COMPUTABILITY (12/10/2024)

* PARAMETRISATION THEOREM

$\varphi_e^{(2)}(x, y)$ computed by $P_e = \gamma^{-1}(e)$

for every $x \in \mathbb{N}$ fixed, we get a function of one argument y

$x = 0 \qquad y \mapsto \varphi_e^{(2)}(0, y)$

$x = 1 \qquad y \mapsto \varphi_e^{(2)}(1, y)$

$\vdots \qquad\qquad \vdots$

smn-theorem: for each $x \in \mathbb{N}$ fixed the program computing the function of y can be constructed algorithmically starting from P_e

$P_e(x, y)$
$\quad \equiv x$
y
$\quad x$
return ...

$\xrightarrow{x = x_0}$

$P_e(\cancel{x}, y)$
$\quad \equiv \cancel{x}\ x_0$
y
$\quad \cancel{x}\ x_0$
return ...

more generally:

$\varphi_e^{(m+n)}(\vec{x}, \vec{y}) \qquad\qquad \varphi^{(n)}_{s(e,\vec{x})}(\vec{y})$

s computable

Theorem (smn theorem)

Given $m, n \geq 1$ there a total computable function $s_{m,n} : \mathbb{N}^{m+1} \to \mathbb{N}$ such that for all $e \in \mathbb{N}$, $\vec{x} \in \mathbb{N}^m$, $\vec{y} \in \mathbb{N}^n$

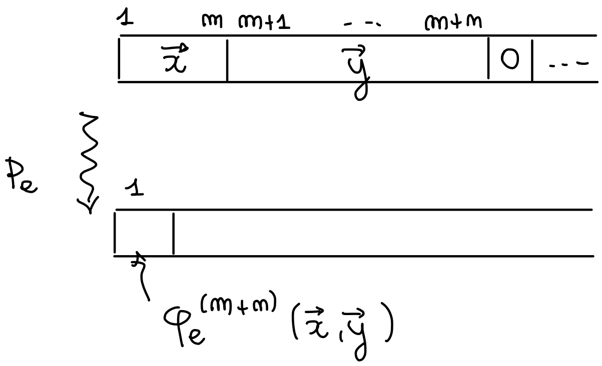
$$\varphi_e^{(m+n)}(\vec{x}, \vec{y}) = \varphi^{(n)}_{s_{m,n}(e, \vec{x})}(\vec{y})$$

proof

intuitively $\quad e \in \mathbb{N}, \quad \vec{x} \in \mathbb{N}^m$

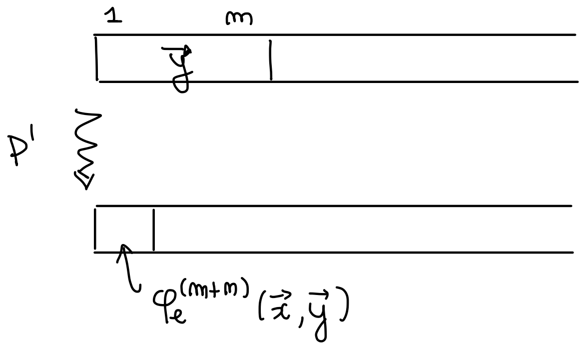
$$e \rightsquigarrow \gamma^{-1}(e) = P_e$$

starting from



we want to construct, given $\vec{x} \in \mathbb{N}$, a program P'

$\nwarrow$ depending on e on $\vec{x}$



P' should
- move $\vec{y}$ to $m+1, -, m+n$
- write $\vec{x}$ in $1, -, m$
- execute P_e

P'	
$T(n, m+n)$	// move y_n to $m+n$
$\vdots$	$\vdots$
$T(1, m+1)$	// move y_1 to $m+1$
$Z(1)$	// write x_1 to R_1
$\left.\begin{array}{l} S(1) \\ \vdots \\ S(1) \end{array}\right\} x_1$ times	
$\vdots$	
$Z(m)$	// write x_m to R_m
$\left.\begin{array}{l} S(m) \\ \vdots \\ S(m) \end{array}\right\} x_m$ times	
$P_e = \gamma^{-1}(e)$	

hence

$$S_{m,n}(e, \vec{x}) = \gamma(P')$$

① sequential composition of programs $\left(e_1, e_2 \rightsquigarrow \gamma\begin{pmatrix} P_{e_1} \\ P_{e_2} \end{pmatrix} \right)$

