COMPUTABILITY (24/10/2023)

basic functions $\cap$ $\mathcal{C}$ — composition, primitive recursion $\rightsquigarrow$ total functions

* Unbounded minimalisation

Given $f : \mathbb{N}^{k+1} \to \mathbb{N}$ (not necessarily total)

$f(\vec{x}, y)$

define $h : \mathbb{N}^k \to \mathbb{N}$

$$h(\vec{x}) = \mu y.\ f(\vec{x}, y) = \text{least } y \text{ s.t. } f(\vec{x}, y) = 0$$

- $\rightarrow$ such y could not exist
- $\rightarrow$ $f(\vec{x}, z)$ could be undefined before finding y ...

$\uparrow$ (undefined)

$$= \begin{cases} y & \text{if there is } y \text{ s.t. } f(\vec{x}, y) \text{ and } \forall z < y\ f(\vec{x}, z)\downarrow \neq 0 \\ \uparrow & \text{if such a } y \text{ does not exist} \end{cases}$$

you can compute $\mu y.\ f(\vec{x}, y)$

$f(\vec{x}, 0) = 0$? yes $\rightsquigarrow$ stop out 0

NO

$f(\vec{x}, 1) = 0$? yes $\rightsquigarrow$ " " 1

NO

$f(\vec{x}, 2) = 0$?

Proposition: Class $\mathcal{C}$ is closed under (unbounded) minimalisation

proof

Let $f : \mathbb{N}^{k+1} \to \mathbb{N}$ in $\mathcal{C}$

we want to prove $h \in \mathcal{C}$

$h : \mathbb{N}^k \to \mathbb{N}$

$h(\vec{x}) = \mu y.\ f(\vec{x}, y)$

let P be a program (std form) for f

1 ...	k	... m	$m+1$ ...	$m+k$	$m+k+1$	$m+k+2$
x_1 ...	x_k	0 ...	x_1 ...	x_k	i	0

$$m = \max\{\rho(P),\ k+1\}$$

$f(\vec{x}, i)$ $\quad$ $i=0$, $i=1$, $\vdots$

the program for h can be

$T(1, m+1)$ // save input $\vec{x}$
$\vdots$
$T(k, m+k)$

LOOP: $P[m+1, \_, m+k, m+k+1 \to 1]$ // $f(\vec{x}, i)$ in R_1

$J(1, m+k+2, \text{END})$ // $f(\vec{x}, i) = 0$?

$S(m+k+1)$ // i++

$J(1, 1, \text{LOOP})$

END : $T(m+k+1, 1)$ // output i

$\square$

Example

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

$$f(x) = \begin{cases} \sqrt{x} & \text{if } x \text{ is a square} \\ \uparrow & \text{otherwise} \end{cases}$$

computable

$$f(x) = \mu y. \text{ “} y^2 = x \text{”}$$
$$= \mu y.\ |y * y - x|$$

$\rightsquigarrow$ computable by minimalisation

Example

$g : \mathbb{N}^2 \to \mathbb{N}$

$$g(x,y) = \begin{cases} x/y & \text{if } y \neq 0 \text{ and } y \text{ divides } x \\ \uparrow & \text{otherwise} \end{cases}$$

$$g(x,y) \neq \underbrace{\mu z.\ |z * y - x|}_{\substack{y=0 \\ x=0}\ \rightsquigarrow\ 0 \quad \text{you want } \uparrow}$$

$$g(x,y) = \mu z.\left(|z * y - x| + \overline{sg}(y)\right)$$

$$\overline{sg}(y) = \begin{cases} 1 & \text{if } y = 0 \\ 0 & \text{if } y \neq 0 \end{cases}$$

