Theory of Formal Languages (Turing Machine Extensions)

Lecture 4: Jan. 05, 2019

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

4.1 Introducing

Variations of Turing Machine exist that are proposed to enhance the power of the Turing Machine. It has been proved that, the problems solved by these new models can also be solved by basic model discussed so far. The solution processes used by these new models are mechanical procedures and any mechanical procedure can be implemented on a basic Turing machine. This gives further strength to the basic TM that it is an ultimate TM. Hence, TMs with multiple tapes or multiple dimensions, or random access have problem solving capability no better than basic TM. In fact, a TM with non-determinism is in no way better than the basic TM, though it guesses and therefore does not work purely on mechanical procedure. This fact was not verified even by Alan Turing himself.

One problem, which was not immediately solved by Turing's thesis was –different problem solvability by TM at different times. This corresponds to rewritability of programs in modern computers and the corresponding machine is called universal Turing machine.

Many variations have been proposed for TM extensions, the major extensions are as follows:

1. Multiple tape Turing machine
2. Multiple track Turing machine
3. Two dimensional Turing machine
4. Multidimensional Turing machine
5. Two-way infinite tape
6. Nondeterminic Turing machine
7. Combinations of the above
8. Universal Turing Machine

Theorem 4.1 *The operations of a TM allowing some or all the above extensions can be simulated by a standard TM. The extensions do not give us machines more powerful than the TM.*

In the course of time, as we progress, we will be supported by evidence that this theorem is true. At some point of time, we may prove this theorem also. The power here means its problem solving capability, and not the relative speed at which it computes. However, the extensions are helpful in designing machines to solve particular problems.

4.2 Multi-tape Turing Machines

A Multi-tape Turing Machine (figure 4.1) consists of k-tapes (for example, $k = 3$), and k number of read-write heads, and each tape extending in infinite in one direction. On a single move, depending on the state of the finite control and symbol read from each tape, the machine can perform following operations:

1. Change the state to a new state,
2. Write a new symbol at the current head position on each tape, and,
3. Each head can move right or left or remain stationary on each tape independently.

However, many a time one tape is considered for the original input string, and only read operation takes place with that, while manipulation of symbols go on other tapes where read/write operations take place. For example, in a two tape Turing machine, each tape has its own read-write head, but the state is common for all tapes and all heads. In each step (transitions) TM reads symbols scanned by all heads, depending on head position and current state, next, Each head writes, moves R (right) or L (left), and control-unit enter into new state. Actions of R-W heads are independent of each other.

Let the the tape position in two tapes is $[a, b]$. The transition function δ is given by:

$$\delta(q_i, [a, b]) = (q_j, [a', b'], d, d),$$

where $d \in \{L, R\}$. Various moves of this two-tape Turing Machines can be given as follows:

δ	a, a	B, a	a, B	B, B
q_0	q_1, a, b, L, R	q_2, b, B, L, L	$q_0, b, B, L, R,$	$\ldots$

A standard TM is multi-tape TM with single tape. The multi-tape TMs are better suited for specific applications, e.g., copying string from one tape to another tape. However, their time-complexity remains **P** (Polynomial) only. Figure 4.1 shows a 3-tape Turing machine, presently in state q_i, and three different heads pointing to three tapes.

A transition in a multi-tape Turing machine, for number of tapes $k \geq 1$ is:

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

and a specific transition is given by,

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, (b_1, \ldots, b_k), (d_1, \ldots, d_k)).$$

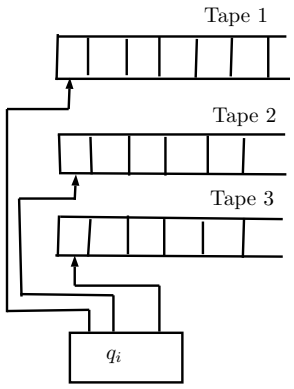

Figure 4.1: Multi-tape Turing machine.

Here, $a_1, \ldots, a_k$ are are original symbols on tapes $T_1, \ldots, T_k$, and $b_1, \ldots, b_k$ are new symbols written on these tapes, and head of these tapes takes movements (displacements) $d_1, ..., d_k$, respectively, where $d_i \in \{L, R\}$. The steps to carry out a transition are:

1. change to next state;
2. write one symbols on each tape;
3. independently reposition each tape heads.

4.2.1 Two-tape Turing Machine

A two-tape Turing machine has two read/write tapes, and corresponding two read/write heads, and one state. After each transition, the read/write heads will write each a symbol on their current position, then moves to right both or left both, or one left and other right, and the machine changes from q_i to q_j.

$$\delta(q_i, [a, b]) = (q_j, [a', b'], d, d),$$

and $d \in \{L, R\}$.

