Lessons touched by this meeting according to schedule:

- 10. 18/11/2024
  - Universal function: definition and computability [§5.1, Appendix of §5]
  - Computability of the inverse function, undecidability of the halting problem and of totality [§5.1]
- 11. 19/11/2024
  - Effective operations on computable functions. Exercises. [§5.3, §6.1.1, §6.1.3, §6.1.4, §6.1.6 with slightly different approach]

Consider the universal function:

Def: (universal function)

Given $k \geq 1$ the universal function of arity k is

$$\psi_U : \mathbb{N}^{k+1} \to \mathbb{N}$$

$$\psi_U(e, \vec{x}) = \varphi_e^{(k)}(\vec{x}) \qquad \text{well-defined}$$

We want to prove we can create a "universal interpreter" that can:

1. Take any program (by its code number e)
2. Take its inputs ($\bar{x}$)
3. Run that program on those inputs
4. Return whatever the original program would return

The proof works by showing we can:

1. Store program state (register contents)
2. Simulate program execution step by step
3. Track when the program finishes
4. Extract the final result

When you see $(\ldots)_1$:

- This means "extract the contents of register 1"
- Register 1 is where programs store their output by convention
- Think of it as "get the return value"

Examples on how to use it in exercises:

$$g : \mathbb{N}^3 \to \mathbb{N}$$

$$g(x, y, z) = \varphi_x(z) * \varphi_y(z)$$

$$= \psi_U(x, z) * \psi(y, z)$$

Let's focus already on the important part of this proof:

COROLLARY 12.3. *The following predicates are decidable:*

(a) $H_k(e, \vec{x}, t) \equiv$ *"$P_e(\vec{x}) \downarrow$ in t or less steps"*

(b) $S_k(e, \vec{x}, y, t) \equiv$ *"$P_e(\vec{x}) \downarrow y$ in t or less steps"*

PROOF. (a) The characteristic function

$$\chi_{H_k}(e, \vec{x}, t) = \begin{cases} 1 & \text{if } H_k(e, \vec{x}, t) \\ 0 & \text{otherwise} \end{cases}$$
$$= \overline{sg}(j_k(e, \vec{x}, t))$$

it is computable by composition.

(b) The characteristic function

$$\chi_{S_k}(e, \vec{x}, y, t) = \chi_{H_k}(e, \vec{x}, t) \cdot \overline{sg}(|(c_k(e, \vec{x}, t))_1 - y|)$$

it is computable by composition.

□

This works for k values; then, we parametrize such search on bounded terms to look for tuples inside of functions.

If $k = 1$ we will usually omit it.

Also, from the theorem we deduce the possibility to express every computable function in Kleene Normal Form (KNF).

COROLLARY 12.4 (Kleene Normal Form). *For every $e, k \in \mathbb{N}$ and $x \in \mathbb{N}^k$*

$$\varphi_e^{(k)}(x) = (\mu z \, . \, |\chi_{S_k}(e, \vec{x}, (z)_1, (z)_2) - 1|)_1$$

OBSERVATION 12.5.

i. This corollary highlights that each computable function can be obtained from primitive recursion functions using minimimalisation at most once (we need to use `while` statements, but one is sufficient).

ii. Minimixmalisation allows us to "search" a single value that has a certain property. The one we used is a technique to search pairs of values generalizable to tuples.

The letter Chi (strange X) means "Characteristic function", and it's used to characterize predicates:

DEFINITION 13.1. A set $A \subseteq \mathbb{N}$ is *recursive* if its characteristic function

$$\chi_A : \mathbb{N} \to \mathbb{N}$$
$$\chi_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

is computable.

EXERCISE 12.6. Let $f : \mathbb{N} \to \mathbb{N}$ computable and injective. Then $f^{-1} : \mathbb{N} \to \mathbb{N}$ is computable.

Focus on this proof – if f is not total, computability is not guaranteed, so we need a way to minimize couples of numbers, so to encode them as an integer number:

f is computable $\leadsto$ there exist $e \in \mathbb{N}$ program for f

$f = \varphi_e$

look for x input, m number of steps s.t. $\underbrace{P_e(x) \downarrow y \text{ in } t \text{ steps}}_{S(e, x, y, t)}$

$$= \text{"} \mu(x,t). \quad S(e,x,y,t) \text{"}$$

$$= \pi_1\left(\mu \underset{\substack{\wr \\ \omega = \pi(x,t)}}{\omega}. \quad S(e, \underbrace{\pi_1(\omega)}, y, \pi_2(\omega))\right)$$

more precisely

$$f^{-1}(y) = \underset{\uparrow}{\pi_1} \left(\mu\omega. \quad |\underset{\uparrow}{\chi_S}(e, \pi_1(\omega), y, \pi_2(\omega)) - 1|\right)$$

$$\omega \;\wr\; \boxed{(\omega)_1, (\omega)_2}, (\omega)_3, (\omega)_4 \; \ldots$$

$$f^{-1}(y) = \left(\mu\omega. \quad |\chi_S(e, \underset{\uparrow}{(\omega)_1}, y, \underset{\uparrow}{(\omega)_2}) - 1|\right)_1$$

