}) \times F \to Q \times \tau^*$$

q_0 = Initial state and

$F \subseteq Q$ is set of final state.

Q10. Give any two examples of languages that are accepted by PDA.

Answer :

A push down automata can accept two types of languages.

(i) Context free language and

(ii) Regular language.

Example

(i) $L(M) = \{WCW^R \mid W \in (a+b)^*\}$ where W^R is reverse of W.

(ii) $L = \{w \mid w \in (a+b)^* \text{ and } n_a(w) = n_b(w)\}$. The given language must accepts equal number of a's followed by equal number of b's.

Q11. Construct the PDA for the following grammar. S → AA|a A → SA|b.

Answer :

Let PDA be $P = (Q, \Sigma, \Gamma, \delta, S, F)$

Where,

$Q = \{p, q\}$

$\Sigma = \{a, b\}$

$\Gamma = \{S, A, a, b\}$

$S = \{P\}$

$F = \{q\}$

δ is given as follows,

$R_1 : \delta(p, \in, \in) = \{(q, S)\}$

$R_2 : \delta(q, \in, S) = \{(q, AA)\}$

$R_3 : \delta(q, \in, S) = \{(q, a)\}$

$R_4 : \delta(q, \in, A) = \{(q, SA)\}$

$R_5 : \delta(q, \in, A) = \{(q, b)\}$

$R_6 : \delta(q, a, a) = \{(q, \in)\}$

$R_7 : \delta(q, b, b) = \{(q, \in)\}$

Let us apply this on string *aabb* accepted by the given grammar.

State	Input	Stack	Rule Used
P	aabb	$\in$	-
q	aabb	S	R_1
q	aabb	AA	R_2
q	aabb	SAA	R_4
q	aabb	aAA	R_3
q	abb	AA	R_6
q	abb	SAA	R_4
q	abb	aAA	R_3
q	bb	AA	R_6
q	bb	bA	R_5
q	b	A	R_7
q	b	b	R_5
q	$\in$	$\in$	R_7

Q12. Write a short notes on DPDA.

Answer :

Model Paper-III, Q1(f)

Deterministic Push Down Automata (DPDA) helps in determining various constructs which remains effective for a given language. DPDA accepts deterministic context free languages as inputs.

Hence few languages supported by DPDA are as follows,

(i) Language $(L) = \{i^m, j^m : m \neq n \geq 0\}$

(ii) Language $(L) = \{i^n \subset j^n : n \geq 0\}$

(iii) Language $(L) = \{i^n j^n : n \geq 0\}$

UNIT 4. PROPERTIES OF CONTEXT-FREE LANGUAGES, INTRODUCTION TO TURING MACHINES

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. What is unit production? If you eliminate the unit productions from the given CFG, what will be the effect on the language by the resultant grammar.

Answer :

Unit Production

Model Paper-I, Q1(g)

A production in which one non-terminal derives another non-terminal i.e., $A \rightarrow B$ is called a unit production.

Example

Consider the grammar,

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow b$

Here, the productions $B \rightarrow C$ and $C \rightarrow D$ are unit productions. These productions can be eliminated directly by substituting their terminal symbol.

If we eliminate unit production from the given CFG then there will not be any effect on language of resultant grammar. That is, the language generated by the given grammar will be the same.

Q2. Differentiate Chomsky and Greibach normal forms.

Answer :

	Chomsky Normal Form		Greibach Normal Form
1.	Noam Chomsky invented this normal form. Hence, it is named as Chomsky Normal Form (CNF)	1.	Sheila Greibach invented this normal form. Hence, it is named as Greibach Normal Form (GNF).
2.	It limits the number of symbols used on the production's right hand side.	2.	It limits the positions wherein terminals and variables can be used instead of limiting the length of production's right hand side.
3.	It can be said that CFG is CNF when each production is in either $A \rightarrow BC$ or $A \rightarrow a$ form.	3.	It can be said that CFG is CNF when each production is in $A \rightarrow ax$ form.
4.	It is used to increase the efficiency of an algorithm.	4.	It is used for the following reasons, (i) Suitable parsing (ii) Acceptance of CFL by non-deterministic PDA and (iii) PDA equivalent CFG.

