CS154 Final Examination

June 7, 2010, 7-10PM

Directions: CS154 students: answer all 13 questions on this paper. Those taking CS154N should answer only questions 8-13. The total number of points on this exam is 200, and the total of the points on the part for CS154N is 100. The exam is open book and open notes. Any materials may be used.

SOLUTIONS

Name: ____

I acknowledge and accept the Honor Code.

Sally Solutions

(signed) ________________________________

Problem	**Score**
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
Total	

If you are taking CS154N, skip to Problem 8.

Problem 1 (20 pts.) The *symmetric difference* of languages L and M, which we shall denote SD(L,M), is the set of strings that are in exactly one of L and M. For example, if L = {00, 101} and M = {11, 00}, then SD(L,M) = {11, 101}.

(a) Suppose L = L(**0*****1***) and M = L(**1*****0***). What are all the strings of length 3 or less in SD(L,M)?

01, 10, 001, 011, 100, 110

(b) Write a regular expression for SD(L,M).

00*11* + 11*00*

(c) Write a formula for SD(L,M) in terms of familar operations (i.e., those covered in class or in the text):

$(L\text{-}M) \cup (M\text{-}L)$

(d) Is SD(L,M) always a CFL? Either explain why or give a counterexample

*No. For example, if M = L(**0*****1*****2***) and L = $\{0^i1^j2^k \mid i \neq j \text{ or } i \neq k\}$, then SD(L,M) is the known non-CFL $\{0^n1^n2^n \mid n \geq 0\}$.*

***Comment**: On the exam, this question was flawed, implying that SD(L,M) was always a CFL. Those taking the exam in the room were told to treat it as "is SD(L,M) always context-free." For those few not taking the exam in the room itself, we gave credit for (d).*

Problem 2 (10 pts.) The language $L_1 = \{0^n1^n2^i \mid i \neq n\}$ is not a CFL, although the proof is rather tricky. However, suppose we are given that L_1 is not context-free. Now consider the language $L_2 = \{0^i1^n2^n \mid i \neq n\}$. We can prove L_2 is not context-free either, by applying two operations, both known to turn CFL's into CFL's, and thereby convert L_2 into L_1.

(a) What is the first operation you use? Give specifics of the operation, if necessary (e.g., if you use "intersection with a regular language," tell what regular language you choose).

Reversal

(a) What is the second operation you use? Give specifics of the operation, if necessary.

Apply the homomorphism h(2) = 0, h(1) = 1, and h(0) = 2. Note: these operations can also be applied in the opposite order.

Problem 3 (10 pts.) Let L = $\{0^i1^j \mid i \leq j\}$. Use the pumping lemma for regular languages to prove L is not regular. Suppose n is the pumping lemma constant. Begin by picking a string w to focus on.

(a) What string w do you choose? *$w = 0^n1^n$.*

The "adversary" picks a decomposition w = xyz, such that $|xy| \leq n$ and $|y| > 0$.

(b) Demonstrate that L is not regular by showing that xy^iz is not in L for some i. Below, tell your choice of i and explain why the resulting string is not in L.

Pick i = 2. Surely, y consists only of 0's, and it is not empty. Thus, xyyz has more 0's than 1's.

Problem 4 (15 pts.) Give a CFG for the language $\{0^i1^j2^k \mid i+j \geq 2k\}$.

S -> 00S2 | 0S | A | 01A2
A -> 11A2 | 1A | ε

You do not have to prove your grammar works, but in the table below, explain what each of your variables generates.

Variable	**Purpose**
S	*Start symbol. Generates at least two 0's for every 2, in strings of the form **0*****S2*** or (finally) **0***(**1**+ε)**A2***.*
A	*Generates at least two 1's for every 2, finally disappearing and leaving a string of the form **0*****1*****2***.*

Problem 5 (10 pts.) Suppose we use the construction of a CFG G whose language is N(P) from a given PDA P, as given in the text or in the class slides. Let P have s states. Suppose also that one of the rules of P, which we shall call *Rule R*, is δ(q, a, X) = {(r, β)}. Let β have length k. As a function of s and k, how many productions of grammar G with variable [qXp] on the left come from Rule R? Note: this problem differs from the one on the midterm not only in that the ambiguity has been cleared up, but the rule in question has a state r other than p.

s^{k-1} as long as k > 0. For k = 0, no productions are generated because r is not p.

