CSC363 Tutorial #2

Turing machines and stuff

January 26, 2022

Learning objectives this tutorial

- Prove that some functions are primitive recursive.
- Prove more functions are primitive recursive.
- Talk about "computable sets".

A bit about myself?

Hi! I'm some 4th year student studying math/cs. I was sick last week ;w;

- Contact: pol.zhang@utoronto.ca, or if you prefer Discord, sjorv#0943
- Hobbies: Gaming, taking naps at inappropriate times

Not my cat. Cats are cute though.

- Favourite food: sushi juice
- Office hours: 1-2pm Friday
- Website (you can find tutorial slides there): sjorv.github.io

PRIM

Question: What does PRIM stand for?

PRIM

Question: What does PRIM stand for?
Question: What are the initial functions in PRIM?

PRIM

Recall that PRIM is a set of functions from $\mathbb{N}^k$ to $\mathbb{N}$, intuitively meant to capture what a "computable" function is.

DEFINITION 2.1.1 (The Primitive Recursive Functions)

1) The **initial functions** *(a) – (c) are primitive recursive:*

(a) The **zero function** *defined by*

$$\mathbf{0}(n) = 0, \qquad \forall n \in \mathbb{N},$$

(b) The **successor function** *defined by*

$$n' = n + 1, \qquad \forall n \in \mathbb{N},$$

(c) The **projection functions** U_i^k *defined by*

$$U_i^k(\overrightarrow{m}) = m_i, \qquad \text{each } k \geq 1, \text{ and } i = 1, \ldots, k,$$

(where we write $\overrightarrow{m} = m_1, \ldots, m_k$*).*

2) If $g, h, h_0, \ldots, h_l$ *are primitive recursive, then so is* f *obtained from* $g, h, h_0, \ldots, h_l$ *by one of the rules:*

(d) **Substitution**, *given by:*

$$f(\overrightarrow{m}) = g(h_0(\overrightarrow{m}), \ldots, h_l(\overrightarrow{m})),$$

(e) **Primitive recursion**, *given by:*

$$\begin{aligned} f(\overrightarrow{m}, 0) &= g(\overrightarrow{m}), \\ f(\overrightarrow{m}, n+1) &= h(\overrightarrow{m}, n, f(\overrightarrow{m}, n)). \end{aligned} \tag{2.1}$$

Keep this definition handy!

Constant functions are in prim

Task: Prove that $f_k : \mathbb{N} \to \mathbb{N}$, given by $f_k(n) = k$ for all $n \in \mathbb{N}$, is primitive recursive.

Constant functions are in prim

Task: Prove that $f_k : \mathbb{N} \to \mathbb{N}$, given by $f_k(n) = k$ for all $n \in \mathbb{N}$, is primitive recursive.
Ans: We know $\mathbf{0}$ (the zero function) and S (the successor function) are primitive recursive, from (a) and (b). Thus repeatedly applying the substitution rule (d),

$$f_k(n) = \underbrace{S(S(\ldots(S(}_{k \text{ times}}\mathbf{0}(n)))\ldots))$$

is primitive recursive.

Addition is in prim

Recall from Lecture 2: the addition function $+ : \mathbb{N}^2 \to \mathbb{N}$, $+(m, n) = m + n$, is in PRIM.

Addition is in prim

Recall from Lecture 2: the addition function $+ : \mathbb{N}^2 \to \mathbb{N}$, $+(m, n) = m + n$, is in PRIM.

Informal Proof:

$$+(x, 0) = x,$$

$$+(x, n+1) = S(+(x, n))$$

so using the rule of primitive recursion (e), $+$ is primitive recursive.

Addition is in prim

Recall from Lecture 2: the addition function $+ : \mathbb{N}^2 \to \mathbb{N}$,
$+(m, n) = m + n$, is in PRIM.
Informal Proof:

$$+(x, 0) = x,$$

$$+(x, n+1) = S(+(x, n))$$

so using the rule of primitive recursion (e), $+$ is primitive recursive.
Formal Proof: We have

$$+(x, 0) = P^1_1(x),$$

$$+(x, n+1) = g(x, n, +(x, n))$$

where $g(a, b, c) = S(P^3_3(a, b, c))$ is primitive recursive by the substitution rule.

