ITCS 532: W5 Homework Solutions

Rob Egrot

Q1

Let $L = \{\mathbf{code}(T) : T \text{ is a TM and } T(I) \text{ halts within 100 steps for some } I\}$. Prove that L is recursive.

- Only strings whose length is at most 100 are relevant.
- We use a Turing machine that simulates $T(I)$ on every string whose length is at most 100, while simultaneously keeping track of the number of simulated steps.
- If $T(I)$ halts within 100 steps then we accept. If $T(I)$ does not halt within 100 steps we move onto the next string.
- If we get through all the strings then we reject.

Q2

Let $L = \{\textbf{code}(T) : T$ is a TM and $T(I)$ halts within $4 \times \text{length}(I)$ steps for some input $I\}$. Prove L is r.e. then prove that L is not recursive by reducing the empty tape halting problem to the decision problem corresponding to L.

To show that L is r.e. we use the fact that the set Σ^* is r.e. and assume we can generate all strings in order. We use the following algorithm:

1. Generate the first string I.
2. Calculate the length of I.
3. Simulate $T(I)$ while tracking the number of simulated steps.
4. If $T(I)$ halts within $4 \times \text{length}(I)$ steps then accept.
5. If $T(I)$ does not halt within $4 \times \text{length}(I)$ steps then generate next I and go to step 2.

Q2

- Given an instance of ETHP T we want an algorithm that constructs a Turing machine T', such that $T(\epsilon)$ halts if and only if $T'(J)$ halts within $4\times\text{length}(J)$ steps for some $J \in \Sigma^*$.
- Define T' to be the machine that acts as follows:
  1. Erase the input.
  2. Move tape head back to first space.
  3. Do what T would do.
- Then:

$$
\begin{aligned}
T \text{ is a yes instance of ETHP} &\implies T(\epsilon) \text{ halts} \\
&\implies \text{There is } n \text{ such that } T(\epsilon) \text{ halts in } n \text{ steps} \\
&\implies T'(J) \text{ halts within } 4|J| \text{ when } |J| = n+2 \\
&\implies T' \text{ is a yes instance of } D_L.
\end{aligned}
$$

- Why within $4|J|$ steps?
- $2(n+2)+1+(n+2)+1+n = 2n+4+1+n+2+1+n = 4n+8 = 4(n+2) = 4|J|$.
- Conversely, if $T(\epsilon)$ does not halt then $T'(J)$ does not halt for any J, and so T' is trivially a no instance of D_L.

Q3

Let $\mathscr{L} = \{0, s\}$ where 0 is a constant, s is a unary function. Let Γ contain the following sentences:

1. $\forall n \neg(0 = s(n))$.
2. $\forall m \forall n((s(m) = s(n)) \rightarrow (m = n))$.

We want to define $+$ to be the standard addition function. Starting with $+(x, 0) = x$ use recursion with the s function to define $+(x, y)$ for general x and y.

- $+(x, 0) = x$.
- $+(x, s(y)) = s(+(x, y))$.

Q4

A function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is *computable* if there is a Turing machine M and an encoding system **code** $: \mathbb{N} \to \{0, 1\}^*$ such that

$$M(\mathbf{code}(m, n)) = \mathbf{code}(f(m, n)) \text{ for all } m, n \in \mathbb{N}.$$

Define the function $b : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ by $b(m, n)$ is the maximum number of steps a Turing machine with m states defined over the alphabet $\{0, 1\}$ can take in a halting computation on an input of length n.

Prove that b is not computable.

Q4

Suppose b is computable. Consider the following algorithm on input **code**(T, I):

1. Count the number of states of T. Call this x.
2. Calculate the length of I. Call this y.
3. Compute $b(x, y)$.
4. Simulate $T(I)$ while keeping track of number of simulated computation steps. If $T(I)$ halts within $b(x, y)$ steps then accept. If simulation passes $b(x, y)$ steps then we can reject, as we know $T(I)$ cannot halt after this point, by definition of b.

The above algorithm would solve the halting problem, which we know is impossible, so b cannot be computable.