

## 6.045 Practice Final Exam

May 16, 2007

Elena Grigorescu

Name: \_\_\_\_\_

- Please write your name on each page.
- This exam is open book, open notes.
- There are two sheets of scratch paper at the end of this exam.
- Questions vary substantially in difficulty. Use your time accordingly.
- If you cannot produce a full proof, clearly state partial results for partial credit.
- Good luck!

Part	Problem	Points	Grade
Part I	1–10	50	
Part II	1	20	
	2	15	
	3	25	
	4	15	
	5	15	
	6	10	
<b>Total</b>		150	

# Part I

## Multiple Choice Questions. (50 points, 5 points for each question)

For each question, any number of the listed answers may be correct. **Clearly** place an “X” in the box next to each of the answers that you are selecting.

**Problem 1:** Which of the following are true statements about regular and nonregular languages? (All languages are over the alphabet  $\{0, 1\}$ )

- If  $L_1 \subseteq L_2$  and  $L_2$  is regular, then  $L_1$  must be regular.
- If  $L_1$  and  $L_2$  are nonregular, then  $L_1 \cup L_2$  must be nonregular.
- If  $L_1$  is nonregular, then the complement of  $L_1$  must also be nonregular.
- If  $L_1$  is regular,  $L_2$  is nonregular, and  $L_1 \cap L_2$  is nonregular, then  $L_1 \cup L_2$  must be nonregular.
- If  $L_1$  is regular,  $L_2$  is nonregular, and  $L_1 \cap L_2$  is regular, then  $L_1 \cup L_2$  must be nonregular.

**Problem 2:** Which of the following are guaranteed to be regular languages ?

- $L_2 = \{ww : w \in \{0, 1\}^*\}$ .
- $L_2 = \{ww : w \in L_1\}$ , where  $L_1$  is a regular language.
- $L_2 = \{w : ww \in L_1\}$ , where  $L_1$  is a regular language.
- $L_2 = \{w : \text{for some } x, |w| = |x| \text{ and } wx \in L_1\}$ , where  $L_1$  is a regular language.
- $L_2 = \{w : w \in L_1 \text{ and no proper prefix of } w \text{ is in } L_1\}$ , where  $L_1$  is a regular language.

**Problem 3:** Which of the following are known to be true?

- If a language  $L$  is recognized by an NFA, then  $L$  is also recognized by some DFA.
- If a language  $L$  is recognized by a nondeterministic Turing machine, then  $L$  is also recognized by some deterministic Turing machine.
- If a language  $L$  is decided by a nondeterministic Turing machine, then  $L$  is also decided by some deterministic Turing machine.
- If a language  $L$  is decided in polynomial time by a nondeterministic Turing machine then  $L$  is also decided in polynomial time by some deterministic Turing machine.
- If a language  $L$  is decided in log space by a nondeterministic Turing machine then  $L$  is also decided in log space by some deterministic Turing machine.

**Problem 4:** Which of the following languages are undecidable ?

- CLIQUE
- $\{\langle M \rangle : M \text{ is a Turing machine and } L(M) = \text{CLIQUE}\}$
- $\{\langle M \rangle : M \text{ is a Turing machine that recognizes a nonempty language}\}$ .
- $\{\langle M \rangle : M \text{ is a Turing machine that recognizes an NP-complete language}\}$
- $\{\langle M \rangle : M \text{ is a Turing machine that recognizes a language that is also recognized by some other Turing machine } M' \text{ with an even number of states}\}$ .

**Problem 5:** Which of the following are true statements about Turing-recognizable languages?

- If  $L_1$  and  $L_2$  are both Turing-recognizable languages, then their union and their intersection must also be Turing-recognizable.
- There exists a language  $L$  such that neither  $L$  nor  $\bar{L}$  is Turing-recognizable.
- If  $L$  is Turing-recognizable but not Turing-decidable, then any Turing machine that recognizes  $L$  must loop (fail to halt) on infinitely many inputs.
- If  $L_1$  and  $L_2$  are nontrivial languages (not equal to  $\emptyset$  or  $\{0, 1\}^*$ ),  $L_1$  is Turing-recognizable, and  $L_2$  is decidable, then it must be the case that  $L_1 \cup L_2 \leq_m L_1$ .
- If  $L_i$  is Turing-recognizable for every  $i \geq 1$ , then  $\bigcup_{i=1}^{\infty} L_i$  is Turing-recognizable.

**Problem 6:** The following construction uses the Recursion Theorem and an assumed decider Turing machine  $D$  for some language (set)  $L$  of Turing machines, to construct a new Turing machine  $R$ :

$R$ : “On input  $x$ :  
Obtain  $\langle R \rangle$ .  
Run  $D$  on input  $\langle R \rangle$ .  
If  $D$  accepts then reject; otherwise accept.”

This construction yields immediate undecidability results for several different languages  $L$  of Turing machines. Which of the following are examples of such languages?

