TheoryPrep **(Due: 6/28/20)**

Assignment #5: Definitions and Proofs

Name: Student name(s)

We've taken some problems from the course texts and CS 121 material. Some problems are marked as (Challenge) or (Additional Problems). Do the rest first, and if you have time, return to these problems. It's **much more important** to have a rigorous understanding of the core problems and how to prove them than to simply finish all the additional problems.

This pset is different from previous weeks. In particular, there will be substantially more problems asking about intuition or interpretation. This is because as we've learned in the videos for this week, an important part of understanding definitions and proofs is understanding their spirit. A good definition is one that captures our intuition effectively and efficiently (ie with not too much complexity or additional constructions), which involves more aspects of design–this is one way in which math can be beautiful!

Problem 1: Understanding Definitions

Learning goal: Definitions are the fundamental building block of mathematics, and being able to develop intuitive understanding for definitions is critical. Especially in CS 121, the course contains many complex definitions, and building intuition for those definitions is integral to succeeding in the course.

(a) Let's make sure we understand the definition of a boolean circuit (and computing a boolean circuit). For each of these questions, you need only explain in English your reasoning–no need for a formal proof.

1. Why does it make sense for n of the vertices in a boolean circuit to have no in-neighbors?
2. The definition of boolean circuits requires $s \geq m$, or in English that the number of gates is at least as many as the number of outputs. Why do we need to require this?
3. Why is it important that a boolean circuit is a DAG?
4. Definition 3.6 computes a the output of a boolean circuit based on a topological sort of the vertices. Why is it important that we proceed in the order prescribed by topological sort? How does this relate to the condition in AON-CIRC programs that the variables on the righthand side of each line must be either inputs or from previous lines?
5. Note that there is no restriction on the number of outgoing edges from a gate. This means a gate could send its signal to two other gates. Recalling the physical implementations of the computers from the reading, provide an example of a physical model that agrees with our choice to allow a gate output to go to multiple gates and an example which would disagree with this choice.

,

(b) A good definition should be so intuitive/natural that you might have come up with by yourself! We'll make a definition for running a NAND-CIRC program. To review, in section 3.5.3, the textbook defines the NAND-CIRC Programming language similarly to the AON-CIRC programming language, so every line is of the form

$$foo = NAND(bar, blah) \tag{0.1}$$

Define what it means for a NAND-CIRC program P to compute some function $f : \{0,1\}^n \to \{0,1\}^m$

Note: After you do this problem, delete the first red box on the last page of the reading. Compare your definition to the one provided. How is it similar and/or different? Which do you prefer and why?

(c) Let's make sure the definition of a Turing Machine makes sense as well. Again, for the below questions, explain your reasoning in English.

1. Why can't we define a Turing Machine as anything that python can compute? This is a real question–take some time to think about it!

2. Recalling that our motivation for a turing machine is a human computer, what does the transition function δ_M correspond to? What does the tape correspond to?

3. Your friend complains that we don't need an $\oslash$ since we can just use 0. Suppose that we replaced $\oslash$ with 0 in the definition. Would this still be a satisfactory definition? Why or why not?

4. Your friend notes that a human computer sometimes starts in the middle of the paper and therefore suggests that we should initialize $i = 10$ in the definition of a Turing Machine. Is this an equivalent definition? Does it make sense?

5. (Additional Problem) Your friend then observes that we are representing states as $[k]$, the set $\{0, 1, ..., k-1\}$, but when a human is doing computation it's more apt to store their memory as a string (wow this friend is really interrogating the definition–what a good theoretical computer scientist!). Therefore, your friend suggests representing states as A^m where $A = \{a, b, c, ..., z, \text{“ ”}\}$ (so alphabetical strings of length m). This means we'd change the definition to be below (note the only changes are $\delta_M :: A^m \times \Sigma \to A^m \times \Sigma \times \{L, R, S, H\}$ and $s =$ “”, but we copy the rest of the definition for convenience)

   A (one tape) Turing Machine with alphabet $\Sigma \supseteq \{0, 1, \triangleright, \oslash\}$ is represented by a transition function $\delta_M : A^m \times \Sigma \to A^m \times \Sigma \times \{L, R, S, H\}$.

   For every $x \in \{0,1\}^*$, the output of M on input x, denoted by $M(x)$ is the result of the following process

   - We initialize T to be the sequence $\triangleright, x_0, x_1, \ldots, x_{n-1}, \oslash, \ldots$ where $n = |x|$.
   - We initialize $i = 0, s =$ “” (so the initial state is the empty string)
   - We then repeat the following process
     (a) Let $(s', \sigma', D) = \delta_M(s, T[i])$.
     (b) Set $s \to s', T[i] \to \sigma'$.
     (c) If $D = R$ then set $i \to i + 1$ If $D = L$ then set $i \to \max(i - 1, 0)$. If $D = H$ then halt.
   - If the process above halts, then M's output, denoted by $M(x)$ is the string $y \in \{0,1\}^*$ obtained by concatenating all the symbols in $\{0, 1\}$ in positions $T[0], \ldots, T[i]$ where i is the final head position. If the process does not halt then we write $M(x) = \perp$

   Is this an equivalent definition?

Problem 2: Reading Proofs

Learning goal: You read a number of longer proofs for this week, and that's an important skill. This section will help you make sure you understood the proofs well by asking you to give intuition and evaluate if these proofs would work for slightly different situations.

(a) An important part of understanding proofs is understanding them intuitively. For each of the longer proofs from listed below, summarize the proof strategy in 1 to 2 English sentences.

