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.

> **Lemma.** (Tseytin Transformation) An example of the Tseytin Transformation is as follows. Denote n as the length of ψ. Let $\psi = ((p \vee q) \wedge r) \to (\neg s)$, then
>
> 1. Take all the subformulas of ψ:
>
> $$\begin{aligned} &\neg s, \\ &p \vee q, \\ &(p \vee q) \wedge r, \\ &((p \vee q) \wedge r) \to (\neg s). \end{aligned}$$
>
> Since the number of subformulas equals to the number of operators, so there the number of subformulas is $O(n)$.
>
> $$\begin{aligned} &x_1 \leftrightarrow \neg s, \\ &x_2 \leftrightarrow p \vee q, \\ &x_3 \leftrightarrow x_2 \wedge r, \\ &x_4 \leftrightarrow x_3 \to x_1. \end{aligned}$$
>
> As a result of step(1), the number of variables which we have introduced is $O(n)$.
>
> 2. Conjunct all substitutions and the substitution for ψ: $\psi' = x_4 \wedge (x_4 \leftrightarrow x_3 \to x_1) \wedge (x_3 \leftrightarrow x_2 \wedge r) \wedge (x_2 \leftrightarrow p \vee q) \wedge (x_1 \leftrightarrow \neg s)$
> 3. All substitutions can be transformed into CNF. Note that each subterm, for instance, $x_2 \leftrightarrow p \vee q$, is a 3-CNF. So expand them and we will gain at most $2^3 n$ formulas, which is still $O(n)$.
>
> Therefore, the length of ψ' is linear to that of ψ. And it is clear that the Tseytin Transformation is polynomial-time computable.

Solution. Denote the problem as TQBF(CNF). Consider Turing machine S:

1. Suppose the input is $\langle\varphi\rangle$. If φ has no quantifiers, Then φ has no variables. It either "True" or "False." Output accordingly.
2. If $\varphi = \exists x\ \psi$, then evaluate ψ with x equals true or false recursively. Accept if either case accepts. Reject otherwise.
3. If $\varphi = \forall x\ \psi$, then evaluate ψ with x equals true or false recursively. Accept if both case accepts. Reject otherwise.

Each recursive level uses constant extra space, so that the space complexity of S is $O(n)$. So TQBF(CNF) $\in$ PSPACE.

Now we will prove that TQBF(CNF) is PSPACE-complete by reducing TQBF to TQBF(CNF). Given $\varphi = Q_1x_1Q_2x_2\cdots Q_kx_k\ \psi \in$ TQBF, then using the Tseytin Transformation, we can transform ψ into ψ' in polynomial time, and the length of ψ' is linear to that of ψ.

Therefore, TQBF(CNF) 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. It is not possible to calculate $x + y$ directly because we only can use an additional $O(\log n)$ space. However, we can construct an algorithm as follows. We use the pair $\langle p, c\rangle$ to describe the current state, where p is the position of the bit we're calculating, and $c \in \{0, 1\}$ is the carry of the result that we have already calculated.

We can calculate the result from the lowest bit to the highest, bit by bit, increment p by one after every calculation of each bit.

Since p will not exceed the length of input n, so $p \le n$, and the space complexity for p is $O(\log n)$. Additionally, c only needs $O(1)$ space to store. Therefore, 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)

Suppose that G is bipartite, and let V_1 and V_2 be the two sets of nodes such that all edges go between V_1 to V_2. Each edge permits us to travel from a node in V_1 to a node in V_2 and vice versa. Thus, if we start at a node in V_1 and follow an edge, we must arrive at a node in V_2. If we follow another edge, we must arrive at a node in V_1. We can continue this process. If it forms a cycle, then the number of nodes in the cycle must be even since we alternate between nodes in V_1 and V_2.

Suppose G doesn't contain a cycle with an odd number of nodes. We can start at any node and assign it to V_1. Then, we assign all neighbors of nodes in V_1 to V_2, and all neighbors of nodes in V_2 to V_1. We continue this process until no more nodes can be assigned. Since all cycles have an even number of nodes, all nodes in the cycle will alternate between V_1 and V_2, and we won't have any conflicts. Thus, G is bipartite.

Therefore, a graph is bipartite if and only if it doesn't contain a cycle that has an odd number of nodes.

(b)

Suppose $G = \langle V, E\rangle$. Construct a nondeterministic Turing machine M as follows. Denote the current state as $\langle v, l\rangle$, where v is the current vertex, l is the length of the current path. For each vertex $v_{\text{init}} \in V$:

1. Start from v_{init}, and $\langle v, l\rangle \leftarrow \langle v_{\text{init},1}\rangle$.
2. (a) Nondeterministically guess the next vertex u.
   (b) If edge $(v, u) \in E$:
      - If $u = v_{\text{init}}$,
         - If l is odd, return $\langle G\rangle \notin$ BIPARTITE.
         - If l is even, return $\langle G\rangle \in$ BIPARTITE.
      - Otherwise, $v \leftarrow u, l \leftarrow l + 1$.

   (c) If $l \geq |V|$, guess another vertex u.
3. Return $\langle G\rangle \in$ BIPARTITE.

Since the length of the longest simple cycle in graph G is less than $|V|$, so $l \leq |V|$. Therefore, the cost of space complexity for storing $\langle v, l\rangle$ is $O(\log n)$.

Hence, BIPARTITE is in NL.

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

Solution. Suppose that $\text{NL} = \text{coNL}$. Consider a problem $Q \in \text{NSPACE}(S(n))$ with an alphabet Σ. Since $S(n) \geq \log n$, so there exists $m \in \mathbb{N}$ such that $S(n) = \log m$. Pad the input $\langle I \rangle$ to $\langle I' \rangle = \langle I\ c^{m-n} \rangle$, where character $c \notin \Sigma$, so $|\langle I' \rangle| = m$.

Suppose the corresponding nondeterministic Turing machine for Q is M. Construct a new nondeterministic Turing machine M' as follows:

1. Parse the input as $\langle I' \rangle = \langle I\ c^k \rangle$.
2. Use M to solve $\langle I \rangle$. Return the result.

We have constructed a new nondeterministic Turing machine M' that solves the problem Q', where Q' corresponds to M' and $Q' \in \text{coNL}$. Now we will construct a new machine M'' that solves Q and belongs to $\text{coNSPACE}(S(n))$.

The machine M'' operates as follows:

1. On input $\langle I \rangle$, where $|\langle I \rangle| = n$:
2. Pad the input to $\langle I' \rangle = \langle I\ c^{m-n} \rangle$, as before.
3. Run M' on input $\langle I' \rangle$ to obtain the result R.
4. If R is "accept," then accept; otherwise, reject.

The machine M'' simulates M' on the padded input $\langle I' \rangle$ and returns the same result. Since M' uses $\log m = S(n)$ space, M'' also uses $\log m = S(n)$ space. Therefore, M'' solves the problem Q using $\log m = S(n)$ space, which means $Q \in \text{coNSPACE}(S(n))$. This leads to $\text{NSPACE}(S(n)) \subseteq \text{coNSPACE}(S(n))$. Reverse the procedure above, then similarly, we can prove thate $\text{coNSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.

Hence $\text{NSPACE}(S(n)) = \text{coNSPACE}(S(n))$.