COMPUTABILITY (18/12/2023)

2nd RECURSION THEOREM

* f is the transforming function

let $f : \mathbb{N} \to \mathbb{N}$ computable total extensional

[extensional in a sense that if two programs compute the same function, also the two transformed programs compute again the same function]

$$P \longrightarrow \boxed{f} \longrightarrow P'$$

$\forall e, e' \in \mathbb{N} \quad \varphi_e = \varphi_{e'}$

$\leadsto \quad \varphi_{f(e)} = \varphi_{f(e')}$

by Myhill - Shepherdson's theorem there a (unique)

recursive functional $\Phi : \mathcal{F}(\mathbb{N}) \to \mathcal{F}(\mathbb{N})$

$\forall e \in \mathbb{N} \quad \Phi(\varphi_e) = \varphi_{f(e)}$

By 1st recursion theorem Φ has a least fixed point $f_\Phi : \mathbb{N} \to \mathbb{N}$

computable

$$\begin{cases} \Phi(f_\Phi) = f_\Phi \\ \exists e_0 \in \mathbb{N} \text{ s.t. } f_\Phi = \varphi_{e_0} \end{cases}$$

$$\varphi_{e_0} = f_\Phi = \Phi(f_\Phi) = \Phi(\varphi_{e_0}) = \varphi_{f(e_0)}$$

(this means the function e0 and the function computed by the transformed function f(e0) is the same)

In summary

Given $f : \mathbb{N} \to \mathbb{N}$ computable total ~~extensional~~

there is $e_0 \in \mathbb{N}$ s.t. $\varphi_{e_0} = \varphi_{f(e_0)}$

2nd recursion theorem
(this theorem excludes this condition*)

* we are saying that we can apply any type of trasformation to the program and the function computed by the program before and after the transformation is always the same ==> more powerful than the 1 recursion theorem

2^{nd} RECURSION THEOREM

Let $f: \mathbb{N} \to \mathbb{N}$ be total computable function

Then there exists $e_0 \in \mathbb{N}$ s.t. $\varphi_{e_0} = \varphi_{f(e_0)}$

Before and after the transformation the program (which changes) is computing the same function

proof

let $f: \mathbb{N} \to \mathbb{N}$ be total computable

observe $x \mapsto \varphi_x(x) = \Psi_U(x,x)$ computable

$x \mapsto f(\varphi_x(x))$ computable

define

$$g(x,y) = \varphi_{f(\varphi_x(x))}(y)$$

convention $\varphi_\uparrow = \uparrow$

(when the index is undefined also the function is undefined)

$$= \Psi_U(f(\varphi_x(x)), y)$$

$$= \Psi_U(f(\Psi_U(x,x)), y) \quad \text{computable}$$

By smn theorem there $s: \mathbb{N} \to \mathbb{N}$ total computable s.t. $\forall x,y$

$$\varphi_{s(x)}(y) = g(x,y) = \varphi_{f(\varphi_x(x))}(y) \quad (*)$$

Since s is computable there is $m \in \mathbb{N}$ s.t. $s = \varphi_m$.

Substituting in $(*)$

$$\varphi_{\varphi_m(x)}(y) = \varphi_{f(\varphi_x(x))}(y) \quad \forall x,y$$

In particular, for $x = m$

$$\varphi_{\varphi_m(m)}(y) = \varphi_{f(\varphi_m(m))}(y) \quad \forall y$$

the functions are the same for all arguments

Hence

$$\varphi_{\varphi_m(m)} = \varphi_{f(\varphi_m(m))}$$

If we let $e_0 = \varphi_m(m)$ we conclude

$$\varphi_{e_0} = \varphi_{f(e_0)}$$

(note that $\varphi_m =_s$ total, hence $\varphi_m(m)\downarrow$) $\square$

<u>Idea</u> :

if $h : \mathbb{N} \to \mathbb{N}$ computable

$$\begin{array}{lccccc} & \varphi_0 & \varphi_1 & \varphi_2 & \varphi_3 & \cdots\cdots \\ h \rightsquigarrow & & & & & \\ & \varphi_{h(0)} & \varphi_{h(1)} & \varphi_{h(2)} & \varphi_{h(3)} & \cdots\cdots \end{array}$$

you can do the above for $h = \varphi_i \qquad i = 0, 1, 2, \ldots$

$$\begin{array}{lcccc} E_0 & \varphi_{\varphi_0(0)} & \varphi_{\varphi_0(1)} & \varphi_{\varphi_0(2)} & \cdots\cdots \\ E_1 & \varphi_{\varphi_1(0)} & \varphi_{\varphi_1(1)} & \varphi_{\varphi_1(2)} & \cdots\cdots \\ E_2 & \varphi_{\varphi_2(0)} & \varphi_{\varphi_2(1)} & \varphi_{\varphi_2(2)} & \cdots\cdots \end{array}$$

