Week 4: EVAL

The purpose of this assignment is to develop your understanding of finite computation, focusing on the material in Chapter 4 and Chapter 5 of the textbook and what we covered in the Week 4 Videos.

Collaboration Policy: You should work on the problems yourself, before discussing with others, and with your cohorts are your cohort meeting. By the Assessed Cohort Meeting, you and all of your cohortmates, should be prepared to present and discuss solutions to all of the assigned problems (including the programming problems). In addition to discussing with your cohortmates, you may discuss the problems with anyone you want, and use any resources you want except for any materials from previous offerings of this course, which are not permitted.

Problem 1 *Finite vs. Infinite functions*

To fully appreciate this week's content, it is critical that you understand the difference between finite and unbounded functions. *TCS Section 1.7* defines a *finite function* to have a fixed-sized input and an *infinite function* to have unbounded input.

Check your understanding of the difference between these two by answering (with proof) the following questions:

– What is the cardinality of the set of all finite functions of the form $\{0,1\}^n \to \{0,1\}$?

– What is the cardinality of the set of all finite functions with binary inputs?

– What is the cardinality of the set of all infinite functions with binary inputs?

Problem 2 *Do we only need NAND?*

We showed that every finite function can be computed by some NAND-Circuit. If NAND-Circuits can do so many things, why don't we use hardware that just implements NAND-Circuits, and why don't we just use the NAND-CIRC programming language?

Name some important ways that actual hardware and software behave differently from NAND-Circuits and the NAND-CIRC programming language. In other words, what are components provided by real hardware and features provided by programming languages that avoid some drawbacks of only implementing NAND-Circuits and the NAND-CIRC language?

Problem 3 *Theory vs. Practice*

This week we saw that NAND circuits (or equivalently the NAND-CIRC programming language) could be used to compute any finite function. We also know that any real computer, since it has finite size/memory capacity, can only compute finite functions. We have a theory of computing that matches anything we can do in practice! Seems like the course should be over then and we should all go to an appropriately socially-distanced beach?

I'm happy to share with you that we still have much computing theory and more models of computing left to explore! What might be the benefits of discussing more models of computing that can be used to implement infinite functions, even though we know those can't be perfectly actualized? To restate this in a more quippy (and therefore more fun) way – *why might it be more practical to use theories that are less like practice*?

Problem 4 *Checking Vocab*

To make sure how we discuss course concepts in English matches the theory we need to be very precise about how we use vocabulary. Each of the statements below does not quite make sense or is ambiguous when considered more formally. Mention at least one thing that is wrong/ambiguous about each, and try to modify the sentence to remove that issue.

1. A Boolean circuit can implement every finite function.
2. AON circuits are universal so they can compute any function.

Problem 5 *EVAL in Python*

Download the *eval.py* program. Follow the instructions in the comments, being sure to implement every function marked with "TODO". This program walks you though the implementation of EVAL in Python.

By the time you're finished with this, you'll have built a Python interpreter for the NAND-CIRC programming language and have an interpreter powerful enough to compute any finite function!

Problem 6 *Equal to Constant Function (TCS exercise 5.3 and* Defining EVAL *video)*

For every $k \in \mathbb{N}$ show that there exists a NAND-CIRC straightline program of no more than $c \cdot k$ lines (where c is a constant) which computes $\texttt{EQUALS}_{x'} : \{0,1\}^k \to \{0,1\}$ where $\texttt{EQUALS}_{x'}(x) = 1$ if and only if $x = x'$.

Problem 7 *Domino Computers*

The *10,000 Domino Computer* (bonus video) attempted to add two 4-bit numbers. Last week you were tasked with building a circuit of your own for addition as a programming problem. Exercise 4.5 of the textbook (which we're not asking you to complete at this time) concludes that you can make a NAND circuit to add together two n-bit numbers using no more than $9n$ gates.

If we assume the Domino Computer used only NAND gates, how many dominoes are required to implement each gate on average? How many dominoes would be required for `LOOKUP_8`?

Use *Theorem 3.12 from TCS* to give a lower bound on the number of dominoes that would be required if this domino computer for `LOOKUP_8` was re-implemented with AON gates (assuming each gate requires the same number of dominoes as NAND does).

⋆ Give a good estimate on the number of dominoes needed to implement a domino computer for any finite function.

⋆⋆ If you have enough dominoes (or something that behaves similarly) handy, build a NAND gate using dominoes and made a video demonstrating that it computes NAND correctly and reliably.

