

CS 475 Fall 2004 Final Exam (Form 0)

INSTRUCTIONS (read carefully)

- Fill in the following information giving name and ID.

NAME:

NET-ID:

- Put your name and ID on every page.
- There are 8 numbered problems. Make sure you have a complete exam.
- The point value of each problem is indicated next to the problem, and below.
- You have 3 hours.
- This is a closed book exam. No notes of any kind are allowed. Do all work in the space provided, using the backs of sheets if necessary. See the proctor if you need more paper.
- **Most of the proofs should be short, involving no more than two or three sentences.**

Problem	Possible	Score
1	15	
2	15	
3	10	
4	15	
5	15	
6	10	
7	10	
8	10	
Total	100	

Problem 1 (15 points)

This problem is T/F. Completely write out “True” if the statement is necessarily true. Completely write out “False” if the statement is not necessarily true. Each correct answer is worth one point. Ambiguous answers will not receive credit.

1. The set of all languages accepted by TMs is countably infinite.
2. If M is a DFA, then there exists a grammar G for $L(M)$ such that all productions are of the form $A \rightarrow BC$, $A \rightarrow a$, and $A \rightarrow \epsilon$.
3. Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA with no ϵ -transitions, and such that for all $q \in Q$ and $a \in \Sigma$, $|\delta(q, a)| \geq 1$. Call M a *determinization of N* if M is a DFA obtained from N by choosing for each q and a , exactly one element of $\delta(q, a)$ as the (deterministic) transition. Then $\cup\{L(M) : M \text{ is a determinization of } N\} = L(N)$.
4. If M has n states, and $S \subseteq L(M)$ is regular, then the minimum DFA for S has at most n states.
5. Every DFA that accepts $\{w : \text{the 4th character of } w \text{ is “0” and the 3rd character from the end of } w \text{ is “1”}\}$ has at least 32 states.
6. The languages denoted by regular expressions $(0 + 1^*)^*$ and $(00^* + 11^* + \epsilon)^*$ are equal.
7. If $A = \{0^n 1^m : n, m \geq 475\}$, $B = \{0^n 1^m : n + m \geq 475\}$, and $C = \{0^n 1^m : n - m \geq 475\}$, and $R = \text{foo}(A, B, C)$, where foo outputs some composition of regularity-preserving operations applied to inputs A, B , and C , then R is regular.
8. If the minimum state DFA for L_1 has n_1 states, the minimum state DFA for L_2 has n_2 states, then the minimum state DFA for $L_1 L_2$ has $n_1 n_2$ states.
9. There is an efficient algorithm to determine, given NFA M , whether or not $|L(M)| \geq 2$.
10. If G_1 and G_2 are CFGs, then $(L(G_1)^* - L(G_2)^*)^*$ is a CFL.
11. If G is a CFG, it is undecidable whether or not there exists a grammar G' with $L(G) = L(G')$ and such that G' contains no left-recursive productions.
12. There is a CFG generating all valid regular expressions.
13. If r is a regular expression, there is a CFG generating $\{w w^r : w \in L(r) \overline{L(r)}\}$
14. There is an efficient algorithm to determine, given an arbitrary string w and CFG G , whether or not there are at least two different derivations of G on w .
15. Define a 475-PDA to be one that accepts iff it consumes the input and leaves exactly 475 symbols on the stack. Then if L is any CFL, there is a 475-PDA accepting L .

Problem 2 (15 points)

In each of the following, information about a language L is given. Circle either **R**, **C**, **REC**, **RE**, **N** depending on which of the following holds:

R: Any language satisfying the information must be regular.

C: Any language satisfying the information must be context-free, but not all languages satisfying the information are regular.

REC: Any language satisfying the information must be recursive, but not all languages satisfying the information are context-free.

RE: Any language satisfying the information must be recursively enumerable, but not all languages satisfying the information are recursive,

N: Not all languages satisfying the information are recursive.

Be sure to erase completely. Ambiguously marked papers will not receive credit.