- Equivalence of Boolean Circuit and AON-CIRC.
- Univerality of NAND
- (Additional Problem) Proof that there does not exist a surjection $f : \mathbb{N} \to \mathbb{P}(\mathbb{N})$ from week 3.
- (Additional Problem) Proof of Hall's Theorem (necessary and sufficient condition for matchings) from week 4.

(b) Recall from the second video the proof of universality of NAND. Suppose we tried to use the same proof to prove the universality of IF. In particular, suppose we used the same proof (with some minor changes) to prove the below theorem (let IF-CIRC programs be composed of lines of the form $foo = IF(bar, blah, zoo)$ that are evaluated similarly to NAND-CIRC by evaluating each line in succession, but setting $foo = blah$ if $bar = 1$ and otherwise setting $foo = zoo$).

Theorem 1 *There exists some constant $c > 0$ such that $\forall n, m > 0$ and $f : \{0,1\}^n \to \{0,1\}^m$ there is an IF-CIRC program of at most $cm2^n$ lines that computes f.*

Would this work? If so, write the proof. If not, explain where the proof fails and then propose a small modification (ie added features) you could make to IF-CIRC to make it universal.

(c) Note: Before you do this problem, make sure you've done Problem 1.b. You should delete the first red box on the last page of the reading to reveal the definition of how to compute the output of a NAND-CIRC program. You should use this definition.

The textbook provides a proof to demonstrate that a function is computable by a boolean circuit if and only if it's computable by an AON-CIRCUIT program of similar length (Theorem 3.9). Suppose we tried to use the same proof to prove that a function is computable by a NAND circuit if and only if it's computable by NAND-CIRC program of similar length. In particular, let's say we were trying to prove the below theorem.

Theorem 2 *Let $f : \{0,1\}^n \to \{0,1\}^m$ a function and $s \geq m$ some natural number. Then f is computable by a NAND circuit of s gates if and only if it's computable by an NAND-CIRC progam of s lines.*

Would this work? If so, write the proof. If not, explain where the proof fails.

(d) (Additional Problem) You're trying to implement NAND circuits in real life with marble computers (see page 21 of the reading for a refresher), but because each gate only shoots out one marble, you realize you can only have one output coming out of each gate. You decide to model this by creating a type of circuits called "marble NAND circuits" that are the same as NAND circuits, except each gate may only have one outgoing edge (the input vertices may have any number of outgoing edges). This captures the constraint that a marble gate can only output one marble. Your friend remarks that this is totally fine since you can be just as efficient even if each gate in your circuit has a singular output. In particular, they claim the following theorem holds by a slightly modified proof of Theorem 3.9.

Theorem 3 *Let $f : \{0,1\}^n \to \{0,1\}^m$ a function and $s \geq m$ some natural number. Then f is computable by a NAND circuit of s gates if and only if it's computable by a marble NAND circuit of at most s gates.*

Would modifying Theorem 3.9 work to prove this? If so, give the proof. If not, explain where the proof fails, and give a weaker result that would hold (recall that a weaker result is something that would be implied by Theorem 3). If you give a weaker result try to either prove it or explain how you would prove it.

(e) (Additional Problem) Sometimes proofs will be very concise and you may have to fill in the details for yourself. Below is a concise proof taken from American Mathematical Monthly, v.116, issue 1, January 2009, page 69. It is proving that $\sqrt{2}$ is irrational (and is slightly different from the commonly-presented proof by contradiction).

Assume $\sqrt{2}$ is rational and choose the least integer $q > 0$ such that $(\sqrt{2}-1)q$ is a nonnegative integer. Let $q' = (\sqrt{2}-1)q$. Clearly $0 < q' < q$, but we can easily show that $(\sqrt{2}-1)q'$ is a nonnegative integer, contradicting the minimality of q.

Write out a more detailed version of this proof that explains and justifies each step. Comment on which proof you think is better–does this depend on the scenario?

Problem 3: Longer Proofs

Learning goal: This week we will focus on doing one longer proof that combines understanding complex definitions and a couple of proof techniques. This will likely be one of the longest proofs you've done in the course, and it will hopefully help you grow your complex proof-writing skills.

(a) Let XOR-CIRC be the programming language where each line is of the form $foo = XOR(bar, blah)$, $foo = 1$, or $foo = 0$ (so each line can take one of those three forms). Compare the power of XOR-CIRC and NAND-CIRC. In particular, you should prove one of the following three things: NAND-CIRC and XOR-CIRC are equally powerful, so any function computable in one is computable in the other; NAND-CIRC is more powerful than XOR-CIRC so everything computable in XOR-CIRC is computable in NAND-CIRC, but there exists a function computable in NAND-CIRC not computable in XOR-CIR; XOR-CIRC is more powerful than NAND-CIRC so everything computable in NAND-CIRC is computable in XOR-CIRC, but there exists a function computable in XOR-CIRC not computable in XOR-CIR; or neither is more powerful, so there exist functions computable in each programming language not computable in the other.

It may be helpful to write a proof outline for your own proof first.

Problem 4: Review

Learning goal: The goal for this section is to reflect on how you can improve your proof skills!

(a) Select the problem from a previous week that you think you would learn the most from redoing. Copy the problem and your instructor's feedback below, rewrite the proof, and add a short reflection on the changes you made.

(b) (Additional Question) Choose another problem you received feedback on and copy the problem and feedback you received, rewrite the proof, and write a sentence about why you made your changes (so do Part A again) OR solve a problem you left blank from a previous week (try to pick from a week you felt least secure on).

Problem 5: Logistics

Purpose: This helps us make sure the course is going at the right speed!

(a) How long did you spend on the videos and readings this week?

(b) How long (including time in problem sessions) did you spend on this problem set?

(c) Do you have any feedback about the course in general (did the videos and readings sufficiently prepare you for the problem set)?