- $L = \{\langle M \rangle : M \text{ accepts all input strings}\}$
- $L = \{\langle M \rangle : M \text{ accepts some input string}\}$
- $L = \{\langle M \rangle : M \text{ accepts the string } 00\}$
- $L = \{\langle M \rangle : M \text{ does not accept the string } 11\}$
- $L = \{\langle M \rangle : M \text{ accepts the string } 00 \text{ and does not accept the string } 11\}$

**Problem 7:** Which of the following are known to be true?

- $3SAT \leq_p 2SAT$
- $2SAT \leq_p 3SAT$
- $PCP \leq_p \text{UNDIRECTED-HAMILTONIAN-CIRCUIT}$
- $\text{VERTEX-COVER} \leq_p \text{UNDIRECTED-HAMILTONIAN-CIRCUIT}$
- $\text{TQBF} \leq_p \text{PATH} \cup \text{DIRECTED-HAMILTONIAN-CIRCUIT}$

**Problem 8:** Which of the following are true statements about space-bounded complexity?

- It is known that  $TOT_{NFA}$ , the totality problem for nondeterministic finite automata, is in PSPACE.
- Savitch's theorem implies that  $\text{NSPACE}(\sqrt{\log n}) \subseteq \text{SPACE}(\log n)$ .
- TQBF is NL-complete.
- It is known that PATH is in L.
- $\text{CLIQUE} \leq_L \text{VERTEX-COVER}$

**Problem 9:** Which of the following are true statements about the proof that  $NL = \text{coNL}$  that was presented in class (and appears in Section 8.6 of Sipser's book)?

- On every branch of the nondeterministic algorithm, every variable  $c_i$  gets set to the correct number of vertices reachable from  $s$  in  $G$  in  $i$  steps by paths of length at most  $i$ .
- On some branch of the algorithm, every  $c_i$  gets set to the correct number of vertices reachable from  $s$  in  $G$  in  $i$  steps by paths of length at most  $i$ .
- When the algorithm reaches its second phase (starting with line 12 in the code on p. 328), the value of  $c_m$  is guaranteed to be equal to the total number of vertices that are reachable from  $s$  in  $G$ , by paths of any length.
- If we omit line 14, which allows the algorithm to choose nondeterministically to skip some steps, then the algorithm still works correctly but may take longer to complete.
- This construction can be easily modified to prove that  $NP = \text{coNP}$ .

**Problem 10:** Which of the following are known to be true statements about primality testing?

$PRIMES \in BPP$

$COMPOSITES \in RP$

$PRIMES \in P$

Fermat's Little Theorem implies that every composite number  $c$  fails the Fermat test. That is, does not satisfy the equation  $a^{c-1} \equiv 1 \pmod{c}$  for some  $a \in \{1, \dots, c-1\}$ .

Every non-Carmichael composite number fails the Fermat test for at least half of the possible choices of the base  $a$ .

# SCRATCH PAPER 1

# Part II

## Problem 1: (Fallacies Abound)

- (a) **Pumping lemma (10 points)** Here is a “proof”, using the pumping lemma, that the language  $L$  of all strings of 0s and 1s of length 100 is not regular. Since the result being “proved” is false (all finite languages are regular), the proof cannot be correct. What is the flaw in the proof?

“Assume for the sake of contradiction that  $L$  is regular. By the pumping lemma, if we choose an element of  $L$ , say  $w = 0^{100}$ , there are strings  $x, y$ , and  $z$ , with  $|y| > 0$ , so that every string of the form  $xy^kz$  (where  $k \geq 0$ ) is in  $L$ . Since there are infinitely many different strings of this form, this contradicts the fact that  $L$  is finite. Therefore,  $L$  is not regular.”

(b) **NP-Completeness of  $\neq$ -SAT (10 points)**

Let  $\phi$  be a cnf-formula. A  $\neq$ -assignment to the variables of  $\phi$  is one where each clause contains two literals with unequal truth values. In other words a  $\neq$ -assignment satisfies  $\phi$  without assigning all the literals in any clause to true. In other words,  $\neq$ -SAT =

$$\{\phi \mid \phi \text{ is a cnf-formula and there is a } \neq\text{-assignment that satisfies } \phi\}.$$

For instance,  $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$  has a satisfying  $\neq$ -assignment given by  $x_1 = \text{true}$ ,  $x_2 = \text{false}$ , and  $x_3 = \text{false}$ .

Sally, a 6.045 student, knows that  $\neq$ -SAT is *NP*-complete. To convince Harry, a 6.046-expert, that this is indeed true, she gives the following reduction from CNF-SAT to  $\neq$ -SAT.

“Given a formula  $\phi$  with  $k$  clauses  $C_1, C_2, \dots, C_k$ , compute the formula  $f(\phi)$  as follows: Introduce a new variable  $x$  and let the clause  $C'_i = C_i \wedge x$ . The formula  $f(\phi)$  is an AND of the clauses  $C'_1, C'_2, \dots, C'_k$ . That is, the reduction adds the *same* dummy variable  $x$  to every clause.”

