Theory of Formal Languages (Turing Machine and Computability)

Lecture 3: Dec. 14, 2018

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

3.1 TM and Computability

We have represented the Turing machines as $M = (Q, \Sigma, \Gamma, \delta, q_0, H)$ and input as $w \in \Gamma^*$. This input results in finite computation sequence $C_0, C_1, C_2, \ldots, C_m$, where each C_i is a configuration of M and C_0 is initial configuration. If this Turing machine M halts and succeeds in C_m, then computation is a successful computation of length m. Then, M has computed a results given in the input w and result is $F_M(w)$. Since the machine M is deterministic then for same input $w = x$, $f_M(w) = f_M(x)$.

For some $w \in \Sigma^*$, $f_M(w)$ is not defined because M halts and fails or does not halt. This means that f_M is a *partial function* from Σ^* to Γ^* and f_M is a function computed by M. Then, we can say that a function $f : \Sigma^* \to \Gamma^*$ is *Turing Computable*, if and only if there is a Turing machine such that $f = f_M$.

Example 3.1 *Turing Computable function.*

The function $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined by $f(w, x) = w + x$ is Turing computable. A Turing machine can also be constructed to compute the function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined by $f(w, x) = w.x$. In the same line, any polynomial function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ can be computed by a Turing machine, hence the function is Turing computable.

Definition 3.2 Church-Turing Thesis. *A function can be computed by an algorithm iff it can be computed by a Turing machine.*

In addition to language acceptance, a TM may be viewed as computer of functions from integers to integers, that is,

$$f : \mathbb{N} \to \mathbb{N}. \tag{3.1}$$

The approaches use representation of integers in union for $i \geq 0$, where integer i is represented by a string 1^{i+1}. If a function has k arguments i_1, i_2, ..., i_k, these arguments are initially placed on the tape separated by 0's, in the form:

$$1^{i_1}01^{i_2}0 \ldots 01^{i_{k-1}}01^{i_k}. \tag{3.2}$$

If a TM halts in state q', with ID (Instantaneous description) $q'1^{m+1}$, then it shows that the value of m has been computed by function f as

$$f(i_1, i_2, ..., i_k) = m. \tag{3.3}$$

It may be noted that TM may compute a function of one argument, or a different function of two or more arguments. For a function of k arguments, all the k-tuples need not to have values. A function with all the arguments present is a *total recursive function*, where as a function with some of the tuples having values, is called *partial recursive function*. All the arithmetic functions, like $[\log n]$, n^1, 2^n etc., are total recursive functions.

The total recursive functions correspond to Recursive language, which always cause the TM to halt. Whereas the partial recursive functions are like Recursively Enumerable languages and they correspond to TMs, which may not halt on every input.

Example 3.3 *Given two positive integers x and y, construct a Turning Machine to computes the sum $x + y$.*

We first choose a convenient way to represent x and y on the tape. We use a unary notation so that weight of each number is represented by a sequence of 1's, i.e., $\{1\}^+$. Therefore $w(x), w(y) \in \{1\}^+$, $|w(x)| = x$, and $|w(y)| = y$.

It is assumed that $w(x)$ and $w(y)$ are placed on the tape in sequence with a 0 separating both. Initially the TM is in state q_0, and read-write head is positioned to first character of $w(x)$. When computation is over, the result $w(x + y)$ is placed on the tape with read-write head positioned to first character of the result $w(x + y)$ and the zero is placed after result. Therefore,

$$q_0 w(x) 0 w(y) \vdash^* q' w(x + y) 0,$$

where q' is a accepting or halting state. We note that the addition operation appears simple. What is required is to simply move the separating 0 to end, so that $w(x)$ and $w(y)$ are concatenated. This results to $w(x + y)$.

Let the TM be,

$$M = (Q, \Sigma, \Gamma, \delta, q_0, H)$$

where,

$Q = \{q_0, q_1, q_2, q_3, q_4\}$, $H = \{q_4\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, B\}$, with left most symbol as B, and

δ is specified as,

In the process of moving 0 from middle of $w(n)$ and $w(y)$ to right, 1 gets added extra temporarily. This is deleted later on.