Multiplication is in PRIM

Task: Now that we know $+$ is in PRIM, prove that the multiplication function $\times : \mathbb{N}^2 \to \mathbb{N}$, $\times(m, n) = mn$, is in PRIM.

Multiplication is in PRIM

Task: Now that we know $+$ is in PRIM, prove that the multiplication function $\times : \mathbb{N}^2 \to \mathbb{N}$, $\times(m, n) = mn$, is in PRIM.
Informal Proof:

$$\times(x, 0) = 0,$$

$$\times(x, n+1) = +(\times(x, n), x)$$

so using the rule of primitive recursion (e), $\times$ is primitive recursive.
Formal Proof: We have

$$\times(x, 0) = \mathbf{0}(x),$$

$$\times(x, n+1) = g(x, n, \times(x, n))$$

where $g(a, b, c) = +(P_1^3(a, b, c), P_3^3(a, b, c))$ is primitive recursive by the substitution rule, since we've proven $+$ is primitive recursive.

“Subtraction” is in PRIM

Task: Show that $\delta : \mathbb{N} \to \mathbb{N}$, $\delta(n) = \begin{cases} n-1 & n \geq 1 \\ 0 & n = 0 \end{cases}$ is in PRIM.

Hint: Define $f(x, n) = \begin{cases} n-1 & n \geq 1 \\ 0 & n = 0 \end{cases}$ (basically ignoring the first parameter). If we show f is primitive recursive, then $\delta(n) = f(n, n)$ is primitive recursive by the substitution rule.

"Subtraction" is in PRIM

Task: Show that $\delta : \mathbb{N} \to \mathbb{N}$, $\delta(n) = \begin{cases} n-1 & n \geq 1 \\ 0 & n = 0 \end{cases}$ is in PRIM.

Hint: Define $f(x, n) = \begin{cases} n-1 & n \geq 1 \\ 0 & n = 0 \end{cases}$ (basically ignoring the first parameter). If we show f is primitive recursive, then $\delta(n) = f(n, n)$ is primitive recursive by the substitution rule.

Proof: Define f as in the hint. We have

$$f(x, 0) = \mathbf{0}(x),$$

$$f(x, n+1) = P^3_2(x, n, f(x, n)) \quad (= n)$$

so f is primitive recursive. Thus $\delta(n) = f(n, n)$ is primitive recursive by substitution rule.

“Subtraction” is in PRIM

Unfortunately, we can’t define actual subtraction as a function from $\mathbb{N}^2$ to $\mathbb{N}$! We are only allowed to output natural numbers :(

“Subtraction” is in PRIM

Unfortunately, we can't define actual subtraction as a function from $\mathbb{N}^2$ to $\mathbb{N}$! We are only allowed to output natural numbers :(

Task: Show that $\dot{-} : \mathbb{N}^2 \to \mathbb{N}$, $\dot{-}(x, y) = \begin{cases} x - y & x \geq y \\ 0 & x < y \end{cases}$ is primitive recursive.

Hint: primitive recursion, using δ from before!

"Subtraction" is in PRIM

Unfortunately, we can't define actual subtraction as a function from $\mathbb{N}^2$ to $\mathbb{N}$! We are only allowed to output natural numbers :(

Task: Show that $\dot{-} : \mathbb{N}^2 \to \mathbb{N}$, $\dot{-}(x, y) = \begin{cases} x - y & x \geq y \\ 0 & x < y \end{cases}$ is primitive recursive.

Hint: primitive recursion, using δ from before!

Proof: We have

$$\dot{-}(x, 0) = P_1^1(x),$$

$$\dot{-}(x, n+1) = \delta(\dot{-}(x, n))$$

so $\dot{-}$ is primitive recursive.

we're being a little informal here! but hopefully you can translate this into a "formal proof" as before.

What is in PRIM?

So far, we've shown the following are in PRIM:

- Any constant function f_k.
- Addition, multiplication.
- "Subtraction" (which doesn't go below zero, to make $\mathbb{N}$ happy).

What is in PRIM?

So far, we've shown the following are in PRIM:

- Any constant function f_k.
- Addition, multiplication.
- "Subtraction" (which doesn't go below zero, to make $\mathbb{N}$ happy).

What about the following functions?

- Absolute difference $(x, y) \mapsto |x - y|$.

What is in PRIM?

