Chapter 1

Turing Machines and Some Computing Devices

1.1 Generalizations of the basic model

1.1.1 Multitrack Turing Machines

A multi-track TM comprises a singly infinite tape with multiple tracks; each track is divided into squares holding tape symbols from a tape alphabet; there is a tape mount comprising tape heads which can be moved by the finite state control of the TM *simultaneously* to the left or right. Thus, if the tape alphabet of a k-track TM M_r be $\Gamma_r = \{\gamma_{r,1}, \gamma_{r,2}, \ldots, \gamma_{r,n}\}$, then any move of the M_r is given by a 5-tuple of the form $\langle q_i, \langle \gamma_{r,i_1}, \gamma_{r,i_2}, \ldots, \gamma_{r,i_k}\rangle / \langle \gamma'_{r,i_1}, \gamma'_{r,i_2}, \ldots, \gamma'_{r,i_k}\rangle, L, q_j\rangle$, where $\gamma_{r,i_l}, \gamma'_{r,i_l} \in \Gamma_r, 1 \leq l \leq k$; such a move represents that if the TM M_r is in state q_i and the heads on the tape mount read the k-tuple $\langle \gamma_{r,i_1}, \gamma_{r,i_2}, \ldots, \gamma_{r,i_k}\rangle$, then the TM M_r writes the k-tuple $\langle \gamma'_{r,i_1}, \gamma'_{r,i_2}, \ldots, \gamma'_{r,i_k}\rangle$ by the heads on the tape mount, moves the *entire mount* by one square *on each track simultaneously* to the left and then attains state q_j.

Theorem 1 *Let $\mathcal{M}_r$ be the class of TMs and $\mathcal{S}$ be the class of single track singly infinite TMs. Then, $\mathcal{M}_r \simeq \mathcal{S}$, that is, these two classes have identical computation power.*

Proof 1 The proof comprises two steps: (i) $\mathcal{S} \subseteq \mathcal{M}_r$, that is, the class of isingle track singly infinite TMs is no more powerful than that of multi-track TMs and (ii) $\mathcal{M}_r \subseteq \mathcal{S}$, that is, the class of multi-track TMs is no more powerful than that of singly infinite TMs.

i. (Part 1 – $\mathcal{S} \subseteq \mathcal{M}_r$): R.T.P: $\forall S \in \mathcal{S}, \exists M_r \in \mathcal{M}_r$ such that S simulates M_r. This is obvious — M_r uses only one of its k tracks where it simulates S leaving the other tracks unused.

ii. (Part 2 – $\mathcal{M}_r \subseteq \mathcal{S}$): Let the tape alphabet Γ_r of M_r have $n(= |\Gamma_r|)$ symbols. The total number of k-tuples (that is, $|\Gamma_r^k|$) is $(|\Gamma_r|)^k = m$, say. The single track singly infinite TM S uses the tape alphabet $\Gamma_s = \{\gamma_{1,s}, \gamma_{2,s}, \ldots, \gamma_{m,s}\}$. More specifically, therefore, Γ_s has as many individual symbols as there are k-tuples over Γ_r. Now, a move $\langle q_i, \langle \gamma_{r,i_1}, \gamma_{r,i_2}, \ldots, \gamma_{r,i_k}\rangle / \langle \gamma'_{r,i_1}, \gamma'_{r,i_2}, \ldots, \gamma'_{r,i_k}\rangle, L(R), q_j\rangle$ of M_r can be simulated by a move $\langle s_i, \gamma_{t,s}/\gamma_{t',s}, L(R), s_j\rangle$ of S, where $\gamma_{t,s}, \gamma_{t',s} \in \Gamma_s, \gamma_{t,s} = \langle \gamma_{r,i_1}, \gamma_{r,i_2}, \ldots, \gamma_{r,i_k}\rangle$ and $\gamma_{t',s} = \langle \gamma'_{r,i_1}, \gamma'_{r,i_2}, \ldots, \gamma'_{r,i_k}\rangle$. (Note that the state set Q_r of M_r has one-to-one correspondence with the state set Q_s of S.

1.1.2 Doubly Infinite Tape

A doubly infinite TM extends to infinity on both directions. Let $\mathcal{D}$ represent the class of doubly infinite TMs and $\mathcal{S}$ represent the class of singly infinite TMs. Again we have, $\mathcal{D} \simeq \mathcal{S}$, that is, these two classes have identical computational power. The fact that $\mathcal{S} \subseteq \mathcal{D}$, i.e., $\forall S \in \mathcal{S}, \exists D \in \mathcal{D}$ such that D simulates S is obvious; recall that whenever S tries to move left of its leftmost square, it reaches reject halt; so, whenever D is forced to move left of its initially scanned square (in course of simulating the moves of S), it should reject halt; (this fact can be detected by having the left neighbouring square of its initially scanned square marked with a special symbol $\star$, say;) thus, D never has to visit the left half of its tape; otherwise, D mimics the moves of S as they are.

