Solution	Marks

Q.2 a. Prove that for every positive integer n; the number of subsets of an n-element set is 2^n (using mathematical induction). **(8)**

Answer:

Let P(n) be the proposition that for every positive integer n; the
number of subsets of an n-element set is 2n: We must show that P(1) is true and that
the conditional statement P(k)) P(k + 1) is true for k = 1; 2; 3; : : :
Let Sn = fa1; a2; : : : ; ang be a set containing n elements.

BASE STEP P(1) is true since the set S1 has only two subsets viz. _ and fa1g. **2 Marks**

INDUCTIVE STEP For the inductive hypothesis we as- sume that P(k) holds for an arbitrary positive integer k: That is, we assume that, for every positive integer k; The number of subsets of a k element set is 2k: (3) Under this assumption, it must be shown that P(k + 1) is , i.e. we shall prove that the number of subsets of a (k + 1)-element set is 2k+1: Consider Sk+1 = fa1; a2; : : : ; ak; ak+1g be a set containing k + 1 elements. The subsets of Sk+1 either contain ak+1 or do not contain it. The subsets not containing ak+1 are precisely the subsets of Sk: Using (3), we get that this number is 2k. The subsets not containing ak+1 are precisely the subsets of Sk[fak+1g : Again using (3) we get that this number is 2k. Thus the total number of subsets of Sk+1 equals 2k + 2k = 2k(1 + 1) = 2k+1: **2 Marks for inductive step**

Since we have completed both, the base step and the inductive step, we have shown that P(n) is true for all positive integers n. **4 Marks**

b. Define Chomsky hierarchy with its automation and production rules. **(8)**

Answer:

- **Type-0 grammars (unrestricted grammars)** include all formal grammars. They generate exactly all languages that can be recognized by a Turing machine. These languages are also known as the recursively enumerable languages. Note that this is different from the recursive languages which can be *decided* by an always-halting Turing machine. **2 Marks for each type/level**
- **Type-1 grammars (context-sensitive grammars)** generate the context-sensitive languages. These grammars have rules of the form with a nonterminal and , and strings of terminals and nonterminals. The strings and may be empty, but must be nonempty. The rule is allowed if does not appear on the right side of any rule. The languages described by these grammars are exactly all languages that can be recognized by a linear bounded automaton (a nondeterministic Turing machine whose tape is bounded by a constant times the length of the input.)
- **Type-2 grammars (context-free grammars)** generate the context-free languages. These are defined by rules of the form with a nonterminal and a string of terminals and nonterminals. These languages are exactly all languages that can be recognized by a non-deterministic pushdown automaton. Context-free languages – or rather the subset of deterministic context-free language – are the theoretical basis for the phrase structure of most programming languages, though their syntax also includes

context-sensitive name resolution due to declarations and scope. Often a subset of grammars are used to make parsing easier, such as by an LL parser.

- **Type-3 grammars (regular grammars)** generate the regular languages. Such a grammar restricts its rules to a single nonterminal on the left-hand side and a right-hand side consisting of a single terminal, possibly followed by a single nonterminal (right regular). Alternatively, the right-hand side of the grammar can consist of a single terminal, possibly preceded by a single nonterminal (left regular); these generate the same languages – however, if left-regular rules and right-regular rules are combined, the language need no longer be regular. The rule is also allowed here if does not appear on the right side of any rule. These languages are exactly all languages that can be decided by a finite state automaton. Additionally, this family of formal languages can be obtained by regular expressions. Regular languages are commonly used to define search patterns and the lexical structure of programming languages.

Grammar	Languages	Automaton	Production rules (constraints)
Type-0	Recursively enumerable	Turing machine	$\alpha \rightarrow \beta$ (no restrictions)
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \gamma$
Type-3	Regular	Finite state automaton	$A \rightarrow a$ and $A \rightarrow aB$

Q.3 a. **Find a deterministic finite accepter that recognizes the set of all strings on X:{a, b} starting with the prefix ab.** **(8)**

Answer:

The only issue here is the first two symbols in the string after they have been read, no further decisions need to be made. We can therefore solve the problem with an automaton that has four states; an initial state, two states for recognizing ab ending in a final trap state, and one nonfinal trap state. If the first symbol is an a, and the second is a b, the automaton goes to the final trap state, where it will stay since the rest of the input does not matter. On the other hand, if the first symbol i$ not an a or the $econd one is not f, b, the automaton enters the nonfinal trap state' The simple solution is shown in **Figure 2.4.**

