Computability

Wojciech Buszkowski

Faculty of Mathematics and Computer Science
Adam Mickiewicz University

Based on: N. Cutland, Computability. An introduction to recursive function theory. Cambridge University Press, 1992.

2. The register machine

Abbreviation: RM

Other names: machine RAM, Minsky machine

R_1	R_2	R_3	$\dots$
r_1	r_2	r_3	$\dots$

R_i - the i–th register ($i \geq 1$)

There are infinitely many registers.

Registers store natural numbers. 0 is stored in the 'empty' registers.

r_i - the number stored in R_i

Instructions: (here $m, n, q \geq 1$)

Name	Symbol	Meaning
Zeroing	$Z(n)$	$r_n := 0$
Successor	$S(n)$	$r_n := r_n + 1$
Copying	$T(m, n)$	$r_n := r_m$
Conditional Jump	$J(m, n, q)$	if $r_m = r_n$ then goto I_q

A program is a finite sequence of instructions $P = (I_1, \ldots, I_s)$; s is *the length of* P. We admit $s = 0$.

A configuration is an infinite sequence $(r_n)_{n \geq 1}$ such that $r_n = 0$ for all but finitely many n; it shows the contents of registers.

Let $P = (I_1, \dots, I_s)$, $s > 0$, be a program. The computation of P on a configuration $(r_n^1)_{n\geq 1}$ can informally be described as follows.

The initial configuration is $(r_n^1)_{n\geq 1}$. We execute the instructions of P, starting from I_1; after I_j we execute I_{j+1} except for three cases:

(1) $I_j = J(m, n, q)$, $1 \leq q \leq s$ and $r_m = r_n$ for the current configuration; then we execute I_q after I_j,

(2) $I_j = J(m, n, q)$, $s < q$ and $r_m = r_n$ for the current configuration; then the computation ends,

(3) $j = s$ and the conditions of (1), (2) do not hold; then the computation ends.

A step of the computation is the execution of one instruction.

The computation of P on a configuration can be finite or infinite. *The result* of a finite computation is r_1 for the final configuration.

We omit a precise definition; it will be given later on in terms of encoding. We write $\mathbf{x}$ for $(x_1, \ldots, x_n)$.

Let P be a program and $\mathbf{x} \in \mathbb{N}^n$.

We write $P(\mathbf{x}) \downarrow$, if the computation of P on $(x_1, \ldots, x_n, 0, 0, \ldots)$ is finite, and $P(\mathbf{x}) \uparrow$ otherwise.

We write $P(\mathbf{x}) \downarrow y$, if $P(\mathbf{x}) \downarrow$ and y is the result of this computation.

We define $f_P^{(n)}$ - *the* n*−ary function computed by* P.

$\mathrm{Dom}(f_P^{(n)}) = \{\mathbf{x} \in \mathbb{N}^n : P(\mathbf{x}) \downarrow\}$

For $\mathbf{x} \in \mathrm{Dom}(f_P^{(n)})$, $f_P^{(n)}(\mathbf{x})$ equals the unique y such that $P(\mathbf{x}) \downarrow y$.

We define $\mathrm{COM}_{RM}^{(n)}$ as the set of all $n-$ary functions computed by programs for RM and:

$\mathrm{COM}_{RM} = \bigcup_{n\geq 1} \mathrm{COM}_{RM}^{(n)}$ (the set of functions computable on RM).

Examples

(1) $f(x, y) = x + y$

the initial configuration: $|\, x \,|\, y \,|\, 0 \,|$

after k macro-steps: $|\, x + k \,|\, y \,|\, k \,|$

Program:

1. $J(2, 3, 5)$ % $r_2 = r_3$? If YES, then STOP.
2. $S(1)$ % $r_1 := r_1 + 1$
3. $S(3)$ % $r_3 := r_3 + 1$
4. $J(1, 1, 1)$ % GO TO 1

(2)

$$f(x, y) = \begin{cases} 1 & \text{if } x < y \\ \infty & \text{otherwise} \end{cases}$$

the initial configuration: $|\, x \,|\, y \,|$

after k macro-steps: $|\, x + k \,|\, y \,|$

Program:

1. $S(1)$ % $r_1 := r_1 + 1$
2. $J(1, 2, 4)$ % $r_1 = r_2$? If YES, then GO TO 4
3. $J(1, 1, 1)$ % GO TO 1
4. $Z(1)$ % $r_1 := 0$
5. $S(1)$ % $r_1 := r_1 + 1$

(3) $f(x, y) = c_<(x, y)$

the initial configuration: $|\,x\,|\,y\,|\,0\,|\,0\,|$

after k macro-steps: $|\,x+k\,|\,y+k\,|\,x\,|\,y\,|$

1.	$J(1, 2, 12)$	
2.	$T(1, 3)$	% $r_3 := r_1$
3.	$T(2, 4)$	% $r_4 := r_2$
4.	$S(1)$	
5.	$S(2)$	
6.	$J(1, 4, 9)$	
7.	$J(2, 3, 12)$	
8.	$J(1, 1, 4)$	% GO TO 4
9.	$Z(1)$	
10.	$S(1)$	% $|\,1\,|\ldots|$
11.	$J(1, 1, 13)$	% STOP
12.	$Z(1)$	% $|\,0\,|\ldots|$

Def. 1. A program $(I_1, \ldots, I_s)$ is said to be *standard*, if $q \leq s + 1$ for any instruction $J(-, -, q)$ occurring in this program.

Def. 2. We say that programs P, Q are *equivalent*, if $f_P^{(n)} = f_Q^{(n)}$ for all $n \geq 1$.

Proposition 1. Every program is equivalent to a standard program (of the same length).

Proof. Replace every $J(-, -, q)$ such that $s < q$ by $J(-, -, s + 1)$.

Def. 3. Let P, Q be standard programs, $P = (I_1^P, \ldots, I_s^P)$, $Q = (I_1^Q, \ldots, I_t^Q)$. We define *the composition* $P; Q$ as the program $(I_1, \ldots, I_{s+t})$ such that:

$I_j = I_j^P$ for all $1 \leq j \leq s$ and $I_j = (I_{j-s}^Q)'$ for all $s < j \leq s + t$,

where $(I_j^Q)' = I_j^Q$ for $I_j^Q \neq J(-, -, -)$, $(I_j^Q)' = J(-, -, s + q)$ for $I_j^Q = J(-, -, q)$. Clearly $P; Q$ is standard.

Proposition 2. $(P; Q); R = P; (Q; R)$ for all standard programs P, Q, R.

We write $P_1; P_2; \ldots; P_n$.

By $\rho(P)$ we denote the greatest register number occurring in P.

Def. 4. Let $k_1, \ldots, k_n, k$ be positive integers such that $k_i > n$ for any $1 \leq i \leq n$ and $k_i \neq k_j$ for $i \neq j$. Let P be a standard program. We define a program $P[k_1, \ldots, k_n \to k]$. This program:

takes the input data from registers $R_{k_1}, \ldots, R_{k_n}$,

copies them in $R_1, \ldots, R_n$, respectively,

cleans the remaining registers needed for program P,

runs P, and copies the result in R_k.

$$
\begin{array}{ll}
1. & T(k_1, 1) \\
\vdots & \\
n. & T(k_n, n) \\
n+1. & Z(n+1) \\
\vdots & \\
\rho(P). & Z(\rho(P)); \\
P; & \\
T(1, k) &
\end{array}
$$

The instructions from $n + 1$ to $\rho(P)$ are absent, if $\rho(P) \leq n$.

Clearly $P[k_1, \ldots, k_n \to k]$ is standard.

Theorem 1. Every partial recursive function is computable on RM.

PROOF. In order to prove $\mathrm{REC} \subseteq \mathrm{COM}_{RM}$ we show that COM_{RM} contains all basic recursive functions and is closed under substitution, primitive recursion and minimum.

$Z(x) = 0$. $P = (Z(1))$.

$S(x) = x + 1$. $P = (S(1))$.

$I_i^n(\mathbf{x}) = x_i$. $P = (T(i, 1))$.

Substitution. Assume that $f, g_1, \ldots, g_k \in \mathrm{COM}_{RM}$ and $h(\mathbf{x}) \simeq f(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}))$.

Let $P, P_1, \ldots, P_k$ be standard programs computing $f, g_1, \ldots, g_k$, respectively. We construct a standard program Q, computing h.

