Theory of Formal Languages (Introduction to Turing Machine)

Lecture 01: Jan. 20, 2021

Prof. K.R. Chowdhary *: Professor of CS*

1.1 Introduction

In 1936 Alan M. Turing defined a class of logical machines which he used as an aid in proving certain results in mathematical logic, and which should prove of interest in connection with the theory of control and switching system. He had shown that given any logical arithmetic computation for which complete instructions for carrying are supplied, it is possible to design a TM (Turing Machine), which can perform this operation. Turing suggested that such a machine is an abstract mathematical model of a human being assumed to perform a computation, with the states of mind of the human, and the symbols on tape corresponding to the numerical answers and to the intermediate results on the scratch paper.

Developments since the time of Turing's paper have been in the direction of digital computer which shows a much more direct resemblance to Turing machines than to human beings. In fact, several earlier computers comprised magnetic tape or perforated paper tapes as auxiliary memories. The one tape machines are called Turing machines.

In the previous chapters, we have discussed the languages' recognition machines like–finite automata (FA) and pushdown automata (PDA). The first recognizes the language of regular expression and second–the context free languages. The pushdown automata are more powerful as they can recognize some languages not recognizable by the finite automata. This is due to the fact pushdown automata has unlimited read/write memory called pushdown stack where reading and writing takes place at one end of this memory. For example, a PDA can recognize the languages like $\{a^n b^n \mid n \geq 0\}$, which cannot be recognized by the FA. However, there still exists the languages like $\{a^n b^n c^n \mid n \geq 0\}$ and $\{ww\}$, which cannot be recognized by the PDA. In this chapter, we will study the machines, which can recognize these and even more complex languages. These machines are called Turning Machines (TM) after their inventor (Alan M. Turing). These machines are similar to the machines discussed earlier, but are more general than those. A TM consist of finite control like FA and PDA, a tape for reading and writing on it, and a head which can read from this tape and can write on it, and while doing this the head can move right and left on the tape.

The Turing machines are very simple in structures. In spite of this, they need not to be augmented for doing complex tasks. The access in the Turing machine is always sequential. Still, machines, which appear more attractive, like those with random access capability, in fact do not increase the computing power. These, visibly more powerful machines can be simulated by the Turing machine, and have no extra computation power than standard TM. Other alternative and powerful TMs seem to form a stable and maximal class of computing devices in terms of computations they perform. It may be noted that, computing power

here means not the speed of computation, but the power of problem solvability through the machine, given the sufficient time to solve it.

A Turing machine satisfies following three general criteria:

1. These are automata, i.e., their function and construction are in the similar lines of the automata discussed earlier,
2. These should be as simple as possible, to define formally, describe and reason about them, and
3. These should be as general as possible so that any computation can be represented using them.

By analogy of TMs with modern computers is as follows: the tape of TM is equal to the memory of computers, and the read-write head is equal to CPU.

1.2 Turing Machine Model for Computation

A Turing-machine computing model (Figure 1.1) consists of a finite-state control, a tape and read-write head. The read-write (RW) head reads the input symbols from the tape, as well as writes or modifies the symbols on the tape.

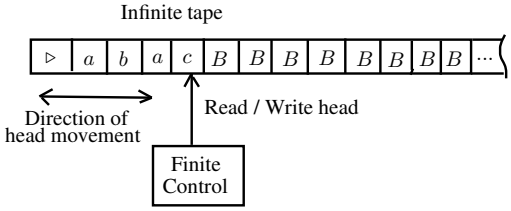

Figure 1.1: Turing Machine.

The finite control operates in discrete steps, where, in each step it performs following operations that are decided by the current state of the finite-control and the tape symbol currently read by the read/write head:

1. Write a symbol on a square on tape from which it has just read a symbol, thus replacing the previous symbol,
2. Change of state of the control to a new state, and
3. Move the RW head to left or write. The direction of move will be decided by the current state and the symbol the RW head has just read.

The tape of the TM ends at its left but extends to right without any limit. Thus, to prevent the read-write head from moving beyond this left end, an end-marker symbol ($\triangleright$) indicates the left end of the tape. The end marker is not counted as symbol in alphabets of input. We will use R (right) and L (left) characters as short forms to indicate the direction of the

head movement. Input to the TM is supplied by writing the string on the tape in advance, starting immediately after the symbol $\triangleright$ and extending towards right up to any length. Note that, at many places in our text, we will be using B (for blank) symbol in place of $\triangleright$.

The rest of the tape squares after string end are blank, indicated by B symbols. The TM is free to alter its input string in any way it sees fit, and can write symbols beyond it. Since the machine's head makes only one square move after reading each square, therefore, for any finite computation only finite number of squares will be scanned on the tape. Mathematical Model of TM A Turing machine is a 6-tuple defined as

$$M = (Q, \Sigma, \Gamma, \delta, s, H) \tag{1.1}$$

where,

Q is finite set of states

Σ is set of input symbols

Γ is set of tape symbols, and $\Gamma = \Sigma \cup \{B, \triangleright\}$, here B is for blank space, and $\triangleright$ is left end marker symbol on tape When B is used as left most end marker, the tape symbols are $\Gamma = \Sigma \cup \{B\}$

$B \in \Gamma$ is a symbol for blank

$s \in Q$ is start state

$H \in Q$ is set of halting states, called accepting states, and,

δ is transition function.