Q3. Write the general procedure to transform a grammar to Chomsky normal form.

Model Paper-III, Q1(h)

Answer :

The procedure for conversion of a grammar to CNF involves the following steps,

Step1

Removing null and unit productions

Step2

Removing terminals from RHS

Step3

Limiting variables on RHS. i.e., converting the productions in the required form (A →BC, A → a).

Q4. Define turing machine.

Answer :

Turing Machine (TM) is a simple mathematical model that performs calculations similar to any computing machine i.e., a general purpose computer. A Turing Machine (TM) is a seven tuple machine (M) denoted as $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where,

Q contains the finite set of states

T is the finite set of allowable tape symbols

B is a blank symbol of Γ

Σ (a subset of Γ not including B), is the set of input symbols

δ is the next move function, a mapping form $Q \times$ F to Q to T $\times$ $\{L, R\}$

q_0 is Q in the start state

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

Q5. Describe the TM that accepts the language,

L = {w ∈{a,b,c}* | w contains equal number of a's, b's and c's}.

Answer :

Model Paper-II, Q1(g)

A turing machine for the given language is,

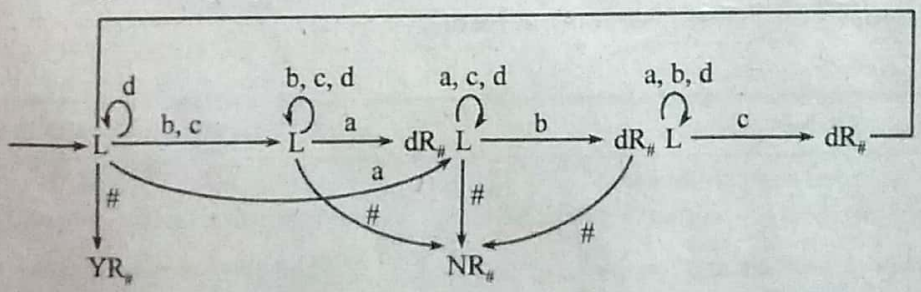


Figure: Turing Machine for Given Language R → Right, L → Left, Y → Accept, N → Reject, # → Blank Symbol

The turing machine works as follows,

1. The TM starts from right by searching for 'a'. On encountering 'a', the TM moves back to the extreme right. Finally, 'a' is replaced by some other non-input symbol, say d.
2. After finishing the search for 'a', the TM searches for 'b' and replaces 'b' by 'd' and moves to the extreme right.
3. Now, the TM searches for 'c' and replaces 'c' by 'd',
4. Step (1) to step (3) completes one cycle of TM. These steps are repeated, until all a's, b's and c's are replaced by 'd'.
5. If all symbols are replaced by 'd' then the TM accepts a string. Otherwise, it rejects a string.

Q6. Write a short note on subroutine.

Model Paper-III, Q1(g)

Answer :

Subroutine is commonly used in programming languages and it can be implemented in turing machine.

It is defined as a self-contained group of instructions that can perform one task. It is a portion of the software that can be stored in memory once, but can be used as many times as required. This leads to less usage of memory and easy software development.

At the time of program execution, a subroutine is called multiple times to carryout its function at different points. After the execution of the subroutine, control is returned back to the main program so that the program may resume its normal operations.

Q7. Discuss the guidelines for designing a turing machine.

Answer :

The guidelines for designing a turing machine are as follows,

(i) The tape or read write head of turing machine must be familiar with the operations to be performed later. Machine must also keep track of the symbols previously scanned. To do this, the machine can move forward towards the next state.

(ii) The machine must be designed such that minimum number of states should be used. To do this, state is changed only when the written symbol is modified or when the movement of read/write head is changed.

Q8. Define multi-tape turing machine.

Answer :

Model Paper-I, Q1(h)

Multi-tape turing machine is a modification of the two-way infinite tape TM. It consists of 'n' tapes and 'n' tape heads. Similar to the two-way infinite tape, each tape is infinite in both directions. Depending on the state of the finite control and the symbol read by each tape head the machine can change the state and independently move each of its tape heads.

