Recursive Function*

Xiaofeng Gao

Department of Computer Science and Engineering
Shanghai Jiao Tong University, P.R.China

CS363-Computability Theory

*Special thanks is given to Prof. Yuxi Fu for sharing his teaching materials.

Outline

The Basic Functions

Lemma. The following basic functions are computable.

1. The *zero function* $\mathbf{0}$.
2. The *successor function* $x+1$.
3. For each $n \geq 1$ and $1 \leq i \leq n$, the *projection function* U_i^n given by $U_i^n(x_1, \ldots, x_n) = x_i$.

Proof

These functions correspond to the arithmetic instructions for URM.

1. $\mathbf{0}$: program $Z(1)$;
2. $x+1$: program $S(1)$;
3. U_i^n: program $T(i, 1)$.

Substitution Theorem

Suppose that $f(y_1, \ldots, y_k)$ and $g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})$ are computable functions, where $\mathbf{x} = x_1, \ldots, x_n$. Then the function $h(\mathbf{x})$ given by

$$h(\mathbf{x}) \quad \simeq \quad f(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}))$$

is a computable function.

Question: what is the domain of definition of $h(\mathbf{x})$?

Note: $h(x)$ is defined iff $g_1(\mathbf{x}), \cdots, g_k(\mathbf{x})$ are all defined and $(g_1(\mathbf{x}), \cdots, g_k(\mathbf{x})) \in Dom(f)$. Thus, if f and $g_1, \cdots, g_k$ are all total functions, then h is total.

Proof (Construction)

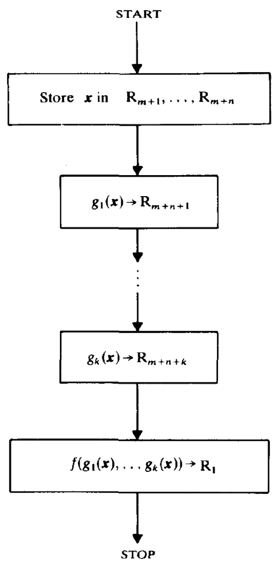
Let $F, G_1, \ldots, G_k$ be programs in standard form that compute $f, g_1, \ldots, g_k$.

Let m be $\max\{n, k, \rho(F), \rho(G_1), \ldots, \rho(G_k)\}$.

Registers:

$$[\ldots]_1^m [\mathbf{x}]_{m+1}^{m+n} [g_1(\mathbf{x})]_{m+n+1}^{m+n+1} \ldots [g_k(\mathbf{x})]_{m+n+k}^{m+n+k}$$

URM Program for Substitution



$$
\begin{array}{rcl}
I_1 & : & T(1, m+1) \\
 & \vdots & \\
I_n & : & T(n, m+n) \\
I_{n+1} & : & G_1[m+1, \ldots, m+n \;\to\; m+n+1] \\
 & \vdots & \\
I_{n+k} & : & G_k[m+1, \ldots, m+n \;\to\; m+n+k] \\
I_{n+k+1} & : & F[m+n+1 \ldots, m+n+k \;\to\; 1]
\end{array}
$$

Computable Function with Variable Sequences

Theorem. Suppose that $f(y_1, \ldots, y_k)$ is a computable function and that $x_{i_1}, \ldots, x_{i_k}$ is a sequence of k of the variables $x_1, \ldots, x_n$ (possibly with repetitions). Then the function h given by

$$h(x_1, \ldots, x_n) \quad \simeq \quad f(x_{i_1}, \ldots, x_{i_k})$$

is computable.

Proof. $h(\mathbf{x}) \simeq f(U_{i_1}^n(\mathbf{x}), \ldots, U_{i_k}^n(\mathbf{x}))$.

Form New Functions

- Rearrangement: $h_1(x_1, x_2) \simeq f(x_2, x_1)$;
- Identification: $h_2(x) \simeq f(x, x)$;
- Adding Dummy Variables: $h_3(x_1, x_2, x_3) \simeq f(x_2, x_3)$.

