CS4510 Automata and Complexity

Spring 2015 // Final Examination// Total: 90 points

NOTES:

- If necessary, make *reasonable* assumptions. But, please make sure that you state your assumptions.
- Please write *neat*, *legible*, *precise*, and *succinct* answers.
- You can use without re-proving: (a) Theorems, Lemmas, and Examples either covered in the class or covered in chapters 1 to 5, 7 and 8 of the Sipser text, and (b) homework problems from this course. You may not use any other source.
- Please see the tests, practice questions, and homework assignments from this course in addition to these practice questions.
- The *clique* problem is the following: Given an undirected graph $G = (V, E$ and a natural number k, is there a subset $V' \subseteq V$ such that: (a) $|V'| = k$, and (b) for any $u, v \in V'$, $(u, v) \in E$?

  The *clique* problem is $\mathcal{NP}$-Complete.
- The *cnfsat* problem is the following: Given a Boolean formula in conjunctive normal form, is there a truth assignment to its variables that make the formula evaluate to true?

  The *cnfsat* problem is $\mathcal{NP}$-Complete.

1. Which if the following languages are regular and which are not? Give justification.

   (a) $\{a^n b^{2m} \mid n, m \geq 0\}$.

   (b) $\{a^n b^m \mid n = 2m\}$.

   (c) $\{xcx \mid x \in \{a, b\}^*\}$.

   (d) $\{xcy \mid x, y \in \{a, b\}^*\}$.

2. Let $\Sigma = \{0, 1\}$. Let $\bar{x}$ denote the Boolean complement of x; that is, $\bar{x}$ is the string obtained by changing all the 0's to 1's and 1's to 0's. Let **rev** x denote the reverse of x: that is **rev** x is the string x written backwards.

   Give a context-free grammar for the language $\{x \mid \textbf{rev}\ x = \bar{x}\}$.

   Briefly justify why your grammar generates this langauge.

3. Let A and B two regular languages over a finite alphabet Σ. Show that the language defined below is context-free:

$$A \diamond B = \{xy \mid x \in A \text{ and } y \in B \text{ and} |x| = |y|\}$$

   Solution: Let M_1 be a DFA that recognizes A and M_2 be a DFA that recognizes B. Construct a PDA P for $A \diamond B$ as follows: On input w, guess a partition of w into x and y, simulate M_1 on x pushing a symbol 1 onto the stack for every symbol in x, simulate M_2 on y popping a symbol from the stack for every symbol in y, and accept iff: (a) both M_1 and M_2 accept and (b) the stack is empty.

4. Let $\mathcal{C}$ be the class of all context-free languages that is closed under *intersection.* That is, for any two languages in $\mathcal{C}$, their intersection is also in $\mathcal{C}$.

(a) Answer TRUE/FALSE justifying your claim: There is a regular language that is not in $\mathcal{C}$.

Solution: FALSE. A regular language is context-free. Further, regular languages are closed under intersection.

(b) Answer TRUE/FALSE justifying your claim: All context-free languages are in $\mathcal{C}$.

Solution: Let $\Sigma = \{a, b, c\}$ and define $L_1 := \{a^i b^i c^j | i, j \geq 0\}$, $L_2 := \{a^i b^j c^j | i, j \geq 0\}$. Then, L_1 and L_2 are context-fee languages. Their intersection $L_1 \cap L_2$ is $\{a^i b^i c^i | i \geq 0\}$ which we saw in class to be not context-free. Therefore, they don't belong to $\mathcal{C}$.

5. Let $ALL_{\text{CFG}} = \{\langle G \rangle \ |G \text{is a context} - \text{free grammar and } L(G) = \Sigma^*\}$. Let $\overline{ALL_{\text{CFG}}}$ be the complement of the language ALL_{CFG}. That is,

$\overline{ALL_{\text{CFG}}} = \{\langle G \rangle \mid G \text{ is a context} - \text{free grammar with alphabet } \Sigma \text{ and } L(G) \neq \Sigma^*\}.$

(That is, for any $G \in \overline{ALL_{\text{CFG}}}$, there is at least one string in Σ^* that is not in $L(G)$.)

(a) Show that $\overline{ALL_{\text{CFG}}}$ is Turing-recognizable.

