CSCI 1010 - Fall'25 Theory of Computation Lorenzo De Stefani

HW9

Due: n/a

Reminder: Submit your assignment on Gradescope by the due date. Submissions must be typeset. Each page should include work for only one problem (i.e., make a new page/new pages for each problem). See the course syllabus for the late policy.

While collaboration is encouraged, please remember not to take away notes from any labs or collaboration sessions. Your work should be your own. Use of other third-party resources is strictly forbidden.

Please monitor Ed discussion, as we will post clarifications of questions there.

Problem 1

Let ϕ be Boolean Formula in CNF, and let $ALLBUT2SAT = \{< \phi > \mid \exists \phi'$ which includes all the clauses of ϕ except exactly two, and which is satisfiable $\}$. Show that $ALLBUT2SAT$ is NP-complete.

To prove that $ALLBUT2SAT$ is NP-Complete, we must show that it is in NP and that it is NP-hard.

$\underline{ALLBUT2SAT \text{ is in } NP}$:

To prove that $ALLBUT2SAT$ is in NP, we will provide an NTM N deciding it. It follows the following algorithm on input $< \phi >$:

1. Check that ϕ is in CNF.
   (a) If not, reject.
   (b) Else continue.
2. Nondeterministically pick a truth assignment for the m Boolean variables in ϕ
3. Evaluate ϕ at the selected assignment.
   (a) If all but at most two clauses are satisfied accept.
   (b) Else, reject.

We now prove that $L(N) = ALLBUT2SAT$.

$L(N) \subseteq ALLBUT2SAT$: Assume $< \phi > \in L(N)$. This means that ϕ is a boolean formula in CNF form and that there exists a truth assignment that satisfies all but at most two of its clauses. This is exactly saying that $\exists \phi'$ which includes all the clauses of ϕ except at exactly two and which is satisfiable. Therefore, $< \phi > \in ALLBUT2SAT$.

$ALLBUT2SAT \subseteq L(N)$: Assume $< \phi > \in ALLBUT2SAT$. This means that $\exists \phi'$ which includes all the clauses of ϕ except at exactly two and which is satisfiable. This is saying that there exists a truth assignment that satisfies all but at most two of ϕs clauses. Therefore, there is a non-deterministic branch of N that accepts $< \phi >$, so $\phi \in L(N)$.

Moreover, each branch of N takes polynomial time in the length of the input:

- Step 1 takes $O(n)$ time
- Step 2 takes $O(m)$ time
- Step 3 takes $O(n)$ time

Therefore, each branch of N requires at more linear time.

<u>$ALLBUT2SAT$ is NP-hard:</u>

To prove that $ALLBUT2SAT$ is NP-Hard, we will show that there exists f, a polynomial time reduction from $ALLBUT2SAT$ to $CNFSAT$, which is also NP-Hard. This means that if we can provide a polynomial time algorithm to decide $ALLBUT2SAT$, then we can provide a polynomial time algorithm to decide $CNFSAT$.

We define $f : ALLBUT2SAT \to CNFSAT$ such that

$$f(< \phi >) =< \phi \wedge y \wedge \neg y \wedge z \wedge \neg z >$$

where y and z are boolean variables that do not appear in ϕ.

We now need to show that $< \phi > \in CNFSAT \iff f(< \phi >) \in ALLBUT2SAT$ and that f is a polynomial time reduction.

$< \phi > \in CNFSAT \Rightarrow f(< \phi >) \in ALLBUT2SAT$: If ϕ is satisfiable, then there is a truth assignment A, mapping each variable to a truth value, that can make all the clauses of ϕ true at the same time. Then $A \cup \{y \mapsto T\} \cup \{z \mapsto$

$T\}$ is an assignment that makes all the clauses of $f(<\phi>) =<\phi \wedge y \wedge \neg y \wedge z \wedge \neg z>$ except $\neg y$ and $\neg z$ true. Therefore, $f(<\phi>) \in ALLBUT2SAT$.

$f(<\phi>) \in ALLBUT2SAT \Rightarrow <\phi> \in CNFSAT$: Suppose all but at most two clauses of $\phi \wedge y \wedge \neg y \wedge z \wedge \neg z$ are satisfiable, then we have two cases. In addition, suppose that at least one clause that is not satisfied is in ϕ. This means that there is a truth assignment satisfying at least three of $\{y, \neg y, z\neg z\}$, which is impossible. Therefore, it must be that ϕ has a satisfying truth assignment, so $<\phi> \in CNFSAT$.

