CS 154 - Introduction to Automata and Complexity Theory

Spring Quarter, 2009

Sample Final Exam (with Solutions)

This is a sample exam to help you prepare for the final exam. **This version of the handout contains sample solutions.**

Instructions: Answer all 11 questions in the space provided. The questions are NOT organized in order of difficulty; if you cannot solve a problem, move on to the next one. You have a total time of 3 hours and the maximum possible score is 360. The point allocation is roughly proportional to the time required to solve a problem, i.e., 2 points for each minute. We strongly recommend that you use a few minutes to go over the exam and plan your strategy. It is important that you be brief in your answers and ensure that it fits in the space provided. *You will lose points for a complicated solution, even if it is correct.*

This exam is an **open-book** and **open-notes** exam, i.e., you are allowed to consult both the text-book, your class notes and homeworks, and any of the handouts from us. You are not permitted to refer to any other material during the exam.

CS154N Students: You are required to solve problems 4–11 only for a maximum possible score of 240. You have 2 hours time to work on this exam.

Problem 1. [40 points]
Decide if the following statements are TRUE or FALSE. *You do not need to justify your answer. You will receive* 5 *points for each correct answer,* -2 *(that is NEGATIVE* 2*) for each incorrect answer, and* 0 *points if you choose not to answer at all.* Think carefully before answering.

1. _____ The union of a regular language with a context-free language must be context-free.

   TRUE: Since context-free languages are closed under union.

2. _____ The language $\{a^i b^j c^k d^l \mid i = j \text{ and } k = l\}$ is context-free.

   TRUE: this is just the concatenation of two context-free languages.

3. _____ The language $\{a^i b^j c^k d^l \mid i + k = j + l\}$ is context-free.

   TRUE: a PDA can easily add and subtract the number of a's, b's, c's, and d's on stack.

4. _____ The class of non-regular languages is closed under complementation.

   TRUE: If the complement of a language L is regular, then L itself must be regular to begin with; therefore, the complement of a non-regular language must be non-regular.

5. _____ The class of non-context-free languages is closed under complementation.

   FALSE: For example, $L_{ww} = \{ww \mid w \in \{0,1\}^*\}$ is non-context-free, but its complement is context-free.

6. _____ If L_1 and L_2 are context-free, then the language $L_1 - L_2$ must be context-free.

   FALSE: consider the case where $L_1 = \Sigma^*$; then the difference is like complementation and CFLs are not closed under complementation.

7. _____ If L_1 is context-free and L_2 is regular, then the language $L_1 - L_2$ must be context-free.

   TRUE: Since $L_1 - L_2 = L_1 \cap \overline{L_2}$ and the context-free languages are closed under intersection with regular languages.

8. _____ The context-free grammar $S \rightarrow a \mid ab \mid SS \mid Sb$ is ambiguous.

   TRUE: Since $S \Rightarrow ab$ and $S \Rightarrow Sb \Rightarrow ab$ shows that ab has two parse trees.

Problem 2. [50 points]
Define a language L to be *filled-out* if for every length $n \geq 0$, there exists a string of length n in L. That is, for all $n \geq 0$, $L \cap \Sigma^n \neq \emptyset$. This problem is concerned with decision problems involving filled-out regular languages.

a). [20 points] Consider any language L over the alphabet $\Sigma = \{0,1\}$. We define the operation *unary* over languages as follows: for each string $w \in L$, we replace it by a string of 1's of the same length as w. More formally, $unary(L) = \{1^{|w|} \mid w \in L\}$.

Show that regular languages are closed under the *unary* operation. (**Hint:** One approach is to start with a DFA M_2 for L, and construct a finite-state machine M_1 for $unary(L)$. Just give a brief high-level description of the construction with a brief justification. You do not need to provide any proofs.)

Solution: In the machine M_2, examine each transition. If the transition is on input symbol 1, leave it untouched. On the other hand, if the transition is on the input symbol 0, modify the transition to be on the input symbol 1. It is clear that M_1 only accepts strings of 1's.

Any accepting sequence of transitions in M_2 maps to an accepting sequence of the same length in M_1 with all transitions being on input symbol 1. Furthermore, for every accepting sequence of transition in M_1, there is an accepting sequence in M_2 of the same length. This implies that the language of M_1 is exactly $unary(L)$.

