CSC363H5 Tutorial 3

I'm back!!! yay

Paul "sushi_enjoyer" Zhang

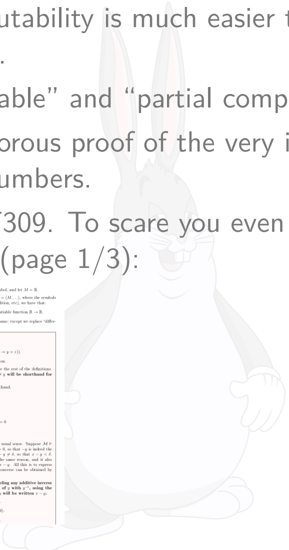
University of Chungi

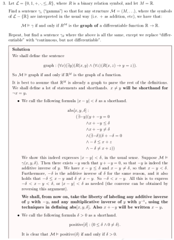
January 27, 2021

Learning objectives this tutorial

By the end of this tutorial, you should...

- Be fully convinced that Turing computability is much easier to understand than G*del computability.
- Have a list of synonyms for "computable" and "partial computable".
- Have a complete, mathematically-rigorous proof of the very intuitive fact that you can label things with numbers.
- Convince yourself to never take MAT309. To scare you even more, here's a proof I wrote in that course (page 1/3):





3. Let $\mathcal{L} = \{0, 1, +, \cdot, \leq, R\}$, where R is a binary relation symbol, and let $M = \mathbb{R}$.

Find a sentence γ_c ("gamma") so that for any structure $\mathcal{M} = (M, \ldots)$, where the symbols of $\mathcal{L} - \{\mathbb{R}\}$ are interpreted in the usual way (i.e. + as addition, etc), we have that:

$$\mathcal{M} \models \gamma \text{ if and only if } R^{\mathcal{M}} \text{ is the } \textbf{graph of} \text{ a differentiable function } \mathbb{R} \to \mathbb{R}.$$

Repeat, but find a sentence γ_d where the above is all the same, except we replace "differentiable" with "continuous, but not differentiable".

Solution

We shall define the sentence

$$\text{graph} : (\forall x)(\exists y)(R(x,y) \wedge (\forall z)(R(x,z) \to y = z)).$$

So $\mathcal{M} \models \text{graph}$ if and only if $\mathbb{R}^{M}$ is the graph of a function.

It is best to assume that $\mathbb{R}^{M}$ is already a graph to parse the rest of the definitions. We shall define a lot of statements and shorthands. $x \neq y$ **will be shorthand for** $\neg x = y$.

- We call the following formula $|x - y| < \delta$ as a shorthand.

$$\begin{aligned} &\text{abs}[x, y, \delta] : \\ &\quad (\exists -y)(y + -y = 0 \\ &\qquad \wedge x + -y \leq \delta \\ &\qquad \wedge x + -y \neq \delta \\ &\qquad \wedge (\exists -\delta)(\delta + -\delta = 0 \\ &\qquad \wedge -\delta \leq x + z \\ &\qquad \wedge -\delta \neq x + z)) \end{aligned}$$

  We show this indeed expresses $|x - y| < \delta$, in the usual sense. Suppose $\mathcal{M} \models \gamma(x, y, \delta)$. Then there exists $-y$ such that $y + -y = 0$, so that $-y$ is indeed the additive inverse of y. We have $x - y \leq \delta$ and $x - y \neq \delta$, so that $x - y < \delta$. Furthermore, $-\delta$ is the additive inverse of δ for the same reason, and it also holds that $-\delta \leq x - y$ and $\delta \neq x - y$. So $-\delta < x - y$. All this is to express $-\delta < x - y < \delta$, or $|x - y| < \delta$ as needed (the converse can be obtained by reversing this argument).

  We shall, from now on, take the liberty of labeling any additive inverse of y with $-y$, and any multiplicative inverse of y with y^{-1}, using the techniques in defining abs$[x, y, \delta]$. Also $x + -y$ will be written $x - y$.

