UVa - cs302: Theory of Computation Spring 2008

Final Exam

3 May 2008

Your Name:

UVa Email Id:

Honor Policy. For this exam, you must **work alone**. **No notes, books, or other resources are permitted.**

You do not need to write a pledge since we assume by default that all students are honorable, but write your initials in the box to confirm that you read and understood the honor policy for this exam: $\boxed{}$.

Directions. Answer all six questions, including all sub-parts. You may use the backs of pages for your scratch work, but we will only grade answers that are written in the provided spaces, or that are found following clearly marked arrows from these spaces. The space for each answer is designed to be big enough to easily fit a full credit, correct answer. If you feel like you need more space to write your answer, then either your answer is incorrect, inelegant, or you are providing more detail than needed for full credit.

You have 3 hours to complete this exam, although it is designed to take about 1 hour (except for the bonus question 6, which could take much longer to solve).

Question	Target	Score
1	5-5-5-5	__ - __ - __ - __
2	2-2-2-2-2-2-2	__ - __ - __ - __ - __ - __ - __
3	2-2-2-2-2-2-4	__ - __ - __ - __ - __ - __ - __
4	10-10-10	__ - __ - __
5	5-5-5-5	__ - __ - __ - __
6	Bonus	
Total	100	

Problem 1: Short Answers. (20) For each question, provide a correct, clear, precise, and concise answer from the perspective of a theoretical computer scientist.

a. [5] What is a *language*?

b. [5] Describe a regular language that *cannot* be recognized by any current laptop.

c. [5] Explain why NP *cannot* stand for *non-polynomial* (even if $P \neq NP$).

d. [5] Explain the basic structure of a proof that language A is NP-Complete.

Problem 2: Zero, One, Infinity. (14)

For each question, circle **0**, **1** or ∞ to indicate wether the value of the described entity is zero, one, or infinite. A correct answer receives full credit without any explanation. A wrong answer with a good explanation may receive some partial credit.

a. [2] Assuming P = NP, the number of NP-Complete problems that are not in P.

0 **1** ∞

b. [2] Assuming P = NP, the number of NP-Hard problems that are not in P.

0 **1** ∞

c. [2] The number of strings that are in the language described by the context-free grammar G (S is the start variable):

$$S \rightarrow 0A$$
$$S \rightarrow 1S$$
$$A \rightarrow S$$

0 **1** ∞

d. [2] The number of strings that are in the language described by the context-free grammar G (S is the start variable):

$$S \rightarrow 0A$$
$$S \rightarrow 1$$
$$A \rightarrow S$$

0 **1** ∞

e. [2] The smallest possible number of strings in a language that is undecidable.

0 **1** ∞

f. [2] The number of languages that are equivalent to $HALT_TM$.

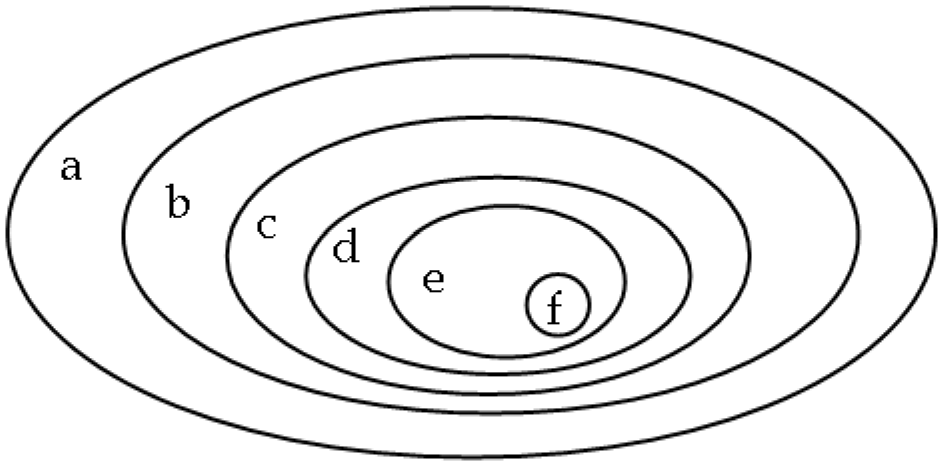
0 **1** ∞

g. [2] The number of languages that are in **TIME**(N^2) that can be decided by a multi-dimensional Turing machine in $O(N)$ steps.

