Theory of Computation

Solutions to Homework 2

Problem 1. Derive a disjunctive normal form of

$$(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \cdots \wedge (x_n \vee y_n).$$

Proof. We prove the following:

$$(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \cdots \wedge (x_n \vee y_n) = \bigvee_{(a_1,a_2,\ldots,a_n)\in\{x,y\}^n} \left(\bigwedge_{i=1}^{n} (a_i)_i \right)$$

by induction. For $n = 1$ the equation is trivial. Assume for $n = k$ the equation holds. We have:

$$\begin{aligned}
& (x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \cdots \wedge (x_{k+1} \vee y_{k+1}) \\
= & \bigvee_{(a_1,a_2,\ldots,a_k)\in\{x,y\}^k} \left(\bigwedge_{i=1}^{k} (a_i)_i \right) \wedge (x_{k+1} \vee y_{k+1}) \\
= & \bigvee_{(a_1,a_2,\ldots,a_k)\in\{x,y\}^k} \bigvee_{a_{k+1}\in\{x,y\}} \left(\bigwedge_{i=1}^{k} (a_i)_i \bigwedge (a_{k+1})_{k+1} \right) \\
= & \bigvee_{(a_1,a_2,\ldots,a_{k+1})\in\{x,y\}^{k+1}} \left(\bigwedge_{i=1}^{k+1} (a_i)_i \right)
\end{aligned}$$

The equation whose right-hand side is a desired DNF is established by mathematical induction. □

Problem 2. Prove that $\mathbf{NP} \neq \mathbf{SPACE}(n)$.

(Hint: You don't need to show $\mathbf{NP} \subsetneq \mathbf{SPACE}(n)$ or $\mathbf{SPACE}(n) \subsetneq \mathbf{NP}$ since they are open questions so far as we know. All you need to do is to prove these two sets are **unequal**.

A log-space reduction from language L_1 to language L_2 is a function R which can be computed by a deterministic log-space Turing machine such that $x \in L_1$ iff $R(x) \in L_2$. In the proof, you can treat log space and polynomial time interchangeably. So as long as your reduction R runs in polynomial time, it is fine.

A complexity class $\mathbf{C}$ is closed under log-space reduction if for any log-space reduction R from L_1 to L_2, $L_1 \in \mathbf{C}$ if $L_2 \in \mathbf{C}$. Show first that

NP is closed under log-space reduction. Then show that **SPACE**(n) is not closed under log-space reduction by the Space Hierarchy Theorem (the version in the textbook is sufficient). For this, suppose $L_1 \in \mathbf{SPACE}(n^2)$ but $L_1 \notin \mathbf{SPACE}(n)$. Now ask yourself what is the space complexity of deciding "$x \in L_2$?", where L_2 consists of those strings $x \in L_1$ padded with $n^2 - n$ redundant symbols after x with $|x| = n$.)

Proof. For any log-space reduction R from L_1 to L_2, which is in **NP**, we may execute the log-space Turing machine R and a nondeterministic polynomial-time Turing machine that decides L_2. The execution time of the first part is a polynomial of the input length since $\mathbf{SPACE}(\log n) \subset P$ and its output length is also bounded by the same polynomial. Hence the execution time of the second part is bounded by the composition of two polynomials, which is in turn a polynomial of the original input length. Therefore $L_1 \in \mathbf{NP}$, and **NP** is closed under log-space reduction.

We proceed to show that **SPACE**(n) is not closed under log-space reduction. For any language $L_1 \in \mathbf{SPACE}(n^2)$, we define a new language L_2 as follows. For any $x \in L_1$ whose length is n, $x\overbrace{\$\cdots\$}^{n^2-n}$ where \$ is a symbol outside the alphabet of L_1. Let R be a Turing machine (a reduction) which pads $n^2 - n$ \$s after the input, whose length is n. Clearly, R runs in polynomial time (log space). Modify the original quadratic-space Turing machine for L_1 to ignore \$. The new Turing machine checks (1) the length of the input string x is a square number, say n^2, (2) the first n symbols are from the alphabet of L_1 and (3) the following $n^2 - n$ symbols are all \$s. Then the new Turing machine simulates the original Turing machine on the first n symbols. This new Turing machine is a linear-space Turing machine because the counting-and-checking phase only requires $O(\log |x|)$ space and the simulation phase requires $O(n^2) = O(|x|)$ space. (Note that x is the padded string, not the original input string anymore). Hence, for the language L_2 decided by this new Turing machine, we have $L_2 \in \mathbf{SPACE}(n)$. Now pick a language $L_1 \in \mathbf{SPACE}(n^2)$ but $L_1 \notin \mathbf{SPACE}(n)$. By the padding argument, there exists a language $L_2 \in \mathbf{SPACE}(n)$ and a log-space reduction R from L_1 to L_2. Thus **SPACE**(n) is not closed under log-space reduction.

□