Problem Set 2: Fine Finite Computation

Due: **4:19pm, Friday, 20 September**

The purpose of this assignment is to develop your understanding of finite computation, focusing on the material in Chapter 3 of the textbook and what we covered in Class 5 and Class 6. We have also included some problems to provide more practice with proof techniques including induction, since it is apparent from Problem Set 1 that most students will benefit from these.

Collaboration Policy: You should do this assignment by yourself and submit your own answers. You may discuss the problems with anyone you want and it is also fine to get help from anyone on problems with LaTeX or Jupyter/Python. You are permitted to use any resources you find for this assignment. You should note in the *Collaborators and Resources* box below the people you collaborated with and any external resources you used (you may exclude resources you used exclusively for help with LaTeX or Jupyter/Python.

Collaborators and Resources: TODO: replace this with your collaborators and resources (if you did not have any, replace this with *None*)

This problem set does not include a Jupyter notebook. Your should complete the assignment by writing your answers in the `ps2.tex`, and submitting your generated PDF file in collab. As with ps0 and ps1, the first thing you should do in ps2.tex is set up your name as the author of the submission by replacing the line, `\submitter{TODO: your name}`, with your name and UVA id, e.g., `\submitter{Grace Hopper (gmh1a)}`. Before submitting your PDF, also remember to (1) list your collaborators and resources, replacing the TODO in `\collaborators{TODO: replace ...}`, and (2) replace the second line in ps2.tex, `\usepackage{uvatoc}` with `\usepackage[response]{uvatoc}` so the directions do not appear in your final PDF.

Problem 1 *Maximum number of Inputs (Induction Practice)*

The *depth* of a circuit is the length of the longest path (in the number of gates) from the an input to an output in the circuit. Prove using induction that the maximum number of inputs for a Boolean circuit (as defined by Definition 3.5 in the book) that produces one output that depends on all of its inputs with depth d is 2^d for all $d \geq 0$. (Note: there are ways to prove this without using induction, but the purpose of this problem is to provide induction practice, so only solutions that are well constructed proofs using the induction principle will be worth full credit.)

Note: Updated 18 September to add the necessary requirement that the circuit has only one output. Otherwise, there are depth 0 circuits with any number of inputs (just connect each input directly to an output without any gates).

Problem 2 *Compare 4 bit numbers (Exercise 3.1 in TCS book)*

Give a Boolean circuit (using only *AND*, *OR*, and *NOT* gates) that computes the function $CMP_8 : \{0,1\}^8 \to \{0,1\}$ such that $CMP_8(a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3) = 1$ if and only if the number represented by $a_0a_1a_2a_3$ is larger than the number represented by $b_0b_1b_2b_3$.

Problem 3 *Compare n bit numbers (Exercise 3.2 in TCS book)*

Prove that there exists a constant c such that for every n there is a Boolean circuit (using only *AND, OR,* and *NOT* gates) C of at most $c \cdot n$ gates that computes the function $CMP_{2n} : \{0,1\}^{2n} \rightarrow 0,1$ such that $CMP_{2n}(a_0 \cdots a_{n-1} b_0 \cdots b_{n-1}) = 1$ if and only if the number represented by $a_0 \cdots a_{n-1}$ is larger than the number represented by $b_0 \cdots b_{n-1}$.

Problem 4 NOR *is universal (Exercise 3.7 in TCS book)*

Let $NOR : \{0,1\}^2 \rightarrow \{0,1\}$ defined as $NOR(a,b) = NOT(OR(a,b))$. Prove that $\{NOR\}$ is a universal set of gates (i.e., anything that can be computed using *AND,OR,NOT* can also be computed using just *NOR*).

Problem 5 *XOR is not universal (based on Exercise 3.5 in TCS book)*

Prove that the set $XOR, 0, 1$ is not universal. (You can use any strategy you want to prove this; see the book for one hint of a possible strategy, but we think you may be able to find easier ways to prove this, and it is not necessary to follow the strategy given in the book.

Problem 6 *Prove that a perceptron cannot compute* XOR.

A *perceptron* is a single layer neural network (Section 3.4.5) that can be modeled by the following function:

$$f(x_0, x_1, \ldots, x_{k-1}) = \sigma(\sum_i w_i x_i)$$

where $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is an activation function. For this question, you may assume the activation function is a rectified linear unit (ReLU), commonly used in deep learning:

$$ReLU(x) = \max(0, x)$$

Prove that there is no way to define *XOR* using a perceptron. That is, show that there is no way to assign the values of w_i such that $f(x_0, x_1) = ReLU(w_0 x_0 + w_1 x_1)$ implements the *XOR* function. You can interpret the output of f ias a Boolean value with values below 0.5 interpreted as False and values ≥ 0.5 interpreted as True (Exercise 3.11 uses a different interpretation which is more complex; if you prefer to use that one instead, this is fine.)

Historical and resource policy note: The proof that a perceptron cannot compute *XOR* is of some historical importance, and it doesn't take much cleverness to find proofs of this (don't click on this link until after submitting your assignment). You should not search for solutions to this problem since the goal of it is for you to think about this yourself and come up with a proof. We think it will be pretty obvious if you write-up a found proof, so expect everyone to be able to explain their proof and how they derived it to us orally if asked. The historical significance of this problem, which is often overblown to the point where some refer to it as the "XOR affair", is that it has been attributed by some as one of the reasons why research in neural networks mostly ceased in the 1980s, except for a few die-hard believers who kept working on it, eventually leading to the explosion of "deep learning" over the past decade, and being awarded the Turing Award in 2018.

Problem 7 ($\star$) *Prove that a two-layer perceptron is universal.*

This will require a bit of creativity and thinking carefully about our definitions. As in Problems 4 and 5, *universal* means that a Boolean circuit where the gates are all two-layer perceptrons can compute the same function as any Boolean circuit.

End of Problem Set 2

Die weitere Erschließung dieses Feldes ist Aufgabe der Zukunft.

