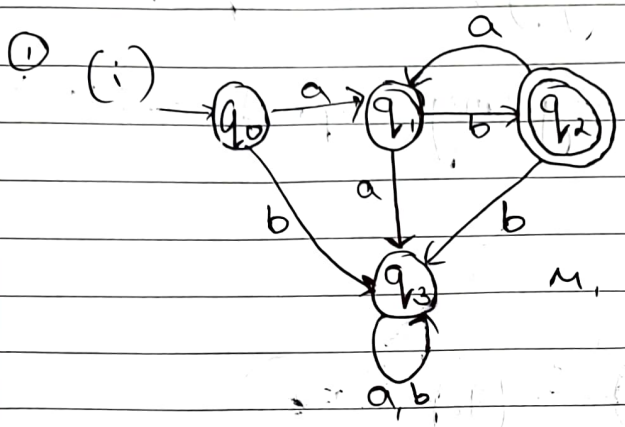
Pratik Singh - 2023201082 - Mtech CSE

Modern Complexity Theory
Assignment 1

1. (i)

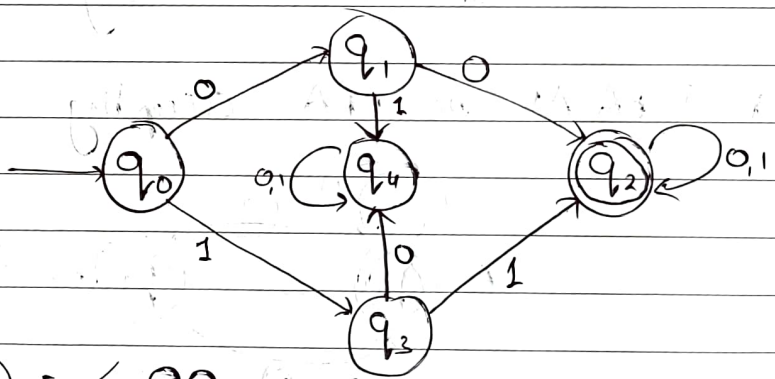


M_1

$L(M_1) = \{ab, abab, ababab \ldots ab\}$

$L(M_1) = \{w \mid$ where string w is an alternate sequence of 'ab' occuring more than or equal to once$\}$

(ii)

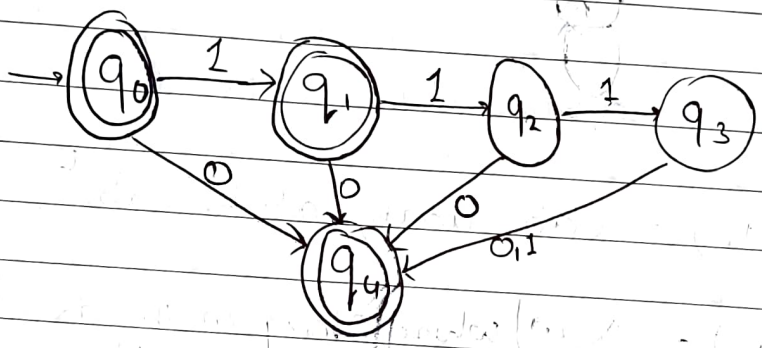


$L(M_1) = \{00, 11, 001\ldots, 000\ldots, 110\ldots, 111\ldots\}$

$L(M_1) = \{w \mid$ where w is a string starting with '00' or with '11'$\}$

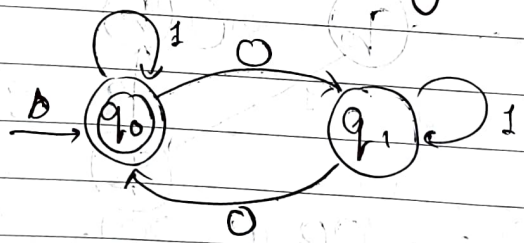
② Give state diagram of DFA's recognizing the following languages. The alphabet is $\{0,1\}$

ⓐ $\{w \mid w$ is any string except 11 and 111$\}$

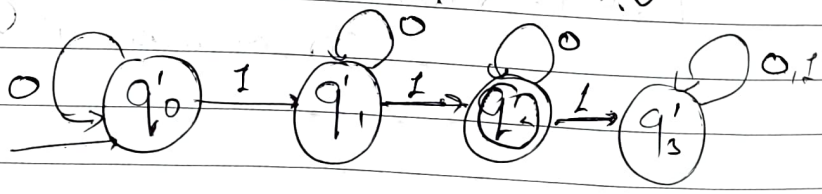


ⓑ $\{w \mid w$ is any string containing even no of 0's or contains exactly two 1's$\}$

Let DFA M_1 be DFA accepting even no. of 0's



Let DFA M_2 be DFA accepting exactly two 1's



taking union of M_1 and M_2, we get.

