Tribhuvan University

**Institute of Science and Technology**

2076

✡

Bachelor Level / Second Year/ Forth Semester/ Science — Full Marks: 60

**Computer Science and Information Technology (CSC 257)** — Pass Marks: 24

(Theory of Computation) — Time: 3 hours.

New Course

*Candidates are required to give their answers in their own words as for as practicable.*

All figures in the margin indicate full marks.

**Attempt all the questions.**

**Section A**

**Long Answer Questions**

Attempt any Two questions (2x 10=20)

1. Define the NFA with ε-transitions and ε-closure of a state. Show that for every regular expression r, representating a language L, there is ε-NFA accepting the same language. Also convert regular expression (a+b)*ab* into equivalent Finite Automata. (2+6+2)

2. How can you define the language acceted by a PDA? Explain how a PDA accepting language by empty stack is converted into an equivalent PDA accepting by final state and vice-versa. (2+4+4)

3. Define a Turing machine. Construct a TM that accept $L = \{ wcw^R \mid w \in \{0,1\}$ and c is ε or 0 or 1. Show that string 0110 is accepted by this TM with sequence of Instantaneous Description (ID). (2+6+2)

**Section B**

**Short Answer Questions**

Attempt any Eight questions. (8x5=40)

4. Give the formal definition of DFA. Construct a DFA accepting all strings of {0,1} with even number of 0s and even number of 1s. (2+3)

5. Define Chomsky Normal Form and Greibach Normal Form in reference to CFG. Give a suitable example of each. (2.5+2.5)

6. Give the regular expressions for following language over alphabet {0,1} (2.5+2.5)

a. Set of all strings with 2nd symbol from right is 1.

b. Set of all strings starting with 00 or 11 and ending with 10 or 01.

7. Show that language $L = \{ 0^m1^m \mid m>=1 \}$ is not a regular language. (5)

8. Describe the Turing machines with multiple tape, multiple track and storage in state. (5)

9. Construct a NFA accepting language of {0,1} with each string ending with 01 and convert it into equivalent DFA. (2+3)

10. Construct a PDA accepting language over { 0,1} representing strings with equal no of 0s and 1s. Show by sequence of IDs that 0101 is accepted by this PDA. (3+2)

11. Define complexity of a Turing machine. Explain about big Oh, big Omega and big Theta notation used for complexity measurement. (1+4)

12. What do you mean by tractable and intractable problems? Explain with reference to TM. (5)

Theory of Computation - 2076

- Ankit Pangeni

DATE

Section 'A'

1. Define the NFA with ε-transition and ε-closure of a state. Show that for every regular expression R representing a language L, there is ε-NFA accepting the same language. Also convert regular expression $(a+b)^* ab^*$ into equivalent Finite Automata

=> The NFA with ε-transition is called an epsilon NFA which is defined as a mathematical model that consists of 5 tuples:

$E = (Q, \Sigma, \delta, q_0, F)$ where,

$Q$: Finite set of state

$\Sigma$: Finite alphabet

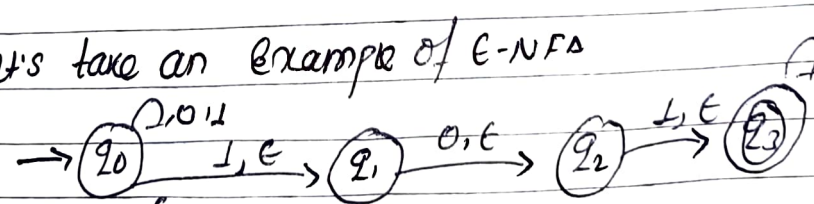
$q_0$: Initial state, $q_0 \in Q$

$F$: Set of final states $F \subseteq Q$

$\delta$: Transition function defined as: $\delta: Q \times \Sigma \cup \{\epsilon\} \to 2^Q$

In case of Epsilon NFA, ε-closure of a state is defined as all the states reachable from a state q on reading Epsilon (ε) only is called Epsilon closure of the state q.

