Tribhuvan University
Institute of Science and Technology
2078

✡

Bachelor Level / Second Year/ Forth Semester/ Science
Computer Science and Information Technology (CSC 257)
(Theory of Computation)
NEW COURSE

Full Marks: 60
Pass Marks: 24
Time: 3 hours.

Candidates are required to give their answers in their own words as for as practicable.
All figures in the margin indicate full marks.

Section A

Long Answer Questions
Attempt any Two questions **(2x 10=20)**

1. Give the formal definition of DFA and NFA. How NFA can be converted into equivalent DFA? Explain with suitable example. (4+6)

2. Find the minimum state DFA for the given DFA below (10)

States	Inputs	
	0	1
→A	B	F
B	E	C
C	B	D
*D	E	F
E	B	C
F	B	A

3. Construct a Turing Machine that accepts the language of odd length strings over alphabet {a,b}. Give the complete encoding for this TM as well as its input string w=abb in binary alphabet that is recognized by Universal Turing Machine. (2+6+2)

Section B

Short Answer Questions
Attempt any Eight questions. **(8x5=40)**

4. Define the term alphabet, prefix and suffix of string, concatenation and Kleen closure with example. (5)

5. Give the regular expressions for the following language over alphabet {a,b}. (2.5 + 2.5)
   a. Set of all strings with substring bab or abb
   b. Set of all strings whose 3rd symbol is 'a' and 5th symbol is 'b'.

6. Show that L={ a^n | n is a prime number} is not a regular language. (5)

7. Explain about the Chomsky's Hierarchy about the language and grammars. (5)

8. Define a Push Down Automata. Construct a PDA that accept L = {$a^n b^n$ | n>=0}. (1+4)

9. Convert the following grammar into Chomsky Normal Form. (5)

$S \rightarrow abSb \mid a \mid aAb$

$A \rightarrow bS \mid aAAb \mid \varepsilon$

10. Define Turing Machine and explain its different variations. (2+3)

11. What do you mean by computational Complexity? Explain about the time and space complexity of a Turing machine. (1+4)

12. Explain the term Intractability. Is SAT problem is intractable? Justify. (1+4)

Theory of computation - 2078
-Ankit Parajuli

DATE

Section A

Q. Give the formal definition of DFA and NFA? How NFA can be converted into equivalent DFA? Explain with suitable example.

⇒ A DFA is a finite automation consisting of five tuples:- $(Q, \Sigma, \delta, q_0, F)$ where,

Q = finite set of state

Σ = set of input symbols $\quad \delta: Q \times \Sigma \to Q$

q_0 = Initial state

F = set of Final States $F \subseteq Q$.

An NFA is a finite automation consisting of five tuples:- $(Q, \Sigma, \delta, q_0, F)$ where,

Q = finite set of states

Σ = set of input symbols

δ = Transition function: $\delta: Q \times \Sigma \to 2^{|Q|}$

q_0 = Initial state

F = set of final states $F \subseteq Q$

Note: NFA doesn't contain any dead state and multiple transition on single input is possible.

⇒ The steps for converting NFA to DFA are:-

Let $M = (Q, \Sigma, \delta, q_0, F)$ is an NFA which accepts the language $L(M)$. There should be equivalent DFA denoted by $M' = (Q', \Sigma', q_0', \delta', F')$ such that $L(M) = L(M')$

Step 1: Initially $Q' = \phi$

Step 2:- Add q_0 of NFA to Q'. Then find the transitions from this start state

classmate

PAGE

DATE

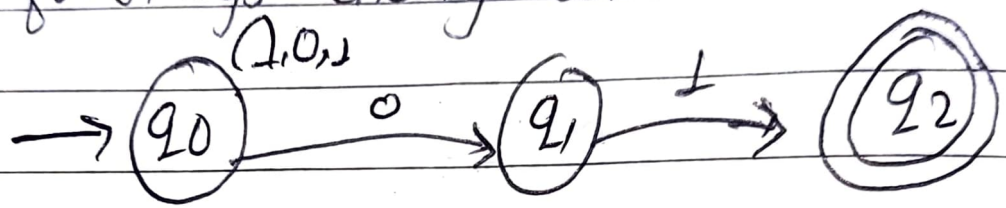
Step-3: In Q', find the possible set of states for each input symbol. If this set of states is not in Q', then add it to Q'.

