

Final Exam*December 15, 2006*YOUR NAME:

Instructions:

This exam is *closed-book, open-notes*. Please turn off electronic devices: cell phones, laptops, PDAs, etc.

You have a total of **180 minutes**. There are 7 questions worth a total of **150 points**. The questions vary in difficulty, so if you get stuck on any question, it might help to leave it for a while and try another one.

Answer each question in the space provided below the question. If you need more space, you can use the reverse side of that page. You may assume without proof any result that was proved in class or on a homework, but state your assumptions clearly. Descriptions of Turing machines can be in the form of Sipser's "high-level descriptions". Show your work in all proofs.

<i>Do not turn this page until the instructor tells you to do so!</i>

Problem 1		Problem 5	
Problem 2		Problem 6	
Problem 3		Problem 7	
Problem 4		Total	

Problem 1: [True or False, with justification] (30 points)

For each of the following 5 questions, state TRUE or FALSE. If TRUE, give a short proof. If FALSE, give a simple counterexample.

- (a) If the language L is context-free, then L^R is also context-free. [Recall that L^R is the language comprising reversals of all strings in L .]
- (b) If the language L^* is regular, then L is regular.
- (c) The set of all *sorted decimal fractions* is countable. [A *sorted decimal fraction* is an infinite sequence of the form $0.d_1d_2d_3\dots$, where the digits d_i are in sorted order: i.e., $d_1 \leq d_2 \leq d_3 \leq \dots$. For example, $0.001333444444\dots$ is a sorted decimal but $0.02411111\dots$ is not.]

Problem 1 (continued)

(d) If languages L_1 and L_2 are both in NP, then L_1L_2 is in NP. [Recall that L_1L_2 denotes the language of all strings w_1w_2 where $w_1 \in L_1$ and $w_2 \in L_2$.]

(e) There exists a specific Turing machine M for which the language $L = \{w \mid M \text{ accepts } w\}$ is undecidable.

Problem 2: [Short Answers] (25 points)

(a) Prove that the language

$$BOTH_{TM} = \{\langle M, w_1, w_2 \rangle \mid M \text{ is a TM that accepts both } w_1 \text{ and } w_2\}$$

is Turing-recognizable. Also show that it is undecidable, using a reduction from A_{TM} .

(b) A language A is said to be in co-NP if its complement \bar{A} is in NP. A is co-NP-complete iff it is in co-NP, and for every $L \in \text{co-NP}$, $L \leq_P A$.

Name a co-NP-complete language, and justify your answer briefly.

Problem 3: [Buggy Proofs] (20 points)

Each question below contains a false “proof” of some statement with *exactly one mistake*. For each part, identify the mistake in the reasoning and explain briefly why it is wrong. [NOTE: Answers that identify more than one mistake will be penalized!]

(a) Statement: $PSPACE \subseteq SPACE(n^2)$.

“Proof”: From the textbook, we know that the language $TQBF$ is PSPACE-complete. Thus, every problem in PSPACE is polynomial-time reducible to $TQBF$. Moreover, from the textbook we know that $TQBF$ is in $SPACE(n^2)$. Thus, every problem in PSPACE belongs to $SPACE(n^2)$. So, $PSPACE \subseteq SPACE(n^2)$.

(b) Statement: The following language L is not regular:

$L = \{uv \mid u, v \in \{0, 1\}^* \text{ and the number of 0s in } u \text{ equals the number of 1s in } v\}$.

“Proof”: The proof is by contradiction. Assume that L is regular. Let p be the pumping length for L as given by the pumping lemma. Consider the string $w = 0^p 1^p$. Clearly, this belongs to L , since we can take $u = 0^p$ and $v = 1^p$. By the Pumping Lemma, we can write $w = xyz$ where $|xy| \leq p$, $|y| \geq 1$, and $xy^i z \in L \forall i \geq 0$. Thus, y must lie entirely in the 0^p part of w . Suppose $y = 0^m$ for $m \geq 1$. Then, $xy^i z = 0^{p+(i-1)m} 1^p$. This string is not in L unless $i = 1$. Thus, we have a contradiction.