Sally claims that this is a valid reduction from CNF-SAT to  $\neq$ -SAT, and it serves to prove that  $\neq$ -SAT is *NP*-hard. Harry thinks Sally is lying, since he observes that the formula  $f(\phi)$  is trivially satisfied by the assignment  $x = \text{true}$ . Whose side would you take in this argument, and why?



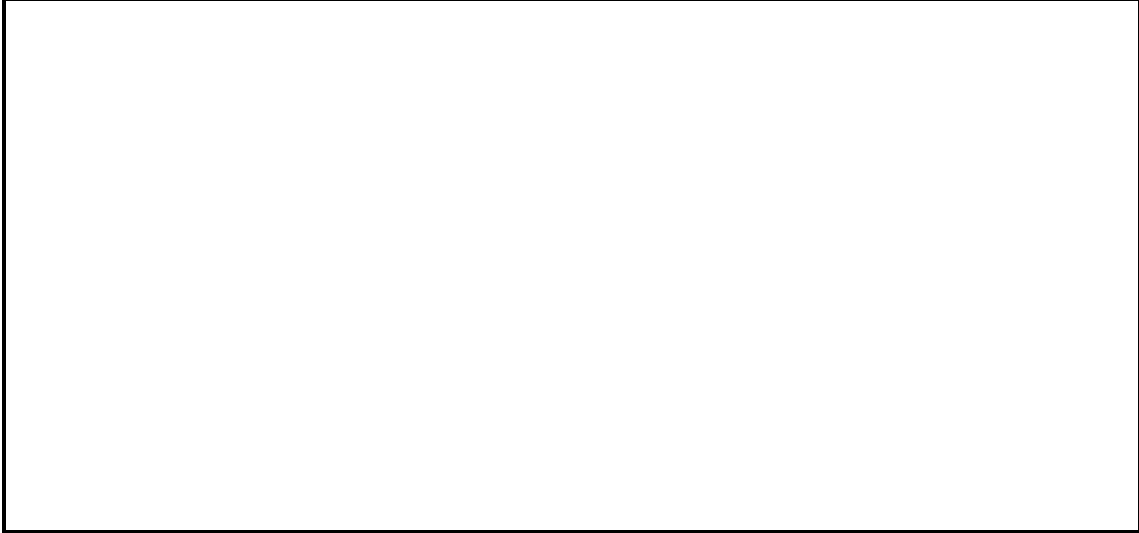
**Problem 2: 42-Minimal machines (15 points)** Let's define an (ordinary, one-tape) Turing machine  $M$  to be *42-minimal* if there is no (ordinary, one-tape) Turing machine  $M'$  that recognizes the same language as  $M$  and has a representation smaller than  $\frac{1}{42}$  times the size of the representation of  $M$ .

- (a) Is it decidable whether or not a given representation  $\langle M \rangle$  of a Turing machine is 42-minimal ?
- (b) If your answer to (a) above is yes, then describe an algorithm to decide 42-minimality; if it is no, prove that no such algorithm exists.

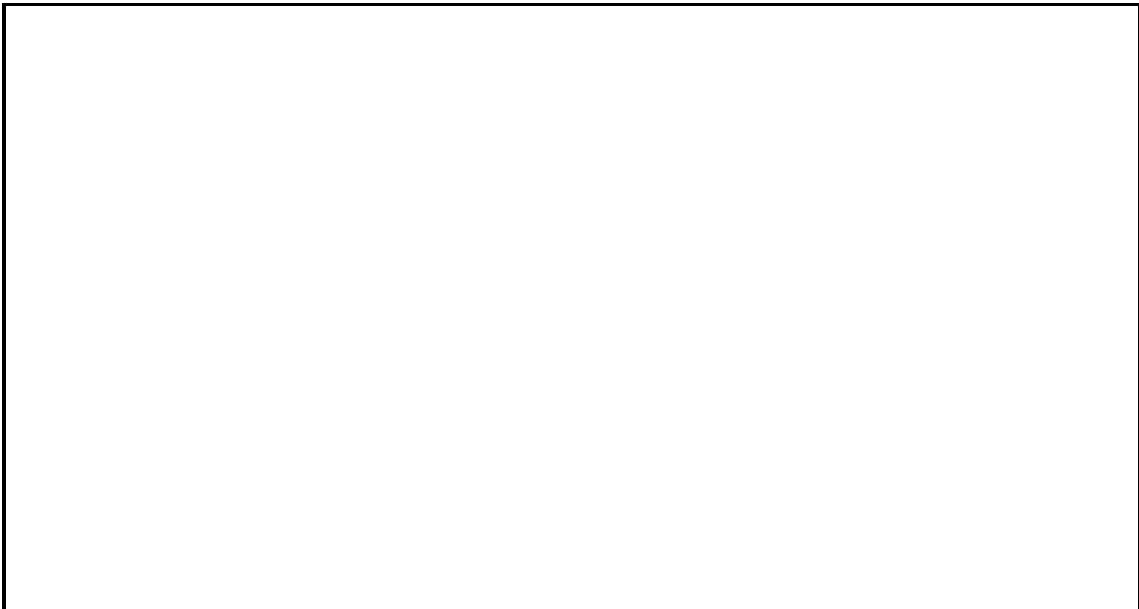
**Problem 3: Vacation packing (25 points)** Harry, an MIT senior, is packing his backpack for his summer trip to the Amazon Jungle. He discovers that he can carry only 42 pounds. He would like to take many items, including bug spray, hairspray, his iPod, underwear, a dictionary of languages spoken by native tribes, his teddy bear, etc. They total much more than 42 pounds. Harry is having trouble figuring out which ones to take.

