Computational Complexity - Quiz I (Solutions)

L. Kovalchick
LDCSEE,
West Virginia University,
Morgantown, WV
{lynn@csee.wvu.edu}

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

1. Design a Deterministic Turing Machine that accepts the regular language $10^* + 01^*$, i.e., the set of strings in which the first symbol does not appear again on the input. You may assume that $\Sigma = \{0, 1\}$. Feel free to choose the tape symbols. (4 points)

   Solution: $M =< Q, \Sigma, \Gamma, \delta, q_0, B, q_{accept}, q_{reject} >$, where: $Q = \{q_0, q_1, q_2, q_{accept}, q_{reject}\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, B\}$, and $\delta =$

state	0	1	B
q_0	$< q_1, 0, R >$	$< q_2, 1, R >$	$< q_{reject}, -, - >$
q_1	$< q_{reject}, -, - >$	$< q_1, 1, R >$	$< q_{accept}, -, - >$
q_2	$< q_2, 0, R >$	$< q_{reject}, -, - >$	$< q_{accept}, -, - >$

   □

2. Show that the function that maps a program e to the smallest equivalent program is not Totally Computable. (4 points)

   Proof: Let us say that there exists a function $f(e)$ that takes as input program e and outputs its smallest equivalent program (let us call this program x). If this were possible, then we could have two equivalent Turing Machines (i.e., M and N), with inputs m and n respectively. It then follows that $f(n) = f(m)$; thus, we have a method for determining whether two Turing Machines are equivalent, but we have shown that this problem is undecidable (Homework 3.3 (i)). Therefore, the function that maps a program to its smallest equivalent program is not Totally Computable. □

3. Show that every infinite computably enumerable set contains a decidable subset. (2 points)

   Proof: Let S be the empty set (i.e., ϕ). By definition, we know that ϕ is a subset of every infinite computably enumerable set. Clearly this is a decidable set, since there exists a Turing Machine that decides ϕ by rejecting every input. Thus, ϕ is a decidable subset of every infinite computably enumerable set. □

4. Professor Chikovski has the following algorithm for the Halting Problem: Given a Turing Machine e and a string x, use a Non-deterministic Universal Turing Machine N_u to guess a configuration of the Turing Machine e. If this configuration is a halting configuration for x, N_u declares that e halts on x. If not, N_u declares that e does not halt on x. Has Professor Chikovski solved the Halting Problem? (Recall that every Non-deterministic Turing Machine can be simulated by a Deterministic Turing Machine.) (3 points)

   Solution: No, Professor Chikovski has not solved the Halting Problem. The premise "if the NDTM guesses a non-halting configuration, then the original Turing Machine halts" is false, because that decision cannot be reached. Every time we reach a halting configuration we answer that the Turing Machine halts, otherwise we must continue possibly indefinitely if a halting configuration doesn't exist. □

5. Prove that: *A set S is computably enumerable if and only if there is a decidable relation $R(x, y)$, such that $x \in S \Leftrightarrow \exists y R(x, y)$.* (3 points)

Proof:

(a) Suppose that S is computably enumerable. If $S = \phi$, then there exists a decidable relation, such that: $x \in S \Leftrightarrow \exists y\,[x = x \bigwedge y \neq y]$, which will always say "NO" to any two elements x and y. If $S \neq \phi$, then by definition $S = range(f)$, where f is a totally computable function. Therefore, for every element $x \in S$, there exists a number y, such that $f(y) = x$. The relation $\exists y\,[f(x) = y]$ is decidable, since the graph of every totally computable function is decidable. Thus,$x \in S \Leftrightarrow \exists y\,[x = x \bigwedge y \neq y]$ $x \in S \Leftrightarrow \exists y R(x, y)$.

(b) Suppose that there exists a decidable relation R, such that $x \in S \Leftrightarrow \exists y R(x, y)$. We define a function: $f(x) = \begin{cases} a, & if\ R(\tau_1(x), \tau_2(x))\ is\ false \\ \tau_1(x), & if\ R(\tau_1(x), \tau_2(x))\ is\ true \end{cases}$ where a is a fixed member of S
It is obvious that $range(f) \subseteq S$ and for every $x \in S$, there exists a y such that x and y are related (decidable relation $R(x, y)$). In this case $f(< x, y >) = x$ and $S \subseteq range(f)$. Finally $S = range(f)$ and S is computably enumerable.

□

6. Consider a computer with a 32-bit, 256K RAM memory and a finite control CPU. The memory is partitioned into a Program area and a Data area as shown in Figure (1). Assume that a word w is written in the Data Area and a program p is written in the Data Area; also assume that the finite control can simulate p on w. The finite control has been programmed to track the contents of the entire memory in one step. What can you say about the following problem: Does p halt on w? (You are allowed to reprogram the finite control to suit your needs.) (4 points)

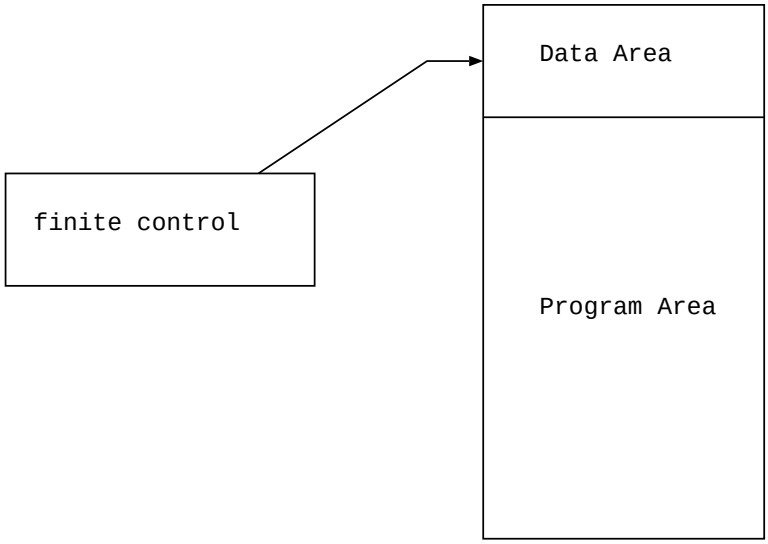

Figure 1: 32-bit computer

Solution: Observe that the memory in the computer is finite. Therefore, there are only a finite number of possible "configurations" (where a configuration is the contents of the memory and the state of the finite control). It follows that the halting problem is decidable. During the simulation of p on w we keep track of the "configurations" and in the case that a "configuration" is repeated we conclude that p does not halt on w. □