ArsDigita University
Month 8: Theory of Computation
Professor Shai Simonson

Problem Set 5

1. Short Answers: Name that Language

For each of the languages below, indicate the smallest complexity class that contains it. (i.e. Regular, Deterministic Context Free, Context Free, Turing Machines (Recursive). Assume an alphabet of {0,1} unless otherwise specified. You do not need to prove your answers.

a. $0^n1^m0^p1^q$, where n+m = p+q, and n,m,p,q > 0.
b. $0^n1^m0^m1^n$, where n,m > 0.
c. $0^n1^m0^p1^q$, where n,m,p,q > 0.
d. The set of strings over alphabet {0,1,2} with an equal number of 0's and 2's, or an equal number of 0's and 1's.
e. 0^m over the alphabet {0} where m is of the form 2k + 1, k > 0.
f. The set of strings with 3n 0's and 4m 1's, for m,n > 0.
g. The set of strings that have at least ten times as many 0's as 1's.
h. The set of strings that are either odd length or contain 5 consecutive 1's.
i. $0^m10^{m!}$, m > 0.
j. The set of strings over alphabet {0,1,2} where the number of 1's equals the number of 2's, and every 0 is followed immediately by at least one 1.

2. Optional: More Turing Machine Design

a. Design a TM program to take a binary integer n as input, and return the binary string with value n+1. You may erase the input if you want.
b. Design a TM subroutine which takes a binary string and copies it to the right of the input with a $ in front. That is, it turns x into $x.
c. Design a TM subroutine turns $x into xx.
d. Design a TM to accept strings of the form ww.

3. An Undecidability Problem.

Prove that the problem of determining if the languages generated by two CFG's are equal is undecidable. You may refer to any result discussed in class.

4. Is it R.E. or is it not?

For each of the following languages, state whether the language is or is not recursively enumerable and whether the complement of the language is or is not recursively enumerable. Give some justification for your answers.

a. The language of all TM's that accept nothing.
b. The language of all TM's that accept everything.
c. The language of all TM's that accept Regular sets.
d. The language of all PDA's that accept everything.
e. The language of all CFG's that are ambiguous

5. A Refutation of the Halting Problem?

Consider the language of all TM's that given no input eventually write a non-blank symbol on their tapes. Explain why this set is decidable. Why does this not contradict the halting problem?

6. The Post Correspondence Problem for One-Character Strings.

Prove that the PCP problem is decidable for strings over the alphabet {0}.

7. Extra Credit: Even Oracles Sometimes Need Oracles.

(Text: 6.22) Let

$$Z = \{ \langle M,w \rangle \mid M \text{ is an oracle TM and } M^{ATM} \text{ accepts } w\}.$$

Use a proof by diagonalization to show that an oracle TM with an oracle for A_{TM} can't decide Z.

8. Extra Credit: Variations on the theme of 3SAT.

a. Prove that the 3SAT variation where each variable x and –x appear in exactly two clauses is still NP-Complete.
b. Prove that the 3SAT variation where each variable x an –x appear in exactly one clause is solvable in polynomial time. (Hint: Think of the polynomial algorithm for 2SAT).

9. Satisfiability for DNF Formulas is in P.

Prove that the problem of determining whether there is a T/F assignment that makes a given *disjunctive normal formula* true can be solved in polynomial time. How do you explain this in light of the fact that any formula in conjunctive normal form can be converted to one in disjunctive normal form, and the satisfiablity of CNF formulas is NP-Complete?

10. A Punchcard Puzzle that is NP Complete.

(Text: 7.26). You are given a box and a collection of cards. Because of the pegs in the box and notches on the cards, each card will fit in the box of either two ways. Each card contains two columns of holes, some of which may not be punched out. The cards can be flipped about the vertical axis so that the columns are interchanged. The puzzle is solved by placing all the cards in the box so as to completely cover the bottom of the box, (i.e., every hole position is blocked by at least one card that has no hole there.) Let

PUZZLE = { $\langle c_1, \dots , c_k \rangle$ | each c_i represents a card and this collection of cards has a solution. }

Show PUZZLE is NP-complete. (See the text for an illustration of the cards.)

11. PSPACE Hard implies NP Hard

(Text: 8.6). Show that any PSPACE-hard language is also NP-hard.

12. A TIC-TAC-TOE-Like Game that is in PSPACE

(Text: 8.10). The Japanese game go-moku is played by two players, “X” and “O”, on a 19x19 grid. Players take turns placing markers, and the first player to achieve 5 of his markers consecutively in a row, column, or diagonal, is the winner. Consider this game generalized to an n by n board. Let

GM = {<P> | P is a position in generalized go-moku,
where player “X” has a winning strategy}

A position means a board with markers on it , such as may occur in the middle of a play of the game. Show that GM is in PSPACE.

13. A Punchcard Puzzle that is PSPACE Complete.

(Text: 8.14). Consider the following two-person variation of the language PUZZLE that is described in problem d. above. Each player starts with an ordered stack of puzzle cards. They take turns placing them in order in the box and may choose which side faces up. Player I wins if, in the final stack, all hole positions are blocked, and Player II wins if some hole position remains unblocked. Show that the problem of determining which player has a winning strategy for a given starting configuration of the cards is PSPACE-complete.

14. Extra Credit: Regular Expression Equivalence is in PSPACE

(Text: 8.16). Let EQ_{REX} = {<R,S>| R and S are equivalent regular expressions}. Show that EQ_{REX} is in PSPACE.

15. Extra Credit: Kleene Star Preserves P

(Text: 7.13). Show that P is closed under the Kleene star operation. (Hint: on input $y = y_1 \ldots y_n$ for yi in Sigma, build a table indicating for each i <=j whether the substring $y_i \ldots y_j$ is in A* for any A in P.)

16. Extra Credit: The Game of Nim and Logarithmic Space

(Text: 8.21). The game of nim is played with a collection of piles of sticks. In one move a player may remove any nonzero number of sticks from a single pile. The players alternately take turns making moves. The player who removes the very last stick loses. Say that we have a game position in NIM with k piles containing $s_1, \ldots, s_k$ sticks. Call the position balanced if, when each of the nubers si, is written in binary and the binary numbers are written as rows of a matrix aligned at the lower order bits, each column of bits contains an even number of 1’s. Prove the following two facts:

a. Starting in an unbalanced position, a single move exists that changes the position to a balanced one.

b. Starting in a balanced position, every single move changes the position into an unbalanced one.

Let

NIM = {<$s_1, \ldots, s_k$> | each s_i is a binary number and Player I has a winning strategy in the NIM game starting at this position.

Using the preceding facts about balanced positions to show that NIM is in L, the class of languages that are decidable in logarithmic space on a deterministic Turing Machine.

17. Extra Credit: Two Counters Are All You Really Need In Life

Prove that two counters are enough to simulate a Turing Machine. (Hint: prove that two counters can simulate a stack, then prove that two counters can simulate four counters).

18. Extra Credit: Infinite Recursive Sets Hiding In R.E. Sets

Show that every infinite recursively enumerable set has an infinite recursive subset. (Hint: Prove that a TM exists that *generates* the r.e. set. That is, it starts with an empty tape and keeps printing strings in the set. Consider the set of all strings x which are among the first 2^|x| strings generated by this TM. Prove that this set is infinite and recursive.)