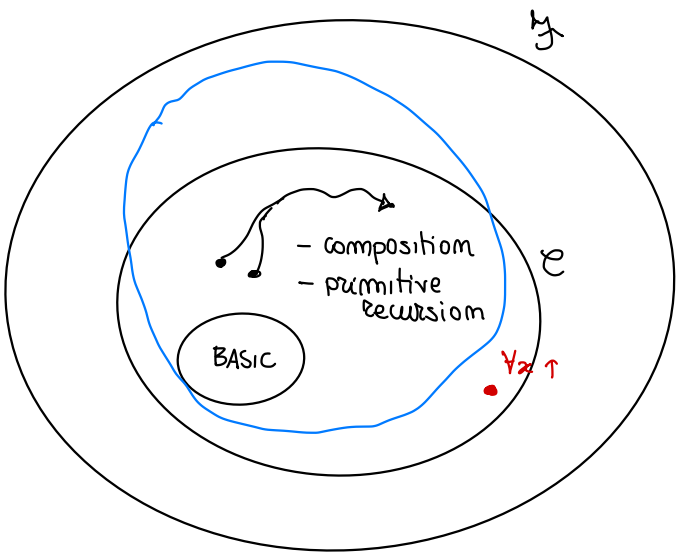
COMPUTABILITY (20/10/2026)



* Unbounded minimalisation

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

$f(\vec{x}, z)$ (not necessarily total)

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

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

$$= \text{least } y \in \mathbb{N} \text{ s.t. } f(\vec{x}, y) = 0$$

$\rightsquigarrow$ such a y might not exist .... or

$\to$ ... it might exist but for some $z < y$

$f(\vec{x}, z) \uparrow$

$\hookrightarrow$ undefined

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

for computing $\mu y.\ f(\vec{x}, y)$

$f(\vec{x}, 0) = 0$? if yes $\to$ out 0

no

$f(\vec{x}, 1) = 0$? if yes $\to$ out 1

no

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

Example:

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

$\mu y.\ f(x,y) \uparrow \quad \forall x$

$\mu y.\ f(x,y) \nexists x+1$ because $f(x,y)\uparrow \quad y=x$

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

proof

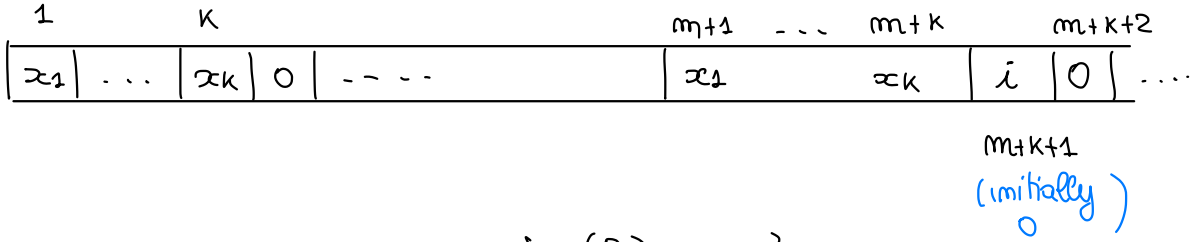
Let $f: \mathbb{N}^{k+1} \to \mathbb{N} \qquad f \in \mathcal{C}$

we want to prove that

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

$h(\vec{x}) = \mu y.\ f(\vec{x},y)$ is computable in $\mathcal{C}$

Let P be a URM program for f (std form)



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

the program for h can be

```
T(1, m+1)                          // save x⃗ to safe memory
⋮
T(k, m+k)
LOOP: P[m+1 ... m+k m+k+1 → 1]     // f(x⃗, i)
      J(1, m+k+2, RES)             // f(x⃗, i) = 0?
      S(m+k+1)                     // i ++
      J(1, 1, LOOP)
RES:  T(m+k+1, 1)                  // out i
```

□

Example :

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

$$f(x) = \begin{cases} x/2 & x \text{ even} \\ \uparrow & x \text{ odd} \end{cases}$$

$$= \mu y. \; |x - 2*y|$$

computable since

($|-|$, $*$ known computable)

