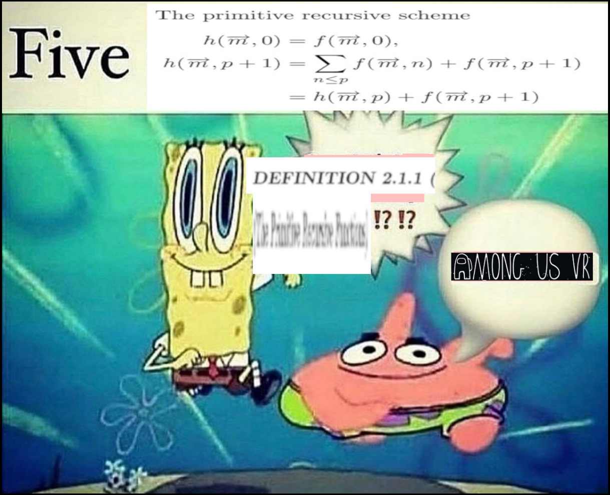
CSC363 Tutorial #2

Primitive Recursive Functions



January 25, 2023

Things covered in this tutorial

Things covered in this tutorial

- What's a "function"?

Things covered in this tutorial

- What's a "function"?
- What are the primitive recursive functions?

Things covered in this tutorial

- What's a "function"?
- What are the primitive recursive functions?
- How can I use composition and primitive recursion?

Things covered in this tutorial

- What's a "function"?
- What are the primitive recursive functions?
- How can I use composition and primitive recursion?
- Could I get a hint for A1 Q3?

Things covered in this tutorial

- What's a "function"?
- What are the primitive recursive functions?
- How can I use composition and primitive recursion?
- Could I get a hint for A1 Q3? No, but you may cite these slides for your homework. (You still have to provide a formal proof that the functions in this tutorial are PRIM.)[1]

[1] Citation: Paul "sjorv" Zhang. "CSC363 Tutorial #2. Primitive Recursive Functions. Five the primitive recursive scheme among us vr.". Chungus Publishing, 2023.

What's a **function**?

What's a **function**?

In the previous tutorial (Turing machines), functions were from Σ^* to Σ^*.

What's a **function**?

In the previous tutorial (Turing machines), functions were from Σ^* to Σ^*. computer science!!1!! 🤮🤮

In this tutorial, all functions are from $\mathbb{N}$ to $\mathbb{N}$ (or from $\mathbb{N}^k$ to $\mathbb{N}$).

What's a **function**?

In the previous tutorial (Turing machines), functions were from Σ^* to Σ^*. computer science!!1!! 🤮🤮

In this tutorial, all functions are from $\mathbb{N}$ to $\mathbb{N}$ (or from $\mathbb{N}^k$ to $\mathbb{N}$).
($\mathbb{N}$ includes 0 in this course.)

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Rules:

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Rules:

1. The following initial functions are in PRIM:

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Rules:

1. The following initial functions are in PRIM:
   - The zero function $Z : \mathbb{N} \to \mathbb{N}$, $Z(n) = 0$.

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Rules:

1. The following initial functions are in PRIM:
   - The zero function $Z : \mathbb{N} \to \mathbb{N}$, $Z(n) = 0$.
   - The successor function $S : \mathbb{N} \to \mathbb{N}$, $S(n) = n + 1$.

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Rules:

1. The following initial functions are in PRIM:
   - The zero function $Z : \mathbb{N} \to \mathbb{N}$, $Z(n) = 0$.
   - The successor function $S : \mathbb{N} \to \mathbb{N}$, $S(n) = n + 1$.
   - For each $k \in \mathbb{N}$ and $i = 1, \ldots, k$, the projection function $C_i^k : \mathbb{N}^k \to \mathbb{N}$, $C_i^k(n_1, n_2, \ldots, n_k) = n_i$.[2]

[2] In other words, C_i^k takes in k arguments, and returns the ith one.

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Rules:

2. The following structural recursion rules:

[3] $\vec{m} \in \mathbb{N}^\ell$.

