Theory of Formal Languages (Turing Machine as Language Recognizer)

Lecture 2: Nov. 30, 2018

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

2.1 TM as Language Recognizer

Example 2.1 *Construct a TM that accepts the language* $L = \{a^n b^n \mid n \geq 0\}$.

Let $M = (Q, \Sigma, \Gamma, \delta, s, H)$ be a Turing machine, which accepts the language L. Let us assume that the string $w = a^n b^n$ exists on the tape, followed by infinite number of blank characters B. The working of M to recognize w is explained in following steps:

1. M replace left most a by X, and then head moves to right until it encounters left most b.
2. Replaces this b by Y, and then moves left to find the right most X. Then moves one step right to left most a.
3. Repeat Step 1 and 2 in order, i.e., 1, 2, 1, 2, ...
4. When searching for b, if M finds a blank character B (i.e., $|a^n| = |b^n|$) then M halts without accepting.
5. If a is not found but it finds b ($a^n \neq b^n$), then M halts without accepting.
6. After changing b to Y, if M finds no more a then it checks that no more b remains. If this is true then $a^n b^n$ is accepted by M (i.e., $|a^n| = |b^n|$).

Based on the above procedure, M can be specified as given below.

$$
\begin{aligned}
Q &= \{q_0, q_1, q_2, q_3, q_4\} \\
\Sigma &= \{a, b\} \\
\Gamma &= \{\triangleright, a, b, X, Y, B\} \\
s &= q_0 \\
H &= \{q4\}
\end{aligned}
$$

The transition function for this machine is given below:

Figure 2.1 shows a transition diagram for this TM accepting language $L = \{a^n b^n \mid n \geq 0\}$.

Table 2.1: Transition table for $a^n b^n$.

state	a	b	X	Y	B
q_0	(q_1, X, R)			(q_3, Y, R)	(q_4, B, L)
q_1	(q_1, a, R)	(q_2, Y, L)		(q_1, Y, R)	
q_2	(q_2, a, L)		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	(q_4, B, L)

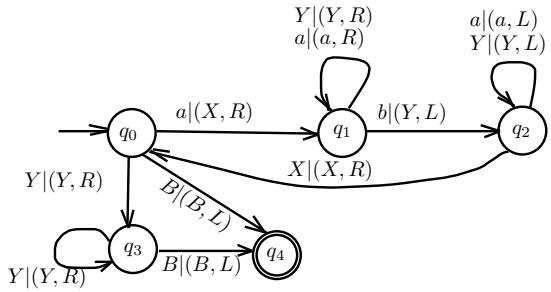

Figure 2.1: Transition diagram for TM accepting Language $L = \{a^n b^n \mid n \geq 0\}$.

Let $w = aabb$. The following are sequence of moves, which leads to the acceptance of w. A character B after $aabb$ indicates end of the input string.

$$
\begin{aligned}
q_0 aabbB &\vdash X q_1 abbB \\
&\vdash X a q_1 bbB \\
&\vdash X a q_2 Y bB \\
&\vdash X q_2 a Y bB \\
&\vdash q_2 X a Y bB \\
&\vdash X q_0 a Y bB \\
&\vdash X X q_1 Y bB \\
&\vdash X X Y q_1 bB \\
&\vdash X X q_2 Y Y B \\
&\vdash X q_2 X Y Y B \\
&\vdash X X q_0 Y Y B \\
&\vdash X X Y q_3 Y B \\
&\vdash X X Y Y q_3 B \\
&\vdash X X Y q_4 Y B
\end{aligned}
$$

The reader may verify that strings aab, $abab$, abb etc., are not accepted.

Example 2.2 *Finding complexity of TM for* $L = \{a^n b^n \mid n \geq 0\}$.

The time (number of steps) taken by the 1-tape Turing machine for this can be calculated as follows:

- Assume that R/W head was initially pointing to input symbol a of a^n. In the first

run, the head moves total $n-1$ steps in a's and 1 step in b (total n steps) in forward direction, and equal in reverse, with total $2n$ steps.

- This is repeated for n-times, making total of $n * n = n^2$ steps.
- Next, there are total n-steps for transitions from q_0 to q_3, plus for loop at q_3 state. And finally one step for q_3 to q_4 transition.
- Thus total transitions are $n^2 + n + 1$ for string $a^n b^n$. For an input $|w| = n$, this value is $(n^2 + n + 1)/2$. This gives a time of $(n^2 + n + 1)/2$, or time complexity of $O(n^2)$.

