COMPUTABILITY (07/11/2023)

* Parametrisation Theorem

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

for any fixed x, one obtains a function of y only

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

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

$\vdots \qquad\qquad \vdots$

the program which computes the functions above for each fixed x can be obtained algorithmically starting from P_e

$P_e\ (x, y)$ $\qquad x = x_0 \rightsquigarrow \qquad$ $P_e\ (\cancel{x}, y)$

$x \qquad\qquad\qquad\qquad \cancel{x} \leftarrow x_0$

$y \qquad\qquad\qquad\qquad y$

$\vdots \qquad\qquad\qquad\qquad \vdots$

more generally $f : \mathbb{N}^{m+n} \to \mathbb{N}$

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

Theorem (smn theorem):

Given $m, n \geq 1$ there is a total computable function

$s_{m,n} : \mathbb{N}^{m+1} \to \mathbb{N}$ such that for all $\vec{x} \in \mathbb{N}^m$, $\vec{y} \in \mathbb{N}^n$, $e \in \mathbb{N}$

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

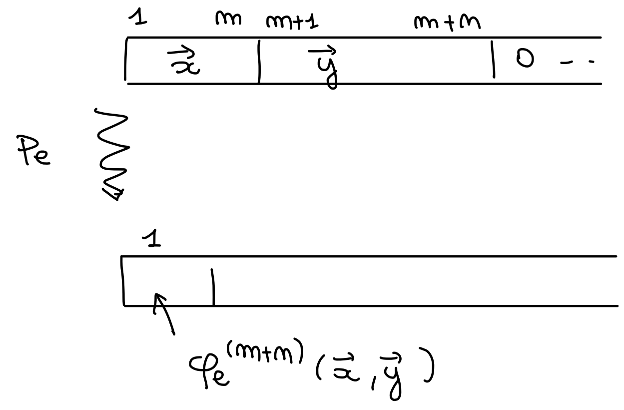
proof

intuitively given $e \in \mathbb{N}$ $\quad \vec{x} \in \mathbb{N}$

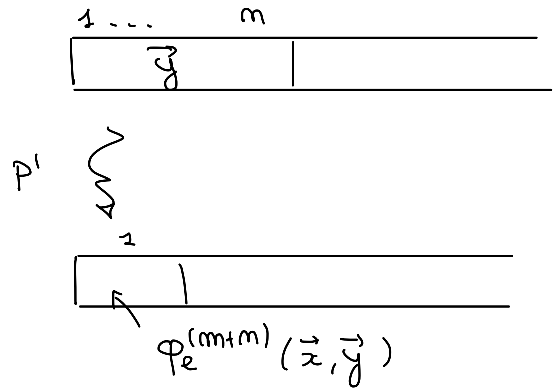
$\uparrow$

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

starting from



you want, for each $\vec{x} \in \mathbb{N}^m$ fixed, a program P' ← depending on $e, \vec{x}$



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

P'

$T(n, m+n)$ // move y_n to R_{m+n}
$\vdots$ $\vdots$
$T(1, m+1)$ // move y_1 to R_{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
$\left.\begin{array}{l} S(1) \\ \vdots \\ S(1) \end{array}\right\} x_m$ times

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

$$S(e, \vec{x}) = \gamma(P')$$

① <u>sequential composition of programs</u> $\quad (e_1, e_2 \rightsquigarrow \gamma\begin{pmatrix} P_{e_1} \\ P_{e_2} \end{pmatrix})$

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

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

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

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

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

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

now

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

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

$$\tau(y_1 \dots y_n) = \prod_{i=1}^{n-1} p_i^{y_i} \cdot p_n^{y_n+1} \dot{-} 2$$

$\ell(e)$ = length of the encoded sequence

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

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

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

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

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

② $\text{transf} : \mathbb{N}^2 \to \mathbb{N}$

$$\left.\text{transf}(m,n) = \gamma\begin{pmatrix} T(n, m+n) \\ \vdots \\ T(1, m+1) \end{pmatrix} = \ldots\right\} *$$

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

$$\left.\text{set}(i,x) = \gamma\begin{pmatrix} Z(i) \\ \left.\begin{matrix} S(i) \\ \vdots \\ S(i) \end{matrix}\right\} x \text{ times} \end{pmatrix} = \ldots\right\} *$$

P'

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

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

$\vdots$

$$* \left\{ \begin{matrix} Z(m) \\ \left.\begin{matrix} S(1) \\ \vdots \\ S(1) \end{matrix}\right\} x_m \text{ times} \end{matrix} \right.$$

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

④ finally

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

$\quad \text{seq}(\text{transf}(m,n),$

$\quad\quad \text{seq}(\text{set}(1, x_1),$

$\quad\quad\quad \ddots$

$\quad\quad\quad\quad \text{seq}(\text{set}(m, x_m), e) \ldots)$

computable (actually primitive recursive) since it is a composition of prim. rec. functions. $\square$

Corollary: Let $f : \mathbb{N}^{m+n} \to \mathbb{N}$ be a computable function.