Ex: Let's take an example of ε-NFA




ε-close(q0) = {q0, q1, q2}

ε-close(q1) = {q1, q2}

ε-close(q2) = {q2, q3}

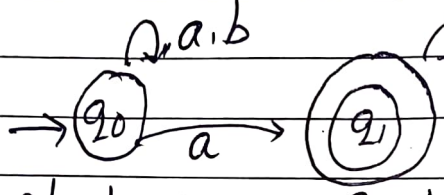
ε-close(q3) = {q3}

DATE

Given Regular Expression

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

Let's Convert it to ε-NFA



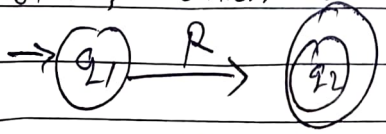

which is a finite automata NFA

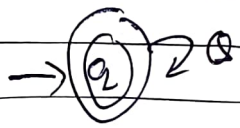
* To prove, every regular expression R, representing a language L, there is ε-NFA accepting the same language.

Let's assume $L = L(R)$ for a regular expression R. We prove that $L = L(M)$ for some ε-NFA M with

- Exactly one accepting state
- No incoming edges at the initial state
- No outgoing edges at the accepting state.

Step-1:- Create a starting state, say $q_1$, and a final state, say $q_2$. Label the transition $q_1$ to $q_2$ as the given regular expression, R, as in fig 1. But, if R is $(S)^*$, then create a single initial state, which will also be the final state



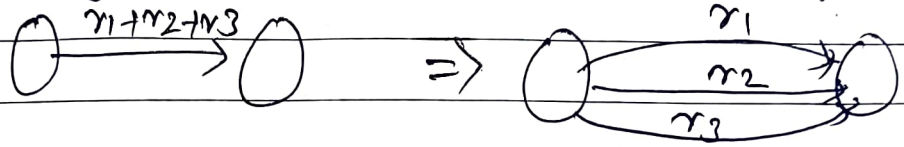

classmate

PAGE

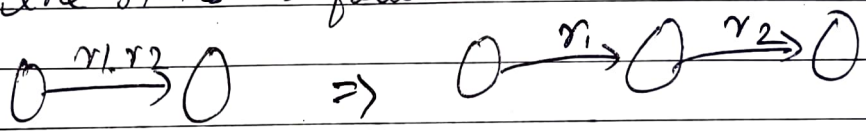
DATE

Step-2: Repeat the following rules by considering the least precedency regular expression operator 1st until no operator is left in the expression.
Precedence of operators in regular expressions is defined as Union < Concatenation < Kleene's Closure

Union operator (+) can be eliminated by introducing parallel edges between the two states as follows

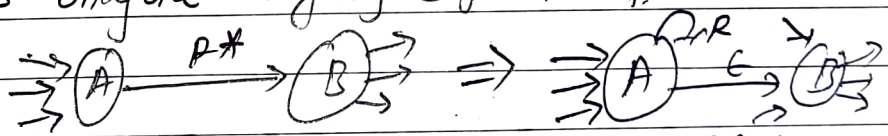



The concatenation operator (.) (or no operator at all) can be eliminated by introducing a new state between the states as follows

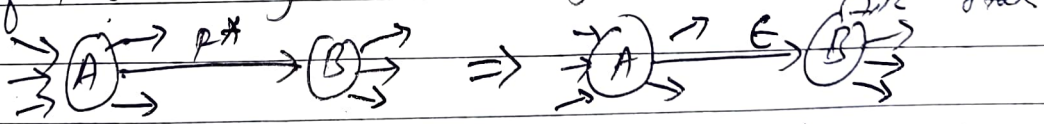



Kleene's Closure(*) can be eliminated by introducing self-loop

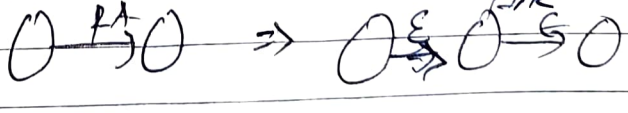
1) If there's only one outgoing edge at left most state




2) If there's only one incoming edge at right most state




3) Else introduce a new state between two states having self-loop




Ex: Given language : $\Sigma = \{a, b\}$ with strings of length at most 2

R.E = (a+b) (a+b) (a+b)*

And corresponding ENFA is:

DATE

Q.2. How can you define the language accepted by PDA? Explain how a PDA accepting language by empty stack is converted into an equivalent PDA accepting by final state & vice-versa.