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Rules:

2. The following structural recursion rules:
   - Composition: If $g : \mathbb{N}^k \to \mathbb{N}$ and $h_1, h_2, \ldots, h_k : \mathbb{N}^\ell \to \mathbb{N}$ are in PRIM, then $f : \mathbb{N}^\ell \to \mathbb{N}$ given by[3]

     $$f(\vec{m}) = g(h_1(\vec{m}), \ldots, h_k(\vec{m}))$$

     is primitive recursive.

[3] $\vec{m} \in \mathbb{N}^\ell$.

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Rules:

2. The following structural recursion rules:
   - Composition: If $g : \mathbb{N}^k \to \mathbb{N}$ and $h_1, h_2, \dots, h_k : \mathbb{N}^\ell \to \mathbb{N}$ are in PRIM, then $f : \mathbb{N}^\ell \to \mathbb{N}$ given by[3]

     $$f(\vec{m}) = g(h_1(\vec{m}), \dots, h_k(\vec{m}))$$

     is primitive recursive. **In other words, you can compose primitive recursive functions.**

[3] $\vec{m} \in \mathbb{N}^\ell$.

Primitive Recursive functions

The primitive recursive functions (PRIM) are a collection of functions $f : \mathbb{N}^k \to \mathbb{N}$ defined using structural recursion.

Rules:

2. The following structural recursion rules:
   - Composition: If $g : \mathbb{N}^k \to \mathbb{N}$ and $h_1, h_2, \ldots, h_k : \mathbb{N}^\ell \to \mathbb{N}$ are in PRIM, then $f : \mathbb{N}^\ell \to \mathbb{N}$ given by[3]

     $$f(\vec{m}) = g(h_1(\vec{m}), \ldots, h_k(\vec{m}))$$

     is primitive recursive. **In other words, you can compose primitive recursive functions.**
   - Primitive Recursion: If $g : \mathbb{N}^\ell \to \mathbb{N}$ and $h : \mathbb{N}^{\ell+2} \to \mathbb{N}$ are in PRIM, then $f : \mathbb{N}^{\ell+1} \to \mathbb{N}$ given by

     $$f(\vec{m}, 0) = g(\vec{m})$$
     $$f(\vec{m}, n+1) = h(\vec{m}, n, f(\vec{m}, n))$$

     is primitive recursive.

[3] $\vec{m} \in \mathbb{N}^\ell$.

Primitive Recursion

Primitive Recursion: If $g : \mathbb{N}^\ell \to \mathbb{N}$ and $h : \mathbb{N}^{\ell+2} \to \mathbb{N}$ are in PRIM, then $f : \mathbb{N}^{\ell+1} \to \mathbb{N}$ given by

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

is primitive recursive.

Primitive Recursion

Primitive Recursion: If $g : \mathbb{N}^\ell \to \mathbb{N}$ and $h : \mathbb{N}^{\ell+2} \to \mathbb{N}$ are in PRIM, then $f : \mathbb{N}^{\ell+1} \to \mathbb{N}$ given by

$$f(\vec{m}, 0) = g(\vec{m})$$
$$f(\vec{m}, n+1) = h(\vec{m}, n, f(\vec{m}, n))$$

is primitive recursive.

In other words, you can use for-loops.

Primitive Recursion

Primitive Recursion: If $g : \mathbb{N}^\ell \to \mathbb{N}$ and $h : \mathbb{N}^{\ell+2} \to \mathbb{N}$ are in PRIM, then $f : \mathbb{N}^{\ell+1} \to \mathbb{N}$ given by

$$f(\vec{m}, 0) = g(\vec{m})$$
$$f(\vec{m}, n+1) = h(\vec{m}, n, f(\vec{m}, n))$$

is primitive recursive.
In other words, you can use for-loops.

```
f(m, n):
  curr = g(m)
  for i in 1..n:
    curr = h(m, i-1, curr)
  return curr
```

This is very powerful!