An Example

The function $f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3$ is computable.

Proof. Since $x + y$ is computable, by substituting $x_1 + x_2$ for x, and x_3 for y in $x + y$ we can claim that f is computable.

Note: When the functions $g_1, \cdots, g_k$ substituted into f, it is not necessarily involving all of the variables $x_1, \cdots, x_n$ to guarantee the computability of the new function.

Recursion Equations

Suppose that $f(\mathbf{x})$ and $g(\mathbf{x}, y, z)$ are functions. The function obtained from $f(\mathbf{x})$ and $g(\mathbf{x}, y, z)$ by recursion is defined as follows:

$$\begin{cases} h(\mathbf{x}, 0) \simeq f(\mathbf{x}), \\ h(\mathbf{x}, y+1) \simeq g(\mathbf{x}, y, h(\mathbf{x}, y)). \end{cases}$$

Domain of h

h may not be total unless both f and g are total.

The domain of h satisfies:

$$\begin{array}{lll} (\mathbf{x}, 0) \in Dom(h) & \text{iff} & \mathbf{x} \in Dom(f); \\ (\mathbf{x}, y+1) \in Dom(h) & \text{iff} & (\mathbf{x}, y) \in Dom(h) \\ & & \text{and } (\mathbf{x}, y, h(\mathbf{x}, y)) \in Dom(g). \end{array}$$

Uniqueness

Theorem. Let $\mathbf{x} = \{x_1, \cdots, x_n\}$,and suppose that $f(\mathbf{x})$ and $g(\mathbf{x}, y, z)$ are functions; then there is a unique function $h(\mathbf{x}, y)$ satisfying the recursion equations

$$\begin{cases} h(\mathbf{x}, 0) \simeq f(\mathbf{x}), \\ h(\mathbf{x}, y+1) \simeq g(\mathbf{x}, y, h(\mathbf{x}, y)). \end{cases}$$

Note: When $n = 0$ ($\mathbf{x}$ do not appear), the recursion equations take the form

$$\begin{cases} h(0) = a, \\ h(y+1) \simeq g(y, h(y)). \end{cases}$$

Computability Theorem

Theorem. $h(\mathbf{x}, y)$ is computable if $f(\mathbf{x})$ and $g(\mathbf{x}, y, z)$ are computable.

Proof

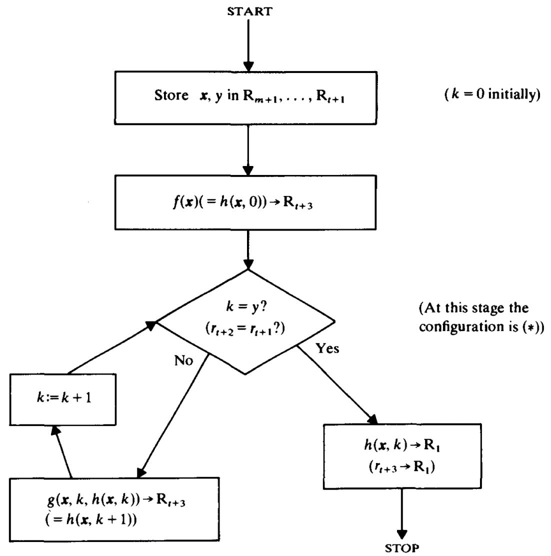
Registers:

$$[\ldots]_1^m [\mathbf{x}]_{m+1}^{m+n} [y]_{m+n+1}^{m+n+1} [k]_{m+n+2}^{m+n+2} [h(\mathbf{x}, k)]_{m+n+3}^{m+n+3}.$$

Program:

$$\begin{array}{lll} & & T(1, m+1) \\ & & \vdots \\ & & T(n+1, m+n+1) \\ & & F[1, 2, \ldots, n \rightarrow m+n+3] \\ I_q & : & J(n+m+2, n+m+1, p) \\ & & G[m+1, \ldots, m+n, m+n+2, m+n+3 \rightarrow m+n+3] \\ & & S(n+m+2) \\ & & J(1, 1, q) \\ I_p & : & T(n+m+3, 1) \end{array}$$