□

Example 2.3 *Show that the language $L = \{a^n b^n c^n \mid n \geq 0\}$ is accepted by a TM.*

We have just seen that language $a^n b^n$ can be accepted by a TM. In that process $a^n b^n$ gets converted to $X^n Y^n$ before the TM halts.

To design a TM to accept $a^n b^n c^n$, we use the almost the previous technique and perform the following conversions:

1. *aaabbbccc* is converted to $XaaYbbZcc$,
2. After the next operation, the tape contents are $XXaYYbZZc$,
3. Finally, the tape contents are $XXXYYYZZZ$.

The above can be carried out by adding one more state after q_2, say q_2'. Hence, this proves that language $a^n b^n c^n$ is accepted by a TM using the approach similar to method of example 2.1.

Example 2.4 *Recognize a binary string that has equal number of* 0*'s and* 1*'s.*

On each pass to right, one 0 and one 1 are changed to X, then the head goes to the left most position of the string. We assume that left most symbol is B, i.e., tape contents at begin is $B\alpha B$, where string α has equal number of 1's and 0's. The machine accepts α when the whole input has been converted to Xs.

The transition table below shows the transitions to accept the input.

state	0	1	X	B
q_0	(q_1, X, R)	(q_2, X, R)	(q_0, X, R)	(q_4, B, L)
q_1	$(q_1, 0, R)$	(q_3, X, L)	(q_1, X, R)	
q_2	(q_3, X, L)	$(q_2, 1, R)$	(q_2, X, R)	
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_3, X, L)	(q_0, B, R)

The above transitions are shown in the transition graph in figure 2.2.

□

Example 2.5 *Construct a TM for the language $\{ww^R \mid w \in \{0,1\}^*\}$.*

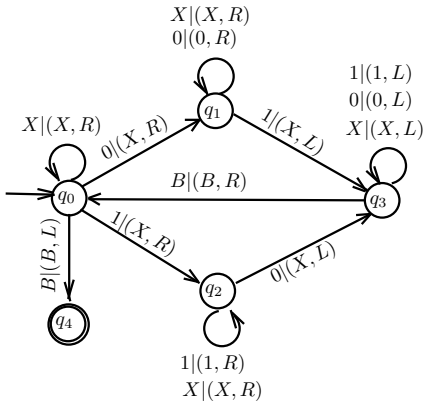

Figure 2.2: TM for $L = \{w \mid w \text{ has equal number of 0's, and 1's}\}$.

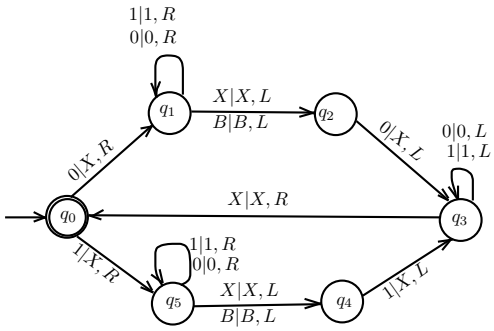

Figure 2.3: TM for $L = \{ww^R \mid w \in \{0,1\}^*\}$.

The figure 2.3 shows the transition diagram for above Turing machine. Note that the language consists even palindromes. The transition table can be easily constructed using this diagram.

The figure 2.4 is TM for odd Palindrome language, i.e., $\{w \mid w = u(a+b)u^R, \text{ and } u \in \{a,b\}^*\}$.

□

Example 2.6 *Constrict a TM to recognize the language* $\{w\#w \mid w \in \{0,1\}^*\}$.

The solution idea is simple – we mark every time the matching symbols pair $1-1$ or $0-0$, in two w's. Following are the steps:

1. At the start, the input w is on the tape and tape-head points to 1st symbol of w.
2. Mark the symbol below head, remembering it as 1 (or 0), depending on its value.
3. Move to right, finding the first unmarked symbol 1 (or 0) after #. If empty cell is found after # or 1 (or 0) is not found, then *reject*.
4. Otherwise, mark 1 (or 0) as X, and return to left, stopping at the first marked symbol after #, mark it, then move one cell to right.

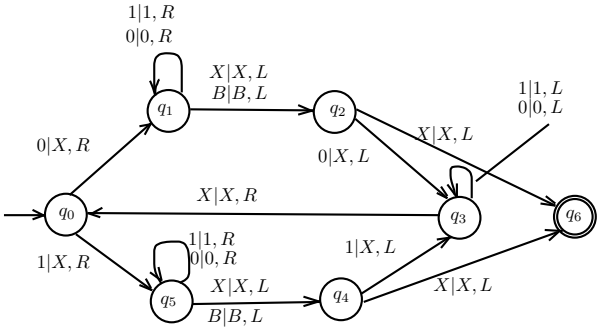

Figure 2.4: TM for $L = \{w \mid w = w^R \in \{0,1\}^*\}$, and w is an odd Palindrome.