The fact that $\mathcal{D} \subseteq \mathcal{S}$, that is, $\forall D \in \mathcal{D}, \exists S \in \mathcal{S}$ such that S simulates D, can be argued as follows. Let the initially scanned square and the squares lying at its right be non-negatively numbered (i.e., 0, 1, 2, etc.,) and those lying at the left of the initially scanned square be negatively numbered. Without loss of generality (W.l.o.g.), let S be a 2-track TM. The upper track contains the tape symbols of D's non-negatively numbered squares and the lower track contains the tape symbols in D's negatively numbered squares. The leftmost square in the lower track is marked with a special symbol \$, say. In other words, in terms of the numbering of squares of D's tape squares, the squares of the 2-tracks of S attains the designations (read from left to right) as $\frac{0}{\$}, \frac{1}{-1}, \frac{2}{-2}, \frac{3}{-3} \ldots,$. Note that as long as D computes remaining in its non-negatively (negatively) numbered squares, S concentrates on the corresponding squares in the upper (lower) track. Thus, S can be thought of having two sub-machines, designated as upper and lower submachines, say; the upper (lower) submachine simulates D's moves in the non-negatively (negatively) numbered squares. As long as D is in the non-negatively numbered squares, the right (left) movements of its head take the head to higher (lower) numbered squares; on the uppertrack, the higher numbered squares are at the right of a given square; hence the right (left) movement of the head of D remains the right move in the upper submachine of S. In contrast, a right (left) movement of D's head in a negatively numbered square takes the head from a square to a higher, i.e., less negatively (loweer, i.e., more negatively) numbered square; in the lower track, the higher (lower) numbered square is at the left of a square; hence in the lower submachine of S, the right (left) movement of D's head becomes left (right) movement of the head mount. The \$ symbol in the lower track with a left movement of S's head mount causes switching between the two submachines.

1.1.3 Multitape Model

A multitape TM comprises k tapes each of which is provided with an independent read-write head connected to a finite state control. Each tape can be considered to be infinite on both directions. On a single move, depending on the state of the finite control and the symbols scanned by *all* the tape heads, the machine

1. writes a new symbol on each cell scanned by the tape heads,

2. moves each tape head *independently* one cell to the left or to the right of the presently scanned square, or keeps it stationary,

3. changes to a new state.

Thus, a typical move of an k-tape TM M_m, say, using a tape alphabet Γ_m in a state q_i can be depicted by a 3-tuple $\langle\langle\gamma_{1,m}, \gamma_{2,m}, \ldots, \gamma_{k,m}\rangle / \langle\gamma'_{1,m}, \gamma'_{2,m}, \ldots, \gamma_{k',m}\rangle, \langle m_1, m_2, \ldots m_k\rangle, q_j\rangle$, where $\gamma_{i,m}, \gamma'_{i.m}, 1 \leq i \leq m$, respectively represent the symbol scanned and the symbol written back on

the $i-th$ tape square, $m_i \in \{l, r, s\}$ represents the movement of the $i-th$ head to the left (l), right (r) or stationary (s) and q_j represents the state attained through the move.

The following theorem captures the fact that by having more than one tape the computational power of the TMs does not increase. W.l.o.g. we show that the multitape TMs have identical power as the multi-track TMs. The latter class being already shown to have an identical power, we can establish that the multitape TMs have identical power as the singly infinite single track TMs.

Example 1 Compare a single tape TM T_1, say, with a 2-tape TM T_2, say, where $accept(T_1) = accept(T_2) = \{ww^R | w \in \{a, b\}^*$ and w^R is the reverse of $w\}$.

For recognizing this language by T_1, the head must move back and forth on the input, checking off symbols from both ends and comparing them. Thus, T_1 needs $O(n^2)$ moves.

In case of T_2, the input is copied on the second tape, the input on one tape is compared with the copy on the second tape from the opposite sides and the length of the input is ensured to be even. Thus, T_2 needs $O(n)$ moves.

Theorem 2 *Let $\mathcal{M}_m$ be the class of multitape TMs and $\mathcal{M}_r$ be the class of multitrack TMs. These two classes have idsentical computation power. Symbolically, $\mathcal{M}_r \simeq \mathcal{M}_m$.*

Proof 2 (Part 1 – $\mathcal{M}_r \subseteq \mathcal{M}_m$) So, R.T.P. $\forall M_r \in \mathcal{M}_r, \exists M_m \in \mathcal{M}_m$ such that M_r simulates M_m. This is easy; since M_r moves the head mount (i.e., all the heads on the muont) to the left or right, M_m moves all the heads *synchronously* to the left or to the right.

