CPSC 421/501 Intro to Theory of Computing (Term 1, 2013-14)
Assignment 4

Due: Friday Oct 25th, in class.

Question 1: [10 marks]

Imagine you are working for a company that has implemented a program. Let's call that program P_1. You are hired as a summer intern to design a new program that is equivalent but runs even faster. Let's call your new program P_2.

Your boss says that before you get paid, you must design a program to demonstrate P_1 and P_2 are equivalent. More formally, you are to design a new program Q which takes two arbitrary programs (i.e., Turing machines) M_1 and M_2 as input. Q must decide if, for all inputs x, the output of M_1 on input x equals the output of M_2 on input x.

Explain why this is going to end up as an unpaid internship.

Question 2: [10 marks]

Let A and B be two disjoint languages over the alphabet Σ. Say that language C **separates** A and B if $A \subseteq C$ and $B \subseteq \overline{C}$. Show that any two disjoint co-recognizable languages are separable by some decidable language. (A language A is said to be **co-recognizable** if its complement, namely $\overline{A}$, is recognizable.)

Question 3: [10 marks]

In class we claimed that P is a nice complexity class because polynomial-time computations are closed under composition. Let's check whether polynomial-time computations are closed under a polynomial number of compositions.

Let M be a program with two inputs: $i \in \mathbb{N}$ and $w \in \Sigma^*$. Let $c > 0$ be a fixed constant (which depends on M but not on i or w) and let $n = |w|$. Suppose that

- On any inputs, M makes only $O(n^c)$ basic computational steps (each of which takes constant time).
- $M(i, w)$ can make $O(n^c)$ calls to $M(i + 1, w)$ when $i < n$.
- $M(n, w)$ only does basic computational steps and does not calls M again as a subroutine.

Does $M(0, w)$ run in time polynomial in n?

Question 4: [10 marks]

(a): In class we claimed that, if a polynomial-time algorithm is discovered for some problem, it is usually possible to discover a reasonably efficient algorithm, say one running in $O(n^5)$ time. We could try to formalize this by saying that $P \subseteq TIME(n^5)$. Is this a true statement?

(a): Another standard complexity class is $E = \bigcup_{c>0} TIME(2^{cn})$. Is it true that $E = EXP$?

OPTIONAL BONUS QUESTION: [20 marks]

Question 5: In any programming language of your choice, write an implementation of the recursion theorem.

For example, suppose we use the programming language Python. Your program should take as input a function $T(s, w)$. Your program should produce as output a new Python function $R(w)$ such that

$$R(w) \;=\; T(\langle R \rangle, w).$$

where $\langle R \rangle$ denotes the source code of the function R.

You should **not** use any native abilities of the language to provide access to the source code. You also should **not** directly use the filesystem to access the source code on disk.

Please send your solution to Prof. Harvey by email, along with a brief explanation of how it works.