Assignment 3 Solutions

April 11, 2002

1. [7.4.3 (a)] Assume that f and g are computed by TMs M_f and M_g respectively.

   $f(g)$: Simulate M_g, storing the "output" on a work tape, and then simulate M_f (using a different set of tapes), using that work tape as input.

   $f + g$: Simulate M_f, then simulate M_g. The outputs will be concatenated together, and so the output will be of length $f + g$.

   $f \cdot g$: Simulate M_f, storing the "output" on a work tape. Then, repeat the following until that work tape is empty: delete the last character from the work tape, and simulate M_g.

   2^g: In addition to the tapes uysed by the simulation of M_g, we will use 2 extra tapes, T_1 and T_2. Begin by writing a single character to T_1. Then, simulate M_g, except, each time M_g tries to output a character, instead, call the following subroutine: Copy T_1 over T_2, then append T_2 to T_1. The result is that the length of T_1 is doubled each time M_g tries to output a character. Then, when the simulation of M_g terminates, simply copy T_1 to the output.

   It is clear (and easily checked) that all of these run in the required time and space bounds.

2. [7.4.3 (b) (ii), (iii), (vi), (viii)] $n \log n$: Both n and $\log n$ are proper complexity functions, and so by the result of part (a), so is $n \log n$.

   n^2: n is a proper complexity function, and so by the result of part (a), $n \cdot n = n^2$ is also a proper complexity function.

   2^{n^2}: n^2 is a proper complexity function (see above), and so by the result of part (a), so is 2^{n^2}.

   $n!$: To see that $n!$ is a proper complexity function, we will contruct a TM. First, copy the input to tapes 1 and 2. Delete the last character from tape 1, multiply tapes 1 and 2 (using the same method as was used in part (a), and storing the result in tape 3), and then copy tape 3 over tape 2. Repeat this process until tape 1 is empty. Then copy tape 2 (which will contain $n!$) to the output. This machine clearly uses the correct amount of time and space.

3. [7.4.4] $C = \{n^k : k > 0\}$: Closed under both left and right polynomial composition. Let $f \in C$, so $f = n^j$ for some j. Also, suppose $p = n^l$ for some constant l. Then $p(f) = f(p) = n^{jl} \in C$.

$C = \{k \cdot n \ : \ k > 0\}$: Not closed under either left or right polynomial composition. Let $f \in C$, so $f = j \cdot n$ for some j. Also, suppose $p = n^l$ for some constant l. Then $p(f) = (j \cdot n)^l = j^l \cdot n^l = O(n^l) \notin C$. Also, $f(p) = j \cdot n^l = O(n^l) \notin C$.

$C = \{k^n \ : \ k > 0\}$: Closed under left, but not right, polynomial composition. Let $f \in C$, so $f = j^n$ for some j. Also, suppose $p = n^l$ for some constant l. Then $p(f) = (j^n)^l = j^{nl} = (j^l)^n \in C$. However, $f(p) = j^{(n^l)} \notin C$.

$C = \{2^{n^k} \ : \ k > 0\}$: Closed under both left and right polynomial composition. Let $f \in C$, so $f = 2^{n^j}$ for some j. Also, suppose $p = n^l$ for some constant l. Then $p(f) = \left(2^{n^j}\right)^l = 2^{ln^j} = O(2^{n^{j+1}}) \in C$. Also, $f(p) = 2^{(n^l)^j} = 2^{n^l j} \in C$.

$C = \{\log^k n \ : \ k > 0\}$: Closed under both left and right polynomial composition. Let $f \in C$, so $f = \log^j n$ for some j. Also, suppose $p = n^l$ for some constant l. Then $p(f) = \left(\log^j n\right)^l = \log^{jl} n \in C$. Also $f(p) = \log^j(n^l) = l^j \log^j n \in O(\log^j n) \in C$.

$C = \{\log n \ : \ k > 0\}$: Closed under right, but not left, polynomial composition. Let $f \in C$, so $f = \log n$ for some j. Also, suppose $p = n^l$ for some constant l. Then $f(p) = \log n^l = l \log n \in O(\log n) \in C$. However, $p(f) = (\log n)^l = \log^l n \notin C$.

4. [7.4.5] To see that **P** and **NP** are closed under union and intersection, suppose that M_1 and M_2 are polynomial–time TMs for L_1 and L_2 respectively. Then $M_\cup$ can accept $L_1 \cup L_2$ as follows: On input x, $M_\cup$ simulates M_1 on input x, stores the result, simulates M_2 on input x, and accepts if and only if either M_1 or M_2 accepted.

   Simialrly, $M_\cap$ can accept $L_1 \cap L_2$ as follows: On input x, $M_\cap$ simulates M_1 on input x, stores the result, simulates M_2 on input x, and accepts if and only if both M_1 and M_2 accepted.