We denote $m = \max(n, k, \rho(P), \rho(P_1), \ldots, \rho(P_k))$.

The input data $x_1, \ldots, x_n$ are stored in registers $R_{m+1}, \ldots, R_{m+n}$ and the values $g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})$ in $R_{m+n+1}, \ldots, R_{m+n+k}$.

$$| \dots | \overset{R_{m+1}}{x_1} | \dots | \overset{R_{m+n}}{x_n} | \overset{R_{m+n+1}}{g_1(\mathbf{x})} | \dots | \overset{R_{m+n+k}}{g_k(\mathbf{x})} |$$

1. $T(1, m+1)$

$\vdots$

n. $T(n, m+n)$;

$P_1[m+1, \dots, m+n \to m+n+1]$;

$\vdots$

$P_k[m+1, \dots, m+n \to m+n+k]$;

$P[m+n+1, \dots, m+n+k \to 1]$

Notice that $Q(\mathbf{x}) \downarrow$ if and only if $P_i(\mathbf{x}) \downarrow$ for all $1 \le i \le k$ and $P(g_1(\mathbf{x}), \dots, g_k(\mathbf{x})) \downarrow$.

Primitive recursion

$h(\mathbf{x}, 0) = f(\mathbf{x})$

$h(\mathbf{x}, y + 1) = g(\mathbf{x}, y, h(\mathbf{x}, y))$

Let P, Q be standard programs computing f, g, respectively. We construct a standard program R, computing h.

Denote $m = \max(n + 2, \rho(P), \rho(Q))$.

R stores the input data $x_1, \ldots, x_n, y$ in registers $R_{m+1}, \ldots, R_{m+n}, R_{m+n+1}$, uses R_{m+n+2} as a counter keeping a current number $0 \le k \le y$, and stores the value $h(\mathbf{x}, k)$ in R_{m+n+3}.

After k macro-steps:

$$| \ldots | \overset{R_{m+1}}{x_1} | \ldots | \overset{R_{m+n}}{x_n} | \overset{R_{m+n+1}}{y} | \overset{R_{m+n+2}}{k} | \overset{R_{m+n+3}}{h(\mathbf{x}, k)} |$$

$$
\begin{array}{ll}
1. & T(1, m+1) \\
\vdots & \\
n. & T(n, m+n) \\
n+1. & T(n+1, m+n+1);
\end{array}
$$

$P[m+1, \ldots, m+n \to m+n+3]$ computes $f(\mathbf{x})$

$J(m+n+1, m+n+2, j)$; (instruction I_i)

$Q[m+1, \ldots, m+n, m+n+2, m+n+3 \to m+n+3]$ computes $h(\mathbf{x}, k)$ and stores it in R_{m+n+3}

$S(m+n+2)$

$J(1, 1, i)$

$T(m+n+3, 1)$ (instruction I_j)

μ–operator

$h(\mathbf{x}) = \mu y(f(\mathbf{x}, y) = 0)$, $f : \mathbb{N}^{n+1} \to \mathbb{N}$, $h : \mathbb{N}^n \to \mathbb{N}$.

Let P be a program computing f. We construct a program Q, computing h.

We define $m = \max(n + 1, \rho(P))$.

Q stores the input data $x_1, \ldots, x_n$ in $R_{m+1}, \ldots, R_{m+n}$,

uses R_{m+n+1} as a counter, keeping a current number $k \geq 0$,

computes $f(\mathbf{x}, k)$ and stores the value in R_{m+n+2},

verifies $f(\mathbf{x}, k) = 0$ (R_{m+n+3} keeps 0)

if YES, returns k as the result, else sets $k := k + 1$ and repeats the loop.

after k macro-steps:

$$|\ldots| \overset{R_{m+1}}{x_1} |\ldots| \overset{R_{m+n}}{x_n} | \overset{R_{m+n+1}}{k} | \overset{R_{m+n+2}}{f(\mathbf{x},k)} | \overset{R_{m+n+3}}{0} |$$

1. $T(1, m+1)$

$\vdots$

n. $T(n, m+n)$;

$P[m+1, \ldots, m+n, m+n+1 \to m+n+2]$

$J(m+n+2, m+n+3, j)$

$S(m+n+1)$

$J(1, 1, n+1)$

$T(m+n+1, 1)$ (instruction I_j))

3. Encoding

We need some auxiliary primitive recursive functions.

the pairing function: $\pi(x, y) = 2^x(2y + 1) - 1 = 2^x(2y + 1)\dot{-}1$

$\pi : \mathbb{N}^2 \mapsto \mathbb{N}$ is a bijection.

$\pi(0, 0) = 0$, $\pi(1, 0) = 1$, $\pi(0, 1) = 2$, $\pi(2, 0) = 3$, $\pi(0, 2) = 4$,
$\pi(1, 1) = 5$

We have: $\pi(x, y) \geq x$, $\pi(x, y) \geq y$.

The converse functions:

$\pi_1(z) = (\mu x < z + 1)(\exists_{y \leq z}\, \pi(x, y) = z)$,
$\pi_2(z) = (\mu y < z + 1)(\exists_{x \leq z}\, \pi(x, y) = z)$.

We have: $\pi_1(\pi(x, y)) = x$, $\pi_2(\pi(x, y)) = y$, $\pi(\pi_1(z), \pi_2(z)) = z$.

Define: $\tau(x, y, z) = \pi(\pi(x, y), z)$. $\tau : \mathbb{N}^3 \mapsto \mathbb{N}$ is a bijection.

$\tau_1(k) = \pi_1(\pi_1(k))$, $\tau_2(k) = \pi_2(\pi_1(k))$, $\tau_3(k) = \pi_2(k)$

W^* - the set of all finite sequences of elements of W, including the empty sequence ϵ.

We define a computable bijection $\langle\cdot\rangle : \mathbb{N}^* \mapsto \mathbb{N}$.

$$\langle\epsilon\rangle = 0$$

$$\langle x_1, \ldots, x_n\rangle = p_1^{x_1} \cdots p_{n-1}^{x_{n-1}} \cdot p_n^{x_n+1} - 1$$

$\langle 0\rangle = 1$, $\langle 0, 0\rangle = 2$, $\langle 1\rangle = 3$, $\langle 0, 0, 0\rangle = 4$, $\langle 1, 0\rangle = 5$

One easily shows: for any $k \in \mathbb{N}$, there is exactly one $\alpha \in \mathbb{N}^*$ such that $\langle\alpha\rangle = k$.

$\langle\alpha\rangle$ is called *the (sequence) number of* α.

$lh(x) =$ the length of the sequence of number x

$(x)_i =$ the $i-$th term of this sequence, if $1 \leq i \leq lh(x)$; otherwise $(x)_i = 0$.

Exercise. Find α such that $\langle\alpha\rangle = 50$.

$\alpha = (k_1, \ldots, k_n)$, if $p_1^{k_1} \cdots p_{n-1}^{k_{n-1}} \cdot p_n^{k_n+1} = 51$.

$51 = 2^0 3^1 5^0 7^0 11^0 13^0 17^1$

So $\alpha = (0, 1, 0, 0, 0, 0, 0)$.

$lh(50) = 7$, $(50)_2 = 1$, $(50)_i = 0$ for any $i \neq 2$

$\exp(x, n) =$ the exponent α_n in the factorization $x = \prod_{n=1}^{\infty} p_n^{\alpha_n}$, if $x \neq 0$ and $n \neq 0$; otherwise $\exp(x, n) = 0$.

Lemma 1. The following functions are (primitive) recursive:

$f(x, n) = \exp(x, n)$,

$g(x) = lh(x)$,

$h(x, i) = (x)_i$,

$f^n(x_1, \ldots, x_n) = \langle x_1, \ldots, x_n\rangle$ for any fixed $n \geq 1$.

INS - the set of all instructions

We define a computable bijection $\nu : \mathrm{INS} \mapsto \mathbb{N}$.

$\nu(Z(n)) = 4(n-1)$

$\nu(S(n)) = 4(n-1) + 1$

$\nu(T(m,n)) = 4\pi(m-1, n-1) + 2$

$\nu(J(m,n,q)) = 4\tau(m-1, n-1, q-1) + 3$

$\nu(I)$ is called *the instruction number of I*.

$\mathrm{PRO}_{RM} = \mathrm{INS}^*$ - the set of all programs for RM

Def. 5. Let $P = (I_1, \ldots, I_s)$ be a program. *The Gödel number of P* is defined as follows.

