Homework 9

Problem 1. Prove that TQBF restricted to formulas where the part following the quantifiers is in conjunctive normal form (CNF) is still PSPACE-complete.

Solution. Since all such restricted TQBF formulas are still in TQBF, and since TQBF is in PSPACE, the restricted TQBF is also in PSPACE. We will prove below that they are PSPACE-complete. Since TQBF is PSPACE-complete, we only need to prove that TQBF can be reduced to restricted TQBF in polynomial time.

For any TQBF formula (quantifiers)$\phi(x)$, we can use the Tseytin Transformation to transform $\phi(x)$ to CNF. We construct a new variable y_i for all subformulas of $\phi(x)$, such as: $y_1 \leftrightarrow p \wedge q$ or $y_2 \leftrightarrow y_3 \vee q$. Since all subformulas are represented by y_i, the definition of each y_i can be expressed with only one logic operator. We use $y_i \leftrightarrow ...$ to represent this. Say $\phi(x)$ is represented by y_k.

Thus, we can construct an equivalent formula $\varphi(x,y) = y_k \wedge (y_k \leftrightarrow ...) \wedge (y_{k-1} \leftrightarrow ...) \wedge ... \wedge (y_1 \leftrightarrow ...)$. For any x, $\phi(x)$ is true if and only if y_k is true and all y_is are defined correctly, which means when y_is are chosen correctly (by above definition), $\varphi(x,y)$ is satisfyable. $\varphi(x,y)$ can be transformed to CNF by transforming each substitutions to CNF, using the fact that $a \leftrightarrow b \wedge c = (a \to b \wedge c) \wedge (a \leftarrow b \wedge c) = (\neg a \vee (b \wedge c)) \wedge (\neg b \vee \neg c \vee a) = (\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg b \vee \neg c \vee a)$ and $a \leftrightarrow b \vee c = (a \to b \vee c) \wedge (a \leftarrow b \vee c) = (\neg a \vee b \vee c) \wedge ((\neg b \wedge \neg c) \vee a) = (\neg a \vee b \vee c) \wedge (\neg b \vee a) \wedge (\neg c \vee a)$. Say the CNF of φ is ψ, then (quantifiers)$\phi(x) \iff$ (quantifiers)$\exists y_1 y_2 ... y_k \varphi(x,y) \iff$ (quantifiers)$\exists y_1 y_2 ... y_k \psi(x,y)$

The above transformation can be done in polynomial time, since k grows linearly with the length of ϕ and the transformation steps grows linearly with k. Thus any TQBF formula can be transformed in polynomial to a restricted TQBF formula, which means the latter is PSPACE-complete.

Problem 2. Let SUM $= \{\langle x, y, z\rangle \mid x, y, z > 0$ are binary integers satisfying $x + y = z\}$. Show that SUM $\in$ L.

Solution. Say $x = x_n x_{n-1} ... x_1, y = y_n y_{n-1} ... y_1, z = z_n z_{n-1} ... z_1$, where n is the length of the longest binary integer, adding zeros to the left of integers shorter than n.

For any input $\langle x, y, z\rangle$, we can check if z is the sum of x and y by checking if $z_i = x_i \oplus y_i \oplus c_{i-1}$ for $i = 1, 2, ..., n$, where $c_0 = 0$ and $c_i = (x_i + y_i + c_{i-1} \geq 2)$.

If for all i the equation holds, then $z = x + y$. We design a Turing machine having a counter counting i, and a register storing c_{i-1}. Each time, the pointer retrieves x_i, y_i, z_i using the counter i, and verify $z_i = x_i \oplus y_i \oplus c_{i-1}$. If it's untrue, reject and halt. Else, update the register $c_i = (x_i + y_i + c_{i-1} \geq 2)$ and the counter $i++$. If the counter hits $n+1$, accept and halt. The Turing machine only uses $O(\log n)$ space to store the counter and the register, since $i \leq n+1$ and c is a single bit, thus SUM $\in$ L.

Problem 3.

(a) An undirected graph is *bipartite* if its nodes may be divided into two sets so that all edges go from a node in one set to a node in the other set. Show that a graph is bipartite if and only if it doesn't contain a cycle that has an odd number of nodes.

(b) Let BIPARTITE $= \{\langle G \rangle \mid G$ is a bipartite graph$\}$. Prove that BIPARTITE is in NL.

Solution.

