3P17/4P17 COMPUTABILITY 2014-2015 Lecture Handout 2

Examples of Turing machines (single-tape, deterministic)

In the following, we will be given tasks and design Turing machines to solve those tasks. We will give an informal description of how the machine works as well as a formal definition.

We will use the following notation to describe the execution of a computation. Suppose the TM is in state q, the tape contents is $\triangleright\mathtt{babcacb}\sqcup\sqcup\sqcup\ldots$ and the head is over the second $\mathtt{b}$ from the left. We will represent this configuration by $\begin{pmatrix} & & & q & & & & \\ \triangleright & \mathtt{b} & \mathtt{a} & \mathtt{b} & \mathtt{c} & \mathtt{a} & \mathtt{c} & \mathtt{b} \end{pmatrix}$

Example 1:

$\Sigma = \{\mathtt{0}, \mathtt{1}\}$. Task: Detect if the input $\sigma \in \Sigma^*$ contains $\mathtt{1}$.

The Turing machine M_1 works as follows: It will starting, as usual, with the read/write head over the first symbol of the input string σ, at each step it will do the following: If the symbol underneath the head is $\mathtt{1}$, then halt with accept . If it is $\mathtt{0}$, then move right and repeat. If it is $\sqcup$, then halt with reject.

$M_1 = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where $Q = \{q_0, q_{\text{acc}}, q_{\text{rej}}\}$, $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, $\Gamma = \{\mathtt{0}, \mathtt{1}, \triangleright, \sqcup\}$, $\delta(q_0, \mathtt{0}) = (q_0, \mathtt{0}, R)$, $\delta(q_0, \sqcup) = (q_{\text{rej}}, \sqcup, R)$, $\delta(q_0, \mathtt{1}) = (q_{\text{acc}}, \mathtt{1}, R)$, $\delta(q_0, \triangleright) = (q_{\text{rej}}, \sqcup, R)$.

Note: In $\delta(q_0, \sqcup) = (q_{\text{rej}}, \sqcup, R)$, the $\sqcup$ and R in the triple were arbitrary. We could just as well have defined $\delta(q_0, \sqcup) = (q_{\text{rej}}, \mathtt{1}, L)$, for example. If the TM executes $\delta(q_0, \sqcup)$, we will have known at that point that there are no $\mathtt{1}$'s in the input string. Therefore, the TM can replace the $\sqcup$ under the head with anything, and move the head in either direction, as long as the next state is q_{rej}. In this task we do not care about the tape contents after we have determined the answer. In other tasks, we may care. Similar applies to $\delta(q_0, \mathtt{1}) = (q_{\text{acc}}, \mathtt{1}, R)$

Note also: $\delta(q_0, \triangleright) = (q_{\text{rej}}, \sqcup, R)$ exists because a TM definition requires to define the transition function for all state-symbol pairs except when the state happens to be a halting state q_{acc} or q_{rej}. Regardless of what $\sigma \in \Sigma^*$ is, our TM here can never have its head over the $\triangleright$, because it starts off on the first symbol of the input (which is the one next to $\triangleright$), and $\triangleright$ is never part of the input alphabet, and it only ever moves right. So the specification of $\delta(q_0, \triangleright) = (q_{\text{rej}}, \sqcup, R)$ is, in a sense superfluous, but we include it to give a well-define TM.

Example execution on input $\mathtt{00101}$ expressed as a series of configurations: $\begin{pmatrix} & q_0 & & & & \\ \triangleright & \mathtt{0} & \mathtt{0} & \mathtt{1} & \mathtt{0} & \mathtt{1} \end{pmatrix}$, $\begin{pmatrix} & & q_0 & & & \\ \triangleright & \mathtt{0} & \mathtt{0} & \mathtt{1} & \mathtt{0} & \mathtt{1} \end{pmatrix}$, $\begin{pmatrix} & & & q_0 & & \\ \triangleright & \mathtt{0} & \mathtt{0} & \mathtt{1} & \mathtt{0} & \mathtt{1} \end{pmatrix}$, $\begin{pmatrix} & & & & q_{\text{acc}} & \\ \triangleright & \mathtt{0} & \mathtt{0} & \mathtt{1} & \mathtt{0} & \mathtt{1} \end{pmatrix}$.