(Part 2 – $\mathcal{M}_m \subseteq \mathcal{M}_r$): So, R.T.P. $\forall M_m \in \mathcal{M}_m, \exists M_r \in \mathcal{M}_r$ such that M_m simulates M_r. Let M_m be a k-tape TM. Let M_r use 2k tracks; it has 2 tracks for each tape of M_m. For the $n-th$ tape of M_m, $(2n-1)th$ track contains only one marker 'X' which indicates M_m's head position of the $n-th$ tape (remaining squares are blank) and $2n-th$ track contains the tape pattern on the $n-th$ tape of M_m. The state of the fintie state control of M_r at any point of simulation of M_m can be captured by a tuple of the form $\langle q_i, \langle h, l, r, h, \ldots\rangle, \langle \gamma_{1,m}, ?, ?, ? \ldots\rangle\rangle$, thereby storing the following pieces of information:

1. the present state q_i of M_m (since M_r needs more than one move to simulate each single move of M_m),
2. the relative head positions of M_m depicted by the k-tuple$\langle h, l, r, h, \ldots\rangle$ with respect to the squares on the 2k tracks of M_r presently being scanned ('h' indicates that it is at the same position as the presently scanned squares of M_r),
3. the scanned symbols of M_m captured so far depicted as $\langle \gamma_{1,m}, ?, ?, ? \ldots\rangle$, where '?' indicates those which are yet to be captured.

To simulate a move of M_m, M_r must visit each of the cells with head markers, recording, in turn, the symbol scanned by each head of M_m. When M_r passes a head marker, it must update the relative position in which the marker can be found. After gathering the tuple $\langle \gamma_{1,m}, \gamma_{2,m}, \ldots, \gamma_{k,m}\rangle$, M_r determines the move of M_m; the latter then visits each of the head markers, in turn, changing the symbols scanned by the symbols to be written and moving the marker by one cell in proper direction, if needed.

1.1.4 Nondeterminism

A nondeterministic TM (NDTM) comprises a finite control and a single doubly infiite tape. For a given state and tape symbol scanned by the tape head, an NDTM has several choices for the next

move. Each choice consists of a tape symbol to write, a direction of the head movement and a new state to acquire. Thus, for any input, the machine produces a computation tree rather than a single computation path produced by a deterministic TM (DTM). *The nondeterministic TM accepts an input if there exists at least one finite sequence of moves, that is, one computation branch, that leads to an accept state; the input is rejected if all the branches reach some reject halts.* An NDTM is said to be a decider of a language if for all the inputs all the branches of the corresponding computation trees halts.

Theorem 3 *If a language L is accepted by an NDTM N, then it is accepted by a DTM D.*

Proof 3 For any state and tape symbol of N, there is a finite number of choices for the next move. These choices can be numbered 1, 2, ... Let r be the maximum number of choices for any state-tape sybol pair. The any finite length sequence of choices of moves of N can be represented by a string over the alphabet {1, 2, ..., r}. Not all such sequences would represent actual sequences of moves that can be taken by N since there may be less than r choices foro a state-symbol pair encountered in a configuration.

Let D have 3 tapes. The first one holds the input. On the second tape, D generates sequences of symbols 1 through r *in a systematic* manner. Specifically, the sequences are generated in a *lexicographic order* (i.e., in an increasing order of length. Among sequences of equal lengths, they are generated using a strict precedence relation $1 \prec 2 \prec \ldots \prec r$).

For each sequence generated on tape 2, D copies the input from tape 1 to tape 3; it then simulates N on tape 3, using the sequence of moves as given on tape 2. If N accepts an accept state, so does D. If there is a sequence of choices leading to acceptance by N, it will eventually be generated on tape 2 and when simulated D accepts. On the other hand if no such sequence of choices of moves of N leads to an accept state, D continues for ever generating the sequences one after another without accepting ever.

1.2 Universal Turing Machine

So far we have only discussed special purpose TMs such as a doubling machine or exponentiating machine, copying or substitution machine, etc. But if the claim that a TM serves as a model for a general purpose machine is to stand, then there should be provision for a general purpose TM which can be programmed to behave as any other TM (on any of the latter's input). Such a TM indeed exists and is called a *Universal Turing Machine*, abbreviated as UTM and denoted as U. Figure **??** provides a schematic diagram of U depicting its input $\langle T, w\rangle$, where T is any TM over its alphabet Γ_T, say, and w is any input word over T's input alphabet Γ_T, say, where $\Sigma_T \subseteq \Gamma_T$. The present section describes the working of U.

