Lecture 13: More on Turing Machines

9 March 2010

This lecture covers the formal definition of a Turing machine and related concepts such as configuration and Turing decidable. It surveys a range of variant forms of Turing machines and shows for one of them (multi-tape) why it is equivalent to the basic model.

1 A Turing machine

A ***Turing machine*** is a 7-tuple

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}),$$

where

- Q: finite set of states.
- Σ: finite input alphabet.
- Γ: finite tape alphabet.
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}$.
- $q_0 \in Q$ is the initial state.
- $q_{\mathrm{acc}} \in Q$ is the ***accepting***/***final*** state.
- $q_{\mathrm{rej}} \in Q$ is the ***rejecting*** state.

TM has a working space (i.e., tape) and its deterministic. It has a reading/writing head that can travel back and forth along the tape and rewrite the content on the tape. TM halts immediately when it enters the accept state (i.e., q_{acc}) and then it accepts the input, or when the TM enters the reject state (i.e., q_{rej}), and then it rejects the input.

Example 1.1 Here we describe a TM that takes it input on the tape, shifts it to the right by one character, and put a \$ on the leftmost position on the tape.

So, let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ (but the machine we describe would work for any alphabet). Let

$$Q = \{q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}\} \cup \left\{ q_c \;\middle|\; c \in \Sigma \right\}.$$

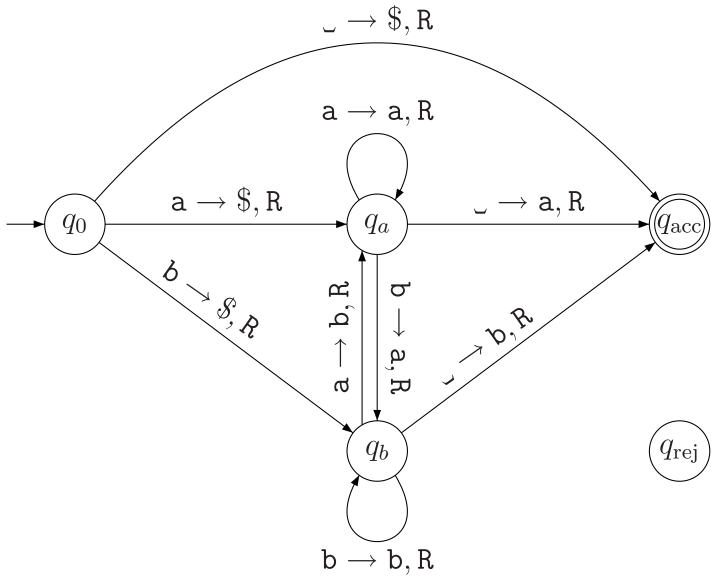


Figure 1: A TM that shifts its input right by one position, and inserts $ in the beginning of the tape.

Now, the transitions function is

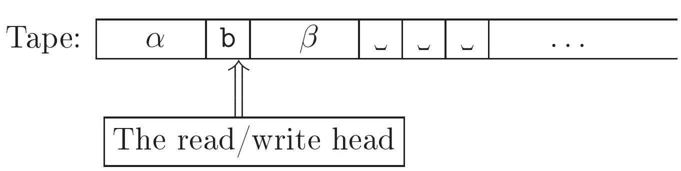
$$
\begin{aligned}
\forall s \in \Sigma \qquad & \delta(q_0, s) = (q_s, \$, \mathtt{R}) \\
\forall s, t \in \Sigma \qquad & \delta(q_s, t) = (q_t, s, \mathtt{R}) \\
\forall s \in \Sigma \qquad & \delta(q_s, \llcorner\!\lrcorner) = (q_{\text{acc}}, \$, \mathtt{R})\,. \\
& \delta(q_0, \llcorner\!\lrcorner) = (q_{\text{acc}}, \$, \mathtt{R})
\end{aligned}
$$

The resulting machine is depicted in Figure 1, and here its pseudo-code:

Shift_Tape_Right
At first tape position,
  remember character and write $
At later positions,
  remember character on tape,
  and write previously remembered character.
On blank, write remembered character and halt accepting.

2 Turing machine configurations

Consider a TM where the tape looks as follows,

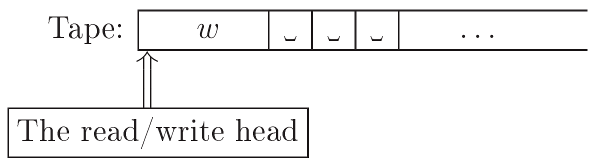


and the current control state of the TM is q_i. In this case, it would be convenient to write the TM ***configuration*** as

$$\alpha q_i \texttt{b} \beta.$$

Namely, imagine that the head is just to the left of the cell its reading/writing, and $\texttt{b}\beta$ is the string to the right of the head.