$M_1 = \{q_0, q_1\}$ $M_2 = \{q_0', q_1', q_2', q_3'\}$

$F = \{M_1 \times Q$

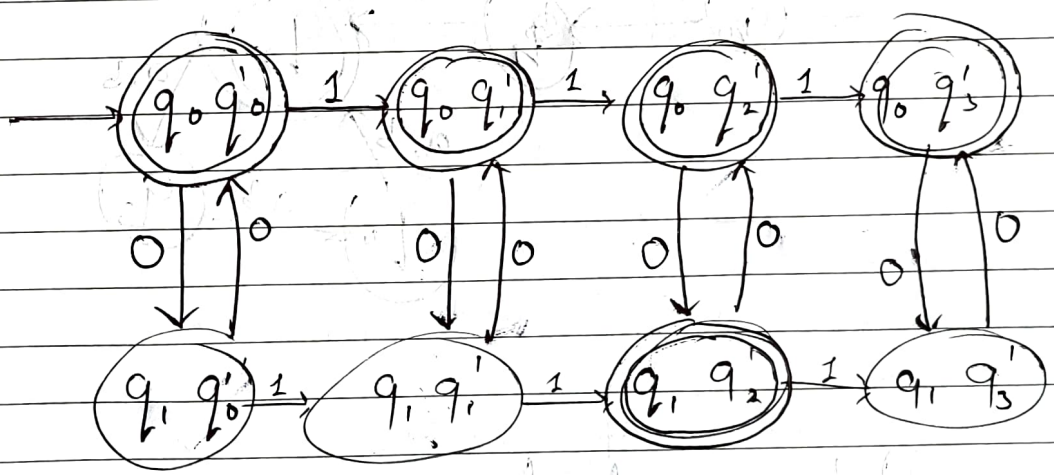
$F_1 = \{q_0\}$ $F_2 = \{q_2'\}$

$F = \{F_1 \times M_2\} \cup \{F_2 \times M_1\}$

$= \{\{q_0\} \times \{q_0', q_1', q_2', q_3'\}\} \cup \{\{q_2\} \times \{q_0, q_1\}$

$\Rightarrow$ Therefore our accepting states are F

$F = \{\{q_0 q_0', q_0 q_1', q_0 q_2', q_0 q_3'\} \cup \{q_2' q_0, q_2' q_1'\}$



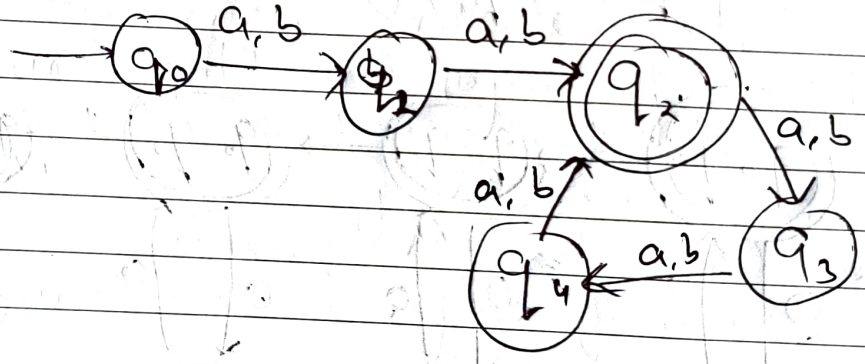
(3) (a) Consider the language L over {a, b} which contains the strings whose lengths are from the arithmetic progression.

$$P = \{2, 5, 8, 11, \ldots\} = \{2 + 3n \mid n \geq 0\}$$

that is

$$L = \{x \in \{a,b\}^* \mid |x| \in P\}$$

Construct a DFA recognising L



(b) In general for any arithmetic progression $P' = \{k' + kn \mid n \geq 0\}$ with $k, k' \in N$ if we consider the language $L' = \{x \in \Sigma^* \mid |x| \in P'\}$ over an alphabet Σ, then design a generalized dfa that recognizes the L'.