5. Repeat this process until all symbols to the left of # are marked. Then move to right, looking for unmarked symbol after the #.
6. If any are found, *reject*, else if a blank symbol is found *accept*.

Note that in the above steps, either parenthesized value, i.e. (or 0) applies, or non-parenthesized value applied.

The figure 2.5 shows the transition diagram.

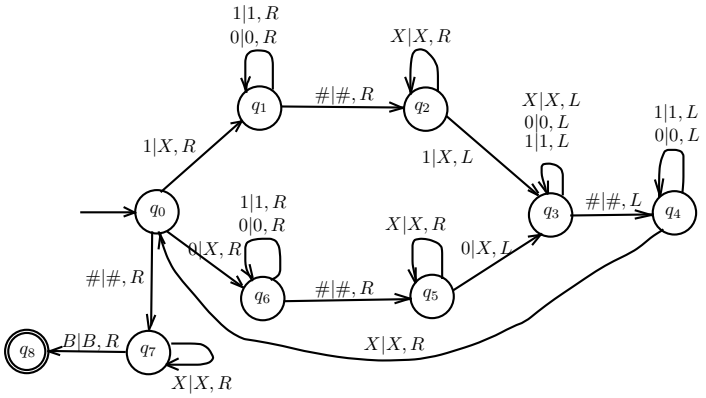

Figure 2.5: TM for $L = \{w\#w \mid w \in \{0,1\}^*\}$.

□

In all the examples above, we were given formal definition of some language $L \in \Sigma^*$, and we have demonstrated that there exists some Turing machine M such that, for any string $w \in \Sigma^*$, if $w \in L$ then $M(w) =$ "yes", and if $w \notin L$ then $M(w) =$ "No". Then we say that M *decides* L. If L is decides by some Turing machine M, then L is called a *Recursive Language*.

We say that M *accepts* L whenever, for any string $w \in \Sigma^*$, if $w \in L$ then $M(w) =$ "Yes", but if $w \notin L$ then $M(w) =$ "Undecided".

If L is *accepted* by some Turing machine M, then the language L is called *Recursively Enumerable*.

Example 2.7 *Show that the language L made of all strings of well-matched parenthesis is* recursive.

We try to give recursive definition for the language L containing all strings of well-matched parenthesis.

base: $() \in L$.

Induction: If $w \in L$ then $w()$, $()w$, (w) are all in L.

Let f defined as $f : \Sigma^* \to \Gamma^*$, and let M be a Turing machine with alphabet Σ. We say that M computes f if for any $w \in \Sigma^*$, there exists $M(w) = f(w)$. If such M exists, then f is *recursive function.*

□

The Turing machines are equivalent in computation power to arbitrarily general (recursive) computer programs. Turing machines can be thought of an algorithms for solving string-related problems. But, calculations are expressed as "1110+111".

To solve the problems with Turing machines, whose instances are mathematical objects, like, graphs, networks, numbers, etc, the first requirement is to represent an instance of the problem with a string. Then, the algorithm for a decision problem is a Turing machine that decides the corresponding language.

Copying a String: We would like to construct a TM which copies input string $w \in \{a, b\}^*$ to location after its current position, and separated by B symbol. As each symbol position is copied to a new location, its original position, say, a is replaced by X, and b by Y, to record that these have been copied. These are finally replaced by the original symbols. The final tape contents will be wBw. Figure 2.6 shows a TM, which produces the output wBw for an input of wB.

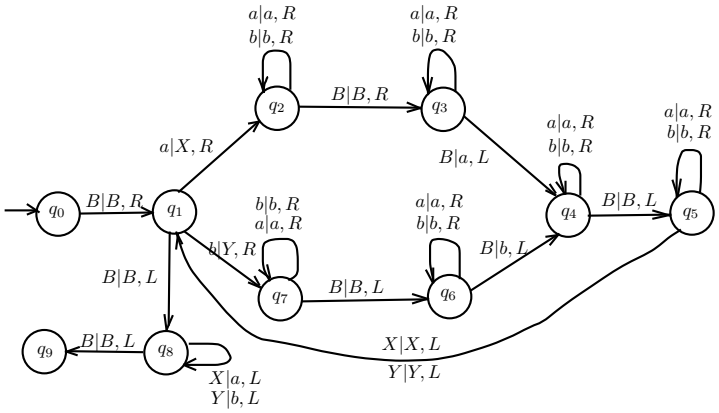

Figure 2.6: Copy a string w to make wBw.