Example 4.2 *Construct a Two-tape TM to recognize language* $L = \{a^n b^n \mid n \geq 0\}$

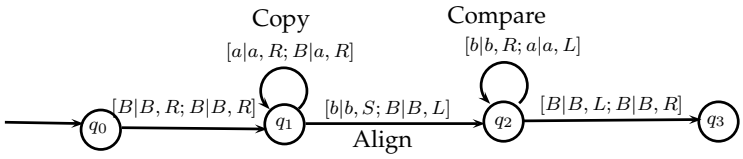

Figure 4.2: Recognition of language $L = \{a^n b^n \mid n \geq 0\}$ using a two-tape TM.

With the original string w on tape 1, the transitions (instructions) performed by the TM are shown in figure 4.2, and major steps are as follows:

1. *Copy:* Copies the string of a's on tape-1 to tape-2.
2. *Align:* Moves head 1 to begin of b's on tape-1, and head-2 on last a on tape-2.
3. *Compare:* Compares number of b's tape-1 with number of a's on tape-2 by moving heads in opposite directions.

The *time complexity* of this TM is computed as follows: There are total $1+n+1+n+1 = 2n+3$ transitions for a length of string $|w| = 2n$. Thus, for input w of length n, number of transitions are $n+3$, as the constant factor 3 remains unchanged. So, the time-complexity is $O(n+3) = O(n)$, which is *Polynomial* in time, i.e., **P**.

Compare the above computed complexity with that of $O(n^2)$ for standard TM for same problem. We note that in two-tape TM it is linear, i.e., its complexity increases proportional to size of input, while in standard TM it increases quadratically with the size of the input, however in both the cases it is in **P**.

□

Example 4.3 *Two-tape TM to recognize language* $L = \{ww^R \mid w \in \{a,b\}^*\}$.

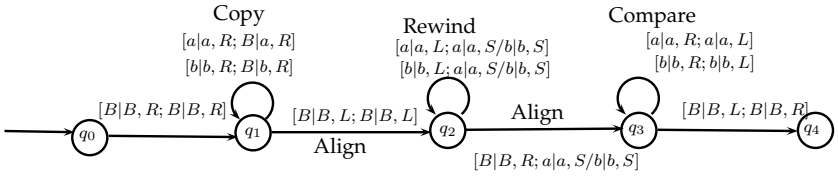

Figure 4.3: Two tape Turing machine to recognize $L = \{ww^R \mid w \in \{a,b\}^*\}$.

Working: With the original string w is on tape 1,

1. Copy string w on tape 2,
2. Move head 1 to extreme left (rewind),
3. Aligns both heads on opposite, i.e, head 1 to extreme left, head 2 to extreme right,
4. Compare tape 1 and 2, by moving heads in opposite directions.

The transition diagram in figure 4.3 also shows these sequence of steps.

Time Complexity of this TM is total number of transitions in copy, rewind, compare, and align operations. For $|w| = n$, following are the count of transitions:

Align: 1 transition

Copy: n transitions

Align: 1 transition

Rewind: n transitions

Align: 1 transition

Compare: n transitions

Align: 1 transition.

Hence, the total transitions are $3n + 4$, and time complexity is $O(n)$. This is a Polynomial complexity (i.e., **P**).

□

Example 4.4 *Construct a 2-tape TM to recognize the language* $L = \{ww \mid w \in \{a, b\}^*\}$.

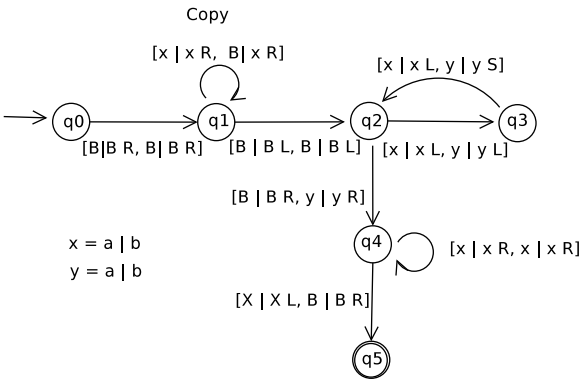

Figure 4.4: Two tape TM to recognize $L = \{ww \mid w \in \{a, b\}^*\}$.

In the figure 4.4, $x \in \{a, b\}$ is symbol on tape 1, and $y \in \{a, b\}$) is symbol on tape 2, independent of that on tape 1. When same symbol x is shown for a transition in both tape 1 and 2, like in $[x|x, R;\ B|x, R]$ at q_0, and $[x|x, R;\ x \mid x, R]$ at q_4, it shows identical values – either both x as a or both as b. Initially the string ww is on tape-1. Following are the steps by TM to recognize the language ww:

1. *Copy:* Copy tape 1 to tape-2, at the end both R/W heads are at right most positions of tape 1 and tape 2. This is done by loop on state q_0.
2. *Align:* Move both heads left, head 1, for 2-steps and head 2, for 1-step each time, in loop formed of states q_2, q_3, so that when head 2 has covered half distance on tape 2, the head 1 is to its extreme left.
3. *Align:* $(q_2 \to q_4)$ Move both heads right, each one step. Due to this, Head-1 moves to first symbol of first w of ww, and head-2 moves to 1st symbol of second w.
4. *Compare:* Comparing the two w of input ww in state q_4.

□