   Since polynomials are closed under addition, both $M_\cup$ and $M_\cap$ run in p[olynomial time, and will be deterministic if and only if M_1 and M_2 were both deterministic. Thus **P** and **NP** are closed under union and intersection.

5. [7.4.7] To see that **NP** is different from **SPACE**(n), first note that **NP** is closed under log-space reductions (see question #8 for proof of this).

   Thus, if **SPACE**(n), were the same as **NP**, it would also be closed under log–space reductions. We will show that this is not the case.

   By the Hierarchy Theorem, we know that **SPACE**(n), is a proper subset of **SPACE**(n^2). Choose $L \in$ **SPACE**(n^2) $-$ **SPACE**(n). Assume that the symbol "#" is not in the alphabet of L. We will show that $L \leq L'$ for some $L' \in$ **SPACE**(n).

   Define $L' = \{x\#^{|x|^2 - |x|} \ : \ x \in L$. Clearly, $L \leq L'$ since a log–space machine can simply append a string of $|x|^2 - |x|$ "#" symbols to the end of the input. This only requires a counter up to $|x|^2 - |x|$, which requires $2 \log |x|$ space.

   To see that $L' \in$ **SPACE**(n), simply note that since $L \in$ **SPACE**(n^2), a turing machine can first check to ensure that there are the correct number of "#" symbols at the end of the input (this only takes $2\log|x|$ space), and then erase these symbols (constant space), and then finally rewind to the beginning of the input tape and decide whether $x \in L$, using $O(|x|^2) = O(n)$ space. Thus, L' can be accepted in linear space and so **SPACE**(n) is not closed under log–space reductions, and so it is a different class than **NP**.

6. [7.4.8] First, modify Lemma 7.1 to state that $H_f \in \mathbf{TIME}(f(n)\log f(n))$. The only change in the proof of Lemma 7.1 is to replace the single–string simulator of multi–string machines (from Theorem 2.1) with a 2–string simularot of multi–string machines which runs in time $O(f(n)\log f(n))$, which is the result of problem 2.8.9.

   Next, modify Lemma 7.2 to state that $H_f \notin \mathbf{TIME}(f(n)/\log(f(n))$.

   Finally, for any proper function $f(n)$, we have shown that $\mathbf{TIME}(f(n)/\log(f(n))$ is a proper subset of $\mathbf{TIME}(f(n)\log f(n))$. Since proper functions are closed under multiplication and composition (and $\log(n)$ is a proper function) it follow that, if f is a proper function, then $\mathbf{TIME}(f(n))$ is a proper subset of $\mathbf{TIME}(f(n)\log^2(f(n)))$.

7. [7.4.17]

   (a) Replace each tape with a 3–track tape as follows: The first track is the same as the original tape. The 2nd track contains the original tape, shifted one block to the right, with each block reversed. The 3rd track contains the original tape shifted one block to the left, with each block reversed.

   The result is that block i of the 3–tape contains the contents of block i of the original tape, the reversed contents of block $i-1$, and the reversed contents of block $i+1$. To ensure that the TM is block–respecting, do not cross block boundaries unless forced to. That is, make use of the "reversed" blocks on the additional tape tracks whenever possible. Then, when a boundary must be crossed (because information from a block other than $i-1$, i, or $i+1$ is needed), the cursor should be moved to the "correct" block (waiting if necessary at block boundaries) Since the blocks are of size $\sqrt{f(|x|}$, this can occur no more than once every $\sqrt{f(|x|}$ steps, and takes no more than $3\sqrt{f(|x|}$ steps, and so the slowdown is only by a constant factor.

   (b) $G_M(x) = (V, E)$, with $V = \{1, 2, \ldots \sqrt{f(|x|)}\}$, and $(i, j) \in E$ if and only if $j = i+1$ or a tape block used in time–block j was *last* used block i. This graph is clearly acyclic since if $(i, j) \in E$, it must be that $i < j$.

   (c) Using the graph $G_M(x)$, our computation on x can use R "registers", where each register has enough space to store the contents of one tape block. Once we have computed all the successors of a node i, we no longer need to store the information for i.

   (d) Since G_M has $\sqrt{f(n)}$ nodes, it can be computed with $\sqrt{\frac{f(n)}{\log f(n)}}$ registers. So by "guessing" such a computation, we can compute $M(x)$ in nondeterministic space $\sqrt{\frac{f(n)}{\log f(n)}}$, or deterministic space $\frac{f(n)}{\log f(n)}$ (by Savitch's Theorem)

8. [8.4.3] First, recall (from the text) that log-space reductions must run in polynomial time. Thus, it follows that the size of the output of a log–space reduction can be at most polynomial in the size of the input, since a TM can only write one character per time unit.