So far, we've shown the following are in PRIM:

- Any constant function f_k.
- Addition, multiplication.
- "Subtraction" (which doesn't go below zero, to make $\mathbb{N}$ happy).

What about the following functions?

- Absolute difference $(x, y) \mapsto |x - y|$.
- The "is zero?" function (inverse sign function):

$$\overline{\mathrm{sg}}(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0. \end{cases}$$

What is in PRIM?

So far, we've shown the following are in PRIM:

- Any constant function f_k.
- Addition, multiplication.
- "Subtraction" (which doesn't go below zero, to make $\mathbb{N}$ happy).

What about the following functions?

- Absolute difference $(x, y) \mapsto |x - y|$.
- The "is zero?" function (inverse sign function):

$$\overline{\mathrm{sg}}(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0. \end{cases}$$

- The "is not zero" function (sign function):

$$\mathrm{sg}(x) = \begin{cases} 0 & x = 0 \\ 1 & x \neq 0. \end{cases}$$

What is in PRIM?

So far, we've shown the following are in PRIM:

- Any constant function f_k.
- Addition, multiplication.
- "Subtraction" (which doesn't go below zero, to make $\mathbb{N}$ happy).

What about the following functions?

- Absolute difference $(x, y) \mapsto |x - y|$.
- The "is zero?" function (inverse sign function):

$$\overline{\mathrm{sg}}(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0. \end{cases}$$

- The "is not zero" function (sign function):

$$\mathrm{sg}(x) = \begin{cases} 0 & x = 0 \\ 1 & x \neq 0. \end{cases}$$

They are all primitive recursive!

DISCLAIMER!!!!

I'm about to lie to you.

In our upcoming definition of a "computable set", we only assume PRIM functions are "computable". This is not true! There are functions not in PRIM that are also computable, such as the Ackermann function.

DISCLAIMER!!!!

I'm about to lie to you.

In our upcoming definition of a "computable set", we only assume PRIM functions are "computable". This is not true! There are functions not in PRIM that are also computable, such as the Ackermann function.

So in reality, there are computable sets out there that don't fit our definition of "computable set". Explaining this will require week 3 lecture material...

DISCLAIMER!!!!

I'm about to lie to you.

In our upcoming definition of a "computable set", we only assume PRIM functions are "computable". This is not true! There are functions not in PRIM that are also computable, such as the Ackermann function.

So in reality, there are computable sets out there that don't fit our definition of "computable set". Explaining this will require week 3 lecture material...

Either way, every "computable set" from our definition will turn out to be "computable" in the actual definition.

Computable sets

Consider $S \subseteq \mathbb{N}$. How do we define the statement "S is computable", in terms of primitive recursion?

---

[1] I am lying to you here! The actual definition uses "recursive" instead of "primitive recursive", and "recursive" constitutes a larger class of functions. You'll learn (or have learned) about it in this week's lecture.

Computable sets

Consider $S \subseteq \mathbb{N}$. How do we define the statement "S is computable", in terms of primitive recursion?
A natural way would be to define "S is computable" by looking at its *characteristic function* $\chi_S : \mathbb{N} \to \mathbb{N}$, given by

$$\chi_S(n) = \begin{cases} 0 & n \notin S \\ 1 & n \in S. \end{cases}$$

Definition: A set $S \subseteq \mathbb{N}$ is *computable* when its characteristic function χ_S is primitive recursive.[1]
Task: Show that the empty set is computable.

[1] I am lying to you here! The actual definition uses "recursive" instead of "primitive recursive", and "recursive" constitutes a larger class of functions. You'll learn (or have learned) about it in this week's lecture.

Computable sets

Consider $S \subseteq \mathbb{N}$. How do we define the statement "S is computable", in terms of primitive recursion?
A natural way would be to define "S is computable" by looking at its *characteristic function* $\chi_S : \mathbb{N} \to \mathbb{N}$, given by

$$\chi_S(n) = \begin{cases} 0 & n \notin S \\ 1 & n \in S. \end{cases}$$

Definition: A set $S \subseteq \mathbb{N}$ is *computable* when its characteristic function χ_S is primitive recursive.[1]
Task: Show that the empty set is computable.
Ans: The empty set's characteristic function is just the zero function, which is primitive recursive.

[1] I am lying to you here! The actual definition uses "recursive" instead of "primitive recursive", and "recursive" constitutes a larger class of functions. You'll learn (or have learned) about it in this week's lecture.