(4) (a) Prove that, if M is an NFA (Non deterministic Finite Automata) then $L(M) = \{w \mid w \text{ is accepted by } M\}$ is DFA-recognizable, where $L(M)$ is the language recognized by the NFA, M.

Sol^n → To prove that $L(M)$ is DFA recognizable, we will use the concept that for every NFA there exists an equivalent DFA that recognizes the same language $L(M)$.

(i) defining NFA.

NFA-M = $\{Q, \Sigma, \delta, q_0, F\}$ where:

Q is a finite set of states
Σ is a finite alphabet
δ is transition function $\delta : Q \times \Sigma \to 2^Q$
q_0 is start state $q_0 \in Q$
F is the set of accepting states $F \subseteq Q$

$L(M) = \{w \mid w \text{ is accepted by } M\}$

(ii) Constructing an equivalent DFA:

DFA $M' = (Q', \Sigma, \delta', q_0', F')$ that recognizes the same language $L(M)$.

→ $Q' = 2^Q$; the power set of Q. Each state of M' is a subset of M' is a subset of the states of M.

→ Σ remains the same

→ $\delta' : Q' \times \Sigma \to Q'$ is defined by $\delta'(S, a)$
$= \bigcup_{q \in S} \delta(q, a)$ for any subset $S \subseteq Q$

and input symbol $a \in \Sigma$.

→ $q_0' = \{q_0\}$ the set containing the start state of M.

∴ $F' = \{S \subseteq Q \mid S \cap F \neq \text{empty}\}$ i.e. all subsets of Q that contain at least one accepting state from M.

→ The DFA M' recognizes the same language as the NFA M, which means $L(M') = L(M)$. This is because DFA simulates all possible state transition of the NFA by keeping track of all possible states M could be in after reading each i/p symbol. Which proves that the language recognized by an NFA is also recognized by a DFA, hence proved.

(b). Prove that FA recognizable languages are closed under union.

Sol^n

FA → recognizable are regular languages

~~RTP: A~~

Let A_1 & A_2 be regular languages.

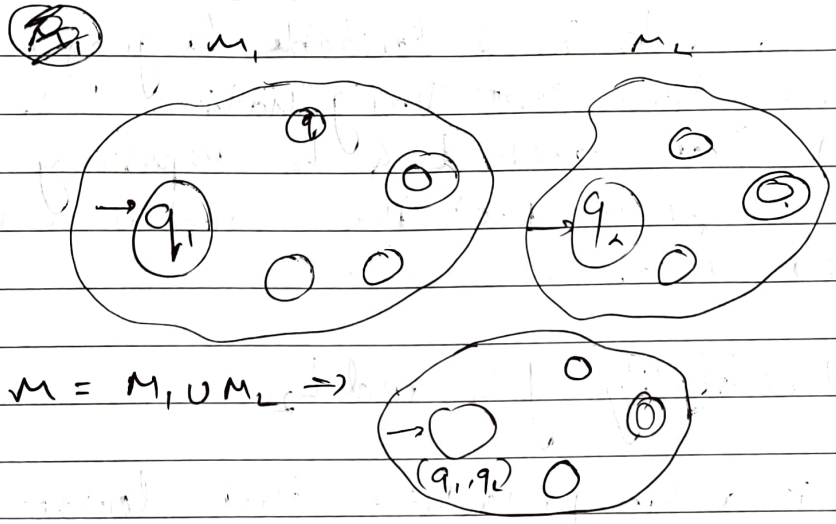
RTP: $A_1 \cup A_2$ is also regular.

Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ a DFA which recognizes A_1

Let $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ a DFA which recognizes A_2

goal is to construct a DFA M that recognizes $A_1 \cup A_2$

$M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A_1 \cup A_2$



① $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$

② Σ is same as that of M_1 & M_2 = Cartesian product of Q_1 & Q_2

③ $\delta : Q \times \Sigma \to Q$

$(Q_1 \times Q_2) \times \Sigma \to (Q_1, Q_2)$

For each $(r_1, r_2) \in Q$ $(= Q_1 \times Q_2)$ and each $a \in \Sigma$ let $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) = (r_1', r_2')$

④ Let $q_0 = (q_1, q_2)$ be the start state of M

⑤ $F = \{(r_1, r_2) \mid r_1 \in F \text{ or } r_2 \in F\}$

(5) Show that the collection of decidable languages is closed under concatenation.