Flow Diagram



Addition

Let *add*: $\mathbb{N}^2 \to \mathbb{N}$, $add(x, y) := x + y$.

$$\begin{aligned} add(x, 0) &= x + 0 = x \\ add(x, y+1) &= x + (y+1) = (x+y) + 1 \\ &= add(x, y) + 1 \end{aligned}$$

Therefore,

$$\begin{aligned} add(x, 0) &= f(x) \\ add(x, y+1) &= g(x, y, add(x, y)) \end{aligned}$$

where

$$\begin{aligned} f : \mathbb{N} \to \mathbb{N}, &\quad f(x) := x, \\ g : \mathbb{N}^3 \to \mathbb{N}, &\quad g(x, y, z) := z + 1. \end{aligned}$$

Multiplication

Let *mult*: $\mathbb{N}^2 \to \mathbb{N}$, $mult(x, y) := x \cdot y$.

$$\begin{aligned} mult(x, 0) &= x \cdot 0 = 0 \\ mult(x, y+1) &= x \cdot (y+1) = x \cdot y + x \\ &= mult(x, y) + x \end{aligned}$$

Therefore,

$$\begin{aligned} mult(x, 0) &= f(x) \\ mult(x, y+1) &= g(x, y, mult(x, y)) \end{aligned}$$

where

$$\begin{aligned} f : \mathbb{N} \to \mathbb{N}, &\quad f(x) := 0, \\ g : \mathbb{N}^3 \to \mathbb{N}, &\quad g(x, y, z) := z + x. \end{aligned}$$

Power Function

Let *power*: $\mathbb{N}^2 \to \mathbb{N}$, $power(x, y) := x^y$

$$\begin{aligned} power(x, 0) &= x^0 \simeq 1 \\ power(x, y+1) &= x^{(y+1)} \simeq x^y \cdot x \end{aligned}$$

Therefore,

$$\begin{aligned} power(x, 0) &= f(x) \\ power(x, y+1) &= g(x, y, power(x)) \end{aligned}$$

where

$$\begin{aligned} f : \mathbb{N} \to \mathbb{N}, &\quad f(x) := 1, \\ g : \mathbb{N}^2 \to \mathbb{N}, &\quad g(x, y, z) := z \cdot x. \end{aligned}$$

Predecessor

Let *pred*: $\mathbb{N} \to \mathbb{N}$, $pred(x) := x \dot{-} 1 = \begin{cases} x - 1 & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$

$$\begin{aligned} pred(0) &= 0 \\ pred(x+1) &= x \end{aligned}$$

Therefore,

$$\begin{aligned} pred(0) &= f(x) = 0 \\ pred(x+1) &= g(x, pred(x)) \end{aligned}$$

where

$$\begin{aligned} f : \mathbb{N} \to \mathbb{N}, &\quad f(x) := 0, \\ g : \mathbb{N}^2 \to \mathbb{N}, &\quad g(x, y) := x. \end{aligned}$$

Conditional Subtraction

Let *sub*: $\mathbb{N}^2 \to \mathbb{N}$, $sub(x,y) := x \dot{-} y \stackrel{\text{def}}{=} \begin{cases} x - y, & \text{if } x \geq y, \\ 0, & \text{otherwise}. \end{cases}$

$$\begin{aligned} sub(x,0) &= x \dot{-} 0 \simeq x \\ sub(x,y+1) &= x \dot{-} (y+1) \simeq (x \dot{-} y) \dot{-} 1. \end{aligned}$$

Therefore,

$$\begin{aligned} sub(x,0) &= f(x) \\ sub(x,y+1) &= g(x,y,sub(x)) \end{aligned}$$

where

$$\begin{aligned} f : \mathbb{N} \to \mathbb{N}, &\quad f(x) := x, \\ g : \mathbb{N}^2 \to \mathbb{N}, &\quad g(x,y,z) := z \dot{-} 1. \end{aligned}$$