Then there is a total computable function $s : \mathbb{N}^m \to \mathbb{N}$

s.t. $\forall \vec{x} \in \mathbb{N}^m, \vec{y} \in \mathbb{N}^n$

$$f(\vec{x}, \vec{y}) = \varphi^{(n)}_{s(\vec{x})}(\vec{y})$$

proof

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

$$f(\vec{x}, \vec{y}) = \varphi_e^{(m+n)}(\vec{x}, \vec{y}) \underset{\text{smn theorem}}{\overset{\uparrow}{=}} \varphi^{(n)}_{S_{m,n}(e,\vec{x})}(\vec{y}) \qquad \forall \vec{x}, \vec{y}$$

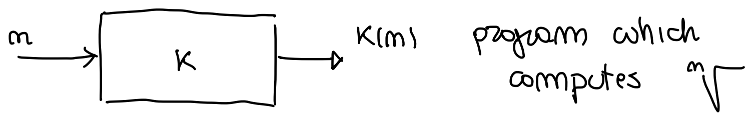
we conclude by setting $s(\vec{x}) = S_{m,n}(e, \vec{x})$ $\square$

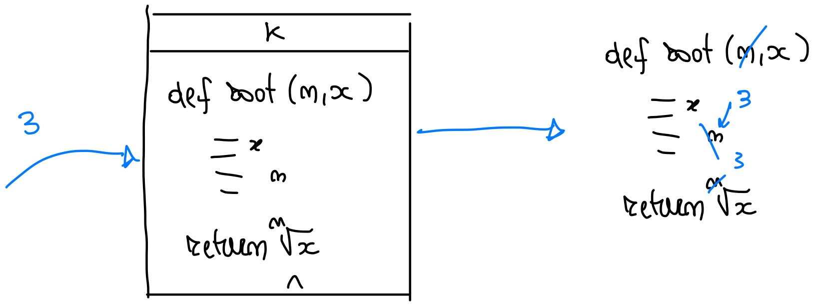
EXAMPLE

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

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

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





the function

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

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

$$= \max z \,.\; \text{“} z^m \le x \text{”}$$

$$= \min z \,.\; \text{“} (z+1)^m > x \text{”}$$

$$= \mu z \le x.\; x+1 \dot{-} (z+1)^m$$

computable

$\Downarrow$

by (corollary of) smn theorem there is $k : \mathbb{N} \to \mathbb{N}$ total computable s.t.

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

EXAMPLE: There is a total computable function $k : \mathbb{N} \to \mathbb{N}$ s.t.

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

(on y^n for $y \in \mathbb{N}$)

$$W_{k(n)} = \{ x \mid \exists y.\ \text{s.t. } x = y^n \}$$

we define

$$f(n, x) = \begin{cases} \downarrow \ \sqrt[n]{x} & \text{if } \exists y \text{ st. } x = y^n \\ \uparrow & \text{otherwise} \end{cases}$$

$$= \mu y.\ \text{"} y^n = x \text{"}$$

$$= \mu y.\ |y^n - x|$$

computable

By the (corollary of the) smn theorem $\exists k : \mathbb{N} \to \mathbb{N}$ total computable s.t. $\forall n, x \in \mathbb{N}$

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

Observe that

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

in fact

$$x \in W_{k(n)} \quad \text{iff} \quad \underbrace{\varphi_{k(n)}(x)}_{} \downarrow \quad \text{iff} \quad \exists y.\ x = y^n$$

$$\overset{f(n,x)}{\parallel}$$

$\square$

EXERCISE: show that there is a total computable function $s : \mathbb{N} \to \mathbb{N}$ s.t.

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

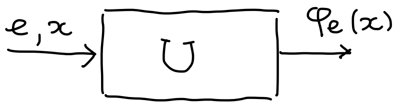
[HOME]

* UNIVERSAL FUNCTION

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

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

Is it computable ?



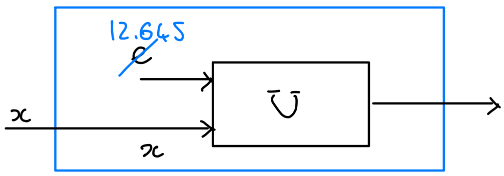
(executes $P_e = \gamma^{-1}(e)$ over x)

when e varies on the natural numbers

$\Psi_U(0, \_) \qquad \Psi_U(1, \_) \qquad \Psi_U(2, \_)$

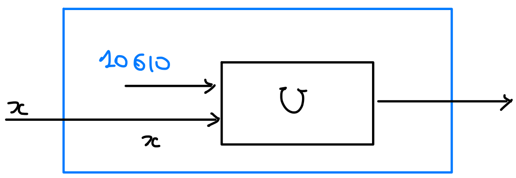
$\varphi_0 \qquad \varphi_1 \qquad \varphi_2$

Turing s.p.a.



12.645

$ 1.000.000



10.610

$ 1.000.000

⋮

Theorem (Universal Program) :

Let $k \geq 1$. Then the universal function

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

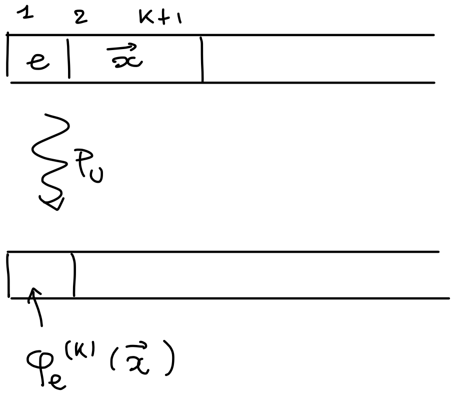
$$\Psi_U(e, \vec{x}) = \varphi_e^{(k)}(\vec{x})$$

is computable

proof

fix $k \geq 1$

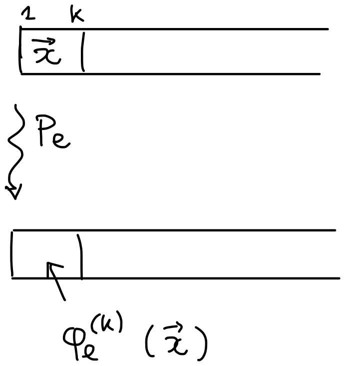
given e, $\vec{x}$



how can P_U work

→ determine $P_e = \gamma^{-1}(e)$

→



by Church-Turing thesis computable

unsatisfactory!

(more to come in the next lesson)