CS 142 & Math 167
Fall 2008
Homework 6
Due Wednesday, October 15

1. **[40 Points] PSPACE = NPSPACE!**
Before embarking on this problem, make sure that you understand the proof we showed in class that QBF is PSPACE-complete.

Recall that we defined PSPACE to be the set of all languages L such that there exists some polynomial $p(n)$ and some **deterministic** TM M such that on input w, M uses at most the first $p(|w|)$ tape cells to determine whether w is in L.

Similarly, we can define NPSPACE to be the set of all languages L such that there exists some polynomial $p(n)$ and some **nondeterministic** TM M such that on input w, if $w \in L$ then there exists some accepting computation path, if $w \notin L$ then there exists no accepting computation path, and *every* computation path uses at most the first $p(|w|)$ tape cells. While it is not known whether P = NP, it is known that PSPACE = NPSPACE. In this problem we'll prove this! Note that NPSPACE is defined in terms of nondeterminism and *not* defined in terms of verifying certificates.

   (a) Professor I. Lai proposes the following approach: "Look, if L is accepted by a nondeterministic TM M in polynomial space, I will build a deterministic TM M' that simulates M, and thus accepts the same language, and still only uses polynomial space! How? Easy! Nondeterministic TM M has at most b nondeterministic choices at each step. M' will enumerate sequences in base b on a work tape and then simulate M on input w using the current sequence to dictate the choices that M would make. If, for a given base b sequence the simulation reveals that M accepts w, then M' will accept w. Otherwise, we will write down the next base b sequence and try again. If none of these sequences work, well then, M can't accept w so M' also rejects w. In order to save space, M' will overwrite the new base b sequence over the old base b sequence. It's so easy that I want to eat ten cans of SPAM to celebrate!" Explain why Professor Lai is celebrating a bit too soon. That is, explain what's wrong with this so-called proof.

   (b) Now we will fix things to make the proof work correctly! The idea is to use techniques similar to those employed in Stockmeyer's Theorem to show that QBF is PSPACE-complete! Notice that since nondeterministic

TM M never uses more than polynomial space, $p(n)$, it cannot enter more than $c^{p(n)}$ different configurations for some constant c. Moreover, any one of these configurations can be recorded in polynomial space.

Using this observation, argue **very carefully** that a deterministic polynomial space TM M' can be constructed to simulate TM M. Make sure to show that no more than polynomial space is really being used in your simulation. Pretty nifty!!

2. **[25 Points] Generalized Lunar Lockout is in PSPACE** In class, we described the Generalized Lunar Lockout game and showed that it is PSPACE-hard. Your job is to prove that the problem is in PSPACE, thus establishing that the problem is PSPACE-complete. Recall that the game is played on a $m \times m$ board. The initial configuration is some set of objects placed on the board. An object may be a stationary robot (a "rock"), a movable robot, or the special movable target robot. One of the cells is designated as the target cell. The question is whether there exists a sequence of moves that bring the special target robot to the target cell. Notice that the number of moves in such a "winning" sequence could potentially be very long!

3. **[35 Points] Regular expressions and PSPACE.**

   Many problems related to regular expressions are known to be in PSPACE but are not known to be in any "lower" complexity class. For example, consider the problem of determining whether a given regular expression over the alphabet $\Sigma = \{0, 1\}$ is equivalent to Σ^*. For example, $0^*1^* + (0 + 1)1^*$ is not the same same as Σ^* but $1^*((\epsilon + 1)(\epsilon + 0))^*$ is equivalent to Σ^*.

   So, we are considering the following problem: Given an encoding of a regular expression, is that regular expression equivalent to Σ^*? We'll call this problem REGEQ (REGular expression EQuivalence). We'll show that REGEQ is in PSPACE in two steps.

   (a) Prove that REGEQ is in PSPACE. Be careful to show that your solution works correctly and really uses only polynomial space! (*Note:* Take a close look at how we prove the equivalence of regular expressions, NFA's, and DFA's. You will also want to use the result that you proved in part (a), but you will need to use it with caution.)

   (b) Next, consider the more general problem REGPAIR which takes as input the encodings of two regular expressions, R_1 and R_2, and determines

whether or not these two regular expressions represent the same language. Prove that REGPAIR is in PSPACE.