CSE 555 HW 5 SAMPLE SOLUTION

Question 1.
Show that if $L$ is PSPACE-complete, then $L$ is NP-hard. Show that the converse is not true.

If $L$ is PSPACE-complete, then for all $A \in \text{PSPACE}$, $A \leq_P L$. We know $\text{SAT} \in \text{PSPACE}$ from Sipser Example 8.3, so $\text{SAT} \leq_P L$. A language is NP-hard if every language in NP is polynomial time reducible to it. $\text{SAT}$ is NP-complete and $\text{SAT} \leq_P L$, so for all $B \in \text{NP}$, $B \leq_P \text{SAT} \leq_P L$, so $L$ is NP-hard.

To see that the converse is not true, first show that $A_{\text{NTM}}$ is NP-hard, as follows. (Our definition of $A_{\text{TM}}$ is for deterministic TMs. $A_{\text{NTM}}$ is the version of $A_{\text{TM}}$ for non-deterministic TMs.) Take an arbitrary $A \in \text{NP}$ and let $M$ be the NTM that decides $A$ in non-deterministic polynomial time. There is a polynomial time computable function $f : \Sigma^* \rightarrow \Sigma^*$ so that $w \in A \iff f(w) \in A_{\text{NTM}}$; specifically, $f(w) = \langle M, w \rangle$. Construct $F$, the TM that computes function $f$, from $M$ as follows:

$F =$ “On input $w$:

1. Write $\langle M, w \rangle$ to the tape and halt.”

Writing $\langle M \rangle$ to the tape takes constant time regardless of the length of $w$, so the function is computed in $O(|w|)$ time. Then, for all $A \in \text{NP}$, $A \leq_P A_{\text{NTM}}$ so $A_{\text{NTM}}$ is NP-hard.

But $A_{\text{NTM}}$ is undecidable. Because every deterministic TM is also an NTM, if $A_{\text{NTM}}$ were decidable, a decider for $A_{\text{NTM}}$ could be to decide $A_{\text{TM}}$. Therefore, $A_{\text{NTM}}$ cannot be in PSPACE, so $A_{\text{NTM}}$ is not PSPACE-complete.

Question 2.

Show that $\text{ALL}_{\text{NFA}} = \{\langle M \rangle : L(M)\Sigma^* \text{ for the NFA } M \text{ whose input alphabet is } \Sigma\}$ is in PSPACE.

Sipser provides an algorithm for $\text{ALL}_{\text{NFA}}$ that runs in nondeterministic space $O(n)$, so $\text{ALL}_{\text{NFA}} \in \text{NPSPACE}$, and NPSPACE = PSPACE by Savitch's theorem. Because $\text{ALL}_{\text{NFA}}$ is in coPSPACE, and the deterministic space complexity classes are closed under complement, $\text{ALL}_{\text{NFA}}$ is in PSPACE.

Question 3.

A string $w = w_1...w_n$ with $w_i \in \Sigma$ for $1 \leq i \leq n$ yields
(a) the string \( w = w_1, \ldots, w_i-1, a, w_{i+1} \ldots w_n \) with \( a \in \Sigma \) by a \textit{substitution}, and

(b) the string \( w = w_1, \ldots, w_{i-1}, w_{i+1} \ldots w_n \) by a \textit{deletion}.

(You can substitute or delete any single symbol \( w_i \).) Let \( L \subseteq \Sigma^* \) be an arbitrary language and suppose that \( w, w' \in \Sigma^* \). We say that there is a \textit{deletion-substitution chain} in \( L \) from \( w \) to \( w' \) if there is a finite integer \( \ell \) so that there exist strings \( z_0, \ldots, z_\ell \) with each \( z_i \in L \) for which \( z_0 = w, z_\ell = w' \), and \( z_i \) is obtained from \( z_{i-1} \) by a deletion or substitution for each \( 1 \leq i \leq \ell \).

Show that, when \( L \) is regular, \( DSC_L = \{ \langle w, w' \rangle : \exists \text{ a deletion-substitution chain from } w \text{ to } w' \in L \} \) is in PSPACE.

