Week 2: Fine Finite Computation

Authors: TODO: Cohort Name (names of all who contributed)

Collaborators and Resources: TODO: Replace with any additional collaborators and non-course resources you used

This is a template to help with your write-up for Week 3. The actual problem you will write up will be selected by your Cohort Leader at the Assessed Cohort Meeting.

Clone the Problem Set 3 Template Repository

See the Week 1 template for directions on Getting Started with LaTeX. Similarly to Week 1, one member of your cohort should create a copy of the Problem Set 3 repository, by following these steps (we recommend doing this together, with the one creating the repository sharing her screen for everyone to follow along):

1. Download the Problem Set 3 template from: https://uvatoc.github.io/ps/ps3.zip
2. In Overleaf, click on Create First Project or New Project in Overleaf and select Upload Project from the menu.
3. Click Select a .zip file and then select the ps3.zip file you downloaded in step 1.
4. Share the repository with your cohortmates by clicking the "Share" button at the top right of the overleaf window, and entering your cohortmates email addresses in the sharing form.

Click on ps3.tex to see the LaTeX source for this file, which is the file you will modify to prepare your solution. The first thing you should do in ps3.tex is set up your cohort name as the author of the submission by replacing the line, `\submitter{TODO: your name}`, with your the name of your cohort (e.g., `\submitter{Cohort Hopper (Ada Lovelace, Don Knuth)}`). For the list of cohort members, this should usually be everyone in your cohort, but if someone did not contribute during the week, they should not be included in your submission list (and should have informed us about their absence separately).

Before submitting your ps3.pdf file, also remember to:

– List your collaborators and resources, replacing the TODO in `\collaborators{TODO: replace ...}` with your collaborators and resources. You do not need to include

– Replace the second line in ps3.tex, `\usepackage{uvatoc}` with `\usepackage[response]{uvatoc}` so the directions do not appear in your final PDF. You can do this by using the LaTeX comment token, `%`. The rest of the line after a `%` is treated as a comment. You'll notice after you to this, when you Recompile the document, most of it will disappear (everything in \directions is left out, so only your solution will appear in the submitted document).

Problem 2 *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.)

Problem 4 *Fixing Brunelle's Mistakes*

In the lecture video *Cost of LOOKUP*, Professor Brunelle set up a proof by induction that `LOOKUP_k` requires $\leq 4 \cdot 2^k$ `NAND` gates to compute. His proof did not actually work, though, and his calls for help in the video were sadly unanswered.

Let's help him out. Write your own inductive proof to show that `LOOKUP_k` requires $\leq 4 \cdot 2^k$ `NAND` gates. (**Hint:** It may be easier to show that the number of gates required is upper-bounded by some other function that is itself upper-bounded by $4 \cdot 2^k$).

Problem 6 *Universality Checkup*

Prove that $\{\texttt{MAJ}, \texttt{0}, \texttt{1}\}$ is not a universal gate set (where `MAJ` is the majority of three inputs function, and `0` and `1` are constants).

Problem 8 *Full Adders (based on Exercise 4.5)*

Show that for every n there is a NAND-CIRC program to compute ADD_n with at most $9n$ lines where $ADD_n : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ is the function that outputs the sum of two input n-bit numbers (where all inputs and outputs are represented in binary).

Hints: We recommend completing the programming problems before tackling this one.

You may find the other parts of Exercise 4.5 in the book helpful, but it is not necessary to solve this problem using those steps.