⇒ Push down automata is a finite automata that accepts the context free languages. PDA can accept the strings of language by two ways:

Acceptance by final state:

Given a PDA 'P', the language accepted by final state, L(P) is $\{w \mid (q, w, z_0) \vdash^* (p, \epsilon, \gamma)\}$ where $p \in F$ and $\gamma \in \Gamma^*$

Acceptance by Empty stack

Given a PDA P, the language accepted by empty stack L(P) is $\{w \mid (q, w, z_0) \vdash^* (p, \epsilon, \epsilon)\}$ where $p \in Q$.

* Conversion of PDA accepting language by empty stack to PDA accepting by final stack

Let us consider the PDA $P_N$ that accepts a language L by empty stack and $P_F$ is the equivalent PDA that accepts L by final stack.

classmate

PAGE

DATE

The construction of $P_F$ from $P_N$ is done as follow.

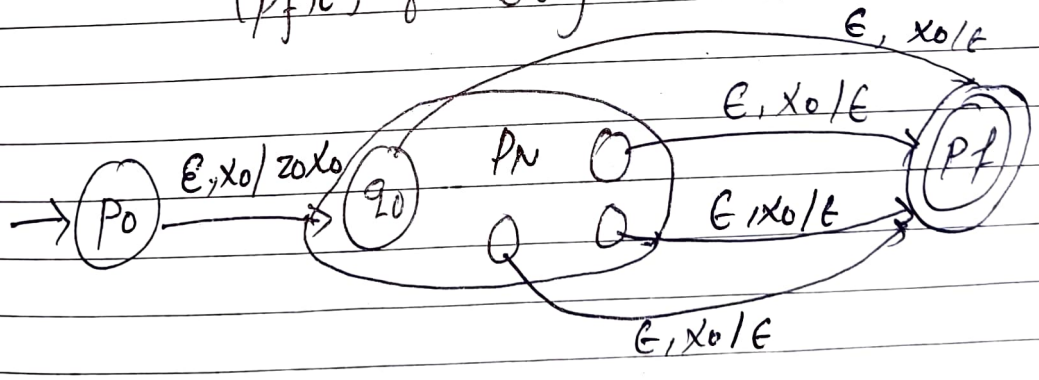
1) Introduce a new symbol $X_0$ ($X_0 \notin \Gamma$) & place it into the bottom of stack $P_F$
2) Introduce a new start state $P_0$ to push $Z_0$, the start symbol of $P_N$ to the top of stack & enter state $q_0$ (start state of $P_N$)
3) Copy the other states & transition functions of $P_N$ in $P_F$.
4) Finally, add another new state $P_f$, which is the accepting state of $P_F$.

Thus, the new constructed $P_F$ has the components

$P_N = \{ Q \cup \{P_0, P_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, P_0, X_0, \{P_F\} \}$

Where $\delta_F$ is defined by

1. $\delta_F(P_0, \epsilon, X_0) = (q_0, Z_0 X_0)$
2. $\forall$ state $q \in Q$, inputs $a$ in $\Sigma$ or $a = \epsilon$, & stack symbols $Y$ in $\Gamma$, $\delta_F(q, a, Y)$ contains all the pairs in $\delta_N(q, a, Y)$ i.e. Copy all the transition functions of $P_N$.
3. Add another transition function $\delta_F(q, \epsilon, X_0) = (P_f, \epsilon)$ for every state $q$ in $Q$




* Conversion of PDA accepting by final state to accepting by empty stack

The construction of $P_N$ from $P_F$ is done as follows:

DATE

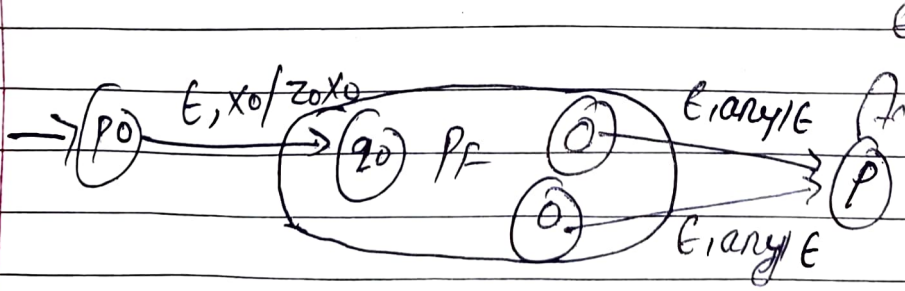
1. Introduce a new symbol $X_0$ ($X_0 \notin \Gamma$) and place it to the bottom of the stack of $P_N$.
2. Introduce a new start state $p_0$ to push $z_0$ (start symbol of $P_F$) to the top of the stack & enter state $q_0$ (start state of $P_F$)
3. For each accepting state of $P_F$, add a transition on $\epsilon$ to a new state $p$.