$\ulcorner P \urcorner = \langle \nu(I_1), \ldots, \nu(I_s) \rangle$

Clearly $\ulcorner \cdot \urcorner$ is a computable bijection from PRO_{RM} onto $\mathbb{N}$.

P_e denotes the program of number e, i.e. a unique program P such that $\ulcorner P \urcorner = e$.

Clearly the function $f(e) = P_e$ is computable. We have:

$P_e = (\nu^{-1}((e)_1), \ldots, \nu^{-1}((e)_{lh(e)}))$,

and ν^{-1} is computable.

We define: $\{e\}^{(n)} = f^{(n)}_{P_e}$. In Cutland's book: $\phi^{(n)}_e$.

The number e is called *the index of* the function $\{e\}^{(n)}$.

We write $\{e\}$ for $\{e\}^{(1)}$.

Exercises. (1) Compute $\ulcorner(Z(1), S(1), S(1))\urcorner$. We have $\nu(Z(1)) = 0$, $\nu(S(1)) = 1$. So the number of this program equals $\langle 0, 1, 1\rangle = 2^0 3^1 5^2 - 1 = 74$.

(2) Find P_{27}. We have $28 = 2^2 3^0 5^0 7^1$, hence $27 = \langle 2, 0, 0, 0\rangle$. $2 = \nu(T(1, 1))$, $0 = \nu(Z(1))$. So $P_{27} = (T(1, 1), Z(1), Z(1), Z(1))$.

Example A. We define a function, which is not recursive.

We consider the following total function.

$$f(x) = \begin{cases} \{x\}(x) + 1 & \text{if } \{x\}(x) \text{ is defined} \\ 0 & \text{otherwise} \end{cases}$$

We show $f \notin \mathrm{COM}_{RM}$. Assume the contrary. Then $f = \{e\}$, for some $e \in \mathbb{N}$.

$\{e\}(e)$ is defined, since f is total. We obtain:

$\{e\}(e) = f(e) = \{e\}(e) + 1$,

which is impossible.

By Theorem 1, $f \notin \mathrm{REC}$.

Proposition 3. Every function computable on RM has infinitely many indices.

Proof. Let $f \in \mathrm{COM}_{RM}$. There exists a standard program P, computing f. Let $P = (I_1, \ldots, I_s)$.

P is equivalent to every program $(I_1, \ldots, I_s, T(1,1), T(1,1), \ldots, T(1,1))$, hence the Gödel numbers of these programs are indices of f. q.e.d.

Def. 6. Let $(r_n)_{n\geq 1}$ be a configuration. The number

$$r = \prod_{n=1}^{\infty} p_n^{r_n}$$

is called *the (code) number of* $(r_n)_{n\geq 1}$.

Remark. Every natural number $r \geq 1$ is the number of a unique configuration.

For any $n \geq 1$ we define two total functions.

$c_n(e, \mathbf{x}, t)$ = the number of the configuration after t steps of the computation of P_e for the entry $\mathbf{x}$, if this computation has at least t steps,

$c_n(e, \mathbf{x}, t)$ = the number of the final configuration, otherwise.

$j_n(e, \mathbf{x}, t)$ = the (ordinal) number k of the instruction I_k (in P_e) executed after t steps of the computation of P_e for the entry $\mathbf{x}$, if this computation has more than t steps,

$j_n(e, \mathbf{x}, t) = 0$, otherwise.

Lemma 2. For any $n \geq 1$, the functions c_n, j_n are (primitive) recursive.

PROOF.

We define two functions.

$\mathrm{con}(e, r, j)$ = the number of the configuration obtained by the execution of I_j in P_e on the configuration of number r, if $1 \leq j \leq lh(e)$; $\mathrm{con}(e, r, j) = r$, otherwise

$\mathrm{ins}(e, r, j)$ = the (ordinal) number q of the instruction to be executed after the execution of I_j in P_e on the configuration of number r, if $1 \leq j \leq lh(e)$ and $1 \leq q \leq lh(e)$; $\mathrm{ins}(e, r, j) = 0$, otherwise

Then, c_n, j_n can be defined by simultaneous recursion.

$$c_n(e, \mathbf{x}, 0) = p_1^{x_1} \cdots p_n^{x_n}$$

$$j_n(e, \mathbf{x}, 0) = sg(e)$$

$$c_n(e, \mathbf{x}, t+1) = \mathrm{con}(e, c_n(e, \mathbf{x}, t), j_n(e, \mathbf{x}, t))$$

$$j_n(e, \mathbf{x}, t+1) = \mathrm{ins}(e, c_n(e, \mathbf{x}, t), j_n(e, \mathbf{x}, t))$$

It suffices to show that con and ins are (primitive) recursive.

We need functions rg_1, rg_2, jp such that:

$rg_1(\nu(Z(n))) = n$, $rg_1(\nu(S(n))) = n$,

$rg_1(\nu(T(m, n))) = m$, $rg_1(\nu(J(m, n, q))) = m$,

$rg_2(\nu(T(m, n))) = n$, $rg_2(\nu(J(m, n, q))) = n$,

$jp(\nu(J(m, n, q))) = q$.

Recall that:

$$[x/y] = (\mu z < x + 1)\,(y = 0 \lor x < (z + 1)y)\,.$$

$$jp(x) = \begin{cases} \tau_3([(x \dot{-} 3)/4]) + 1 & \text{if } rm(x, 4) = 3 \\ 0 & \text{otherwise} \end{cases}$$

$$rg_2(x) = \begin{cases} \pi_2([(x \dot{-} 2)/4]) + 1 & \text{if } rm(x,4) = 2 \\ \tau_2([(x \dot{-} 3)/4]) + 1 & \text{if } rm(x,4) = 3 \\ 0 & \text{otherwise} \end{cases}$$

$$rg_1(x) = \begin{cases} [x/4] + 1 & \text{if } rm(x,4) = 0 \\ [(x \dot{-} 1)/4] + 1 & \text{if } rm(x,4) = 1 \\ \pi_1([(x \dot{-} 2)/4]) + 1 & \text{if } rm(x,4) = 2 \\ \tau_1([(x \dot{-} 3)/4]) + 1 & \text{otherwise} \end{cases}$$

Accordingly rg_1, rg_2 and jp are primitive recursive.

We define a (primitive) recursive function:

$$fc(x,n) = pr(n)^{\exp(x,n)} .$$

$$\mathrm{con}(e, r, j) = \begin{cases} [r/fc(r, rg_1((e)_j))] \\ \text{ if } r \geq 1 \wedge 1 \leq j \leq lh(e) \wedge rm((e)_j, 4) = 0 \\ r \cdot pr(rg_1((e)_j)) \\ \text{ if } r \geq 1 \wedge 1 \leq j \leq lh(e) \wedge rm((e)_j, 4) = 1 \\ [r/fc(r, rg_2((e)_j))] \cdot pr(rg_2((e)_j))^{exp(r, rg_1((e)_j))} \\ \text{ if } r \geq 1 \wedge 1 \leq j \leq lh(e) \wedge rm((e)_j, 4) = 2 \\ r \;\; \text{otherwise} \end{cases}$$

$$\text{ins}(e, r, j) = \begin{cases} j + 1 \text{ if } r \geq 1 \wedge 1 \leq j < lh(e) \wedge rm((e)_j, 4) < 3 \\ j + 1 \text{ if } r \geq 1 \wedge 1 \leq j < lh(e) \wedge rm((e)_j, 4) = 3 \\ \quad \wedge\, exp(r, rg_1((e)_j)) \neq exp(r, rg_2((e)_j)) \\ jp((e)_j) \text{ if } r \geq 1 \wedge 1 \leq j \leq lh(e) \wedge rm((e)_j, 4) = 3 \\ \quad \wedge\, exp(r, rg_1((e)_j)) = exp(r, rg_2((e)_j)) \\ \quad \wedge\, 1 \leq jp((e)_j) \leq lh(e) \\ 0 \quad \text{otherwise} \end{cases}$$

This finishes the proof of Lemma 2.

The fundamental equation

$$\text{(FE)}\ \ \{e\}^{(n)}(\mathbf{x}) \simeq exp(c_n(e, \mathbf{x}, \mu t\,(j_n(e, \mathbf{x}, t) = 0)), 1)$$

Theorem 2. Every function computable on RM is partial recursive.

