General - Uncategorized

Common features of definitions of **Algorithm**:
- **finite** description of the procedure in terms of elementary operations
- **deterministic** => next step is uniquely determined if there is one

procedure may not terminate on some input data, but we can recognise when it does terminate and *what* the *result* will be

We can think of *algorithms* as **Computable Functions** *(e.g. RM-computable, TM-computable, etc.)*

Partial Function $f : X \nrightarrow Y$ is such that:

$X \nrightarrow Y$ is the *set of partial functions* from X to Y

$f(x) = y$ means $(x, y) \in f$

$f(x) = y \wedge f(x) = y' \implies y = y'$

$f(x)\downarrow$ means $\exists y \in Y (f(x) = y)$; $f(x)\uparrow$ means $\neg f(x)\downarrow$

f is **Total** if $\forall x \in X$ we have $f(x)\downarrow$

$X \to Y$ is *set of total functions* from X to Y

General Halting Problem is the decision problem with:

the set S of all pairs (A, D), where A is an **algorithm** and D is some **input datum** on which A is designed to operate

$A(D)\downarrow$ holds for $(A, D) \in S$ if A applied to D *eventually halts*

The *Halting Problem* is **unsolvable (undecidable)**, i.e.

no algorithm $H : S \to \mathbb{B}$ exists s.t. $H(A, D) = \begin{cases} \mathbf{tt} & A(D)\downarrow \\ \mathbf{ff} & A(D)\uparrow \end{cases}$

Register Machine Definitions

A **Register Machine (RM)** is specified by:

finite **registers** $R_0, \dots, R_n$ each can store one nat. $\mathbb{N}$

a **program** => finite list of *instructions*

each **instruction** of the form $L_j : B_j$

L_j is **label** of $j+1$-th *instruction*, for $i = 0, 1, \dots$

B_j is **instruction body**, and is one of:

- $R^+ \to L'$ => add 1 to *reg.* R_j & jump to *inst.* L'
- $R^- \to L', L''$ => **if** $R > 0$ sub. 1 from R_j & jump to L' **else** jump to L''
- $HALT$ => stop executing instructions

initial/final reg. contents related by *partial function*

RM Configuration => $c = (\ell, r_0, \dots, r_n)$

ℓ = current label ; r_j = current contents of R_j

$R_j = x$ [in config. c] means $c = (\ell, r_0, \dots, r_n)$ with $r_j = x$

Initial Configuration => $c_0 = (0, r_0, \dots, r_n)$

RM Computation => finite *(or infinite) sequence* of *configurations* $c_0, c_1, c_2, \dots$

$c_0 = (0, r_0, \dots, r_n)$ is *initial configuration*

each c_{i+1} determined from $c_i = (\ell, r_0, \dots, r_n)$ by performing *inst.* labelled L_ℓ

Halting Computation => *finite sequence* $c_0, c_1 \dots, c_m$

Halting Configuration $c_m = (\ell, r, \dots)$

inst. labelled L_ℓ is **either** $HALT$ i.e. **Proper Halt**; **or** performs *jump to nonexistent label* i.e. **Erroneous Halt**

e.g. $L_0 : R_1^+ \to L_2$, $L_1 : HALT$ *halts erroneously*

Non-Halting Computation => *infinite sequence* $c_0, c_1, \dots$ e.g. $L_0 : R_1^+ \to L_0$, $L_1 : HALT$

RM Graphical Representation

one node per *inst.* $L_\ell : B_\ell$

each node labelled by $[L_\ell]$ where $[L_\ell]$ is *register* of *inst. body* B_ℓ

Arcs are *jumps*; **initial instruction** is $START$

Instruction	Representation
$R^+ \to L$	$R^+ \to [L]$
$R^- \to L, L'$	$R^- \to [L]$, $R^- \to [L']$
$HALT$	$HALT$
L_0	$START \to [L_0]$

$f : \mathbb{N}^n \to \mathbb{N}$ is **(RM) Computable Function** if:

There is RM M with $\geq n+1$ registers, such that for all $(x_1, \dots, x_n) \in \mathbb{N}^n, y \in \mathbb{N}$ we have *initial configuration* $c_0 = (0, x_1, \dots, x_n, 0, 0, \dots)$ and M halts with $R_0 = y$ **iff** $f(x_1, \dots, x_n) = y$

Basic computable functions => **addition** ; **multiplication** ; **projection** $(x, y) \mapsto x$; **constant** $x \mapsto k$; **truncated subtraction** $(x, y) \mapsto \max(0, x - y)$; **integer division** $(x, y) \mapsto \begin{cases} \lfloor x/y \rfloor & y > 0 \\ 0 & y = 0 \end{cases}$; **integer remainder** $(x, y) \mapsto \begin{cases} x - y \cdot \lfloor x/y \rfloor & y > 0 \\ x & y = 0 \end{cases}$; **exp. and log. base-2** $x \mapsto 2^x$, $x \mapsto \begin{cases} \lfloor \log_2(x) \rfloor & x > 0 \\ 0 & x = 0 \end{cases}$

