Parametrisation Theorem

\[ \varphi_e^{(2)}(x, y) \] computed by \[ \varphi_e = y^{-1}(e) \]

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

\[
\begin{align*}
    x = 0 & \quad y \mapsto \varphi_e^{(2)}(0, y) \\
    x = 1 & \quad y \mapsto \varphi_e^{(2)}(1, y) \\
    \vdots & \quad \vdots
\end{align*}
\]

the program which computes the functions above for each fixed \( x \)
can be obtained algorithmically starting from \( \varphi_e \)

\[
\begin{array}{c}
    \varphi_e(x, y) \\
    \vdots
\end{array}
\]

more generally \( f: \mathbb{N}^m \rightarrow \mathbb{N} \)

\[
\varphi_e^{(m+m)}(\vec{x}, \vec{y}) = \varphi^{(m)}_{\delta(e, \vec{x})}(\vec{y})
\]

Theorem (smm theorem):

Given \( m, m > 1 \) there is a total computable function

\[ S_{mm}: \mathbb{N}^{m+1} \rightarrow \mathbb{N} \]

such that for all \( \vec{x} \in \mathbb{N}^m \), \( \vec{y} \in \mathbb{N}^m \), \( e \in \mathbb{N} \)

\[
\varphi_e^{(m+m)}(\vec{x}, \vec{y}) = \varphi_{S_{mm}(e, \vec{x})}(\vec{y})
\]

Proof

intuitively given \( e \in \mathbb{N} \)

\[
\varphi_e = y^{-1}(e)
\]
starting from

\[
\begin{array}{c|c|c|c}
1 & m & m+1 & m+m \\
\overline{x} & \overline{y} & & 0
\end{array}
\]

you want, for each \( z \in \mathbb{N}^m \) fixed, a program \( \overline{p}' \) depending on \( e, z \)

\[
\begin{array}{c|c|c|c}
1 & \ldots & m & \\
\overline{z} & \overline{y} & & \\
\end{array}
\]

\( \overline{p}' \) has to
- move \( \overline{y} \) to \( m+1 \ldots m+m \)
- write \( \overline{x} \) \( m \) \( m \) \( m \)
- execute \( \overline{p} \)

\[
\begin{array}{l}
T(m, m+m) \quad \overline{p}' \quad \text{// move \( y_m \) to} \quad R_{m+m} \\
\vdots \\
T(1, m+1) \quad \text{// move \( y_2 \) to} \quad R_{m+1} \\
\vdots \\
Z(1) \end{array}
\]

\[
\begin{array}{l}
S(1) \quad \text{// write} \quad x_1 \quad \text{\( m \) times} \\
\vdots \\
S(m) \quad \text{// write} \quad x_m \quad \text{\( m \) times} \\
\vdots \\
S(1) \\
\end{array}
\]

\[
\overline{p} = \gamma^{-1}(e)
\]

\[
S(e, \overline{x}) = \gamma(\overline{p}')
\]
\[ (\text{sequential composition of programs}) \quad (e_1, e_2) \mapsto \gamma \left( \frac{P_{e_1}}{P_{e_2}} \right) \]

\[(1a) \quad \text{upd} : N^2 \rightarrow N \]

\[
\text{upd}(e, h) = \gamma \left( \text{program obtained from } P_e = g^{-1}(e) \right.
\]

(by updating all jump instructions $J(m, m, i) \mapsto J(m, m, i + h))

\[
\tilde{\text{upd}}(i, h) = \beta \left( \text{instruction obtained from } P_e^{-1}(i), \text{ updating the } \right)
\]

\[\text{target if it is a jump} \]

\[\text{note: } \beta(J(m, m, i)) = \gamma(m-1, m-2, 6-1) \times 4 + 3 \]

\[
\begin{cases}
  i & \text{if } \text{rem}(4, i) \neq 3 \\
  \gamma(v_2(q), v_2(q), v_3(q + h)) \times 4 + 3 & \text{if } \text{rem}(4, i) = 3 \\
  q = q_4(4, i) 
\end{cases}
\]

\[
= i \times \sigma_3(12 \text{rem}(4, i) - 3 1) + \\
\gamma(v_2(q), v_2(q), v_3(q + h)) \times 4 + 3 \times \sigma_3(12 \text{rem}(4, i) - 3 1)
\]

\[\text{now} \]

\[
\text{upd}(e, h) = \tau \left( \tilde{\text{upd}}(a(e, 1), h), \tilde{\text{upd}}(a(e, 2), h), \tilde{\text{upd}}(a(e, P(e)), h) \right)
\]

\[
= \prod_{i=1}^{l(e)-1} \frac{\sigma_2(a(e, i), h)}{p_i} \cdot \tilde{\text{upd}}(a(e, P(e)), h) + 2 
\]

\[
\tau(v_1, \ldots, v_m) = \prod_{i=1}^{m-1} p_i^{y_i} \cdot p_m^{y_{m+1}} = 2
\]

\[e(e) = \text{length of the encoded sequence} \]

\[1 \leq i \leq e(e) \quad a(e, i) = i^{th} \text{ component} \]

\[ \cdot \quad \text{c} : N^2 \rightarrow N \]

\[c(e_1, e_2) = \tau(a(e_1, 1), \ldots, a(e_1, l(e_1)), a(e_2, 1), \ldots, a(e_2, l(e_2))) \]

\[\cdot \quad \text{seq} : N^2 \rightarrow N \]

\[\text{seq}(e_1, e_2) = \gamma \left( \frac{P_{e_1}}{P_{e_2}} \right) = c(e_1, \text{upd}(e_2, l(e_2))) \]
2. \text{transf} : \mathbb{N}^2 \rightarrow \mathbb{N}
\text{transf} (m, m) = \begin{pmatrix}
T(m, m+m) \\
T(1, m+1)
\end{pmatrix}

