CS 181: Homework 1

Einar Balan

1. Let S, T be two finite sets such that there is a one-to-one mapping from S to T and a one-to-one mapping from T to S. Show that $|S| = |T|$ (i.e., the two sets have the same number of elements). [1 point]

   Answer Towards a contradiction, assume $|S| \neq |T|$. This means that there must be at least one more element in S than T or vice versa.

   Define $F : S \to T$ and $G : T \to S$, which are both one to one.

   Because F is one to one, $\forall x \neq x' \in S \implies F(x) \neq F(x') \in T$. That is, for all distinct elements in S there is a unique mapping in T for that element. Consider the case where $|S| < |T|$. By the pidgeonhole principle, we know that there is at least one distinct x, x' pair s.t. their mappings are the same. Thus F cannot be one to one and we have reached a contradiction. Therefore $|S|$ is not $> |T|$.

   We repeat the same logic for G to conclude that $|T|$ is not $> |S|$. So $|S| = |T|$ if F and G are one to one.

   $\square$

2. Exercise 2.4 [1 point]. (Note: The constant 1000 is there for slack and not a specific one.)

Answer We can encode the graph using adjacency lists. For the vertex set $[n]$, we can produce a human readable encoding where the ith index of the list corresponds to a list of nodes that the ith node shares an edge with. We can then encode each value in the list using the optimal prefix free encoding described in class w/ length $|E(n)| + 2log_2(|E(x)|) + 2$. After doing this for each value, we can pass the list through the same encoder and concatenate the output of every list together. This will be a valid encoding. Put more formally,

$$l_i = \{PFE(n) : \text{ if edge (i, n) exists } \forall n \in [n]\}$$

$$E : G_n \to \{0,1\}^{cnlogn}$$
$$E = PFE(l_0) \circ PFE(l_1) \circ ... \circ PFE(l_n)$$

In order to decode:

- keep reading until 01 reached
- chop the part that was read off and decode the numbers using standard PFE decoder for natural numbers to get a list of neighbors in ith index
- draw graph w/ edges to neighbors indicated by list
- repeat until all bits have been read

To count the number of bits:

- for $n \in N, |E(n)| = log_2(n)$
- the PFE of each number will take $log_2(n) + 2log_2(log_2(n)) + 2$ bits (this is the length of the more efficient PFE discussed briefly in class)
- so each list will be $10(log_2(n) + 2log_2(log_2(n)) + 2)$ bits
- the PFE of a list will then be $10(log_2(n) + 2log_2(log_2(n)) + 2) + 2log_2(10(log_2(n) + 2log_2(log_2(n)) + 2)) + 2$ bits
- so in total we have $n(10(log_2(n)+2log_2(log_2(n))+2)+2log_2(10(log_2(n)+2log_2(log_2(n))+2)) + 2)$ bits since we have n lists
- this is less than $1000nlog_2n$ for sufficiently large n

$\square$

3. Prove that the set $\{AND, NOT\}$ is universal. [1 point]

Answer As shown in class, NAND is universal. To show $\{\text{AND}, \text{NOT}\}$ is universal, we can show that any NAND circit can be implemented using only AND/NOT and vice versa.

- to construct NAND in terms of AND and NOT, we can replace very NAND gate with an AND gate followed by a NOT gate
- to construct NOT using only NAND
  - NOT(a) = NAND(a, a)
- to construct AND using only NAND
  - AND(a, b) = NOT(NAND(a, b))
  - AND(a, b) = NAND(NAND(a, b), NAND(a, b))

So $\{\text{AND}, \text{NOT}\}$ is universal.

□

4. Exercise 3.4. To be more precise, the problem is asking you to show that there is a function that **cannot** be computed by a Boolean circuit that is only allowed to use AND/OR (so NOT gates not allowed). As a further hint, you can show that there is a function that takes two inputs and has one output that cannot be computed in such a way (no matter how many AND/OR gates you use). [1 point]

Answer First, we show that the circuit described, C, is monotone. We can easily show that the operations AND and OR are monotone as follows:

AND	
00	0
01	0
10	0
11	1

OR	
00	0
01	1
10	1
11	1

Both of these functions satisfy the property that if any bit in the input is changed from a 0 to a 1, the output can never decrease in value. To be more concrete, for two binary strings that are bitwise $\leq$ each other, $C(x) \leq C(x')$.

If we have any composition of AND and OR (both of which we have shown to be monotone), we will also have a monotone function that satisfies $x, x' \in \{0,1\}^n, x_i \leq x'_i$ for every $i \in [n] \implies C(x) \leq C(x')$. Consider two monotonic functions f and g. We know that if $x \leq x'$, then $f(x) \leq f(x')$. This implies that $g(f(x)) \leq g(f(x'))$ since the output values of the function are no different than the input values and can be treated functionally identically as inputs to other monotone functions.

Therefore, C must be monotone since it is a composition of two monotone functions. Now we will show that C cannot compute a function, and is therefore the set {AND, OR} is not universal. Consider f:

f	
00	1
01	0
10	0
11	0

f is not monotone bc $f(00) > f(01)$ (changing the second bit from 0 to 1 causes a decrease in value). It follows that C must not be able to compute f since C can only compute monotone functions. Therefore, C is not universal.

□