CSE200: Computability and complexity
Homework set 1 solution

Shachar Lovett

1 Single tape Turing machine

We need to prove that the standard 3-tape Turing machine (with separate tapes for input, work and output) can be simulated by a single Turing machine. The idea is to interleave the three tapes on a single tape, and to add to the alphabet symbols to denote the location of the head.

Let M be a 3-tape Turing machine with alphabet Σ, which we assume contains a special blank symbol $\Box$. Let $\tilde{\Sigma}$ denote a copy of Σ where $\tilde{\sigma}$ will be used to denote that the reading head is in this location. Our single tape Turing machine M_1 will use alphabet $\Sigma_1 = (\Sigma \cup \tilde{\Sigma})^3$ where each cell contains three symbols, corresponding to each work tape in M.

Let us assume that the input is in the correct format, e.g. if $x_1, \ldots, x_n \in \Sigma^n$ is the input of M then the input for M_1 is

$$(\tilde{x_1}, \tilde{\Box}, \tilde{\Box}), (x_2, \Box, \Box), \ldots, (x_n, \Box, \Box), (\Box, \Box, \Box), \ldots$$

To simulate a single step of M, the machine M_1 will first scan the tape for the symbol under each of the three reading heads (one in each split of the alphabet) and store the values (as part of its internal state). It will then apply the transition function of M and get the update steps required to move to the next step. Specifically, for each tape $1, 2, 3$, which symbol to write, to which direction the reading head in this tape should move, and to which state of M should we transition. Then, M_1 will implement these by scanning its tape and applying these (in a straightforward way).

We note that if M uses time $T(n)$ and space $S(n)$ then M_1 will use time $O(T(n)S(n))$ and space $O(S(n))$.

2 Universal Turing machines

Let U be a universal Turing machine, e.g. $U(\langle M \rangle, x) = M(x)$. Lets assume as a model for M the single-tape Turing machine we analyzed in the first question with a fixed alphabet Σ. The description of M is given by

$$\langle M \rangle = |Q|;\ \delta : Q \times \Sigma \to Q \times \Sigma \times \{L, S, R\}.$$

Note that we cannot assume that the number of states of M is constant. We will assume U has several work tapes (this can then be simulated by any other equivalent model, as we saw already). We further assume without loss of generality that the initial and halting states for M are known in advance; e.g. $q_{init} = 0, q_{halt} = 1$. If we don't want to assume this, we can have them be part of the description of M.

- Work tape W_{state} will be used to store the current state of M. We initialize it as q_{init}.
- Work tape W_M will be used to store the content of the single tape of M. We initialize it by x and the reading head in its beginning. It will function as a complete simulation of the run of M.
- Work tape $W_{internal}$ will be used for internal computation of the simulator U.
- Work tape W_{output} will be used for the output.

The simulator U does the following in a loop until it halts (or if not, it loops forever):

- Let $q = W_{state}$ be the current state of M, and σ the current symbol under the reading head in W_M.
- Check if $q = q_{halt}$. If so, copy the contents of W_M to W_{output}; clear all the other tapes of U; and halt.
- Otherwise, we need to find the entry (q, σ) in the description of δ in order to simulate one step of M. In order to do so, we scan over all the entries in δ one by one until we find the correct entry. Note that while σ can be stored in the internal states of U, q cannot as its length is unbounded. Hence, we will do the comparisons to indices of entries by a bit-by-bit comparison.
- Once we found the correct entry, we copy the new state of M (given by the transition function) to the tape W_{state}; and we record in the internal state of U the new symbol to be written and the direction in which the head should move. We then scan over the tape W_M and implement these changes.

3 Rice's theorem

Let $COMP$ denote the set of computable partial functions and let $\mathcal{P} \subset \mathcal{COMP}$ be a property of partial function which is nontrivial, e.g. $\mathcal{P} \neq \emptyset, \mathcal{P} \neq \mathcal{COMP}$. We need to prove that given a description of a Turing machine $\langle M \rangle$, it is not computable to see whether the function it computes is in $\mathcal{P}$ or not. That is, we need to prove that the function

$$p(\langle M \rangle) = \begin{cases} 1 & \text{if } (x \mapsto M(x)) \in \mathcal{P} \\ 0 & \text{if } (x \mapsto M(x)) \notin \mathcal{P} \end{cases}$$

is not computable.

Say p was computable by a Turing machine M_p. We will show that we can use it to compute the halting problem, which will give us a contradiction. Let $u(x) = undefined$ be the partial function which outputs "undefined" for all inputs. Let us assume without loss of generality that $u \notin \mathcal{P}$. Otherwise, we can replace $\mathcal{P}$ by its complement, which is computable iff $\mathcal{P}$ is. Let $f \in \mathcal{P}$ be some function, computable by a Turing machine M_f. Note that for f to exist we need the assumption that $\mathcal{P}$ is neither empty nor the entire set of complete functions.

Now, given a description of a Turing machine $\langle M \rangle$ and input x, define a Turing machine $N_{M,x}$ which does the following. On input y, it

1. Simulates $M(x)$.
2. Outputs $M_f(y)$.

Note that $\langle N_{M,x} \rangle$ is computable given $\langle M \rangle, x$; and that $\langle N_{M,x} \rangle \in \mathcal{P}$ iff M halts on x. So, if we could decide membership in $\mathcal{P}$ we could also decide if M halts on x, e.g. decide the halting problem. Contradiction.

4 Randomized Turing machines

Let M be a randomized Turing machine running in time $T(n)$. It uses at most $T(n)$ random bits in its run. We can make these random bits explicit: define $M(x,r)$ to be the output of running M of input x with random bits $r \in \{0,1\}^{T(n)}$. Note that $M(x,r)$ is now a deterministic Turing machine, and M (as a randomized Turing machine) computes a boolean function f if for any input x,

$$\Pr_{r \in \{0,1\}^{T(n)}}[M(x,r) = f(x)] \geq 2/3.$$

In order to derandomize this, let $F(x) = |\{r \in \{0,1\}^{T(n)} : M(x,r) = 0\}|$ be the number of random strings making $M(x,r)$ evaluate to zero. We can compute $F(x)$ in time $O(T(n)2^{T(n)})$ by running over all options for r, for each option simulating $M(x,r)$, and keeping a counter for the total number of random strings r which made $M(x,r)$ evaluate to zero. Note that the space requirements are just $O(T(n) + S(n))$ since we can use the same space to simulate each individual run of $M(x,r)$. Moreover, given $F(x)$ we can compute $f(x)$, as

$$f(x) = \begin{cases} 0 & \text{if } F(x) \geq (2/3) \cdot 2^{T(n)} \\ 1 & \text{if } F(x) \leq (1/3) \cdot 2^{T(n)} \end{cases}$$