COMPUTABILITY (03/11/2026)

* Parametrisation (smn) Theorem

program $e \in \mathbb{N}$ $P_e = \gamma^{-1}(e)$ and $\varphi_e^{(z)}(x, y)$

for any fixed $x \in \mathbb{N}$ one obtains a function of one argument y

$x = 0$ $\quad y \mapsto \varphi_e^{(z)}(0, y)$

$x = 1$ $\quad y \mapsto \varphi_e^{(z)}(1, y)$

$\vdots$ $\quad \vdots$

smn - theorem : for each fixed $x \in \mathbb{N}$, the program computing the function of y defined as above $y \mapsto \varphi_e^{(z)}(x, y)$ can be constructed algorithmically using P_e and x

$P_e(x, y)$:

x

y

return ...

$x = x_0$

$P_e(\cancel{x}, y)$:

x_0

x

y

return ...

generally $\varphi_e^{(m+n)} : \mathbb{N}^{m+n} \to \mathbb{N}$

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

↑ total computable

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 $e \in \mathbb{N}$, $\vec{x} \in \mathbb{N}^m$, $\vec{y} \in \mathbb{N}^n$

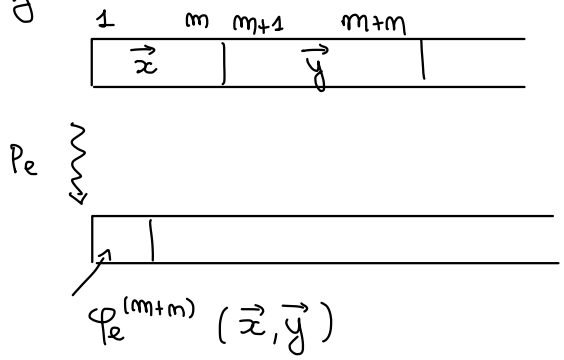
$$\varphi_e^{(m+n)}(\vec{x}, \vec{y}) = \varphi_{s_{m,n}(e,\vec{x})}^{(n)}(\vec{y})$$

proof

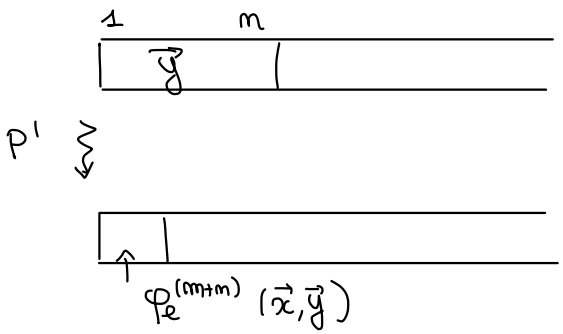
intuitively given $e \in \mathbb{N}$, $\vec{x} \in \mathbb{N}^m$

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

program P_e does the following



function $S_{m,m}$ "must construct" a program P' (depending on $e, \vec{x}$)



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

P'

```
T(m, m+m)        // move y to m+1, ..., m+m
⋮
T(1, m+1)

Z(1)             // write x_1 to R_1
S(1)
⋮                x_1 times
S(1)

⋮

Z(m)             // write x_m to R_m
S(m)
⋮                x_m times
S(m)

P_e = γ^{-1}(e)
```

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

formally

(1) sequential composition of programs is computable

$$e_1, e_2 \quad \rightsquigarrow \quad \gamma\begin{pmatrix} P_{e_1} \\ P_{e_2} \end{pmatrix}$$

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

$$upd(e,h) = \gamma\left(\begin{array}{l}\text{program obtained from } P_e = \gamma^{-1}(e) \\ \text{by updating each jump } J(m,n,t) \rightsquigarrow J(m,n,t+h)\end{array}\right)$$

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

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

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

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

now

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

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

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

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

$a(e,i)$ = component i of tuple for $1 \le i \le \ell(e)$

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

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

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

$= \dots$

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

$$seq(e_1, e_2) = \gamma \begin{pmatrix} Pe_1 \\ Pe_2 \end{pmatrix} = c(e_1, upd(e_2, \ell(e_1)))$$

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

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

$$= \dots$$

③ $set : \mathbb{N}^2 \to \mathbb{N}$

$$set(i, x) = \gamma \begin{pmatrix} Z(i) \\ S(i) \\ \vdots \\ S(i) \end{pmatrix} \left.\vphantom{\begin{matrix} a \\ b \\ c \\ d \end{matrix}}\right\} x \text{ times} \quad (*)$$

$$= \dots$$

$(*)$ $\begin{cases} T(n, m+n) \\ \vdots \\ T(1, m+1) \end{cases}$ P'

$(*)$ $\begin{cases} Z(1) \\ S(1) \\ \vdots \\ S(1) \end{cases}$ x_1 times

$\vdots$

$(*)$ $\begin{cases} Z(m) \\ S(m) \\ \vdots \\ S(m) \end{cases}$ x_m times

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

④ finally

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

$$= seq(transf(m, n), seq(set(1, x_1), seq(set(2, x_2), \dots seq(set(m, x_m), e))\dots)$$

