Exercises for Computability and Complexity, Spring 2017, Sheet 6 – Solutions

Please return your solutions in the Thursday lecture on March 23.

Exercise 1 Consider the set ***T*** of all single-tape TMs with tape alphabet $\Sigma = \{0, 1, ⊔, \triangleright\}$. Design a coding scheme by which every TM *M* in ***T*** becomes coded by a codeword <*M*> $\in \{0, 1, \#\}^*$. Describe your coding scheme in formal notation and use it to encode the the ultra-simple TM *M* with tape alphabet $\Sigma = \{0, 1, ⊔, \triangleright\}$ and states $\{s, yes, no\}$ that has the following transition table:

$p \in K$	$\sigma \in \Sigma$	$\delta(q, \sigma)$
s	0	(*yes*,0, –)
s	1	(*s*,1, →)
s	⊔	(*no*, ⊔, –)
s	▷	(*s*, ▷, →)

Solution. Here is one of a zillion of possibilities: Let $M \in$ ***T*** have l states $s_1, s_2, ..., s_l$ (including the *h*, yes, no states). Encode a state s_i by $<s_i> :=$ #bin(*i*), where bin(*i*) is the binary representation of the number *i*. Encode the three cursor move directions by <→> = #01, <←> = #10, <–> = #00, and tape symbols by <0> = #0, <1> = #1, <⊔> = #00, <▷> = #11. Let $R = s_i\ \sigma\ (s_j, \sigma, d)$ be a row in a transition table Tab(*M*) of *M*, where *d* is one of →, ←, –. Code *R* by $<R> = <s_i><\sigma><s_j><\sigma><d>$. Let $R_1, ..., R_m$ be the rows of Tab(*M*). Code the transition table by $<\text{Tab}(M)> = <R_1>...<R_m>$ and we are done, because the TM is uniquely specified by this table. For the example, put $s_1 = s$, $s_2 = h$, $s_3 =$ "yes", $s_4 =$ "no", leading to <*s*> = #1, <*h*> = #2, etc. Then the four rows given in the table in Exercise 1 translate to

<*M*> = #1#0#11#0#00#1#1#1#1#01#1#00#100#00#00#1#11#1#11#01

Exercise 2 (rather easy) Prove that $H_2 = \{<M>;x \mid Code(<M>)$ and $Standard(x)$ and there exists some y with $Standard(y)$ such that $M(x) = y\}$ from Proposition 6.3 is undecidable.

Solution. Take any word <*N*> with *Code*(<*N*>). We can effectively construct a TM $K_{<N>}$ with tape alphabet {0, 1, #} which, for all inputs $x \in \{0, 1, \#\}^*$, yields the following result:

$$K_{<N>}(x) \quad = \quad \text{if } N(x) \text{ halts then } K_{<N>}(x) = 1, \text{ else } K_{<N>}(x) = \nearrow$$

($K_{<N>}$ simply simulates *N*(*x*), and if this halts, $K_{<N>}$ erases its tape and writes a 1, then halts). It clearly holds that *N*(*x*) halts iff there exists some *y* such that $K_{<N>}(x) = y$. (namely, $y = 1$), which in turn is equivalent with $<K_{<N>}>;x \in H_2$. If H_2 were decidable, so would *H*, mission impossible.

Exercise 3 (medium difficult) Show that the language

$$L = \{<M> \in \{0, 1, \#\}^* \mid M \text{ halts on no input}\}$$

is not recursively enumerable. *Hint: in addition to a reduction argument, you might wish to also work in Proposition 3.1 from the lecture notes.*

Solution. First consider the complement language

L^c = {$w \in$ {0, 1, #}* | w is not a codeword w = <M> for any TM M, or w is a codeword
w = <M> for some TM M, and M halts on some input}

L^c is recursively enumerable: it can be accepted by a TM N which first checks whether w is a valid TM codeword. If no, N immediately accepts. If yes, that is, if w = <M>, N simulates M on all input words $<x_1>, <x_2>, \ldots$ in a "dovetailing" fashion, that is, N first simulates M on input x_1 for k steps, then on inputs x_1 and x_2 for $2k$ steps each, then on inputs x_1, x_2 and x_3 for $3k$ steps, etc. If in one of these stages M is found to halt, N accepts.

Now if L would be recursively enumerable too, then L would be decidable. This can be seen, e.g., by reducing the language H_0 = {<M> | *Code*(<M>) and M halts on the empty input} from the lecture notes to L: assume L is decidable. Modify M, obtaining M' such that M' behaves like M on the empty input and runs into infinity on any nonempty input. Then, <M'> $\in L$ iff <M> $\in H_0$, thus we could decide H_0, contradiction.