Sign

Let sg: $\mathbb{N} \to \mathbb{N}$,

$$\mathrm{sg}(x) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x = 0, \\ 1, & \text{if } x \neq 0. \end{cases} :$$

$$\begin{aligned} \mathrm{sg}(0) &\simeq 0, \\ \mathrm{sg}(x+1) &\simeq 1. \end{aligned}$$

$$\overline{\mathrm{sg}}(x) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } x = 0, \\ 0, & \text{if } x \neq 0. \end{cases} :$$

$$\overline{\mathrm{sg}}(x) \simeq 1 \dot{-} \mathrm{sg}(x).$$

Other Examples

Absolute Function (ABS): $|x - y| \simeq (x \dot{-} y) + (y \dot{-} x)$.

Factorial: $x!$

$$\begin{aligned} 0! &\simeq 1, \\ (x+1)! &\simeq x!(x+1). \end{aligned}$$

Minimum: $\min(x,y) \simeq x \dot{-} (x \dot{-} y)$.

Maximum: $\max(x,y) \simeq x + (y \dot{-} x)$.

Remainder

$\mathrm{rm}(x,y) \stackrel{\text{def}}{=}$ the remainder when y is devided by x:

$$\mathrm{rm}(x,y+1) \stackrel{\text{def}}{=} \begin{cases} \mathrm{rm}(x,y)+1, & \text{if } \mathrm{rm}(x,y)+1 \neq x, \\ 0, & \text{if } \mathrm{rm}(x,y)+1 = x. \end{cases}$$

The recursive definition is given by

$$\begin{aligned} \mathrm{rm}(x,0) &= 0, \\ \mathrm{rm}(x,y+1) &= (\mathrm{rm}(x,y)+1)\mathrm{sg}(|x - (\mathrm{rm}(x,y)+1)|). \end{aligned}$$

Quotient

$\mathsf{qt}(x,y) \stackrel{\text{def}}{=}$ the quotient when y is devided by x:

$$\mathsf{qt}(x,y+1) \stackrel{\text{def}}{=} \begin{cases} \mathsf{qt}(x,y)+1, & \text{if } \mathsf{rm}(x,y)+1 = x, \\ \mathsf{qt}(x,y), & \text{if } \mathsf{rm}(x,y)+1 \neq x. \end{cases}$$

The recursive definition is given by

$$\begin{aligned} \mathsf{qt}(x,0) &= 0, \\ \mathsf{qt}(x,y+1) &= \mathsf{qt}(x,y) + \overline{\mathsf{sg}}(|x-(\mathsf{rm}(x,y)+1)|). \end{aligned}$$

Conditional Division

$$\mathsf{div}(x,y) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } x|y, \\ 0, & \text{if } x\not| y. \end{cases} \quad : \quad \mathsf{div}(x,y) = \overline{\mathsf{sg}}(\mathsf{rm(x,y)}).$$

Definition by Cases

Suppose that $f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})$ are computable functions, and $M_1(\mathbf{x}), \ldots, M_k(\mathbf{x})$ are decidable predicates, such that for every $\mathbf{x}$ exactly one of $M_1(\mathbf{x}), \ldots, M_k(\mathbf{x})$ holds. Then the function $g(\mathbf{x})$ given by

$$g(\mathbf{x}) \simeq \begin{cases} f_1(\mathbf{x}), & \text{if } M_1(\mathbf{x}) \text{ holds}, \\ f_2(\mathbf{x}), & \text{if } M_2(\mathbf{x}) \text{ holds}, \\ \vdots & \\ f_k(\mathbf{x}), & \text{if } M_k(\mathbf{x}) \text{ holds}. \end{cases}$$

is computable.

Proof. $g(\mathbf{x}) \simeq c_{M_1}(\mathbf{x})f_1(\mathbf{x}) + \ldots + c_{M_k}(\mathbf{x})f_k(\mathbf{x})$.