(1.a) $upd : \mathbb{N}^2 \to \mathbb{N}$

$$upd(e,h) = \gamma \begin{pmatrix} \text{program obtained from } P_e = \gamma^{-1}(e) \text{ by updating} \\ \text{all jump instructions } J(m,n,t) \rightsquigarrow J(m,n,t+h) \end{pmatrix}$$

$$\widetilde{upd}(i,h) = \beta \begin{pmatrix} \text{instruction obtained from } \beta^{-1}(i) \text{, updating the} \\ \text{target of the jump if it is a jump} \end{pmatrix}$$

RECALL : $\beta(J(m,n,t)) = \nu(m-1, n-1, t-1) * 4 + 3$

$$= \begin{cases} i & \text{if } rm(4,i) \neq 3 \\ \nu(\nu_1(q), \nu_2(q), \nu_3(q)+h) * 4 + 3 & \text{if } rm(4,i) = 3 \\ & q = qt(4,i) \end{cases}$$

$$= i * sg(|rm(4,i)-3|) + (\nu(\nu_1(q), \nu_2(q), \nu_3(q)+h) * 4 + 3) \cdot \overline{sg}(|rm(4,i)-3|)$$

now

$$upd(e,h) = \tau(\widetilde{upd}(a(e,1),h), \widetilde{upd}(a(e,2),h), \dots, \widetilde{upd}(a(e,\ell(e)),h))$$

$$= \left(\prod_{i=1}^{\ell(e)-1} p_i^{\widetilde{upd}(a(e,i),h)} \right) \cdot p_{\ell(e)}^{\widetilde{upd}(a(e,\ell(e)),h)+1} \dot{-} 2$$

$$\tau(y_1, \dots, y_m) = \left(\prod_{i=1}^{m-1} p_i^{y_i} \right) \cdot p_m^{y_m+1} \dot{-} 2$$

$\ell(e)$ = length of the sequence $\tau^{-1}(e)$

$1 \le i \le \ell(e)$ $\quad a(e,i) = i^{th}$ component

- $c : \mathbb{N}^2 \to \mathbb{N}$

$c(e_1,e_2)$ = code of the concatenation of $\tau^{-1}(e_1)$ and $\tau^{-1}(e_2)$

$$= \tau(a(e_1,1) \dots a(e_1,\ell(e_1)) \; a(e_2,1) \dots a(e_2,\ell(e_2)))$$

$= \dots$

- $seq : \mathbb{N}^2 \to \mathbb{N}$

$$seq(e_1,e_2) = \gamma\begin{pmatrix} P_{e_1} \\ P_{e_2} \end{pmatrix} = c(e_1, upd(e_2, \ell(e_1)))$$

(2) $\text{trasf} : \mathbb{N}^2 \to \mathbb{N}$

$$\text{transf}(m, n) = \gamma \begin{pmatrix} T(n, m+n) \\ \vdots \\ T(1, m) \end{pmatrix} = \dots$$

(3) $\text{set} : \mathbb{N}^2 \to \mathbb{N}$

$$\text{set}(i, x) = \gamma \left(\begin{array}{l} Z(i) \\ \left.\begin{array}{l} S(i) \\ \vdots \\ S(i) \end{array}\right\} x \text{ times} \end{array} \right) = \dots$$

(4) finally

$S_{m,n}(e, \vec{x}) =$

$\text{seq}(\text{transf}(m, n),$
$\quad \text{seq}(\text{set}(1, x_1),$
$\quad\quad \text{seq}(\text{set}(2, x_2),$
$\quad\quad\quad \ddots$
$\quad\quad\quad\quad \text{seq}(\text{set}(m, x_m), e) \dots)$

computable (primitive recursive) since it arises as the composition of primitive recursive functions. $\square$

P'

$$\left\{ \begin{array}{l} T(n, m+n) \\ \vdots \\ T(1, m+1) \end{array} \right.$$

$$\left\{ \begin{array}{l} Z(1) \\ \left.\begin{array}{l} S(1) \\ \vdots \\ S(1) \end{array}\right\} x_1 \text{ times} \end{array} \right.$$