Step-4: In DFA, the final state will be all the states which contain F (final states of NFA)

⇒ Given, $\Sigma = \{0,1\}$

Let's convert the NFA to DFA

NFA for strings ending with 01 i's



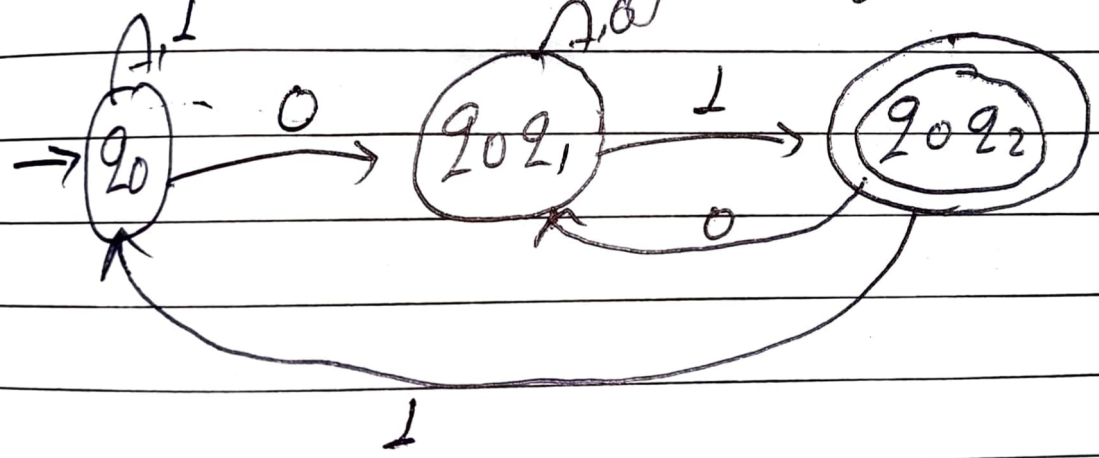
Transition table is:

δ	0	1
$\rightarrow q_0$	q_0, q_1	q_0
q_1	ϕ	q_2
$* q_2$	ϕ	ϕ

Now, let's make transition table for equivalent DFA

δ	0	1
$\rightarrow q_0$	q_0q_1	q_0
q_0q_1	q_0q_1	q_0q_2
$*q_0q_2$	q_0q_1	q_0

Let's draw state diagram for DFA



DATE

Q2. Find the minimum state DFA for the given DFA below.

States	Inputs 0	1
→A	B	F
B	E	C
C	B	D
*D	E	F
E	B	C
F	B	A

⇒ Based on the given transition table, let's make the equivalence class of accepting and non-accepting states.

0-Equivalent (π_0) = {ABCEF} {D}

1-Equivalent (π_1) = {ABE} {C} {D}

2-Equivalent (π_2) = {AF} {BE} {C} {D}

3-Equivalent (π_3) = {AF} {BE} {C} {D}

Since, π_2 and π_3 have same set of states. So, we stop here.

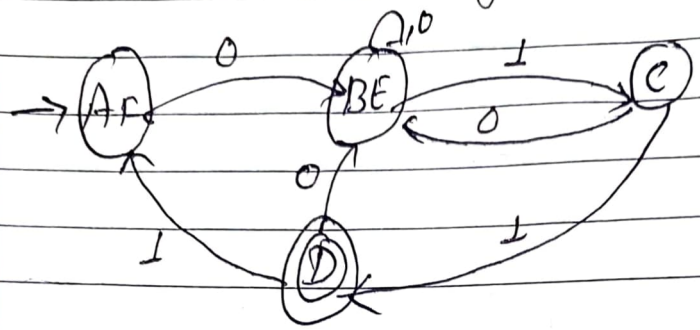
Now, let's make transition table for minimum state DFA

States	Inputs 0	1
→AF	B	AF
BE	BE	C
C	BE	D
*D	BE	AF

PAGE

DATE

It's transition state diagram :-

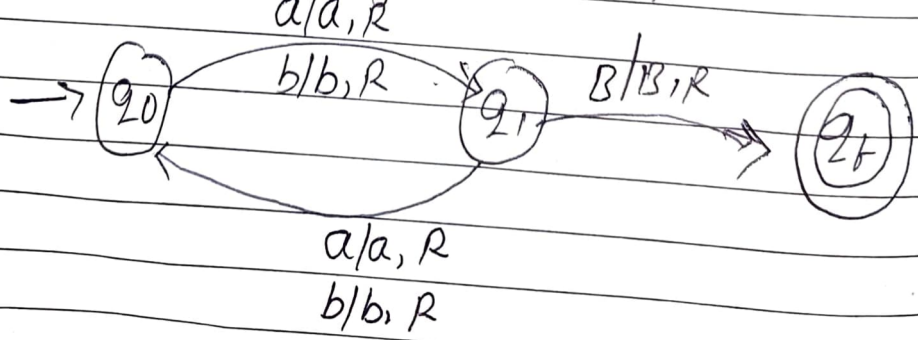


Hence, we minimized the given DFA from 6 states to 4 states.