Solution: Enumerate strings over the alphabet of G in order. For each string w generated test to see if it is in $L(G)$ using the decider for A_{CFG}. (We saw that A_{CFG} is decidable.) If $L(G) \neq \Sigma^*\}$ then there is a string $w \notin L(G)$ and it will be found eventually. If $L(G) = \Sigma^*\}$ then this procedure will not halt.

(b) Show that A_{TM} is not mapping reducible to ALL_{CFG}.

Solution: If there is such a reduction then there is a mapping reduction from $\overline{A_{\text{TM}}}$ to $\overline{ALL_{\text{CFG}}}$ which would imply that $\overline{ALL_{\text{CFG}}}$ is not Turing recognizable.

6. Consider the following two problems about Turing machines.

(a) $P_1 = \{\langle M \rangle \mid L(M) \text{ has at least 100 strings}\}$.

(b) $P_2 = \{\langle M \rangle \mid L(M) \text{ is accepted by a DTM that does not halt in an even number of steps}\}$.

(a) Show that one of the above two properties is *non-trivial* and the other is *trivial.* (A problem P is *nontrivial* if it holds for some, but not all, Turing machines.)

Solution: The language P_2 is trivial and the language P_1 is non-trivial. Clearly the language $L_1 = \Sigma^*$ (a Turing machine which always accepts) has more than 100 strings, and the language $L_2 = \phi$ (a Turing machine which always rejects) has less than 100 strings. Hence, $L_1 \in P_1$ whereas $L_2 \notin P_1$.

We show that P_2 is trivial, that is it contains all possible Turing machines. Let M be a Turing machine in P_2. Make a new Turing machine M' as follows:

M' makes an initial *dummy step*, and after that works as M. At each step of M, M' executes an additional dummy step in addition to M's step. And M' accepts iff M accepts. Thus M' takes $2k + 1$ steps for k steps of machine M. The machine M' recognizes the same language as M. Further, for all halting computations, M' halts in an odd number of steps. Therefore, the Turing machine M' also belongs to property P_2.

(b) For the nontrivial problem, show that the membership of a Turing machine M in it depends only on the language of M. That is, for any Turing machines M_1, M_2, where $L(M_1) = L(M_2)$, we have $\langle M_1 \rangle$ is in the problem iff $\langle M_2 \rangle$ is in the problem. (Therefore, we can conclude from Rice's theorem that the nontrivial problem is undecidable.)

Solution: If $L(M_1) = L(M_2)$ then $|L(M_1)| = |L(M_2)|$, and both will either contain ≥ 100 strings or less than 100 strings.

7. Show that $\mathcal{P} = \mathcal{NP}$ if there is a polynomial time algorithm to transform an input formula in conjunctive normal form into an equivalent formula in disjunctive normal form.

   Solution:

   - Suppose there is a poly-time algorithm M to convert a CNF formula ϕ into an equivalent DNF formula ψ. Since the machine itself takes polynomial time, the new formula ψ must be of size at most $O(|\phi|^c)$ where $c > 0$ is the exponent in the running time of M. If there is a poly-time algorithm for deciding whether a DNF formula is satisfiable, then we can apply it on ψ to decide whether it is satisfiable. Since ϕ and ψ are equivalent, this will constitute a poly-time algorithm for *cnfsat*. Since *cnfsat* is an $\mathcal{NP}$-complete problem, by the argument below, we have that $\mathcal{P} = \mathcal{NP}$.
   - By a property of poly-time mapping reducibility, if a language A is reducible to a language B and $B \in \mathcal{P}$ then $A \in \mathcal{P}$. Therefore, if an $\mathcal{NP}$-complete problem $A \in \mathcal{P}$ then $\mathcal{NP} = \mathcal{P}$ since every language in $\mathcal{NP}$ is reducible to A.
   - Finally, we describe a poly-time algorithm for deciding DNF-satisfiability. A DNF formula ψ consists of an 'or' of 'and clauses, and hence is satisfiable iff at least one of the clauses c is satisfiable. Since each clause is an 'and' of literals, it can be satisfied iff all literals are assigned value true. This is possible iff both x and $\neg x$ do not occur in the same clause. Thus to check for satisfiability, we go through all the clauses $c \in \psi$ and output 'no' iff we find a clause with complementary literals x and $\neg x$. Clearly this algorithm runs in polynomial time.