Problem 6 (20 pts.) Suppose L_0 is some language over {0,1}*, and h is a homomorphism whose domain is {0,1} (that is, h(0) and h(1) are defined, and h(a) is not defined for any other symbol *a*). Let L_1 = $h(L_0)$ and $L_2 = h^{-1}(L_1)$. In this problem, you must figure out whether L_2 is guaranteed to be contained in L_0, and/or vice-versa. A proof must start by considering an arbitrary string w in one language, and without reference to the exact values of h(0) and h(1) prove that w is in the other language. A counterexample must be a specific L, h, and w, such that w is in one language and not the other.

(a) Is $L_0 \subseteq L_2$? Give a proof or a counterexample.

Yes. If w is in L_0, then h(w) is in L_1. Therefore w is in L_2.

(b) Is $L_2 \subseteq L_0$? Give a proof or a counterexample.

No. Let L = {0}, h(0) = ε, and h(1) = ε. Then L_1 = {ε}, and L_2 = {0,1}*.

Problem 7 (15 pts.) Here is a context-free grammar:

S -> AS | SB | 0
A -> BA | AS | 1
B -> SB | BA | 0

Note that each of the right sides AS, SB, and BA occurs twice. In the space below, show the table you get by applying the CYK algorithm to this grammar and the string 01100.

A,B,S				
A,B,S	*A,B,S*			
A,B	*A,S*	*A,B,S*		
A,B		*A,S*	*B,S*	
B,S	*A*	*A*	*B,S*	*B,S*
0	1	1	0	0

All students, including those taking CS154N, should do all the following problems.

Problem 8 (20 pts.) The Turing Machine M has start state q, final state f, and no other states. Its input symbols are 0 and 1, and the blank B is the only other tape symbol. It has only two transitions: δ(q,0) = (q,0,R) and δ(q,B) = (f,1,R). We shall give an inductive proof that L(M) contains every string of 0's. It is also true that L(M) contains only those strings, but we shall not prove that here. To begin, we shall prove by induction that for every string of the form $0^n\alpha$, where α is any string of 0's and 1's, $q0^n\alpha \vdash^* 0^nq\alpha$.

(a) (2 pts.) On what is your induction? *n*

(b) (2 pts.) What is the basis case? *n = 0*.

(c) (4 pts.) Prove the basis case:

$q\alpha \vdash^* q\alpha$ by the basis of the definition of $\vdash^*$.

(d) (8 pts.) Prove the induction (Note: this proof is very simple and you should not use more than four lines):

*$q0^n\alpha \vdash^* 0^{n-1}q0\alpha$ by the inductive hypothesis (applied with 0α in place of α). One move of M then gives us $q0^n\alpha \vdash^* 0^nq\alpha$. **Note**: a number of people tried to put the single step first as $q0^n\alpha \vdash 0q0^{n-1}\alpha \vdash^* 0^nq\alpha$, invoking the IH to claim the last n-1 moves were valid. The problem is, that the IH says nothing at all about what happens when we start with the head not at the left end of the ID.*

(e) (4 pts.) Complete the proof by showing that the inductively proved statement $q0^n\alpha \vdash^* 0^nq\alpha$ for any α can be used, together with a few simple observations, to conclude that M accepts 0^n for any n.

Let $\alpha = \varepsilon$. Then $q0^n \vdash^* 0^nq$. The second rule given for M says that $0^nq \vdash^* 0^n1f$, so M accepts 0^n.

Problem 9 (10 pts.) What are the satisfying truth assignments for the boolean formula (x+ -y)(y+ -z)? Note: we're using - for NOT, + for OR, and juxtaposition for AND, as in the slides. Represent a truth assignment by a string abc, where a, b, and c are each 0 (false) or 1 (true), and represent the truth values of x, y, and z, respectively. For example, the truth assignment where x and y are true and z is false is represented by 110.

000, 100, 110, and 111

Problem 10 (20 pts.) The Knapsack-With-Bonus Problem is: given a list of integers $i_1, i_2, \ldots, i_k$ and a *bonus* b, can we partition the integers into two disjoint sets S and T such that the sum over all integers i_j in S of i_j+b equals the sum over all i_j in T of i_j+b. For example, if the integers are 300, 400, 500, 600, and 700, and the bonus is b = 100, we could pick S = {300, 400, 500} and T = {600, 700}. Then, the sum for S would be (300+100) + (400+100) + (500+100) = 1500, and the sum for T would be (600+100) + (700+100) = 1500.

We know the Partition-Knapsack Problem discussed in class (partition a set of integers into two sets with equal sums) is NP-complete.

(a) Prove that Knapsck-With-Bonus is NP-complete by describing a polynomial-time reduction from Partition-Knapsack to Knapsack-With-Bonus. You are not required to prove your reduction works, but give a brief explanation to be considered for partial credit.