Addition is in PRIM

The function $+ : \mathbb{N}^2 \to \mathbb{N}$, $f(m,n) = m + n$ is in PRIM, due to primitive recursion:

```
+(m, n):
  curr = C^1_1(m)
  for i in 1..n:
    curr = S(C^3_3(m, i-1, curr))
  return curr
```

[4] Unfortunately, we have to put some PRIM function in place of g, according to template...

Addition is in PRIM

The function $+ : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = m + n$ is in PRIM, due to primitive recursion:

```
+(m, n):
  curr = C^1_1(m)
  for i in 1..n:
    curr = S(C^3_3(m, i-1, curr))
  return curr
```

Explanation:

- C_1^1 is the identity function: $C_1^1(m) = m$. [4]
- `curr = S(C^3_3(m, i-1, curr))` adds 1 to `curr`:

$$C_3^3(m, i-1, \texttt{curr}) = \texttt{curr}$$

$$S(\texttt{curr}) = \texttt{curr} + 1$$

[4] Unfortunately, we have to put some PRIM function in place of g, according to template...

Addition is in PRIM

The function $+ : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = m + n$ is in PRIM, due to primitive recursion:

```
+(m, n):
  curr = C^1_1(m)
  for i in 1..n:
    curr = S(C^3_3(m, i-1, curr))
  return curr
```

Formally,

$$\begin{aligned} +(m, 0) &= C_1^1(\vec{m}) \\ +(m, n+1) &= S(C_3^3(m, n, +(m, n))). \end{aligned}$$

Multiplication is in PRIM

Your turn! Show that the function $f : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = m \cdot n$ is in PRIM.

Multiplication is in PRIM

Your turn! Show that the function $f : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = m \cdot n$ is in PRIM.
Here's the template for primitive recursion again:

```
f(m, n):
  curr = g(m)
  for i in 1..n:
    curr = h(m, i-1, curr)
  return curr
```

Hint: You know that the addition function $+ : \mathbb{N}^2 \to \mathbb{N}$, $+(m, n) = m + n$ is in PRIM now.

If you're finished, try showing $f : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = m^n$ is in PRIM too.

More PRIM functions!

Show that $f : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = m - n$ is in PRIM.

More PRIM functions!

~~Show that $f : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = m - n$ is in PRIM.~~

Unfortunately, the above function is not $\mathbb{N} \to \mathbb{N}$. You can't output negative numbers!

More PRIM functions!

~~Show that $f : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = m - n$ is in PRIM.~~

Unfortunately, the above function is not $\mathbb{N} \to \mathbb{N}$. You can't output negative numbers!

Try this instead. **Task:** Show that the "try to subtract 1" function

$$\delta : \mathbb{N} \to \mathbb{N}, \delta(m) = \begin{cases} m - 1 & m > 0 \\ 0 & m = 0 \end{cases}$$

is in PRIM.

More PRIM functions!

~~Show that $f : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = m - n$ is in PRIM.~~
Unfortunately, the above function is not $\mathbb{N} \to \mathbb{N}$. You can't output negative numbers!
Try this instead. **Task:** Show that the "try to subtract 1" function

$$\delta : \mathbb{N} \to \mathbb{N}, \delta(m) = \begin{cases} m - 1 & m > 0 \\ 0 & m = 0 \end{cases}$$

is in PRIM.

Hint 1: Try to prove $f : \mathbb{N}^2 \to \mathbb{N}$, $f(m, n) = \delta(n)$ is in PRIM.

```
f(m, n):
  curr = g(m)
  for i in 1..n:
    curr = h(m, i-1, curr)
  return curr
```

Hint 2: In the above, h won't need the value of m or `curr`.

More PRIM functions!

Task: Show that the “try to subtract” function

$$\dot{-} : \mathbb{N}^2 \to \mathbb{N}, \dot{-}(m, n) = \begin{cases} m - n & m \geq n \\ 0 & m < n \end{cases}$$

is in PRIM.

More PRIM functions!