As such, the start ***configuration***, with a word w is



And this configuration is just $q_0 w$.

An ***accepting*** configuration for a TM is any configuration of the form $\alpha q_{\mathrm{acc}} \beta$.

We can now describe a transition of the TM using this configuration notation. Indeed, imagine the given TM is in a configuration $\alpha q_i \texttt{a} \beta$ and its transition is

$$\delta(q_i, \texttt{a}) = (q_j, \texttt{c}, \texttt{R}),$$

then the resulting configuration is $\alpha \texttt{c} q_j \beta$. We will write the resulting transition as

$$\alpha q_i \texttt{a} \beta \quad \Rightarrow \quad \alpha \texttt{c} q_j \beta.$$

Similarly, if the given TM is in a configuration

$$\gamma \ \texttt{d} \ q_k \ \texttt{e} \ \tau,$$

where γ and τ are two strings, and $\texttt{d}, \texttt{e} \in \Sigma$. Assume the TM transition in this case is

$$\delta(q_k, \texttt{e}) = (q_m, \texttt{f}, \texttt{L}),$$

then the resulting configuration is $\gamma \ q_m \ \texttt{d} \ \texttt{f} \ \tau$. We will write this transition as

$$\underbrace{\gamma \ \texttt{d} \ q_k \ \texttt{e} \ \tau}_{c} \quad \Rightarrow \quad \underbrace{\gamma \ q_m \ \texttt{d} \ \texttt{f} \ \tau}_{c'}.$$

In this case, we will say that c ***yields*** c', we will use the notation $c \mapsto c'$.

As we seen before, the ends of tape are special, as follows:

- You can not move off the tape from the left side. If the head is instructed to move to the left, it just stays where it is.
- The tape is padded on the right side with spaces (i.e., ␣). Namely, you can think about the tape as initially as being full with spaces (spaced out?), except for the input that is written on the beginning of the tape.

3 The languages recognized by Turing machines

Definition 3.1 For a TM M and a string w, the Turing machine M ***accepts*** w if there is a sequence of configurations

$$C_1, C_2, \ldots, C_k,$$

such that

(i) $C_1 = q_0 w$, where q_0 is the start state of M,

(ii) for all i, we have C_i yields C_{i+1} (using M transition function, naturally), and

(iii) C_k is an accepting configuration.

Definition 3.2 The ***language*** of a TM M (i.e., Turing machine M) is

$$L(M) = \left\{ w \;\middle|\; M \text{ accepts } w \right\}.$$

The language L is called ***Turing recognizable***.

Note, that if $w \in L(M)$ then M halts on w and accepts it. On the other hand, if $w \notin L(M)$ then either M halts and rejects w, or M loops forever on the input w. Specifically, for an input w a TM can either:

(a) accept (and then it halts),

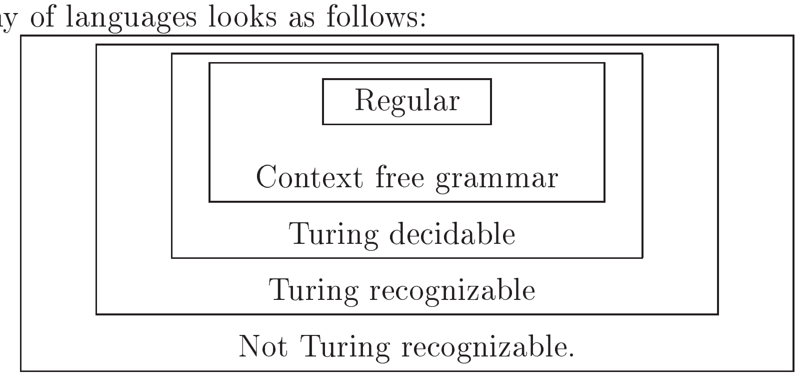
(b) reject (and then it halts),

(c) or be in an infinite loop.

Definition 3.3 A TM that halts on all inputs is called a ***decider***.

As such, a language L is ***Turing decidable*** if there is a decider TM M, such that $L(M) = L$.

The hierarchy of languages looks as follows:



4 Variations on Turing Machines

There are many variations on the definition of a Turing machine which do not change the languages that can be recognized. Well-known variations include doubly-infinite tapes, a stay-put option, non-determinism, and multiple tapes. Turing machines can also be built with very small alphabets by encoding symbol names in unary or binary.

4.1 Doubly infinite tape

What if we allow the Turing machine to have an infinite tape on both sides? It turns out the resulting machine is not stronger than the original machine. To see that, we will show that a doubly infinite tape TM can be simulated on the standard TM.