Since he has taken 6.046 but not 6.045, Harry decides to develop a general algorithm to solve the problem. He formulates the problem, in terms of a set  $I$  of items, each with a “weight”  $w(i)$  and a “value”  $v(i)$ . His problem is, given a set  $I$  of items with weights and values, and a maximum weight  $M$ , to determine the maximum total value that can be achieved with items whose total weight is at most  $M$ .

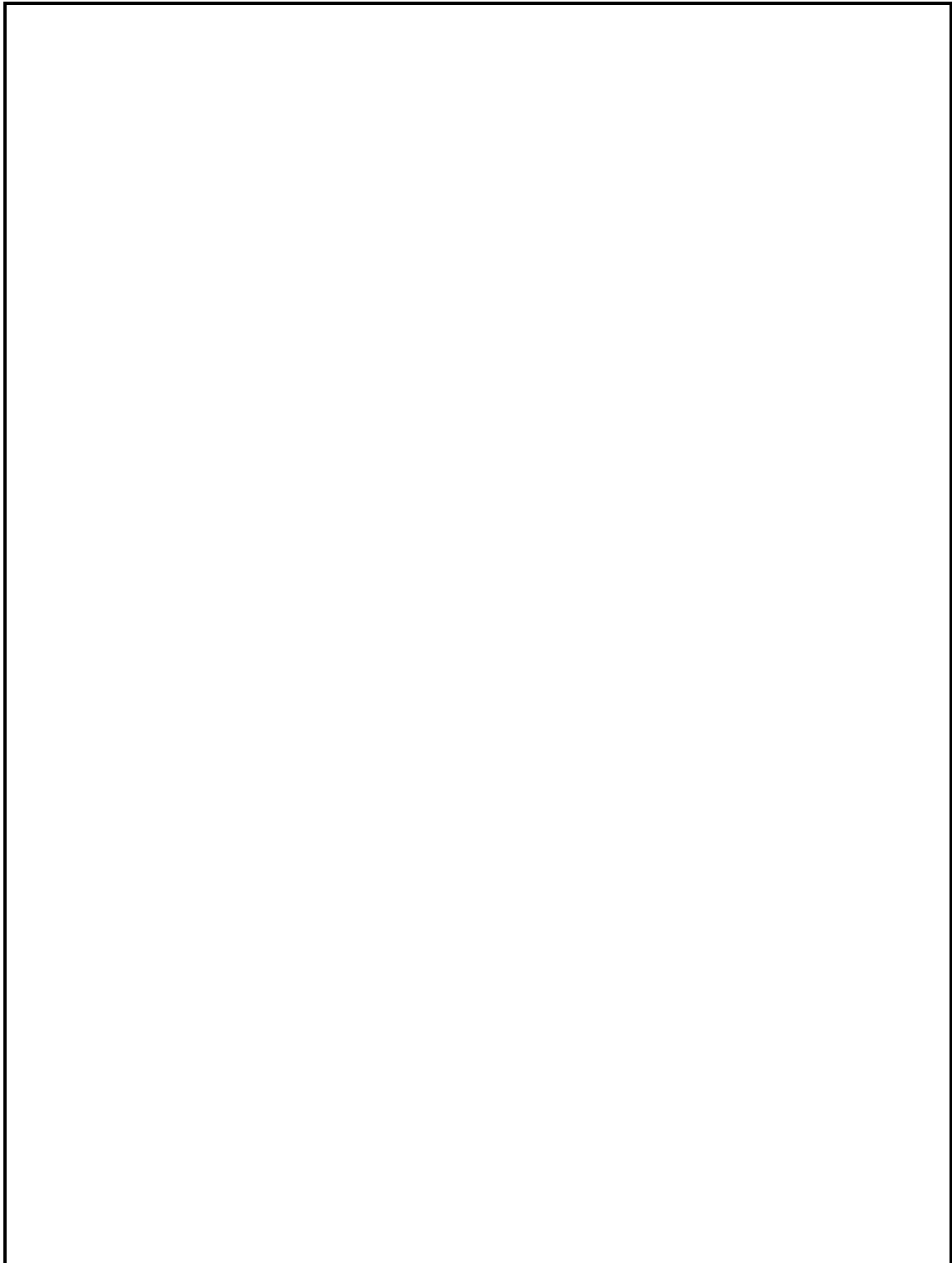
- (a) Define Harry’s *optimization problem* formally.