Thus, the new constructed $P_N$ has components

$$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$$

Where $\delta_N$ is defined by

1. $\delta_N(p_0, \epsilon, X_0) = (q_0, z_0 X_0)$
2. $\forall$ states $q \in Q$, inputs $a \in \Sigma$ or $a = \epsilon$, and stack symbol $Y \in \Gamma$, $\delta_N(q, a, Y)$ contains all the pairs in $\delta_F(q, a, Y)$ i.e. copy all the transition functions of $P_N$.
3. $\forall$ accepting state $q \in F$ & stack symbols $Y \in \Gamma$ or $Y = X_0$, $\delta_N(q, \epsilon, Y) = (p, \epsilon)$.
4. $\forall$ stack symbols $Y \in \Gamma$ or $Y = X_0$, $\delta_N(p, \epsilon, Y) = (p, \epsilon)$



DATE

Q.3. Define a Turing Machine. Construct a TM that accepts $L = \{ WCW^R \mid W \in \{0,1\}$ and C is $\in$ or 0 or 1 - Show that 0110 is accepted by this TM with sequence of Instantaneous Description (ID)

⇒ A Turing machine is a finite automation with a two-way access to an infinite tape. Formally, a TM is defined by seven tuples, $M = \{ Q, \Sigma, \delta, q_0, F, \Gamma, B \}$

$Q$ = Finite set of states
$\Sigma$ = Finite set of input symbols
$q_0$ = Start State $q_0 \in Q$
$F$ = Set of Final States $F \subseteq Q$
$\Gamma$ = Tape Symbols, $\Sigma$ is always subset of $\Gamma$
$B$ = Blank Symbol, $B \in \Gamma$ but $B \notin \Sigma$
$\delta$ = Transition function defined by

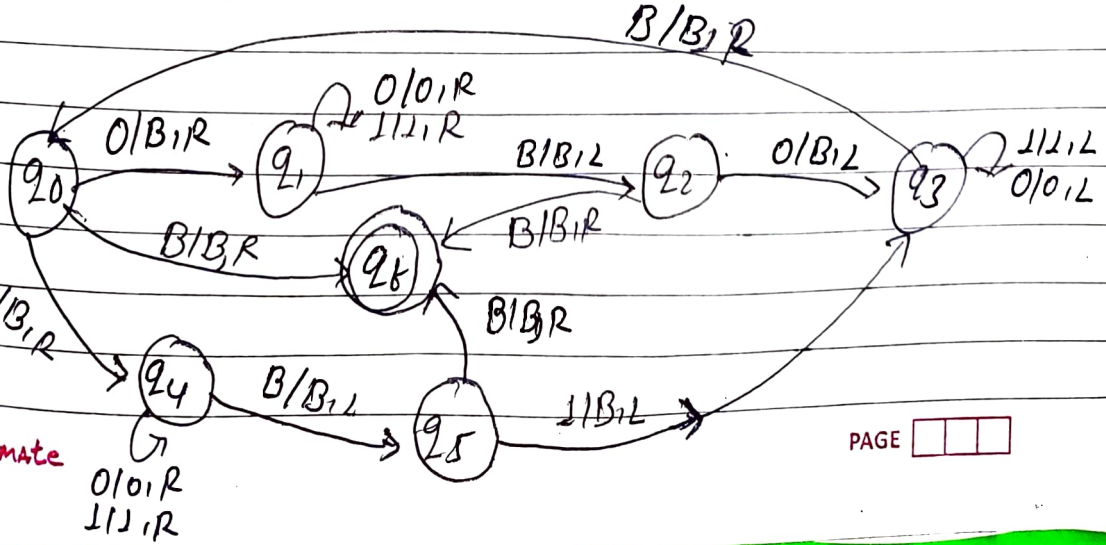
$$Q \times \Gamma \rightarrow Q \times \Gamma \times (R, L, S)$$

Where, R, L, S is the movement of head
ie. Right, Left or Stationary.

