Coding programs as numbers

There's no H such that $H(A,D) = \begin{cases} 1 & \text{if } A(D)\downarrow \\ 0 & \text{otherwise.} \end{cases}$ for all (A,D).

Informal proof, by contradiction. If there were such an H, let C be the algorithm:

> *"input A; compute $H(A,A)$; if $H(A,A) = 0$ then return 1, else loop forever."*

why is A a "datum on which A is designed to operate"?

So $\forall A\,(C(A)\downarrow \leftrightarrow H(A,A) = 0)$ (since H is total)
and $\forall A\,(H(A,A) = 0 \leftrightarrow \neg A(A)\downarrow)$ (definition of H).
So $\forall A\,(C(A)\downarrow \leftrightarrow \neg A(A)\downarrow)$.
Taking A to be C, we get $C(C)\downarrow \leftrightarrow \neg C(C)\downarrow$, contradiction!

Computable functions

Recall:

Definition. $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ is (register machine) computable if there is a register machine M with at least $n+1$ registers $\mathtt{R}_0, \mathtt{R}_1, \ldots, \mathtt{R}_n$ (and maybe more)
such that for all $(x_1, \ldots, x_n) \in \mathbb{N}^n$ and all $y \in \mathbb{N}$,

the computation of M starting with $\mathtt{R}_0 = 0$, $\mathtt{R}_1 = x_1$, $\ldots$, $\mathtt{R}_n = x_n$ and all other registers set to 0, halts with $\mathtt{R}_0 = y$

if and only if $f(x_1, \ldots, x_n) = y$.

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...

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

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

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

(Notation: $0\mathtt{b}x \triangleq x$ *in binary*.)
E.g. $27 = 0\mathtt{b}11011 = \langle\langle 0, 13 \rangle\rangle = \langle 2, 3 \rangle$

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

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

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

$\langle -, - \rangle$ 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

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

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

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:

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

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:

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

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

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:

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

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

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

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

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

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

Numerical coding of programs

If P is the RM program $\boxed{\begin{array}{l} \mathtt{L}_0 : body_0 \\ \mathtt{L}_1 : body_1 \\ \quad\vdots \\ \mathtt{L}_n : body_n \end{array}}$

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:

$$\begin{cases} \ulcorner \mathtt{R}_i^+ \to \mathtt{L}_j \urcorner & \triangleq & \langle\langle 2i, j \rangle\rangle \\ \ulcorner \mathtt{R}_i^- \to \mathtt{L}_j, \mathtt{L}_k \urcorner & \triangleq & \langle\langle 2i+1, \langle j, k \rangle \rangle\rangle \\ \ulcorner \mathtt{HALT} \urcorner & \triangleq & 0 \end{cases}$$

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

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

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

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 $prog(e)$

- $786432 = 2^{19} + 2^{18} = 0b11\underbrace{0 \ldots 0}_{18\ "0"s} = \ulcorner [18, 0] \urcorner$
- $18 = 0b10010 = \langle\langle 1, 4 \rangle\rangle = \langle\langle 1, \langle 0, 2 \rangle\rangle\rangle = \ulcorner \mathtt{R}_0^- \rightarrow \mathtt{L}_0, \mathtt{L}_2 \urcorner$
- $0 = \ulcorner \mathtt{HALT} \urcorner$

So $prog(786432) =$

$$\boxed{\begin{array}{l} \mathtt{L}_0 : \mathtt{R}_0^- \rightarrow \mathtt{L}_0, \mathtt{L}_2 \\ \mathtt{L}_1 : \mathtt{HALT} \end{array}}$$

Example of $prog(e)$

- $786432 = 2^{19} + 2^{18} = 0b11\underbrace{0\ldots0}_{18\ "0"s} = \ulcorner[18, 0]\urcorner$
- $18 = 0b10010 = \langle\langle 1, 4\rangle\rangle = \langle\langle 1, \langle 0, 2\rangle\rangle\rangle = \ulcorner \mathtt{R}_0^- \rightarrow \mathtt{L}_0, \mathtt{L}_2 \urcorner$
- $0 = \ulcorner \mathtt{HALT} \urcorner$

So $prog(786432) =$

$$\boxed{\begin{array}{l}\mathtt{L}_0 : \mathtt{R}_0^- \rightarrow \mathtt{L}_0, \mathtt{L}_2 \\ \mathtt{L}_1 : \mathtt{HALT}\end{array}}$$

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

Example of $prog(e)$

- $786432 = 2^{19} + 2^{18} = 0b11\underbrace{0\ldots0}_{18\ "0"s} = \ulcorner[18, 0]\urcorner$
- $18 = 0b10010 = \langle\langle 1, 4\rangle\rangle = \langle\langle 1, \langle 0, 2\rangle\rangle\rangle = \ulcorner \mathtt{R}_0^- \to \mathtt{L}_0, \mathtt{L}_2 \urcorner$
- $0 = \ulcorner \mathtt{HALT} \urcorner$

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

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).

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:

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

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

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:

$\mathtt{R}_1 \equiv \mathtt{P}$ code of the RM to be simulated

$\mathtt{R}_2 \equiv \mathtt{A}$ code of current register contents of simulated RM

$\mathtt{R}_3 \equiv \mathtt{PC}$ program counter—number of the current instruction (counting from $\mathtt{0}$)

$\mathtt{R}_4 \equiv \mathtt{N}$ code of the current instruction body

$\mathtt{R}_5 \equiv \mathtt{C}$ type of the current instruction body

$\mathtt{R}_6 \equiv \mathtt{R}$ current value of the register to be incremented or decremented by current instruction (if not $\mathtt{HALT}$)

$\mathtt{R}_7 \equiv \mathtt{S}$, $\mathtt{R}_8 \equiv \mathtt{T}$ and $\mathtt{R}_9 \equiv \mathtt{Z}$ are auxiliary registers.

Overall structure of U's program

1 copy PCth item of list in P to N; goto 2

2 if $\mathtt{N} = 0$ then copy 0th item of list in A to $\mathtt{R}_0$ and halt, else (decode 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 A to R; goto 4

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