   To see that **P**, **NP**, and **EXP** are closed under log–space reductions, suppose that L is in **P** (or **NP**, or **EXP**) and suppose that $L' \leq L$, by the reduction R (which runs in log space). A TM can accept L' as follows: On input x, compute $R(x)$ in log–space (and thus polynomial time) and then check to see if $R(x) \in L$.

   If $L \in \mathbf{P}$, then this takes time which is polynomial in $|R(x)|$, which is itself polynomial in $|x|$. Since polynomials are closed under composition, the time taken is polynomial in $|x|$, and so $L' \in \mathbf{P}$, as desired.

   If $L \in \mathbf{NP}$, the argument is identical to the one above: the machine is nondeterministic, but the time bound is the same.

   If $L \in \mathbf{EXP}$, the time taken is exponential, since exponential functions are closed under left polynomial composition (see question 3).

   Since **NP** is closed under reductions, so is **coNP**, since if $L' \leq L$ by a reduction R, then R also reduces $\overline{L'}$ to $\overline{L}$. This is because $x \in L'$ if and only if $R(x) \in L$, so it follows that $x \notin L'$ if and only if $R(x) \notin L$.

   To see that **L**, **NL**, and **PSPACE** are closed under log–space reductions, we will use a similar argument as we did for the time classes, but we must account for space instead of time.

   Once again, suppose that L is in **L** (or **NL**, or **PSPACE**) and suppose that $L' \leq L$, by the reduction R (which runs in log space). A TM can accept L' as follows: On input x, compute $R(x)$ in log–space (and thus polynomial time) and then check to see if $R(x) \in L$. Since $|R(x)|$ is polynomial in $|x|$, $|R(x)| = O(|x|^k)$ for some constant k.

   If $L \in \mathbf{L}$, then the space used is $O(\log |x|)$ to perform the reduction, and $\log |x|^k = k \log |x|$ to check if $R(x) \in L$. Thus the total space used is $O(log|x|)$, as desired.

   If $L \in \mathbf{NL}$, the argument is identical to the one above: the machine is nondeterministic, but the time bound is the same.

   If $L \in \mathbf{PSPACE}$, then the space used to check to see if $R(x) \in L$ is polynomial in the size of $R(x)$, which is polynomial in the size of x, since polynomials are closed under reductions.

   Thus, all these classes are closed under reductions.

   However, $\mathbf{TIME}(n^2)$, is not closed under reductions. To see this, choose $L \in \mathbf{TIME}(n^4) - \mathbf{TIME}(n^2)$, with "#" not appearing in the alphabet for L. Such an L exists, as a result of the hierarchy theorem. We will show that there exists some $L' \in \mathbf{TIME}(n^2)$, with $L \leq L'$.

   Define $L' = \{x\#^{|x|^2-|x|} \ : \ x \in L\}$. It is clear that $L \leq L'$, since a log–space TM can simply append a string of $|x|^2 - |x|$ "#" symbols to the end of its input. A TM can accept L' in quadratic time, as follows: First, it checks to make sure that there are the correct number of "#" symbols at the end of the input, and erases these symbols. This takes linear time. Then, it checks to make sure that x (what remains on the tape after the "#" symbols are removed) is in L. This time taken is $O(|x|^4) = O(n^2)$, as desired. Thus, $\mathbf{TIME}(n^2)$, is not closed under reductions.

9. [Bonus] Suppose $\mathbf{TIME}(n) = \mathbf{NTIME}(n)$, and suppose that f is a proper complexity function. We must show that $\mathbf{TIME}(f(n)) = \mathbf{NTIME}(f(n))$. It is sufficient to show that $\mathbf{NTIME}(f(n)) \subseteq \mathbf{TIME}(f(n))$, since the reverse inclusion is obvious.

   Suppose that $L \in \mathbf{NTIME}(f(n))$. Assume without loss of generality that "#" is not an element of L's alphabet. Define $L' = \{x\#^{f(|x|)-|x|} \ : \ x \in L\}$. Clearly $L' \in \mathbf{NTIME}(n)$. Thus, by our assumption, $L' \in \mathbf{TIME}(n)$. Let M' be the deterministic TM which accepts L' in linear time. Then define the deterministic TM M as follows: On input x, M first appends $f(|x|) - |x|$ "#" symbols to the end of the input. Since f is a proper complexity function, this can be done in $O(f(|x|))$ time. M then simulates M' on the resulting tape. This takes linear time (in terms of the tape length), but the tape has length $f(|x|)$, so the total time taken in $O(f(x))$. Thus, M is a deterministic TM which accepts L in time $f(x)$, and so $\mathbf{TIME}(n) = \mathbf{NTIME}(n)$, as desired.