4 Marks

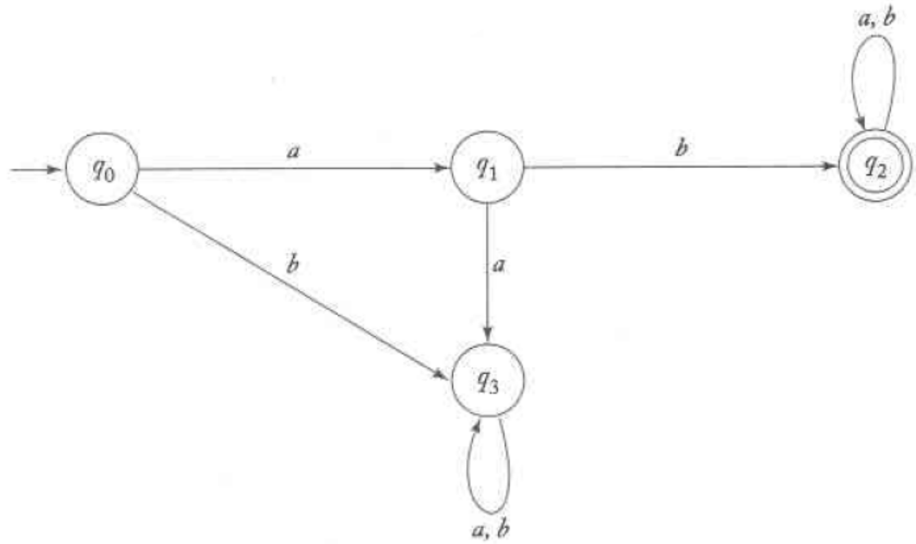


Fig.2.4

4 Marks

b. Define a non-deterministic automata. Convert the NFA M = ({q_0, q_1, q_2}, {0, 1}, δ, q_0, {q_2}) into a DFA. The transition function δ is given as:-(8)

δ	0	1
→ q_0	q_0	q_0, q_1
q_1	q_2	q_2
q_2	ϕ	ϕ

Answer:

NDFA → See Text Book I. → Page 54, 2.3.2.

δ_D for DFA is given as: 2 Marks.

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

⇐ 2 Marks for table

$Q_D = P(Q) \Rightarrow$ Power set of Q. (1-mark each)

$\Sigma_D = \{0,1\}$.

$q_{0_D} = \{q_0\}$.

$F_D \Rightarrow$ all subsets of Q_D containing q_2 as an ~~state~~ element.

eg → $\{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}$.

$\delta_D(\{q_i, q_j\}, a) = \delta(q_i, a) \cup \delta(q_j, a)$

Transition function can be constructed by this rule

4 Marks

Q.4 a. Find a regular expression for the language,
L = {W €{0,1}*: W has no pair of consecutive zeros}. (8)

Answer:

One helpful observation is that whenever a 0 occurs, it rnust be followed immediately by a 1.such as substring may be preceded and followed by an arbitrary number of 1's. This suggest that the answer involves the repetition of strings of the form 1 ....101 .....1, that is, the language denoted by the regular expression(1*011*)*. However, the answers still incomplete, since the string ending in 0 or consisting of all 1's are unaccounted for.

After taking care of these special cases we arrive at the answer.

$$r = (1^*011^*)^*(0+\lambda)+1^*(0+\lambda)$$

L as the repetition of the strings 1 and 01, the shorter expression

$$r = (1+01)^*(0+\lambda)$$

rnight be reached. Although the two expressions look different, **both answer are correct,** as they denote the same language. Generally, there are an unlimited number: of regular expressions for any given language.

8 Marks

b. Convert the regular expression to NFA with $\in$-transition.
r.e. $\Rightarrow (0+1)^* \mid (0+1)$ **(8)**

Answer: **Refer page 97 of Text Book-I**

Q.5 a. Prove that if L and M are regular languages, then $L \cap M$ is also regular.(8)

Answer: **Refer Page 126 of Text Book-I**

b. Using the pumping lemma to show that L:$\{a^n b^n: n \geq 0\}$ is not regular. (8)

Answer:

Assume that L is regular, so that the pumping lemma must hold. We do not know the value of m, but whatever it is, we can always choose n=m
Therefore, the substring y musl consist entirely of a's. Suppose lyl = k
The the string obtained by using i = 0
$W_0=a^{m-k} b^m$.
And is clearly not in .L. this contradicts the pumping lemma and thereby indicates that the assumption that L is regular must be false.