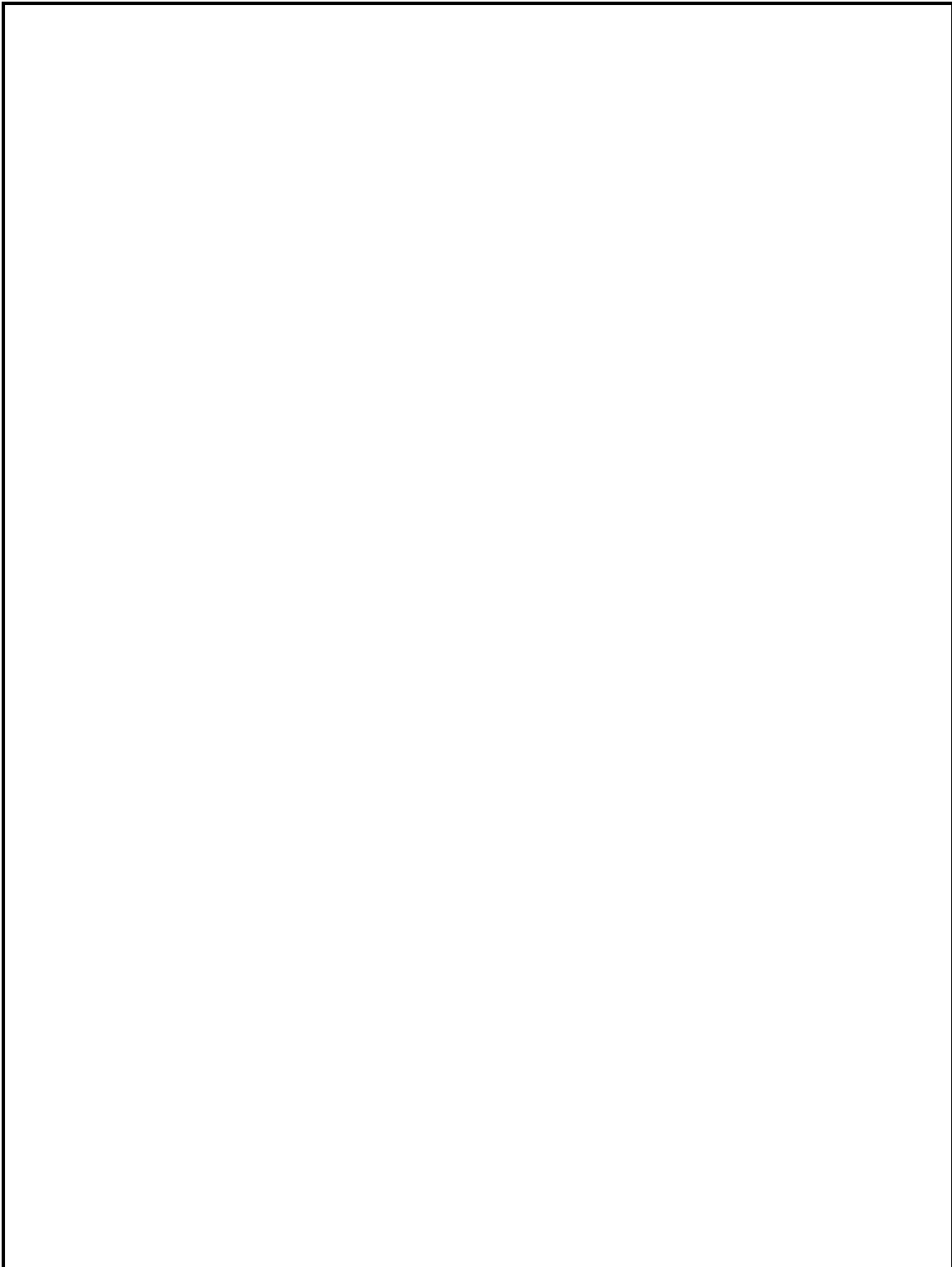
- (b) Define formally a *language problem* BACKPACK, which expresses the question of whether a given value can be achieved within a given weight.



- (c) Prove that the resulting problem is NP-complete (thereby demolishing Harry's hopes of obtaining an easy, general solution).



- (d) Explain how, if Harry could query an “oracle” for the BACKPACK language decision problem for free, he could solve his optimization problem in polynomial time.



**Problem 4: Generalized Checkers (15 points)** Define **Generalized-Checkers** to be the set of winning positions for the first player in a checker game played on a square checkerboard of any size  $n \times n$ , where  $n$  is an even number.

More precisely, we define a *position* of an  $n \times n$  checkerboard to be any placement of at most  $(\frac{n}{2} - 1)\frac{n}{2}$  red checkers and at most  $(\frac{n}{2} - 1)\frac{n}{2}$  black checkers on the red squares of the checkerboard, with no two checkers occupying the same square. Also, any number of the checkers may be designated as “kings”.