The transition function is a mapping,

$$\delta : (Q - H) \times \Gamma \to Q \times \Gamma \times \{L, R\} \tag{1.2}$$

where L, R indicates that head takes left or right move. Thus, a transition in TM can be represented by

$$\delta(p, a) = (q, b, d) \tag{1.3}$$

where $p \in Q - H$, $q \in Q$, terminals $a, b \in \Gamma$, and the displacement $d \in \{L, R\}$. A transition indicates that a TM in state p (not a halting state), and symbol on the tape square being read is a then TM replaces a by b on that square, changes the state to q, and moves the head one square to left or to right depending on the value of d. If $q \in H$ then the TM has halted. Once the TM is halted then it does not cause further transitions any more, hence is not defined for $\delta(p, a)$, where $p \in H$, and a is any symbol in Γ. If the head moves to extreme left end of the tape then δ is defined as,

$$\delta(p, \triangleright) = (q, \triangleright, R) \tag{1.4}$$

This shows that the leftmost symbol $\triangleright$ on tape is never erased. Also, the machine does not write $\triangleright$ on the tape except at the extreme left end of the tape. Unlike in Pushdown

Automaton, transition function $\delta(q, a)$ for a TM is always a single value (not a set). Therefore, TM is always deterministic and $\delta(q, a)$ is not defined for halting state (H). However, non-deterministic TMs are possible as variations of basic TMs, but these are in no way more powerful than the standard Turing machines.

1.3 Configurations and Moves of TM

A *Configuration* of a TM is denoted by $\alpha_1 q \alpha_2$, where $\alpha_2 \in \Sigma^*$ is a string which extends from RW head up to right most non-blank symbol, and $\alpha_1 \in \Sigma^*$ is a string which extends from head position to left up to end marker. It is assumed that the head is pointing to the left most symbol of α_2. If $\alpha_2 = \varepsilon$ then the head is pointing to blank (B).

A move of a TM can be defined as follows. Let us assume that,

$$X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n$$

be an ID (Instantaneous Description) shown in figure 1.2, where X_i is the symbol pointed by RW head currently. Consider that there is a transition

$$\delta(q, X_i) = (p, Y, d).$$

It shows that TM at state q and head pointing to symbol X_i, write Y at that place, changes state to p and the RW head takes a displacement d (which may be left, right or may remain stationary).

If $i - 1 = n$ then $X_i = B$. If $i = 1$ then $X_i =\triangleright$ and the head will move to right, else it will fall off the tape or we say it crashes. If $i > 1$ and $i = n$ then for $d = L$, we write a move as,

$$X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n \vdash_M X_1 X_2 \ldots X_{i-2} p X_{i-1} Y X_{i+1} \ldots X_n.$$

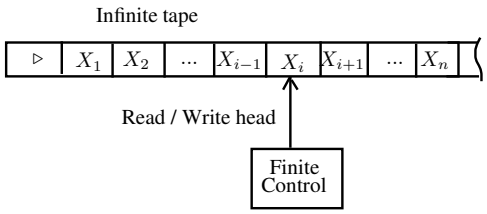

Figure 1.2: TM representing $ID = X_1 X_2 \ldots\ X_{i-1} q X_i X_{i+1}\ \ldots X_n$.

Alternatively, for $i > 1$ and $d = R$, a move is written as,

$$X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n \vdash_M X_1 X_2 \ldots X_{i-1} Y p X_{i+1} \ldots X_n.$$

For $i - 1 = n$, the entire string is to the left of q, and $X_i ... X_n$ part is empty. If two IDs are related by $\vdash_M$ then we say that second ID results from the first by a single move. If two

IDs are related by $\vdash_M^*$ then we say that second ID results from the first ID by zero or more number of moves, but certainly by finite number of moves.

Consider that language $L = L(M)$ accepted by the TM M is such that at start, head points to the left most square of the tape (i.e., just after $\triangleright$), the start state is q_0, and ultimately the M enters into the final state. Thus, language L is formally represented as:

$$L = \{w \mid w \in \Gamma^*, q_0 w \vdash_M^* \alpha p \beta \text{ for some } p \in H, \text{and } \alpha, \beta \in \Gamma^*\}. \tag{1.5}$$

The following example explains accepting of input strings by a Turing machines.

Definition 1.1 Acceptance by Turing machine. *A TM M accepts a string w if there exists a sequence of configurations $C_1, C_2, \ldots, C_n$ such that,*

1. *C_1 is starting configuration of M on input w, expressed as $q_0 w$,*
2. *C_n is accepting configuration, expressed as $uq_h v$, and $u, v \in \Gamma^*$, q_h is halting or accepting state, and*
3. *for each $1 \leq i \leq n$, C_i yields C_{i+1}.*

$L(M)$, the language of M, i.e., strings recognized by M, is the set of strings accepted by M.

Example 1.2 *Turing machine as an algorithm to erase the memory content.*

Consider a TM, $M = (Q, \Sigma, \Gamma, \delta, s, H)$, where $Q = \{q_0, q_1\}$, $\Sigma = \{a\}$, $\Gamma = \{a, B, \triangleright\}$, $s = q_0$, $H = \{q_1\}$ and transition function δ is given as:

$$\delta(q_0, a) = (q_0, B, R), \text{ and } \delta(q_0, B) = (q_1, B, L).$$

For an input string $w = aaaa$, the corresponding IDs and moves are as follows:

$$\begin{aligned} q_0aaaaB &\vdash Bq_0aaaB \vdash BBq_0aaB \vdash BBBq_0aB \\ &\vdash BBBBq_0B \vdash BBBq_1B \end{aligned}$$

Therefore, the above TM simply erases all the characters of input string and then halts. Figure 1.3 shows the transition graph for this Turing machine.

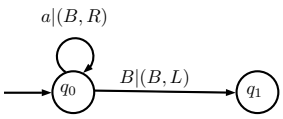

Figure 1.3: Transition diagram for TM erasing its tape contents.