1. **R C REC RE N** L satisfies exactly one of the pumping lemma for regular languages and the pumping lemma for context-free languages.
2. **R C REC RE N** $L = A \cap B \cap C$, where A is regular, B context-free, and C the complement of a recursive language.
3. **R C REC RE N** $L = \{a^i b^j c^k : M_i \text{ on input } w_j \text{ halts and accepts in } k \text{ steps}\}$.
4. **R C REC RE N** L is accepted by a PDA that uses its stack at most 475 times.
5. **R C REC RE N** $L = L_1 \cap h^{-1}(L_2)$ where L_1 is regular, L_2 is a CFL, and h is a homomorphism.
6. **R C REC RE N** L is the complement of a recursively enumerable language.
7. **R C REC RE N** L is the shuffle of two context-free languages.
8. **R C REC RE N** $L = \{0^i : \text{there exists some number } x > 475 \text{ such that } i = x^{475}\}$.
9. **R C REC RE N** L is accepted by a one-state PDA by final state.
10. **R C REC RE N** $L = \{i : L(M_i) \text{ contains the string "cs475iscool"}\}$.
11. **R C REC RE N** L can be generated in the reverse of the standard lexicographical ordering.
(Careful!)
12. **R C REC RE N** $L = \text{shuffle}(A, B)$, where A is recursive and \overline{B} is context-free.
13. **R C REC RE N** $L \subseteq 0^*$ and for every k , $|L \cap \{0^k, 0^{k+1}, \dots, 0^{k+475}\}| = 1$.
14. **R C REC RE N** L contains all strings except those of the form $a^n b^n$.
15. **R C REC RE N** L contains all strings except those of the form $a^n b^n c^n$.

Problem 3 (10 points)

Suppose there are four problems (languages) A, B, C , and D . Each of these languages may or may not be recursively enumerable. However, we know the following about them:

- There is a reduction (i.e., an algorithm, not necessarily polynomial time) from A to B .
- There is a reduction from B to C .
- There is a reduction from D to C .

For each statement below, indicate whether each is

CERTAIN to be true, regardless of what problems A through D are.

MAYBE true, depending on what problems A through D are.

NEVER true, regardless of what problems A through D are.

1. _____ The complement of B is not recursive, but the complement of C is.
2. _____ If A is recursive, then the complement of B is recursive.
3. _____ If C is recursive, then the complement of D is recursive.
4. _____ If C is recursively enumerable, then $B \cup D$ is recursively enumerable.
5. _____ If C is recursively enumerable, then $B \cap D$ is recursively enumerable.

Suppose there are four problems (languages) A, B, C , and D . Each of these languages may or may not be in the class NP. However, we know the following about them:

- There is a polynomial-time reduction from A to B .
- There is a polynomial-time reduction from B to C .
- There is a polynomial-time reduction from D to C .

For each statement below, indicate whether each is

CERTAIN to be true, regardless of what problems A through D are, and regardless of the resolution of unknown relationships among complexity classes (of which $\mathcal{P} = \mathcal{NP}$ is one example.)

MAYBE true, depending on what problems A through D are, and/or depending on the resolution of unknown relationships such as “ $\mathcal{P} = \mathcal{NP}$?”

NEVER true, regardless of what problems A through D are, and regardless of the resolution of unknown relationships such as “ $\mathcal{P} = \mathcal{NP}$?”

6. _____ If A is NP-complete, then C is NP-complete.
7. _____ A is NP-complete and C is in \mathcal{P}
8. _____ If A is NP-complete and B is in \mathcal{NP} , then B is NP-complete.
9. _____ If C is NP-complete, then D is in \mathcal{NP} .
10. _____ B is not in \mathcal{P} and A is not in \mathcal{NP} .

Problem 4 (15 points)

Let Σ be a finite alphabet, let $G \subseteq \Sigma^*$ be a set of “good” words, and let $B \subseteq \Sigma^*$ be a set of “bad” words (not the kind you are muttering now). *Further, assume for this entire problem, that $G \cap B = \emptyset$, that is, there are no words that are both good and bad.* We are interested in designing a decision algorithm A to distinguish good from bad. Let $L(A)$ be the set of words accepted by algorithm A . The only requirements on A are that

- $G \subseteq L(A)$
- $B \cap L(A) = \emptyset$
- For each $x \in \Sigma^* - (G \cup B)$, A halts and either accepts or rejects; it doesn't matter which.

Make your arguments short! Use the space provided as a guide.

(a) **1pt:** Prove that if either G or B is recursive, then there is an A as specified above.

(b) **2pts:** Prove that if $G \cup B = \Sigma^*$ and each is recursively enumerable, then there is an A as specified above.

(c) **6pts:** Show that there are recursively enumerable G and B for which there is no A as specified above.
Hint: Let $G = \{i : M_i \text{ halts and rejects } i\}$ and let $B = \{i : M_i \text{ halts and accepts } i\}$, and now try to get hypothetical A 's head to explode.

Part (d) is on the next page.

(d) **6pts:** Show that if the complements \overline{G} and \overline{B} are each recursively enumerable, then there is an algorithm A as specified above.

Problem 5 (15 points)

While rummaging through the *Grab Bag of Theoretical Surprises* that she had won on eBay, Joanna Theorist spotted a (large) roll of “magic” tape. It had the label “Everything you always wanted to know about the halting problem, but didn’t have enough time to ask”.