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

Let us consider $x = 2$ and $y = 3$, hence $w(2) = 11$, and $w(3) = 111$. Various moves of TM can be given as follows.

$$\begin{aligned}
q_0110111 &\vdash B1q_010111B \\
&\vdash B11q_00111B \\
&\vdash B111q_1111B \\
&\vdash B1111q_111B \\
&\vdash B11111q_11B \\
&\vdash B111111q_1B \\
&\vdash B11111q_21B \\
&\vdash B1111q_310B \\
&\vdash B111q_3110B \\
&\vdash B11q_31110B \\
&\vdash B1q_311110B \\
&\vdash Bq_3111110B \\
&\vdash q_3B111110B \\
&\vdash Bq_4111110B
\end{aligned}$$

Thus the sum of the numbers 2 and 3 provides the result 5 (since $wt(5) = 11111$).

□

Example 3.4 *Show that function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined by $f(x, y) = x - y$ is Turing Computable.*

To demonstrate that the above subtraction function is Turing computable, we construct a Turing machine that computes $f(x, y)$. We represent the integers x and y as unary number. The approach shall be as follows: search the beginning of y and for each 1 in y go back to the start of x and delete a symbol 1. For subtraction to be possible, x should be grater that y.

□

Example 3.5 *Design a TM that multiplies two positive integers in unary notation.*

Let the configuration on the tape before and after the solution in as shown in figure 3.1. Considering it as a basic TM model, let the numbers to be multiplied are x and y, and their product is xy. Assume that x and y are represented by their weights, i.e., $w(x)$, $w(y)$, respectively. A typical case is shown in figure 3.1, for x, y as $4, 4$. The steps for multiplying x and y can be as follows.

1. Write a zero after y.
2. Replace each 1 in x by X, and for each such replacement copy y at the right of original $y0\alpha$. Here α is xy constructed so far, initially, null.
3. When all x's contents are X's, the result of $x \times y$ is available as $\alpha = xy$.

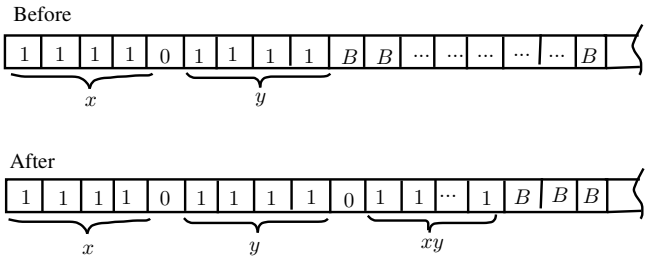

Figure 3.1: Performing Operation of Multiplication on TM.

□

Example 3.6 *Construct a TM that increments any binary string.*

Let the machine be $M = (Q, \Sigma, \Gamma, \delta, s, H)$, where $Q = \{q_0, q_1, q_3\}$, and $s = q_0$. Various transitions for this machine are given below.

- In q_0, the RW head moves to right until it reads a blank (B), and then switch to state q_1.
- In state q_1, move to left changing 1's to 0's until it reaches to a 0 or blank (B), changing it to 1.
- Since there are no transitions, so the machine halts at q_2 (see figure 3.2).

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

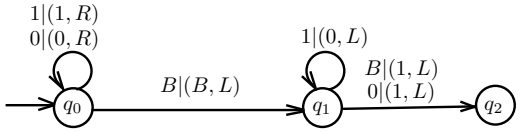

Figure 3.2: Turing Machine to increment a binary string.

□

Example 3.7 *Turing machine as function computer of concatenation.*

We compute the binary function of concatenation of strings over $\{a, b\}^*$. The initial configuration is q_0BuBvB. One or both or none of u and v may be null. The final configuration is q_fBuvB.

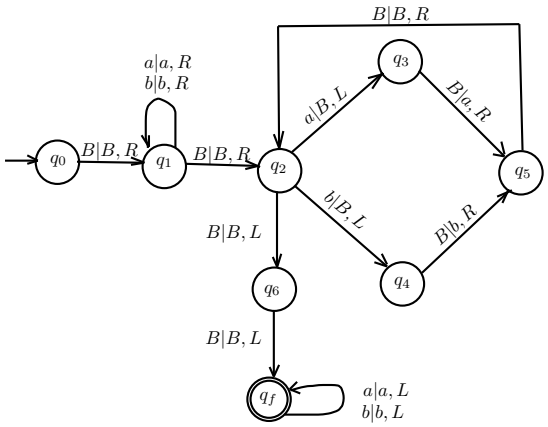

Figure 3.3: Turing Machine to concatenate strings.

□