1.2.1 Encoding of the input $\langle T, w\rangle$

To encode T's description, it is sufficient to encode its state transition table (STT) which, in turn, is a finite set of quintuples of the form $\langle ps, \gamma_s, \gamma_w, m, ns\rangle$, where, ps represents present state, γ_s the scanned tape symbol of T, γ_w the tape symbol to be written, m the head movement and ns the next state; thus ps, $ns \in Q_T$, the set of states of T, $\gamma_s, \gamma_w \in \Gamma_T$, T's tape alphabet and $m \in \{L, R\}$. We use *unary encoding* whereby any information is represented by blocks of 1's and blanks as delimiters. Thus, a unary alphabet $\Sigma_1 = \{1\}$. In particular a non-negative number $n \in \mathbb{N}$ can be represented by a block of $n+1$'s. The mechanism of encoding T's description, i.e., unary encoding of the finite STT, is as follows. Let $Q_T = \{q_0, q_1, \ldots q_{k-1}\}$. The state q_i is encoded by a block $i+1$

1's. Similarly, let $\Gamma_T = \{\gamma_0, \gamma_1, \ldots, \gamma_n\}$; a tape symbol γ_j is represented by a block of $j+1$ 1's. The direction of T's head movement $m \in \{L, R\}$; hence, L can be represented by a block of one 1 and R can be represented by a block of two 1's. Thus, an STT-entry $\langle q_i, \gamma_j, \gamma_l, L, q_t \rangle$ is represented as $\Delta\Delta 11\ldots1(i+1\ 1\text{'s})\Delta 11\ldots1(j+1\ 1\text{'s})\Delta 11\ldots1(l+1\ 1\text{'s})\Delta 1\Delta 11\ldots1(t+1\ 1\text{'s})\Delta\Delta$.

Thus, a quintuple is delimited by two consecutive Δs and the STT is delimited by three consecutive Δs. Similarly, T's tape pattern is over Γ_T can be encoded with one Δ acting as a delimiter between two consecutive T's tape symbols in the pattern. The pattern is delimited by two consecutive Δs.

1.2.2 Simulation of T on the input w by U

The tape of U is as given in figure 1.1.

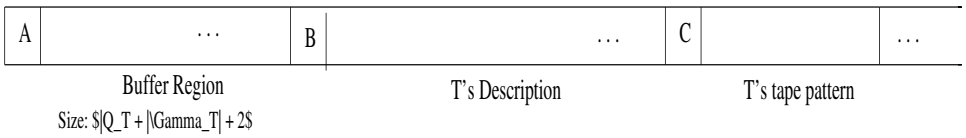

Figure 1.1: Regions of UTM's Singly Infinite Tape

The tape pattern of T being $\gamma_1\gamma_2\ldots\gamma_k$ can be represented by stringing together the unary representations of $\gamma_1, \gamma_2, \ldots, \gamma_k$ with a single Δ between two blocks of 1's. The TM T being of singly infinite type, the string $\gamma_1\gamma_2\ldots\gamma_k$ can be assumed to be followed by an infinite string of Δ's (blanks). Of the three regions of U's tape, the buffer region following the special symbol A is meant to copy the present state ps and the scanned symbol γ_s components of T. Hence it is of fixed size, specifically of the value $|Q_T| + |\Gamma_T| + 2$, where $|Q_T|$ is the number of states of T, $|\Gamma_T|$ is the number of tape symbols of T and the extra 2 squares are for the delimiting Δ's following the ps and the γ_s blocks.

The UTM U starts its operation with its tape inscribed as follows. The buffer region is blank. The machine description region is preceded by a special symbol B and followed by three blank squares. The m/c T's description (i.e., the quintuples of its STT) starts with a quintuple corresponding to the initial state of T. *Dynamically, B precedes a ps-block of 1's before simulation of each move of T by U.* The tape description region contains the encoded description of T's initial tape pattern with the special symbol C immediately preceding the initially tape scanned symbol in T; again, *dynamically, C would precede the γ_s-block (i.e., T's scanned symbol) before simulation of any move of T by U.* The head of U initially scans the square containing the symbol B.

The phases of the simulation are as follows.