Transform the list of integers into the same list and a bonus b = 0.

(b) Describe a polynomial-time reduction from Knapsack-With-Bonus to Partition-Knapsack. You are not required to prove your reduction works, but give a brief explanation to be considered for partial credit.

Transform the list of integers by replacing each integer x by x+b.

Problem 11 (15 pts.) In the table below is an instance of Post's Correspondence Problem:

Index	**First String**	**Second String**
1	00	001
2	0101	11
3	101	01
4	01	010

There is no Turing-machine algorithm to decide whether or not there is a solution to PCP. But you're smarter than a Turing machine. Figure it out for this instance.

(a) (10 pts.) Either give a sequence of indexes that is a solution, or prove that no solution exists.

This turned out to be easier than I realized. 1, 3 is a solution; both strings become 00101.

(b) (5 pts.) Does your ability to answer (a) imply that human beings are able to solve problems that Turing machines (or equivalently, computers) cannot solve? You may assume you correctly answered (a), even if you did not. Explain briefly.

No. What we know is that there is no algorithm that takes as input an arbitrary instance of PCP and answers it. All you have done is solve one instance of PCP, not written a program that solves any given instance. A TM can do what you did. It simply checks that its input is the one given in this question and prints "1,3" on its tape.

Problem 12 (15 pts.) There are five languages (or equivalently, problems) A, B, C, D, and E. All we know about them is the following:

1. A is in **P**.
2. B is in **NP**.
3. C is NP-complete.
4. D is Recursive.
5. E is Recursively Enumerable but not Recursive.

For each of the five statements below, tell whether it is:

- CERTAIN to be true, regardless of what problems A through E are and regardless of the resolution of unknown relationships among complexity classes, of which "is **P** = **NP**?" is one example.
- MAYBE true, depending on what languages A through E are, and/or depending on the resolution of unknown relationships such as **P** = **NP**?
- NEVER true, regardless of what A through E are and regardless of the resolution of unknown relationships such as **P** = **NP**?

(a) There is a reduction from E to D. *NEVER. It would prove that E was recursive.*

(b) There is a polytime reduction from C to B. *MAYBE. B could be NP-complete as well. But if B were in **P**, then you would have a proof that **P** = **NP**.*

(c) There is a polytime reduction from A to B. *MAYBE. The reducer can solve A before it decides what to generate. However, if B is empty, there is no way to map inputs that have answer "yes." Similarly, if B is all strings, there is no way to handle the inputs with answer "no."*

(d) There is a polytime reduction from B to the complement of C.

*MAYBE. B could be very easy, say in **P**, in which case there is a trivial polytime reduction. But suppose B were NP-complete, and C were non-tautology, whose complement is the Tautology problem. Then all of **NP** would polytime reduce to Tautology, and therefore show that **NP** is contained in co-**NP**, something not known to be true.*

(e) There is a reduction from D to C. *CERTAIN. If D is recursive, the reducer can solve it and then decide which of two strings to map its input to. Note that the problem of (c), where the target of the reduction could be all or nothing, is not an issue here. C could not be NP-complete unless it had both true and false instances, since then there could not be a Karp reduction from any language in NP except itself.*

Problem 13 (20 pts.) Design a Turing machine that, given a sequence of 0's and 1's on its tape followed by blanks, deletes the first 0 or 1 and shifts all the following 0's and 1's one position left, and halts. The TM starts in state q_0, with the head at the leftmost of the 0's and 1's.

It is not important where the head winds up at the end, or in what state it halts. For example, given initial ID $q_0$001, it should halt with 01BB... on its tape, and given initial ID $q_0$1011, it should halt with 011BB... on its tape. Your TM should not use any tape symbols other than 0, 1, and B (the blank). In the table below, describe the transition function δ for your TM. That is, in the row for a state q and the column for a symbol X, show what δ(q,X) is. *Hint*: Copy one symbol at a time from its original position to the position to its left.

State	0	1	B
q_0	(p, B, R)	(p, B, R)	
p	(z, 0 , L)	(w, 1, L)	
z			(q_0, 0, R)
w			(q_0, 1, R)

In the table below, explain informally what each of your states does.

State	**Purpose**
q_0	Blank the current position and move right to pick up the symbol to the right.
p	Pick up that symbol, using the state (z=0, w=1) to remember it and move left to the position just made blank.
z	Deposit a 0 and move right to repeat the cycle with the next position.
w	Ditto, but deposit 1.