Now, let's talk about the projection functions w_1 and w_2. These functions are used to extract the first and second components of a pair, respectively. Formally:

w_1(⟨x, y⟩) = x w_2(⟨x, y⟩) = y

In other words, given the encoding of a pair ⟨x, y⟩, w_1 returns the first element x, and w_2 returns the second element y.

Basically, they are used to map x, y as projection elements to transform a predicate into a mathematical expression (coding a couple as an integer). Consider this example which extends what was written before; basically, we use this encoding to replace x, y, t (example taken from exercise 8.26 – one of the very few to make us understand because the process is clearly written – would love it if was always like that):

$$\begin{aligned} sc_A(x) &= \mathbf{1}(\mu(y,z,t).H(x,y,t) \wedge S(x,z,y,t)) \\ &= \mathbf{1}(\mu w.H(x,(w)_1,(w)_3) \wedge S(x,(w)_2,(w)_1,(w)_3)) \end{aligned}$$

Another example to comment upon:

* Exercise: Let $Q(x)$ be a decidable predicate

$f_1, f_2 : \mathbb{N} \to \mathbb{N}$ computable

define

$$f(x) = \begin{cases} f_1(x) & \text{if } Q(x) \\ f_2(x) & \text{otherwise} \end{cases}$$

Then f is computable

proof

since f_1, f_2 are computable there are $e_1, e_2 \in \mathbb{N}$ s.t. $f_1 = \varphi_{e_1}$

$f_2 = \varphi_{e_2}$

$$f(x) \cancel{=} f_1(x) \cdot \chi_Q(x) + f_2(x) \cdot \chi_{\neg Q}(x)$$

$$f(x) \text{"}=\text{"} \Big(\mu(y,t).\ \big((S(e_1, x, y, t) \land Q(x)) \lor (S(e_2, x, y, t) \land \neg Q(x))\big)\Big)_y \text{"}$$

$$= \Big(\underset{\substack{\uparrow \\ (\omega)_1 = t \\ (\omega)_2 = y}}{\mu \omega}.\ \underbrace{\big((S(e_1, x, (\omega)_2, (\omega)_1) \land Q(x)) \lor (S(e_2, x, (\omega)_2, (\omega)_1) \land \neg Q(x))\big)}_{\text{decidable predicate}}\Big)_2$$

$\boxed{\text{computable}}$

where by

$\mu x.\ \underset{\uparrow \text{decidable}}{P(x)}$ we mean $\mu x.\ \underbrace{|\chi_P(x) - 1|}_{\text{computable}}$

Let's jump to exercises:

Exercise 6.22. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{if } \varphi_y(y) \downarrow \text{ for each } y \leqslant x \\ 0 & \text{otherwise} \end{cases}$$

Is it computable? Justify your answer.

We proceed by contradiction. Assume f is computable. Then ∃e. f = ϕe.

Let P(x) = "∀y≤x. ϕy(y)↓" be our condition. We can express P(x) formally using the halting predicate:

P(x) = ∏y≤x χH(y,y) where χH(y,y) = sg(μt. H(y,y,t))

Now consider $f(e)$:

Case 1: If $P(e)$ holds, then:

$f(e) = \phi e(e) + 1$ (by definition of f)

   $= f(e) + 1$ (since we assumed $f = \phi e$)

This implies $f(e) = f(e) + 1$, which is a contradiction.

Case 2: If $\neg P(e)$ holds, then:

$f(e) = 0$ (by definition of f)

$\phi e(e) = f(e) = 0$ (since we assumed $f = \phi e$)

But this means $\phi e(e)\downarrow$, contradicting $\neg P(e)$ which requires some $\phi y(y)\uparrow$ for $y \le e$.

Therefore, no computable function can match f's behavior. Thus $f \notin C$.

Exercise 6.13. Say if there is a total non-computable function $f : \mathbb{N} \to \mathbb{N}$ such that

$$f(x) \neq \varphi_x(x)$$

only on a single argument $x \in \mathbb{N}$. If the answer is negative provide a proof, if the answer is positive give an example of such a function.

