VII. Problem Index

Yuxi Fu

BASICS, Shanghai Jiao Tong University

Motivation

By Church-Turing Thesis one may study computability theory using any of the computation models.

It is much more instructive however to carry out the study in a model independent manner.

The first step is to assign index to computable function.

Agenda

We shall study three fundamental theorems about indexing.

Synopsis

1. Gödel Index
2. S-m-n Theorem
3. Enumeration Theorem
4. Recursion Theorem

1. Gödel Index

Basic Idea

We see a number as an index for a problem/function if it is the Gödel number of a programme that solves/calculates the problem/function.

Definition

Suppose $a \in \omega$ and $n \geq 1$.

$$
\begin{aligned}
\phi_a^{(n)} &= \text{the } n \text{ ary function computed by } P_a \\
&= f_{P_n}^{(n)}, \\
W_a^{(n)} &= \text{the domain of } \phi_a^{(n)} = \{(x_1, \ldots, x_n) \mid P_a(x_1, \ldots, x_n) \downarrow\}, \\
E_a^{(n)} &= \text{the range of } \phi_a^{(n)}.
\end{aligned}
$$

The super script (n) is omitted when $n = 1$.

Example

Let $a = 4127$. Then $P_{4127} = S(2); T(2,1)$.

If the program is seen to calculate a unary function, then

$$\begin{aligned} \phi_{4127}(x) &= 1, \\ W_{4127} &= \omega, \\ E_{4127} &= \{1\}. \end{aligned}$$

If the program is seen to calculate an n-ary function, then

$$\begin{aligned} \phi^{(n)}_{4127}(x_1, \ldots, x_n) &= x_2 + 1, \\ W^n_{4127} &= \omega^n, \\ E^n_{4127} &= \omega^+. \end{aligned}$$

Gödel Index for Computable Function

Suppose f is an n-ary computable function.

A number a is an index for f if $f = \phi_a^{(n)}$.

Padding Lemma

Padding Lemma. Every computable function has infinite indices. Moreover for each x we can effectively find an infinite recursive set A_x of indices for ϕ_x.

Proof.

Systematically add useless instructions to P_x. □

Computable Functions are Enumerable

We may list for example all the elements of $\mathcal{C}$ as

$$\phi_0, \phi_1, \phi_2, \ldots.$$

Diagonal Method

Fact. There is a total unary function that is not computable.

Proof.

Suppose $\phi_0, \phi_1, \phi_2, \ldots$ is an enumeration of $\mathcal{C}$. Define

$$f(n) \quad = \quad \begin{cases} \phi_n(n) + 1, & \text{if } \phi_n(n) \text{ is defined,} \\ 0, & \text{if } \phi_n(n) \text{ is undefined.} \end{cases}$$

By Church-Turing Thesis the function $f(n)$ is not computable. □

2. S-m-n Theorem

How do different indexing systems relate?

S-m-n Theorem, the Unary Case

Given a binary function $f(x, y)$, we get a unary computable function $f(a, y)$ by fixing a value a for x.

Let e be an index for $f(a, y)$. Then

$$f(a, y) \simeq \phi_e(y).$$

S-m-n Theorem states that the index e can be computed from a.

S-m-n Theorem, the Unary Case

Fact. Suppose that $f(x, y)$ is a computable function. There is a primitive recursive function $k(x)$ such that

$$f(x, y) \simeq \phi_{k(x)}(y).$$

S-m-n Theorem, the Unary Case

Let F be a program that computes f. Consider the following

$$\begin{array}{l} T(1,2) \\ Z(1) \\ \left.\begin{array}{l} S(1) \\ \quad\vdots \\ S(1) \end{array}\right\} a \text{ times} \\ F \end{array}$$

The above program can be effectively constructed from a.

Let $k(a)$ be the Gödel number of the above program.
It can be effectively computed from the above program.

Example

1. Let $f(x, y) = y^x$. Then $\phi_{k(x)}(y) = y^x$. For each fixed n, $k(n)$ is an index for y^n.
2. Let $f(x, y) \simeq \begin{cases} y, & \text{if } y \text{ is a multiple of } x, \\ \uparrow, & \text{otherwise.} \end{cases}$.

   Then $\phi_{k(n)}(y)$ is defined if and only if y is a multiple of n.

S-m-n Theorem.

For m, n, there is an injective primitive recursive $(m+1)$-function $s_n^m(x, \widetilde{x})$ such that for all e the following holds:

$$\phi_e^{m+n}(\widetilde{x}, \widetilde{y}) \simeq \phi^n_{s_n^m(e,\widetilde{x})}(\widetilde{y}).$$

S-m-n Theorem is also called **Parameter Theorem**.