Algebra of decidability

Suppose that $M(\mathbf{x})$ and $Q(\mathbf{x})$ are decidable predicates; then the following are also decidable.

1. *not* $M(\mathbf{x})$
2. $M(\mathbf{x})$ *and* $Q(\mathbf{x})$
3. $M(\mathbf{x})$ *or* $Q(\mathbf{x})$

Proof:

1. $1 \dot{-} c_M(\mathbf{x})$
2. $c_M(\mathbf{x}) \cdot c_Q(\mathbf{x})$
3. $\max(c_M(\mathbf{x}), c_Q(\mathbf{x}))$

Bounded Sum and Bounded Product

Bounded sum:

$$\begin{aligned}\sum_{z<0} f(\mathbf{x}, z) &\simeq 0, \\ \sum_{z<y+1} f(\mathbf{x}, z) &\simeq \sum_{z<y} f(\mathbf{x}, z) + f(\mathbf{x}, y)\end{aligned}$$

Bounded product:

$$\begin{aligned}\prod_{z<0} f(\mathbf{x}, z) &\simeq 1, \\ \prod_{z<y+1} f(\mathbf{x}, z) &\simeq (\prod_{z<y} f(\mathbf{x}, z)) \cdot f(\mathbf{x}, y)\end{aligned}$$

They are computable if $f(\mathbf{x}, z)$ is total and computable.

Bounded Sum and Bounded Product

By substitution the following functions are also computable

$$\sum_{z<k(\mathbf{x},\mathbf{w})} f(\mathbf{x}, z)$$

and

$$\prod_{z<k(\mathbf{x},\mathbf{w})} f(\mathbf{x}, z)$$

if $k(\mathbf{x}, \mathbf{w})$ is total and computable.

Bounded Minimization Operator, or μ-Operator

$\mu z < y(\cdots)$: the least z less than y such that $\cdots$

$$\mu z<y(f(\mathbf{x}, z) = 0) \stackrel{\text{def}}{=} \begin{cases} \text{the least } z < y, & \text{such that } f(\mathbf{x}, z) = 0; \\ y & \text{if there is no such } z. \end{cases}$$

μ-Operator

Theorem.

If $f(\mathbf{x}, z)$ is total and computable, then so is $\mu z<y\,(f(\mathbf{x}, z) = 0)$.

Proof

Consider $h(\mathbf{x}, v) = \prod_{u \leq v} \mathsf{sg}(f(\mathbf{x}, u))$ (Computable).

Given $\mathbf{x}$, y, suppose $z_0 = \mu z < y(f((x), y) = 0)$. Easy to see,

if $v < z_0$, then $h((x), v) = 1$;

if $z_0 \leq v < y$, then $h((x), v) = 0$;

Thus $z_0 = \sum_{v<y} h((x), v)$.

So $\mu z{<}y(f(\mathbf{x}, z) = 0) \simeq \sum_{v<y}(\prod_{u \leq v} \mathsf{sg}(f(\mathbf{x}, u)))$ is computable.

Bounded Minimization Operator, or μ-Operator

Corollary: If $f(\mathbf{x}, z)$ and $k(\mathbf{x}, \mathbf{w})$ are total and computable functions, then so is the function

$$\mu z{<}k(\mathbf{x}, \mathbf{w}) \ \ (f(\mathbf{x}, z) = 0).$$

Proof. By substitution of $k(\mathbf{x}, \mathbf{w})$ for y in the computable function $\mu z{<}y\,(f(\mathbf{x}, z) = 0)$.

Suppose that $R(\mathbf{x}, y)$ is a decidable predicates. Then the following statements are valid:

1. the function $f(\mathbf{x}, y) \simeq \mu z{<}y\ \ R(\mathbf{x}, y)$ is computable;
2. the following predicates are decidable:
   a) $M_1(\mathbf{x}, y) \equiv \forall z < y R(\mathbf{x}, z)$;
   b) $M_2(\mathbf{x}, y) \equiv \exists z < y R(\mathbf{x}, z)$.

Proof.