8 Marks

Q.6 a. Prove that language L = $\{a^n b^m : n \neq m\}$ is context free language. (8)

Answer:

Take the case n>m. We first generate a string with an equal number of a's and b's, then add extra a's on the left. This is done with

$S \rightarrow AS_1,$
$S_1 \rightarrow aS_1 b \mid \lambda,$
$A \rightarrow aA \mid a.$

4 Marks

We can use sirnilar reasoning for the case n <m and we get the answer

$S \rightarrow AS_1 \mid S_1B,$
$S_1 \rightarrow a S_1 b \mid \lambda,$
$A \rightarrow aA \mid a,$
$B \rightarrow bB \mid b.$

The resulting **grammar is context-free**; hence L is a context-free language. However, the grammar **is not linear**

4 Marks

b. Construct an npda for the language **(8)**
L = { w €{a,b}* : $n_a(w) = n_b(w)$}.

Answer:

Tlte complete solution is is an npda is given as:

$\delta(q_0, \lambda, z)=\{(q_f, z)\}$,
$\delta(q_0, a, z)=\{(q_0, 0z)\}$,
$\delta(q_0, b, z)=\{(q_0, 1z)\}$,.
$\delta(q_0, a, 0)=\{(q_0, 00)\}$,
$\delta(q_0, b, 0)=\{(q_0, \lambda)\}$,
$\delta(q_0, a, 1)=\{(q_0, \lambda)\}$,
$\delta(q_0, b, 1)=\{(q_0, 11)\}$,

1 Mark for each

In processing the string baab, the npda, makes the moves

$(q_0, baab, z) \vdash (q_0, aab, 1z) \vdash (q_0, ab, z)$
$\vdash (q_0, b, 0z) \vdash (q_0, \lambda, z) \vdash (q_f, \lambda, z)$

and hence the string is accepted.

2 Marks for processing

Q.7 a. Convert the grammar with start symbol S, to Chomsky normal form. Show all the relevant steps briefly. (8)

$S \rightarrow \varepsilon \mid c\,ST \mid TSc \mid SS$,
$T \rightarrow a \mid b$

Answer:

Step 1: Remove unit and ε productions.
There are no unit productions. The ε production $S \rightarrow \varepsilon$ can be removed after adding the new rules $S \rightarrow cT$ and $S \rightarrow Tc$.

2 Marks

Step 2: Add non-terminal symbols for elements of Σ.
We add three non-terminal symbols A,B,C and the three rules $A \rightarrow a$, $B \rightarrow b$ and $C \rightarrow c$. We then rewrite the two rules $S \rightarrow cT$ and $S \rightarrow Tc$ respectively as $S \rightarrow CT$ and $S \rightarrow TC$. Moreover, $S \rightarrow cST$ is rewritten as $S \rightarrow CST$, and $S \rightarrow TSc$ as $S \rightarrow TSC$.

3 Marks

Step 3: Handle productions with more than two non-terminal symbols on the right sides.
We replace the rule $S \rightarrow CST$ by the two rules $S \rightarrow CU$ and $U \rightarrow ST$. Similarly, we replace the rule $S \rightarrow TSC$ by the two rules $S \rightarrow TV$ and $V \rightarrow SC$.
The grammar in Chomsky normal form, therefore, consists of the following rules. The non-terminal symbols A,B are redundant and not shown here.

$S \rightarrow CT \mid TC \mid CU \mid TV \mid SS$,
$U \rightarrow ST$,
$V \rightarrow SC$,
$T \rightarrow a \mid b$,
$C \rightarrow c$.

3 Marks

b. Show that language $L = \{a^n b^n : n \geq 0, n \neq 100\}$ is context free. (8)

Answer:

Let
L1= {$a^{100}b^{100}$}.
Then, because L1 is finite, it is regular. Also, it is easy to see that,
L1 = {a^n b^n : n≥0} ∩ L1' (L1' is complement of L1)
Therefore, by the closure of regular languages under complementation and the closure of context-free languages under regular intersection, the desired result follows.

Q.8 a. Proceed with the following tasks: (8)

(i). Draw a state diagram of a Turing Machine M recognizing the language L = { a^n b^n a^n n ≥ 0} over the alphabet ∑ = {a, b}.