3. \text{set} : \mathbb{N}^2 \rightarrow \mathbb{N}
\text{set} (i, x) = \begin{pmatrix}
\exists (i) \\
S(i) \\
S(i)
\end{pmatrix} \times \text{times}

4. \text{finally}
\begin{align*}
\text{s}_{m,m} (e, x) &= \\
&= \text{seq} (\text{transf} (m, m), \\
&\hspace{1cm} \text{seq (set (1, x),} \\
&\hspace{2cm} \ldots \\
&\hspace{1cm} \text{seq (set (m, x_m), e)} \ldots )
\end{align*}

\text{computable (actually primitive recursive) since it is a composition of prim. rec. funct.}

\text{Corollary: Let } f : \mathbb{N}^{m+m} \rightarrow \mathbb{N} \text{ be a computable function. Then there is a total computable function } S : \mathbb{N}^m \rightarrow \mathbb{N} \\
\text{s.t. } \forall \bar{z} \in \mathbb{N}^m, \bar{y} \in \mathbb{N}^m
\begin{align*}
f(\bar{z}, \bar{y}) &= \Phi^{(m)}_{S(\bar{z})} (\bar{y})
\end{align*}

\text{Proof}
\begin{align*}
\text{since } f \text{ is computable there is } e \in \mathbb{N} \text{ s.t. } f &= \Phi^{(m+m)}_e \\
f(\bar{z}, \bar{y}) &= \Phi^{(m+m)}_e (\bar{z}, \bar{y}) = \Phi^{(m)}_{\text{s}_{m,m} (e, \bar{z})} (\bar{y}) \forall \bar{z}, \bar{y}
\end{align*}

\text{we conclude by setting } S(\bar{z}) = \text{s}_{m,m} (e, \bar{z})
Prove that there is a total computable function $k : \mathbb{N} \to \mathbb{N}$ such that

$$
\forall m \in \mathbb{N} \quad \forall x \in \mathbb{N} \quad (m)(x) = \lceil \sqrt[m]{x} \rceil
$$

The function

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

$$
f(m, x) = \lceil \sqrt[m]{x} \rceil
$$

$$
= \max \ Z \quad \text{"} Z^m \leq x \text{"}
$$

$$
= \min \ Z \quad \text{"} (Z+1)^m > x \text{"}
$$

$$
= \mu Z \leq x, \quad x + 1 = (Z+1)^m
$$

is computable.

By (unnamed theorem) there is $k : \mathbb{N} \to \mathbb{N}$ total computable s.t.

$$
\phi_{k(m)}(x) = f(m, x) = \lceil \sqrt[m]{x} \rceil
$$
EXAMPLE: There is a total computable function $K : \mathbb{N} \rightarrow \mathbb{N}$ st. 

$\forall m \phi_{k(m)}$ is defined only on $m^{th}$ powers 
(on $y^m$ for $y \in \mathbb{N}$)

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

we define

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

= $\mu y. \ "y^m = x"$

= $\mu y. \ \lfloor y^m - x \rfloor$

computable

By the (corollary of the) same theorem $\exists K : \mathbb{N} \rightarrow \mathbb{N}$ total computable s.t. $\forall m, x \in \mathbb{N}$

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

Observe that

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

In fact 

$$x \in W_{k(m)} \iff \phi_{k(m)}(x) \downarrow \iff \exists y. x = y^m$$

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

$$W_{S(z)}^{(k)} = \{ (y_1, ..., y_k) \mid \sum_{i=1}^{k} y_i = x \}$$

[Homework]
*Universal Function*

\[
\psi_0 : \mathbb{N}^2 \rightarrow \mathbb{N}
\]

\[
\psi_0(e, x) = \varphi_x(x) \quad \text{well-defined}
\]

Is it computable?

\[
\begin{array}{ccc}
e, x & \rightarrow & \mathcal{U}
\end{array}
\]

\[
(\text{execute } \varphi_e(x) \text{ over } x)
\]

where \(e\) varies on the natural numbers

\[
\psi_0(0, \_), \psi_0(1, \_), \psi_0(2, \_)
\]

\[
\vdots \quad \vdots \quad \vdots
\]

\[
\varphi_0, \varphi_1, \varphi_2
\]

**Theorem (Universal Program):**

Let \(k \geq 1\). Then the universal function

\[
\psi_0 : \mathbb{N}^{k+1} \rightarrow \mathbb{N}
\]

\[
\psi_0(e, x) = \varphi_x^{(k)}(x)
\]

is computable.
proof

fix \( k \geq 1 \)

given \( e, \bar{x} \)

\[
\begin{array}{c|c}
1 & 2 \\
\hline
e & \bar{x} \\
\hline
\end{array}
\]

\[ \not\exists \sigma \subseteq P_0 \]

\[ \sigma \]

\[ \varphi_{P_0}^{(k)}(\bar{x}) \]

how can \( P_0 \) work

\[ \rightarrow \text{determine } P_e = \varphi^{-1}(e) \]

\[ \begin{array}{c|c}
1 & 2 \\
\hline
\bar{x} & \_ \\
\hline
\end{array} \]

\[ \exists \sigma \subseteq P_e \]

\[ \sigma \]

\[ \varphi_{P_e}^{(k)}(\bar{x}) \]

by Church-Turing Thesis

computable

unsatisfactory!

(\text{more to come in the next lesson})