Example 2:

$\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Task: compute the function $f(\sigma) = a\sigma$.

The Turing machine M_2 works as follows: Move the head to the first blank square. Move it left one to the right-most symbol. Replace that symbol, move one right and write that symbol in the square. Move left one. Repeat this until the head is over $\triangleright$. Move right one square and write a.

$\delta(q_0, \mathtt{a}) = (q_0, \mathtt{a}, R)$, $\delta(q_0, \mathtt{b}) = (q_0, \mathtt{b}, R)$ *Move head to the end*

$\delta(q_0, \sqcup) = (q_1, \sqcup, L)$ *Upon reaching the end, move left one and get read to copy*

At this point head may be over a *or* b

$\delta(q_1, \mathtt{a}) = (q_a, \sqcup, R)$, $\delta(q_1, \mathtt{b}) = (q_b, \sqcup, R)$ *If* a *cut* a*, if* b *cut* b

$\delta(q_a, \sqcup) = (q_0, \mathtt{a}, L)$, $\delta(q_b, \sqcup) = (q_0, \mathtt{b}, L)$ *Paste as appropriate, go left to repeat*

$\delta(q_1, \triangleright) = (q_2, \triangleright, R)$ *Got to the end, go back to write* a

$\delta(q_2, \sqcup) = (q_{\text{acc}}, \mathtt{a}, R)$ *Write* a*, halt with accept*

All other (state,symbol) pairs leave the symbol unchanged, go R and halt with q_{rej}.

$M_2 = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where $Q = \{q_0, q_{\text{acc}}, q_{\text{rej}}, q_1, q_2, q_a, q_b\}$ $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, $\Gamma = \{\triangleright, \sqcup, \mathtt{a}, \mathtt{b}\}$, δ as above.

Multitape Turing machines

A **multitape** Turing machine M is a TM with multiple tapes and an independent read-write head for each tape. It is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ with Q, Σ, Γ finite sets and $q_0, q_{\text{acc}}, q_{\text{rej}} \in Q$. These have the same meaning as in a single tape TM.

Γ is the alphabet for all tapes (i.e., any tape can use any of the symbols in Γ, though we may choose not to use all of symbols for a given tape).

The transition functions is changed to allow reading, writing, and moving heads on all or some tapes simultaneously:

$$\delta : Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\} \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k,$$

where k is the number of tapes. Here S means **stationary**.

Initially, the input is on tape 1 and the others are blank. Each tape head starts over the first square after $\triangleright$ on its respective tape.

Turing machines M_1 and M_2 are **equivalent** if for any input σ over the same alphabet Σ, M_1 and M_2 either both halt with accept on σ, both halt with reject on σ, or both do not halt on σ. We usually show equivalence by showing each can simulate the other.

Theorem 1. *Every multitape Turing machine has an equivalent single-tape Turing machine.*

Proof. Let M be a k-tape TM. We demonstrate a single tape TM M_1 which recognises the same language. M_1 simulates M. If Γ is the tape alphabet of M, M_1's tape alphabet Γ_1 includes Γ but also has $\widehat{\mathtt{x}}$ for any $\mathtt{x} \in \Gamma$, and delimiter symbol $*$ to separate simulated tapes on a single tape. On input $\sigma = \sigma_1 \sigma_2 \ldots \sigma_n$,

M_1 prints $\triangleright \widehat{\sigma_1} \sigma_2 \ldots \sigma_n * \widehat{\sqcup} * \widehat{\sqcup} * \ldots * \widehat{\sqcup} *$

M_1 simulates each move of M in two passes of the tape: first pass, scan through to determine symbols under $\widehat{\ }$, then second pass to update its tape in accordance with transition function δ of M.

If at any point M_1 moves one of the virtual heads to the right over $*$, it shifts it and everything after one position right on its tape and places $\widehat{\sqcup}$ instead. It then continues as before. $\square$

How would you design a (single- or multitape) Turing machine to recognise palindromes?