Notes on Turing Machines

Definition: A *Turing Machine* (TM) is a 3-tuple (Σ, k, δ):

1. Σ — a *finite* set (the alphabet), $\Sigma \supseteq \{0, 1, \triangleright, \emptyset\}$.
2. $k \in \mathbb{N}$ — the states (we just represent each state with a natural number, instead of using an arbitrary set like we did earlier with DFAs and NFAs)
3. δ: $[k] \times \Sigma \to [k] \times \Sigma \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}, \mathbf{H}\}$. The transition function takes a state and an alphabet symbol (read from the tape), and outputs a state, a symbol that is written on the tap, and either moves one square Left, moves one square Right, Stays, or Halts.

Execution of a TM

Definition: The execution of a TM, $M = (\Sigma, k, \delta)$ on input $x \in \{0,1\}^*$ is the result of this process:

1. Initialize T as $(\triangleright, x_0, x_1, \ldots, x_{n-1}, \emptyset, \emptyset, \emptyset, \ldots)$ where $n = |x|$.
2. Initialize two natural number variables, $i = 0$ (which represents the tape index) and $s = 0$ (which represents the current state).
3. Repeat until break is reached:
   (a) $(s', \sigma', D) = \delta(s, T[i])$.
   (b) $s{:=}s'$, $T[i]{:=}\sigma'$.
   (c) if $D = \mathbf{R}$: $i{:=}i + 1$
   if $D = \mathbf{L}$: $i{:=}\max\{i - 1, 0\}$
   if $D = \mathbf{H}$: **break**
4. If the process finishes (reaches the **break** to stop the repeat), the output is: $M(x) = T[1], \ldots, T[m]$ where $m > 0$ is the smallest integer such that $T[m + 1] \notin \{0, 1\}$. Otherwise (i.e., the machine does NOT halt), $M(x) = \perp$.

No-Input Variation

The execution of a TM, $M = (\Sigma, k, \delta)$ ~~on input $x \in \{0,1\}^*$~~ is the result of this process:

1. Initialize T as $(\triangleright, ~~x_0, x_1, \ldots, x_{n-1},~~ \emptyset, \emptyset, \emptyset, \ldots)$ ~~where $n = |x|$~~. Initialize T as $(\triangleright, \emptyset, \emptyset, \emptyset, \ldots)$.
2. ... (the rest of the definition is the same as above)

Self-Rejecting Language

For many definitions, including this one, we need a function that turns an arbitrary finite binary string into a Turing Machine (much of Turing's paper is showing, in a somewhat tedious way, that you can turn a TM description into a finite binary string, which now we are sufficiently comfortable with that we take this as obvious without needing to go through the steps). The difficulty in the reverse direction is that while there are many ways to map finite binary strings to TMs, none of them will be complete (that is, some finite binary strings will not correspond to a valid TM). To address this, we define the $\mathcal{M}(w) : \{0,1\}^* \to \mathit{TuringMachine}$ function that coverts an arbitrary binary string to a TM. It works for any $w \in \{0,1\}^*$ but is defined to produce a special "reject everything" machine for inputs w that do not correspond to valid Turing Machines:

$$\mathcal{M}(w) = \begin{cases} \text{TM described by } w & w \text{ corresponds to a valid Turing Machine} \\ \mathit{RejectMachine} & \text{Otherwise} \end{cases}$$

The self-rejecting language is defined as:

$$\mathit{SelfRejecting} = \{w \in \{0,1\}^* \mid w \notin \mathcal{L}(\mathcal{M}(w))$$