CS 181 Homework 1

Hanna Co

Due: October 18, 2021

1. If S,T are two finite sets, prove that there is a one- to-one mapping from S to T if and only if there is an onto mapping from T to S. [**1 point**]

($\Rightarrow$) There is a one-to-one mapping from S to T if there is an onto mapping from T to S.
An onto mapping existing from T to S means that every image in S has a preimage in T. This means, we can take the inverse and get a mapping from T to S. Since these mappings are functions, we know that an element in S cannot map to multiple elements in T, thus we know that this mapping is one-to-one.
($\Leftarrow$) There is an onto mapping from T to S if there is a one-to-one mapping from S to T.
This means that each element in S is mapped to a unique element in T. It follows that the inverse of this mapping must be an onto mapping frmo T to S. This is because for any x in S, we have a one-to-one function f such that $f(x) = y$ where y is a unique element in T. Thus, f' will take y and map it back to x. Therefore, we have our onto mapping, f'.

Since we have proved this statement in both directions, we conclue that if S, T are two finite sets, then there is a one-to-one mapping from S to T if and only if there is an onto mapping from T to S.

2. Exercise 2.4 [1 point]. Show that there is a string representation of directed graphs with vertex set $[n]$ and degree at most 10 that uses at most $1000nlogn$ bits. More formally, show the following: Suppose we define for every $n \in N$, the set G_n as the set containing all directed graphs (with no self loops) over the vertex set $[n]$ where every vertex has degree at most 10. Then, prove that for every sufficiently large n, there exists a one-to-one function $E : G_n \to \{0,1\}^{[1000nlogn]}$. (Note: The constant 1000 is there for slack and not a specific one)

Rather than have an nxn matrix and using 1's and 0's to represent the edges, we can simply represent the outoging edges as the binary representation of the numbered vertex. Say that we have our vertices numbered from $1 - n$. To represent the largest number, we use $logn$ bits.We know each vertex can have degree of at most 10, so to represent the edges of each vertex, we use $10logn$ bits. We need to do this for every vertex, so we end up using $10nlogn$ bits. If there is a vertex with less than 10 outgoing edges, we can simply pad our string with 0's, to maintain consistency.

3. Exercise 3.3 [1 point]. The problem is essentially asking you to show that you can compute AND using just OR/NOT functions.
Prove that the set $\{OR/NOT\}$ is universal, in the sense that one can compute NAND using these gates.

It is possible to compute AND by using OR and NOT. We can first apply NOT to both bits, OR them together, and then NOT the result. To get NAND, we simply disregard the last NOT operation.

4. Exercise 3.4. Prove that for every n-bit input circuit C that contains only AND and OR gates, as well as gates that compute the constant functions 0 and 1, C is monotone, in the sense that if $x, x' \in \{0,1\}^n, x_i \leq x'_i$ for every $i \in [n]$, then $C(x) \leq C(x')$. Conclude that the set $\{AND, OR, 0, 1\}$ is not universal.
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 are 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]

We have the following truth tables for AND and OR:

AND		
0	0	0
0	1	0
1	0	0
1	1	1

OR		
0	0	0
0	1	1
1	0	1
1	1	1

From these truth tables, we can see that AND and OR are monotonic functions: as their input increases, their output will not decrease. The truth table shows this for the case where the inputs are a single bit, but this can be generalized to input strings of any length, as we can simply break it down into smaller cases. Additionally, we can prove that the composition of monotonic functions are also monotonic. Say we have two monotonic functions, f and g, and two inputs, x_i and x_j, where $x_i < x_j$. Since f is monotonic, we know that $f(x_i) < f(x_j)$. g is also monotonic, so we then conclude that since $g(f(x_i)) < g(f(x_j))$, the composition of two monotonic functions is also monotonic. Thus, we can conclude that composition of AND and OR is also monotonic.

We also know from the following truth table that NOT is not monotonic.

NOT	
0	1
1	0

Therefore, since NOT is not monotonic, and compositions of AND and OR must be monotonic, we can then conclude that AND and OR are not universal, as we are not able to represent every function with some combination of the two.