Solutions to Midterm Problems

May 3, 2021

1 Problem 1

(10 points) Using the definition of Big Oh, \( f = O(g) \) iff there are positive integers \( c \) and \( n_0 \) such that for every \( x > n_0 \), \( f(x) \leq cg(x) \) prove or disprove the two statements below. Your proofs should be short and precise.

1. \( 10n^2 = O(n^2 + 3 \log_2 n) \)
   
   \textbf{proof:} We have that \( 10n^2 \leq 10(n^2 + 3 \log_2 n) \) so the definition is satisfied with \( n_0 = 1 \) and \( c = 12 \)

2. \( 2^{4n} = O(2^n) \). This is false.
   
   \textbf{proof:} Assume, by contradiction, that for some \( c > 0 \), \( 2^{4n} \leq c2^n \) for all sufficiently large \( n \). Consider the inequality for values \( n + u \) with integers \( u > 0 \). We should have \( 2^{4(n+u)} \leq c2^{n+u} \), which implies \( 2^{4u} < c2^u \), and therefore \( 23u \leq c \). This is clearly false since \( c \) is a constant and \( 2^{3u} \) grows unboundedly.

2 Problem 2

- line (1) correct
- line (2) incorrect. A string is in the union iff it is in at least one of the two sets
- line (3) incorrect. \( M_1 \) may not halt. This causes no problems if \( M_2 \) does not accept \( y \) either, but if it halts and accepts, \( y \) is in the union and will be rejected by \( M \)
- line (4) incorrect. In the case \( M_1 \) accepted \( y \), \( M_2 \) may not halt, making \( M \) reject a string in the union. As seen above, the statement may not be executed when it should be.
- line (5) incorrect. It is enough for one of the machines to accept.

\textbf{Comment.} Of course the union of two c.e. languages is c.e., but we have to avoid the pitfall of machines that may not halt. We use dove-tailing: do 1 step of \( M_1 \), then 1 step of \( M_2 \), then 1 more step of \( M_1 \), etc.
3 Problem 3

The required code is part of the code for a Turing machine that accepts the string $w\#w$ described in Chapter 3 of Sipser (page 167 and ff.)

4 Problem 4

4.1 (i)

Prove that the language $TWO = \{\langle M \rangle \mid M$ accepts at least two distinct inputs$\}$ is not decidable.

proof: We reduce $A_{TM} = \{\langle M, w \rangle \mid$ Turing machine $M$ accepts $w \}$ to $TWO$. Consider the string $\langle M, w \rangle$. Design a Turing machine $N$ that

- on input 0 accepts
- on any other input, simulates $M$ on input $w$

Now, if $M$ does not accept $w$, $N$ accepts exactly one input. Otherwise it accepts all inputs. So $\langle N \rangle \in TWO$ iff $\langle M, w \rangle \in A_{TM}$. Since $A_{TM}$ is not decidable, $TWO$ cannot be decidable.

4.2 (ii)

Prove that the language $NOT - TWO = \{\langle M \rangle \mid M$ accepts at most two distinct inputs$\}$ is not recognizable.

proof: There are several proof strategies. Given part (i) perhaps the easiest is to use an almost identical strategy. We will reduce the problem $NOT - A_{TM} = \{\langle M, w \rangle \mid M$ does not accept $w \}$ to $NOT - TWO$. This proves that $NOT - TWO$ is not enumerable since if a language $A$ is reducible to a language $B$, and $B$ is c.e. then $A$ is c.e.

It is OK to just claim the $NOT - A_{TM}$ is not recognizable. It is also easy to prove this: we know that $A_{TM}$ is recognizable. If $NOT - A_{TM}$ were also recognizable we would run in parallel the two recognizers (meaning: simulate one step of the first machine, then one step of the second). One of them must halt. This would imply that $A_{TM}$ is decidable, which is a contradiction.

The reduction is simply to map each string $\langle M, w \rangle$ into the Turing machine $N_{\text{string}}\langle M, w \rangle$ defined as follows

- on input 0, accept
- on input 1, accept
- on any other input, simulate $M$ on input $w$

It is clear that $N_{\text{string}}\langle M, w \rangle \in NOT - TWO$ iff $M$ does not accept $w$ iff $\langle M, w \rangle \in NOT - A_{TM}$. 
5

5.1 (i)

The simulation of nondeterministic Turing machines by deterministic ones is described in Sipser (for example in page 178). The possible computations can be represented as a tree, where the root corresponds to the initial configuration of the Turing machine on its input, and each vertex has children representing the results of possible nondeterministic moves by the machine. If the machine can make at most $cn$ moves on inputs of length $n$, the tree will have depth $cn$, and each vertex will have degree at most $d$ (the maximum number of possible successors of a state $\times$ input symbol pair in the definition of the Turing machine. So the number of vertices of this tree will be at most $2^{cdn}$. Using a breadth first search takes time linear in the size of the tree, so we can decide in time $2^{O(n)}$ whether the input was accepted.

5.1.1 (ii)

If $A \leq_{LIN} B$ then there is a linear time computable function $f$ such that for every string $x \in A$ iff $f(x) \in B$. Moreover, there is a constant $a0$ such that if $|x|$ the length of $x$ is $n$, then the string $y = f(x)$ has length at most $an$, because a linear time bounded Turing machine can write at most $an$ symbols (it can write at most one symbol per move.) Similarly, if $B \leq_{LIN} C$, then is a linear time computable function $g$ such that for every string $y \in B$ iff $g(y) \in C$. Moreover, there is a constant $b0$ such that if $|y|$ the length of $y$ is $m$, then the string $z = g(y)$ has length at most $bm$.

But then $x \in A$ iff $g(f(x)) \in C$. Moreover $g(f(n))$ can be computed in time $ban$ which is $O(n)$, so it is a deterministic linear time computable function. So $g(f())$ is a linear time reduction of $A$ to $C$, and so $A \leq_{LIN} C$.

5.2 (iii)

We have to prove that

1. If $C \in DLIN$ then $NLIN = DLIN$

2. If $NLIN = DLIN$ then $C \in DLIN$

We have seen this proof in the context of decidability, and also in NP-completeness.

The second statement is immediate: If $C$ is NLIN-complete, it is in $NLIN$, and if $NLIN = DLIN$ then it is in $DLIN$. (One could also argue that if $C$ not in $DLIN$, it is a witness to $DLIN \neq DLIN$)

To prove 1. consider an arbitrary language $L \in NLIN$. By the definition ofNLIN-completeness, there is a deterministic linear time computable function $f$ such that for every $x$, $x \in L$ iff $f(x) \in C$. In particular, if the length of $x$ is $n$, the length $m$ of $y = f(x)$ is bounded by $an$ (as in the proof of (ii) above.) Since $C \in DLIN$ there is a deterministic algorithm to decide membership of length $m$ strings in $C$ with running time bounded by $bn$. But then we can decide whether $x \in L$ by computing $f(x)$, then deciding whether $f(x) \in C$. The runtime of this process is bounded by $abn$ which is $O(n)$, so $L \in DLIN$. 