Soln:

Let L_1 & L_2 be 2 decidable languages. By definition there exist Turing machine M_1 & M_2 that decide L_1 & L_2 respectively. We need to prove that the concatenation L_1L_2 is also decidable.

1. Decidability of L_1 and L_2

- M_1 is a Turing machine that decides L_1. This means M_1 halts and accepts if $w \in L_1$ and halts and rejects if $w \notin L_1$.
- M_2 is a Turing machine that decides L_2. This means M_2 halts and accepts if $w \in L_2$ and halts & rejects if $w \notin L_2$.

2. Concatenation of languages:

The concatenation L_1L_2 is defined as:

$$L_1L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$$

3. Construction of a Turing machine M to decide L_1L_2

We construct a Turing machine M that decides L_1L_2 as follows:

① Input $w \in \Sigma^*$

② Non deterministic splitting: Non deterministically

Split w into 2 parts $w = xy$ where $x \in \Sigma^*$ and $y \in \Sigma^*$

(3) Simulation on x : Simulate M_1 on x if M_1 rejects x then reject w.

(4) Simulation on y : Simulate M_2 on y : if M_2 rejects y reject w.

(5) Acceptance - If both M_1 & M_2 accepts x & y then accept w.

4. formation.

$$L(M) = \{ w \in \Sigma^* \mid \exists x, y \in \Sigma^* \text{ such that } w = xy, x \in L_1, y \in L_2 \}$$

$$L(M) = L_1 . L_2$$

Hence proved since M_1 & M_2 are deciders, M always halts $\therefore$ M is a decider for $L_1 L_2$ hence closed

(6) Prove that every nondeterministic Turing machine has an equivalent deterministic Turing machine

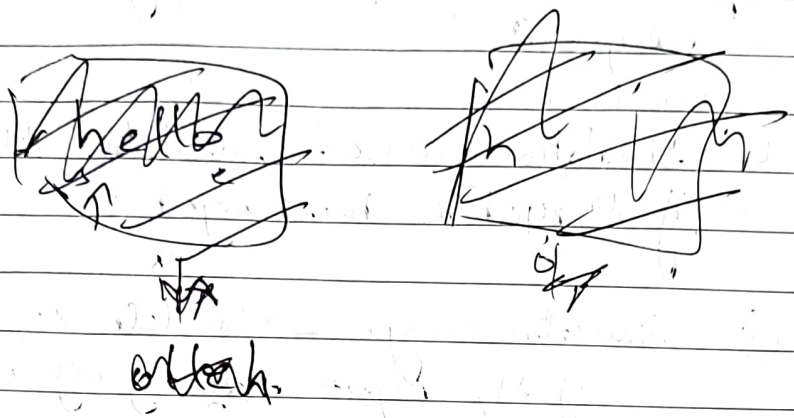
Sol Let $N = (Q, \Sigma, \Gamma, \delta_N, q_0, q_{accept}, q_{reject})$ be a nondeterministic Turing machine where

Q is the set of states

Σ is the i/p alphabet

Γ is the tape alphabet ($\Sigma \subseteq \Gamma$, where $\sqcup$ is the blank symbol)

$\delta_N : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$ is the transition



function where P(S) denote the power set of S.

$q_0 \in Q$ is the initial state
$q_{accept} \in Q$ is the accepting state
$q_{reject} \in Q$, the rejecting state & $q_{reject} \neq q_{accept}$.

Let $D = (Q', \Sigma, \Gamma, \delta_D, q_0', q_{accept}', q_{reject}')$ be the deterministic Turing machine we are constructing to simulate N.

Step 1: View computation of N as a tree.
- The computation of N on input w can be viewed as a tree T, where
  - each node represents a configuration $C = (q, u, v)$ of N, where $q \in Q$ is the current state $u \in \Gamma^*$ is the tape content to the left of the head & $v \in \Gamma^*$ is the tape content under and to the right of the head
  - The root of the tree is the initial configuration $C_0 = (q_0, \sqcup w)$
  - Each edge from a node corresponds to a possible transition defination by δ_N

→ If N accepts w, there exists a path from the root to a node corresponding to an accepting configuration $C_{accept} = (q_{accept}, u, v)$.

