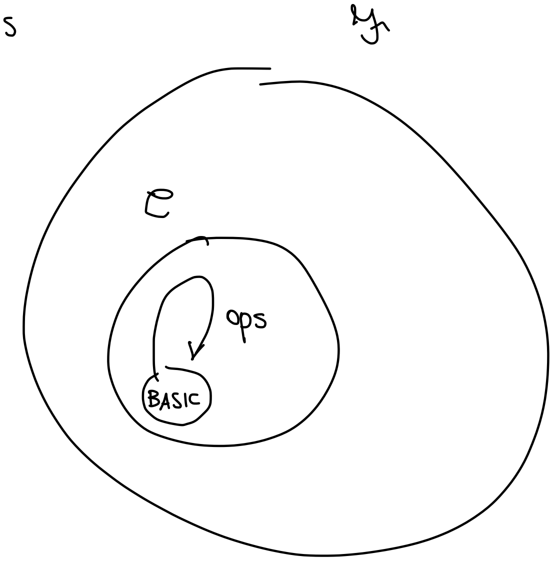
Computability (25/10/2021)

→ class $\mathcal{C}$ of URM-computable functions



* **OBSERVATION**: Every finite domain function is computable

proof Let $\theta : \mathbb{N} \to \mathbb{N}$ be a finite domain function

$$\theta(x) = \begin{cases} y_1 & x = x_1 \\ y_2 & x = x_2 \\ \vdots & \\ y_n & x = x_n \\ \uparrow & \text{otherwise} \end{cases}$$

$\mathrm{dom}(\theta) = \{x_1, \ldots, x_n\}$

$$\theta(x) = \sum_{i=1}^{n} y_i \cdot \overline{sg}(|x - x_i|) + \underbrace{\mu z . \left(\prod_{i=1}^{n} |x - x_i| \right)}$$

$\overline{sg}(|x - x_i|)$: 1 if $x = x_i$, 0 otherwise

$\prod_{i=1}^{n} |x - x_i|$: $= 0$ if $x \in \mathrm{dom}(\theta)$, $\neq 0$ otherwise

$\mu z . \left(\prod_{i=1}^{n} |x - x_i| \right)$: $= 0$ if $x \in \mathrm{dom}(\theta)$, $\uparrow$ otherwise

$\Rightarrow \theta \in \mathcal{C}$ by closure properties

OBSERVATION: Let $f: \mathbb{N} \to \mathbb{N}$ computable TOTAL injective

The inverse

$$f^{-1}: \mathbb{N} \to \mathbb{N}$$

$$f^{-1}(x) = \begin{cases} y & \text{if } f(y) = x \\ \uparrow & \text{if } \nexists y \text{ s.t. } f(y) = x \end{cases}$$

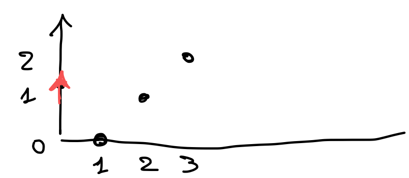
proof

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

OK

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

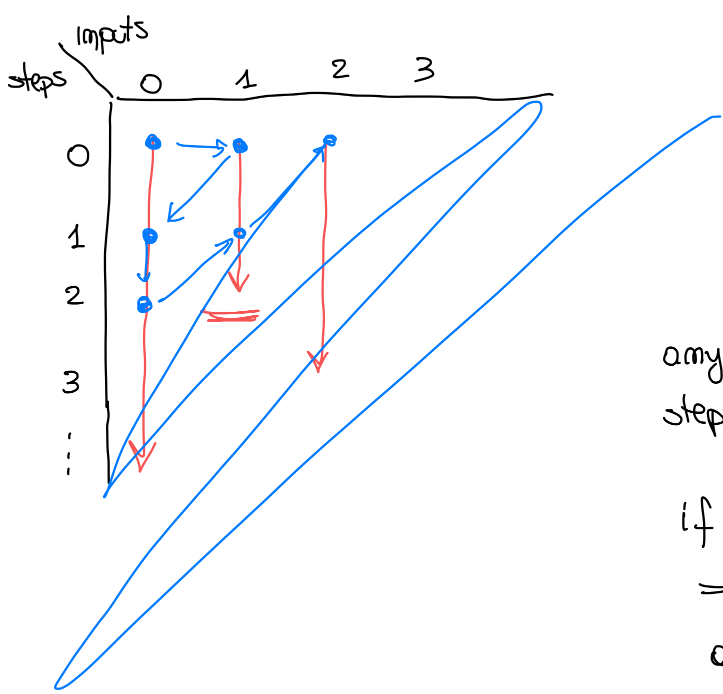
$$f(x) = \begin{cases} x - 1 & x > 0 \\ \uparrow & x = 0 \end{cases}$$



$$f^{-1}(x) = s(x) \neq \overset{g(x)}{\overset{\shortparallel}{\mu y}} . |f(y) - x| \uparrow \forall x$$

$y = 0$

If f is not total? The result holds but for now we can't prove it



f computable

P program for f

any input y for every number of steps K

if $f(y) = x$

$\Rightarrow$ P stops on x in K steps and we will find it

* Partial Recursive Functions

computational models TM, λ-calculus, ...