- We call the following formula $\delta > 0$ as a shorthand.

$$\text{positive}[\delta] : (0 \leq \delta \wedge 0 \neq \delta).$$

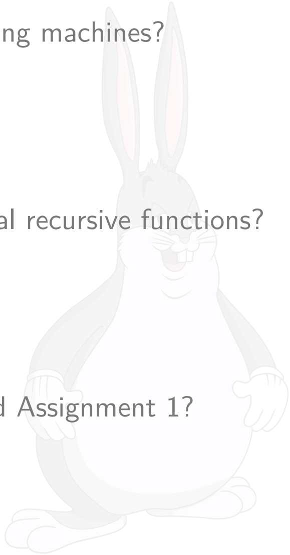
  It is clear that $\mathcal{M} \models \text{positive}(\delta)$ if and only if $\delta > 0$.

Quiz 2 is administered in this tutorial.[1]

Question 1 (1 point): Do you hate Turing machines?

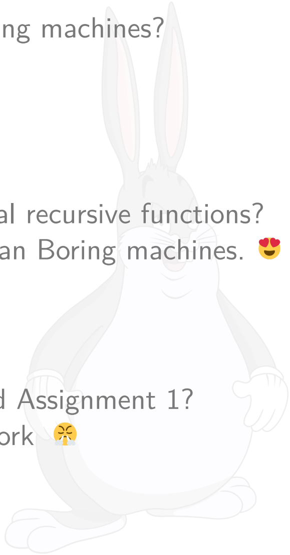
Question 2 (1 point): Do you like partial recursive functions?

Question 3 (1 point): Have you finished Assignment 1?



[1] no it isn't, but stay tuned!

Answer key



Question 1 (1 point): Do you hate Turing machines?
Answer: yep, i hate Turing machines! 🔪

Question 2 (1 point): Do you like partial recursive functions?
Answer: yes! they are so much better than Boring machines. 😍

Question 3 (1 point): Have you finished Assignment 1?
Answer: yes! i love doing csc363 homework 😤

let's review some words!

Task: List all synonyms of *computable* you have encountered so far in this course.
Task: List all synonyms of *partial computable* you have encountered so far in this course.

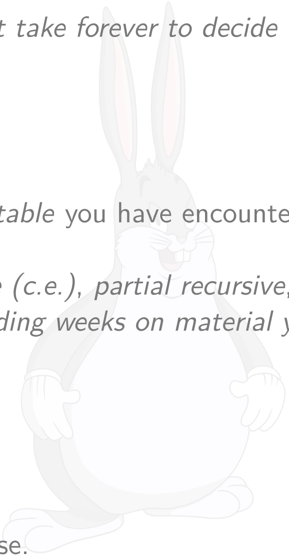
let's review some words!

Task: List all synonyms of *computable* you have encountered so far in this course.

Answer: *decidable*, *nice*, *not weird*, *won't take forever to decide whether something is in it or not*

Task: List all synonyms of *partial computable* you have encountered so far in this course.

Answer: *listable*, *computably enumerable (c.e.)*, *partial recursive*, *Diophantine*, *the reason why we are spending weeks on material you'll probably never see in a software job*



Note: *primitive recursive* is neither of those.

the reason why you're here today...

is to prove this one statement!

> *If $A \subseteq \mathbb{N}$ is an infinite computable set, then there exists an injective computable function $f : \mathbb{N} \to \mathbb{N}$ such that A is the range of f.*

- professor `helo_fish.jpg`, probably, 2021

Oh wait, `helo_fish.jpg` is back! she is no longer sad and feeling quite flushed right now.

helo_fish_flushed.jpg

mmm... idk, happy early valentines day i guess? ;-;
(btw, sowwy i couldn't hold tutorial last week!)

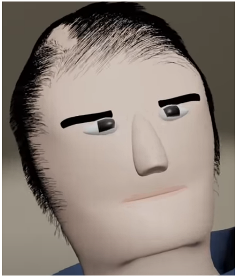
`helo_fish_flushed.jpg` wants to grant you one wish. Of course your wish is to know what an infinite computable set is! Say "`helo_fish_flushed.jpg`, what is an infinite computable set?"

helo_fish_flushed.jpg

bruh.

Okay, now helo_fish_flushed.jpg can go since she has granted your wish. Say goodbye to helo_fish_flushed.jpg!

Okay question time.

