Coding programs as numbers

Turing/Church solution of the Entscheidungsproblem uses the idea that (formal descriptions of) algorithms can be the data on which algorithms act.

To realize this idea with Register Machines we have to be able to code RM programs as numbers. (In general, such codings are often called Gödel numberings.)

To do that, first we have to code pairs of numbers and lists of numbers as numbers. There are many ways to do that. We fix upon one. . .

"Effective" numerical codes

RM program

initial contents of R1, ..., Rn

final contents of R0 (if halts)

$$\mathrm{Prog},\ [x_1, \ldots, x_n] \longmapsto y$$

run the RM

"Effective" numerical codes

$$\text{Prog}, [x_1, \ldots, x_n] \longmapsto y$$

code $\downarrow$ $\uparrow$ decode

$$\langle \ulcorner \text{Prog} \urcorner, \ulcorner [x_1, \ldots, x_n] \urcorner \rangle$$

a number

want numerical codings

$$\langle -,- \rangle, \ulcorner - \urcorner, \ulcorner [-, \ldots, -] \urcorner$$

so that

$$\cdot \xrightarrow{\text{decode}} \cdot \xmapsto{\text{run}} \cdot$$

is RM computable

Numerical coding of pairs

$\{0,1,2,3,\dots\}$

For $x, y \in \mathbb{N}$, define $\begin{cases} \langle\!\langle x, y \rangle\!\rangle \triangleq 2^x(2y+1) \\ \langle x, y \rangle \triangleq 2^x(2y+1) - 1 \end{cases}$

"equals, by definition"

Numerical coding of pairs

For $x, y \in \mathbb{N}$, define $\begin{cases} \langle\langle x, y\rangle\rangle \triangleq 2^x(2y+1) \\ \langle x, y\rangle \triangleq 2^x(2y+1) - 1 \end{cases}$

So

$$\boxed{0\mathrm{b}\langle\langle x, y\rangle\rangle} = \boxed{0\mathrm{b}y \mid 1 \mid \underbrace{0\cdots0}_{x \text{ 0s}}}$$

$$\boxed{0\mathrm{b}\langle x, y\rangle} = \boxed{0\mathrm{b}y \mid 0 \mid \underbrace{1\cdots1}_{x \text{ 1s}}}$$

(Notation: $0\mathrm{b}x \triangleq x$ *in binary*.)

E.g. $27 = 0\mathrm{b}11011 = \langle\langle 0, 13\rangle\rangle = \langle 2, 3\rangle$

Numerical coding of pairs

For $x, y \in \mathbb{N}$, define $\begin{cases} \langle\langle x, y \rangle\rangle & \triangleq & 2^x(2y+1) \\ \langle x, y \rangle & \triangleq & 2^x(2y+1) - 1 \end{cases}$

So

$$\boxed{0\mathrm{b}\langle\langle x, y \rangle\rangle} = \boxed{0\mathrm{b}y \mid 1 \mid 0\cdots 0}$$

$$\boxed{0\mathrm{b}\langle x, y \rangle} = \boxed{0\mathrm{b}y \mid 0 \mid 1\cdots 1}$$

$\langle -, - \rangle$ gives a bijection (one-one correspondence) between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$.

$\langle\langle -, - \rangle\rangle$ gives a bijection between $\mathbb{N} \times \mathbb{N}$ and $\{n \in \mathbb{N} \mid n \neq 0\}$.

Numerical coding of lists

$\boldsymbol{list}\,\mathbb{N} \triangleq$ set of all finite lists of natural numbers, using ML notation for lists:

- empty list: $[]$
- list-cons: $x :: \ell \in \boldsymbol{list}\,\mathbb{N}$ (given $x \in \mathbb{N}$ and $\ell \in \boldsymbol{list}\,\mathbb{N}$)
- $[x_1, x_2, \ldots, x_n] \triangleq x_1 :: (x_2 :: (\cdots x_n :: [] \cdots))$

Numerical coding of lists

$\mathit{list}\,\mathbb{N} \triangleq$ set of all finite lists of natural numbers, using ML notation for lists.

For $\ell \in \mathit{list}\,\mathbb{N}$, define $\ulcorner \ell \urcorner \in \mathbb{N}$ by induction on the length of the list ℓ:

$$\begin{cases} \ulcorner [] \urcorner & \triangleq \ \mathbf{0} \\ \ulcorner x :: \ell \urcorner & \triangleq \ \langle\langle x, \ulcorner \ell \urcorner \rangle\rangle = 2^x (2 \cdot \ulcorner \ell \urcorner + 1) \end{cases}$$

Thus $\ulcorner [x_1, x_2, \ldots, x_n] \urcorner = \langle\langle x_1, \langle\langle x_2, \cdots \langle\langle x_n, 0 \rangle\rangle \cdots \rangle\rangle \rangle\rangle$

Numerical coding of lists

$list\,\mathbb{N} \triangleq$ set of all finite lists of natural numbers, using ML notation for lists.