Problem 4: (25 points)

Consider the following language:

$$A = \{\langle M \rangle \mid M \text{ is a TM that accepts the input string } 0110\}$$

(a) Prove that A is undecidable.

(b) Prove that the complement of A is not Turing-recognizable.

Problem 5: (25 points)

Recall that a directed graph G is *strongly connected* if every two nodes u and v of G are connected by a (directed) path in each direction (i.e., from u to v and back).

Consider the language

$$STRONGLY-CONNECTED = \{\langle G \rangle \mid G \text{ is a strongly connected directed graph}\}$$

Prove that *STRONGLY-CONNECTED* is NL-complete. Remember to not only clearly state your reduction, but also prove that it is correct and log-space.

[Hints: Use a reduction from $PATH = \{\langle G, s, t \rangle \mid G \text{ is an directed graph with a path from } s \text{ to } t\}$. In mapping a $\langle G, s, t \rangle$ to a $\langle G' \rangle$, you only need to add edges to G .]

Solve Sipser Problems 8.17, 8.18 instead

Problem 6: (20 points)

Suppose that NASA is planning to send the unmanned spaceship Plutonic to explore the (dwarf) planet Pluto.

Due to a severe budget crunch, Plutonic is being designed with sensors and actuators of very limited capability. It “knows” and controls its environment (by “environment” we mean its position, velocity, etc.) only through a set of Boolean variables x_1, x_2, \dots, x_n representing Boolean predicates on its environment. We refer to (x_1, x_2, \dots, x_n) as a *state*. Plutonic can change its state by flipping the values of the x_i s. Flipping bits is expensive (uses up scarce power), so the fewer x_i s it flips, the better it is.

Plutonic “knows” that an error has occurred if a Boolean function $\mathcal{E}(x_1, x_2, \dots, x_n)$ evaluates to 1. If an error occurs in state s , Plutonic tries to flip some x_i s to reach a different state $\hat{s} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)$ such that $\mathcal{E}(\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n) = 0$. To conserve power, Plutonic needs to flip the fewest number of bits to move from an *error state* s to some non-error state \hat{s} (note: $\mathcal{E}(s) = 1, \mathcal{E}(\hat{s}) = 0$).

NASA engineers know that both \mathcal{E} and its negation $\bar{\mathcal{E}}$ are always satisfiable. We will call such a Boolean function \mathcal{E} an *error Boolean function*. What makes matters hard is that \mathcal{E} can be an arbitrary error Boolean function that varies over time, so they must design Plutonic to flip the minimum number of bits *for any error Boolean function* \mathcal{E} to move from *any error state* s to some non-error state.

Consider the problem of finding the minimum number of bits to flip, expressed as a language:

$$A = \{ \langle \mathcal{E}, k, s \rangle \mid \mathcal{E} \text{ is an error Boolean function requiring less than } k \text{ bit flips to move from error state } s \text{ to some non-error state} \}$$

Prove that this language is NP-hard.

[Hints: (1) Use a reduction from SAT; (2) pay careful attention to your choices of k and s in the reduction; (3) your solution can be shorter than the question!]

Solve Sipser Problem 7.30 instead

(extra page for Problem 6 if you need one; *please turn over for Problem 7*)

Problem 7: [Freebie: Final Feedback] (5 points)

Here is a list of topics we covered in this course. Out of these, circle the topic you liked the best and write an “B” next to it. Also circle the topic you liked the least, and write an “L” next to it.

Regular languages: DFAs, NFAs, regular expressions

Context-free languages: CFGs, PDAs

Turing machines and variants

Decidable languages

Undecidable languages

Logical theories

Time complexity & P

NP, co-NP, NP-completeness

Space complexity, PSPACE, PSPACE-completeness

L, NL, NL-completeness

Hierarchy theorems

Special topics: Probabilistic complexity classes & Typical-case complexity of SAT

Thank you, and happy holidays!