Theory Qualifier
June 2, 1987

Instructions

Four hours, closed book. All questions are of equal weight.

There are two parts, a *basic part* and an *advanced part*, each consisting of four questions. Of the four advanced questions, there is one from each of the following areas:

1. recursive function theory
2. constructive mathematics and program logics
3. algorithm design and analysis
4. computational complexity

For a *basic pass*, you must answer all four basic questions satisfactorily. For an *advanced pass*, you must answer all four basic questions and any two of the four advanced questions satisfactorily. Indicate clearly which two advanced questions you are attempting; only two will be graded.

The questions will be graded according to the following criteria, in order of importance:

1. correctness and completeness
2. clarity and precision
3. optimality of result

Partial credit will be awarded where appropriate. Unclear presentations and irrelevancies will be penalized. *Please read all questions carefully before answering.*

GOOD LUCK!

Part I—Basic

1. Let A be a regular set. Consider the two sets

$$\{x \mid \exists n \geq 0\ \exists y \in A\ y = x^n\}$$
$$\{x \mid \exists n \geq 0\ \exists y \in A\ x = y^n\}.$$

   One is necessarily regular and one not. Which? Give proof for both.

2. Consider the following simple programming language:

   - atomic programs:
     skip do nothing
     fail terminate abnormally
   - compound programs:
     $p; q$ sequential composition
     if B **then** p **else** q **fi** conditional

   The Boolean test B in the conditional is just a propositional letter; no Boolean operators are allowed. Note that there are no looping constructs. A program terminates normally by falling out the bottom without ever executing a **fail**. Show that the equivalence problem for these programs is co-NP complete.

3. Let T be an undirected tree with n vertices and maximum vertex degree 3. Show that there exists an edge of T whose removal splits T into two subtrees, each with at most $\lceil 2n/3 \rceil$ vertices. Give a linear-time algorithm on a unit-cost RAM to find such an edge.

4. Give a recursive set $A \subseteq \Sigma^*$ such that neither A nor $\Sigma^* - A$ contains an infinite regular subset.

Part II—Advanced

1. (recursive function theory) Show that $A \in \Sigma_2^0 \cap \Pi_2^0$ if and only if A is Turing reducible to the halting problem.

2. (constructive mathematics and program logics) Show that when any of the following three rules are added to a complete intuitionistic natural deduction system, the other two are derivable:

   (a) law of the excluded middle:
   $$\frac{}{\varphi \vee \neg\varphi}$$

   (b) reductio ad absurdum:
   $$\frac{\begin{matrix}\neg\varphi \\ \vdots \\ \bot\end{matrix}}{\varphi}$$

   (c) double-not elimination:
   $$\frac{\neg\neg\varphi}{\varphi}$$

3. (algorithm design and analysis) Let G be a weighted undirected graph with edge weights drawn from a totally ordered set. Design an *on-line* algorithm to build a minimum-weight spanning forest for G. That is, assuming that the edges of G and their weights become known one at a time, show how to update the spanning forest as every new edge comes in. Give a complete analysis, including an explicit description of all data structures. ($O(mn)$ = passing; $O(m \log n)$ = superstar.)

4. (computational complexity) Prove that linear space and polynomial time are different.