For $\ell \in list\,\mathbb{N}$, define $\ulcorner \ell \urcorner \in \mathbb{N}$ by induction on the length of the list ℓ:

$$\begin{cases} \ulcorner [] \urcorner & \triangleq \; 0 \\ \ulcorner x :: \ell \urcorner & \triangleq \; \langle\!\langle x, \ulcorner \ell \urcorner \rangle\!\rangle = 2^x (2 \cdot \ulcorner \ell \urcorner + 1) \end{cases}$$

For example:

$\ulcorner [3] \urcorner = \ulcorner 3 :: [] \urcorner = \langle\!\langle 3, 0 \rangle\!\rangle = 2^3 (2 \cdot 0 + 1) = 8 = 0\text{b}1000$

$\ulcorner [1,3] \urcorner = \langle\!\langle 1, \ulcorner [3] \urcorner \rangle\!\rangle = \langle\!\langle 1, 8 \rangle\!\rangle = 34 = 0\text{b}100010$

$\ulcorner [2,1,3] \urcorner = \langle\!\langle 2, \ulcorner [1,3] \urcorner \rangle\!\rangle = \langle\!\langle 2, 34 \rangle\!\rangle = 276 = 0\text{b}100010100$

Numerical coding of lists

$list\,\mathbb{N} \triangleq$ set of all finite lists of natural numbers, using ML notation for lists.

For $\ell \in list\,\mathbb{N}$, define $\ulcorner \ell \urcorner \in \mathbb{N}$ by induction on the length of the list ℓ:

$$\begin{cases} \ulcorner [] \urcorner & \triangleq & 0 \\ \ulcorner x :: \ell \urcorner & \triangleq & \langle\!\langle x, \ulcorner \ell \urcorner \rangle\!\rangle = 2^x(2 \cdot \ulcorner \ell \urcorner + 1) \end{cases}$$

For example:

$$\ulcorner [3] \urcorner = \ulcorner 3 :: [] \urcorner = \langle\!\langle 3, 0 \rangle\!\rangle = 2^3(2 \cdot 0 + 1) = 8 = 0\mathrm{b}1000$$

$$\ulcorner [1,3] \urcorner = \langle\!\langle 1, \ulcorner [3] \urcorner \rangle\!\rangle = \langle\!\langle 1, 8 \rangle\!\rangle = 34 = 0\mathrm{b}100010$$

$$\ulcorner [2,1,3] \urcorner = \langle\!\langle 2, \ulcorner [1,3] \urcorner \rangle\!\rangle = \langle\!\langle 2, 34 \rangle\!\rangle = 276 = 0\mathrm{b}100010100$$

Numerical coding of lists

$\mathit{list}\,\mathbb{N} \triangleq$ set of all finite lists of natural numbers, using ML notation for lists.

For $\ell \in \mathit{list}\,\mathbb{N}$, define $\ulcorner \ell \urcorner \in \mathbb{N}$ by induction on the length of the list ℓ:

$$\begin{cases} \ulcorner [] \urcorner & \triangleq \; \mathbf{0} \\ \ulcorner x :: \ell \urcorner & \triangleq \; \langle\langle x, \ulcorner \ell \urcorner \rangle\rangle = \mathbf{2}^{x}(\mathbf{2} \cdot \ulcorner \ell \urcorner + \mathbf{1}) \end{cases}$$

$$\boxed{0\mathrm{b}\ulcorner [x_1, x_2, \ldots, x_n] \urcorner} = \boxed{1 \mid 0 \cdots 0}\,\boxed{1 \mid 0 \cdots 0} \cdots \boxed{1 \mid 0 \cdots 0}$$

x_n 0s x_{n-1} 0s x_1 0s

Numerical coding of lists

$\mathit{list}\,\mathbb{N} \triangleq$ set of all finite lists of natural numbers, using ML notation for lists.