PROOF. Let $f : \mathbb{N}^n \to \mathbb{N}$ be computable on RM. Then $f = \{e\}^{(n)}$ for some $e \in \mathbb{N}$. By (FE), $f \in$ REC. q.e.d.

Corollary 1. $\mathrm{COM}_{RM} = \mathrm{REC}$.

Def. 7. The function $U_n : \mathbb{N}^{n+1} \to \mathbb{N}$, defined by:

$U_n(e, \mathbf{x}) \simeq \{e\}^{(n)}(\mathbf{x})$,

is called *the universal function for n*−ary partial recursive functions.

Corollary 2. For any $n \geq 1$, the function U_n is partial recursive.

Theorem 3. (*Kleene normal form theorem*) For any $n \geq 1$, there exists a primitive recursive relation $T_n \subseteq \mathbb{N}^{n+2}$ and a primitive recursive function $\delta : \mathbb{N} \mapsto \mathbb{N}$ such that the equation:

$\{e\}^{(n)}(\mathbf{x}) \simeq \delta(\mu z T_n(e, \mathbf{x}, z))$,

holds for all $e \in \mathbb{N}$, $\mathbf{x} \in \mathbb{N}^n$.

Proof. We define a relation $S_n \subseteq \mathbb{N}^{n+3}$:

$S_n(e, \mathbf{x}, y, t)$ iff $P_e(\mathbf{x}) \downarrow y$ in at most t steps.

S_n is primitive recursive, since we have:

$S_n(e, \mathbf{x}, y, t) \Leftrightarrow j_n(e, \mathbf{x}, t) = 0 \wedge exp(c_n(e, \mathbf{x}, t), 1) = y$.

We define T_n as follows:

$T_n(e, \mathbf{x}, z)$ iff $S_n(e, \mathbf{x}, \pi_1(z), \pi_2(z))$.

Clearly $\{e\}^{(n)}(\mathbf{x}) \simeq \pi_1(\mu z T(e, \mathbf{x}, z))$. So $\delta = \pi_1$. q.e.d.

Effective μ−operator

Let $f : \mathbb{N}^{n+1} \mapsto \mathbb{N}$ (total) satisfy *the effectiveness condition*:

(EC) for any $\mathbf{x} \in \mathbb{N}^n$ there exists $y \in \mathbb{N}$ such that $f(\mathbf{x}, y) = 0$.

We define a function $h : \mathbb{N}^n \mapsto \mathbb{N}$ (total) by:

$h(\mathbf{x}) = \mu y(f(\mathbf{x}, y) = 0) = \min\{y \in \mathbb{N} : f(\mathbf{x}, y) = 0\}$.

We say that h arises from f by the effective μ−operator.

By REC_t we denote the family of total recursive functions.

Theorem 4. REC_t is the smallest family of total numerical functions, which contains all basic recursive functions and is closed under substitution, primitive recursion and the effective μ−operator.

PROOF. Let $\mathcal{F}$ denote the smallest family as above. $\mathcal{F} \subseteq \mathrm{REC}_t$, since REC_t satisfies these conditions. We show $\mathrm{REC}_t \subseteq \mathcal{F}$. Let $f \in \mathrm{REC}_t$, and let e be an index of f. The μ−operator appearing in (FE) is effective, and c_n, j_n , exp are in $\mathcal{F}$, hence $f \in \mathcal{F}$. q.e.d.

(R9) Let functions $g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$ be partial recursive. Let relations $R_1, \ldots, R_k \subseteq \mathbb{N}^n$ be recursive and satisfy the condition:

($*$) for any $\mathbf{x} \in \mathbb{N}^n$ there is exactly one $1 \leq i \leq k$ such that $R_i(\mathbf{x})$ holds.

Then, the function $h : \mathbb{N}^n \to \mathbb{N}$ defined by:

$$h(\mathbf{x}) \simeq \begin{cases} g_1(\mathbf{x}) & \text{if } R_1(\mathbf{x}) \\ \vdots & \\ g_k(\mathbf{x}) & \text{if } R_k(\mathbf{x}) \end{cases}$$

is partial recursive.

We assume:

$$h(\mathbf{x}) \neq \infty \text{ iff for some } 1 \leq i \leq k\,,\ R_i(\mathbf{x}) \wedge g_i(\mathbf{x}) \neq \infty\,.$$

PROOF.

Let $e_1, \ldots e_k$ be indices of $g_1, \ldots, g_k$, respectively.

This means: $g_i(\mathbf{x}) \simeq \{e_i\}^{(n)}(\mathbf{x})$, for any $1 \leq i \leq k$ and any $\mathbf{x} \in \mathbb{N}^n$. We define an auxiliary function:

$$g(\mathbf{x}) = \begin{cases} e_1 & \text{if } R_1(\mathbf{x}) \\ \vdots & \\ e_k & \text{if } R_k(\mathbf{x}) \end{cases}$$

By (R4), g is recursive. We have:

$$h(\mathbf{x}) \simeq U_n(g(\mathbf{x}), \mathbf{x}) \text{ for any } \mathbf{x} \in \mathbb{N}^n.$$

Consequently, $h \in$ REC. q.e.d.

We define two relations.

$\mathrm{HALT}^{(2)}(x, y)$ iff $P_x(y) \downarrow$

$\mathrm{HALT}(x)$ iff $P_x(x) \downarrow$

Theorem 5. The relations $\mathrm{HALT}^{(2)}$ and HALT are not recursive.

PROOF. Suppose that HALT is recursive. Consider the function f from Example A. We know that $f \notin \mathrm{REC}$. We can define f as follows.

$$f(x) = \begin{cases} U_1(x, x) + 1 & \text{if } \mathrm{HALT}(x) \\ 0 & \text{if } \neg\, \mathrm{HALT}(x) \end{cases}$$

By (R9), $f \in \mathrm{REC}$. Contradiction. So HALT is not recursive.

We have: $\mathrm{HALT}(x) \Leftrightarrow \mathrm{HALT}^{(2)}(x, x)$. By (R1), $\mathrm{HALT}^{(2)}$ is not recursive, either. q.e.d.

4. Church's thesis

Also: the Church thesis, the Church-Turing thesis

Abbreviation: (CT)

> (CT) *The class of partial numerical functions, computable in a general sense, coincides exactly with the class of partial recursive functions.*

Let COM denote the class of partial numerical functions, computable in a general sense.

(CT) COM = REC

We know that REC = COM_{RM}. Obviously $\mathrm{COM}_{RM} \subseteq \mathrm{COM}$, since programs for RM are certain algorithms. So REC ⊆ COM.

The converse COM ⊆ REC is a hypothesis. It cannot be proved, since COM has not been precisely defined as a mathematical notion.

Arguments supporting (CT)

(1) Other models of computation were studied:

- Turing machines
- Markov algorithms
- Post systems

and others. For any model, it has been proved that the partial numerical functions computable in this model coincide with partial recursive functions. The latter also coincide with the functions definable in lambda calculus and other logical formalisms.

The same can be proved (tediously) for all existing programming languages.

(2) All results of recursion theory become intuitively sound (often obvious), if one replaces ‘recursive’ with ‘computable’.

Applications of (CT)

I. *Positive*

If we know that f is computable (we know an arbitrary algorithm computing f), then we infer $f \in \mathrm{REC}$.

These applications are not essential. Sometimes one argues in this way just to shorten the proof. In all (known) cases, a complete proof of $f \in \mathrm{REC}$ can be provided.

II. *Negative*

If we know that $f \notin \mathrm{REC}$, then we infer that f is not computable (there exists no algorithm computing f).

These applications are essential. The only method of proving that a function is not computable is to show that it is not recursive (equivalently: not computable in an abstract model of computation, which yields all recursive functions).

Recall that a relation $R \subseteq \mathbb{N}^n$ is computable (in a general sense) iff c_R is computable.

For a relation, one also says 'solvable' or 'decidable'.

(CT) *for relations*: A numerical relation is computable iff it is recursive.

This follows from (CT) for functions. R is computable iff c_R is computable iff $c_R \in \text{REC}$ iff R is recursive.

So $\text{HALT}^{(2)}$ and HALT are not computable.

The halting problem for RM: Verify $P(x)\downarrow$, for arbitrary $P \in \text{PROG}_{RM}$, $x \in \mathbb{N}$.

Claim. The halting problem for RM is unsolvable (undecidable).