0 **1** ∞

Problem 3: Drawing Classes. (16)

Each label in the diagram below corresponds to one of the following computability and complexity classes: (1) **TIME**(N^2), (2) P, (3) Decidable, (4) NP, (5) **TIME**(1), (6) Regular.



Assuming P $\neq$ NP, identify the class (1-6) associated with each label (circle the correct answer):

a. (1) **TIME**(N^2)   (2) P   (3) Decidable   (4) NP   (5) **TIME**(1)   (6) Regular

b. (1) **TIME**(N^2)   (2) P   (3) Decidable   (4) NP   (5) **TIME**(1)   (6) Regular

c. (1) **TIME**(N^2)   (2) P   (3) Decidable   (4) NP   (5) **TIME**(1)   (6) Regular

d. (1) **TIME**(N^2)   (2) P   (3) Decidable   (4) NP   (5) **TIME**(1)   (6) Regular

e. (1) **TIME**(N^2)   (2) P   (3) Decidable   (4) NP   (5) **TIME**(1)   (6) Regular

f. (1) **TIME**(N^2)   (2) P   (3) Decidable   (4) NP   (5) **TIME**(1)   (6) Regular

g. Which of the given classes include the language $\{0^i 1^j | j > i\}$? (circle *all* classes that include this language)

(1) **TIME**(N^2)   (2) P   (3) Decidable   (4) NP   (5) **TIME**(1)   (6) Regular

Problem 4: Hardness Proofs. (30)

a. [10] Prove the language $\{0^i10^i | i \geq 0\}$ is not regular.

b. [10] Is the language $MORE_{TM}$ defined below undecidable? (Answer clearly, and provide a convincing proof supporting your answer.)

> $MORE_{TM} = \{\langle A, B \rangle\}$ where A and B are descriptions of Turing machines, and the size of the language accepted by A is larger than the size of the language accepted by B.

c. [10] Is the language *NO-SUBSET* defined below in the class NP-Hard? (Answer clearly, and provide a convincing proof supporting your answer.)

NO-SUBSET = $\{\langle \{x_1, x_2, \ldots, x_n\}, m\rangle\}$ where each x_i is a number represented in binary, and there is no subset of the x_i's which sums to m, a number represented in binary.

Problem 5: Closure. (20)

a. [5] Are the decidable languages closed under concatenation? That is, if A and B are decidable languages, is the language $C = \{ab | a \in A \text{ and } b \in B\}$ a decidable language? (Provide a convincing proof to support your answer.)

b. [5] Describe what one would need to do to prove a language A is **not** in the class NP-Complete.

In *The Limits of Quantum Computers* (your Spring Break reading), Scott Aaronson writes: "If we really could build a magic computer capable of solving an NP-complete problem in a snap, the world would be a very different place: we could ask our magic computer to look for whatever patterns might exist in stock-market data or in recordings of the weather or brain activity. Unlike with todays computers, finding these patterns would be completely routine and require no detailed understanding of the subject of the problem. The magic computer could also automate mathematical creativity. Given any holy grail of mathematics—such as Goldbachs conjecture or the Riemann hypothesis, both of which have resisted resolution for well over a century—we could simply ask our computer to search through all possible proofs and disproofs containing up to, say, a billion symbols."

c. [5] Explain why the proof-finding problem is in NP.

d. [5] Do you agree with the claim that a computer that could solve NP-complete problems in polynomial time could also automate mathematical creativity? (Write a brief paragraph arguing for or against Aaronson's claim.)

Problem 6: Busy Bunny. (Bonus)

Is the *BUSY-BUNNY* language defined below NP-Complete? (Prove or disprove.)

> *BUSY-BUNNY* = $\{\langle n, s, k\rangle\}$ where k is the maximum number of 1 symbols that can be on the final tape of a Turing machine with n states and 2 symbols that halts within s steps, starting from a blank input tape.

End of Exam
Enjoy your summer!