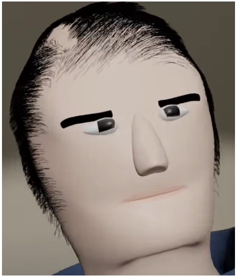


In fact, we only need partial computability:

If $A \subseteq \mathbb{N}$ is an infinite ~~computable~~ partial computable set, then there exists an injective computable function $f : \mathbb{N} \to \mathbb{N}$ such that A is the range of f.

Task: Prove this. (5 mins)

Okay question time.



In fact, we only need partial computability:

If $A \subseteq \mathbb{N}$ is an infinite ~~computable~~ partial computable set, then there exists an injective computable function $f : \mathbb{N} \to \mathbb{N}$ such that A is the range of f.

~~**Task:** Prove this. (5 mins)~~
I'll lead you through the proof instead, because again, Greek letters spook people.
Task: Read and understand the statement to keep in your head (1-2 min).

Okay question time.

Recall: if $A \subseteq \mathbb{N}$ is partial computable, then there exists a **computable** function $f : \mathbb{N} \to \mathbb{N}$ such that A is the range of f. (but f might not be injective!)

In other words,

$$A = \{f(0), f(1), f(2), \ldots\}$$

(but there may be repeats in the above list, as f might not be injective!)

Our task is to find an *injective* function $h : \mathbb{N} \to \mathbb{N}$ such that

$$A = \{h(0), h(1), h(2), \ldots\}$$

(the above list can't have repeats!)

How do we remove repeats intuitively?

Say A is the set of odd numbers, and f was some weird function that wanted to enumerate all the odd numbers, but really likes the number 69.

$$A = \{69, 1, 69, 3, 69, 5, 69, 7, \ldots\} = \{f(0), f(1), f(2), f(3), \ldots\}$$

Task: How would you make an injective function h that generates the same set, but without repeats? (Don't need you to be formal here, just describe what to do)

How do we remove repeats intuitively?

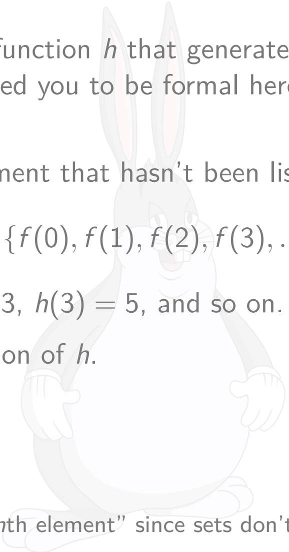
Task: How would you make an injective function h that generates the same set, but without repeats? (Don't need you to be formal here, just describe what to do)

Answer: Choose $h(n)$ to be the nth[2] element that hasn't been listed yet.

$$A = \{69, 1, 69, 3, 69, 5, 69, 7, \ldots\} = \{f(0), f(1), f(2), f(3), \ldots\}$$

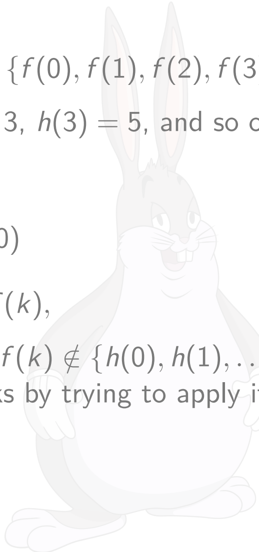
In this case, $h(0) = 69$, $h(1) = 1$, $h(2) = 3$, $h(3) = 5$, and so on.

Now we just have to formalize the definition of h.



[2] Technically A is a *set* and doesn't have an "nth element" since sets don't have an order. But we can order A like $f(0), f(1), \ldots$

How do we remove repeats intuitively?



$$A = \{69, 1, 69, 3, 69, 5, 69, 7, \ldots\} = \{f(0), f(1), f(2), f(3), \ldots\}$$

In this case, $h(0) = 69$, $h(1) = 1$, $h(2) = 3$, $h(3) = 5$, and so on.