This means: there exists no algorithm which, for any program P on RM and any natural number x, verifies whether $P(x)\downarrow$ or not.

Computability on other domains

Now by *a domain* we mean a pair (D, α) such that:

- D is an infinite countable set (of finite objects),

- α is a computable bijection of D onto $\mathbb{N}$,

- α^{-1} is computable.

Let (D, α), (E, β) be domains. We consider partial functions $f : D^n \to E$.

For any f, we define $f^c : \mathbb{N}^n \to \mathbb{N}$.

$f^c(\mathbf{x}) = \beta(f(\alpha_n^{-1}(\mathbf{x})))$, where

$\alpha_n^{-1}(\mathbf{x}) = (\alpha^{-1}(x_1), \ldots, \alpha^{-1}(x_n))$.

Shortly: $f^c = \beta \circ f \circ \alpha_n^{-1}$. Then $f = \beta^{-1} \circ f^c \circ \alpha_n$.

Clearly: f is computable iff f^c is computable.

Def. 8. A function $f : D^n \to E$ is said to be *partial recursive*, if $f^c \in \mathrm{REC}$.

(Gen-CT) For any domains (D,α), (E,β), the functions $f : D^n \to E$, $n \geq 1$, computable in a general sense coincide with partial recursive functions in the sense of Def. 8.

This immediately follows from (CT).

Example. Let (D,α) be a domain. Then, $\alpha : D \mapsto \mathbb{N}$ is recursive. Now $\beta : \mathbb{N} \mapsto \mathbb{N}$ is the identity function I_1^1.

We have: $\alpha^c = \beta \circ \alpha \circ \alpha^{-1} = \beta$ and $\beta \in \mathrm{REC}$. So $\alpha^c \in \mathrm{REC}$.

Consequently $\langle\cdot\rangle$ is a total recursive function from $\mathbb{N}^*$ to $\mathbb{N}$, if it is treated as a unary function on the domain $(\mathbb{N}^*, \langle\cdot\rangle)$.

5. Three theorems on recursive functions

Theorem 6. (*s-m-n theorem*) Let $m, n \geq 1$. There exists a total recursive function $s : \mathbb{N}^{m+1} \mapsto \mathbb{N}$ such that for all $e \in \mathbb{N}$, $\mathbf{x} \in \mathbb{N}^n$ and $y_1, \ldots, y_m \in \mathbb{N}$ the following equation holds:

$$\{s(e, y_1, \ldots, y_m)\}^{(n)}(\mathbf{x}) \simeq \{e\}^{(n+m)}(\mathbf{x}, y_1, \ldots, y_m).$$

PROOF. We transform P_e into a program Q such that
$\ulcorner Q \urcorner = s(e, y_1, \ldots, y_m)$.

The idea:

Entry $|x_1| \ldots |x_n|$

Then $|x_1| \ldots |x_n| y_1| \ldots |y_m|$ (put y_i in R_i for $n + 1 \leq i \leq n + m$)

Run P_e

$S(n+1)$

$\vdots \qquad y_1$ times

$S(n+1)$

$\vdots$

$S(n+m)$

$\vdots \qquad y_m$ times

$S(n+m)$;

P_e

Clearly Q depends on $e, y_1, \ldots, y_m$. The function $s(e, y_1, \ldots, y_m) = \ulcorner Q \urcorner$ is computable. Applying (CT) positively, one may infer that s is recursive.

We provide a complete proof.

$x * y = \mu z$

$[lh(z) = lh(x) + lh(y) \wedge \forall_{i<lh(x)} (z)_{i+1} = (x)_{i+1} \wedge \forall_{i<lh(y)} (z)_{lh(x)+i+1} = (y)_{i+1}]$

We need $f_1(y) = \ulcorner(S(n+1), \ldots, S(n+1))\urcorner$, where $S(n+1)$ occurs y times. Recall that $\nu(S(n+1)) = 4n+1$.

$$\begin{cases} f_1(0) & = & 0 \\ f_1(y+1) & = & f_1(y) * \langle 4n+1 \rangle \end{cases}$$

In a similar way we define $f_k(y) = \ulcorner(S(n+k), \ldots, S(n+k))\urcorner$, where $S(n+k)$ occurs y times, for $1 < k \leq m$.

We also need: $g(e, y) =$ the number of the program resulting from P_e after one has replaced each $J(-, -, q)$ with $J(-, -, q+y)$.

We define:

$s(e, y_1, \ldots, y_m) = f_1(y_1) * \cdots * f_m(y_m) * g(e, y_1 + \cdots + y_m)$.

$g(e, y) = \mu z\, [lh(z) = lh(e) \wedge$

$\wedge \forall_{i<lh(e)}(jp((e)_{i+1}) = 0 \Rightarrow (z)_{i+1} = (e)_{i+1}) \wedge$

$\wedge \forall_{i<lh(e)}(jp((e)_{i+1}) \neq 0 \Rightarrow$

$\Rightarrow (z)_{i+1} = 4\tau(rg_1((e)_{i+1}), rg_2((e)_{i+1}), jp((e)_{i+1}) + y) + 3)]$

q.e.d.

Theorem 6'. Let $f : \mathbb{N}^{n+m} \to \mathbb{N}$ be partial recursive. There exists a total recursive function $s' : \mathbb{N}^m \mapsto \mathbb{N}$ such that for all $\mathbf{x} \in \mathbb{N}^n$ and all $y_1, \ldots, y_m \in \mathbb{N}$ the following equation holds:

$$\{s'(y_1, \ldots, y_m)\}^{(n)}(\mathbf{x}) \simeq f(\mathbf{x}, y_1, \ldots, y_m).$$

Proof. Let e be an index of f. Then $s'(y_1, \ldots, y_m) = s(e, y_1, \ldots, y_m)$.
q.e.d.

Proposition 4. Let $k, n \geq 1$. There exists a total recursive function $sb : \mathbb{N}^{k+1} \mapsto \mathbb{N}$ such that for all $e, e_1, \ldots, e_k \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{N}^n$ the following equation holds:

$$\{sb(e, e_1, \ldots, e_{(k)})\}^{(n)}(\mathbf{x}) \simeq \{e\}^k(\{e_1\}^{(n)}(\mathbf{x}), \ldots, \{e_k\}^{(n)}(\mathbf{x})).$$

Proof. We define:

$f(\mathbf{x}, e, e_1, \ldots, e_k) \simeq \{e\}^{(k)}(\{e_1\}^{(n)}(\mathbf{x}), \ldots, \{e_k\}^{(n)}(\mathbf{x})) =$

$= U_k(e, U_n(e_1, \mathbf{x}), \ldots, U_n(e_k, \mathbf{x}))$.

By Theorem 6', there exists a total recursive function s' such that:

$\{s'(e, e_1, \ldots, e_k)\}^{(n)}(\mathbf{x}) \simeq f(\mathbf{x}, e, e_1, \ldots, e_k)$.

We take $sb = s'$. q.e.d.

This shows that there is a program which from indices of any functions f ($k-$ary) and $g_1, \ldots, g_k$ ($n-$ary) computes an index of the function h which arises from $f, g_1, \ldots, g_k$ by substitution.

Unary relations $R \subseteq \mathbb{N}$ are subsets of $\mathbb{N}$. We denote them by A, B, C. We write $x \in A$ for $A(x)$.

Def. 9. For $A, B \subseteq \mathbb{N}$, we define a relation $\leq_m$ as follows: $A \leq_m B$ iff there exists a total recursive function $f : \mathbb{N} \mapsto \mathbb{N}$ such that

$$\forall_{x \in \mathbb{N}}(x \in A \Leftrightarrow f(x) \in B).$$

We read $A \leq_m B$ as: A is *many-one-reducible* to B.

The subscript m stems from ‘many-one’. One also considers a more restricted relation $A \leq_1 B$, where f is required to be one-one.

Proposition 5. If $A \leq_m B$ and B is recursive, then A is recursive.

Proof. We have $c_A(x) = c_B(f(x))$. q.e.d.

By $\mathrm{REC}^{(n)}$ we denote the family of $n-$ary partial recursive functions.

A family $\mathcal{F} \subseteq \mathrm{REC}^{(1)}$ is said to be *non-trivial*, if $\mathcal{F}$ is nonempty and different from $\mathrm{REC}^{(1)}$.