For $\ell \in \mathit{list}\,\mathbb{N}$, define $\ulcorner \ell \urcorner \in \mathbb{N}$ by induction on the length of the list ℓ:

$$\begin{cases} \ulcorner [] \urcorner & \triangleq \mathbf{0} \\ \ulcorner x :: \ell \urcorner & \triangleq \langle\!\langle x, \ulcorner \ell \urcorner \rangle\!\rangle = \mathbf{2}^x(\mathbf{2} \cdot \ulcorner \ell \urcorner + \mathbf{1}) \end{cases}$$

$$\boxed{0\mathrm{b}\ulcorner [x_1, x_2, \ldots, x_n] \urcorner} = \boxed{\mathbf{1} \mid \mathbf{0}\cdots\mathbf{0}}\,\boxed{\mathbf{1} \mid \mathbf{0}\cdots\mathbf{0}}\cdots\boxed{\mathbf{1} \mid \mathbf{0}\cdots\mathbf{0}}$$

Hence $\ell \mapsto \ulcorner \ell \urcorner$ gives a bijection from $\mathit{list}\,\mathbb{N}$ to $\mathbb{N}$.

Numerical coding of programs

If P is the RM program $\boxed{\begin{array}{c} L_0 : body_0 \\ L_1 : body_1 \\ \vdots \\ L_n : body_n \end{array}}$

then its numerical code is

$$\ulcorner P \urcorner \triangleq \ulcorner [\ulcorner body_0 \urcorner, \ldots, \ulcorner body_n \urcorner] \urcorner$$

where the numerical code $\ulcorner body \urcorner$ of an instruction body is defined by:

$$\begin{cases} \ulcorner R_i^+ \to L_j \urcorner & \triangleq \langle\langle 2i, j \rangle\rangle \\ \ulcorner R_i^- \to L_j, L_k \urcorner & \triangleq \langle\langle 2i+1, \langle j, k \rangle \rangle\rangle \\ \ulcorner \mathtt{HALT} \urcorner & \triangleq 0 \end{cases}$$

Any $x \in \mathbb{N}$ decodes to a unique instruction $body(x)$:

if $x = 0$ then $body(x)$ is $\mathtt{HALT}$,
else ($x > 0$ and) let $x = \langle\langle y, z \rangle\rangle$ in
  if $y = 2i$ is even, then
    $body(x)$ is $\mathtt{R}_i^+ \to \mathtt{L}_z$,
  else $y = 2i + 1$ is odd, let $z = \langle j, k \rangle$ in
    $body(x)$ is $\mathtt{R}_i^- \to \mathtt{L}_j, \mathtt{L}_k$

So any $e \in \mathbb{N}$ decodes to a unique program $prog(e)$, called the register machine program with index e:

$$prog(e) \triangleq \boxed{\begin{array}{c} \mathtt{L}_0 : body(x_0) \\ \vdots \\ \mathtt{L}_n : body(x_n) \end{array}} \text{ where } e = \ulcorner [x_0, \ldots, x_n] \urcorner$$

Example of $prog(e)$

- $786432 = 2^{19} + 2^{18} = 0\mathtt{b}11\underbrace{0\ldots0}_{18\ ''0''s} = \ulcorner[18, 0]\urcorner$
- $18 = 0\mathtt{b}10010 = \langle\!\langle 1, 4\rangle\!\rangle = \langle\!\langle 1, \langle 0, 2\rangle\rangle\!\rangle = \ulcorner \mathtt{R}_0^- \to \mathtt{L}_0, \mathtt{L}_2 \urcorner$
- $0 = \ulcorner \mathtt{HALT} \urcorner$

So $prog(786432) = \boxed{\begin{array}{l} \mathtt{L}_0 : \mathtt{R}_0^- \to \mathtt{L}_0, \mathtt{L}_2 \\ \mathtt{L}_1 : \mathtt{HALT} \end{array}}$

Example of $prog(e)$

- $786432 = 2^{19} + 2^{18} = 0b11\underbrace{0\ldots0}_{18\ \text{"0"s}} = \ulcorner[18, 0]\urcorner$
- $18 = 0b10010 = \langle\langle 1, 4\rangle\rangle = \langle\langle 1, \langle 0, 2\rangle\rangle\rangle = \ulcorner R_0^- \to L_0, L_2\urcorner$
- $0 = \ulcorner \mathtt{HALT} \urcorner$

So $prog(786432) =$

$$\boxed{\begin{array}{l} L_0 : R_0^- \to L_0, L_2 \\ L_1 : \mathtt{HALT} \end{array}}$$

N.B. jump to label with no body (erroneous halt)

What function is computed by a RM with prog(786432) as its program?