We assume that the first player owns the black checkers and that his/her home side of the board is the top of the board. The rules are the usual ones for checkers.<sup>1</sup> We define **Generalized-Checkers** as the language

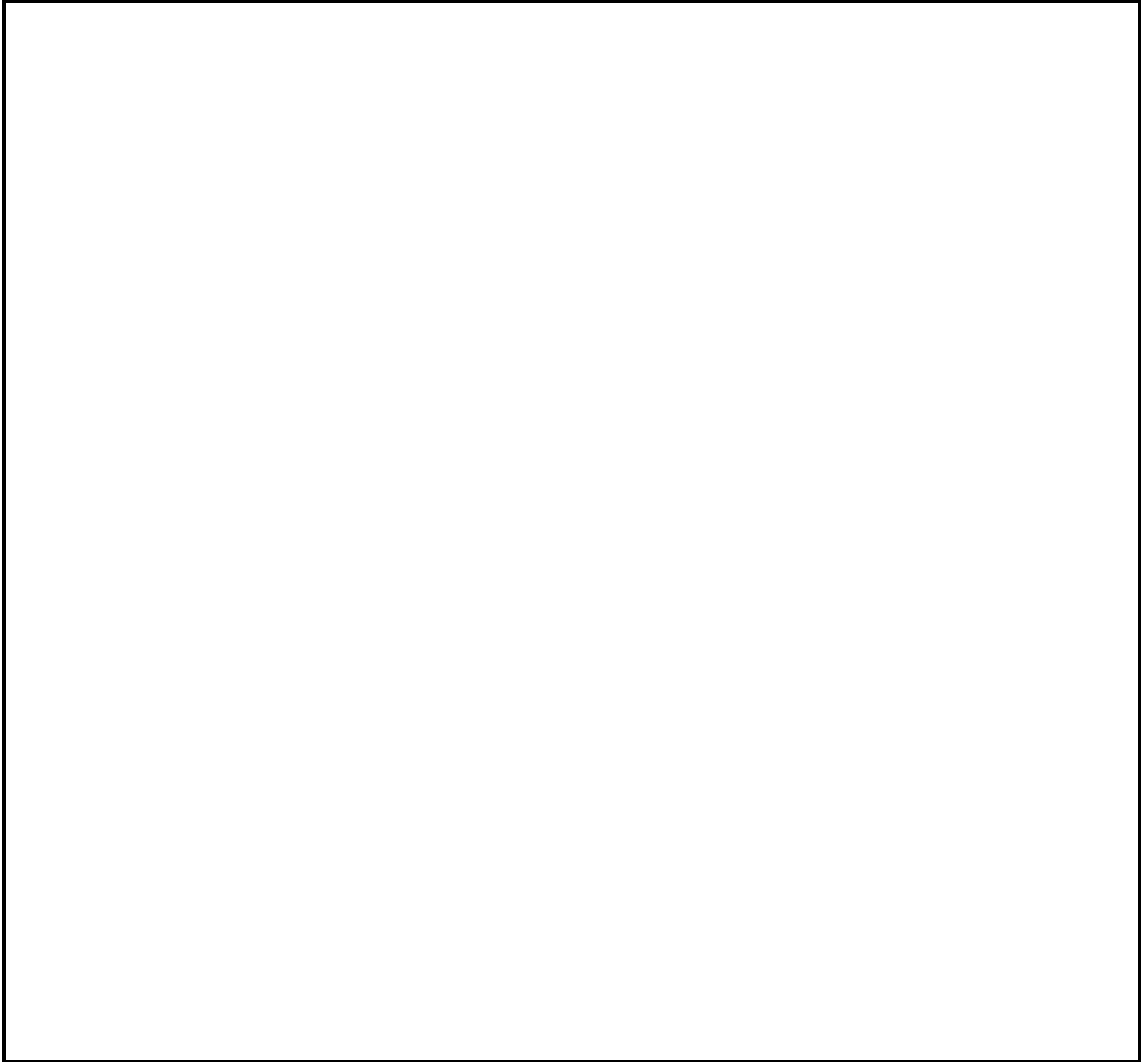
$\{(n, P) : P \text{ is a position of an } n \times n \text{ checkerboard in which the first player has a winning strategy}\}$ .

(a) Describe a polynomial-space algorithm to decide the language **Generalized-Checkers**.

