CS 278 - HW1

Joshua Turcotti

February 27, 2021

1 Space Hierarchy Theorem

Assume $f(n) = O(g(n))$. Let $h(n) = \sqrt{f(n)g(n)}$ such that $f = o(h)$ and $h = o(g)$. Let $\mathcal{U}$ be the universal turing machine given in the problem statment. Let L be exactly the language consisting of those $x \in \{0,1\}^*$ such that $\mathcal{U}$ can run the TM M_x on input x and terminate with $h(n)$ space and output 0. We claim that $L \in \mathbf{SPACE}[g] - \mathbf{SPACE}[f]$. Since $h = O(g)$, $L \in \mathbf{SPACE}[g]$ is trivial. For the sake of contradiction, assume $L \in \mathbf{SPACE}[f]$. Now there exists some TM M that can determine L in space f. Further, we know that $\mathcal{U}$ can simulate M with space Cs on all inputs x for which D requires space s. Choose n_0 such that $n > n_0$ implies $h(n) > Cf(n)$. Now choose x of size greater than n_0 such that $M = M_x$. It must be the case that M runs and terminates on input x with space $f(x)$, so $\mathcal{U}$ can simulate the execution of M on x with space $Cf(x) < h(x)$, which implies that L contains x if and only if the language determined by M does not. But this is a contradiction, as M was defined to determine exactly the language L. Thus $L \notin \mathbf{SPACE}[f]$, and so our assertion $L \in \mathbf{SPACE}[g] - \mathbf{SPACE}[f]$ holds.

2 Log-Space Reductions

In a lemma that will be relevant to both parts a and b below, we note that any $O(\log(|x|))$-space deterministic turing machine with a single read-only input tape, the possible presence of a write-once output tape, and an arbitrary positive (but constant) number of read-write work tapes can be simulated by a $O(\log(|x|))$-space deterministic turing machine with a single read-write work tape. If the former machine machine has n read-write work tapes, then we can use the $(kn + l)$th cell in the single read-write work tape of the latter machine (for $l < n$) to simulate the kth cell of the lth tape of the former machine. If the former machine was bound to have $O(f(|x|))$ cells used per tape, the latter machine will still be bound to have $O(nf(|x|)) = O(f(|x|))$ cells used per tape. Now we proceed.

(a) We consider the Turing machine M that computes $f(x)$ from x using $O(\log |X|)$ space, writing $f(x)$ to a write-once output tape. We will construct M' that computes the ith bit of $f(x)$ using $O(\log |X|)$ space, using no output tape. Specifically, we note from above that we may give M a second read-write work tape as long as no more than $O(\log |X|)$ of its cells are used. This tape is initially set entirely to 0. M' simulates M exactly, except that whenever M would write a bit to the output tape, instead M' checks the current value of the entire second work tape and if it is equal to i, outputs that bit, and otherwise performs the binary addition algorithm on that entire tape to add 1 to its value. Since i is bounded above by $|x|^c$, this tape will never have more than $c \log |x|$ bits written to it, establishing that M' indeed only uses $O(\log |X|)$ space.

(b) We consider the Turing Machine M that rejects or accepts inputs x from a read-only input tape, using a read-write work tape with $O(\log |x|)$ cells, to determine the language $B \in \mathbf{L}$. Given a log-space computable reduction f from A to B, we will show $A \in \mathbf{L}$ by constructing a Turing Machine M' to deteremine A. Beginning with M, add two more read-write work tapes, t_1 and t_2, both initialized to 0. To run M' on the input x, simulate M, except that whenever M moves its input head to the right (resp. left), increment (resp. decrement) the binary number represented on t_1, and whenever M reads a bit from the input tape, run the procedure from part a on the input pair (x, contents of t_1) using the work tape t_2, and then procede as if the output from that procedure was the bit read from the input. In this way, M' will act exactly as M on the remaining work tape, and on its state when not subroutining the procedure for part a, and thus will accurately determine if $f(x) \in B$, and thus if $x \in A$, using $O(\log |x|)$ space. This establishes that $A \in \mathbf{L}$.

3 Immerman Szelepcsenyi's Theorem

(a) To show that $A' \in \mathbf{NL}$, we must show that there exists a Turing Machine M' such that $y \in A' \iff \exists z : M'(y,z) = 1$ and that runs with $O(\log |y|)$-space. Since we know $A \in \mathbf{NSPACE}[n]$, let M be the machine such that $x \in A \iff \exists z : M(x,z) = 1$ and that runs with $O(|x|)$ space. Let M' accept the pair (y,z) iff M accepts the first $\log(|y|)$ bits of y and all the remaining bits of y are 0. The first check takes space $O(\log |y|)$ and the second check takes space $O(1)$, so M' is an $O(\log |y|)$-space machine that (by the definition of A') determines A', proving $A' \in \mathbf{NL}$.

(b) Since $\mathbf{NL} = \mathbf{coNL}$, we already have $A' \in \mathbf{coNL}$. This gives us the existence of a Turing Machine N' such that $y \in A' \iff \forall z : N'(y,z) = 1$ that runs with $O(\log |y|)$ space. Define the Turing Machine N that accepts x iff N' accepts $(x, 0^{2^{|x|}-|x|})$. We note that by the definition of A', N determines exactly the language A, and runs with space $O(\log 2^{|x|}) = O(|x|)$, proving $A \in \mathbf{coNSPACE}[n]$. Since A was an arbitrary language in $\mathbf{NSPACE}[n]$, we have shown $\mathbf{NSPACE}[n] \subseteq \mathbf{coNSPACE}[n]$

(c) Let $A \in \textbf{coNSPACE}[n]$. Then $\bar{A} \in \textbf{NSPACE}[n]$, so by part b $\bar{A} \in \textbf{coNSPACE}[n]$. But then by the definition of $\textbf{coNSPACE}[n]$ as the set of complements of $\textbf{NSPACE}[n]$, $\bar{\bar{A}} = A \in \textbf{NSPACE}[n]$. This allows us to conclude $\textbf{coNSPACE}[n] \subseteq \textbf{NSPACE}[n]$, and thus $\textbf{coNSPACE}[n] = \textbf{NSPACE}[n]$.