The resulting machine M_1 is an NFA since a state in M_2 with outgoing transitions on 0 and 1 will now have multiple outgoing transitions on the symbol 1.

b). [15 points] Consider a DFA M over the unary alphabet $\Gamma = \{1\}$. Give a decision algorithm for testing whether $L(M)$ is filled-out or not. You must justify the correctness of your algorithm.

Solution: For unary alphabets, being filled-out is equivalent to having $L = \Gamma^*$, which in turn is equivalent to having $\overline{L} = \emptyset$. Our decision procedure will first construct a DFA $\overline{M}$ for $\overline{L}$ by complementing the final states of M. Then, we can test whether $L(\overline{M})$ is empty or not using the decision algorithms for DFAs described in the book.

c). [15 points] Consider a DFA M over the binary alphabet $\Sigma = \{0,1\}$. Give a decision algorithm for testing whether $L(M)$ is filled-out or not. You must justify the correctness of your algorithm.

Solution: To test whether $L(M)$ is filled-out or not, it suffices to test whether $unary(L)$ is filled-out or not. So, we start with the DFA M for L and convert it to an NFA M_1 for $unary(L)$,

as described in part (a). Then, using the standard subset construction, we convert M_1 into a DFA M_2 for $unary(L)$. Now, we can use the decision algorithm from part (b) to test whether $L(M_2)$ is filled-out or not.

Problem 3. [30 points]
Show that the following language is not context-free using the Pumping Lemma.

$$L = \{(0^k 1^k)^k \mid k \geq 1\}.$$

Solution: Assume that L is context-free and apply the pumping lemma. Let n be the pumping lemma constant and define $m \geq \max\{2, n\}$ (basically we need m to be greater than 1 for the following proof). Choose $z = (0^m 1^m)^m$. Let the pumping lemma decompose z into $uvwxy$ with $|vwx| \leq n$ and $|vx| > 0$. Consider the pumped string $z' = uv^0wx^0y = uwy$

Suppose that vwx contains only one symbol, say 0's without loss of generality (a similar argument applies for the case of 1's). It is clear that vwx must belong to a single block of 0's. Then, the string z' will have a block of 0's of length less than n, while all other blocks of 0's and 1's will have length exactly n, giving us a string that does not belong to L — a contradiction.

Otherwise, vwx can straddle at most two consecutive blocks since $|vwx| \leq n \leq m$, say the first block is of 1's and the second block is of 0's (a similar argument applies to the case where the 1's and 0's are switched). Now, in the string z' we would have reduced the size of at least one of these two blocks to below m, while all other blocks of 0's or 1's will stay at length m. Again, this gives us a string that does not belong to L – a contradiction.

Problem 4. [40 points]
Consider the following classes of languages numbered as **1**–**7**.

1. Empty.

2. Finite.

3. Regular.

4. Context-free.

5. Recursive.

6. Recursively enumerable.

7. All possible languages.

For each of the following languages specify the *lowest-numbered* class to which it *surely* belongs; you *do not* need to provide any justification for your answers. For example, for a context-free language L that is not regular, the right answer is 4 (even though L clearly belongs to all classes of number larger than 4). Similarly, suppose we know that L is recursively enumerable and that, although it could possibly be recursive, the available information does not guarantee that it is recursive, then the right answer is 6.

1. _____ **Class 7** _____ The complement of an undecidable language.

2. _____ **Class 5** _____ The complement of a language in NP.

3. _____ **Class 5** _____ The complement of a context-free language.

4. _____ **Class 7** _____ The intersection of a recursive language and a language that is *not* recursively enumerable.

5. _____ **Class 6** _____ The intersection of a recursive language and a recursively-enumerable language.

6. _____ **Class 5** _____ $L = \{a^i b^j c^k d^l \mid i = j \text{ and } j = k\}$.

7. _____ **Class 4** _____ $L = \{a^i b^j c^k d^l \mid i = l \text{ and } j = k\}$.

8. _____ **Class 3** _____ $L = \{a^i b^j c^k d^l \mid i \times j \times k \times l \text{ is divisible by } 5\}$.

