CS 181d, Complexity Theory
Fall 2012
Takehome Final Exam

This exam is subject to the following guidelines:

- You may use your class notes and the solution sets. No other materials or resources are permitted.
- You may spend as much time as you wish on the exam, but it is **due at the CS office by 5 PM on Friday, October 19.** If nobody is present, please slide your exam under the office door.
- You may cite results from class, homework problems, or solution sets without re-proving them. If you cite a result, be explicit about what result you're citing and where we showed that result (lecture, homework, solution set).
- There are four problems on this exam. **Choose any three!** They are all worth the same number of points. Although you need only submit solutions to three problems, if you submit solutions to four problems, the problem with the lowest score will be dropped.

1. **Prof. Lai's Million Dollar Proof!**

   Professor I. Lai of the Pasadena Institute of Technology believes that he has proved that P $\neq$ NP. (A correct proof of this assertion has a one million dollar prize associated with it, courtesy of the Clay Institute.)

   Recall that DTIME(n^k) is the set of languages decidable in time $O(n^k)$ on a deterministic one-tape Turing Machine. Also, you may assume that it is true that the function n^k is time-constructible for every positive integer k and that the set of time-constructible functions is closed under composition, multiplication, addition. Here's his proof:

   "By way of contradiction, assume that P = NP. Then 3SAT $\in$ P and so 3SAT is in in DTIME(n^c) time for some positive integer c. Since every problem in NP is polynomial-time reducible to 3SAT, every problem in NP is solvable in time $O(n^d)$ for some positive integer d. That is, P $\subseteq$ DTIME(n^d). Note that d may be larger than c to account for the time required to perform the reduction from a given problem in NP to 3SAT. However, from the Time Hierarchy Theorem, we know that there exists some language solvable in time $O(n^{3d})$ that is not solvable in $O(n^d)$. We used n^{3d} here because it is strictly larger than $(n^d)^2$ as

required by our proof of the Time Hierarchy Theorem. Thus, there exists a problem L that is in DTIME(n^{3d}), and thus solvable in polynomial time, such that $L \notin$ DTIME(n^d), contradicting P $\subseteq$ DTIME(n^d)."

Explain why Prof. Lai won't be collecting his million dollar prize and retiring to Bora Bora as he'd planned.

2. **Non-Recursively Enumerable Languages!**

   In this problem we will consider two languages that aren't even recursively enumerable!

   (a) Consider the language

   $$L_1 = \{< M > | L(M) \text{ is recursive}\}$$

   Prove that L_1 is not recursively enumerable. (*Note:* Just because $L(M)$ is recursive does not mean that M itself halts on all of its inputs! M may not be a "nice" machine that halts on all input, but some other nice machine might exist for L that does halt on all inputs.)

   (b) Now consider the following related language

   $$L_2 = \{< M > | L(M) \text{ is NOT recursive}\}$$

   Prove that L_2 is not recursively enumerable.

3. **Alternating Turing Machines!**

   Throughout this problem, assume that all TMs have just one tape.

   Recall that nondeterministic TM (NDTM) accepts if there exists a "computation path" that ends in the accepting state. Another way to look at this is that when all paths of a NDTM's "computation tree" are done, each leaf passes the message "accept" (if that leaf was an accepting state) or "reject" (if that leaf was a rejecting state) to its parent. The parent then determines if **there exists** a child that passed "accept". If so, the parent passes "accept" to its parent and otherwise it passes "reject". This process continues all the way up to the root. The NDTM accepts if the root sees that there exists a child that passed it "accept".

   We can generalize NDTM's by allowing each node in the computation tree to be either an "existential" (or "OR") node or a "universal" (or "AND") node. More precisely, an *alternating Turing Machine (ATM)* is a nondeterministic

TM in which each state (other than the unique accepting and rejecting states) is labeled as either "existential" or "universal". When the ATM is run on an input string w, it creates a computation tree. A universal node passes "accept" to its parent if and only if all of its children passed it "accept". An existential node passes "accept" to its parent if and only if at least one of its children passed it "accept". The root is either universal or existential. If the root is universal and receives "accept" from all of its children, the ATM accepts, otherwise it rejects. If the root is existential and receives "accept" from at least one of its children, the ATM accepts otherwise it rejects.

Let ATIME($t(n)$) denote the set of all languages decided in time $O(t(n))$ by some ATM. Let ASPACE($t(n)$) denote the set of all languages decided in space $O(t(n))$ by some ATM. We define AP, APSPACE, and AL to be the classes of languages decided by ATMs in polynomial time, polynomial space, and log space, respectively.

In this problem, we will explore some of the properties of these complexity classes. Recall that DTIME($f(n)$) refers to the class of problems solvable by a deterministic TM in time $O(f(n))$ and similarly DSPACE($f(n)$) refers to the class of problems solvable by a deterministic TM in space $O(f(n))$.

(a) Two boolean formulas s and t (made up of variables and boolean operators NOT, AND, OR, and implication) are said to be equivalent if they have the same set of variables and then evaluate to the same value for every possible valuation of the variables. A boolean formula is said to be *minimal* if there exists no shorter formula to which it is equivalent. (Where the length of a formula is the total number of variables and operators.) Consider the problem

$$\text{MIN-FORMULA} = \{< s > \mid s \text{ is a minimal boolean formula}\}$$

Show that MIN-FORMULA is in AP.

(b) Show that for $f(n) \geq n$,

$$\text{ATIME}(f(n)) \subseteq \text{DSPACE}(f(n)) \subseteq \text{ATIME}(f^2(n)).$$

(c) Show that for $f(n) \geq \log n$,

$$\text{ASPACE}(f(n)) \subseteq \text{DTIME}(2^{O(f(n))}).$$

(d) **Optional BONUS Problem:** Show that for $f(n) \geq \log n$,

$$\text{ASPACE}(f(n)) = \text{DTIME}(2^{O(f(n))})$$

(e) Using all of the results above (including the result of the optional bonus part above), briefly explain why AL = P, AP = PSPACE, and APSPACE = EXPTIME. (EXPTIME is the class of languages accepted by deterministic TMs in exponential time.) Wow!

4. **[20 Points] Multi-Tape Time Constructibility.**

In class and on the homework we mentioned that the set of time constructible functions is closed under a variety of operations. Here we will see how such closure properties can be proved. However, we will change the model slightly just to avoid having to deal with a lot of piddly issues.

In particular, in this problem, we consider time constructibility in multi-tape Turing Machine means. Specifically, throughout this a problem a TM may have some constant number of tapes, each with its own tape head. The tapes are assumed to be infinite in both directions. The input string appears on the first tape with the tape head under the leftmost (non-blank) symbol of the string. In one time step, the machine consults all of its tape heads, writes new symbols under each tape head, and moves each tape head independently to the left, right, *or keeps it in the same location.*

We say that a function f is time constructible in this model if there exists a TM (as defined above) such that on any input of length n the machine runs for exactly $f(n)$ steps and then halts. Note that the running time depends only on the length of the input.

Prove that if f and g are time constructible then so is $f \circ g$ where $f \circ g$ is the composition of f and g. That is, $(f \circ g)(n) = f(g(n))$. Note that $f(g(n)) \neq f(g(n)) + g(n)$.