Example

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

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

~~$=$~~ $\mu z. \; |y*z - x|$

↑

if $y=0$ and $x=0$ $\rightsquigarrow$ 0

we want ↑

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

$\overline{sg}(y)$: 0 when $y \neq 0$; >0 " $y = 0$

OBSERVATION : Every finite (domain) function is computable

proof

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

i.e. $dom(\vartheta) = \{x_1, \dots, x_m\}$

$$\vartheta(x) = \begin{cases} y_1 & \text{if } x = x_1 \\ y_2 & \text{if } x = x_2 \\ \vdots & \\ y_m & \text{if } x = x_m \\ \uparrow & \text{otherwise} \end{cases}$$

it is computable

$$\theta(x) = \sum_{i=1}^{m} y_i * \overline{sg}\,|x - x_i| + \mu z.\ \prod_{i=1}^{m} |x - x_i|$$

$\overline{sg}\,|x-x_i|$: 1 if $x = x_i$; 0 if $x \neq x_i$

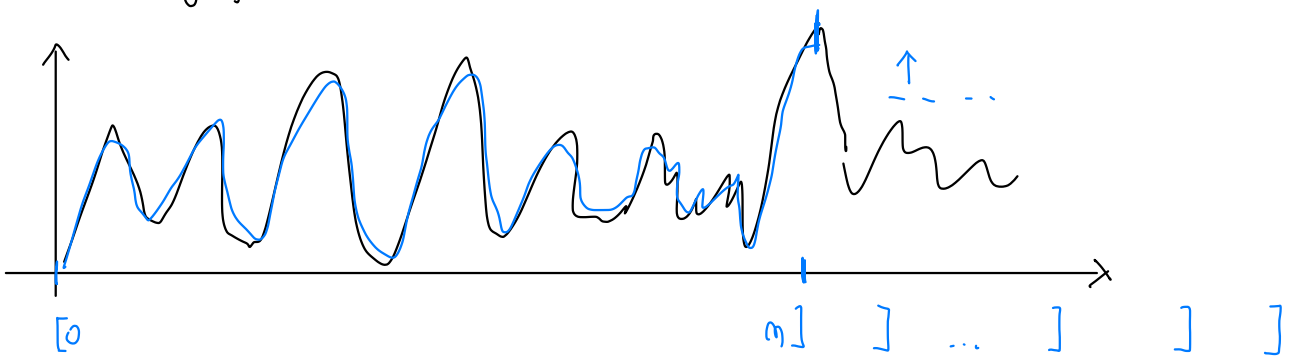
y_i: y_i if $x = x_i$; 0 otherwise

$\prod_{i=1}^{m} |x-x_i|$: 0 if $x = x_i$ for some i; $\uparrow$ otherwise

$\mu z.\ \prod_{i=1}^{m} |x-x_i|$: 0 if $x = x_i$ for some i; >0 otherwise

computable by minimalisation and composition. □

Note: Every function can be approximated by finite (computable) ones



Example:

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

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

computable (finite function)