Problem 5. [20 points]
This problem refers to languages over some finite alphabet Σ. We suggest that you use closure properties to answer the following questions.
(a). [10 points] Show that if L is a recursive language and F is a finite language, then $L - F$ is recursive. (The notation $L - F$ denotes all strings in L that do not belong to F.)

Solution: Clearly, $L - F = L \cap \overline{F}$. All finite languages are regular and hence recursive. Since recursive languages are closed under complement, $\overline{F}$ must be recursive. Since recursive languages are closed under intersection, $L \cap \overline{F}$ must be recursive.
(b). [10 points] Show that if L is a non-recursive language and F is a finite language, then $L \cup F$ is non-recursive.

Solution: Define $F_1 = F \cap \overline{L}$ – these are the strings in F that do not belong to L. Clearly, F_1 is finite since it is a subset of a finite language F.

Assume that $L_1 = L \cup F$ is recursive, for the sake of contradiction. Now, observe that $L = L_1 - F_1$. By part (a), if L_1 is recursive then $L_1 - F_1$ must also be recursive. This implies that L is recursive which contradicts the given fact that L is non-recursive. If follows that $L_1 = L \cup F$ must also be non-recursive.

Problem 6. [20 points] Prove that the following language L_{comp} is non-recursive:

$$L_{comp} = \{<M_1, M_2> \mid \ L(M_1) = \overline{L(M_2)}\}.$$

The strings in L_{comp} encode two Turing machines M_1 and M_2 such that the language of M_1 is the complement of the language of M_2.

To prove this result you may use the fact that the language L_{all} is non-recursive, where

$$L_{all} = \{<M> \mid \ L(M) = \Sigma^*\}.$$

Solution: We give a reduction from L_{all} to L_{comp}. Since L_{all} is known to be non-recursive, it follows that L_{comp} is non-recursive.

The reduction f works as follows: Given the machine M which is an instance of L_{all}, the reduction f creates an instance of L_{comp} by setting $M_1 = M$ and choosing M_2 as a Turing machine with the empty language. We can choose M_2 to be any Turing machine without any final states, e.g., a Turing with only an initial state, no final states and no transitions. The reduction f is easily computable by a halting Turing machine. We observe that $\overline{L(M_2)} = \Sigma^*$ so $L(M_1) = \overline{L(M_2)}$ if and only if $L(M_1) = \Sigma^*$. It follows that $< M_1, M_2 >$ is in L_{comp} if and only if $M \in L_{all}$.

Problem 7. [40 points]
Consider the following two languages over the alphabet $\Sigma = \{0, 1\}$.

$$L_U = \{< M, w > \mid \text{TM } M \text{ accepts input } w\}$$

$$L_{15} = \{< \widehat{M} > \mid \ |L(\widehat{M})| = 15\}$$

The language L_U is the universal language, while the language L_{15} corresponds to the following decision problem: given a Turing machine $\widehat{M}$, does its language have size exactly 15?

Show that L_{15} is undecidable via a reduction from L_U, assuming that L_U is undecidable. (**Hint:** The number of strings of length at most 3 is exactly 15.)

Solution: We propose the following reduction from L_U to L_{15}.

> Given a Turing machine M and input w, construct a Turing machine $\widehat{M}$ which behaves as follows on being given input $\widehat{w}$.
>
> 1. $\widehat{M}$ simulates the behavior of M on input w;
> 2. if M halts on w and accepts, then $\widehat{M}$ examines its own input $\widehat{w}$, halting in a final state if $|\widehat{w}| \leq 3$ and halting in a non-final state otherwise;
> 3. if M halts on w and rejects, then $\widehat{M}$ rejects its own input $\widehat{w}$.

This reduction can be computed by a halting Turing machine.

Observe that in the case where $w \in L(M)$, the language $L(\widehat{M})$ is exactly the set of strings over $\Sigma = \{0, 1\}$ of length at most 3, which implies that $|L(\widehat{M})| = 15$. In the case where $w \notin L(M)$, we have that $L(\widehat{M})$ is the empty language and hence $|L(\widehat{M})| = 15$. Thus, we have established that: $< M, w > \in L_U$ if and only if $< \widehat{M} > \in L_{15}$.

Thus, since L_U reduces to L_{15}, we obtain that L_{15} is undecidable.

