DUE: Tuesday, October 15

1. (4 points) Show that the language \((0+1)^*11(0+1)^*\) is recursive by describing a TM to accept it. Show the transition function \(\delta\); do not just use pseudocode. And show that the machine always enters state \(t\) or \(r\).

2. (20 points) Consider the following language which consists of Turing Machines which halt on some input

\[ L_H = \{M_i \mid (\exists z \in \Sigma^*) M_i \text{ halts on input } z\} \]

\(a\) Is \(L_H\) recursively enumerable? Justify your answer.

\(b\) Is \(L_H\) recursive? Justify your answer.
1. The language is accepted by a TM which goes right on the tape and enters state \( t \) if a 11 is ever encountered. If a blank symbol is encountered before encountering a 11 it enters state \( r \). Since the machine only goes to the right, we know it will either encounter a 11 or a blank symbol \( b \), so it will always accept or reject any input \( z \in \Sigma^* \).

Our Turing Machine moves right in state \( p \) until it encounters a 1, in which case it enters state \( q \) and keeps moving right. If in state \( q \) the next symbol is a 1, it accepts the string by entering state \( t \). If it makes it to the blank part of the tape not having encountered a 11, it enters state \( r \).

\[
\begin{align*}
\delta(s,\rhd) &= (p,\rhd, R) \\
\delta(p,0) &= (p,0, R) \\
\delta(p,1) &= (q,1, R) \\
\delta(q,0) &= (p,0, R) \\
\delta(q,1) &= (t,1, R) \\
\delta(p,b) &= \delta(q,b) = (r,b, R) \\
\delta(t,0) &= \delta(t,1) = \delta(t,b) = (t,0, R) \\
\delta(r,0) &= \delta(r,1) = \delta(r,b) = (r,0, R)
\end{align*}
\]

2. \( L_H \) is r.e. To enumerate it, we give all Turing Machines the chance to run for 1 step, then 2 steps, then..., by dovetailing over all TMs.

\[
\begin{array}{ccccccc}
&M_0 & 1 & 2 & 4 & 7 & 11 \\
&M_1 & 3 & 5 & 8 & 12 \\
&M_2 & 6 & 9 & 13 \\
&M_3 & 10 & 14
\end{array}
\]

For the \( t \) steps of the simulation of \( M_i \) we dovetail over the inputs and the number of steps that \( M_i \) is run on each input.

\[
\begin{array}{ccccccc}
&y_0 & 1 & 2 & 4 & 7 & 11 \\
&y_1 & 3 & 5 & 8 & 12 \\
&\Sigma^* & y_2 & 6 & 9 & 13 \\
&y_3 & 10 & 14
\end{array}
\]
If we determine that $M_i$ ever halts on any input, we list it. If $M_i$ halts on input $y_j$, then it does so in a finite number of steps, say $p$. Ultimately the $t$ steps allocated to $M_i$ will be large enough for input $y_j$ to be tested for $p$ steps.

**b** $L_H$ is not recursive. We know that the halting problem is not recursive, so we show how to reduce the halting problem to the problem of deciding on membership in $L_H$.

Assume that $L_H$ is recursive. There must be a total TM $M_H$ which decides for any TM $M_i$ whether $M_i$ halts on any input. For any TM $M_i$ and input $z$, we show how to use $M_H$ to decide if $M_i$ halts on input $z$. Design a new TM $M^*$ which takes as input $M_i$ and any input, and then erases the input and replaces it with $z$. TM $M^*$ halts on some input (in fact it halts on all inputs) if and only if $M_i$ halts on input $z$. So because we assumed that $L_H$ is recursive, we could decide if an arbitrary TM $M_i$ halts on input $z$, which we know is undecidable. So $L_H$ can not be recursive.