II. Recursive Function

Yuxi Fu

BASICS, Shanghai Jiao Tong University

Hilbert's Program

The epochal address of **David Hilbert** (23Jan.1862-14Feb.1934) to the International Congress of Mathematicians (Paris, 1900):

1. Consistency of the Axioms of Arithmetic.

This is the second among Hilbert's 23 problems published in 1900 (ten were announced in the Paris Congress).

2. Entscheidungsproblem (decision problem) of first order logic.

Gödel's Incompleteness Theorem

Hilbert retired in 1930, and gave a special speech to the annual meeting of the Society of German Scientists and Physicians in his birth place, Königsberg.

It was in this occasion he made his famous remark
"Wir müssen wissen. Wir werden wissen."

Kurt Gödel (28April.1906-14Jan.1978), a young man from Vienna and one year past his PhD, announced his Incompleteness Theorem in a roundtable discussion at the Conference on Epistemology held jointly with the Society meetings.
(one day before Hilbert's speech!)

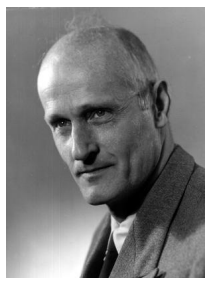
History of Recursion Theory

Gödel's remarkable idea is the arithmetization of syntax.

He used primitive recursive functions to do encoding.

In 1931 Gödel was aware of the Ackermann function.
By taking a suggestion from Herbrand, he developed in 1934 a formal system of Herbrand-Gödel recursive functions.

Kleene (5Jan.1909-25Jan.1994) introduced in 1936 the μ-recursive functions, based on the system of the primitive recursive functions and an unbounded search operator.



Synopsis

1. Primitive Recursive Function
2. Ackermann Function
3. Recursive Function

1. Primitive Recursive Function

Recursion Theory offers a mathematical model for the study of effective calculability.

- All effective objects can be encoded by natural numbers.
- All effective procedures can be modeled by functions from numbers to numbers.

Initial Function

1. The zero function $0 \stackrel{\text{def}}{=} \lambda x_1 \ldots \lambda x_n.0$.
2. The successor function $\mathsf{s}(x) \stackrel{\text{def}}{=} \lambda x.x+1$.
3. The projection function $\mathsf{U}_i^n(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \lambda x_1 \ldots \lambda x_n.x_i$.

Composition

Suppose $f(y_1, \ldots, y_k)$ is a k-ary function and $g_1(\widetilde{x}), \ldots, g_k(\widetilde{x})$ are n-ary functions, where $\widetilde{x}$ abbreviates $x_1, \ldots, x_n$.

The composition function $h(\widetilde{x})$ is defined by

$$h(\widetilde{x}) \quad = \quad f(g_1(\widetilde{x}), \ldots, g_k(\widetilde{x})),$$

Recursion

Suppose that $f(\widetilde{x})$ is an n-ary function and $g(\widetilde{x}, y, z)$ is an $(n+2)$-ary function.

The recursion function $h(\widetilde{x})$ is defined by

$$h(\widetilde{x}, 0) = f(\widetilde{x}), \tag{1}$$
$$h(\widetilde{x}, y+1) = g(\widetilde{x}, y, h(\widetilde{x}, y)). \tag{2}$$

Clearly there is a unique function that satisfies (1) and (2).

Primitive Recursive Recursion

The set of primitive recursive function is the least set generated from the initial functions, composition and recursion.

Dummy Parameter

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

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

is primitive recursive.

Proof.