8. Show that the following problem is $\mathcal{NP}$-Complete:

   INPUT: an undirected graph G and a clique H in G

   PROBLEM: H is NOT a maximum-sized clique in G.

   Solution: We call this problem NOT-MAX-CLIQUE(G,H). First, we show that the problem is in NP.

   A nondeterministic verifier V will guess a set $V' \subseteq V$ of size $|H| + 1$, and then check whether V' forms a clique in G. If yes, then it will accept as V' is a clique of size greater than H. Otherwise, it will reject. Clearly, V accepts iff H is not the maximum size clique in G.

   To show NP-completeness, we reduce from the *clique* problem. Let $(G(V,E), k)$ be an instance of the *clique* problem. We make a new graph G' on $|V| + k$ vertices as an instance of the above problem. G' consists of *two disjoint graphs*: the first is (a copy of) the graph G itself, and the second is H, a clique on k vertices. Now, we give (G', H) as an instance of the NOT-MAX-CLIQUE problem.

   Now we proceed with the proof of correctness.

   First, if G has a clique of size greater than k, then G' will have clique of size greater than H (since $|H| = k$) in its own copy of G. Hence, H is not a maximum size clique in G'.

   Now suppose than H is not a maximum size clique in G', and that there is a larger clique H'. But then since the larger clique H' should completely lie inside the copy of G in G' since G and H are disjoint. But then we will have a clique H' of size greater than $k = |H|$ in G itself. This will form a solution to the *clique* problem.

9. Show that, if every $\mathcal{NP}$-hard language is also $\mathcal{PSPACE}$-hard then $\mathcal{PSPACE} = \mathcal{NP}$.

   Solution: Suppose every $\mathcal{NP}$-hard language is also $\mathcal{PSPACE}$-hard. Then , *cnfsat* is $\mathcal{PSPACE}$-hard. Therefore, every language in $\mathcal{PSPACE}$ is reducible to *cnfsat*. Since $cnfsat$ is in $\mathcal{NP}$, we have $\mathcal{PSPACE} \subseteq \mathcal{NP}$. The conclusion follows since $\mathcal{NP} \subseteq \mathcal{PSPACE}$.

10. Let *ODD-SAT* be the following problem:

    INPUT: a CNF Boolean formula F.

    PROBLEM: F has an odd number of satisfying assignments.

    Show that *ODD-SAT* is in $\mathcal{PSPACE}$.

    Solution: Evaluate the formula for all possible truth assignments and count the number of assignments for which the formula is satisfiable. Accept iff this number is odd. This takes polynomial space. (See example 8.3 from the text.)

11. The graph isomorphism ($\mathcal{GI}$) problem is the following: Given two undirected graphs (G, H), is there a bijection ϕ between their vertices such that adjacency's are preserved? (That is, for any edge (u, v) in G we have that $(\phi(u), \phi(v))$ is an edge in H.)

    Let $\mathcal{GNI}$ denote the complement of $\mathcal{GI}$. Prove the following claim: If $\mathcal{GNI}$ is $\mathcal{PSPACE}$-Complete then $\mathcal{GI}$ is $\mathcal{NP}$-Complete.

    Solution: First, show that $\mathcal{GI}$ is in $\mathcal{NP}$ (left as an exercise).

    Now, From the hypothesis it follows that all languages in $\mathcal{PSPACE}$ is reducible to $\mathcal{GNI}$. In particular, the language *unsat* (the complement of *cnfsat*) is reducible to $\mathcal{GNI}$. This is because, *unsat* is in $\mathcal{CONP}$ and $\mathcal{CONP}$ is in $\mathcal{PSPACE}$. Therefore, *cnfsat* is reducible to $\mathcal{GI}$. Since all languages in $\mathcal{NP}$ is reducible to *cnfsat* and since reducibility is transitive, it follows that all languages in $\mathcal{NP}$ is reducible to $\mathcal{GI}$. Since $\mathcal{GI}$ is in $\mathcal{NP}$, the claim follows .

12. Construct a generalized geography instance given a quanified Boolean formula.

13. What is the time taken by the simulating DTM in Savitch's theorem?

14. Example 8.18 of the text.