Numerical Coding

Numerical Coding of Pairs => for $\alpha, \beta \in \mathbb{N}$ define:

$\langle\!\langle \alpha, \beta \rangle\!\rangle \triangleq 2^\alpha (2\beta + 1)$ is a *bijection* between $\mathbb{N}^2$ and $\mathbb{N}^+$ where $\mathbb{N}^+ = \{n \in \mathbb{N} \mid n > 0\}$

Binary form => $0\mathbf{b}\langle\!\langle \alpha, \beta \rangle\!\rangle = 0\mathbf{b} \left[\beta \,\middle|\, 1 \,\middle|\, \underbrace{0 \cdots 0}_{\alpha} \right]$

$\langle \alpha, \beta \rangle \triangleq \langle\!\langle \alpha, \beta \rangle\!\rangle - 1 = 2^\alpha (2\beta + 1) - 1$

$\langle -, - \rangle : \mathbb{N}^2 \to \mathbb{N}$ is a *bijection* between $\mathbb{N}^2$ and $\mathbb{N}$

Binary form => $0\mathbf{b}\langle \alpha, \beta \rangle = 0\mathbf{b} \left[\beta \,\middle|\, 0 \,\middle|\, \underbrace{1 \cdots 1}_{\alpha} \right]$

EXAMPLE: $27 = 0\mathbf{b}11011 = \langle\!\langle 0, 13 \rangle\!\rangle = \langle 2, 3 \rangle$

Numerical Coding of Lists:

$List\ \mathbb{N}$ is set of all finite lists of *natural numbers*:

- **Empty List** => $[] \in List\ \mathbb{N}$
- **List Cons** => $x :: \ell \in List\ \mathbb{N} \iff x \in \mathbb{N} \wedge \ell \in List\ \mathbb{N}$
- **Notation** => $[x_1, x_2, \dots, x_n] \triangleq x_1 :: (x_2 :: (\cdots x_n :: [] \cdots))$

Define *bijection* $\ulcorner - \urcorner : List\ \mathbb{N} \to \mathbb{N}$ inductively:

$\ulcorner [] \urcorner \triangleq 0$; $\ulcorner x :: \ell \urcorner \triangleq \langle\!\langle x, \ulcorner \ell \urcorner \rangle\!\rangle = 2^x (2 \ulcorner \ell \urcorner + 1)$

Binary form =>

$$0\mathbf{b}\ulcorner [x_1, \dots, x_{n-1}, x_n] \urcorner = 0\mathbf{b} \left[1 \,\middle|\, \underbrace{0 \cdots 0}_{x_n} \,\middle|\, 1 \,\middle|\, \underbrace{0 \cdots 0}_{x_{n-1}} \,\middle|\, \cdots \,\middle|\, 1 \,\middle|\, \underbrace{0 \cdots 0}_{x_1} \right]$$

Let $P = [L_0 : B_0, L_1 : B_1, \dots, L_n : B_n]$ be an **RM program**:

Numerical Coding of Bodies $\ulcorner B \urcorner$ is defined by

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

and $\ulcorner B \urcorner$ defines a *bijection*

Numerical Coding of RM Programs $\ulcorner P \urcorner \triangleq \ulcorner [\ulcorner B_0 \urcorner, \ulcorner B_1 \urcorner, \dots, \ulcorner B_n \urcorner] \urcorner$ defines a *bijection*

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

- $x = 0 \implies body(x) = HALT$
- $x > 1 \implies x = \langle\!\langle y, z \rangle\!\rangle$ and:
  - $y = 2i \implies body(x) = R_i^+ \to L_z$
  - $y = 2i+1 \implies z = \langle j, k \rangle$ and $body(x) = R_i^- \to L_j, L_k$

Any $e \in \mathbb{N}$ decodes to *unique program* $prog(e)$ called **program with index** e

Register Machine Gadgets

RM Gadgets => *partial RM graph* with **1 entry wire** & **≥ 1 more exit wires**

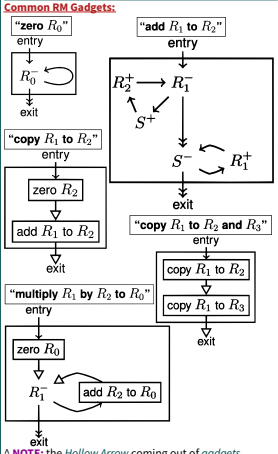
Input/output registers specified in *gadget's name*

Scratch registers => *internal* registers for *temporary storage*

these *scratch registers* are **initially** *set to zero* & **upon exit** *must return to zero*

Compose *smaller gadgets* to create *bigger gadgets* => can **rename** clashing scratch register names

Common RM Gadgets:



^ **NOTE:** the *Hollow Arrow* coming out of *gadgets* represents **all outputs** combined

RM Gadgets for Coded Lists:

foo bar