$h(\widetilde{x}) = f(\mathrm{U}^n_{i_1}(\widetilde{x}), \ldots, \mathrm{U}^n_{i_k}(\widetilde{x}))$. □

Basic Arithmetic Function

$x + y$:

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

xy:

$$\begin{aligned} x0 &= 0, \\ x(y + 1) &= xy + x. \end{aligned}$$

x^y:

$$\begin{aligned} x^0 &= 1, \\ x^{y+1} &= x^y x. \end{aligned}$$

Basic Arithmetic Function

$x \dot{-} 1$:

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

$$x \dot{-} y \stackrel{\text{def}}{=} \begin{cases} x - y, & \text{if } x \geq y, \\ 0, & \text{otherwise.} \end{cases} :$$

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

Basic Arithmetic Function

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

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

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

$$\overline{\mathsf{sg}}(x) = 1 \dot{-} \mathsf{sg}(x).$$

Basic Arithmetic Function

$|x - y| = (x \dot{-} y) + (y \dot{-} x)$.

$x!$:

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

$\min(x, y) = x \dot{-} (x \dot{-} y)$.

$\max(x, y) = x + (y \dot{-} x)$.

Basic Arithmetic Function

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

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

The recursive definition is given by

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

Basic Arithmetic Function

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

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

The recursive definition is given by

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

Basic Arithmetic Function

$$\mathsf{div}(x,y) \overset{\text{def}}{=} \begin{cases} 1, & \text{if } x \text{ divides } y, \\ 0, & \text{otherwise.} \end{cases} :$$

$$\mathsf{div}(x,y) = \overline{\mathrm{sg}}(\mathsf{rm}(x,y)).$$

Bounded Sum and Bounded Product

Bounded sum:

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

Bounded product:

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

Bounded Sum and Bounded Product

By composition the following functions are also primitive recursive if $k(\widetilde{x}, \widetilde{w})$ is primitive recursive:

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

and

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

Bounded Minimization Operator

Bounded search:

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

Bounded Minimization Operator

Bounded search:

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

Proposition.

If $f(\widetilde{x},z)$ is primitive recursive, then so is $\mu z{<}y(f(\widetilde{x},z)=0)$.

Proof.

$\mu z{<}y(f(\widetilde{x},z)=0) = \sum_{v<y}(\prod_{u<v+1}\mathsf{sg}(f(\widetilde{x},u)))$. □

Bounded Minimization Operator

If $f(\widetilde{x}, z)$ and $k(\widetilde{x}, \widetilde{w})$ are primitive recursive functions, then so is the function

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

Primitive Recursive Predicate

Suppose $M(x_1, \ldots, x_n)$ is an n-ary predicate of natural numbers. The characteristic function $c_M(\widetilde{x})$, where $\widetilde{x} = x_1, \ldots, x_n$, is

$$c_M(a_1, \ldots, a_n) = \begin{cases} 1, & \text{if } M(a_1, \ldots, a_n) \text{ holds,} \\ 0, & \text{if otherwise.} \end{cases}$$

The predicate $M(\widetilde{x})$ is primitive recursive if c_M is primitive recursive.

Closure Property

Proposition. The following statements are valid:

- If $R(\widetilde{x})$ is a primitive recursive predicate, then so is $\neg R(\widetilde{x})$.
- If $R(\widetilde{x}), S(\widetilde{x})$ are primitive recursive predicates, then the following predicates are primitive recursive:
  - $R(\widetilde{x}) \wedge S(\widetilde{x})$;
  - $R(\widetilde{x}) \vee S(\widetilde{x})$.
- If $R(\widetilde{x}, y)$ is a primitive recursive predicate, then the following predicates are primitive recursive:
  - $\forall z < y.R(\widetilde{x}, z)$;
  - $\exists z < y.R(\widetilde{x}, z)$.

Proof.

For example $c_{\forall z<y.R(\widetilde{x},z)}(\widetilde{x}, y) = \prod_{z<y} c_R(\widetilde{x}, z)$. □

Definition by Case

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

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

is primitive recursive.

Proof.

$g(\widetilde{x}) = c_{M_1}(\widetilde{x}) f_1(\widetilde{x}) + \ldots + c_{M_k}(\widetilde{x}) f_k(\widetilde{x})$. □

More Arithmetic Function

Proposition. The following functions are primitive recursive.