Proof: We proceed by contradiction. Suppose there exists a total non-computable function $f : N \to N$ such that $f(x) \neq \phi x(x)$ for exactly one $x_0 \in N$.

Let us define a new function $g : N \to N$ as follows:

$g(x) = \{ \phi x(x) \text{ if } x = x_0 \; f(x) \text{ otherwise} \}$

Now let's establish that g is computable:

1. For $x = x_0$: $g(x_0) = \phi x_0(x_0)$ is computable by definition of ϕ
2. For $x \neq x_0$: $g(x) = f(x) = \phi x(x)$ since f differs from $\phi x(x)$ only at x_0

Therefore, g can be computed by first checking if $x = x_0$ (which is decidable), and then either computing $\phi x_0(x_0)$ or $\phi x(x)$ accordingly.

Since g is computable, there exists an index e such that $g = \phi e$.

But now consider:

- If $e = x_0$: then $g(e) = g(x_0) = \phi x_0(x_0) = \phi e(e)$
- If $e \neq x_0$: then $g(e) = f(e) = \phi e(e)$

In both cases, $g(e) = \phi e(e)$.

However, since $g = f$ except at x_0, and f differs from $\phi x(x)$ at exactly one point, g must also differ from $\phi x(x)$ at exactly one point. This contradicts the fact that $g(e) = \phi e(e)$.

Therefore, our initial assumption must be false, and no such function f can exist. ☐

Exercise (2020-11-23)

State the smn-theorem. Use it for proving there exists $k: \mathbb{N} \to \mathbb{N}$ total and computable s.t. $\forall n \in \mathbb{N}$ we have $|W_x| = 2^x$ and $|E_x| = x + 1$.

Solution

The smn-theorem states that, given $m, n \geq 1$ there is a computable total function $s_{m,n}: \mathbb{N}^{m+1} \to \mathbb{N}$ $s.t. \forall e \in \mathbb{N}, \vec{x} \in \mathbb{N}^m, \vec{y} \in \mathbb{N}^n$

$$\phi_e^{m+n}(\vec{x}, \vec{y}) = \phi_{s_{m,n}(e,\vec{x})}^{(n)}(\vec{y})$$

Define:

$$g(x, y) = \begin{cases} \lfloor \log_2(y+1) \rfloor & \text{se } y < 2^x \\ \uparrow & \text{altrimenti} \end{cases}$$

which is computable.

Infact, $g(x, y)$ when defined, is the greatest z s.t. $2^x \leq y + 1$ and the minimum s.t. $2^{z+1} > y + 1$, so:

$$g(x, y) = \mu z. \overline{sg}(2^{z+1} \dot{-} (y+1)) + \mu w.(y + 1 \dot{-} 2^x)$$

So, by the smn-theorem, there exists a function $s: \mathbb{N} \to \mathbb{N}$ s.t. $\forall x, y \in \mathbb{N}$ we have $g(x, y) = \phi_{s(x)}(y)$ and so s is the desired function. Infact:

- $W_x = \{y \mid g(x, y) \downarrow\} = [0, 2^x - 1]$ e quindi $|W_x| = |[0, 2^x - 1]| = 2^x$
- $E_x = \{g(x, y) \mid 0 \leqslant y < 2^x\} = \{\lfloor \log_2(y+1) \mid 0 \leqslant y < 2^x\} = [0, x]$ e quindi $|E_x| = |[0, x]| = x$.

In this exercise:

- Wx = {y | g(x,y)↓} = [0, 2x - 1] This means Wx contains all y values from 0 to 2x-1 where g(x,y) is defined Therefore |Wx| = 2x (it contains 2x elements)
- Ex = {g(x,y) | 0 ≤ y < 2x} = {⌊log2(y+1)⌋ | 0 ≤ y < 2x} = [0,x] This is the set of values that g(x,y) outputs when y is in the domain Therefore |Ex| = x+1 (it contains x+1 elements)

To understand why:

- For Wx: g(x,y) is defined when y < 2x (from the definition) So Wx contains all numbers from 0 to 2x-1
- For Ex: When y runs from 0 to 2x-1:
    - log2(y+1) runs from log2(1) to log2(2x)
    - Taking the floor ⌊log2(y+1)⌋ gives us all integers from 0 to x
    - Hence Ex = [0,x]

Exercise 2

State the s-m-n theorem and use it to prove that there exists a total computable function $s : \mathbb{N} \to \mathbb{N}$ such that $|W_{s(x)} \cap E_{s(x)}| = 2x$.

