CSc 30400 Introduction to Theory of Computer Science

4th Homework Set - Solutions

Due Date: 4/22

1 Practice questions

P 1. The TM decides the language $L = \{0^{2^n} : n \geq 0\}$.

(a) Rejects

(b) Rejects

(c) Accepts

2 Easy questions

E 1. The transition diagrams are shown on figures 1, 2, 3,

E 2. Repeat:

- Erase and remember* the first symbol before the # which has not been erased yet.
- Compare it with the first symbol after the # which has not been erased yet. If they are the same erase it, move to the beginning of the input and start over.

If everything but the # is erased accept else reject.

*: We can remember that we saw a 0 or an 1 by moving to states q_0 or q_1 respectively.

E 3. We use book's model to address this question. Remember that in the book's model there is a unique accept and a unique reject state which when reached the computation stops automatically and the machine either accepts or rejects respectively.

Suppose that L is decidable. Then there is a Turing Machine M deciding L. We create a Turing Machine M' as follows: Exchange the accept with the reject state. M' should accept for every input not in L and reject for every input in L. Thus M' decides L^c which means that the complement of L is also decidable.

E 4. (12 points) True or false? (Justify your answer)

(a) False. For example $\mathbb{N} \subsetneq \mathbb{Z}$ but $|\mathbb{N}| = |\mathbb{Z}|$. This statement is only true for finite sets.

(b) True. Since B is countable there is an enumeration for B. Start numbering the elements of A one by one following the enumeration for B while skipping the elements not belonging in B. This is an effective enumeration of all elements of A.

(c) True. Start numbering elements of A and B alternatively following the enumerations for A and B. Skip those elements belonging to the intersection which have already received a number.

(d) False. The set of predicates which is the set of functions from $\mathbb{N}$ (a countable set) to $\{0, 1\}$ (a finite set) is uncountable.

E 5. No, this is not doable. By the Church-Turing thesis any problem which admits a solution via a program can also be decided by a Turing Machine and vice versa. However the particular problem that the question describes in terms of Turing Machines is the equivalence problem: given two Turing Machines M_1, M_2, decide whether $\mathcal{L}(M_1) = \mathcal{L}(M_2)$). This problem is known to be undecidable. Thus there is no way that we can create a program for that purpose.

3 Hard questions

H 1.

- A LR-DTM is by definition a very special case of a DTM: it is a DTM with only left and right moves. Thus any LR-DTM is at the same time an ordinary DTM.
- Given a DTM M_S we need to create an equivalent LR-DTM M. The new machine M has to simulate all the “stay” transitions of M_S without using the “stay” ability. Thus we should replace each “stay” transition with one right and one left.

High-level instructions for a LR-DTM simulating a "stay" move:

- Overwrite the current symbol and move the head to the right.
- Without changing the contents move the head to the left.

Low-level description of the transformation: Suppose that $M_S = (Q, \Sigma, \Gamma, \delta, q_0, q_f)$. We create $M = (Q', \Sigma, \Gamma, \delta', q_0, q_f)$, where Q' contains all the states that Q contains and some more and δ' contains every "left" and "right" transition that δ contains and some more transitions as described below. Suppose that the DTM M_S has transitions of the form $\delta(q, a) = (q', b, S)$ for some $q, q' \in Q$ and $a, b \in \Gamma$. For each of those transitions, add in Q' a new state q'' ($q'' \notin Q$). Add the following transitions: $\forall c \in \Gamma$, $\delta'(q'', c) = (q', c, L)$ and last $\delta'(q, a) = (q'', b, R)$.

The right transition should overwrite the current symbol and then move the head to the right while moving from state q to state q''. The left one should place the head back to the initial box while moving from q'' to q'. We should also make sure that the left move doesn't affect the contents of the tape. q'' takes care of that! We have to add the transitions $\delta'(q'', c) = (q', c, L)$, to make sure that whatever symbol is written in the right box will take us back to the left box on state q' and furthermore that this symbol will remain unchanged.