Proof of S-m-n Theorem

Proof. Given $e, x_1, \ldots, x_m$, we can effectively construct the following program and its index

$$\begin{array}{l} T(n, m+n) \\ \vdots \\ T(1, m+1) \\ Q(1, x_1) \\ \vdots \\ Q(m, x_m) \\ P_e \end{array}$$

where $Q(i, x)$ is the program $Z(i), \underbrace{S(i), \ldots, S(i)}_{x \text{ times}}$.

The injectivity is achieved by padding enough useless instructions.

3. Enumeration Theorem

What makes it possible that every C program can be executed in a computer?

General Remark

There are universal programs that embody all the programs.

A program is universal if upon receiving the Gödel number of a program it simulates the program indexed by the number.

Intuition

Consider the function $\psi(x, y)$ defined as follows

$$\psi(x, y) \simeq \phi_x(y).$$

In an obvious sense $\psi(x, \_)$ is a universal function for the unary functions

$$\phi_0, \phi_1, \phi_2, \phi_3, \ldots.$$

Universal Function

The universal function for n-ary computable functions is the $(n+1)$-ary function $\psi_U^{(n)}$ defined by

$$\psi_U^{(n)}(e, x_1, \ldots, x_n) \simeq \phi_e^{(n)}(x_1, \ldots, x_n).$$

We write ψ_U for $\psi_U^{(1)}$.

Question: Is $\psi_U^{(n)}$ computable?

Enumeration Theorem.

For each n, the universal function $\psi_U^{(n)}$ is computable.

Proof.

Given a number e, decode the number to get the program P_e; and then simulate the program P_e. If the simulation ever terminates, then return the number in R_1. By Church-Turing Thesis, $\psi_U^{(n)}$ is computable. □

Application: Undecidability

Proposition. The problem 'ϕ_x is total' is undecidable.

Proof.

If 'ϕ_x is total' were decidable, then by Church's Thesis

$$f(x) = \begin{cases} \psi_U(x, x) + 1, & \text{if } \phi_x \text{ is total,} \\ 0, & \text{if } \phi_x \text{ is not total.} \end{cases}$$

would be a total computable function that differs from every total computable function. □

Application: Effectiveness of Function Operation

Proposition. There is a total computable function $s(x,y)$ such that $\phi_{s(x,y)} = \phi_x\phi_y$ for all x,y.

Proof.

Let $f(x,y,z) \simeq \phi_x(z)\phi_y(z) \simeq \psi_U(x,z)\psi_U(y,z)$.

By S-m-n Theorem there is a total function $s(x,y)$ such that $\phi_{s(x,y)}(z) \simeq f(x,y,z)$. □

Application: Effectiveness of Set Operation

Proposition. There is a total computable function $s(x, y)$ such that $W_{s(x,y)} = W_x \cup W_y$.

Proof.

Let

$$f(x, y, z) = \begin{cases} 1, & \text{if } z \in W_x \text{ or } z \in W_y, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

By S-m-n Theorem there is a total function $s(x, y)$ such that $\phi_{s(x,y)}(z) \simeq f(x, y, z)$. Clearly $W_{s(x,y)} = W_x \cup W_y$. □

Application: Effectiveness of Recursion

Consider f defined by the following recursion

$$f(e_1, e_2, \widetilde{x}, 0) \simeq \phi_{e_1}^{(n)}(\widetilde{x}) \simeq \psi_U^{(n)}(e_1, \widetilde{x}),$$

and

$$\begin{array}{rcl} f(e_1, e_2, \widetilde{x}, y+1) & \simeq & \phi_{e_2}^{(n+2)}(\widetilde{x}, y, f(e_1, e_2, \widetilde{x}, y)) \\ & \simeq & \psi_U^{(n+2)}(e_2, \widetilde{x}, y, f(e_1, e_2, \widetilde{x}, y)). \end{array}$$

By S-m-n Theorem, there is a total computable function $r(e_1, e_2)$ such that

$$\phi_{r(e_1,e_2)}^{(n+1)}(\widetilde{x}, y) \simeq f(e_1, e_2, \widetilde{x}, y).$$

Application: Non-Primitive Recursive Total Function

Theorem. There is a total computable function that is not primitive recursive.

Proof.

1. The primitive recursive functions have a universal function.
2. Such a function cannot be primitive recursive by diagonalisation. □

4. Recursion Theorem

Recursion Theorem (Kleene, 1938).

Let f be a total unary computable function. Then there is a number n such that $\phi_{f(n)} = \phi_n$.

Proof.

By S-m-n Theorem there is an injective primitive recursive function $s(x)$ such that for all x

$$\phi_{s(x)}(y) \simeq \begin{cases} \phi_{\phi_x(x)}(y), & \text{if } \phi_x(x)\downarrow; \\ \uparrow, & \text{otherwise.} \end{cases} \quad (1)$$

Let v be such that $\phi_v = s; f$. Obviously ϕ_v is total and $\phi_v(v)\downarrow$. It follows from (1) that

$$\phi_{s(v)} = \phi_{\phi_v(v)} = \phi_{f(s(v))}.$$

We are done by letting n be $s(v)$. □

More about Recursion Theorem

Fact. If f is partial then $\phi_{f(n)} = \phi_n$ whenever $f(n)\downarrow$.