Theorem 7. (*Rice's theorem*) Let $\mathcal{F} \subseteq \mathrm{REC}^{(1)}$ be non-trivial. Then, the set:

$$A_{\mathcal{F}} = \{e \in \mathbb{N} : \{e\} \in \mathcal{F}\}$$

is not recursive.

PROOF. First, assume additionally $\emptyset \notin \mathcal{F}$.

We fix a function $f \in \mathcal{F}$. Then, $f(x) \neq \infty$ for some x.

We define a function:

$$h(x, e) \simeq \begin{cases} f(x) & \text{if } \mathrm{HALT}(e) \\ \infty & \text{otherwise} \end{cases}$$

We have: $h(x, e) \simeq f(x) + (U_1(e, e) \dot{-} U_1(e, e))$. So $h \in \mathrm{REC}$.

By Theorem 6', there exists a total recursive function $s : \mathbb{N} \mapsto \mathbb{N}$ such that

$$\{s(e)\}(x) \simeq h(x, e) \text{ for all } e, x \in \mathbb{N}\,.$$

The following implications are true.

$\mathrm{HALT}(e) \Rightarrow \forall_x(h(x, e) \simeq f(x)) \Rightarrow \forall_x(\{s(e)\}(x) \simeq f(x)) \Rightarrow$

$\Rightarrow \{s(e)\} = f \Rightarrow s(e) \in A_{\mathcal{F}}$

$\neg\, \mathrm{HALT}(e) \Rightarrow \forall_x(h(x, e) \simeq \infty) \Rightarrow \forall_x(\{s(e)\}(x) \simeq \infty) \Rightarrow$

$\Rightarrow \{s(e)\} = \emptyset \Rightarrow s(e) \notin A_{\mathcal{F}}$

Consequently, $\forall_x(e \in \mathrm{HALT} \Leftrightarrow s(e) \in A_{\mathcal{F}})$, which yields $\mathrm{HALT} \leq_m A_{\mathcal{F}}$.

By Theorem 5 and Proposition 5, $A_{\mathcal{F}}$ is not recursive.

If $\emptyset \in \mathcal{F}$, we prove as above that $\mathbb{N} \setminus A_{\mathcal{F}} = \{e \in \mathbb{N} : \{e\} \notin \mathcal{F}\}$ is not recursive, Then, $A_{\mathcal{F}}$ is not recursive, by (R2). q.e.d.

Corollary 3. The following relations are not recursive:

(1) $R_{m,k}(e) \Leftrightarrow \{e\}(m) \simeq k$, for fixed m, k,

(2) $R(e, x, y) \Leftrightarrow \{e\}(x) \simeq y$,

(3) $R(e) \Leftrightarrow \forall_x(\{e\}(x) \not\simeq \infty)$ (i.e. $\{e\}$ is total),

(4) $R(e) \Leftrightarrow \forall_x(\{e\}(x) \simeq \infty)$ (i.e. $\{e\}$ is empty),

(5) $R_f(e) \Leftrightarrow \{e\} = f$ for fixed $f \in \mathrm{REC}^{(1)}$,

(6) $R(e_1, e_2) \Leftrightarrow \{e_1\} = \{e_2\}$.

By (CT), the following problems are unsolvable:

(1) $P(m) \downarrow k$, for an arbitrary program P and fixed m, k, (2) $P(x) \downarrow y$, for arbitrary P, x, y, (3) f_P is total, for an arbitrary P, (4) f_P is empty, for an arbitrary P, (5) $f_P = f$, for an arbitrary P and a fixed $f \in \mathrm{REC}^{(1)}$, (6) $f_P = f_Q$, for arbitrary P, Q.

Theorem 8. (*the 2nd recursion theorem*) For any $f \in \mathrm{REC}^{(n+1)}$ there exists $e \in \mathbb{N}$ such that:

$$\forall_{\mathbf{x}}\,(f(\mathbf{x}, e) = \{e\}^{(n)}(\mathbf{x})).$$

Proof. We fix $f \in \mathrm{REC}^{(n)}$. We define a function:

(1) $g(\mathbf{x}, y) \simeq \{y\}^{(n+1)}(\mathbf{x}, y) \simeq U_{n+1}(y, \mathbf{x}, y)$.

By Corollary 2 and (R1), $g \in \mathrm{REC}$. By Theorem 6', there exists a total recursive function $s : \mathbb{N} \mapsto \mathbb{N}$ such that:

(2) $\forall_{\mathbf{x},y}\,(\{s(y)\}^{(n)}(\mathbf{x}) \simeq g(\mathbf{x}, y))$.

We define a partial recursive function:

(3) $h(\mathbf{x}, y) \simeq f(\mathbf{x}, s(y))$.

Let a be an index of h. Then:

(4) $\forall_{\mathbf{x},y}\,(h(\mathbf{x}, y) \simeq \{a\}^{(n+1)}(\mathbf{x}, y))$.

We define $e = s(a)$.

We have:

$$f(\mathbf{x}, e) \simeq f(\mathbf{x}, s(a)) \overset{(3)}{\simeq} h(\mathbf{x}, a) \overset{(4)}{\simeq} \{a\}^{(n+1)}(\mathbf{x}, a) \simeq$$

$$\overset{(1)}{\simeq} g(\mathbf{x}, a) \overset{(2)}{\simeq} \{s(a)\}^{(n)}(\mathbf{x}) \simeq \{e\}^{(n)}(\mathbf{x}).$$

q.e.d.

Theorems 6, 6’, 8 are due to S.C. Kleene.

Example B. There exists $e \in \mathbf{N}$ such that:

$\forall_x (\{e\}(x) = e)$.

We consider the function $g(x, y) = y$. By Theorem 8, there exists e such that $\{e\}(x) \simeq g(x, e) = e$ for all x. So $\{e\}(x) = e$ for all x.

Proposition 6. Let $n \geq 1$. There exists a total recursive function $rc \in \mathrm{REC}^{(2)}$ such that the following equations hold for all $\mathbf{x} \in \mathbb{N}^n$, $y, e_1, e_2 \in \mathbb{N}$.

$\{rc(e_1, e_2)\}^{(n+1)}(\mathbf{x}, 0) \simeq \{e_1\}^{(n)}(\mathbf{x})$

$\{rc(e_1, e_2)\}^{(n+1)}(\mathbf{x}, y + 1) \simeq \{e_2\}^{(n+2)}(\mathbf{x}, y, \{rc(e_1, e_2)\}^{(n+1)}(\mathbf{x}, y))$

Proof. We define a partial recursive function.

$$g(\mathbf{x}, y, e_1, e_2, e) \simeq \begin{cases} \{e_1\}^{(n)}(\mathbf{x}) & \text{if } y = 0 \\ \{e_2\}^{(n+2)}(\mathbf{x}, y \dot{-} 1, \{e\}^{(n+3)}(\ldots)) & \text{if } y \neq 0 \end{cases}$$

Here $\ldots$ stands for $\mathbf{x}, y \dot{-} 1, e_1, e_2$. By Theorem 8, there exists $a \in \mathbb{N}$ such that: $g(\mathbf{x}, y, e_1, e_2, a) \simeq \{a\}^{(n+3)}(\mathbf{x}, y, e_1, e_2)$ for all $\mathbf{x}, y, e_1, e_2$.

By Theorem 6', there exists a total function $s \in \mathrm{REC}^{(3)}$ such that: $g(\mathbf{x}, y, e_1, e_2, e) \simeq \{s(e_1, e_2, e)\}^{(n+1)}(\mathbf{x}, y)$ for all $\mathbf{x}, y, e_1, e_2, e$.

We define: $rc(e_1, e_2) = s(e_1, e_2, a)$. q.e.d.

Example C. *The Ackermann function*

The Ackermann function A is defined by the following recursive equations. A is not primitive recursive.

$A(0, y) = y + 1$

$A(x + 1, 0) = A(x, 1)$

$A(x + 1, y + 1) = A(x, A(x + 1, y))$

We consider the lexicographical ordering on $\mathbb{N}^2$.

$(x, y) \leq_l (x', y') \Leftrightarrow x < x' \vee (x = x' \wedge y \leq y')$

This is a well-ordering: reflexive, antisymmetric, transitive, total, and satisfying the condition

(WO) every nonempty subset of $\mathbb{N}^2$ has a minimal element (the least element in this subset).