Finally, we can show that f takes polynomial time: all f does is add four clauses to the encoding of the boolean formula, which takes constant time. Therefore, f is $O(1)$, which is polynomial.

Problem 2

Let A be a language such that $A \in$ NP.

1. Let B be a language such that $B \in$ NP. Prove that $AB = \{x \cdot y \mid x \in A \text{ and } y \in B\} \in$ NP, so that NP is closed under concatenation.
2. Prove that $A^* = \{x_1 x_2 \ldots x_n \mid n \geq 0 \text{ and } x_i \in A\} \in$ NP, so that NP is closed under the Kleene star operation.

Solution:

For any two languages in NP, L_1 and L_2, let M_1 and M_2 be the NTMs that decide them in polynomial time. Construct an NTM M' which decides $L_1 L_2$ in polynomial time.

$M' =$ "On input w:

1) For each partitioning of w into two substrings $w = w_1 w_2$:

2) Run M_1 on w_1, and run M_2 on w_2. If both accept, accept; otherwise continue with the next division.

3) If w is not accepted after you try all possible divisions, reject."

M' accepts w iff w can be expressed as $w_1 w_2$ s.t. M_1 accepts w_1 and M_2 accepts w_2. Stage 2 runs in poly time and is repeated for at most O(n) times, so the algorithm runs in polynomial time.

<u>NP closed under Kleene star:</u>

Let $A \in$ NP. Then there exists M_A a non-deterministic polynomial-time decider for A. To show that A^* is in NP, we will construct a machine that runs in nondeterministic polynomial time that decides A^*.

Non-deterministic decider: We will construct a decider M' for A^* in the following way:

If $w = \epsilon$, M' accepts w.

For any $k \leq |w|$, M' non-deterministically considers all the ways to split w into w_1, ..., w_K such that $w_1w_2...w_k = w$. If w_1, ..., $w_k \in A$, M_A accepts each w_1, ..., w_k then M' accepts w. Else it doesn't. There is a finite number of possible splits since w is finite and M_A is a decider, so M' necessarily halts with either an acceptance or a rejection if there is no acceptance.

$L(M') = A^*$ **and** M' **decides** A^*:

First, we show that $A^* \subset L(M')$. Let $w \in A^*$. Then there exist w_1, ..., $w_K \in A$ such that $w_1w_2...w_k = w$. Thus, in one of the non-deterministic branches, M' simulates M_A on each w_i, accepting all the strings, so M' accepts w.

Now, we show that $\overline{A^*} \subset \overline{L(M')}$. Let $w \in \overline{A^*}$, so $w \notin A^*$. There exist no w_1, ..., w_K with $w_1w_2...w_k = w$ such that w_1, ..., $w_K \in A$. Therefore, none of the branches of M' end in acceptance, so M' rejects w.

Runtime analysis: Since A is in NP, M_A takes non-deterministic polynomial time $O(n^k)$. Each non-deterministic branch runs M_A a finite amount of times, so the total takes $O(n^k) + ... + O(n^k) = O(n^k)$ nondeterministic polynomial runtime. Hence, the total non-deterministic execution time of M' is equivalent to $O(n^k)$.

For any language A in NP, let M be the NTM that decides A in polynomial time. Construct NTM M' which decides A^* in polynomial time.

M' = "On input w:

1) On input w, use the choice inputs to break w into substrings $w = w_1w_2 \dots w_k$. Assuming a choice input of 1 or 0 for each of the x symbols in w: if the choice input is 1, then break off a substring after the current input symbol. If the choice input is 0, do not break off a substring after the

current input symbol.

2) For each substring w_i: Run M on w_i. Reject if it rejects.

3) Accept."

M' accepts w iff w can be expressed as $w_1 w_2 \dots w_k$ such that M accepts each w_i, which is true iff $w \in A^*$. Steps 1 and 3 clearly run in polynomial time; step 2 loops at most n times, and each loop takes polynomial time, so the whole thing takes polynomial time.

