Solutions to Problem Set 7

1. Let A and B be two languages. Then show that:

   (a) If A and B are in NP, then so are $A \cup B$ and $A \cap B$.

   (b) If A and B are NP-complete, then $A \cup B$ and $A \cap B$ *need not be* NP-complete.

   [10 + 15 = 25 points]

   SOLUTION:

   (a) If A is in NP, then there is a deterministic Turing machine (verifier) V_A such that $x \in A$ if and only if $\exists y\ |y| \leq p(|x|)$ and V_A accepts $\langle x, y\rangle$ (see Sipser, page 265-266). Similarly, we have a machine V_B for B.

   Then for the language $A \cup B$, we define the machine $V_{A\cup B}$, which runs both V_A and V_B on the given input and accepts if either does. For $x \in A \cup B$, there is a string y_A such that V_A accepts $\langle x, y_A\rangle$ or a string V_B accepts $\langle x, y_B\rangle$. Taking y to be y_A or y_B (whichever exists), $V_{A\cup B}$ will accept $\langle x, y\rangle$. Similarly, for $A \cap B$, we can define $V_{A\cap B}$, which takes an input $\langle x, y_A, y_B\rangle$ accepts if and only if V_A accepts $\langle x, y_A\rangle$ and V_B accepts $\langle x, y_B\rangle$.

   (b) We argue about intersection first. Let L be any NP-complete language. Then we define the languages

$$A = 0L = \{0x \mid x \in L\}$$
$$B = 1L = \{1x \mid x \in L\}$$

   Then we can see that both A and B are NP-complete. This is so because any reduction from (say) SAT to L can be converted to a reduction to A by adding a 0 to the output and similarly for B. It is also easy to see that they are both in NP if L is. But then $A \cap B = \emptyset$ which cannot be NP-complete.

   One can derive the argument for union by exactly the same reasoning by noticing that if A and B are NP-complete, then $\overline{A}$ and $\overline{B}$ are co-NP complete and showing that $A \cup B$ is not NP-complete is the same as showing that $\overline{A} \cap \overline{B}$ is not co-NP complete. Thus, for an NP-complete language L, we can take $\overline{A} = 0\overline{L}$ and $\overline{B} = 1\overline{L}$. This gives

$$A = \overline{0\overline{L}} = (1\{0,1\}^*) \cup \{0x \mid x \in L\}$$
$$B = \overline{1\overline{L}} = (0\{0,1\}^*) \cup \{1x \mid x \in L\}$$

   Also, note that reductions to L can be easily modified to reductions to reductions to A and B, by appending 0 and 1 respectively at the beginning. Thus, A and B are NP-complete. However, $A \cup B = \{0,1\}^*$, which cannot be NP-complete.

2. Let $U = \{\langle M, x, \#^t\rangle | NDTM\ M$ accepts input x within t steps on at least one branch$\}$. Show that U is NP-complete.
   [15 points]

Solution: Given any NP language L, we have an NDTM M_L such that $\forall x \in L$, M_L accepts x on at least one branch in at most $p_L(|x|)$ steps, where $p_L()$ is a fixed polynomial depending on the machine. Also, M_L does not accept any $x \notin L$. Then, given x, we create $y = \langle M_L, x, \#^{p_L(|x|)} \rangle$ in polynomial time. By the previous argument, $x \in L$ iff $y \in U$. Thus, U is NP-hard.

To show that U is also in NP, we can create an NDTM M_U, which given an input $u = \langle M, x, \#^t \rangle$, simulates M on x for t steps. M_U nondeterministically guesses all the branches of M and accepts u iff M accepts u. Since the input has length at least t and we simulate M for at most t steps, the running time is polynomial in the length of the input (note this is the reason we need t in unary). It is easy to see that M_U accepts exactly the language U, thus proving $U \in NP$. Hence, U is NP-complete.

3. For a function $g : \mathbb{N} \to \mathbb{N}$, we say a language L is in $\mathbf{SIZE}(g(n))$ if there exists a family of circuits $C_1, C_2, \ldots$ (with C_i having i inputs and one output) such that:

   - $\forall n \in \mathbb{N}$ the size of C_n is at most $g(n)$
   - $\forall x \in \{0,1\}^n$ $x \in L \Leftrightarrow C_n(x) = 1$.

   In the class we saw a proof that $\mathbf{SIZE}(2^{o(n)}) \subsetneq \mathbf{SIZE}(2^n)$ i.e. for every large enough n there exists a function $f : \{0,1\}^n \to \{0,1\}$ that is not computable by circuits of size $2^{o(n)}$. This problem asks you to show such a "separation result" for a smaller function. Show that $\mathbf{SIZE}(n^3/100 \log n) \subsetneq \mathbf{SIZE}(n^3)$.
   [20 points]

   Solution: We saw is class that any circuit of size S can be described by $4S \log(2S)$ bits. Hence, any circuit of size $n^3/100 \log n$ can be described by $4 \cdot \frac{n^3}{100 \log n} \cdot \log\left(\frac{n^3}{100 \log n}\right) < 12n^3/100$ bits. Thus, the number of functions in $\mathbf{SIZE}(n^3/100 \log n)$ is at most $2^{12n^3/100}$.

   However, we also know that any function on k bits can be computed by circuits of size at most $3 \cdot 2^k - 4$. We then consider the set B of all the functions which only look at the first $\log(n^3/5)$ bits of the input. There are $2^{n^3/5}$ such functions. Hence, $\mathbf{SIZE}(n^3/100 \log n) \subsetneq B$, since $2^{n^3/5} > 2^{12n^3/100}$. But all these functions can be computed by circuits of size at most $3 \cdot 2^{\log(n^3/5)} - 4 \leq 3n^3/5 < n^3$. Hence $B \subset \mathbf{SIZE}(n^3)$. Thus, we have

$$\mathbf{SIZE}(n^3/100 \log n) \subsetneq B \subset \mathbf{SIZE}(n^3) \Rightarrow \mathbf{SIZE}(n^3/100 \log n) \subsetneq \mathbf{SIZE}(n^3)$$