$\vdots$

$$\left\{ \begin{array}{l} Z(m) \\ \left.\begin{array}{l} S(m) \\ \vdots \\ S(m) \end{array}\right\} x_m \text{ times} \end{array} \right.$$

$P_e = \gamma^{-1}(e)$

<u>Corollary</u>: Let $f : \mathbb{N}^{m+n} \to \mathbb{N}$ computable. Then there is a total computable function

$$s : \mathbb{N}^m \to \mathbb{N}$$

s.t. $\forall \vec{x} \in \mathbb{N}^m, \vec{y} \in \mathbb{N}^n \qquad f(\vec{x}, \vec{y}) = \varphi^{(n)}_{s(\vec{x})}(\vec{y})$

<u>proof</u>

since f is computable there is $e \in \mathbb{N}$ s.t. $\varphi_e^{(m+n)} = f$

$$f(\vec{x}, \vec{y}) = \varphi_e^{(m+n)}(\vec{x}, \vec{y}) \underset{\uparrow}{=} \varphi^{(n)}_{S_{m,n}(e, \vec{x})}(\vec{y})$$

smn theorem
$S_{m,n} : \mathbb{N}^{m+1} \to \mathbb{N}$ total computable

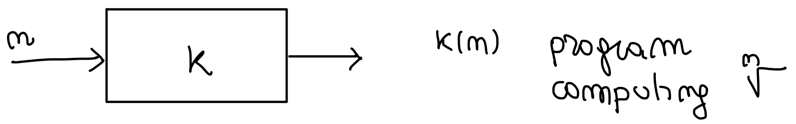
and conclude by letting $s(\vec{x}) = S_{m,n}(e, \vec{x})$ $\square$

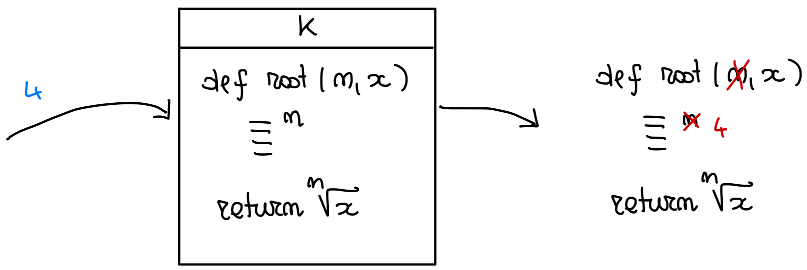
Example :

Prove that there is a total computable function $K : \mathbb{N} \to \mathbb{N}$ such that

$\forall m \in \mathbb{N} \ \forall x \in \mathbb{N}$

$$\varphi_{K(m)}(x) = \lfloor \sqrt[m]{x} \rfloor$$





the function

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

$$f(m, x) = \lfloor \sqrt[m]{x} \rfloor$$

$$= \text{search max } z \qquad \text{“} z^m \leq x \text{”}$$

$$= \min z \quad . \quad (z+1)^m > x$$

$$= \mu z \leq x \ . \ \overline{sg}\left((z+1)^m \dot{-} x\right) \qquad \text{computable}$$

hence by smn theorem (corollary of)

there is $K : \mathbb{N} \to \mathbb{N}$ total computable such that

$$f(m, x) = \varphi_{K(m)}(x) \qquad \forall m, x$$

$\forall m, x$

$$\varphi_{K(m)}(x) = f(m, x) = \lfloor \sqrt[m]{x} \rfloor$$

EXAMPLE : There is a total computable function $k : \mathbb{N} \to \mathbb{N}$ such that

$\forall m$ $\quad \varphi_{k(m)}$ is defined only on m^{th} powers

(numbers y^m for some y)

$$W_{k(m)} = \{ x \mid \exists y.\ x = y^m \}$$

define $\quad f : \mathbb{N}^2 \to \mathbb{N}$

$$f(m, x) = \begin{cases} \downarrow & \text{if } \exists y \text{ s.t. } x = y^m \\ \uparrow & \text{otherwise} \end{cases}$$