<u>OBSERVATION</u>: Every finite (domain) function is computable

proof

Let $\vartheta : \mathbb{N} \to \mathbb{N}$ be a finite (domain) function

$$\vartheta(x) = \begin{cases} y_1 & x = x_1 \\ y_2 & x = x_2 \\ \vdots & \\ y_n & x = x_n \\ \uparrow & \text{otherwise} \end{cases} \qquad \mathrm{dom}(\vartheta) = \{x_1, \dots, x_n\}$$

$$\vartheta = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

it is computable

$$\vartheta(x) = \underbrace{\sum_{i=1}^{n} y_i \cdot \underbrace{\overline{sg}(|x - x_i|)}_{\substack{1 \text{ if } x = x_i \\ 0 \text{ otherwise}}}}_{\substack{y_i \text{ if } x = x_i \\ 0 \text{ otherwise}}} + \underbrace{\overbrace{\mu z. \underbrace{\prod_{i=1}^{n} |x - x_i|}_{\substack{0 \text{ if } x \in \mathrm{dom}(\vartheta) \\ \neq 0 \text{ otherwise}}}}^{h(x,z) \text{ constant in } z}}_{\substack{0 \text{ if } x \in \mathrm{dom}(\vartheta) \\ \uparrow \text{ otherwise}}}$$

$\square$

Example:

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

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \text{ and } P \neq NP \\ 1 & \text{if } x = 0 \text{ and } P = NP \\ \uparrow & \text{otherwise} \end{cases}$$

computable

$g : \mathbb{N} \to \mathbb{N}$, fixed a program P

$$g(x) = \begin{cases} 0 & \text{if } x = 0 \text{ and } P(x) \downarrow \\ 1 & \text{if } x = 0 \text{ and } P(x) \uparrow \\ \uparrow & \text{otherwise} \end{cases}$$

OBSERVATION: Let $f : \mathbb{N} \to \mathbb{N}$ computable and injective
& total

Then

$$f^{-1}(y) = \begin{cases} x & \text{if } x \text{ is s.t. } f(x) = y \\ \uparrow & \text{if there is no } x \text{ s.t. } f(x) = y \end{cases}$$

computable

proof

$$f^{-1}(y) = \mu x. \, |f(x) - y|$$

$\square$

Not working for non total functions

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

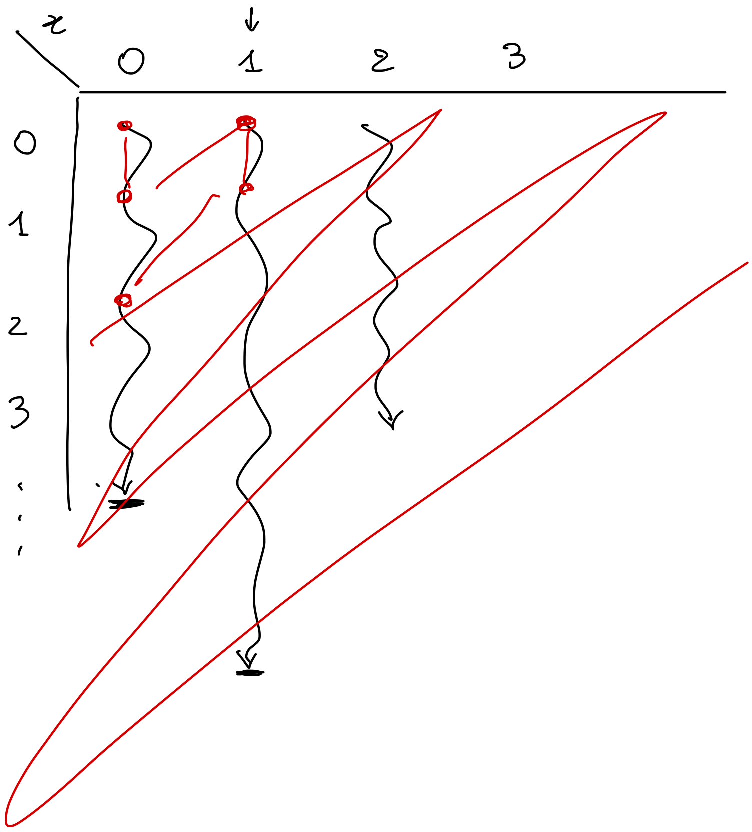
$$f(x) = \begin{cases} x - 1 & x > 0 \\ \uparrow & x = 0 \end{cases}$$

computable

$$= (x \dot{-} 1) + \mu z. \, \overline{sg}(x)$$

$$f^{-1}(y) = y + 1 \neq \underbrace{\mu x. \, |f(x) - y|}_{\text{alway undefined}}$$

if f is non total



try program P computing f

for every possible number of steps

on every possible input

to be formalized

Partial Recursive functions

computational models : TM, λ-calculus, Post systems, ..., URM-machine

<u>Church Turing Thesis</u> : A function is computable by an effective procedure iff it is URM-model