The smn-theorem states that, given $m, n \geq 1$ there is a computable total function $s_{m,n}: \mathbb{N}^{m+1} \to \mathbb{N}$ $s.t. \forall e \in \mathbb{N}, \vec{x} \in \mathbb{N}^m, \vec{y} \in \mathbb{N}^n$

$$\phi_e^{m+n}(\vec{x}, \vec{y}) = \phi_{s_{m,n}(e,\vec{x})}^{(n)}(\vec{y})$$

By the s-m-n theorem, it suffices to define a computable function f : N^2 → N such that for fixed x, the function λy.f(x,y) has the desired properties, i.e.

W_{λy.f(x,y)} = {z | f(x,z) ↓} has cardinality 2x and

E_{λy.f(x,y)} = {f(x,z) | f(x,z) ↓} also has cardinality 2x.

We can define f as follows:

f(x,y) =

⌊y/2⌋ mod 2x   if y is even

↑   otherwise

Equivalently, using functions we know to be primitive recursive:

f(x,y) = [qt(2,y) mod 2x] + μz.[rm(2,y)]

Intuitively, for a fixed x, f is defined only on even numbers y and its value cycles through {0,1,...,2x-1}.

More formally, fixing x, we have:

- W_{λy.f(x,y)} = {y | y is even} which has cardinality 2x

- E_{λy.f(x,y)} = {f(x,y) | y is even} = {0,1,...,2x-1} which also has cardinality 2x

Since f is computable (in fact, primitive recursive), by the s-m-n theorem there exists a total computable function s : N → N such that ϕ_s(x)(y) = f(x,y) for all x,y ∈ N.

Therefore, s satisfies the desired properties:

- W_s(x) = {y | f(x,y) ↓} has cardinality 2x

- E_s(x) = {f(x,y) | f(x,y) ↓} also has cardinality 2x

So |W_s(x) ∩ E_s(x)| = 2x for all x ∈ N, as required.

Exercise (2019-02-08)

Given a function $f: \mathbb{N} \to \mathbb{N}$ define $Z(f) = \{g: \mathbb{N} \to \mathbb{N} \mid \forall x \in \mathbb{N}. g(x) = f(x) \vee g(x) = 0\}$. Show that set $Z(id)$ is not countable. It is true that for all f, we have $Z(f)$ is not countable?

For Z(id), we can prove it's uncountable by diagonalization:

Suppose by contradiction that Z(id) is countable. Then there exists an enumeration of functions in Z(id): g_0, g_1, g_2, ...

Define a new function h: ℕ → ℕ as: h(x) = { 0 if $g_x(x)$ = x x if $g_x(x)$ = 0 }

Now observe that:

- h is well-defined since for any x, $g_x(x)$ must be either x or 0 (by definition of Z(id))
- h ∈ Z(id) since for each x, h(x) is either x or 0
- But h differs from every function in the enumeration: For any n, h(n) ≠ $g_n(n)$ by construction

This contradicts our assumption that Z(id) is countable.

2. Using ϕx notation, we could also approach it this way:

Define F: ℕ → ℕ as: F(x) = { 0 if ϕx(x) = x x if ϕx(x) ≠ x or ϕx(x)↑ }

Then F ∈ Z(id) but differs from every computable function in Z(id). Since there are uncountably many such functions, Z(id) must be uncountable.

Exercise (2015-04-20.partial)

State the smn-theorem and use it to show there exists a total computable function $s: \mathbb{N} \to \mathbb{N}$ s.t. $\forall x \in \mathbb{N}$, $W_{s(x)} = \{(k+2)^2 \mid k \in \mathbb{N}\}$

Solution

The smn-theorem states that, given $m, n \geq 1$ there is a computable total function $s_{m,n}: \mathbb{N}^{m+1} \to \mathbb{N}$ $s.t. \forall e \in \mathbb{N}, \vec{x} \in \mathbb{N}^m, \vec{y} \in \mathbb{N}^n$

$$\phi_e^{m+n}(\vec{x}, \vec{y}) = \phi^{(n)}_{s_{m,n}(e,\vec{x})}(\vec{y})$$

To prove it, we define a function of two arguments such that:

$$g(x,y) = \begin{cases} k, & \textit{if there exists some } k \ s.t. \ y = (x+k)^2 \\ \uparrow, & \textit{otherwise} \end{cases}$$

so we set a minimalization to look for that value, like $g(x,y) = \mu k. |(x+k)^2 - y|$. Such function is total and computable, and for the smn-theorem, there exists a function $k: \mathbb{N} \to \mathbb{N}$ s.t. $\phi_{s(x)}(y) = g(x,y)\ \forall x, y \in \mathbb{N}$. So, as desired:

- $W_{s(x)} = \{x \mid g(x,y) \downarrow\} = \{\exists k \in \mathbb{N} \mid y = (x+k)^2\} = \{x \mid (x+k)^2 \in \mathbb{N}\}$

Present to make everyone understand meaning and notations:

Exercise 6.32. Let A be a recursive set and let $f_1, f_2 : \mathbb{N} \to \mathbb{N}$ be computable functions. Prove that the function $f : \mathbb{N} \to \mathbb{N}$ defined below is computable:

$$f(x) = \begin{cases} f_1(x) & \text{if } x \in A \\ f_2(x) & \text{if } x \notin A \end{cases}$$

Does the result hold if we weaken the hypotheses and assume A only r.e.? Explain how the proof can be adapted, if the answer is positive, or provide a counterexample, otherwise.

Solution: Let $e_1, e_2 \in \mathbb{N}$ be indexes for f_1, f_2, respectively, namely $\varphi_{e_1} = f_1$ and $\varphi_{e_2} = f_2$. Observe that we can define f as

$$f(x) = (\mu w.((S(e_1, x, (w)_1, (w)_2) \ \wedge \ \chi_A(x) = 1) \vee (S(e_2, x, (w)_1, (w)_2) \ \wedge \ \chi_A(x) = 0)))_1$$

showing that f is computable. Relaxing the hypotheses to recursive enumerability of A, the result is no longer true. Consider for instance $f_1(x) = 1$, $f_2(x) = 0$ and $A = K$, which is r.e. Then f defined as above would be the characteristic function of K which is not computable. □

Link from some primitive recursive exercises:
https://proofwiki.org/wiki/Category:Primitive_Recursive_Functions

Exercise: Define the class PR of primitive recursive functions and, using only the definition, prove that the function pmax : N^2 → N, defined by pmax(x,y) = max(2^x, 3^y), is primitive recursive.

Solution: The class PR of primitive recursive functions is the smallest class of functions that contains the basic functions:

1. Zero function: z(x) = 0 for each x ∈ N;

2. Successor function: s(x) = x + 1 for each x ∈ N;

3. Projection functions: U^k_j(x_1, ..., x_k) = x_j for each (x_1, ..., x_k) ∈ N^k and 1 ≤ j ≤ k.

and is closed under the following operations:

1. Composition: If f_1, ..., f_n : N^k → N and g : N^n → N are in PR, then the function h : N^k → N defined by h(x̄) = g(f_1(x̄), ..., f_n(x̄)) is also in PR.

2. Primitive Recursion: If f : N^k → N and g : N^(k+2) → N are in PR, then the function h : N^(k+1) → N defined by:

h(x̄, 0) = f(x̄)

h(x̄, y+1) = g(x̄, y, h(x̄, y))

is also in PR.

To show that pmax(x,y) is in PR, we can build it up from simpler functions in PR:

1. The exponentiation functions exp_2(x) = 2^x and exp_3(y) = 3^y can be defined by primitive recursion:

exp_2(0) = 1

exp_2(x+1) = 2 · exp_2(x)

exp_3(0) = 1

exp_3(y+1) = 3 · exp_3(y)

2. The maximum function max(x,y) can also be defined by primitive recursion:

max(x,0) = x

max(x,y+1) = max(s(x), y)

3. Finally, pmax(x,y) can be defined by composition:

pmax(x,y) = max(exp_2(x), exp_3(y))

Since exp_2, exp_3, and max are all in PR, and PR is closed under composition, we conclude that pmax is also in PR.

<u>Exercise</u>: Define the class PR of primitive recursive functions and, using only the definition, prove that the function f : N → N, defined by f(x) = x^2 + 2x, is primitive recursive.

<u>Solution</u>: (Definition given above)

To show that f(x) = x^2 + 2x is in PR, we can build it up from simpler functions in PR:

1. The square function sq(x) = x^2 can be defined by primitive recursion: sq(0) = 0 sq(x+1) = sq(x) + 2x + 1
2. The multiplication function mult(x,y) = xy can also be defined by primitive recursion: mult(x,0) = 0 mult(x,y+1) = mult(x,y) + x
3. The function 2x can be defined by composition of the multiplication function and the constant function 2: 2x = mult(2,x)
4. Finally, f(x) can be defined by composition of sq, mult, and addition: f(x) = sq(x) + mult(2,x)

Since sq, mult, and the constant function 2 are all in PR, and PR is closed under composition and addition (which can be defined by primitive recursion), we conclude that f is also in PR.