Given language $L = \{ WCW^R \mid W \in \{0,1\} \}$ and C is $\in$ or '0' or '1'

Let's take an Example String 0100010 where C is '0'

| B | B | 0 | 1 | 0 | 0 | 0 | 1 | 0 | B | B |
|---|---|---|---|---|---|---|---|---|---|---|




classmate

PAGE

DATE

Lets Check the acceptance of string 0110 with sequence of Instantaneous description.

$q_0$ 0110 $\vdash$ B $q_1$ 110
$\vdash$ B1 $q_1$ 10
$\vdash$ B11 $q_1$ 0
$\vdash$ B110 $q_1$ B
$\vdash$ B11 $q_2$ 0B
$\vdash$ B1 $q_3$ 1 BB
$\vdash$ B $q_3$ 11 BB
$\vdash$ $q_3$B 11 BB
$\vdash$ B $q_0$ 11 BB
$\vdash$ BB $q_4$ 1 BB
$\vdash$ BB 1 $q_4$ BB
$\vdash$ BB $q_5$ 1 BB
$\vdash$ B $q_3$ BBB
$\vdash$ BB $q_0$ BB
$\vdash$ BBB $q_f$ B

Halt and accept.

Hence 0110 is accepted by the given Turing machine.

## Section B:-

DATE

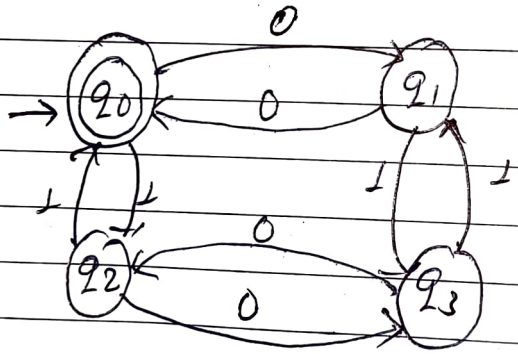
Q4. Give the formal definition of DFA. Construct a DFA accepting all strings of $\{0,1\}$ with even number of 0's and even number of 1's

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

$Q$ = Finite set of states
$\Sigma$ = Set of input symbols (alphabet)
$\delta$ = Transition function $\delta: Q \times \Sigma \to Q$
$q_0$ = Initial state
$F$ = Set of Final states $F \subseteq Q$

Given, $\Sigma = \{0,1\}$
The DFA for accepting all strings with even number of 0's and even number of 1's.

$L = \{\epsilon, 00, 11, 0011, 0101, 0110, \ldots\}$




Where, $Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0,1\}$
$q_0 = \{q_0\}$
$F = \{q_0\}$

$\delta$ is defined as

$\delta(q_0,0) = q_1$
$\delta(q_0,1) = q_2$
$\delta(q_1,0) = q_0$
$\delta(q_1,1) = q_3$
$\delta(q_2,0) = q_3$
$\delta(q_2,1) = q_0$
$\delta(q_3,0) = q_2$
$\delta(q_3,1) = q_1$



PAGE

DATE

Q.5. Define Chomsky Normal Form and Greibach Normal Form in reference to CFG. Give a suitable example of each.

⇒ A Context Free Grammar (CFG) is in Chomsky Normal Form (CNF) if all production rules satisfy one of the following rules:

$A \to \epsilon$
$A \to a$
$A \to BC$

Where, A, B, C are non-terminal symbols and a is a terminal symbol.

Thus, a grammar in CNF is one which shouldn't have $\epsilon$-production, unit production & useless symbols.

Example: $S \to AB$
$S \to C$
$A \to a$
$B \to b$

Where, S, A, B are non-terminal
a, b, C are terminal symbols

A Context Free Grammar (CFG) is in Greibach Normal Form (GNF) if all its production rules are in the form.

$A \to aV^*$
$A \to a$

Where, A is a non-terminal
a is a terminal and
V is a set of non terminals

Example: $X \to bABDE$
$X \to b$

Where, b is a terminal
X, AB, CDE are non-terminals

PAGE

DATE

Q.6. Give the regular expressions for following language over alphabet {0,1}

a. Set of all strings with 2nd symbol from right is 1
b. Set of all strings starting with 00 or 11 and ending with 10 or 01.