---

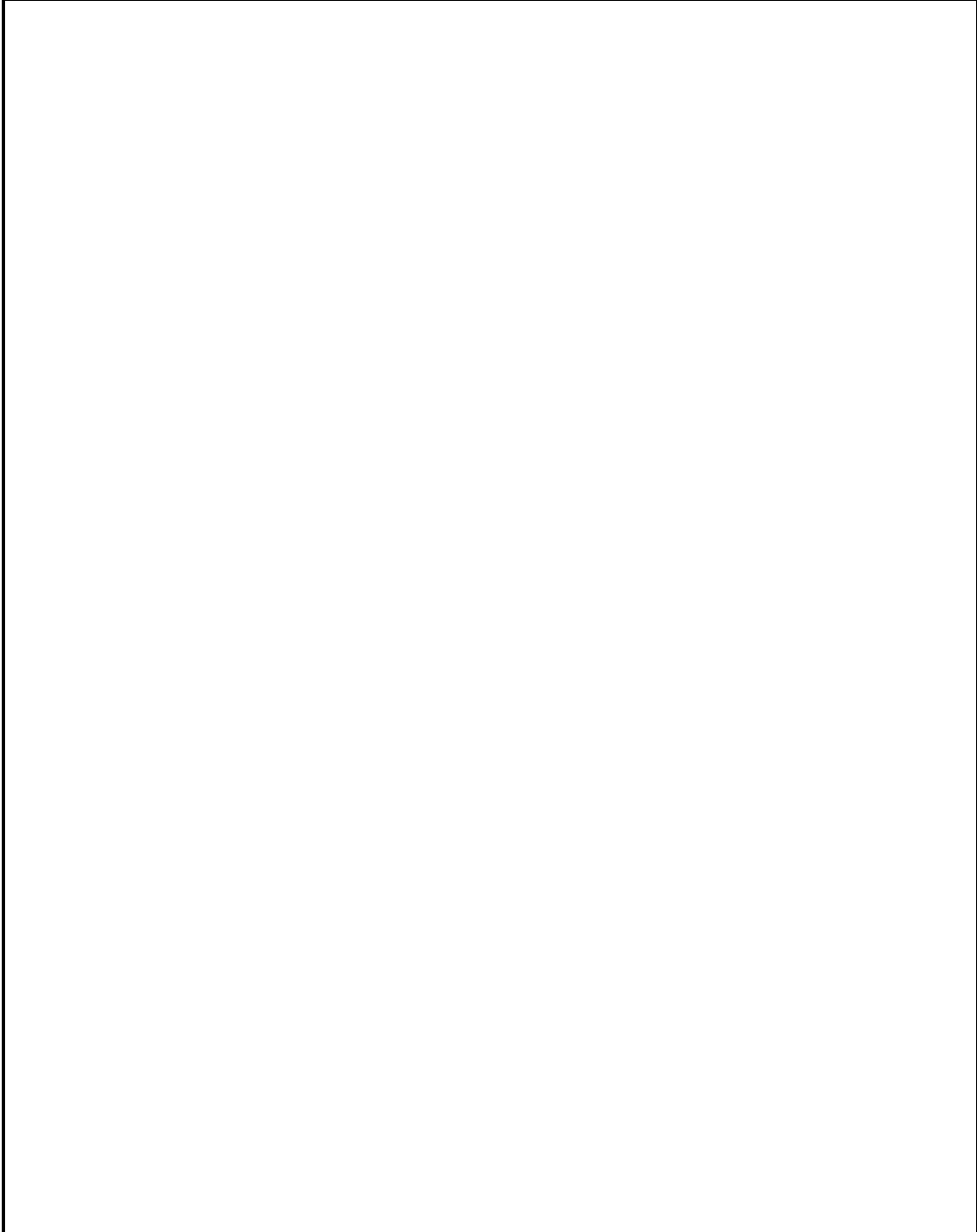
<sup>1</sup>Review: In one turn a player may move a single checker one square diagonally to an unoccupied square, or may cause a single checker to make a series of one or more jumps over the opponent’s checkers, to unoccupied squares on the other side. Checkers that are jumped over are removed. Non-king checkers may only move or jump in diagonal directions from the player’s home side of the board to the other side, but kings can move or jump in any diagonal direction. A non-king gets promoted to a king if it reaches the opponent’s first row. A player wins when he/she removes all the opponent’s checkers.

- (b) Show that your algorithm indeed uses polynomial space by giving an upper bound on the space used by your algorithm.