brackets

composition of computable functions is computable

(primitive recursive) □

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

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

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

$$f(\vec{x}, \vec{y}) = \varphi_{s(\vec{x})}(\vec{y})$$

proof

since f is computable $f = \varphi_e^{(m+m)}$ for some $e \in \mathbb{N}$

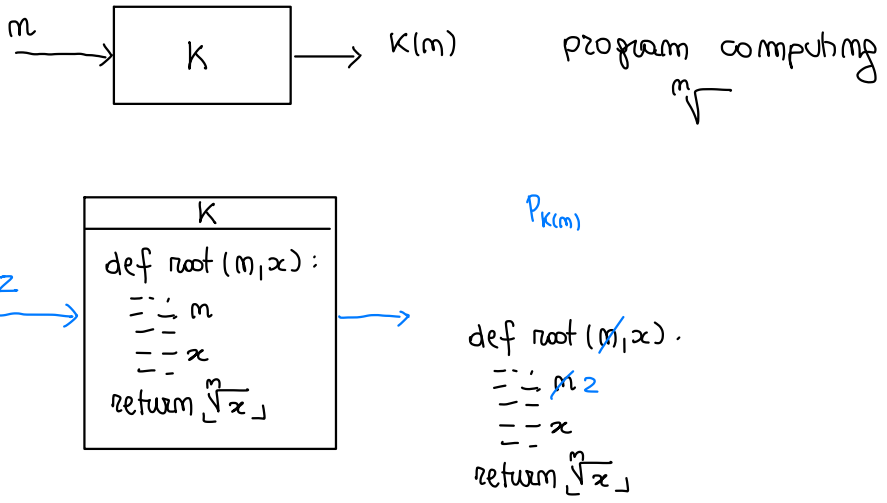
by smn theorem

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

which is what we want, with $s(\vec{x}) = s_{m,m}(e,\vec{x})$ □

EXERCISE: Prove that there exists a total computable function $K: \mathbb{N} \to \mathbb{N}$ such that $\forall m \in \mathbb{N}$, $x \in \mathbb{N}$

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



the function

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

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

$= \max z \;.\; z^m \leq x$

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

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

⇓

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

$$\varphi_{K(m)}(x) = f(m,x) = \lfloor \sqrt[m]{x} \rfloor \qquad \forall m,x \in \mathbb{N}$$

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

$\forall m \in \mathbb{N}$ $\varphi_{K(m)} : \mathbb{N} \to \mathbb{N}$ is defined only on m^{th} powers

(y^m for some $y \in \mathbb{N}$)

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

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

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

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

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

By (corollary of) the smn theorem there is $K : \mathbb{N} \to \mathbb{N}$ total computable s.t. $\forall m, x \in \mathbb{N}$

$$\varphi_{K(m)}(x) = f(m, x) = \begin{cases} \sqrt[m]{x} & \text{if } x = y^m \text{ for some } y \\ \uparrow & \text{otherwise} \end{cases}$$

Observe that indeed $W_{K(m)} = \{ x \mid \exists y . \; x = y^m \}$

in fact

$x \in W_{K(m)}$ iff $\varphi_{K(m)}(x) \downarrow$ iff $\exists y \; x = y^m$

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

EXERCISE : Show that there is a total computable function $S : \mathbb{N} \to \mathbb{N}$

s.t.

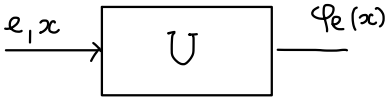
$$W^{(K)}_{S(m)} = \{ (y_1, \dots, y_K) \mid \sum_{i=1}^{K} y_i = m \}$$

UNIVERSAL FUNCTION

Let $\psi_U : \mathbb{N}^2 \to \mathbb{N}$

$$\psi_U(e, x) = \varphi_e(x)$$ well-defined

Is it computable ?



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

when e ranges over natural numbers

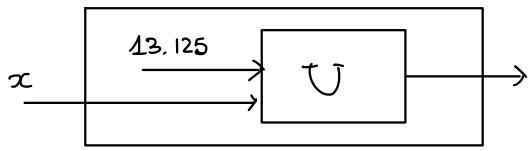
$\psi_U(0, -)$ | φ_0

$\psi_U(1, -)$ | φ_1

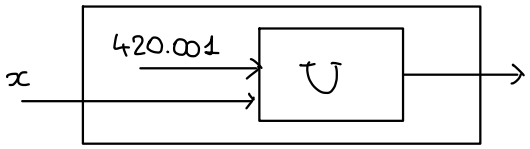
$\psi_U(2, -)$ | φ_2

Turing company

LLM math data



web service



⋮

Theorem (Universal program)

Let $k \geq 1$

Then there the universal function

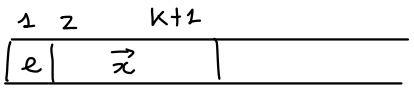
$$\psi_U^{(k)} : \mathbb{N}^{k+1} \to \mathbb{N}$$

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

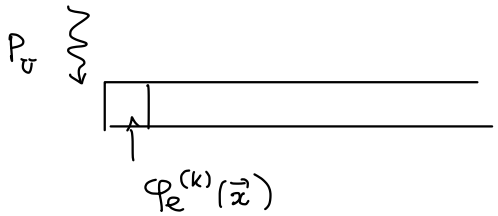
is computable

proof

fix $k \geq 1$

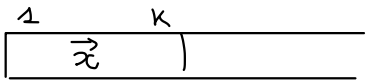


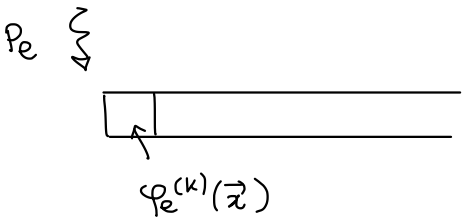
given e, $\vec{x}$



how is P_U working

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





by Church - Turing thesis

Ψ_U computable

UNSATISFACTORY

(formal proof in the next lesson)