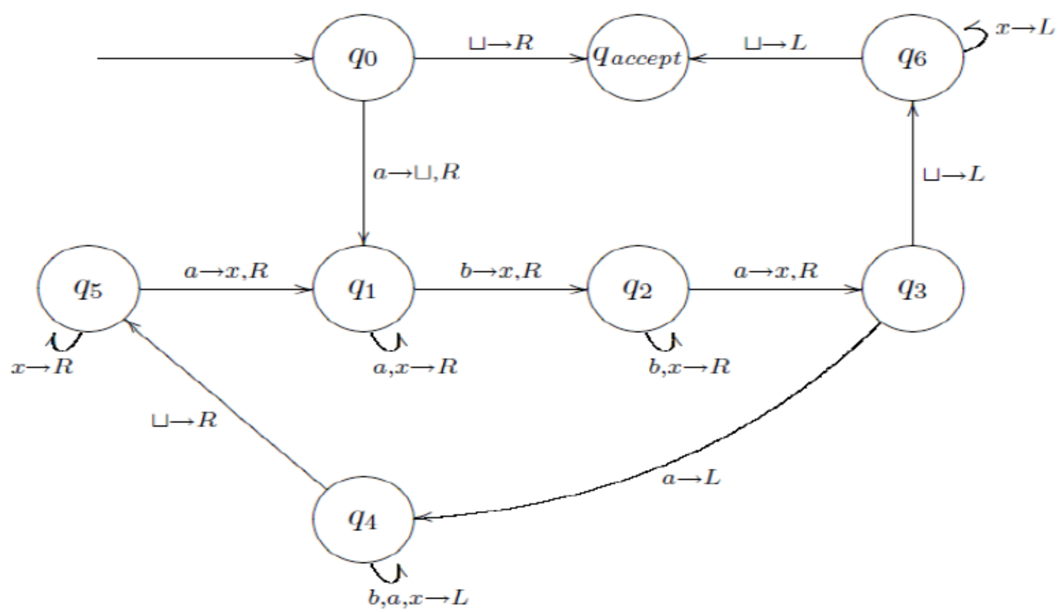
(ii). Consider the input string w = aabbaa. Write the whole sequence of configurations that M will enter when run on w.

(iii). Does M accept w?

Answer:

The correct solutions are as follows (note that your solution might still be correct even though your Turing machine looks differently):

1.



4 Marks

All missing transitions in the picture go implicitly to the state q_{reject}.

2. For the input string aabbaa the machine M will pass through the following sequence of configurations:

q0aabbaa →	|_| q1abbaa →	|_|aq1bbaa →
|_|axq2baa →	|_|axbq2aa →	|_|axbxq3a →
|_|axbq4xa →	|_|axq4bxa →	|_|aq4xbxa →
|_|q4axbxa→	q4|_| axbxa→	|_|q5axbxa →
|_|xq1xbxa→	|_|xxq1bxa→	|_|xxxq2xa →
|_|xxxxq2a→	|_|xxxxxq3→	|_|xxxxq6x →
|_|xxxq6xx →	|_|xxq6xxx→	|_|txq6xxxx→
|_|q6xxxxx→	q6|_|xxxxx→	q_{accept}|_| xxxxx

3 Marks

3. Yes.

1 Mark

b. **Define Turing Machine and explain it's working. Also define the language accepted by a TM. (8)**

Answer: **Refer Article 8.2.2 & 8.2.5, pages 295 & 306 of Text Book-I**

Q.9 a. Define the Turing Machine Halting Problem. (8)

Answer:

The Halting Problem:

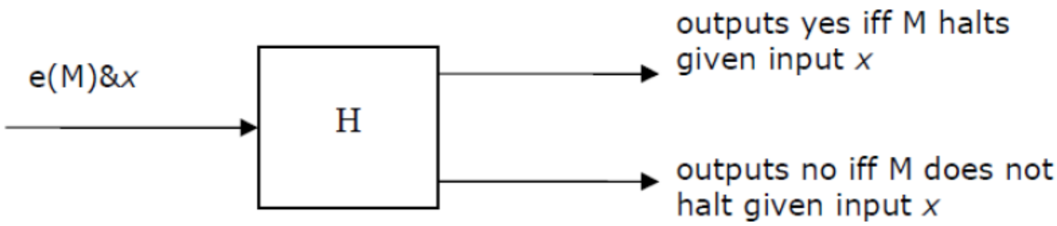
In computability theory, the **halting problem** can be stated as follows: "*Given a description of an arbitrary computer program, decide whether the program finishes running or continues to run forever*". This is equivalent to the problem of deciding, given a program and an input, whether the program will eventually halt when run with that input, or will run forever.