Q.3. Construct a Turing Machine that accepts the language of odd length strings over alphabet $\{a, b\}$. Give a complete encoding for this TM as well as its input string w=abb in binary alphabet that is recognized by Universal Turing Machine.

⇒ Given, $\Sigma = \{a, b\}$

Turing Machine that accepts the language of odd length strings over $\{a, b\}$ is



classmate

PAGE

DATE

∴ Turing Machine $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ is.

$(\{q_0, q_1, q_f\}, \{a, b\}, \{a, b, B\}, \delta, q_0, B, q_f)$

Where, δ is defined as

$m_1 = \delta(q_0, a) = (q_1, a, R)$

$m_2 = \delta(q_0, b) = (q_1, b, R)$

$m_3 = \delta(q_1, a) = (q_0, a, R)$

$m_4 = \delta(q_1, b) = (q_0, b, R)$

$m_5 = \delta(q_1, B) = (q_f, B, R)$

Let the encoding structure be

$e(q_0) = 1$ $\quad e(a) = 1$ $\quad e(L) = 1$

$e(q_1) = 11$ $\quad e(b) = 11$ $\quad e(R) = 11$

$e(q_f) = 111$ $\quad e(B) = 111$

Now, let '0' be the delimiter for elements within the tuples, & '00' be the delimiter between the tuples of the turing machine.

So, $e(m_1) = 1010110101011$

$e(m_2) = 10110110110111$

$e(m_3) = 11010101011$

$e(m_4) = 1101101011011011$

$e(m_5) = 1101101110110111$

Also, Encoding of input string $w = abb$ is

$e(w) = 1011011$

So, the final description of universal turing machine

$TM = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ is

$\langle TM, w \rangle = \langle$ 10110 111 00 101100 1011 0111 00

1010 11010110 1011011 011011 0 1101 010101 10

11011010110 11 0 11011 0111 011011 00 1 00 11100

111, 1011011 $\rangle$

Section B DATE

Q.4. Define the term alphabet, prefix and suffix of string, concatenation and kleen closure with example

Alphabet: In automata theory, alphabet is a finite, non-empty set of symbols. It is denoted by Σ (Sigma). Ex: $\Sigma = \{0, 1\}$ is a binary alphabet

Prefix of a string: A string 's' is called a prefix of a string 'w' if it is obtained by removing zero or more trailing symbols of w.

Ex: w = ankit

s = ank is a prefix of w.

Suffix of a string: A string 's' is called a suffix of another string 'w' if it is obtained by removing zero or more leading symbols in w.

Ex: w = ankit

s = kit is suffix of w.

Concatenation: If x and y are two strings over an alphabet, the concatenation of x and y is written by xy and consists of symbols of x followed by those of y.

Ex: x = ankit, y = pangeni

Then xy = ankitpangeni

Kleen closure (Σ^*): The set of all the strings of any length over an alphabet Σ is called kleen closure of Σ & it is denoted by Σ^*.

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cdots$

Ex: $A = \{0\}$, then $A^* = \{0^n ; n = 0, 1, 2, \cdots\}$

$A^* = \{\in, 0, 00, 000, \cdots\}$

classmate PAGE

DATE

Q5. Give the regular expressions for the following language over alphabet $\{a,b\}$

a. Set of all strings with substring bab or abb
b. Set of all strings whose 3rd symbol is 'a' and 5th symbol is 'b'.

⇒ Given $\Sigma = \{a,b\}$

a ⇒ Regular expression for set of strings with substring bab or abb is.

$$(a+b)^* bab\ (a+b)^* + (a+b)^* abb (a+b)^*$$

b ⇒ Regular Expression for set of strings whose 3rd symbol is 'a' & 5th symbol is 'b' is

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

Q6. Show that $L = \{a^n \mid n$ is a prime number$\}$ is not a regular language.