Problem 3

Given a graph $G = (V, E)$, we say that subset $V' \subseteq V$ is an "*α-diverse k-clique*" in G if:

- $|V'| = k$,
- For all pairs $v_i, v_j \in V'$, $(v_i, v_j) \in E$ (i.e., V' is a k-clique).
- For all pairs $v_i, v_j \in V'$, v_i is connected to α vertices that are not connected to v_j (and vice versa).

Let $DCLIQUE = \{\langle G, k, \alpha \rangle | G \text{ is a graph with an } \alpha - diverse\ k - clique\}$.

Show that $DCLIQUE$ is NP-complete.

Solution:

First, we will show that $DCLIQUE$ is in NP. To do so, we will construct a verifier TM that runs in polynomial time. Let the certificate be c be a subset $S \subset V$ of vertices in G. Our algorithm will do the following:

1. On input w, check that w is an encoding $\langle \langle G, k, \alpha \rangle, S \rangle$, where $G = (V, E)$ is a graph, k is a non-negative integer, α is a non-negative integer, and S is a subset of vertices in V. If not, reject.
2. Check that $|S| = k$. If not, reject.
3. For each distinct $u, v \in S$, go through the adjacency matrix E and check that there is an edge between u and v. If there exists some u, v that doesn't have an edge between them, reject.
4. For each distinct $u, v \in S$, do the following subroutine:
   - Let $b = 0$.

- Loop through each vertex z in V. If there exists an edge between z and u but not an edge between z and v, increase b by 1.
- After looping through all the vertices in V, reject if $b < \alpha$.

5. If we have not rejected until here, accept.

First, we will prove that this runs in polynomial time. The format check can be done in polynomial time. There are at most k^2 pairs u, v. For each pair, we go through at most n^2 entries of the adjacency matrix. So, steps 1-3 take at most $k^2 \cdot n^2$ steps. In step 4, we loop through k^2 pairs u, v and for each pair, we loop through n vertices, and for each vertex, we go through at most n^2 entries of the adjacency matrix. So, steps 4-5 take at most $k^2 \cdot n \cdot n^2 = k^2 \cdot n^3$. So, overall, the TM takes at most $k^2 \cdot n^2 + k^2 \cdot n^3 = O(k^2 n^3)$ steps, which is polynomial. Suppose $w \in DCLIQUE$. Then, it is in the right format, and it must be that S is an α-diverse k-clique so step 3 passes. Step 4 checks that each u is connected to at least α vertices that v is not connected to, and since S is an α-diverse clique, we accept in step 5, by definition. Suppose $w \notin DCLIQUE$. Then, if it is the wrong format, then we reject in either steps 1 or 2. Then, S is a subset of k vertices. If S is not a k-clique, we reject in step 3. Suppose, heading for a contradiction, that S is a k-clique and we accept in step 5. Then, for each distinct $u, v \in S$, we counted $b \geq \alpha$ vertices that are connected to u but not v. This means that S is α-diverse, so $w \in DCLIQUE$, which is a contradiction. So, we reject in step 5, as desired.

Now, we will show that $DCLIQUE$ is NP-hard. We will do a reduction from $CLIQUE$. Let our computable function f be defined as follows:

1. On input w, check if $w = \langle G, k \rangle$, where $G = (V, E)$ is a graph, and k is a non-negative integer. If the format is invalid, output "hi". If it's the correct format, output $\langle G, k, 0 \rangle$.

It is clear that this function runs in polynomial time, since checking the format and adding the 0 can all be done in polynomial time. Suppose $w \in CLIQUE$. Then, $w = \langle G, k \rangle$, so it is in the correct format. Then, $f(w) = \langle G, k, 0 \rangle$. Since $w \in CLIQUE$, it must be that G has a k-clique. This clique is also 0-diverse, since for each u, v, we have that u is always connected to at least 0 other vertices that v is connected to. So, $f(w) \in DCLIQUE$. Suppose $w \notin CLIQUE$. If it's the wrong format, then $f(w)$ is "hi", which is also in the wrong format so $f(w) \notin DCLIQUE$. If it's

in the right format, then w does not have a k-clique. Suppose, heading for a contradiction, that $f(w) \in DCLIQUE$. This means that G has a 0-diverse k-clique. Note that any 0-diverse k-clique is a k-clique, so G has a k-clique, which is a contradiction. So, it must be that $f(w) \notin DCLIQUE$, completing the proof.