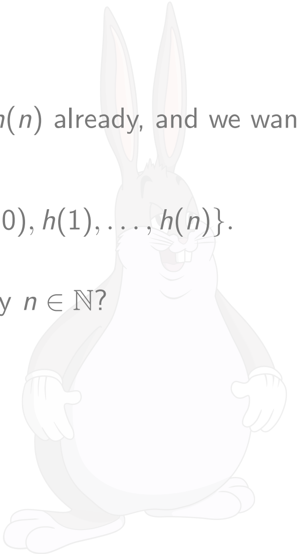
So to construct such an h, we have

$$h(0) = f(0)$$

$$h(n+1) = f(k),$$

where k is the minimal integer such that $f(k) \notin \{h(0), h(1), \ldots, h(n)\}$.
Task: Make sense of why the above works by trying to apply it on the example I gave.

Some building blocks first!



Suppose we have defined $h(0), h(1), \ldots, h(n)$ already, and we want to define $h(n+1)$. For $n \in \mathbb{N}$, Let

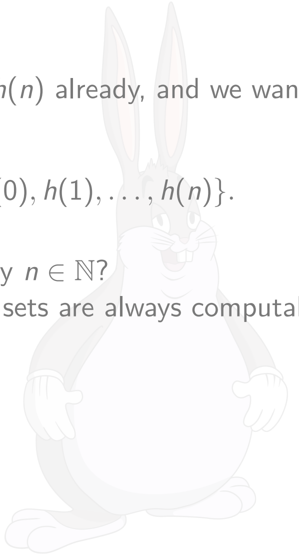
$$S_n = \{h(m) : m \leq n\} = \{h(0), h(1), \ldots, h(n)\}.$$

Task: Why is S_n a computable set for any $n \in \mathbb{N}$?

Some building blocks first!

Suppose we have defined $h(0), h(1), \ldots, h(n)$ already, and we want to define $h(n+1)$. For $n \in \mathbb{N}$, Let

$$S_n = \{h(m) : m \leq n\} = \{h(0), h(1), \ldots, h(n)\}.$$



Task: Why is S_n a computable set for any $n \in \mathbb{N}$?
Answer: S_n is finite for any n, and finite sets are always computable (according to professor Chungus).

Some building blocks first!

Suppose we have defined $h(0), h(1), \dots, h(n)$ already, and we want to define $h(n+1)$. For $n \in \mathbb{N}$, Let

$$S_n = \{h(m) : m \leq n\} = \{h(0), h(1), \dots, h(n)\}.$$

Task: Why is the following function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ computable? (Give a Turing machine argument)

$$g(n,k) = \begin{cases} 0 & f(k) \notin S_n \\ 1 & f(k) \in S_n \end{cases}$$

Some building blocks first!

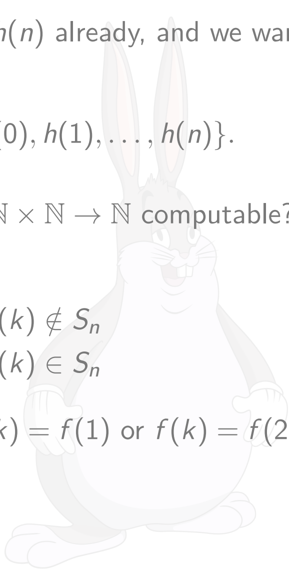
Suppose we have defined $h(0), h(1), \ldots, h(n)$ already, and we want to define $h(n+1)$. For $n \in \mathbb{N}$, Let

$$S_n = \{h(m) : m \leq n\} = \{h(0), h(1), \ldots, h(n)\}.$$

Task: Why is the following function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ computable? (Give a Turing machine argument)

$$g(n,k) = \begin{cases} 0 & f(k) \notin S_n \\ 1 & f(k) \in S_n \end{cases}$$

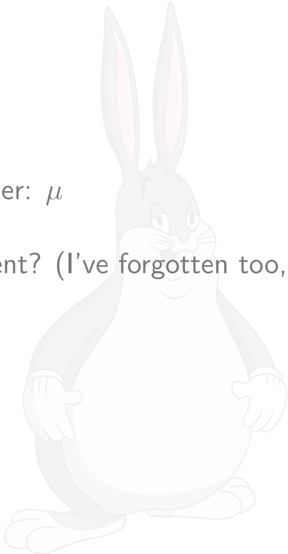
Answer: Just check if $f(k) = f(0)$ or $f(k) = f(1)$ or $f(k) = f(2)$, until $f(k) = f(n)$.