1. $f(\mathbf{x}, y) = \mu z < y(\overline{sg}(C_R(x, z)) = 0)$.
2. a) $c_{M_1}(\mathbf{x}, y) = \prod_{z<y} c_R(\mathbf{x}, z)$.
   b) $M_2(\mathbf{x}, y) \equiv$ not $(\forall z < y(\text{not } R(\mathbf{x}, z)))$

Theorem. The following functions are computable.

(a) $D(x) =$ the number of divisors of x;

(b) $Pr(x) = \begin{cases} 1, & \text{if } x \text{ is prime,} \\ 0, & \text{if } x \text{ is not prime.} \end{cases}$;

(c) $\mathsf{p}_x =$ the x-th prime number;

(d) $(x)_y = \begin{cases} k, & k \text{ is the exponent of } p_y \text{ in the prime factorisation of } x, \text{ for } x, y > 0, \\ 0, & \text{if } x = 0 \text{ or } y = 0. \end{cases}$.

Proof.

(a) $D(x) \simeq \sum_{y \leq x} \mathrm{div}(y, x)$.

(b) $Pr(x) \simeq \overline{\mathsf{sg}}(|D(x) - 2|)$.

(c) p_x can be recursively defined as follows:

$$\begin{aligned} \mathsf{p}_0 &\simeq 0, \\ \mathsf{p}_{x+1} &\simeq \mu z \leq (\mathsf{p}_x! + 1)(z > \mathsf{p}_x \text{ and } z \text{ is prime}). \end{aligned}$$

(d) $(x)_y \simeq \mu z < x(\mathsf{p}_y^{z+1} \nmid x)$.

Prime Coding

Suppose $s = (a_1, a_2, \ldots, a_n)$ is a finite sequence of numbers. It can be coded by the number

$$b = \mathsf{p}_1^{a_1+1}\mathsf{p}_2^{a_2+1} \ldots \mathsf{p}_n^{a_n+1}.$$

Then the length of s can be recovered from

$$\mu z < b((b)_{z+1} = 0),$$

and the i-th component can be recovered from

$$(b)_i \dot{-} 1.$$

Unbounded Minimization

μ-function:

$$\mu y(f(\mathbf{x}, y) = 0) \simeq \begin{cases} \text{the least } y \text{ such that} \\ \quad (i) \quad f(\mathbf{x}, y) \text{ is defined for all } z \leq y, \text{ and} \\ \quad (ii) \quad f(\mathbf{x}, y) = 0, \\ \text{undefined if otherwise.} \end{cases}$$

Theorem

If $f(\mathbf{x}, y)$ is computable, so is $\mu y(f(\mathbf{x}, y) = 0)$.

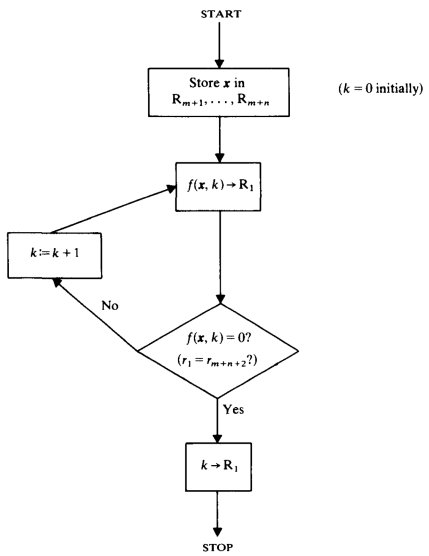
Proof

Let F be a program in standard form that computes $f(\mathbf{x}, y)$. Let m be $\max\{n+1, \rho(F)\}$.