Since PSPACE = NPSPACE, showing that \( DSC_L \in \text{NPSPACE} \) is equivalent to showing that \( DSC_L \in \text{PSPACE} \).

We devise an NTM, \( N_L \), that decides \( DSC_L \). Because \( L \) is regular, there is a DFA \( D \) such that \( L(D) = L \); \( N_L \) can have \( D \) hardcoded into it in order to decide when \( w \in L \). Simulating \( D \) on \( w \) can be performed using a constant amount of space, because we only need to keep track of the state \( D \) is currently in. We need to compute and store \( \ell \) so we know when our algorithm can terminate (when no chain can exist). Notice that, in a deletion-substitution chain, no \( |z_i| \) can ever be longer than \( |w| \), but it may be required to perform several substitutions before a deletion can be performed. The maximum length of a possible chain is the number of all possible \( z_i \) such that \( |z_i| \leq |w| \). Then,

\[
\ell = \sum_{i=0}^{\lfloor |w|/|\Sigma| \rfloor} |\Sigma|^i = |\Sigma|^{|w|+1} - 1
\]

The magnitude of \( \ell \) is exponential in the size of \( w \), but if we store \( \ell \) in base \( \Sigma \), then the storage required is \( |w| + 1 \). We also store \( i \) in base \( \Sigma \), so the space complexity for this is also \( O(|w|) \). Then \( N_L \) follows:

\( N_L = \text{"On input } \langle w, w' \rangle: \)

1. If \( |w| < |w'| \), reject (there cannot be a substitution-deletion chain)
2. Simulate \( D \) on \( w \) and reject if \( D \) does.
3. Simulate \( D \) on \( w' \) and reject if \( D \) does.
4. Compute and store \( \ell \) on the tape (as explained above).
5. For \( 1 \leq i \leq \ell \):
   a. Nondeterministically select one symbol in \( w \) and nondeterministically delete it or substitute another symbol from \( \Sigma \).
   b. If \( w = w' \), accept.
   c. Simulate \( D \) on \( w \) and reject if \( D \) does.
(6) No chain with length \(1 \leq i \leq \ell\) has been found so reject."

If a chain exists with the specified properties, \(N_L\) will find it because there is a sequence of nondeterministic choices leading to acceptance. (The space for \(w\) is reused for each substitution or deletion, so this requires no additional space.) We only need \(O(1)\) space to store the current state of \(D\) during simulations, and \(O(|w|)\) space to store \(i\) and \(\ell\), so the algorithm has space complexity \(O(|w|) \in \text{NPSPACE} = \text{PSPACE}\).

**Question 4.**

We say that language \(L\) is in \(\text{coNP}\) if and only if language \(L\) is in \(\text{NP}\). By direct analogy with the definition of \(\text{NP}\)-completeness, precisely define \(\text{coNP}\)-completeness. (Use polynomial-time reducibility.)

A tautology is a logical formula that is always true. Show that \(\{\langle \psi \rangle : \psi\) is an unquantified boolean formula that is a tautology\}\) is \(\text{coNP}\)-complete.

A language \(L\) is \(\text{coNP}\)-complete if it satisfies two conditions:

1. \(L\) is in \(\text{coNP}\). (\(L\) is in \(\text{NP}\))
2. every language in \(\text{coNP}\) is polynomial time reducible to \(L\). (The complement of every language in \(\text{NP}\) is poly reduce to not \(L\))

Let \(T = \{\langle \psi \rangle : \psi\) is an unquantified boolean formula that is a tautology\}\). \(T \in \text{coNP} \iff T \in \text{NP}\). \(T\) is the language of strings that are either not unquantified Boolean formulas or there is some truth assignment to the variables of \(\psi\) such that the formula is false. To see that \(\overline{T} \in \text{NP}\), we provide a polynomial time verifier, \(V\). The certificate, \(c\), is the truth assignment that makes the formula false. Let \(n\) be the length of \(\langle \psi, c \rangle\) and \(\ell\) be the number of variables in \(\psi\). We use the standard convention of \(0 = \text{false}, 1 = \text{true}\).