Example of $prog(e)$

- $786432 = 2^{19} + 2^{18} = 0b11\underbrace{0\ldots0}_{18\ \text{"0"}s} = \ulcorner[18, 0]\urcorner$
- $18 = 0b10010 = \langle\!\langle 1, 4\rangle\!\rangle = \langle\!\langle 1, \langle 0, 2\rangle\rangle\!\rangle = \ulcorner R_0^- \to L_0, L_2\urcorner$
- $0 = \ulcorner \mathtt{HALT} \urcorner$

So $prog(786432) =$ $\boxed{\begin{array}{l} L_0 : R_0^- \to L_0, L_2 \\ L_1 : \mathtt{HALT} \end{array}}$

N.B. In case $e = 0$ we have $0 = \ulcorner[\,]\urcorner$, so $prog(0)$ is the program with an empty list of instructions, which by convention we regard as a RM that does nothing (i.e. that halts immediately).

$$666 = \color{red}{0b}\,1010011010$$

$$= \ulcorner [\,1, 1, 0, 2, 1\,] \urcorner$$

prog(666) =

$$
\begin{array}{|l|}
\hline
L_0 : R_0^+ \to L_0 \\
L_1 : R_0^+ \to L_0 \\
L_2 : \text{HALT} \\
L_3 : R_0^- \to L_0, L_0 \\
L_4 : R_0^+ \to L_0 \\
\hline
\end{array}
$$

(never halts!)

What partial function does this compute?

"Effective" numerical codes

$$\text{Prog}, [x_1, \ldots, x_n] \longmapsto y$$

code $\downarrow$ $\uparrow$ decode

$$\langle \ulcorner \text{Prog} \urcorner, \ulcorner [x_1, \ldots, x_n] \urcorner \rangle$$

a number

Want numerical codings

$$\langle -,- \rangle, \ulcorner - \urcorner, \ulcorner [-, \ldots, -] \urcorner$$

so that

$$\cdot \xrightarrow{\text{decode}} \cdot \xmapsto{\text{run}} \cdot$$

is RM computable

Universal register machine, U

High-level specification

Universal RM U carries out the following computation, starting with $\mathtt{R}_0 = 0$, $\mathtt{R}_1 = e$ (code of a program), $\mathtt{R}_2 = a$ (code of a list of arguments) and all other registers zeroed:

- decode e as a RM program P
- decode a as a list of register values $a_1, \ldots, a_n$
- carry out the computation of the RM program P starting with $\mathtt{R}_0 = 0, \mathtt{R}_1 = a_1, \ldots, \mathtt{R}_n = a_n$ (and any other registers occurring in P set to 0).

Mnemonics for the registers of U and the role they play in its program:

$R_1 \equiv P$ code of the RM to be simulated

$R_2 \equiv A$ code of current register contents of simulated RM

$R_3 \equiv PC$ program counter—number of the current instruction (counting from 0)

$R_4 \equiv N$ code of the current instruction body

$R_5 \equiv C$ type of the current instruction body

$R_6 \equiv R$ current value of the register to be incremented or decremented by current instruction (if not HALT)

$R_7 \equiv S$, $R_8 \equiv T$ and $R_9 \equiv Z$ are auxiliary registers.

R_0 result of the simulated RM computation (if any).

Overall structure of U's program

1 copy PCth item of list in P to N (halting if PC > length of list); goto 2

2 if $N = 0$ then halt, else decode N as $\langle\langle y, z\rangle\rangle$; $C ::= y$; $N ::= z$; goto 3

{at this point either $C = 2i$ is even and current instruction is $R_i^+ \to L_z$, or $C = 2i + 1$ is odd and current instruction is $R_i^- \to L_j, L_k$ where $z = \langle j, k\rangle$}

3 copy ith item of list in A to R; goto 4

4 execute current instruction on R; update PC to next label; restore register values to A; goto 1

Overall structure of U's program

[1] copy $\mathtt{PC}$th item of list in $\mathtt{P}$ to $\mathtt{N}$ (halting if $\mathtt{PC} >$ length of list); goto [2]

[2] if $\mathtt{N} = 0$ then halt, else decode $\mathtt{N}$ as $\langle\!\langle y, z \rangle\!\rangle$; $\mathtt{C} ::= y$; $\mathtt{N} ::= z$; goto [3]

{at this point either $\mathtt{C} = 2i$ is even and current instruction is $\mathtt{R}_i^+ \to \mathtt{L}_z$,
or $\mathtt{C} = 2i + 1$ is odd and current instruction is $\mathtt{R}_i^- \to \mathtt{L}_j, \mathtt{L}_k$ where $z = \langle j, k \rangle$}

[3] copy ith item of list in $\mathtt{A}$ to $\mathtt{R}$; goto [4]

[4] execute current instruction on $\mathtt{R}$; update $\mathtt{PC}$ to next label; restore register values to $\mathtt{A}$; goto [1]

To implement this, we need RMs for manipulating (codes of) lists of numbers. . .