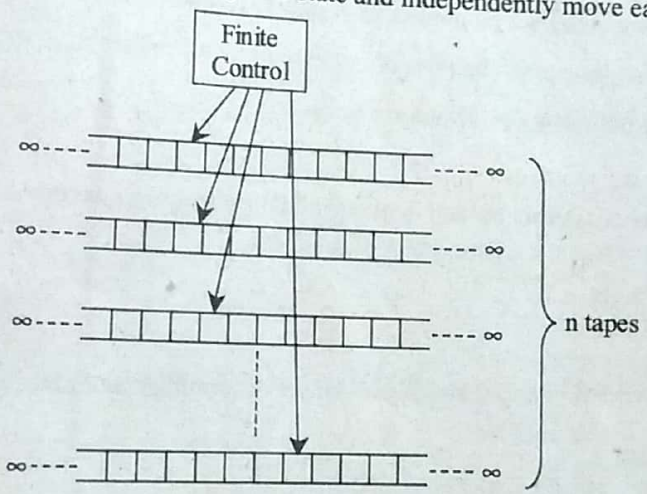


Figure: Multitape Turing Machine

Q9. Define universal turing machine.

Answer :

Model Paper-II, Q1(h)

A turing machine is said to be Universal TM (U_m) if it is capable of simulating computation of turing machine (M) by taking its description and string w as input.

It is constructed in the following way.

Consider some turing machine,

$M = (Q, \{0, 1\}, \{0, 1, B\}, \delta, q, B, q_1\}$

Where,

Q is the set of states (say $q_0, q_1, q_2 \ldots q_n$)

$\{0, 1\}$ is the input alphabet denoted by Σ

$\{0, 1, B\}$ is the tape alphabet denoted by Γ where symbols are represented as $a_1, a_2, a_3 \ldots a_n$

δ is the transition function

q_0 is the initial state

B is the symbol to represent blank cells

q_1 is the only final state.

UNIT 5

UNDECIDABILITY AND INTRACTABLE PROBLEMS

SIA GROUP

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. Discuss in brief about undecidability.

Answer :

Model Paper-I, Q1(i)

A problem P which cannot be solved by any algorithmic approach is referred as undecidable. In other words, problems for which no algorithm exists are undecidable.

A given problem is undecidable if,

- Its corresponding language is nonrecursive and
- It has no solution or answer or algorithm.

Above consequence is also referred as undecidability.

Q2. Give the decidable problem for recursive and recursively enumerable language.

Answer :

Model Paper-II, Q1(i)

A recursive enumerablely language is decidable if it is recursive.

1. For a given recursive enumerablely language L, L' is recursive.
2. For given two recursive languages L_1 and L_2
   (i) $L_1 \cup L_2$ is recursive
   (ii) L_1' is recursive or L_2' is recursive.

Q3. Discuss in brief about PCP.

Answer :

Let, "Σ" represents finite alphabet and $A = \{i_1, i_2, \ldots i_n\}$ and $B = \{j_1, j_2, \ldots j_n\}$ represents two separate lists of strings defined over "Σ" such that $|P| = |Q|$. Then according to Post's Correspondence Problem (PCP), it must be determined whether/not existence of list $a_1, a_2, \ldots a_n$ $(n \geq 1)$ and $i_{a_1}, i_{a_2} \ldots i_{a_n} = j_{a_1}, j_{a_2} \ldots j_{a_n}$.

Q4. Show that the following post correspondence problem has a solution and give the solution.

i	List A W_i	List B X_i
1	11	11
2	100	001
3	111	11

Answer :

In this case, PCP has a solution. The required sequence is given by,

$i_1 = 1$, $i_2 = 2$, $i_3 = 3$ i.e., (1, 2, 3) and $m = 4$.

1	2	3	1	2	3
1 1	100	111	111	001	11
w_1	w_2	w_3	x_1	x_2	x_3

The sequence is $w_1w_2w_3 = x_1x_2x_3 = 11100111$.