$h(x) = \varphi_x(x)$

in the proof we took the diagonal transformed by f

$$h(x) = f(\varphi_x(x)) = f(\psi_U(x,x)) = \varphi_m(x)$$

$$\begin{array}{lcccc} E_0 & \varphi_{\varphi_0(0)} & \varphi_{\varphi_0(1)} & \varphi_{\varphi_0(2)} & \cdots\cdots \\ E_1 & \varphi_{\varphi_1(0)} & \varphi_{\varphi_1(1)} & \varphi_{\varphi_1(2)} & \cdots\cdots \\ E_2 & \varphi_{\varphi_2(0)} & \varphi_{\varphi_2(1)} & \varphi_{\varphi_2(2)} & \cdots\cdots \\ & & & & \\ E_m & \varphi_{f(\varphi_0(0))} & \varphi_{f(\varphi_1(1))} & \varphi_{f(\varphi_2(2))} & \end{array}$$

$\varphi_x(x)$

$\varphi_{\varphi_m(m)}$

$\parallel$

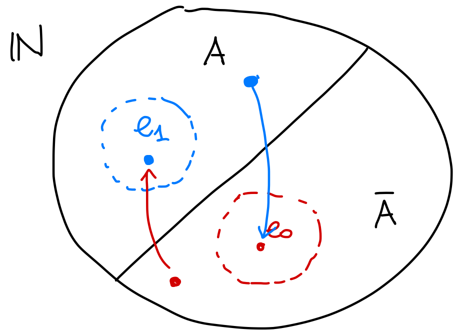
$\varphi_{f(\varphi_m(m))}$

Rice's Theorem

Let $A \subseteq \mathbb{N}$ saturated $A \neq \emptyset$, $A \neq \mathbb{N}$ then A not recursive

proof (alternative proof using 2nd recursion theorem)

Let $A \subseteq \mathbb{N}$ $A \neq \emptyset$, $A \neq \mathbb{N}$ saturated



$A \neq \emptyset \rightsquigarrow \exists e_1 \in A$

$A \neq \mathbb{N} \rightsquigarrow \exists e_0 \in \bar{A}$

Assume by contradiction A recursive and defined:

$f : \mathbb{N} \to \mathbb{N}$

$$f(x) = \begin{cases} e_0 & \text{if } x \in A \\ e_1 & \text{if } x \notin A \end{cases}$$

$$= e_0 \cdot \chi_A(x) + e_1 \cdot \chi_{\bar{A}}(x)$$

$$\left(\begin{array}{llll} \text{if } x \in A \rightsquigarrow & \chi_A(x) = 1 & \chi_{\bar{A}}(x) = 0 & e_0 \cdot 1 + e_1 \cdot 0 = e_0 \\ \text{if } x \notin A \rightsquigarrow & \chi_A(x) = 0 & \chi_{\bar{A}}(x) = 1 & e_0 \cdot 0 + e_1 \cdot 1 = e_1 \end{array} \right)$$

if A recursive, f computable total

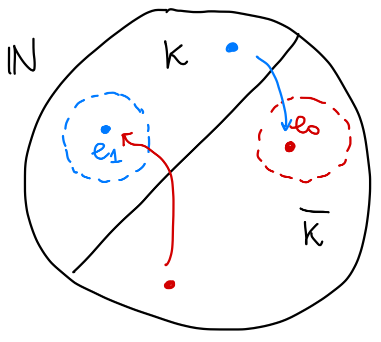
but for all $e \in \mathbb{N}$ $\varphi_e \neq \varphi_{f(e)}$

- $e \in A \Rightarrow f(e) = e_0 \notin A$ and since A saturated $\varphi_e \neq \varphi_{f(e)}$
- $e \notin A \Rightarrow f(e) = e_1 \in A$ thus, since A saturated $\varphi_e \neq \varphi_{f(e)}$

This contradicts the 2nd recursion theorem. $\rightsquigarrow$ A not recursive $\square$

Proposition: The halting set $K = \{ x \in \mathbb{N} \mid \varphi_x(x)\downarrow \}$ is not recursive.

proof (alternative proof using 2nd recursion theorem)



if $e_0 \in \mathbb{N}$ s.t. $\varphi_{e_0}(x)\uparrow \ \forall x$

we have $e_0 \in \overline{K}$

if $e_1 \in \mathbb{N}$ s.t. $\varphi_{e_1}(x) = 1 \ \forall x$

we have $e_1 \in K$

define $f : \mathbb{N} \to \mathbb{N}$ s.t.

$$f(x) = \begin{cases} e_0 & \text{if } x \in K \\ e_1 & \text{if } x \notin K \end{cases}$$

$$= e_0 \cdot \chi_K(x) + e_1 \cdot \chi_{\overline{K}}(x)$$

if K were recursive then $\chi_K, \chi_{\overline{K}}$ would be computable and thus f would be computable.