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

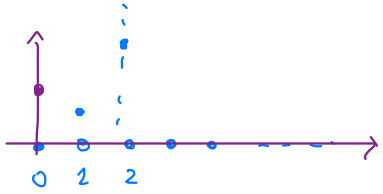
Then

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

is computable

proof

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



Not a proof without totality assumption

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

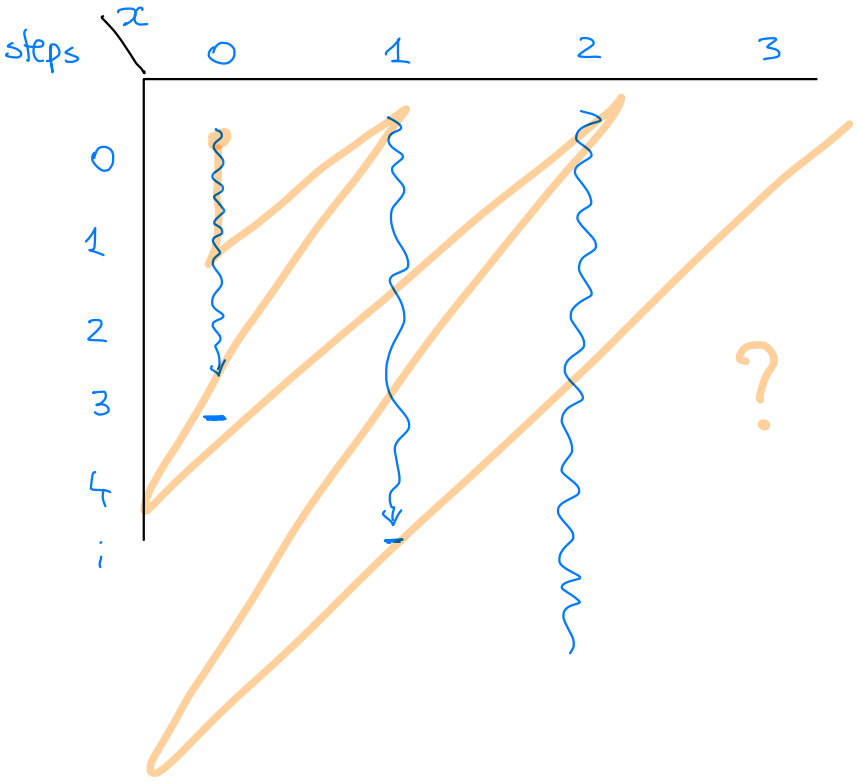
$$f(x) = \begin{cases} \uparrow & x = 0 \\ x - 1 & x > 0 \end{cases}$$

computable

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

$$f^{-1}(y) = y + 1 \quad \neq \quad \mu x.\ |f(x) - y| \uparrow \quad \forall y$$

When f is not total



PARTIAL RECURSIVE FUNCTIONS

computational model : TM , λ-calculus, Post-system, . . , URM

Church Turing Thesis :

A function is computable by an effective procedure iff URM-computable

Program

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

→ $\mathcal{R} = \mathcal{C}$

Def : The class of partial recursive functions $\mathcal{R}$ is the least class of functions which (w.r.t. $\subseteq$)

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

closed by
(1) composition
(2) primitive recursion
(3) minimalisation

namely

→ define a class of functions $\mathcal{A} \subseteq \mathcal{G}$ rich if

it contains
(a) zero
(b) successor
(c) projections

closed by
(1) composition
(2) primitive recursion
(3) minimalisation

→ $\mathcal{R}$ is such that
- $\mathcal{R}$ is rich
- for all $\mathcal{A} \subseteq \mathcal{G}$ rich $\quad \mathcal{R} \subseteq \mathcal{A}$

→ NOTE :
- $\mathcal{G}$ is rich
- if $\mathcal{A}_i$ $i \in I$ $\mathcal{A}_i$ rich $\Rightarrow$ $\bigcap_{i \in I} \mathcal{A}_i$ rich

$$\Rightarrow \quad \mathcal{R} = \bigcap_{\mathcal{A} \text{ rich}} \mathcal{A}$$

Equivalently: $\mathcal{R}$ is the class of functions that you can obtain from the basic functions using (1), (2), (3)

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

proof

$(\mathcal{R} \subseteq \mathcal{C})$ $\mathcal{C}$ is rich, $\mathcal{R}$ is the least rich class

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

$(\mathcal{C} \subseteq \mathcal{R})$ let $f : \mathbb{N}^k \to \mathbb{N}$ be in $\mathcal{C}$ $\overset{?}{\rightsquigarrow}$ $f \in \mathcal{R}$

there is a URM-program P for f $\left(f_P^{(k)} = f \right)$

x_1 ... x_k	0 .....

P ⇓

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

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

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

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

→ if $f(\vec{x}) \downarrow$ then $P(\vec{x}) \downarrow$ in a number of steps $\mu t.\ J_P(\vec{x}, t)$

hence

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

→ if $f(\vec{x}) \uparrow$ then $P(\vec{x}) \uparrow$ $\rightsquigarrow$ $\forall t$ $J_P(\vec{x}, t) > 0$

thus

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

therefore

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

Therefore

$$f(\vec{x}) = C_P^1(\vec{x}, \mu t.\ J_P(\vec{x}, t)) \quad \forall \vec{x}$$

if we prove that $C_P^1, J_P \in \mathcal{R}$ then we conclude $f \in \mathcal{R}$

[WILL DO NEXT LESSON]