Task: Show that the "try to subtract" function

$$\dot{-} : \mathbb{N}^2 \to \mathbb{N}, \dot{-}(m, n) = \begin{cases} m - n & m \geq n \\ 0 & m < n \end{cases}$$

is in PRIM.

```
f(m, n):
  curr = g(m)
  for i in 1..n:
    curr = h(m, i-1, curr)
  return curr
```

Hint: Use the "try to subtract 1" function δ.

More PRIM functions!

Task: Show that the "is greater than zero?" function

$$\mathrm{sg} : \mathbb{N} \to \mathbb{N}, \mathrm{sg}(n) = \begin{cases} 0 & n = 0 \\ 1 & n > 0 \end{cases}$$

is in PRIM.

```
f(m, n):
  curr = g(m)
  for i in 1..n:
    curr = h(m, i-1, curr)
  return curr
```

More PRIM functions!

Task: Show that the "is equal than zero?" function

$$\overline{\mathrm{sg}} : \mathbb{N} \to \mathbb{N}, \overline{\mathrm{sg}}(n) = \begin{cases} 1 & n = 0 \\ 0 & n > 0 \end{cases}$$

is in PRIM.
Hint: You could do this using primitive recursion, but a shorter solution would be to use sg.

Recap

Recap

We've shown that the following functions are in PRIM:

Recap

We've shown that the following functions are in PRIM:

- $+$,

Recap

We've shown that the following functions are in PRIM:

- $+$,
- $\cdot$,

Recap

We've shown that the following functions are in PRIM:

- $+$,
- $\cdot$,
- Exponentiation,

Recap

We've shown that the following functions are in PRIM:

- $+$,
- $\cdot$,
- Exponentiation,
- δ,

Recap

We've shown that the following functions are in PRIM:

- $+$,
- $\cdot$,
- Exponentiation,
- δ,
- $\dot{-}$,

Recap

We've shown that the following functions are in PRIM:

- $+$,
- $\cdot$,
- Exponentiation,
- δ,
- $\dot{-}$,
- sg, $\overline{\mathrm{sg}}$.

Recap

We've shown that the following functions are in PRIM:

- $+$,
- $\cdot$,
- Exponentiation,
- δ,
- $\dot{-}$,
- sg, $\overline{\mathrm{sg}}$.

Recap

We've shown that the following functions are in PRIM:

- $+$,
- $\cdot$,
- Exponentiation,
- δ,
- $\dot{-}$,
- sg, $\overline{\mathrm{sg}}$.

Note that sg, $\overline{\mathrm{sg}}$ allow you to use "if-statements".

Is PRIM everything?

Is PRIM everything?

Nope. Some functions, such as the Ackermann Function, are not in PRIM unfortunately.

Is PRIM everything?

Nope. Some functions, such as the Ackermann Function, are not in PRIM unfortunately.

```
Ackermann(m, n):
  if m = 0:
    return n + 1
  if n = 0:
    return Ackermann(m - 1, 1)
  return Ackermann(m-1, Ackermann(m, n-1))
```

Is PRIM everything?

Nope. Some functions, such as the Ackermann Function, are not in PRIM unfortunately.

```
Ackermann(m, n):
  if m = 0:
    return n + 1
  if n = 0:
    return Ackermann(m - 1, 1)
  return Ackermann(m-1, Ackermann(m, n-1))
```