$\sqrt[m]{x}$

$$= \mu y \quad \text{“} y^m = x \text{”}$$

$$= \mu y.\ |y^m - x| \qquad \text{computable}$$

By corollary of smn theorem there is $k : \mathbb{N} \to \mathbb{N}$ total computable such that $\forall m, x$

$$\varphi_{k(m)}(x) = f(m, x) = \begin{cases} \sqrt[m]{x} & \text{if } \exists y \text{ s.t. } x = y^m \\ \uparrow & \text{otherwise} \end{cases}$$

Observe $\quad W_{k(m)} = \{ x \mid \exists y.\ x = y^m \}$

in fact

$$x \in W_{k(m)} \quad \text{iff} \quad \underset{f(m,x)}{\overset{\varphi_{k(m)}(x)}{\shortparallel}}\downarrow \quad \text{iff} \quad \exists y \quad x = y^m$$

$\square$

EXERCISE : show that there is a total computable function s.t.

$$W^{(k)}_{s(x)} = \{ (y_1, \dots, y_k) \mid \sum_{i=1}^{k} y_i = x \}$$

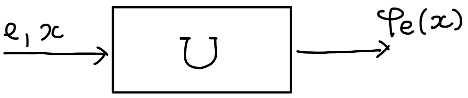
[EXERCISE]

Universal Function

Let $\quad \Psi_U : \mathbb{N}^2 \to \mathbb{N}$

$$\Psi_U(e, x) = \varphi_e(x) \qquad \text{well defined}$$

Is it computable ?



(executes $P_e = \gamma^{-1}(e)$ over x)

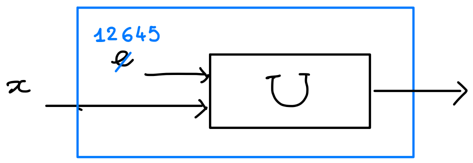
when e varies on the natural numbers

$$\Psi_U(0, \_) \qquad \Psi_U(1, \_) \qquad \Psi_U(2, \_) \qquad \text{- - - -}$$

$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

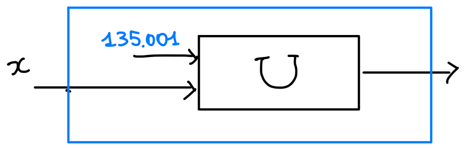
$$\varphi_0 \qquad\qquad \varphi_1 \qquad\qquad \varphi_2$$

Turing corp.



12.645

1.000.000 \$



135.001

1.000.000 \$

⋮

Theorem (Universal Program)

Let $k \geq 1$ then the universal function

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

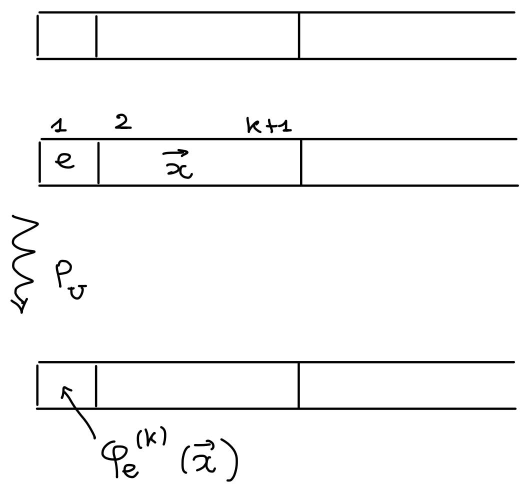
$$\Psi_U(e, \vec{x}) = \varphi_e^{(k)}(\vec{x})$$

is computable

proof

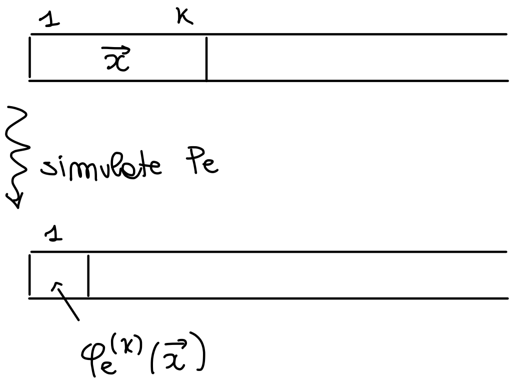
fix $K \geq 1$

given e, $\vec{x}$



how can P_U work

→ determine $P_e = \gamma^{-1}(e)$



by Church–Turing thesis

computable

unsatisfactory!

(more to come in the next lesson)