(a) If a graph is bipartite, then it can be divided into two sets A and B such that all edges go from a node in A to a node in B. If there is an odd cycle in it, say $v_1 v_2 ... v_{2n+1}$. If $v_i \in A$, then there must be $v_{i+1} \in B$. So we know that $v_{2k+1} (\forall k)$ is in the same part with v_1. Since v_1 and v_{2n+1} are connected, v_1 and v_{2n+1} are in the same set, which means the cycle is not bipartite. Thus, if a graph is bipartite, then it doesn't contain a cycle with an odd number of nodes.

If a graph has no odd cycle, say $G = \{v_1, v_2, ... v_n\}$. First we consider the case where G is a connected graph. We choose $v_1 \in A$. Each time, we find a node which is connected to a classified node and put it into the different set from its neighbour. Since G is connected, we can always classify all nodes. This algorithm enables that there is an even path between v_1 and v_i for $v_i \in A$, and an odd path between v_1 and v_i for $v_i \in B$. If some node has neighbour in the same class, say $v_i, v_j \in A$, then there is a cycle $v_i v_1 v_j$. Since v_i and v_j are both connected to v_1 with a path with even length, say $2x, 2y$, this cycle is an odd cycle with length $2x + 2y + 1$, which raises contradiction.

So, a graph is bipartite if and only if it doesn't contain any odd cycle.

(b) We will prove that BIPARTITE $\in$ coNL, and since coNL = NL, BIPARTITE $\in$ NL. We can use an non-deterministic TM to solve the co-problem of BIPARTITE by guessing odd cycles. For graph G with n nodes, the NTM guesses an initial node first, and each time guesses a node that is connected to the previous node. We keep a counter to count the length of the cycle, and to prevent the NTM's guesses going into an infinite loop. If the NTM's guesses goes back to the initial node and the counter is odd, then the NTM accepts. If the NTM's guesses goes back to the initial node and the counter is even, or the counter hits $n + 1$ (which means the guess cycle enters a loop), halt. If there is an odd cycle in G, there is always one odd cycle with length $\leq n$, as larger cycles must have repeated nodes and can be decomposed to smaller cycles, and composition of even cycles are still even. The NTM accepts G if and only if there is an odd cycle with size $\leq n$ in G, which is equivalent to the existence of an odd cycle at any size in G, which means G is not bipartite. This NTM takes $O(\log n)$ space, since the counter is at most $n + 1$ and the current node is a single node. Thus BIPARTITE $\in$ coNL. Since coNL $=$ NL, BIPARTITE $\in$ NL.

Problem 4. Let $S(n) \geq \log n$ be a space-constructible function. Show that NSPACE$(S(n))$ $=$ coNSPACE$(S(n))$ is a consequence of NL $=$ coNL.

Solution. Suppose $A \in$ NSPACE$(S(n))$, then there is an NTM M deciding A in $O(S(n))$ space. For any x with length n, $x \in A \iff$ there is a path from the initial state to the accept state in the configuration graph of M with input x. Denote this graph as $G_{M,x}$. Suppose A has space complexity $S(n)$, and the writable tape of M is limited in $S(n)$ size, the number of possible configurations in G_{M_x} is $2^{O(S(n))}$. In fact, it's the multiplication of NTM state numbers (constant), possible tape patterns $(2^{S(n)})$ and pointer location $(n + S(n))$. As $S(n) \geq \log n$, $2^{S(n)} \times (n + S(n)) = 2^{S(n)+\log n} + 2^{S(n)+\log S(n)} \leq 2^{2S(n)} + 2^{2S(n)} = 2^{O(S(n))}$. Say $G_{M,x}$ has size $V = 2^{O(S(n))}$, the PATH problem on this graph is in NL.

If NL $=$ coNL, then in $O(\log V) = O(S(n))$ space we can verify whether there is no path from the initial state to the accept state in $G_{M,x}$, which means we can verify in $O(S(n))$ space that there exists no proof string supporting x, indicating $x \notin A$. Thus, $A \in$ coNSPACE$(S(n))$. Since A is arbitrary, NSPACE$(S(n)) \subseteq$ coNSPACE$(S(n))$.

Meanwhile, for any $A \in$ coNSPACE$(S(n))$, $\bar{A} \in$ NSPACE$(S(n))$. Since NSPACE$(S(n)) \subseteq$ coNSPACE$(S(n))$, we have $\bar{A} \in$ coNSPACE$(S(n))$, and $A \in$ NSPACE$(S(n))$. So we have NSPACE$(S(n))$ $=$ coNSPACE$(S(n))$ if NL $=$ coNL.