Since f is total, by 2nd recursion theorem there is $e \in \mathbb{N}$ s.t $\varphi_e = \varphi_{f(e)}$

$\to$ $e \in K$ $\leadsto$ $f(e) = e_0$ $\qquad \varphi_e(e)\downarrow \neq \varphi_{f(e)}(e) = \varphi_{e_0}(e)\uparrow$

$\to$ $e \notin K$ $\leadsto$ $f(e) = e_1$ $\qquad \varphi_e(e)\uparrow \neq \varphi_{f(e)}(e) = \varphi_{e_1}(e) = 1\downarrow$

contradiction !

Hence K is not recursive. $\square$

* K is not saturated

$K = \{ x \in \mathbb{N} \mid \varphi_x(x)\downarrow \}$

We want to show that there are $e, e' \in \mathbb{N}$ s.t.

$$\boxed{\begin{array}{ll} \varphi_e = \varphi_{e'} & \\ e \in K & e' \notin K \end{array}}$$

* Assume that there is $e \in \mathbb{N}$ s.t.

$$\varphi_e(x) = \begin{cases} 0 & \text{if } x = e \\ \uparrow & \text{otherwise} \end{cases}$$

then

- $e \in K$ since $\varphi_e(e) = 0 \downarrow$
- there $e' \in \mathbb{N}$ $e' \neq e$ s.t. $\varphi_e = \varphi_{e'}$
- $e' \notin K$ $\varphi_{e'}(e') = \varphi_e(e') \uparrow$ $e \neq e'$

* We need to show that there exists $e \in \mathbb{N}$ s.t.

$$\varphi_e(x) = \begin{cases} 0 & \text{if } x = e \\ \uparrow & \text{otherwise} \end{cases}$$

intuition

kleeny.py

```
def P (x) :
    if x = "                    "
        then return 0
    else loop
```

program we are defining

read (" kleene.py ")

formally

$$g(e, x) = \begin{cases} 0 & \text{if } x = e \\ \uparrow & \text{otherwise} \end{cases}$$

$= \mu z. \ |x-e|$ computable

by smn theorem there is $s: \mathbb{N} \to \mathbb{N}$ total computable s.t.

$$\varphi_{s(e)}(x) = g(e,x) = \begin{cases} 0 & \text{if } x = e \\ \uparrow & \text{otherwise} \end{cases}$$

since s is total computable, by 2nd recursion theorem there is $e_0 \in \mathbb{N}$ s.t. $\varphi_{e_0} = \varphi_{s(e_0)}$. Hence

$$\varphi_{e_0}(x) = \varphi_{s(e_0)}(x) = g(e_0, x) = \begin{cases} 0 & \text{if } x = e_0 \\ \uparrow & \text{otherwise} \end{cases}$$

Hence e_0 is the desired program. Thus K not saturated.

EXERCISE: RANDOM NUMBERS (from 1st lesson)

→ $n \in \mathbb{N}$ is random if all programs producing n in output are "larger" than n

two questions:

→ there are infinitely many random numbers

→ the property of being random is not decidable

Try again:

→ size of a program? $|P_e| = e$

→ define $n \in \mathbb{N}$ random if

for all $e \in \mathbb{N}$ s.t. $\varphi_e(0) = n$ it holds $e > n$

EXERCISE :

Let $f: \mathbb{N} \to \mathbb{N}$ be a function

and consider

$$B_f = \{ e \in \mathbb{N} \mid \varphi_e = f \}$$

Are B_f, $\overline{B_f}$ recursive / r.e. ?

(1) f not computable

$B_f = \emptyset$, $\overline{B_f} = \mathbb{N}$ recursive (hence r.e.)

(2) f computable

B_f is saturated

$B_f \neq \emptyset \quad B_f \neq \mathbb{N}$ $\overset{\text{Rice}}{\Longrightarrow}$ B_f not recursive

$\overline{B_f}$ not recursive

can it be r.e. ? yes it can!

$f = \emptyset \qquad (f(x) \uparrow \ \forall x)$

$$\overline{B_f} = \{ e \mid \varphi_e \neq \emptyset \}$$

$$= \{ e \mid \underbrace{\exists x. \ \underbrace{\varphi_e(x)\downarrow}} \}$$

$$sc_{\overline{B_f}}(x) = \mathbb{1}(\mu w. \ H(x, (w)_1, (w)_2))$$

complete the exercise!