1. (Phase 1): Copy the block of 1's following B (i.e., the ps-block) into the buffer region after the symbol A;
   write an auxiliary A after the copied block in the buffer region; erase B (replace with Δ); go to Phase 2;

2. (Phase 2): Copy the block of 1's following C (i.e., the scanned symbol γ_s) after the auxiliary A in the buffer region;
   erase the auxiliary A; put B at the beginning of the description of T (which is a fixed distance i.e., $|Q_T| + |\Gamma_T| + 2$ squares from the leftmost square in U's tape (marked by A); go to phase 3;

3. (Phase 3): Match the 2-tuple $\langle ps, \gamma_s \rangle$ in the buffer region with the 2-tuple following B;
   if no match, shift B right to the next quintuple (marked by a leading pattern $\Delta\Delta$); while shifting right B, if end of T's description (indicated by 3 consecutive Δ's) is encountered, then halt (reject – since T rejects on unspecified scanned symbol);
   if a match is obtained, then (the desired quintuple is found) go to phase 4;

4. (Phase 4): Erase the buffer region; shift B right so that it stands before the component indicating the output symbol γ_w; go to phase 5;

5. (Phase 5): T's tape pattern is updated by replacing the block after C by the block after B (this requires collapsing and then insertion); shift right B so that it stands before the directional component; go to phase 6;

6. (Phase 6): Depending on the number of 1's after B, C is moved to the left or right by one block; while shifting C left, if it runs off the tape, then U halts (reject – since T reject halts); while shifting C right, if end of T's tape pattern is reached, then a block corresponding to blank symbol (of T) is to be introduced; go to phase 7;

7. (Phase 7): Shift right B so that it stands in front of the ns (next state) component; (so, at this point B indicates the block for the present state and C indicates the block for scanned symbol of T;) go to phase 1;

Exercise 1.1 Design of Turing Machines
Submit on 20.8.'19 (next Monday) in a file *so that you can attach the subsequent submissions in that file.*
For all the following TM design problems, your solutions must have the following components:

i The assumptions that you have made regarding

   (a) the input encoding, if any,

   (b) position of the input and the tape head on the tape, etc.,

ii The stage(phase)-wise narrative description of the machine behaviour,

iii a *neat* state transition diagram with the phases give in the above narrative description clearly indicated in the diagram; alternatively, you can give the submachines of the phases separately.

1. *Doubling m/c* Should compute the function $f : \mathbb{N} \to \mathbb{N}, n \mapsto 2n$, where n is represented using the unary alphabet $\Sigma = \{1\}$. Hint: Copy the input block of 1's without destroying the latter.

2. *Exponentiating m/c*:Should compute the function $f : \mathbb{N} \to \mathbb{N}, n \mapsto 2^n$, using the Doubling m/c of Question 1 as a submachine. *Note: This would give you an idea how TM models permit subroutine usage.*

3. Should compute the function $f(m, n) : \mathbb{N}^2 \to \mathbb{N}, \langle m, n \rangle \mapsto m + n$, using the binary alphabet $\Sigma = \{0, 1\}$.

4. Should *decide* the language $L = \{a^n b^n c^n, n \geq 0\}$ over $\Sigma = \{a, b, c\}$ (i.e., the Turing Machine (TM) should come to an accept halt state for any input string $\omega \in \Sigma^*$, a reject halt state for any $\omega \in (\Sigma^* - L)$ and hence should never loop).

5. Copy machine 1: Initial tape: $C1110\cdots\Delta A111101\cdots\Delta\Delta\cdots$
   Block of 1's to the right of A delimited by a 0 has to be copied to the right of C – assume that sufficient space is available between C and A.

6. Copy machine 2:

   (a) Initial tape: $A11110\cdots B1101110\cdots\Delta\Delta$
   Block of 1's following A is to be copied following B *shifting the existing blocks of 1's following B right by as many places as needed*. Therefore, the final tape: $A11110\cdots B111101101110\cdots\Delta\Delta$

   (b) Replacement :
   Initial tape: $A11110\cdots B1101110\cdots\Delta\Delta$
   The block of 1's following B has to be *replaced* by the block following A. Therefore, the final tape: $A11110\cdots B111101110\cdots\Delta\Delta$
   (Hint: Conceive it as a serially composite machine of two submachines – the first one for *collapsing* the block of 1's following B so that a) appears next to B and then the second submachine is deployed to *insert* the 1's of the block of 1's following A.

7. Initial tape: $((\leftarrow e_1 \rightarrow (\leftarrow e_2 \rightarrow))(\leftarrow e_3 \rightarrow)\cdots)(\leftarrow e_n \rightarrow)\Delta\cdots$, where $e_i, 1 \leq i \leq$, are expressions involving two variables x, y and arithmetic operators $+, -, \times, /$.
   The machine is to accept halt on matched parentheses. Note that the well-formedness of the arithmetic expressions e_i's need not be checked by the TM.