$(0, 0) <_l (0, 1) <_l (0, 2) <_l \ldots <_l (1, 0) <_l (1, 1) <_l (1, 2) <_l \ldots$

The pairs on the right-hand side of the second and the third defining equation are less than the pair on the left-hand side of this equation. Therefore these equations correctly define a unique function by induction on $\leq_l$.

$A(1,1) = A(0, A(1,0)) = A(0, A(0,1)) = A(0,2) = 3$

$A(2,1) = A(1, A(2,0)) = A(1, A(1,1)) = A(1,3) = A(0, A(1,2)) = A(0, A(0, A(1,1))) = A(0, A(0,3)) = A(0,4) = 5$

We show $A \in \mathrm{REC}$. We define a partial recursive function:

$$g(x,y,e) \simeq \begin{cases} y+1 & \text{if } x = 0 \\ \{e\}^{(2)}(x \dot{-} 1, 1) & \text{if } x \neq 0 \wedge y = 0 \\ \{e\}^{(2)}(x \dot{-} 1, \{e\}^{2}(x, y \dot{-} 1)) & \text{if } x \neq 0 \wedge y \neq 0 \end{cases}$$

By Theorem 8, there exists $e \in \mathbb{N}$ such that $\{e\}^{(2)}(x,y) \simeq g(x,y,e)$ for all x, y. Clearly $A = \{e\}^{(2)}$, hence $A \in \mathrm{REC}$.

6. Recursively enumerable relations

Def. 10. A relation $R \subseteq \mathbb{N}^n$ is said to be *recursively enumerable*, if there exists a recursive relation $S \subseteq \mathbb{N}^{n+1}$ such that:

$$\forall_{\mathbf{x}} (R(\mathbf{x}) \Leftrightarrow \exists_y S(\mathbf{x}, y)).$$

One often writes r.e. for 'recursively enumerable'.

Proposition 7. Every recursive relation is r.e.

Proof. Let $R \subseteq \mathbb{N}^n$ be recursive. We have:
$\forall_{\mathbf{x}} (R(\mathbf{x}) \Leftrightarrow \exists_y (R(\mathbf{x}) \wedge y = y))$.

Example D. HALT and $\mathrm{HALT}^{(2)}$ are r.e.

$\mathrm{HALT}^{(2)}(x, y) \Leftrightarrow \exists_t (j_1(x, y, t) = 0)$

$\mathrm{HALT}(x) \Leftrightarrow \exists_t (j_1(x, x, t) = 0)$

Corollary 4. Not every r.e. relation is recursive.

(RE.1) Let $R \subseteq \mathbb{N}^k$ be r.e., and let $g_1, \ldots, g_k : \mathbb{N}^n \mapsto \mathbb{N}$ be recursive (and total). Then, the relation $T \subseteq \mathbb{N}^n$, defined by:

$$T(\mathbf{x}) \Leftrightarrow R(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})),$$

is r.e.

Proof. There exists a recursive relation S such that:

$R(\mathbf{x}) \Leftrightarrow \exists_y S(\mathbf{x}, y)$.

Accordingly:

$T(\mathbf{x}) \Leftrightarrow \exists_y S(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}), y)$.

By (R1) and Def. 10, T is r.e.. q.e.d.

(RE.2) If relations R_1, R_2 are r.e., then the relations $R_1 \vee R_2$, $R_1 \wedge R_2$ are r.e..

Proof. We have:

$\exists_y S_1(\mathbf{x}, y) \vee \exists_y S_2(\mathbf{x}, y) \Leftrightarrow \exists_y (S_1(\mathbf{x}, y) \vee S_2(\mathbf{x}, y))$,

$\exists_y S_1(\mathbf{x}, y) \wedge \exists_y S_2(\mathbf{x}, y) \Leftrightarrow \exists_z (S_1(\mathbf{x}, \pi_1(z)) \wedge S_2(\mathbf{x}, \pi_2(z)))$. q.e.d.

(RE.3) If $R \subseteq \mathbb{N}^{n+1}$ is r.e., then the relation $T \subseteq \mathbb{N}^n$, defined by:

$T(\mathbf{x}) \Leftrightarrow \exists_z R(\mathbf{x}, z)$,

is r.e..

Proof. Let S be recursive and: $R(\mathbf{x}, z) \Leftrightarrow \exists_y S(\mathbf{x}, z, y)$. We have:

$\exists_z \exists_y S(\mathbf{x}, z, y) \Leftrightarrow \exists_z S(\mathbf{x}, \pi_1(z), \pi_2(z))$. q.e.d.

Theorem 9. For any $R \subseteq \mathbb{N}^n$ the following conditions are equivalent:

(i) R is r.e.,

(ii) there exists a partial recursive function f such that $R = \mathrm{Dom}(f)$.

Proof. We show (i) $\Rightarrow$ (ii). Let R be r.e. There exists a recursive relation S such that: $R(\mathbf{x}) \Leftrightarrow \exists_y S(\mathbf{x}, y)$. We define:

$f(\mathbf{x}) = \mu y\, S(\mathbf{x}, y)$.

Clearly $f \in \mathrm{REC}$, by (R3), and $R = \mathrm{Dom}(f)$.

We show (ii) $\Rightarrow$ (i). Let $f \in \mathrm{REC}$ be n−ary, and let $R = \mathrm{Dom}(f)$. Let e be an index of f. We have:

$R(\mathbf{x}) \Leftrightarrow \exists_t\, (j_n(e, \mathbf{x}, t) = 0)$.

So R is r.e.. q.e.d.

Accordingly, the recursively enumerable relations are precisely the domains of partial recursive functions.

Theorem 10. (*the Post theorem*) For any relation $R \subseteq \mathbb{N}^n$ the following conditions are equivalent:

(i) R is recursive,

(ii) both R and $\neg R$ are r.e..

Proof. (i) $\Rightarrow$ (ii). Let R be recursive. Then, $\neg R$ is recursive, by (R2). So R and $\neg R$ are r.e., by Proposition 7.

We show (ii) $\Rightarrow$ (i). Assume that R and $\neg R$ are r.e.. There exist recursive relations S_1, S_2 such that:

$R(\mathbf{x}) \Leftrightarrow \exists_y S_1(\mathbf{x}, y), \quad \neg R(\mathbf{x}) \Leftrightarrow \exists_y S_2(\mathbf{x}, y)$.

We define: $f(\mathbf{x}) = \mu y(S_1(\mathbf{x}, y) \vee S_2(\mathbf{x}, y))$.

By (R2), (R3), $f \in \mathrm{REC}$. We have: $\forall_{\mathbf{x}} \exists_y (S_1(\mathbf{x}, y) \vee S_2(\mathbf{x}, y))$.

Consequently, f is total, and $R(\mathbf{x}) \Leftrightarrow S_1(\mathbf{x}, f(\mathbf{x}))$. So R is recursive, by (R1). q.e.d.

Example E. The relations $\neg$ HALT, $\neg$ HALT$^{(2)}$ are not r.e..

We know that HALT and HALT$^{(2)}$ are r.e., but not recursive. By Theorem 10, their negations cannot be r.e.. We have:

$\neg\mathrm{HALT}^{(2)}(x, y) \Leftrightarrow \forall_t \neg(j_1(x, y, t) = 0)$.

It follows that the relation $\forall_y R(\mathbf{x}, y)$ need not be r.e. for a recursive relation R.

Def. 11. Let $f : \mathbb{N}^n \to \mathbb{N}$. The relation $G_f \subseteq \mathbb{N}^{n+1}$, defined by:

$G_f(\mathbf{x}, y) \Leftrightarrow f(\mathbf{x}) \simeq y$,

is called *the graph of* f.

Theorem 11. (*the graph theorem*) For any $f : \mathbb{N}^n \to \mathbb{N}$ the following conditions are equivalent:

(i) $f \in \mathrm{REC}$,

(ii) G_f is r.e..

PROOF. (i) $\Rightarrow$ (ii). Assume (i). Let e be an index of f. We have:

$G_f(\mathbf{x}, y) \Leftrightarrow \exists_t\, S_n(e, \mathbf{x}, y, t)$.

(ii) $\Rightarrow$ (i). Assume (ii). There exists a recursive relation S such that:

$G_f(\mathbf{x}, y) \Leftrightarrow \exists_z\, S(\mathbf{x}, y, z)$.

We define: $g(\mathbf{x}) \simeq \mu u\, S(\mathbf{x}, \pi_1(u), \pi_2(u))$.