Church Turing Thesis : A function is computable via an effective procedure iff URM-computable

* Partial Recursive Functions :

- class $\mathcal{R}$ of computable functions in the model
- $\mathcal{R} = \mathcal{C}$

Def: The class of partial recursive functions is the least class of functions which

with respect to $\subseteq$

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

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

Well given definition because

→ call a class of functions $\mathcal{A}$ rich if it includes (a), (b), (c) and it is closed under (1), (2), (3)

→ $\mathcal{R}$ is rich & for all $\mathcal{A}$ rich class $\mathcal{R} \subseteq \mathcal{A}$

→ observation : if $\mathcal{A}_i$ with $i \in I$ is such that $\mathcal{A}_i$ is rich $\forall i \in I$

$$\Rightarrow \bigcap_{i \in I} \mathcal{A}_i \text{ is rich}$$

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

Equivalently: $\mathcal{R}$ is the set of functions which can be obtained from the basic functions (a), (b), (c) using (1), (2), (3).

$\boxed{\text{EXERCISE}}$

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

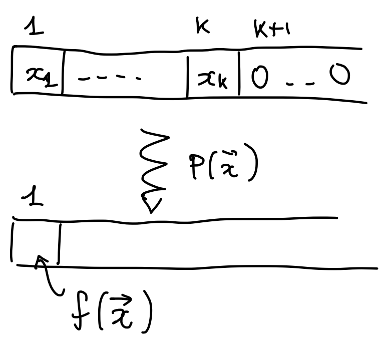
proof

($\mathcal{R} \subseteq \mathcal{C}$) $\mathcal{C}$ is a rich class, $\mathcal{R}$ is the smallest rich class.

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

there exists P URM program such that $f_P^{(k)} = f$

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



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

$C_P^1(\vec{x}, t)$ = content of R_1 after t steps of $P(\vec{x})$

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

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

- given $\vec{x} \in \mathbb{N}^k$

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

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

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

→ if $f(\vec{x})\uparrow$ $\quad \mu t.\, J_P(\vec{x}, t)\uparrow$ because $P(\vec{x})\uparrow$

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

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

if we knew that $\boxed{C_P^1, J_P \in \mathcal{R}}$ $\rightsquigarrow f \in \mathcal{R}$

program P

→ $\left\{ \begin{array}{l} I_1 \\ \vdots \\ I_s \end{array} \right.$ standard form

memory

r_1	r_2	- - -	r_m	0	0	- - - 0

encode

$$C = \prod_{i \geq 1} p_i^{r_i} = \prod_{i=1}^{m} p_i^{r_i}$$

$$r_i = (C)_i$$

2	0	1	0	0 . . .

$$C = p_1^2 \cdot p_2^0 \cdot p_3^1 \cdot \cancel{p_4^0 \cdots}$$

$$= 2^2 \cdot 3^0 \cdot 5^1$$

$$= 20$$

$C_P, J_P : \mathbb{N}^{k+1} \to \mathbb{N}$

$C_P(\vec{x}, t)$ = content of registers after t steps of $P(\vec{x})$

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

we define C_P, J_P by primitive recursion:

$$C_P(\vec{x}, 0) = \prod_{i=1}^{k} p_i^{x_i}$$

1		k		
x_1	..	x_k	0	0 ...

$$J_P(\vec{x}, 0) = 1$$

"recursion" cases: use $C = C_P(\vec{x}, t)$, $J = J_P(\vec{x}, t)$

$$C_P(\vec{x}, t+1) = \begin{cases} qt\left(p_m^{(C)_m}, C\right) & \text{if } 1 \leq J \leq \ell(s) \text{ and } I_J = Z(m) \\ p_m \cdot C & \text{if } 1 \leq J \leq \ell(s) \text{ and } I_J = S(m) \\ qt\left(p_m^{(C)_m}, C\right) \cdot p_n^{(C)_m} & \text{if } 1 \leq J \leq \ell(s) \text{ \& } I_J = T(m,n) \\ C & \text{otherwise } \left(\begin{array}{l} 1 \leq J \leq \ell(s) \text{ and } I_J = J(\dots) \\ \text{or} \\ J = 0 \end{array}\right) \end{cases}$$

$C =$

r_1	..	r_m	..	r ...

$p_1^{r_1}\, p_2^{r_2} \cdots p_m^{\cancel{r_m}\; 0}$ ... $r_m = (C)_m$

$$J_P(\vec{x}, t+1) = \begin{cases} J+1 & \text{if } 1 \leq J < \ell(P) \text{ and } \left(\begin{array}{l} I_J = Z \text{ or } S \text{ or } T \\ \text{or } \left(I_J = J(m,n,t) \text{ and } (C)_m \neq (C)_n\right) \end{array}\right) \\ u & \text{if } 1 \leq J \leq \ell(P) \text{ and } I_J = J(m,n,u) \text{ and } (C)_m = (C)_n \text{ and } 1 \leq u \leq \ell(P) \\ 0 & \text{otherwise} \end{cases}$$