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

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

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

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

More Arithmetic Function

Proof.

(i) $D(x) = \sum_{y<x+1} \mathrm{div}(y, x)$.

(ii) $Pr(x) = \overline{\mathrm{sg}}(|D(x) - 2|)$.

(iii) p_x can be recursively defined as follows:

$$\begin{aligned} \mathsf{p}_0 &= 0, \\ \mathsf{p}_{x+1} &= \mu z < (2 + \mathsf{p}_x!) \left(1 \dot{-} (z \dot{-} \mathsf{p}_x) Pr(z) = 0\right). \end{aligned}$$

(iv) $(x)_y = \mu z{<}x(\mathrm{div}(\mathsf{p}_y^{z+1}, x) = 0)$. □

Encoding a Finite Sequence

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

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

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

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

Not all Computable Functions are Primitive Recursive

Using the fact that all primitive recursive functions are total, a diagonalisation argument shows that non-primitive recursive computable functions must exist.

The same diagonalisation argument applies to all finite axiomatizations of computable total function.

2. Ackermann Function

Ackermann Function

The Ackermann function [1928] is defined as follows:

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

The equations clearly define a total function.

Ackermann Function is Monotone

The Ackermann function is monotone:

$$\begin{aligned} \psi(n, m) &< \psi(n, m+1), \\ \psi(n, m) &< \psi(n+1, m). \end{aligned}$$

The Ackermann function grows faster on the first parameter:

$$\psi(n, m+1) \leq \psi(n+1, m)$$

$\psi(n, m) + C$ is dominated by $\psi(J, m)$ for some large enough J:

$$\begin{aligned} \psi(n, m) + \psi(n', m) &< \psi(\max(n, n') + 4, m), \\ \psi(n, m) + m &< \psi(n+4, m). \end{aligned}$$

Ackermann Function Grows Fast

Lemma. Let $f(\widetilde{x})$ be a k-ary primitive recursive function. Then there exists some J such that for all $n_1, \ldots, n_k$ we have that

$$f(n_1, \ldots, n_k) < \psi(J, \sum_{i=1}^{k} n_k).$$

Proof.

The proof is by structural induction.

(i) f is one of the initial functions. In this case take J to be 1.

Ackermann Function Grows Fast

(ii) f is the composition function $h(g_1(\widetilde{x}), \ldots, g_m(\widetilde{x}))$. Then

$$\begin{aligned} f(\widetilde{n}) &= h(g_1(\widetilde{n}), \ldots, g_m(\widetilde{n})) \\ &< \psi(J_0, \sum_{i=1}^{m} g_i(\widetilde{n})) < \psi(J_0, \sum_{i=1}^{m} \psi(J_i, \sum_{j=1}^{k} n_j)) \\ &< \psi(J_0, \psi(J^*, \sum_{j=1}^{k} n_j)) < \psi(J^*, \psi(J^*+1, \sum_{j=1}^{k} n_j)) \\ &= \psi(J^*+1, \sum_{j=1}^{k} n_j + 1) \leq \psi(J^*+2, \sum_{j=1}^{k} n_j). \end{aligned}$$

Now set $J = J^* + 2$.

Ackermann Function Grows Fast

Suppose f is defined by the recursion:

$$\begin{aligned} f(\widetilde{x}, 0) &\simeq h(\widetilde{x}), \\ f(\widetilde{x}, y+1) &\simeq g(\widetilde{x}, y, f(\widetilde{x}, y)). \end{aligned}$$

Then $h(\widetilde{n}) < \psi(J_h, \sum \widetilde{n})$ and $g(\widetilde{n}, m, p) < \psi(J_g, \sum \widetilde{n} + m + p)$.

It is easy to prove

$$f(n_1, \ldots, n_k, m) < \psi(J, \sum_{i=1}^{k} n_k + m)$$

by induction on m. □

Faster than any Primitive Recursive Function

Now suppose $\psi(x, y)$ was primitive recursive.
By composition $\psi(x, x)$ would be primitive recursive.