In fact, the magic tape was infinitely long, and it indeed provided answers to the halting problem. In particular, in cell i there was a “1” written if M_i halts on input i , and there was a “0” written if M_i does not halt on input i . Thus the tape describes exactly which values i are in the set $K = \{i : M_i \text{ halts on input } i\}$.

Now Joanna was very excited, because with the magic tape she could compute many new and interesting things. She could connect the magic tape to her favorite Turing machine M , and let M look up answers on the magic tape as it pleased. For example, with this magic tape she could write a trivial TM program to decide membership in K : On input i , the TM simply looks in the i -th cell of the magic tape (by counting up to i as it moves), and accepts iff the i -th cell contains a “1”.

(a) 5pts: Goldbach’s conjecture is that every even number is the sum of two primes. Assuming that the magic tape is correct, explain how Joanna can use the tape to determine whether or not Goldbach’s conjecture is true.

(b) 5pts: Show that using the magic tape, even the diagonal language L_d becomes decidable.

Part (c) is on the next page.

- (c) Joanna is delighted with her good fortune. Her startling paper about Goldbach's conjecture rocked the world, and she has been cranking out new results one after another. In fact, she is convinced that she can decide any problem - that is produce an algorithm for *any* language - using the very helpful magic tape, of course. Explain why Joanna is mistaken, by showing that there are languages for which membership is still undecidable, even if the magic tape may be used in the computation. Hint: either give a counting argument, or construct a language similar to L_d . *Your argument should not be more than several sentences.*

Problem 6 (10 points)

Prove that every infinite recursive language contains a nonrecursive subset. *Hint:* It may help to think about why Σ^* contains a nonrecursive subset.

Does every infinite recursive language contain a *non-recursively enumerable* subset? In at most two sentences, explain why or why not.

Problem 7 (10 points)

Define a *set automaton* (SA) to be similar to a PDA, but instead of a stack, an SA has a data structure called a set. In addition to the input alphabet, there is a special (finite) set alphabet Γ . More precisely, an SA is a finite automaton, together with a set S that is initialized to contain only a special initial symbol $Z_0 \in \Gamma$. A transition is of the form $\delta(q, a, X) = (p, S')$, indicating that if the machine is in state q , with its read head scanning the symbol a , and if the element X is currently in the set S , then the machine may change to state p , remove X from S , add each element of S' to S (where $S' \subseteq \Gamma$), and move the read head forward one cell. As an example, if $\delta(q, 0, X) = (p, \{X, Y\})$, then on reading a “0” in state q , the machine may add Y to the set S provided that X is already in the set. Note that if X is not in S , then this transition is not applicable. Further note that if there is also a transition $\delta(q, 0, X') = (p', S')$, and both X and X' are currently in S , then the machine nondeterministically chooses which transition to apply. An SA accepts a string w if there is some sequence of transitions on input w that cause it to end up in a final state as it moves past the right end of w .

Part (a) 5pts:

What is the class of languages accepted by SA's? *Justify your answer in one or two short sentences.*

Part (b) 5pts:

Now define a multiset automaton (MSA) similar to an SA, except that S is now a multiset, and thus may contain many copies of any $X \in \Gamma$. A transition $\delta(q, a, X) = (p, S')$ is applicable iff there is at least one occurrence of X in the multiset S , and the result of applying the transition is to remove one occurrence of X from S , and to place into S all of the symbols of S' (where S' may be a multiset). For example, suppose the current state of an MSA is q , that it is scanning the symbol a , and that the multiset $S = \{X, X, Y\}$. Then the transition $\delta(q, a, X) = (p, \{Y, Y, Z\})$ is applicable, since S contains at least one occurrence of X , and if applied, the machine ends up in state p , with multiset $S = \{X, Y, Y, Y, Z\}$.

Prove or disprove: There exists a non-context-free language that is accepted by some MSA.
(Try to stay within the space provided.)

Part (c) (OPTIONAL worth maximum $\{2, 100 - x\}$ where x is your exam score) score if bonus))

What is the class of languages accepted by MSAs? Give an answer, and at most one sentence argument.

Problem 8 (10 points)

At the end of each semester Professor GoodHart is forced to solve the EXAM DESIGN problem. The professor has a list of problems and knows which students will *really enjoy* each of them. In order to ensure that the students enjoy the exam, it must include at least one question which each student will really enjoy. At the same time, the professor does not want to spend all of break grading. The exam should contain as few problems as possible.

- (a) 2pts: Define the language “ED” which most accurately captures the optimization problem EXAM DESIGN.

ED = {

}

- (b) 8pts: Prove that ED is NP-complete. *Hint:* vertex cover.