H 2. (a) We already know that $\mathbb{N} \times \mathbb{N}$ is countable. Thus there is an one-to-one and onto function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We define an one-to-one and onto function $g : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ as follows: $\forall n_1, n_2, n_3 \in \mathbb{N}$, $g(n_1, n_2, n_3) = f(n_1, f(n_2, n_3))$. The fact that g is indeed one-to-one and onto comes from f being one-to-one and onto:

- For every number $n \in \mathbb{N}$ there exist two numbers $x, y \in \mathbb{N}$ such that $n = f(x, y)$ (since f is onto). Of course, for the same reason, for every number $y \in \mathbb{N}$ there exist two numbers $w, z \in \mathbb{N}$ such that $y = f(w, z)$. Thus, for every number $n \in \mathbb{N}$ there exists a triple $(x, w, z) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ such that $n = f(x, f(w, z)) = g(x, w, z)$.
- Suppose that, for two triples $(m_1, m_2, m_3), (n_1, n_2, n_3)$ it is that $g(m_1, m_2, m_3) = g(n_1, n_2, n_3)$. Thus $f(m_1, f(m_2, m_3)) = f(n_1, f(n_2, n_3))$. Since f is one-to-one we have that $m_1 = n_1$ and $f(m_2, m_3) = f(n_2, n_3)$. Again because of f being one-to-one we have that $m_2 = n_2$ and $m_3 = n_3$. Thus $(m_1, m_2, m_3) = (n_1, n_2, n_3)$.

(b) We give an one-to-one and onto function f mapping $2^{\mathbb{N}}$ to the uncountable set $\{0,1\}^{\mathbb{N}}$ of predicates. For every $A \subseteq \mathbb{N}$, $f(A) = \chi_A$, i.e. the characteristic function of the set A. f is one-to-one since different subsets have different characteristic functions. Also it is onto since every predicate can be formulated as a characteristic function of some subset of $\mathbb{N}$.

H 3. (a) Yes, this is a correct proof. Since L_1, L_2 are both decidable M_1 and M_2 are going to halt on every input. If M_1 accepts on input s then $s \in L_1$ thus in $L_1 \cup L_2$. So we should accept. If M_1 rejects then the decision depends on M_2: If it accepts then $s \in L_2$ thus $s \in L_1 \cup L_2$, so we should accept. Otherwise $s \notin L_1$ and $s \notin L_2$ thus $s \notin L_1 \cup L_2$ which means that we should reject.

(b) This is not a correct proof. Consider the case where M_1 loops on input s but M_2 accepts s. The instructions suggest that M_2 never begins the execution and M is going to loop, thus there is no way that s can be accepted by M. But $s \in L_2$ so $s \in L_1 \cup L_2$ which is a contradiction.

H 4. $Halt_{1000}$ is a computable predicate. We need to change the universal Turing Machine a bit and add one extra tape which is going to countdown the remaining steps. On input $(< M >, x)$ the new machine U' should simulate M running on input x while counting the number of performed transitions. If M halts on input x reply yes. If the additional tape of U' counting the steps empties and M has not halted then reply no.

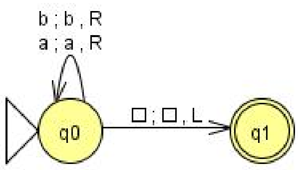

Figure 1: A TM for question E 1 a.

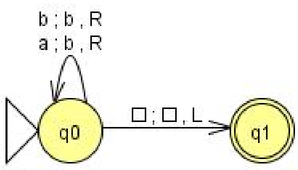

Figure 2: A TM for question E 1 b.

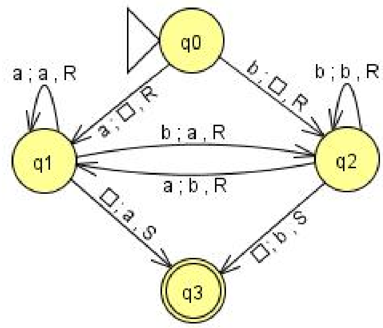

Figure 3: A TM for question E 1 c.