# Homework 1

## Contents



This is due Thursday April 25 by 11:59pm, for full credit. It can be turned in Friday April 26 by 11:59pm for a 10% reduction.

You can work with 1-2 other students on it, you just have to indicate who else you worked with, and everyone has to write their own solutions.

This homework is designed to be a review of Weeks 1-2 of the course (Chapter 1-4 of the book).

You submit **via gradescope on bruinlearn**: you do not turn in anything on this page.

In terms of workflow, I would recommend:

- write it out by hand: everything is short and you should be able to write it out by hand easily
- after you done with it, scan it or take photos of it to put into gradescope in bruinlearn
- be working near a computer or phone so you can use the links on this page, which

reference back to the book

The first three problems give one a sense of what 'developing a library' of primitive recursive functions is like. This is what Skolem did in 1923, and it is similar with any presentation of the primitive recursive functions: one has to spend a lot of time showing that many functions are primitive recursive.

# Problem 1

Show that exponentiation $n^m$ is primitive recursive, by identifying the base and iterator. Emulate the arguments we gave for addition and multiplication.

# Problem 2

In the text, we gave the formal definition of "definition by primitive recusion" for two-place functions.

Here is the definition for one-place functions.

Suppose a *base number* $b$ and an *iterator function* $I : \mathbb{N}^2 \to \mathbb{N}$ are given. Then the function $f : \mathbb{N} \to \mathbb{N}$ defined by *primitive recursion* from this base and iterator is the function which satisfies the following, for all $m$:

$$f(m) = \begin{cases} b & \text{if } m = 0 \\ I(m-1, f(m-1)) & \text{if } m > 0 \end{cases}$$

Show that factorial $f(n) = n!$ is primitive recursive by finding a base number $b$ and iterator $I$ which define it, and where we already know from previous work that $I$ is primitive recursive.

Show that the following predecessor function is primitive recursive:

$$\mathrm{pred}(m) = \begin{cases} 0 & \text{if } m = 0 \\ m - 1 & \text{if } m > 0 \end{cases}$$

*Note*: this example is, to my knowlede, the reason why we allow the iterator to have one more argument than the function which it defines.

# Problem 4

Recall the Ackermann function $\varphi_\ell(n, m)$

Give yourself one page and try to calculuate as much of $\varphi_3(3, 3)$ as you can.

The moral of this exercise is that it is very natural to see the Turing machine model as emerging from an attempt to describe the process of evaluating a function like this: the cells of the tape are the different locations on the page, the machine head is the pencil or pen, and the instructions are the rules embedded in the different defintional clauses of the function.

# Problem 5

Recall that Heyting says that:

> $\neg\varphi$ can be asserted if and only if we possess a construction which from the supposition that a construction $\varphi$ were carried out, leads to a contradiction ([Heyting, 1956] p. 102).

Note that this is just the arrow clause of BHK applied to $\varphi \to \bot$, where $\bot$ is an abbreviation for something which can never be asserted (i.e. a contradiction). That is, in the tradition of intuitionistic logic, $\neg\varphi$ is an abbreviation for $\varphi \to \bot$.

Answer the following four questions:

1. In intuitionistic logic, what is $\neg\neg\varphi$ an abbreviation for? Your answer should consist just of a single formula with two arrows in it.
2. According to BHK, when can you assert $\neg\neg\varphi$?
3. According to BHK, should you be able to assert $\varphi \to \neg\neg\varphi$ and why? Give an answer in 1-2 complete English sentences.
4. According to BHK, should you be able to assert $\neg\neg\varphi \to \varphi$ and why? Give an answer in 1-2 complete English sentences.

# Problem 6

Using the Turing machine visualizer, write a program which when given an input of a string

formed from the three letters $a, b, c$, adds the expression $ba$ to the very right.

## Problem 7

Using the Turing machine visualizer, write a program which looks for the first two consecutive $b$'s and changes them both to $r$.

## Problem 8

Emulate the proof of the non-computability of the halting set $K$ to show that the following set is not computable:

$$K_0 = \{(e, n) : \varphi_e(n) \downarrow\}$$

In the proof, it can be assumed that the pairing function sending natural numbers $e$ and $n$ to natural number $(e, n)$ is computable, and it can be assumed that given a natural number one can compute whether it is a pair $(e, n)$ and if so one can determine $e, n$ thereby.

## Problem 9

If $A, B$ are computable sets of natural numbers, then so too are $A \cap B$ and $A \cup B$.

First, use Kleene's result to show this. *Hint*: this one involves producing definitions of $A \cap B$ and $A \cup B$ in terms of definitions of $A, B$.

Second, use the original Turing model to show

this. *Hint* use the universal machine.

## Problem 10

A key feature of computability is that it a property of sets, rather than a property of programs. (It is extensional as opposed to intensional).

For each computable set $A$ and Turing machine program $i$ which computes $A$, describe a distinct Turing machine program $j$ which computes $A$.

Previous
Chapter 4

Next
References