\(V = \) “On input \(\langle \psi, c \rangle\):

1. If \(\psi\) has any quantifiers or \(\psi\) is not a Boolean formula, accept.
2. Scan the tape and replace the variables of \(\psi\) with the assignment \(c\). If the literal \(x\) appears in the formula, replace it with its truth value. If \(\overline{x}\) appears, replace it with the opposite truth value. If \(c\) is not a truth assignment to the variables of \(\psi\) (if \(|c| \neq \ell\) or it is improperly formatted), accept.
3. Repeat until there is either a single 0 or 1 on the tape:
   a. Evaluate the formula by collapsing binary logical operators \(\land, \lor\) based on their rules and the truth value to the left and right, respecting precedence dictated by parentheses, and shifting the rest of the formula left after collapsing.
4. If 1 is on the tape, reject. If 0 is on the tape, accept.

Step 1 is completed in \(O(n)\) steps. Step 2 can be completed by zigzagging across the tape. For each pass, we replace at least 1 variable, or accept, so step 2 takes \(O(\ell n)\) steps. During
step 3, we are guaranteed to collapse at least one truth value for each iteration, but may have to shift almost $n$ tape symbols left, so this step takes $O(\ell n)$ steps. Step 4 is completed in constant time, so the verifier runs in time $O(\ell n)$.

To show that $T$ is coNP-complete, we either need to show that every language in coNP is polynomial time reducible to $T$, or we need to show that another coNP-complete problem is polynomial time reducible to $T$. For the second approach, we need a first coNP-complete problem. Consider $\overline{SAT}$ and consider the reduction from $A$ to $SAT$ presented in the proof of the Cook-Levin theorem. Let $A$ be an arbitrary language in NP and let $\phi$ be the formula that simulates the NTM for $A$. We know that $w \in A \iff \phi \in SAT$. We also have that $w \not\in A \iff \phi \not\in SAT$, or $w \in \overline{A} \iff \phi \in \overline{SAT}$. As $A \in NP \iff A \in \text{coNP}$, we have that every coNP language is polynomial time reducible to $SAT$.

Now we show $\overline{SAT} \leq_P T$. Notice that if $\psi$ is a tautology, then its negation is always false (it is unsatisfiable). Then, $\psi \in \overline{SAT} \iff \overline{\psi} \in T$.

This is already a boolean formula. However, we can carry this further to rewrite the formula to avoid the global negation, if desired. We define a recursive routine, $\text{NEGATE}$, that performs negation of an unquantified Boolean formula to show that the reduction can be performed in polynomial time.

$\text{NEGATE} = \text{"On input } \psi:\text{"

1. If $\psi = \psi_a \lor \psi_b$, replace $\lor$ with $\land$ and call $\text{NEGATE}(\psi_a)$, $\text{NEGATE}(\psi_b)$.
2. If $\psi = \psi_a \land \psi_b$, replace $\land$ with $\lor$ and call $\text{NEGATE}(\psi_a)$, $\text{NEGATE}(\psi_b)$.
3. If $\psi$ is a literal, $x$, replace $x$ with $\overline{x}$ and return.
4. If $\psi$ is a negated literal, $\overline{x}$, replace $\overline{x}$ with $x$ and return.

To see that $\text{NEGATE}$ runs in polynomial time, let $k$ be the number of Boolean logical operators in $\psi$ and $n$ the length of $\psi$. Each time $\text{NEGATE}$ is called, it changes exactly one symbol, so it runs in $O(1)$ time. $\text{NEGATE}$ is called twice for each of the $k$ logical operators, so there are $2k < 2n = O(n)$ total calls, so this $\text{NEGATE}$ runs in polynomial time.

Then $\overline{SAT} \leq_P T$ and $T \in \text{coNP}$, so $T$ is coNP-complete.