⇒ Lets use pumping lemma to prove the language is not regular

DATE

- Suppose that L is regular.
- Let pumping length = P exists such that any string $s \in L$ $S = a^P$
- Let's divide S into xyz
  1st, let's suppose $P = 3$ (prime number)
  Then, $S = a^P = a^3 = aaa$.

Now, let's see the cases in which we can divide S into three parts xyz

Case : a a a
x y z

$x = \{a\}$
$y = \{a\}$
$z = \{a\}$

- Now, let's show that $xy^iz \notin L$ for some i
  let, $i = 2$, then xy^2z is

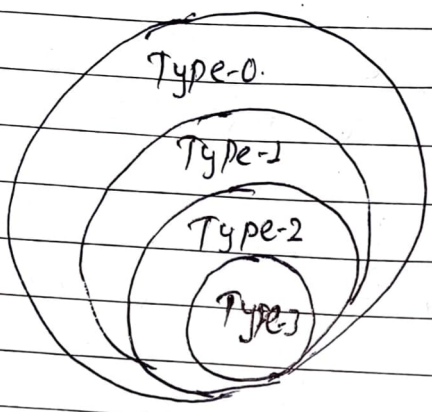
xy^2z: aaaa $\notin L$ since, a^4 and 4 is not a prime number.

So, by contradiction, L is not regular and hence it can't be pumped.

DATE

Q.7. Explain about Chomsky's Hierarchy about the language and grammars

→ Chomsky hierarchy represents the class of languages that are accepted by the different machines. According to Chomsky hierarchy, grammar is divided into 4 types as follows



- Type 0 is known as unrestricted grammar. They are recognized by Turing Machine. It is used to generate recursive enumerable language.
- Type 1 is known as Context sensitive grammar It is recognized by Linear Bounded Automata All type 1 grammar should be Type 0.
- Type 2 is known as context free grammar It is recognized by Push Down Automata. All type 2 grammar should be Type 0 & Type 1
- Type 3 is known as Regular grammar. It is recognized by Finite automata. It is the most restricted form of grammar.

classmate

PAGE

DATE

Here, the restriction of grammar increases from going to type 0 from type 3. The more restriction increases, the more easier the grammar's language becomes & requires less powerful machine.

Q. Define a Push down Automata. Construct a PDA that accept $L = \{a^n b^n \mid n \geq 0\}$

→ Push down automata is a machine that accepts the languages formed by type-2 grammar i.e. Context free grammar.

Formally, a PDA is defined by seven tuples $(Q, \Sigma, \delta, q_0, F, \Gamma, Z_0)$ where Q = Finite set of state

q_0 = Start state

Σ = Input symbols

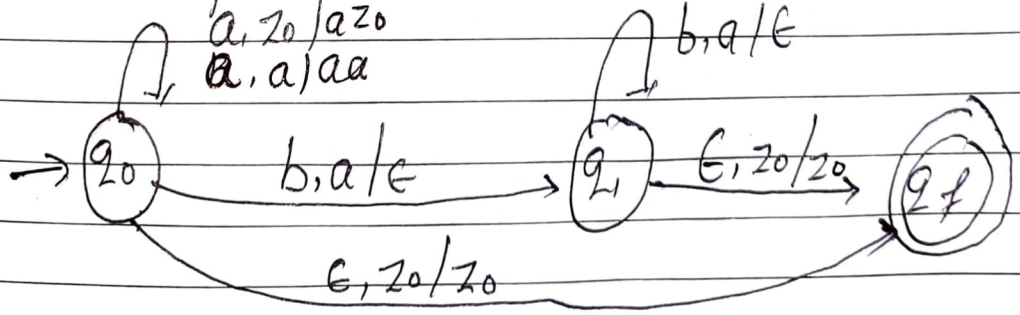
Γ = Stack alphabet (push into stack)

Z_0 = Stack start symbol.