So, consider a TM M that uses a doubly infinite tape. We will simulate this machine by a standard TM. Indeed, fold the tape of M over itself, such that location $i \in [-\infty, \infty]$ is mapped to location

$$h(i) = \begin{cases} 2|i| & i \leq 0 \\ 2i - 1 & i > 0. \end{cases}$$

on the usual tape. Clearly, now the doubly infinite tape becomes the usual one-sided infinite tape, and we can easily simulate the original machine on this new machine. Indeed, as long as we are far from the folding point on the tape, all we need to do is to just move in jumps of two (i.e., move `L` is mapped into move `LL`). Now, if we reach the beginning of the tape, we need to change between odd location and even location, but that's also easy to do with a bit of care. We omit the easy but tedious details.

Another approach would be to keep the working part of the doubly-infinite tape in its original order. When the machine tries to move off the lefthand end, push everything to the right to make more space.

4.2 Allow the head to stay in the same place

Allowing the read/write head to stay in the same place is clearly not a significant extension, since we can easily simulate this ability by moving the head to the right, and then moving it back to the left. Formally, we allow transitions to be of the form

$$\delta(q, \mathtt{c}) = (q', \mathtt{d}, \mathtt{S}),$$

where `S` denotes the command for the read/write head to stay where it is (rewriting the character on the tape from `c` to `d`).

4.3 Non-determinism

This does not buy you anything, but the details are not trivial, and we will delay the discussion of this issue to later.

4.4 Multi-tape

Consider a TM that has k tapes, where $k > 1$ is a some finite integer constant. Here each tape has its own read/write head, but there is only one finite control. The transition function of this machine, is a function

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{\mathtt{L}, \mathtt{R}, \mathtt{S}\}^k ,$$

and the initial input is placed on the first tape.

5 Multiple tapes do not add any power

We next prove that one of these variations (multi-tape) is equivalent to a standard Turing machine. Proofs for most other variations are similar.

Claim 5.1 *A multi-tape* TM N *can be simulated by a standard* TM*.*

Proof: We will build a standard (single tape) TM simulating N.

Initially, the input w is written on the (only) tape of M. We rewrite the tape so that it contains k strings, each string matches the content of one of the tapes of N. Thus, the rewriting of the input, would result in a tape that looks like the following:

$$\$w \underbrace{\$ \sqcup \$ \sqcup \ldots \$ \sqcup}_{k-1 \text{times}} \$.$$

The string between the ith and $(i+1)$th \$ in this string, is going to be the content of the ith tape. We need to keep track on each of these tapes where the head is supposed to be. To this end, we create for each character $\mathtt{a} \in \Gamma$, we create a dotted version, for example $\boxed{\dot{\mathtt{a}}}$. Thus, if the initial input $w = xw'$, where x is a character, the new rewritten tape, would look like:

$$\$\dot{x}w' \underbrace{\$ \dot{\sqcup} \$ \dot{\sqcup} \ldots \$ \dot{\sqcup}}_{k-1 \text{times}} \$.$$

This way, we can keep track of the head location in each one of the tapes.

For each move of N, we go back on M to the beginning of the tape and scan the tape from left to right, reading all the dotted characters and store them (encoding them in the current state), once we did that, we know which transition of N needs to be executed:

$$q_{\langle c_1,\ldots,c_k \rangle} \quad \rightarrow \quad q'_{\langle d_1,D_1,d_2,D_2,\ldots,d_k,D_k \rangle},$$

where $D_i \in \{\mathtt{L}, \mathtt{R}, \mathtt{S}\}$ is the instruction where the ith head must move. To implement this transition, we scan the tape from left to right (first moving the head to the start of the tape), and when we encounter the ith dotted character c_i, we replace it by (the undotted) d_i, and we move the head as instructed by D_i, by rewriting the relevant character (immidiately near the head location) by its dotted version. After doing that, we continue the scan to the right, to perform the operation for the remaining $i+1, \ldots, k$ tapes.

After completing this process, we might have $\dot{\$}$ on the tape (i.e., the relevant head is located on the end of the space allocated to its tape). We use the **Shift_Tape_Right** algorithm we describe above, to create space to the left of such a dotted dollar, and write in the newly created spot a dotted space. Thus, if the tape locally looked like

$$\ldots \mathtt{ab} \dot{\$} \mathtt{c} \ldots$$

then after the shifting right and dotting the space, the new tape would look like

$$\ldots \mathtt{ab} \dot{\sqcup} \$\mathtt{c} \ldots$$

By doing this shift-right operation to all the dotted \$'s, we end up with a new tape that is guaranteed to have enough space if we decide to write new characters to any of the k tapes of N.

Its easy to now verify that we can now simulate N on this Turing machine M, which uses a single tape. In particular, any language that N recognizes is also recognized by M, which is a standard TM, establishing the claim. ∎