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. $\underbrace{P_e(x) \downarrow y \text{ in } t \text{ steps}}_{S(e, x, y, t)}$

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

$$= \pi_1\Big(\mu\, \omega.\ S(e, \underbrace{\pi_1(\omega)}, y, \pi_2(\omega))\Big)$$

$\omega = \pi(x, t)$

more precisely

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

ω

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

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

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. $f_1 = \varphi_{e_1}$
$f_2 = \varphi_{e_2}$

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

$$f(x) \overset{''''}{=} \Big(\mu(y,t). \big((S(e_1, x, y, t) \wedge Q(x)) \vee (S(e_2, x, y, t) \wedge \neg Q(x))\big)\Big)_y \ ''''$$

$$= \Big(\mu w . \big(\underbrace{(S(e_1, x, (w)_2, (w)_1) \wedge Q(x)) \vee (S(e_2, x, (w)_2, (w)_1) \wedge \neg Q(x))}_{\text{decidable predicate}}\big)\Big)_2$$

$(w)_1 = t$
$(w)_2 = y$

$\boxed{\text{computable}}$

where by
$\mu x . \underset{\text{decidable}}{P(x)}$ 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 $\exists e.\ f = \phi e$.

Let $P(x) = $ "$\forall y \leq x.\ \phi y(y)\downarrow$" be our condition. We can express P(x) formally using the halting predicate:

$P(x) = \prod y \leq x\ \chi H(y,y)$ where $\chi H(y,y) = sg(\mu t.\ H(y,y,t))$

Now consider f(e):

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

f(e) = ϕe(e) + 1   (by definition of f)

= f(e) + 1   (since we assumed f = ϕe)

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

Case 2: If ¬P(e) holds, then:

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

ϕe(e) = f(e) = 0   (since we assumed f = ϕe)

But this means ϕe(e)↓, contradicting ¬P(e) which requires some ϕy(y)↑ for y≤e.

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 $s_{m,n}: \mathbb{N}^{m+1} \to \mathbb{N}$ $s.t. \forall e \in \mathbb{N}, \vec{x} \in \mathbb{N}^m, \vec{y} \in \mathbb{N}^n$

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

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

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

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 : $N^2 \to N$, 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 $x \in N$;

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

3. Projection functions: $U^k_j(x_1, \ldots, x_k) = x_j$ for each $(x_1, \ldots, x_k) \in N^k$ and $1 \leq j \leq k$.

and is closed under the following operations:

1. Composition: If $f_1, \ldots, f_n : N^k \to N$ and $g : N^n \to N$ are in PR, then the function $h : N^k \to N$ defined by $h(\bar{x}) = g(f_1(\bar{x}), \ldots, f_n(\bar{x}))$ is also in PR.

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

$h(\bar{x}, 0) = f(\bar{x})$

$h(\bar{x}, y+1) = g(\bar{x}, y, h(\bar{x}, y))$

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:

$exp_2(0) = 1$

$exp_2(x+1) = 2 \cdot exp_2(x)$

$exp_3(0) = 1$

$exp_3(y+1) = 3 \cdot exp_3(y)$

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.

Exercise 6.5(p). Say that a function $f : \mathbb{N} \to \mathbb{N}$ is *decreasing* if it is total and for each $x, y \in \mathbb{N}$, if $x \leqslant y$ then $f(x) \geqslant f(y)$. Is there a decreasing function which is not computable? Justify your answer.

Solution: Let $k = \min\{f(x) \mid x \in \mathbb{N}\}$ and let $x_0 \in \mathbb{N}$ be such that $f(x_0) = k$. Therefore, since f is decreasing, $f(x) = k$ for all $x \geqslant x_0$. If we define

$$\theta(x) = \begin{cases} f(x) & \text{if } x < x_0 \\ \uparrow & \text{otherwise} \end{cases}$$

we can write f as

$$f(x) = \begin{cases} \theta(x) & \text{if } x < x_0 \\ k & \text{otherwise} \end{cases}$$

Since θ is finite, it is computable. Let $\theta = \varphi_e$. Therefore

$$f(x) = (\mu w. ((x < x_0 \ \wedge \ S(e, x, (w)_1, (w)_2) \vee (x \geqslant x_0 \ \wedge \ (w)_1 = k)))_1$$

hence it is computable.