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

a) The regular expression for the set of strings with 2nd symbol from right is 1 is

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

$L = \{10, 11, 0011, 1010, 11111, 00001 0, \ldots\}$

b) The regular expression for the set of strings starting with 00 or 11 and ending with 10 or 01 is

$L = \{0010, 0001, 1110, 1101, 000010, 111101, \ldots\}$

$$(00+11)(0+1)^*(10+01)$$

DATE

Q.7. Show that Language $L = \{0^m 1^m \mid m \geq 1\}$ is not a regular language.

=) Let's prove using pumping lemma

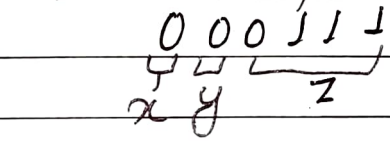
- Suppose L is a regular language.
- Let pumping length = p exists such that any string $s \in L$ is $S = 0^p 1^p$
- Let's divide S into xyz.

First, let's suppose $p = 3$

Then, $S = 0^3 1^3 = 000111$

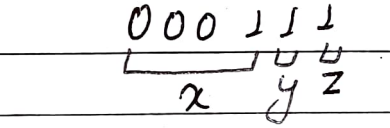
Now, let's see all possible cases in which we can divide S into three parts xyz

Case 1:- y is in '0' part



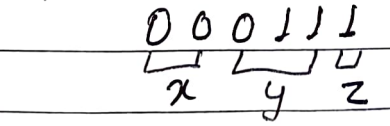

$x = \{0\}$
$y = \{0\}$
$z = \{0111\}$

Case 2:- y is in '1' part




$x = \{0001\}$
$y = \{1\}$
$z = \{1\}$

Case 3:- y is in '0' as well as '1' part




$x = \{00\}$
$y = \{011\}$
$z = \{1\}$

- Now, we show that $xy^iz \notin A$ for some $i$.

Let's take $i = 2$, then $xy^2z$ is

PAGE

classmate

DATE

Case 1:

$xy^2z$ : 000 0 111

Since, no. of 0s ≠ no. of 1's So, $xy^2z \notin L$

Case 2:

$xy^2z$ : 000 111 1

Since, no. of 0's ≠ no. of 1's So, $xy^2z \notin L$

Case 3:-

$xy^2z$ : 000110111 $\notin L$

Hence, it doesn't satisfy any cases. So, by contradiction, it is not a regular language

Q.8. Describe the Turing machine with multiple tape, multiple track and storage in state.

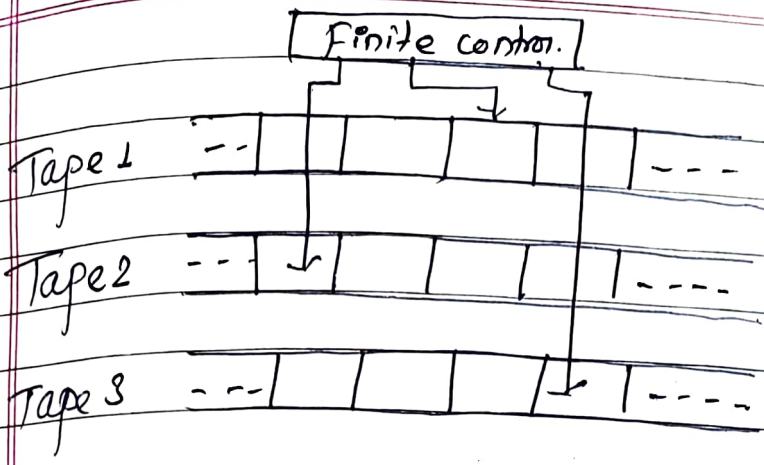
⇒ Turing Machine with multiple tape

A Turing machine can have more than one tape. Adding an extra tape adds no power to the computational model, but my the ability to accept the language is increased. A multiple tape TM consists of finite control and finite number of tapes. Each tape is divided into cell and each cell can hold any symbol of finite tape alphabets. The set of tape symbols include a blank & the input symbol.