$g \in \mathrm{REC}$, by (R3). We will show: $f(\mathbf{x}) \simeq \pi_1(g(\mathbf{x}))$.

So $f \in \mathrm{REC}$.

We show $f(\mathbf{x}) \simeq \pi_1(g(\mathbf{x}))$.

$f(\mathbf{x}) = \infty \Rightarrow \neg\exists_y\, G_f(\mathbf{x}, y) \Rightarrow \neg\exists_y\, \exists_z\, S(\mathbf{x}, y, z) \Rightarrow$

$\Rightarrow \neg\exists_u\, S(\mathbf{x}, \pi_1(u), \pi_2(u)) \Rightarrow g(\mathbf{x}) = \infty$

Now, assume $f(\mathbf{x}) \neq \infty$. There exists a unique y such that $G_f(\mathbf{x}, y)$; clearly $y = f(\mathbf{x})$. Consequently, there exists z such that $S(\mathbf{x}, y, z)$, hence there exists u such that $S(\mathbf{x}, \pi_1(u), \pi_2(u))$; take $u = \pi(y, z)$.

So $g(\mathbf{x}) \neq \infty$ and $S(\mathbf{x}, \pi_1(g(\mathbf{x})), \pi_2(g(\mathbf{x})))$. This yields $\exists_z\, S(\mathbf{x}, \pi_1(g(\mathbf{x})), z)$, and consequently $G_f(\mathbf{x}, \pi_1(g(\mathbf{x})))$. By the uniqueness of y, $y = \pi_1(g(\mathbf{x}))$.

Hence $f(\mathbf{x})$ is defined iff $g(\mathbf{x})$ is defined iff $\pi_1(g(\mathbf{x}))$ is defined, and the desired equation holds. q.e.d.

(RE.4) If $g_1, \dots, g_k : \mathbb{N}^n \to \mathbb{N}$ are partial recursive and relations $R_1, \dots, R_k \subseteq \mathbb{N}^n$ are r.e. and satisfy the condition:

($\bullet$) for any $\mathbf{x} \in \mathbb{N}^n$ there is at most one $1 \leq i \leq k$ such that $R_i(\mathbf{x})$,

then the function h, defined by:

$$h(\mathbf{x}) \simeq \begin{cases} g_1(\mathbf{x}) & \text{if } R_1(\mathbf{x}) \\ \vdots & \\ g_k(\mathbf{x}) & \text{if } R_k(\mathbf{x}) \\ \infty & \text{otherwise} \end{cases}$$

is partial recursive.

Proof. We have:

$h(\mathbf{x}) \simeq y \Leftrightarrow (R_1(\mathbf{x}) \wedge g_1(\mathbf{x}) \simeq y) \vee \cdots \vee (R_k(\mathbf{x}) \wedge g_k(\mathbf{x}) \simeq y)$.

G_h is r.e., by Theorem 11 and (RE.2). So $h \in$ REC, by Theorem 11. q.e.d.

Example. In the proof of Rice's theorem we used the function:

$$h(x,e) \simeq \begin{cases} f(x) & \text{if HALT}(e) \\ \infty & \text{otherwise} \end{cases}$$

We inferred $h \in \mathrm{REC}$ from $h(x,e) = f(x) + (U_1(e,e) \dot{-} U_1(e,e))$. Now, this follows from (RE.4).

Proposition 8. Let $A, B \subseteq \mathbb{N}$. If $A \leq_m B$ and B is r.e., then A is r.e.

Proof. Assume $A \leq_m B$. Then: $x \in A \Leftrightarrow f(x) \in B$, for all x. So $A(x) \Leftrightarrow B(f(x))$, for all x. So A is r.e., if B is r.e., by (RE.1). q.e.d.

Theorem 12. For any $A \subseteq \mathbb{N}$, $A \neq \emptyset$, the following conditions are equivalent:

(i) A is r.e.,

(ii) there exists a total recursive function $f : \mathbb{N} \mapsto \mathbb{N}$ such that $A = \mathrm{Rn}(f) = \{f(x) : x \in \mathbb{N}\}$.

PROOF. (ii) $\Rightarrow$ (i). Assume (ii). We have:

$\forall_y (y \in A \Leftrightarrow \exists_x f(x) = y)$.

Consequently, A is r.e..

(i) $\Rightarrow$ (ii). Assume (i). There exists a recursive relation S such that: $y \in A \Leftrightarrow \exists_x S(y, x)$, for all y.

We fix $k \in A$ and define:

$$f(x) = \begin{cases} \pi_1(x) & \text{if } S(\pi_1(x), \pi_2(x)) \\ k & \text{otherwise} \end{cases}$$

f is a total recursive function, by (R4). We show $A = \mathrm{Rn}(f)$.

Let $y \in A$. Then $S(y, x)$, for some x. Take $z = \pi(y, x)$. We have $S(\pi_1(z), \pi_2(z))$, hence $f(z) = \pi_1(z) = y$. So $y \in \mathrm{Rn}(f)$.

Let $y \in \mathrm{Rn}(f)$. Then $f(x) = y$, for some x. We consider two cases.

1°. $y = k$. Then $y \in A$.

2°. $y \neq k$. Then, for some $x \in \mathbb{N}$, $y = \pi_1(x)$ and $S(\pi_1(x), \pi_2(x))$. So $\pi_1(x) \in A$, hence $y \in A$. q.e.d.

A total recursive function $f : \mathbb{N} \mapsto \mathbb{N}$ is called *a recursive sequence*.

$f(n) = a_n$, for $n \in \mathbf{N}$. $f = (a_n)_{n \in \mathbb{N}}$.

A nonempty set is r.e. iff it is the set of all terms of some recursive sequence.

For a relation $R \subseteq \mathbb{N}^n$, one defines a partial function $c_R^{\sim} : \mathbb{N}^n \to \mathbb{N}$ as follows:

$$c_R^{\sim}(\mathbf{x}) \simeq \begin{cases} 1 & \text{if } R(\mathbf{x}) \\ \infty & \text{otherwise} \end{cases}$$

Proposition 9. R is r.e. iff $c_R^{\sim} \in \mathrm{REC}$.

PROOF. Assume that R is r.e.. Then $c_R^{\sim} \in \mathrm{REC}$, by (RE.4).

Assume $c_R^{\sim} \in \mathrm{REC}$. Then, $R = \mathrm{Dom}(c_R^{\sim})$ is r.e., by Theorem 9. q.e.d.

Every algorithm computing $c_R^{\sim}$ is called *a positive algorithm* for R. A positive algorithm for R can be characterized by the following conditions:

(P1) if $R(\mathbf{x})$ holds, then the algorithm returns 1 (yes),

(P2) if $R(\mathbf{x})$ fails, then the algorithm does not terminate,

for any entry $\mathbf{x}$.

Sometimes it is convenient to replace the second condition with:

(P2') if $R(\mathbf{x})$ fails, then the algorithm returns 0 (no) or does not terminate.

Proposition 10. *R* is r.e. iff there exists an algorithm, satisfying (P1), (P2').

Proof. ($\Rightarrow$) follows from Proposition 9. We prove ($\Leftarrow$). Let *P* be a program for RM, satisfying (P1), (P2'). This program computes a function f such that: $R(\mathbf{x}) \Leftrightarrow f(\mathbf{x}) \simeq 1$, for all $\mathbf{x}$. By Theorem 11, *R* is r.e.. q.e.d.

For relations (sets), which are r.e. but not recursive, only positive algorithms can be provided.

One defines:

$W_e^{(n)} = \mathrm{Dom}(\{e\}^{(n)})$.

One writes W_e for $W_e^{(1)}$. e is called *the index of* $W_e^{(n)}$.

$R \subseteq \mathbb{N}^n$ is r.e. iff there exists $e \in \mathbb{N}$ such that $R = W_e^{(n)}$. This follows from Theorem 9.

The arithmetical hierarchy (the Kleene-Mostowski hierarchy)

We define classes of numerical relations $\Sigma_k^0, \Pi_k^0, \Delta_k^0$ for $k \in \mathbb{N}$.

Σ_0^0 is the class of recursive relations.

$\Pi_k^0 = \{\neg R : R \in \Sigma_k^0\}$

$\Delta_k^0 = \Sigma_k^0 \cap \Pi_k^0$

Σ_{k+1}^0 consists of all relations $\exists_y R(\mathbf{x}, y)$ with $R \in \Pi_k^0$.