Problem 8. [15 points]
Recall that Rice's Theorem states that every nontrivial property of the recursively enumerable languages is undecidable. For each of the languages given below, choose one of the following three statements which best describes the situation.

A. The given language is decidable.

B. The given language is undecidable and this follows from a *direct* application of Rice's Theorem.

C. The given language is undecidable but we cannot directly apply Rice's Theorem to show this.

Indicate your choice by writing A, B, or C for each language description. Note that in all cases M refers to a Turing machine.

1. $L = \{< M > \mid L(M)$ is a context-free language$\}$.
2. $L = \{< M > \mid L(M)$ is recursively enumerable$\}$.
3. $L = \{< M > \mid$ when started on a blank tape, Turing machine M eventually halts in a final state$\}$.
4. $L = \{< M_1, M_2 > \mid L(M_1) = L(M_2)\}$.
5. $L = \{< M, w > \mid M$ accepts w in less than 100 steps$\}$.

Solution:

1. B (clearly a property of languages)
2. A (all answers are "YES" by definition)
3. B (since the property reduce to $\epsilon \in L(M)$, a property of languages)
4. C (not a property of recursively enumerable languages, but of pairs of languages)
5. A (just run the machine for 100 steps)

Problem 9. [40 points]
Decide if the following statements are TRUE or FALSE. *You do not need to justify your answer. You will receive* 5 *points for each correct answer,* -2 *(that is NEGATIVE* 2*) for each incorrect answer, and* 0 *points if you choose not to answer at all.* Think carefully before answering.

1. _____ **TRUE** _____ The class NP is closed under intersection.

2. _____ **FALSE** _____ If L_1 is in NP and L_2 is in P, then $L_1 \cap L_2$ cannot be NP-complete.

3. **FALSE** It is known that the complement of the satisfiability problem does not have a polynomial-time algorithm.

4. **TRUE** If B is recursively enumerable and $A <_{poly} B$, then A must be recursively enumerable.

5. **FALSE** If A is recursive and $A <_{poly} B$, then B must be recursive.

6. **FALSE (what if P=NP?)** Suppose L_1 is in P and L_2 is NP-complete. Then, it is known that there cannot be a polynomial-time reduction from L_2 to L_1.

7. **FALSE** If A is NP-complete and $A <_{poly} B$, then B must be NP-complete.

8. **TRUE** If $A <_{poly} B$, then it is possible that A is in P but B is not in NP.

Problem 10. [30 points]

(a) [10 points] It is not known whether the complement of every NP language is also in NP. The following is a fallacious proof that this is always the case. Find the error in the proof.
Theorem. If $L \in NP$, then $\overline{L} \in NP$.
Proof: Given any polynomial-time NTM N for the language L, make its final states into non-final states and vice versa. Clearly, the resulting NTM N' accepts exactly the language $\overline{L}$ and has a polynomial running time. Hence, $\overline{L} \in NP$.

Solution: The machine N' will accept all those strings for which N behaves as follows: *at least one execution path of N leads to a non-final state of N.* But this could include many strings which are accepted by N. In fact, the set of strings rejected by N' are only those strings which are rejected by N along *all possible* execution paths of N. Thus, it is clear that the machine N' does not accept exactly the language $\overline{L}$ but in fact some superset of that language.

(b) [10 points] Let $L_1, L_2 \subseteq \Sigma^*$ be such that $L_1 <_{poly} L_2$. Prove that $\overline{L}_1 <_{poly} \overline{L}_2$.

Solution: We can use exactly the same reduction as in $L_1 <_{poly} L_2$. Let f be the reduction function which gives $L_1 <_{poly} L_2$. Clearly, it has the following properties: f is computable in polynomial time, and $w \in L_1$ if and only if $f(w) \in L_2$. Observe that the latter property is equivalent to saying that $w \notin L_1$ if and only if $f(w) \notin L_2$. This implies that $\overline{L}_1 <_{poly} \overline{L}_2$.

(c) [10 points] Suppose that L_1 is NP-complete and $\overline{L}_1 \in NP$. Prove that for all $L \in NP$, it must be the case that $\overline{L} \in NP$.