Computable sets

Task: Show that the singleton set $\{0\}$ is computable.

Computable sets

Task: Show that the singleton set $\{0\}$ is computable.
Ans: The characteristic function of $\{0\}$ is just the inverse sign function

$$\overline{\mathrm{sg}}(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0. \end{cases}$$

which, as we have shown, is computable.

Computable sets

Task: Show that any singleton set $\{k\}$, with $k \in \mathbb{N}$ is computable.

Computable sets

Task: Show that any singleton set $\{k\}$, with $k \in \mathbb{N}$ is computable.
Ans: The absolute difference $(x, y) \mapsto |x - y|$ is primitive recursive. Thus

$$\overline{\mathrm{sg}}(|x - k|) = \begin{cases} 1 & x = k \\ 0 & x \neq k \end{cases}$$

is primitive recursive. But this is just the characteristic function of $\{k\}$!

Computable sets

Task: Show that any *finite* set $\{k_1, \ldots, k_m\}$, with $k_1, \ldots, k_m \in \mathbb{N}$, is computable.

Computable sets

Task: Show that any *finite* set $\{k_1, \ldots, k_m\}$, with $k_1, \ldots, k_m \in \mathbb{N}$, is computable.
Ans: What if we added the indicator functions of each singleton set $\{k_1\}, \ldots, \{k_m\}$? Notice that

$$\overline{\mathrm{sg}}(|x - k_1|) + \ldots + \overline{\mathrm{sg}}(|x - k_m|) > 0$$

if and only if x is in $\{k_1, \ldots, k_m\}$. Thus

$$\mathrm{sg}(\overline{\mathrm{sg}}(|x - k_1|) + \ldots + \overline{\mathrm{sg}}(|x - k_m|)) = 1$$

if and only if x is in $\{k_1, \ldots, k_m\}$, so our characteristic function is primitive recursive.

Computable sets

Task: Suppose $S_1, S_2 \subseteq \mathbb{N}$ are both computable. Show that $S_1 \cup S_2$ is computable.

Computable sets

Task: Suppose $S_1, S_2 \subseteq \mathbb{N}$ are both computable. Show that $S_1 \cup S_2$ is computable.
Ans: Add the indicator functions!

$$\mathrm{sg}(\chi_{S_1}(x) + \chi_{S_2}(x)) = 1$$

if and only if x is in S_1 or x is in S_2, so our characteristic function is primitive recursive.

Computable sets

What other sets are computable?

- The even numbers $\{0, 2, 4, \ldots\}$. (Prove the remainder function $(x, y) \mapsto x\%y$ is in PRIM!)
- The prime numbers.
- Pretty much every set that ever comes up in number theory!

What's this all about?

We're trying to build a mathematical definition of "computable", from multiple different perspectives:

- Computation with Turing machines;

[2] Unless the curriculum changes, that is.

What's this all about?

We're trying to build a mathematical definition of "computable", from multiple different perspectives:

- Computation with Turing machines;
- Computation with URMs (briefly);

[2] Unless the curriculum changes, that is.

What's this all about?

We're trying to build a mathematical definition of "computable", from multiple different perspectives:

- Computation with Turing machines;
- Computation with URMs (briefly);
- Computation with primitive recursive functions;

[2] Unless the curriculum changes, that is.

What's this all about?

We're trying to build a mathematical definition of "computable", from multiple different perspectives:

- Computation with Turing machines;
- Computation with URMs (briefly);
- Computation with primitive recursive functions;
- ~~Computation with Lambda Calculus~~ Oh no! We're not gonna cover this unfortunately :([2]

[2] Unless the curriculum changes, that is.

What's this all about?

We're trying to build a mathematical definition of "computable", from multiple different perspectives:

- Computation with Turing machines;
- Computation with URMs (briefly);
- Computation with primitive recursive functions;
- ~~Computation with Lambda Calculus~~ Oh no! We're not gonna cover this unfortunately :([2]

These all turn out to give an "equivalent" definition of what is computable. Fundamentally, there are things that computers cannot do, regardless of the framework of computation we use!
Primitive recursion is probably the most "abstract" and thus the hardest to grasp intuitively, but it is worthwhile from a historical perspective.

[2] Unless the curriculum changes, that is.