Step 2:- Design Deterministic Machine D to explore the computation Tree.

- The deterministic machine D simulates N by systematically exploring the tree T using a BFS approach.
- D maintains a Q of configurations to be explored.
- Initially Q contains only the root configuration $C_0 = (q_0, Bw)$.
- In each step, D deques a configuration C from Q applies all possible transition defined by δ_N, and enques the resulting configuration back into Q.

Step 3:- Formal construction of D

- we define the state set Q' of D as the set of all possible configurations of N. Formally $Q' = Q \times \Gamma^* \times \Gamma^*$
- The transition function δ_D is defined to simulate all possible transitions of N by enqueing new configurations
  - For each configuration $C = (q, u, v)$ where $q \in Q$ & $v = av'$ with $(a \in \Gamma)$ if $\delta_N(q, a) = \{(q_1, b_1, D_1), \ldots, (q_k, b_k, D_k)\}$ then for each $i = 1, \ldots, k$:
    δ_D defines a transition from (q, u, v) to configuration (q_i, u', v''), where:
    - u' & v'' are obtained by writing b in place of a moving the head according to D_i & adjusting u & v accordingly.

D halts & accepts if it reaches a configuration where $q = q_{accept}$.

Step 4: Correctness :-

soundness :- if N accepts w, then there exists a path from C_0 to C_{accept} in the tree T. Since D explores all possible paths using BFS, it will eventually reach C_{accept} & accept.

completeness - if N does not accept w, then no path leads to C_{accept}. The BFS traversal by D will eventually exhaust all possible configurations and D will not ~~accept~~ find C_{accept}. D can be designed to be rejected in this case.

Hence proved

(7) Let L be a language. Prove that, L is a decidable if & only if both L & the complement $\bar{L}$ or L^c are Turing recognizable. Give an example of a language which is not decidable but Turing recognizable.

Sol^n

(13)

Sol^

- A language L is Turing recognizable if there exists a Turing machine M such that for any input w:
  - if $w \in L$, then M eventually halts & accepts
  - if $w \notin L$, M may either reject or run indefinitely
- A language is decidable if there exists a Turing machine D such that for any i/p w:
  - D halts & accepts if $w \in L$
  - D halts & rejects if $w \notin L$.

Proof: we have 2 directions to prove. If A_L is decidable, we can easily see that both A_L & its complement $\bar{A}_L$ are Turing recognizable. Any decidable language is Turing recognizable & the complement of a decidable language also is decidable.

For the other direction, if both L & $\bar{L}$ are Turing-recognizable, we let M_1 be the recognizer for A & M_2 be the recognizer for $\bar{L}$. The following Turing machine M is a decider for A.

M = "On input w:
1. Run both M_1 & M_2 on i/p w in ||"
2. If M_1 accepts accept; if M_2 accepts reject".

Running the 2 machines in parallel means that M has 2 tapes, one for simulating M_1 and the other

for simulating M_2. In this case M takes turn simulating M_1 and the other for simulating M_2. In this case, M takes turn simulating one step of each machine, which continues until one of them accepts.

Now we show that M decides L. Every string w is either L or $\overline{L}$. Therefore M_1 or M_2 must accept w. Because M halts whenever M_1 or M_2 accept, M always halts & so it is a decider. Furthermore it accepts all strings in A & rejects all string not in A. So M is a decider for A & thus A is decidable.

ex 1- Halting problem language (L_{Halt})

$L_{Halt} = \{\langle M, w\rangle \mid M \text{ is a Turing machine \& } M \text{ halts on input } w\}$

L_{Halt} is Turing recognizable because we can identify when a Turing machine halts on a given i/p but it is not Turing decidable because we cannot determine whether the machine will halt or run forever.