CLASSMATE

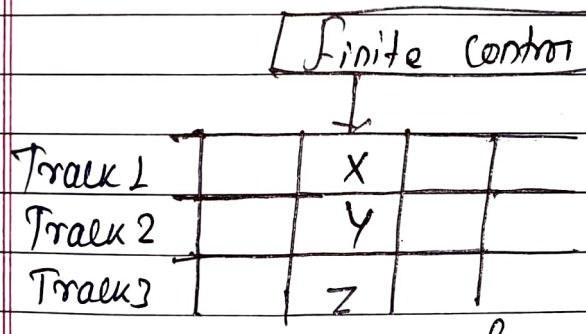
PAGE

DATE




## Turing Machine with Multiple tracks

The tape of turing machine can be considered as having multiple tracks. Each track can hold one symbol, and the tape alphabet of the TM consists of tuples with one component for each track. Following figure illustrates the TM with multiple track




The tape alphabet Γ is a set consisting of tuples like Γ = {(x,y,z)--}
The tape head moves up & down
Scanning symbols in tape at one positi

## Turing machine with storage in state

In turing machine, a state can be used to hold a finite amount of data. The finite control of machine consists of state q and some data portion In this case, a state is considered as a tuple. - (state, data)

PAGE

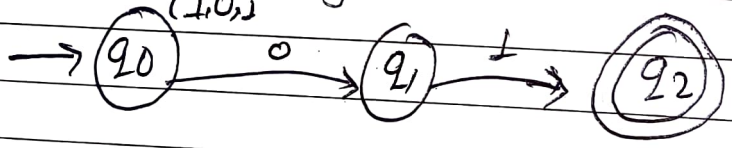
classmate

DATE

$\delta$ is defined by,
$\delta([q,A];x) = ([q_1, x], y, R)$ means $q$ is the state and data portion of $q$ is $A$. The symbol scanned on the tape is considered to be copied into the second component of the state & moves right entering state $q_1$ & replacing the symbol by $y$.

Q.9. Construct a NFA accepting language of $\{0,1\}$ with each string ending with 01 and convert it to equivalent DFA

⇒ Given, $\Sigma = \{0,1\}$
Let's convert the NFA to DFA
NFA for strings ending with 01 i's



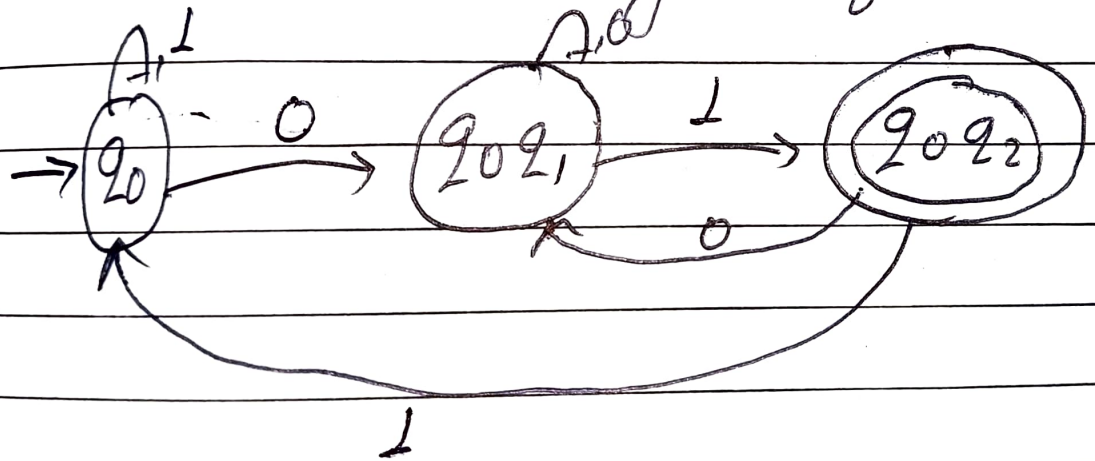

Transition table is:

| $\delta$ | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_0, q_1$ | $q_0$ |
| $q_1$ | $\phi$ | $q_2$ |
| $* q_2$ | $\phi$ | $\phi$ |

Now, let's make transition table for equivalent DFA

| $\delta$ | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_0 q_1$ | $q_0$ |
| $q_0 q_1$ | $q_0 q_1$ | $q_0 q_2$ |
| $*q_0 q_2$ | $q_0 q_1$ | $q_0$ |