**Question 5.**

A boolean formula is in disjunctive normal form (DNF) if it is the disjunction of clauses, where each clause is the conjunction of literals (variables or their negations). We consider totally quantified boolean formulas of the form $Q_1x_1...Q_mx_m[\psi]$ where $\psi$ is in disjunctive normal form. Show that the language of true TQBFs in DNF is PSPACE-complete.

Call this problem DNF-TQBF. DNF-TQBF $\in$ PSPACE because every instance of DNF-TQBF is an instance of TQBF. So modify the algorithm in Sipser theorem 8.9 by checking that the formula given is in DNF before the rest of the algorithm executes. This step checks
if (1) each clause is a conjunction of literals and (2) the entire formula is a disjunction of clauses and rejects if either condition is violated. The modified algorithm uses no more space than the original, and so DNF-TQBF ∈ PSPACE.

To see that DNF-TQBF is PSPACE-complete, we might try to reduce TQBF to DNF-TQBF by converting any arbitrary $\phi$ to $\phi'$ DNF form. However, this can cause $\phi'$ to be exponentially larger, and we cannot run in less time than the amount of space we must use, so this may take exponential time. (For example, if $\phi$ is in 3-CNF with $c$ clauses, then $\phi'$ will have $3^c$ terms, as there are 3 ways to pick the literal to satisfy each of the $c$ clauses.)

Instead we follow the proof of Theorem 8.9 and make the necessary revisions. That proof constructs a quantified formula $\phi$ in which the unquantified boolean formula $\psi$ is a conjunction of boolean formulas arising exactly as in the proof of the Cook-Levin theorem (Theorem 7.37 in Sipser). The argument in Corollary 7.42 of Sipser shows that we can assume that $\psi$ is in conjunctive normal form. Most of the quantifiers in $\phi$ concern only individual variables, but as given in Theorem 8.9 the formula is not in prenex normal form because of the appearance of (“weird”) quantifiers like $\forall(c_3, c_4) \in \{(c_1, m_1), (m_1, c_2)\}$. Recall that each of $c_1, c_2, c_3, c_4, m_1$ represent $O(n^k)$ boolean variables. Let’s get rid of these weird quantifiers. For each weird quantifier, make a new boolean variable $s$. When $s$ is true, we make $O(n^k)$ clauses in CNF that require that if $s$ is true then $(c_3, c_4) = (c_1, m_1)$. To do this, for each boolean variable $a$ from $c_1$ and the corresponding boolean variable $b$ from $c_3$, form clauses $(s \lor a \lor b) \land (\overline{s} \lor \overline{a} \lor b)$. Do the same for the boolean variables from $c_4$ and $m_1$. Now repeat this process using $s$ in place of $\overline{s}$ with the boolean variables from $c_3$ and $m_1$, and from $c_4$ and $c_2$. Finally replace the weird quantifier with $\exists c_3 \exists c_4 \forall s$, and add all of the clauses so produced as conjunctions to $\psi$. This eliminates a weird quantifier while keeping $\psi$ in CNF, and adding $O(n^k)$ clauses to $\psi$. Repeat this on the $O(n^k)$ weird quantifiers to get a formula $\theta$ in prenex normal form in which $\psi$ is in CNF, so that $\theta$ and $\phi$ are logically equivalent.

Consider $\overline{\theta}$, the negation of $\theta$. Changing every $\forall$ to exists and every $\exists$ to $\forall$ in $\theta$, and complementing $\psi$ (using NEGATE above), we obtain the formula for $\overline{\theta}$. Now $\overline{\psi}$ is in disjunctive normal form because $\psi$ is in conjunctive normal form. Having earlier eliminated the weird quantifiers, $\overline{\theta}$ is in prenex normal form with a DNF boolean formula.

Now $\overline{\theta} \in$ DNF-TQBF if and only if $\phi \not\in$ TQBF. Hence DNF-TQBF is PSPACE-complete. (Technically, we should conclude that it is coPSPACE-complete, but these classes coincide.)