CS764 — Computational Complexity — Spring 2014

Problem Set 2

Basic problems (required): submit solutions by Monday, June 23.
Advanced problems (optional): submit solutions by end of lecture period.

Basic problems

These problem can be solved using the results and techniques discussed in class, and some bits of basic mathematics.

1. Let $E_0(n) = n$ and $E_{m+1}(n) = 2^{E_m(n)}$ for $m \geq 0$. A language is called *elementary* if it can be accepted in time $E_m(n)$ for some m. Show that the class of elementary languages has no complete languages with respect to log-space (or polynomial-time) reductions. Hint: assume a complete language, and derive a contradiction.

2. Recall that a context-free grammar (CFG) has the form (V, T, Q, S), where T and V are disjoint finite sets of symbols (terminals and variables, respectively), Q is a set of productions, and $S \in V$ is the goal (or start) symbol. Each production in Q has the form $A \mapsto \alpha$, where $A \in V$ and $\alpha \in (T \cup V)^*$. Then $w \in T$ is in the language of C if and only if one can produce w from S by repeatedly using the rules in Q.

   For example, if $V = \{S\}$, $T = \{0, 1\}$ and $Q = \{S \mapsto \varepsilon, S \mapsto 0S0, S \mapsto 1S1\}$, then (V, T, Q, S) is a CFG for the even-length palindromes over $\{0, 1\}^*$. (Here ε denotes the empty string.)

   Let CFE $= \{\, G = (V, T, Q, S) \mid$ no strings can be generated from $G \,\}$—the set of CFGs that generate the empty language.

   Show that the language CFE is P-complete. You can either use a generic reduction or use a reduction from the P-complete problem CVAL.

3. Consider the following game played between two players, Path and Block. The initial position is a directed graph G with two marked vertices s and t. At each turn, Path marks a vertex of the graph and then Block removes an unmarked vertex from the graph. The game ends when each vertex has been either marked by Path or removed by Block. Path wins the game if the graph remaining at the end of the game contains a directed path from s to t.

   Show that the set of initial graphs for which Path has a winning strategy is *PSPACE*-complete.

4. Let RL' be the class of languages accepted in worst-case space $O(\log n)$ by some coin-flipping Turing machine that

   - on any input, halts with probability one,
   - accepts members with probability at least 1/2, and
   - rejects nonmembers with probability one.

   Show that $RL' = NL$.

   (Hint: An RL'-machine can run for a long time on some sequences of random choices.)

Advanced problems

These problems require techniques not discussed in class, and/or some inventiveness—but solutions lie within reach.

5. Recall the language FVAL from the first problem set.

   (a) Define a model of Turing machines with random-access input. Rather than an input tape, it should have an index tape; writing a value i (in binary) on the index tape designates reading symbol i of the input string. Define one or more special states that either accept or reject based on the value of symbol i. (Charge a time of $\log i$ for such an operation—since it depends on the entire index tape.)

   (b) Show that FVAL $\in$ $ATIME(\log n)$. Use your model of random-access inputs; thus the machine can read each symbol of the input on some branch of its computation.

6. Consider the language of regular expressions augmented to allow the negation symbol $\neg$, where $\neg r$ denotes the complement of set r. Since the regular languages are closed under complement, this does not allow the representation of non-regular languages, but it does allow more succinct descriptions (for example, $\neg((01)^*) \equiv (01)^*(1 + 00)(0 + 1)^*$). Using standard constructions of finite automata and regular expressions, an augmented regular expression can be transformed into an equivalent standard regular expression (without $\neg$) or finite automaton.

   Show that the problem of deciding whether two given regular expressions with negation are equivalent is decidable but not elementary (defined in Problem 1).