2- Post correspondence Problem (PCP)

def^ → given 2 list of strings $A = [a_1, a_2, a_3 \dots a_n]$ and $B = [b_1, b_2 \dots b_n]$ over some alphabet the problem asks if there is any index $i_1, i_2 \dots i_k$ such that concatenation of the string from A

equals the concatenation of the strings from B.

$$a_{i_1} a_{i_2} \cdots a_{i_n} = b_{i_1} b_{i_2} \cdots b_{i_n}$$

PCP is Turing recognizable because Turing machine can search for a sol^n by generating all possible sequences if match then halts if not may search indefinitely never halting hence recognizable yet undecidable

(8) Show that the collection of Turing-recognizable languages is closed under intersection

Sol^n Let M_1 be a Turing machine recognizing L_1
Let M_2 be a Turing machine recognizing L_2

Construct a new machine M which simulates on both M_1 & M_2 on some i/p w in parallel where M_1 & M_2 recognizes L_1 & L_2 individually

On input w:

1. Simulate M_1 on i/p w.
2. Simulate M_2 on i/p w.
3. If both M_1 & M_2 accept w then M halts & accepts w.
4. If either M_1 or M_2 rejects w, M does nothing

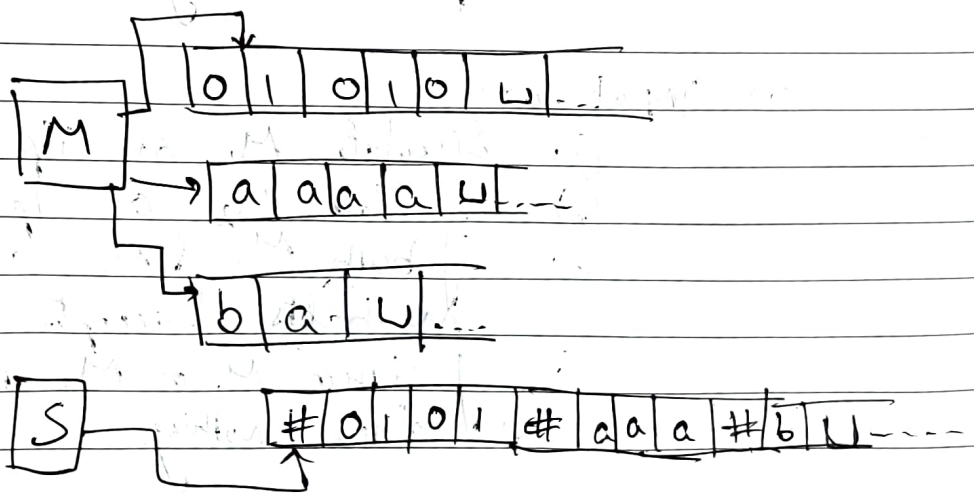
if $w \in L_1 \cap L_2$
- both M_1 & M_2 will eventually accept w so M will also accept w.
- if $w \notin L_1 \cap L_2$
  - atleast one of M_1 or M_2 will reject it or run indefinitely
  - if both or any one run indefinitely M may run indefinitely as well which is not recognizable.

Hence Proved

(9). Prove that every language accepted by a k-type Turing machine is also accepted by a single tape Turing machine

sol:

We show that to convert a multitape TM M to an equivalent single-tape TM S. The key idea is to show how to simulate M on S.



Let M has k tapes. Then S simulates the effect of k tapes by storing their information on its single tape. It uses the new symbol # as a delimiter to separate the contents of different tapes. S also keeps track of locations heads. It does so by writing a tape symbol w/ a dot above it to mark the place where the head on that tape would be.

S = "On input $w = w_1 \dots w_n$:

(1) First S puts its tape into the format that represents all k tapes of M. The formatted tape contains

$$\#w_1 w_2 \dots w_n \#\dot{\sqcup}\#\dot{\sqcup}\#\dots\#$$

(2) To simulate a single, S scans its tape from the first #, which marks the left hand end, to the (k+1)st #, which marks the right hand end, in order to determine the symbols under the virtual heads. Then S makes a second pass to update the tapes according to the way that M's transition function dictates.

(3) If at any point S moves one of the virtual heads to the right onto a # this action signifies that M has moved the corresponding head onto the previously unread blank portion of that tape. So S writes a blank symbol on this tape cell & shifts contents from this cell until the rightmost # one cell to the right.

S halts & accepts if
M halts & accepts

Hence Proved

(10) Show that any language decided by a turing machine TM with 1 work tape that is ∞ in both direction in T(n) steps can also be decided by a TM w/ work tape ∞ in only one direction in O(T(n)) steps.