Solution: Since L_1 is NP-complete, it must be NP-hard which means that for all languages $L \in NP$ it must be the case that $L <_{poly} L_1$. By part (b) of this problem, it follows that for all languages $L \in NP$ it must be the case that $\overline{L} <_{poly} \overline{L}_1$. Now, since $\overline{L}_1 \in NP$, it must be the case that for all languages $L \in NP$, $\overline{L} \in NP$. (This is based on the fact that: if $X <_{poly} Y$ and $Y \in NP$, then $X \in NP$.)

Problem 11. [35 points]
For a 3-CNF (or 3-SAT) formula, define a NAT truth assignment (NAT stands for Not-All-True) as a *satisfying* truth assignment with the additional guarantee that *every* clause in the formula contains at least one FALSE literal. Your goal is to establish the NP-completeness of the NAT-SAT problem:

INPUT: A 3-CNF boolean formula G.
PROBLEM: Does G have a satisfying NAT truth assignment?

We propose the following reduction from 3-SAT to NAT-SAT. The reduction starts with a 3-SAT formula $F(X_1, \ldots, X_n)$ with m clauses $C_1, \ldots, C_m$, and produces another 3-SAT formula $G(X_1, \ldots, X_n, Y_0, Y_1, \ldots, Y_m)$, where $Y_0, \ldots, Y_m$ are new variables. The reduction proceeds as follows: each clause $C_i = Z_{i1} \vee Z_{i2} \vee Z_{i3}$ in F is replaced by the following two clauses in G

$$(Z_{i1} \vee Z_{i2} \vee Y_i) \text{ and } (\overline{Y}_i \vee Z_{i3} \vee Y_0).$$

Note that Y_0 appears in the second clause replacing each clause, but the new variable Y_i is only used in the two clauses replacing the specific clause C_i.

(a) [5 points] Show that negating the value of every variable in a satisfying NAT truth assignment gives another satisfying NAT truth assignment.

Solution: In a NAT truth assignment, each clause contains at least one TRUE literal and at least one FALSE literal. If we negate the value of every variable in a NAT truth assignment, the value of each literal in each clause will also get negated. Thus, the resulting truth assignment will still have at least one TRUE literal and at least one FALSE literal, making it another NAT trust assignment.

(b). [15 points] Show that if G has a satisfying NAT truth assignment, then F is satisfiable.
Solution: Suppose G has a satisfying NAT trust assignment T_G. We assume that in T_G the value assigned to Y_0 is FALSE; otherwise, if Y_0 is TRUE, we could use part (a) and negate each variable to obtain another satisifying NAT truth assignment where Y_0 is indeed FALSE.

We now show that each clause C_i in F must be satisfied by T_G's assignment of values to the variables $X_1, \ldots, X_n$. If T_G does not assign value TRUE to any of the literals in C_i, then the first clause replacing C_i causes Y_i to be assigned TRUE in the NAT truth assignment. But now, the second clause replacing C_i has all three of its literals set to FALSE, which gives a contradiction. This means that T_G must ensure that at least one Z_{i1}, Z_{i2}, and Z_{i3} is assigned TRUE.

(c). [10 points] Show that if F is satisfiable, then G has a satisfying NAT truth assignment.

Solution: Given a truth assignment T_F for F, we construct a NAT truth assignment T_G for G as follows. For the variables $X_1, \ldots, X_n$ use the same values as in T_F. The variable Y_0 is assigned the value FALSE. For each Y_i, assign it the value TRUE if neither of Z_{i1} and Z_{i2} are TRUE; otherwise, set Y_i to FALSE. It is easy to verify that each of the two clauses replacing C_i now have at least one TRUE literal and at least one FALSE literal.

(d). [5 points] Assuming that the statements in (b) and (c) are true, prove that NAT-SAT is NP-complete. **SOlution:** The reduction is a straight-forward replacement of clauses and can be computed in polynomial-time. By parts (b) and (c), the reduction is correct in that: F is satisfiable if and only if G has a satisfying NAT trust assignment.

All that remains to be shown is that NAT-SAT is in NP. For this, consider the NTM M which first guesses a truth assignment and then verifies that each clause has at least one TRUE literal and one FALSE literal. Clearly, this correctly solves NAT-SAT in nondeterministic polynomial time.