m \ n	0	1	2	3	4	n
0	1	2	3	4	5	$n+1$
1	2	3	4	5	6	$n+2 = 2+(n+3)-3$
2	3	5	7	9	11	$2n+3 = 2\cdot(n+3)-3$
3	5	13	29	61	125	$2^{(n+3)}-3$
4	13 $= 2^{2^2}-3$ $= 2\uparrow\uparrow 3-3$	65533 $= 2^{2^{2^2}}-3$ $= 2\uparrow\uparrow 4-3$	$2^{65536}-3$ $= 2^{2^{2^{2^2}}}-3$ $= 2\uparrow\uparrow 5-3$	$2^{2^{65536}}-3$ $= 2^{2^{2^{2^{2^2}}}}-3$ $= 2\uparrow\uparrow 6-3$	$2^{2^{2^{65536}}}-3$ $= 2^{2^{2^{2^{2^{2^2}}}}}-3$ $= 2\uparrow\uparrow 7-3$	$\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{n+3}-3$ $= 2\uparrow\uparrow(n+3)-3$
5	65533 $= 2\uparrow\uparrow(2\uparrow\uparrow 2)-3$ $= 2\uparrow\uparrow\uparrow 3-3$	$2\uparrow\uparrow\uparrow 4-3$	$2\uparrow\uparrow\uparrow 5-3$	$2\uparrow\uparrow\uparrow 6-3$	$2\uparrow\uparrow\uparrow 7-3$	$2\uparrow\uparrow\uparrow(n+3)-3$
6	$2\uparrow\uparrow\uparrow\uparrow 3-3$	$2\uparrow\uparrow\uparrow\uparrow 4-3$	$2\uparrow\uparrow\uparrow\uparrow 5-3$	$2\uparrow\uparrow\uparrow\uparrow 6-3$	$2\uparrow\uparrow\uparrow\uparrow 7-3$	$2\uparrow\uparrow\uparrow\uparrow(n+3)-3$
m	$(2\to 3\to(m-2))-3$	$(2\to 4\to(m-2))-3$	$(2\to 5\to(m-2))-3$	$(2\to 6\to(m-2))-3$	$(2\to 7\to(m-2))-3$	$(2\to(n+3)\to(m-2))-3$

It grows too quickly to be captured using for-loops.

Is PRIM everything?

Is PRIM everything?

Nope. Some functions, such as the Ackermann Function, are not in PRIM unfortunately.

Is PRIM everything?

Nope. Some functions, such as the Ackermann Function, are not in PRIM unfortunately.

```
Ackermann(m, n):
  if m = 0:
    return n + 1
  if n = 0:
    return Ackermann(m - 1, 1)
  return Ackermann(m-1, Ackermann(m, n-1))
```

Is PRIM everything?

Nope. Some functions, such as the Ackermann Function, are not in PRIM unfortunately.

```
Ackermann(m, n):
  if m = 0:
    return n + 1
  if n = 0:
    return Ackermann(m - 1, 1)
  return Ackermann(m-1, Ackermann(m, n-1))
```

Proof (hard!): show that the set of functions

$$\mathcal{A} = \{f : \exists t \in \mathbb{N}, \forall x_1, \ldots, x_n \in \mathbb{N}, f(x_1, \ldots, x_n) < A(t, \max_i x_i)\}$$

contains all PRIM functions, via structural induction.

Is PRIM everything?

Nope. Some functions, such as the Ackermann Function, are not in PRIM unfortunately.

```
Ackermann(m, n):
  if m = 0:
    return n + 1
  if n = 0:
    return Ackermann(m - 1, 1)
  return Ackermann(m-1, Ackermann(m, n-1))
```

Proof (hard!): show that the set of functions

$$\mathcal{A} = \{f : \exists t \in \mathbb{N}, \forall x_1, \ldots, x_n \in \mathbb{N}, f(x_1, \ldots, x_n) < A(t, \max_i x_i)\}$$

contains all PRIM functions, via structural induction. This shows that $A(m, n)$ grows strictly faster than any PRIM function, and hence cannot be PRIM itself.

Is PRIM everything?

[5] In other words, you can't write a Turing machine (or equivalently a Python program) to compute the halting problem.

Is PRIM everything?

Nope.

[5] In other words, you can't write a Turing machine (or equivalently a Python program) to compute the halting problem.

Is PRIM everything?

Nope. Some other functions, such as the halting problem, are not even computable![5]

[5] In other words, you can't write a Turing machine (or equivalently a Python program) to compute the halting problem.

Is PRIM everything?

Nope. Some other functions, such as the halting problem, are not even computable![5]

To avoid spoiling content, I am legally required to not speak any further on the halting problem.

[5] In other words, you can't write a Turing machine (or equivalently a Python program) to compute the halting problem.