Computational Complexity - Midterm (Solutions)

M. Mladenovski
LDCSEE,
West Virginia University,
Morgantown, WV
{martinm@csee.wvu.edu}

X. Wang
LDCSEE,
West Virginia University,
Morgantown, WV
{xiawang@csee.wvu.edu}

1. Show that $NL \subseteq P$. (5 points)

   Proof: Observe that NL=NSPACE(log(n)). The number of possible configurations in space log(n) is $C^{\log(n)}$, where C is a constant. The time required for all of these configurations to be explored is $C^{\log(n)}$. For some k, $C = 2^k$. $C^{\log(n)} = 2^{k \cdot \log(n)} = 2^{\log(n^k)} = n^k$. Therefore NL⊆P. □

2. Let $L = \{< e, x > | \exists y, \phi_e(x) = y,\ i.e.,\ machine\ M_e\ when\ started\ with\ x\ halts\ with\ y\ as\ output.\}$. Is L decidable? Explain. (5 points)

   Proof: The language L is undecidable. To show this, it is sufficient to make a many-to-one reduction from the language $L_u = \{(e, x) \mid M_e\, accepts\, x\}$ to the language L. The function that reduces L_u to L is: $f(< e, x >) =< e, x >$, which is the identity function. Also the condition "$M_e\, accepts\, x$" from the language L_u is identical with the condition "$\exists y$, $\phi_e(x) = y$ i.e. machine M_e when started with x halts with y as output" from language L. Since the language L_u is undecidable, it follows that the language L is undecidable. □

3. Let $L = \{< e > | M_e\ writes\ a\ non-blank\ symbol\ at\ least\ once\ when\ started\ on\ a\ blank\ tape.\}$. Is L decidable? Explain. (5 points)

   Solution: From the transition functions of the Turing Machine with Goedel number e, it is easy to create a graph in the following way. First the initial state is added like a vertex in the graph. Next, from the initial state we add to the graph only those states (as vertices) that we can reach with a blank input tape. And so on, for every newly added state we do the same thing. If during one of these transitions something will be written on the tape, we say ACCEPT. When the graph is completed (the number of states in the Turing machine is finite) there are two possibilities, the Turing machine goes to the accepting state (accepting state vertex is reachable from the initial vertex) without writing a non-blank symbol, or loops forever (a cycle exists in the graph) without writing anything. In both cases we say REJECT. Thus this problem is decidable. □

4. Show that $A \leq_m B \Rightarrow A \leq_T B$. (4 points)

   Proof: Since $A \leq_m B$, there exists a computable function f such that $x \in A$ if and only if $f(x) \in B$. Define Turing machine M^B with oracle B that works as follows. Given input x, compute $f(x)$. M^B accepts x if $f(x) \in B$, otherwise it rejects. It is easy to see that M^B decides A; therefore $A \leq_T B$. □

5. Prove or disprove: *If A and B are c.e., then so are $A \bigcup B$ and $A \bigcap B$.* (3 points)

   Proof: Since A and B are c.e., there exist partial functions f_1 and f_2 such that $dom(f_1) = A$ and $dom(f_2) = B$. Define partial functions g_1 and g_2 such that:

$$\begin{aligned} g_1(x) &= 1,\ if\ \ f_1(x)\downarrow\ \ or\ \ f_2(x)\downarrow \\ &= \uparrow,\ otherwise. \end{aligned}$$

and

$$\begin{aligned} g_2(x) &= 1, \ if \ \ f_1(x)\downarrow \ \ and \ \ f_2(x)\downarrow; \\ &= \uparrow, \ otherwise. \end{aligned}$$

It is easy to see that $dom(g_1) = A \cup B$ and $dom(g_2) = A \cap B$. Therefore, both $A \cup B$ and $A \cap B$ are c.e. □

6. Prof. Roberts invents a Turing Machine abstraction that has 3 possible moves at each step, viz., $\{\leftarrow, \rightarrow, \uparrow\}$, where the $\uparrow$ indicates that the head does not move at all. Demonstrate to Professor Roberts, that his abstraction is not more powerful than the standard Turing Machine. (3 points)

   Proof: Let M be one of Prof. Roberts' Turing Machines. We can show that M can be simulated by a standard Turing Machine N. For each step of M, if M moves to the left or the right, then N behaves exactly as M; if M does not move at all, then N moves two steps such that for the first step, N moves to the right without writing down any symbol or state transition, and for the second step, N moves back to the left, writes down whatever M wrote and switches to the same state as M. Therefore, M is no more powerful than a standard Turing Machine. □

7. Let A and B be disjoint *c.e.* sets. Show that $A \leq_T A \bigcup B$. What happens if A and B are not disjoint? (5 points)

   Proof: Since A and B are *c.e.*, there exist Turing Machines M_A and M_B such that A is the language of M_A and B is the language of M_B. In case that $A \cap B = \emptyset$, we define a Turing Machine $M^{A \cup B}$ with oracle $A \cup B$ that works as follows. Given a word x, if $x \in A \cup B$ then $M^{A \cup B}$ halts and rejects. Otherwise each step of $M^{A \cup B}$ simulates one step of M_A and one step of M_B. If M_A halts, then $M^{A \cup B}$ halts and accepts. If M_B halts, then $M^{A \cup B}$ halts and rejects. We can see that A is decided by $M^{A \cup B}$ if A and B are disjoint. In case that $A \cap B \neq \emptyset$, $M^{A \cup B}$ does not necessarily decide A. Given $x \in A \cap B$, if M_B halts first, then $M^{A \cup B}$ will falsely reject x. □

1 Extra Credit

1. Show that $\exists A, B[A \leq_T B \ \ and \ \ A \not\leq_m B]$. (6 points)

   Proof: Let $K = \{x | \phi_x(x)\downarrow\}$. And let $A = \bar{K}$ and $B = K$. Then $A \leq_T B$ since for all sets A, $A \leq_T \bar{A}$. But $A \not\leq_m B$ because of the fact that K is c.e. and $\bar{K}$ is not c.e. contradicts Lemma 3.2. □