Program

→ class $\mathcal{R}$ of partial recursive functions

→ prove $\mathcal{R} = \mathcal{C}$

<u>Def</u> : The class of <u>partial recursive functions</u> $\mathcal{R}$ is the least class of functions w.r.t. $\subseteq$

which → contains
(a) zero
(b) successor
(c) projections

→ closed under
(1) composition
(2) primitive recursion
(3) minimalisation

in detail

- define a class of functions $\mathcal{A}$ to be <u>rich</u> if
  → it contains (a), (b), (c)
  → it is closed w.r.t. (1), (2), (3)
- $\mathcal{R}$ is a rich class s.t. for all rich classes $\mathcal{A}$ $\mathcal{R} \subseteq \mathcal{A}$
- <u>NOTE</u> : given $\mathcal{A}_i$ $i \in I$ rich classes then $\bigcap_{i \in I} \mathcal{A}_i$ rich
- the class of all functions is rich

⇝

$$\mathcal{R} = \bigcap_{\mathcal{A} \text{ rich class}} \mathcal{A}$$

<u>Equivalently</u> : $\mathcal{R}$ is the class of functions which you can obtain from the basic fuctions using a finite number of times (1), (2), (3)

(EXERCISE)

Theorem : $\mathcal{C} = \mathcal{R}$

proof

$(\mathcal{R} \subseteq \mathcal{C})$ $\quad \mathcal{C}$ is rich , $\mathcal{R}$ is smallest rich class

$\leadsto \mathcal{R} \subseteq \mathcal{C}$

$(\mathcal{C} \subseteq \mathcal{R})$ let $f : \mathbb{N}^k \to \mathbb{N}$ $\quad f \in \mathcal{C} \leadsto f \in \mathcal{R}$

there is a URM-program for f , call it P

$x_1 \dots x_k$	$0 \; 0 \; \dots$

$P \Downarrow$

$f(\vec{x})$	$- \; - \; - \; -$

$$\begin{cases} c_P^1 : \mathbb{N}^{k+1} \to \mathbb{N} \\ c_P^1(\vec{x}, t) = \text{content of } R_1 \text{ after } t \text{ steps of computation of } P(\vec{x}) \end{cases}$$

$$\begin{cases} J_P : \mathbb{N}^{k+1} \to \mathbb{N} \\ J_P(\vec{x}, t) = \begin{cases} \text{instruction to be executed after } t \text{ steps of } P(\vec{x}) \\ 0 \quad \text{if } P(\vec{x}) \text{ terminates in } t \text{ steps or fewer} \end{cases} \end{cases}$$

let $\vec{x} \in \mathbb{N}^k$

→ if $f(\vec{x})\downarrow$ then $P(\vec{x})\downarrow$ in a number of steps

$t_0 = \mu t . J_P(\vec{x}, t)$

hence

$$f(\vec{x}) = c_P^1(\vec{x}, t_0) = c_P^1(\vec{x}, \mu t . J_P(\vec{x}, t))$$

→ if $f(\vec{x})\uparrow$ then $P(\vec{x})\uparrow$

hence $\mu t . J_P(\vec{x}, t)\uparrow$

$$f(\vec{x}) = c_P^1(\vec{x}, \underbrace{\mu t . J_P(\vec{x}, t)}_{\uparrow})\uparrow$$

In all cases

$$f(\vec{x}) = \varsigma_p^1(\vec{x}, \mu t. J_p(\vec{x}, t))$$

If we knew $\varsigma_p^1, J_p \in \mathcal{R}$ we could conclude $f \in \mathcal{R}$

[TO BE CONTINUED]