

CS 132 Homework 2 Solutions

January 18, 2005

- 9.45 Show that if there is a TM $T = (Q, 1, \Lambda, q_0, \delta)$ computing the function $f : \mathcal{N} \rightarrow \mathcal{N}$, then there is another one, T' , whose tape alphabet is $\{1\}$.

Let tape alphabet of T be $\Lambda = \{a_1, \dots, a_n\}$. We will use an encoding function, e , such that

$$e(a_i) = 1^i \Delta^{n+1-i}$$

and

$$e(\Delta) = \Delta^{n+1}.$$

Now we can construct a TM $T' = (Q', \{1\}, \{1\}, q'_0, \delta')$. Q' will contain q_0, \dots, q_n for every $Q \in Q$; the halt states are $h_{a,n}$ and $h_{r,n}$. δ' will be such that for any transition $\delta(r, X) = (r', Y, D)$ in T ,

$$\delta'^n(r_0, e(X)) = (r'_0, e(Y), D),$$

where δ'^n indicates n applications of δ' to the successive symbols in $e(X)$. Finally, since both input and output comes in the form of 1^k , we must translate the input to $e(1^k)$, and translate the output back into all 1s.

It is easy to show by induction that iff $\delta^*(q_0, x) = (h_a, y, D)$ then $\delta'^*(q_0, e(x)) = (h_a, e(y), D)$. Thus when we add the encoding and decoding to T' it will compute exactly the same function.

- 10.3 Is the following statement true or false? If L_1, L_2, \dots are recursively enumerable subsets of Σ^* , then $\bigcup_{i=1}^{\infty} L_i$ is recursively enumerable. Give reasons for your answer.

False. Let $L = \{x_1, x_2, \dots\}$ be a language that is not recursively enumerable. By itself, each word in L , is a recursively enumerable language, namely $\{x_i\}$ (which is finite and hence regular); but by assumption $\bigcup_{i=1}^{\infty} x_i = L$ is uncomputable.

- 10.4 Suppose L_1, \dots, L_k form a partition of Σ^* ; in other words, their union is Σ^* and any two are disjoint. Show that if each L_i is recursively enumerable, then each L_i is recursive.

There are at least two methods of proving this. One is to use machines accepting L_1, \dots, L_k to create a machine that *decides* L_i . To do this we simulate all machines in parallel, one step at a time. We then accept immediately if T_i accepts, and reject immediately if $T_j, j \neq i$ accepts. The details of such a parallel simulation are numerous and can be taken from theorem 9.2.

A slightly simpler proof is the following: L_i is r.e. by assumption; $\overline{L_i} = \bigcup_{j \neq i} L_j$ is a union of r.e. languages; so by theorem 10.3, $\overline{L_i}$ is r.e. Since both L_i and its complement are r.e., by theorem 10.5 both are recursive.

10.5 Prove theorem 10.7, which says that a language is recursive if and only if there is a Turing machine enumerating it in canonical order.

First we will show that if there is a TM T that enumerates a language L in canonical order, then L is recursive. To do this we will use T to construct T' , a TM that will decide L . Given a string x , T' will act as follows: it will simulate T until T writes a $\#$ to the output tape, denoting the end of a newly outputted string. Call this new string y . If $x = y$, T' halts and accepts; if y is higher in the canonical ordering of Σ^* than x , T' halts and rejects; otherwise T' resumes the simulation of T and repeats the process. Since there are a finite number of strings lower in the canonical ordering than x , this process must halt after a finite number of rounds.

Now we will show that if L is recursive there exists a TM T that enumerates it in canonical order. Since L is recursive let T' be the TM which decides it; we will use this TM to construct T . T will have three tapes: tape one will store the enumerated output, tape two will store the current input, and tape three will hold a simulation of T' . T will begin with the encoding of T' on tape three, and the other two tapes blanks. T will then repeat the following: (1) simulate T' on the contents of tape two; (2) if T' accepts the current input on tape two, copy it to tape one followed by ' $\#$ '; (3) replace the contents of tape two with the next word in the canonical ordering of Σ^* . Clearly, T thus writes only those words which T' accepts to tape one, and does so in canonical order.

10.8 (Extra Credit) Describe algorithms to enumerate these sets.

- a. The set of all pairs (n, m) for which n and m are relatively prime positive integers.
- b. The set of all strings over $\{0, 1\}$ that contain a non-null substring of the form www .

Both of these sets are clearly recursive, so let T_a be a TM which computes the characteristic function of the set in part a, and T_b a machine that computes the characteristic function of strings in part b. We can easily enumerate these sets by successively simulating T_a or T_b on all strings in Σ^* , and outputting those for which T_a or T_b outputs a 1.

- c. Only $n = \{1, 2\}$ satisfy $x^n + y^n = z^n$, so this outputting this set is easy. (See <http://mathworld.wolfram.com/FermatsLastTheorem.html> for details).