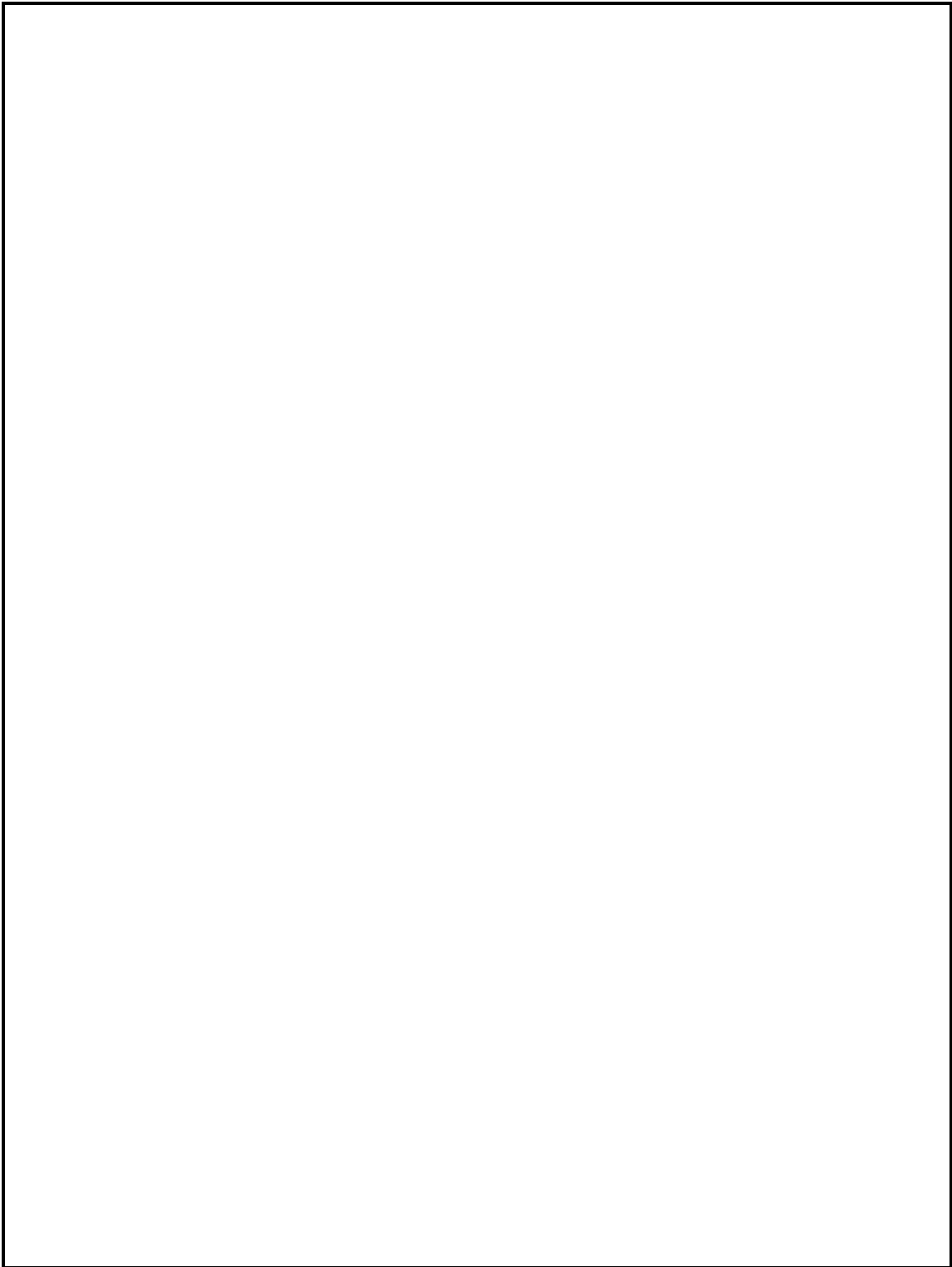


**Problem 5: Acceptance problem for NFAs (15 points)** Let the language  $A_{NFA}$  be the acceptance problem for NFAs, that is,  $A_{NFA} = \{ \langle M, w \rangle \mid M \text{ is an NFA and } w \text{ is a word accepted by } M \}$ .

- (a) Show that  $A_{NFA}$  is in the class NL, by describing an algorithm. Be sure to explain why your algorithm in fact works in nondeterministic log space.



(b) Prove that  $A_{NFA}$  is NL-hard, using a reduction from another problem we already know is NL-hard.





**Problem 6: Randomized complexity classes (10 points)** Suppose that  $L$  is a language over  $\{0, 1\}$ . Suppose that  $M$  is a probabilistic Turing machine that, on every  $w \in \{0, 1\}^*$ , on every computation branch, outputs one of  $\{\text{accept, reject, don't-know}\}$ . Suppose that, for every word  $w \in \{0, 1\}^*$ :

- (i) If  $w \in L$  then: the probability that  $M$  outputs “accept” is at least  $\frac{1}{4}$ , the probability that  $M$  outputs “reject” is 0, and the probability that  $M$  outputs “unknown” is at most  $\frac{3}{4}$ .
- (ii) If  $w \notin L$  then: the probability that  $M$  outputs “reject” is at least  $\frac{1}{4}$ , the probability that  $M$  outputs “accept” is at most  $\frac{1}{4}$ , and the probability that  $M$  outputs “unknown” is at most  $\frac{1}{2}$ .

Assume that there exists such a Turing Machine  $M$  that decides  $L$ .

- (a) Give an algorithm to decide membership in  $L$  that demonstrates that  $L \in BPP$ . (Use the definition of BPP from Sipser, p. 369, which says that the error probability is at most  $\frac{1}{3}$ , for every input.)

- (b) Prove that your algorithm shows that  $L \in BPP$ .

(c) Does your algorithm also demonstrate that  $L$  is in  $RP$ ? Why or why not?

(d) Does your algorithm also demonstrate that  $L$  is in  $coRP$ ? Why or why not?

**END OF EXAM.**

SCRATCH PAPER 1

## SCRATCH PAPER 2