Registers: $[\ldots]_1^m [\mathbf{x}]_{m+1}^{m+n} [k]_{m+n+1}^{m+n+1} [0]_{m+n+2}^{m+n+2}$.
Program:

$$\begin{array}{rcl} & & T(1, m+1) \\ & & \vdots \\ & & T(n, m+n) \\ I_p & : & F[m+1, m+2, \ldots, m+n+1 \rightarrow 1] \\ & & J(1, m+n+2, q) \\ & & S(m+n+1) \\ & & J(1, 1, p) \\ I_q & : & T(m+n+1, 1) \end{array}$$

Flow Diagram



Corollary

Suppose that $R(x, y)$ is a decidable predicate; then the function

$$\begin{aligned} g(x) &= \mu y R(\mathbf{x}, y) \\ &= \begin{cases} \text{the least } y \text{ such that } R(\mathbf{x}, y) \text{ holds,} & \text{if there is such a } y, \\ \text{undefined,} & \text{otherwise.} \end{cases} \end{aligned}$$

is computable.

Proof. $g(\mathbf{x}) = \mu y(\overline{\mathrm{sg}}(c_R(\mathbf{x}, y)) = 0)$.

Discussion

The μ-operator allows one to define partial functions.

E.g., given $f(x, y) = |x - y^2|$, $g(x) \simeq \mu y(f(x, y) = 0)$,

we have g is the non-total function

$$g(x) = \begin{cases} \sqrt{x}, & \text{if } x \text{ is a perfect square} \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

Remark

Using the μ-operator, one may define total functions that are not primitive recursive.

Remark: The set of primitive recursive functions are those definable from the basic functions using substitution and recursion.

Ackermann Function

The Ackermann function is defined as follows:

$$\begin{aligned}\psi(0, y) &\simeq y+1,\\ \psi(x+1, 0) &\simeq \psi(x, 1),\\ \psi(x+1, y+1) &\simeq \psi(x, \psi(x+1, y)).\end{aligned}$$

Ackermann Function

Fact. The Ackermann function is computable.

Definition. A finite set **S** of triples is said to be suitable if the followings hold:
(i) if $(0, y, z) \in S$ then $z = y + 1$;
(ii) if $(x + 1, 0, z) \in S$ then $(x, 1, z) \in S$;
(iii) if $(x + 1, y + 1, z) \in S$ then $\exists u.((x + 1, y, u) \in S) \wedge ((x, u, z) \in S)$.
Three conditions correspond to the three clauses in the definition of ψ.

The definition of a suitable set **S** ensures the following property:
If $(x, y, z) \in \mathbf{S}$, then
(i) $z = \psi(x, y)$;
(ii) **S** contains all the earlier triple $(x_1, y_1, \psi(x_1, y_1))$ that are needed to calculate $\psi(x, y)$.

Computability Proof

Moreover, for any particular pair of numbers (m, n) there is a suitable set **S** such that $(m, n, \psi(m, n)) \in \mathbf{S}$. For instance, let **S** be the set of triples $(x, y, \psi(x, y))$ that are used in the calculations of $\psi(m, n)$.

Note a triple (x, y, z) can be coded up by single positive number $2^x 3^y 5^z$. A finite set $\{u_1, \ldots, u_k\}$ can be coded up by $p_{u_1} \cdots p_{u_k}$.

Hence a finite set of triples can be coded by a single number v. Let $\mathbf{S}_v$ denote the set of triples coded by the number v. then

$$(x, y, z) \in \mathbf{S}_v \Leftrightarrow p_{2^x 3^y 5^z} \text{ divides } v.$$

So '$(x, y, z) \in \mathbf{S}_v$' is a decidable predicate of x, y, z, and v; and if it holds, then $x, y, z < v$.

Computability Proof (Cont.)

Let $R(x, y, v)$ be "v is a legal code and $\exists z < v((x, y, z) \in \mathbf{S}_v)$".

$R(x, y, v)$ is decidable using the techniques and functions of earlier sections.

Thus the function $f(x, y) = \mu v R(x, y, v)$ is a computable function that searches for the code of a suitable set containing (x, y, z) for some z.

As a result, the Ackermann function $\psi(x, y) = \mu z((x, y, z) \in \mathbf{S}_{f(x,y)})$ is computable.

Ackermann Function

The Ackermann function is not primitive recursive.
It grows faster than all the primitive recursive functions.