Exercises for Computability and Complexity, Spring 2017, Sheet 5 – Solution

Please return on Thursday, March 16, in class.

This problem sheet features only a single problem, which is ***optional.*** *It is the infamous problem that I placed on the B group miniquiz 1 sheets. The problem hard, but it can be solved with the ideas that were presented in this course so far. I am curious whether somebody will go for it. I will award bonus points for insightful treatments (need not necessarily be complete proofs).*

Problem 1 (optional) Prove the following claim: If L is recursively enumerable but not recursive, then there exists another language L' which is likewise r.e. but not recursive, such that $L \cup L'$ is recursive.

Solution. Let $L \subseteq \Sigma^*$ be recursively enumerable but not recursive, and M a Turing machine that accepts it. From M we construct another TM M' which accepts a language L' such that L' is r.e. but not recursive, and furthermore $L \cup L' = \Sigma^*$, i.e. this is recursive.

Let $(w_n)_{n = 1, 2, ...}$ be the alphabetical enumeration of Σ^*, and for $w \in \Sigma^*$, let $I(w)$ be the index of w in this enumeration.

We first show that there is a totally defined, recursive function $f: \mathbb{N} \to \mathbb{N}$, such that there exist infinitely many $v \in L$ where M needs at most $f(I(v))$ steps to accept v. One way to obtain such f goes like this:

Initialize $p = 0$.

By a dovetailing scheme, simulate M first for 1 step on w_1, then for 2 steps on w_1 and w_2, ... etc, – in the k-dovetail run, for k steps on w_1 to w_k. Whenever this simulation finds that M accepts w_l in m steps, and l is greater than p, set $f(n) = m$ for all $p \leq n \leq l$. Update p to l.

It is straightforward to show that f is total recursive and there exist infinitely many $v \in L$ where M needs at most $f(I(v))$ steps to accept v.

Using f we construct M' as follows. On input w, M' simulates M for at most $f(I(w))$ steps. If M does not accept w within this time, then M' accepts w (from this it follows that $L \cup L' = \Sigma^*$). If M accepts w within this time, M' first computes the number $k(w) = | \{i \leq I(w) \mid \text{runtime of } M \text{ on input } w_i \text{ is at most } f(i)\} |$ (in order to compute k, M' has to simulate M on all words v that come before w in the alphabetical enumeration, but only up to $f(I(v))$ steps). Then M' simulates M on input w_k. It is easy to see that in this way, M' simulates M on all words $u \in \Sigma^*$, ultimately running the simulation of M on u_i when M' is started on that w that has $k(w) = i$. When M accepts input w_k, M' accepts too (namely its original input w); otherwise M', simulating M, runs forever. The language L' thus accepted by M' is not recursive, because if it would be, then L could be decided with the use of M' (how? an extra little sub-exercise).