Q5. Define P Problem.

Answer : Model Paper-III, Q1(i)

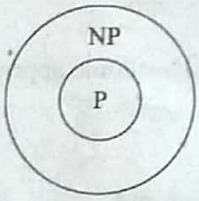
P-problem is a problem that can be solved in polynomial time deterministic turing machines over the size of the input.

P-problems are usually decision problems that include P-class. All problems in P-class are said to be tractable if the time and space complexity for machine is reasonable to solve that problem. Otherwise, the problem of P-class is called as intractable.

Q6. Discuss about NP Problem.

Answer : Model Paper-II, Q1(j)

NP-problems are those problems that can be solved in polynomial time by Nondeterministic Turing Machine (NTM). Usually, NP-problems cannot be solved efficiently. The relation between the P and NP-problems is $P \subseteq NP$ which means that the P-problems is subset of or equal to the set of NP-problems. Graphically, it can be represented as,



Q7. Give few examples on P and NP class problems.

Answer : Model Paper-I, Q1(j)

P-Class Problems

Examples of this type include,

1. Transitive closure
2. Matrix multiplication
3. Kruskal's algorithm for minimum spanning.

NP-Class Problems

Examples of this type include,

1. Four color problem
2. Travelling salesman problem
3. The vertex cover problem
4. The hamiltonian cycle or hamiltonian circuit problem.

Q8. What are the differences between NP-hard and NP-complete classes?

Answer : Model Paper-III, Q1(j)

	NP-Hard		NP-Complete
1.	A decision problem P_i is said to be NP-hard, if every problem in NP-hard is reduced to P_i in polynomial time.	1.	A decision problem P_i is said to be NP-complete, if it is NP-hard and also in the class NP itself.
2.	Symbolically it is represented as $P_j \rightarrow$ NP where in P_j is a polynomial not equal to P_i.	2.	Symbolically it is represented as $P_i \rightarrow$ NP.
3.	The problem P_i is harder as the other problems in NP.	3.	The problem P_i is the most complex of all.
4.	If any NP-hard problem is proved as intractable then decision problem P_i must also be intractable.	4.	If any NP-complete problem is intractable, then every NP-complete problem is also intractable.

Q9. Discuss Rice's theorem and its properties.

Answer :

Rice's theorem states that, problem P is undecidable if P follows nontrivial property of regular expression.

Rice's theorem helps in determining whether a property 'P' of regular expression is decidable or undecidable. By using this theorem, the following properties of the Regular Expressions (R.E) are undecidable.

(i) Context freedom

(ii) Finiteness

(iii) Emptiness

(iv) Regularity.

The Rice's theorem is not applicable if the property 'P' is trivial. A decision problem is said to be trival if it generates either yes or no for every instance. Trivial is used to describe a property that is possessed either by all the recursively enumerable languages or by none of them.

Q10. What are the undecidable problems about turing machine specification.

Answer :

Some of the undecidable problems about the turing machine that accepts the languages accepted by TM are as follows,

1. Determining that the language accepted by a turing machine is empty or not.
2. Determining that the language accepted by a turing machine is regular or not.
3. Determining that the language accepted by a turing machine is context-free or not.
4. Determining that the language accepted by a turing machine is finite or not.

Mostly, the questions/problems about the state of the turing machine are decidable.

Q11. Write a short notes on Non-deterministic polynomial time.

Answer :

Nondeterministic Polynomial time (NP) is the Polynomial time in which the NP-problems can be solved by Nondeterministic Turing Machine (NTM). A language L belongs to NP class, if there exist a polynomial time complexity $T(n)$ and a non-deterministic Turing Machine M, and satisfy the condition, $L = L(M)$. If the input of length (n) is given to M then maximum $T(n)$ moves exist. Since, every DTM is NTM that declines the condition that $P \subseteq NP$. However, NP consists of problems that do not belong to the class P. The NTM running in a Polynomial time is capable of guessing the exponential number of possible solutions to a given problem and verifies each solution at the same time.