An algorithm can be defined as a procedure that always terminates and gives an answer. Formally, an algorithm can be defined as a Turing machine that always halts (whether succeeds or fails).

A problem is said to be solvable if there is an algorithm that solves the given problem (every instance).

A problem is said to be unsolvable if no algorithm exists that solves the given problem.

The Halting Problem for Turing machines is defined as follows:
Given a TM $M=(Q,\Sigma,\Gamma,\delta,q0,F)$ and an input string $x \in \Gamma^*$, will M eventually halt?
Then, the Halting problem would be solvable if a Turing machine H that behaves like illustrated below, can be constructed:



In the above, the symbol & is a separator and e(M) is an encoding of M, i.e. e(M) is for example the set of 5-tuples (q,x1,p,r,R) that describe the Turing machine.

The Halting problem is then:
"Does there exist an effective procedure (computable function) for deciding, for every pair (e(M),x); does M halt for x?"

8 Marks

b. Define the Post Correspondence Problem. (8)
Let $\sum = \{0, 1\}$ and take A and B as
$w_1 = 11$, $w_2 = 100$, $w_3 = 111$
$v_1 = 111$, $v_2 = 001$, $v_3 = 11$. Give a PC solution for this problem.

© IETE

If we take
$w_1 = 00$, $w_2 = 001$, $w_3 = 1000$
$v_1 = 0$, $v_2 = 11$, $v_3 = 011$
Then, is there PC solution exist? Justify your answer.

Answer:

The **Post correspondence problem** is an undecidable decision problem that was introduced by Emil Post in 1946.[1] Because it is simpler than the halting problem and the *Entscheidungsproblem* it is often used in proofs of undecidability.

Definition of the problem — **4 Marks**

The input of the problem consists of two finite lists $\alpha_1, \ldots, \alpha_N$ and $\beta_1, \ldots, \beta_N$ of words over some alphabet A having at least two symbols. A solution to this problem is a sequence of indices $(i_k)_{1 \le k \le K}$ with $K \ge 1$ and $1 \le i_k \le N$ for all k, such that

$$\alpha_{i_1} \ldots \alpha_{i_K} = \beta_{i_1} \ldots \beta_{i_K}.$$

The decision problem then is to decide whether such a solution exists or not.

Eg. Consider the following two lists:

α_1	α_2	α_3	β_1	β_2	β_3
a	*ab*	*bba*	*baa*	*aa*	*bb*

A solution to this problem would be the sequence (3, 2, 3, 1), because

$$\alpha_3\alpha_2\alpha_3\alpha_1 = bba + ab + bba + a = bbaabbbaa = bb + aa + bb + baa = \beta_3\beta_2\beta_3\beta_1.$$

Furthermore, since (3, 2, 3, 1) is a solution, so are all of its "repetitions", such as (3, 2, 3, 1, 3, 2, 3, 1), etc.; that is, when a solution exists, there are infinitely many solutions of this repetitive kind.

$W_1 = 11$, $W_2=100$, $W_3= 111$

$V_1= 111$, $V_2 =001$, $V_3 = 11$.

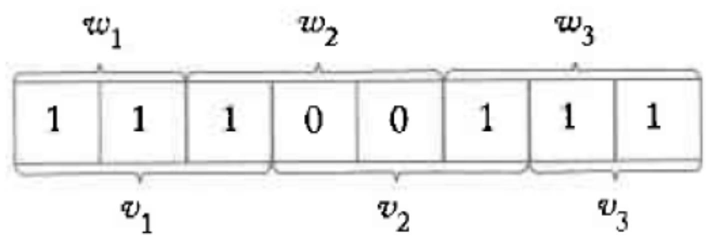
For this case, there exists a **PC-solution as Figure 12.7** shows.
If we take

$W_1 = 00$, $W_2=001$, $W_3= 1000$

$V_1= 0$, $V_2 =11$, $V_3 = 011$

NO, there cannot, be any PC-solution simply because any string composed of elements of A will be longer than the corresponding string from B.

Figure 12.7



4 Marks

TEXT BOOK

I. Introduction to Automata Theory, Languages and Computation, John E Hopcroft, Rajeev Motwani, Jeffery D. Ullman, Pearson Education, Third Edition, 2006