I hope you remember how to pronounce this Greek letter!

Task: Pronounce the following Greek letter: μ

Task: What does $\mu y[g(\overline{x}, y) = 0]$ represent? (I've forgotten too, dw)



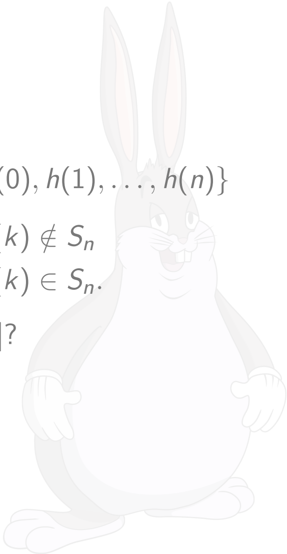
I hope you remember how to pronounce this Greek letter!

Task: Pronounce the following Greek letter: μ
Answer: μ

(i only remember μ's from love live school idol project lol)
(and no, i don't really like this anime)

Task: What does $\mu y[g(\overline{x}, y) = 0]$ represent? (I've forgotten too, dw)
Answer: $\mu y[g(\overline{x}, y) = 0]$ is the **minimum** $y \in \mathbb{N}$ such that $g(\overline{x}, y) = 0$.
(This minimum might not exist! in which case this is left undefined)

I hope you remember how to pronounce this Greek letter!



Recall:

$$S_n = \{h(m) : m \leq n\} = \{h(0), h(1), \ldots, h(n)\}$$

$$g(n,k) = \begin{cases} 0 & f(k) \notin S_n \\ 1 & f(k) \in S_n. \end{cases}$$

Task: (in words) What is $\mu k[g(n,k) = 0]$?

I hope you remember how to pronounce this Greek letter!

Recall:

$$S_n = \{h(m) : m \leq n\} = \{h(0), h(1), \ldots, h(n)\}$$

$$g(n,k) = \begin{cases} 0 & f(k) \notin S_n \\ 1 & f(k) \in S_n. \end{cases}$$

Task: (in words) What is $\mu k[g(n,k) = 0]$?
Answer: $\mu k[g(n,k) = 0]$ is the first $k \in \mathbb{N}$ such that $f(k) \notin S_n$.

But remember, we wanted to set $h(n+1) = f(k)$ where k is the first integer with $f(k) \notin S_n$! So we can let

$$h(n+1) = f(\mu k[g(n,k) = 0]).$$

We can formalize this now.

We have:

$$h(0) = f(0)$$

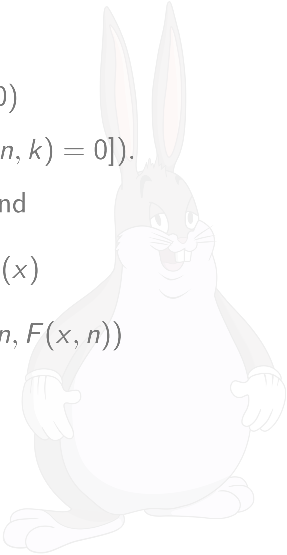
$$h(n+1) = f(\mu k[g(n,k) = 0]).$$

Recall: if f_1 and f_2 are partial recursive, and

$$F(x, 0) = f_1(x)$$

$$F(x, s(n)) = f_2(x, n, F(x, n))$$

then F is partial recursive.



We can formalize this now.

We have:

$$h(0) = f(0)$$

$$h(n+1) = f(\mu k[g(n,k) = 0]).$$

So if we let $f_1(x) = f(0)$ (it maps to the constant $f(0)$), and $f_2(x, n, F(x, n)) = f(\mu k[g(n,k) = 0])$, then F defined by

$$F(x, 0) = f_1(x) = f(0)$$

$$F(x, s(n)) = f_2(x, n, F(x, n)) = f(\mu k[g(n,k) = 0])$$

then F is partial recursive.

One last thing: set $h(n) = F(0, n)$ (and notice that F doesn't actually use x! it's absolutely useless.)

Task: Make sense of this.

yay we proved it! now what?

nothing. idk that's the only question i had to cover this tut, so 🗿
here's croissant sushi. bye! 🍣 🍣 🥐 🥐