Let's draw state diagram for DFA



DATE

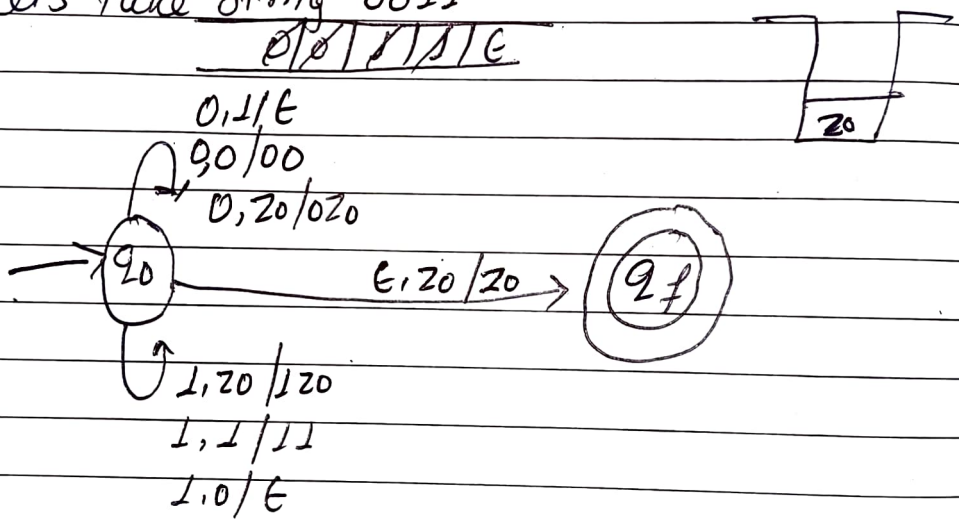
Q.10. Construct a PDA accepting language over $\Sigma\{0,1\}$ representing strings with equal no. of 0's and 1's. Show by sequence of IDs that 0101 is accepted by this PDA

⇒ Given, $\Sigma = \{0,1\}$

PDA accepting strings with equal no. of 0's & 1's

$L = \{ \epsilon, 0011, 0101, 1100, 10, 110010, \text{----} \}$

Let's take String 0011




Let's check the acceptance of 0101

$(q_0, 0101, Z_0) \vdash (q_0, 101, 0Z_0)$
$\vdash (q_0, 01, Z_0)$
$\vdash (q_0, 1, 0Z_0)$
$\vdash (q_0, \epsilon, Z_0)$
$\vdash (q_f, Z_0)$

which is accepted

Since $q_f$ is a final state.

classmate

PAGE

DATE

Q.11 Define Complexity of a Turing Machine. Explain about big oh, big Omega and big Theta notation used for complexity measurement.

=> For a Turing machine, the time complexity refers to the measure of the number of times the tape moves when the machine is initialized for some input symbols & the space complexity is the number of cells of the tape written

$T(n) = O(n \log n)$ & $S(n) = O(n)$

(You can write from Internet)

Q.12. What do you mean by tractable & intractable problems? Explain with reference to TM.

=> The problems that can be solved within reasonable time and space constraints are called tractable problems.

The problems that can't be solved in polynomial time but requires exponential time algorithm are called intractable or hard problems

Lets take the class P and class NP of problems solvable in polynomial time by deterministic and non-deterministic Turing machine. When the space and time required for implementing the steps of particular algorithm are reasonable, we can say that problem is tractable. The problems are intractable

PAGE

classmate

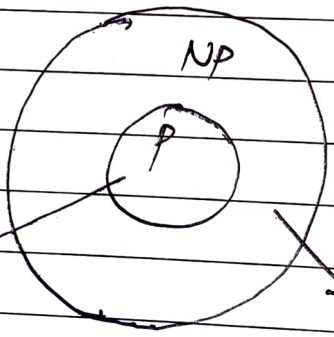
DATE

If the time required for any algorithm/problem is at least $f(n)$, where $f$ is an exponential function of $n$

Ex: P is the class of tractable problems ie. they can be implemented by a p-time TM & the complement of problem in P: is intractable.

is n prime? ⟷ (complement) Is n composite?
(tractable) (intractable)

Another Example: P class problem can be solved in polynomial time so it is tractable but NP-class can't be solved in polynomial time but can be verified in polynomial time Ex: Sudoku




$P \subseteq NP$

Tracable problem

Non tracable problem

classmate