F = Set of Final state

δ = Transition function, $\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma \to Q \times \Gamma^*$

PDA that accept $L = \{a^n b^n \mid n \geq 0\}$



Hence, PDA is: $(\{q_0, q_1, q_f\}, \{a, b\}, \delta, \{q_0\}, \{q_f\}, \{a, b, Z_0\}, Z_0)$

classmate

PAGE

DATE

where, δ is defined as:

$\delta(q_0, a, z_0) = (q_0, a z_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, \epsilon)$
$\delta(q_1, b, a) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, z_0) = (q_f, z_0)$
$\delta(q_0, \epsilon, z_0) = (q_f, z_0)$

Q.9. Convert the following grammar into Chomsky Normal Form

$S \rightarrow abSb \mid a \mid aAb$
$A \rightarrow bS \mid aAAb \mid \epsilon$

⇒ Since, the start symbol S is in right side, so add new start symbol

$B \rightarrow S$
$S \rightarrow abSb \mid a \mid aAb \mid ab$
$A \rightarrow bS \mid aAAb \mid \epsilon$

Remove new productions $(A \rightarrow \epsilon)$

$B \rightarrow S$
$S \rightarrow abSb \mid a \mid aAb \mid ab$
$A \rightarrow bS \mid aAAb \mid aAb \mid ab$

Remove unit productions $(B \rightarrow S)$

$B \rightarrow abSb \mid a \mid aAb \mid ab$
$S \rightarrow abSb \mid a \mid aAb \mid ab$
$A \rightarrow bS \mid aAAb \mid aAb \mid ab$

PAGE

DATE

Split the rules into parts.

$B \to aD \mid a \mid aE \mid ab$
$S \to aD \mid a \mid aE \mid ab$
$A \to bS \mid aF \mid aE \mid ab$
$C \to Sb$
$D \to bC$
$E \to Ab$
$F \to AE$

Replace terminal symbols with variables

$B \to GD \mid a \mid GE \mid GH$
$S \to GD \mid a \mid GE \mid GH$
$A \to HS \mid GF \mid GE \mid GH$
$C \to SH$
$D \to HC$
$E \to AH$
$F \to AE$
$G \to a$
$H \to b$

Q.10) Define Turing Machine & explain its variations

⇒ A turing machine is a finite automaton with a two way access to an infinite tape. Formally, a Turing Machine TM is defined by 7-tuple $M = (Q, \Sigma, \delta, q_0, F, \Gamma, \beta)$

The variations of TM are

i) Multitape TM
ii) Multitrack TM
iii) Non-deterministic TM

Extra from book

PAGE

DATE

Q-11. What do you mean by computational complexity? Explain about time & space complexity of a Turing machine.

⇒ Computational complexity is a measure of the amount of computing resources (time & space) that a particular algorithm consumes when it runs. The complexity of a computational problem can be measured by choosing a specific abstract machine as a model of computation & considering how much resource machine of that type requires for the solution of that problem.

Time and space Complexity of Turing Machine

The time and space complexity of a TM can be measured in terms of number of moves and the number of tape squares required for computing an instance of a problem. The most obvious measure of the size of any instance is the length of input string. The worst case is considered as the max time or space required by any string of that length.

CLASSMATE

PAGE

The time & space complexity of a TM is defined as:-

- Let T be a TM, the time complexity of T is the function T_t defined on the natural numbers as: for $n \in N$, $T_t(n)$ is the maximum no. of moves T can make on any input string of length n. If there is an input string x such that $|x| = n$, then T loops forever on input T, $T_t(n)$ is undefined.

- The space complexity of T is the function S_t defined as $S_t(n)$ is the maximum number of the tape squares used by T for any input string of length n. If T is multi-tape, no. of tape squares means maximum of the no. of individual tapes. If for some input of length n, it causes T to loop forever, $S_t(n)$ is undefined.

DATE

Q.12. Explain the term intractibility Is SAT problem intractibe? Justify

→ Intractability is a technique for solving problems which are not solvable in polynomial time. The problems that can be solved within reasonable time & space constraints are called tractable. The problems that can't be solved in polynomial time but require exponential time algorithm are called intractable or hard problems

If you prove SAT as ~~np-hard and~~ np-complete then it is a np-class problem. And NP class problem is intractible problem

So, let us prove SAT is NP-complete :-

Proof: To show that SAT is NP-complete we have to show properties 1st is SAT is NP & 2nd is SAT is NP hard. To show, SAT is intractable, we just show SAT is NP.

SAT problem is the question "Given a Boolean combinational circuit, is it satisfiable?". Given the circuit satisfiability problem, take a circuit x & a certificate y with the set of values that produce output 1, we can verify that whether the given certificate satisfies the circuit in polynomial time. So, we can say that circuit satisfability problem is NP So, it is intractible.

classmate

PAGE