According to the Lemma

$$\psi(n, n) < \psi(J, n)$$

for some J and all n, which would lead to the contradiction

$$\psi(J, J) < \psi(J, J).$$

Faster than any Primitive Recursive Function

The Ackermann function grows faster than every primitive recursive function.

3. Recursive Function

Minimization Operator, or Search Operator

Minimization function, or μ-function, or search function:

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

Here $\simeq$ is the computational equality.

Minimization Operator, or Search Operator

Minimization function, or μ-function, or search function:

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

Here $\simeq$ is the computational equality.

The recursion operation is a well-founded going-down procedure.

The search operation is a possibly divergent going-up procedure.

Recursive Function

The set of recursive functions is the least set generated from the initial functions, composition, recursion and minimization.

Decidable Predicate

A predicate $R(\widetilde{x})$ is decidable if its characteristic function

$$c_R(\widetilde{x}) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } R(\widetilde{x}) \text{ is true,} \\ 0, & \text{otherwise.} \end{cases}$$

is a recursive function.

Decidable Predicate

A predicate $R(\widetilde{x})$ is decidable if its characteristic function

$$c_R(\widetilde{x}) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } R(\widetilde{x}) \text{ is true,} \\ 0, & \text{otherwise.} \end{cases}$$

is a recursive function. The predicate $R(\widetilde{x})$ is partially decidable if its partial characteristic function

$$\chi_R(\widetilde{x}) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } R(\widetilde{x}) \text{ is true,} \\ \uparrow, & \text{otherwise.} \end{cases}$$

is a recursive function.

Closure Property

Proposition. The following statements are valid:

- If $R(\widetilde{x})$ is decidable, then so is $\neg R(\widetilde{x})$.
- If $R(\widetilde{x}), S(\widetilde{x})$ are (partially) decidable, then the following predicates are (partially) decidable:
  - $R(\widetilde{x}) \wedge S(\widetilde{x})$;
  - $R(\widetilde{x}) \vee S(\widetilde{x})$.
- If $R(\widetilde{x}, y)$ is (partially) decidable, then the following predicates are (partially) decidable:
  - $\forall z < y.R(\widetilde{x}, y)$;
  - $\exists z < y.R(\widetilde{x}, y)$.

Definition by Cases

Suppose $f_1(\widetilde{x}), \ldots, f_k(\widetilde{x})$ are recursive and $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ are partially decidable. For every $\widetilde{x}$ at most one of $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ holds. Then the function $g(\widetilde{x})$ given by

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

is recursive.

Minimization via Decidable Predicate

Suppose $R(x, y)$ is a partially decidable predicate. The function

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

is recursive.

Proof.

$g(\widetilde{x}) = \mu y(\overline{\mathsf{sg}}(\chi_R(\widetilde{x}, y)) = 0)$. □

Comment

The μ-operator allows one to define partial functions.

Comment

The μ-operator allows one to define partial functions.

The diagonalisation argument does not apply to the set $\mathfrak{F}$ of recursive functions.

Comment

The μ-operator allows one to define partial functions.

The diagonalisation argument does not apply to the set $\mathfrak{F}$ of recursive functions.

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

Minimization Operator is a Search Operator

It is clear from the above proof why the minimization operator is sometimes called a search operator.

Definable Function

A function is definable if there is a recursive function calculating it.

Ackermann Function is Recursive

Proposition. The Ackermann function is recursive.

Proof.

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 $\exists u.((x + 1, y, u) \in S \wedge (x, u, z) \in S)$.

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

Let $R(x, y, v)$ be "v is a legal code and $\exists z < v.(x, y, z) \in S_v$".

The Ackermann function $\psi(x, y) \simeq \mu z((x, y, z) \in S_{\mu v R(x,y,v)})$. $\square$

Do recursive functions capture all computable functions?

Do recursive functions capture all computable functions?

Gödel himself wasn't very sure about it in 1934.