Fact. The fixpoint n can be computed from an index of f by an injective primitive recursive function.

Proof.
Let $v(z)$ be an injective primitive recursive function such that $\phi_{v(z)} \simeq s; \phi_z$. Then let $n(z) \simeq s(v(z))$. □

Fact. There is an infinite set of fixpoints for f.

Proof.
By Padding Lemma there is an infinite set of indices v such that $\phi_v \simeq s; f$. □

Fact. If f is a total computable function, there is a number n such that $W_{f(n)} = W_n$ and $E_{f(n)} = E_n$.

Self Referential Definition

Fact. Let $f(x,y)$ be a computable function. Then there is an index e such that

$$\phi_e(y) \simeq f(e,y).$$

Proof.

By S-m-n Theorem there is a total computable function $s(x)$ such that $\phi_{s(x)}(y) \simeq f(x,y)$. We are done by applying Recursion Theorem. □

Self Referential Definition

The previous corollary makes it meaningful to define a computable function $\phi_e(y)$ by a computable function $f(e, y)$.

Self Referential Definition

There is a number n such that $\phi_n(x) = x^n$.

There is a number n such that $W_n = \{n\}$. This number is obtained by applying the above corollary to the function

$$f(x,y) = \begin{cases} 0, & \text{if } x = y, \\ \uparrow, & \text{otherwise.} \end{cases}$$

Self Printing Program

Fact. There is a program P such that for all x, $P(x)\downarrow\gamma(P)$.

Proof.

It says that there is a number n such that

$$\phi_n(x) = n$$

for all x. Simply apply one of the corollaries to $f(z,x) = z$. □

Applying Recursion Theorem

Theorem. Suppose that f is a total increasing function such that

- if $m \neq n$ then $\phi_{f(m)} \neq \phi_{f(n)}$,
- $f(n)$ is the least index of the function $\phi_{f(n)}$.

Then f is not computable.

Proof.

Suppose f satisfies the conditions of the theorem.

By the first condition $f(n) > n$ if n is large enough.

By the second condition $\phi_{f(n)} \neq \phi_n$ for all large enough n.

This contradicts to Recursion Theorem. □

Diagonalisation Implicit in Recursion Theorem

We may think of ϕ_x as providing an effective enumeration of

$$\phi_{\phi_x(0)}, \phi_{\phi_x(1)}, \phi_{\phi_x(2)}, \ldots, \phi_{\phi_x(i)}, \ldots.$$

The diagonal function $\phi_x(x)$ enumerates

$$\phi_{\phi_0(0)}, \phi_{\phi_1(1)}, \phi_{\phi_2(2)}, \ldots.$$

Let f be total computable. There is some total computable $s(x)$, due to S-m-n Theorem, and some index m for $s(x)$ such that

$$\phi_{\phi_m(x)}(y) \simeq \phi_{s(x)}(y) \simeq \phi_{f(\phi_x(x))}(y).$$

Both ϕ_m and $f(\phi_x(x))$ enumerate the following

$$\phi_{\phi_m(0)}, \phi_{\phi_m(1)}, \phi_{\phi_m(2)}, \ldots, \phi_{\phi_m(i)}, \ldots.$$

Diagonalisation Implicit in Recursion Theorem

$$
\begin{array}{llllll}
\phi_{\phi_0(0)}, & \phi_{\phi_0(1)}, & \phi_{\phi_0(2)}, & \ldots, & \phi_{\phi_0(i)}, & \ldots \\
\phi_{\phi_1(0)}, & \phi_{\phi_1(1)}, & \phi_{\phi_1(2)}, & \ldots, & \phi_{\phi_1(i)}, & \ldots \\
\phi_{\phi_2(0)}, & \phi_{\phi_2(1)}, & \phi_{\phi_2(2)}, & \ldots, & \phi_{\phi_2(i)}, & \ldots \\
\vdots & \vdots & \vdots & & \vdots & \\
\phi_{\phi_m(0)}, & \phi_{\phi_m(1)}, & \phi_{\phi_m(2)}, & \ldots, & \phi_{\phi_m(m)}, & \ldots \\
\vdots & \vdots & \vdots & & \vdots &
\end{array}
$$

The diagonal intersects the m-row at $\phi_{\phi_m(m)} = \phi_{f(\phi_m(m))}$.

It is important that $\phi_m(m)$ must be defined.

Diagonalisation offers an intuition about Recursion Theorem.

It also explains the power of Recursion Theorem.

Generalized Recursion Theorem. Suppose $f(x, z)$ is a total computable function. There is an injective primitive recursive function $n(z)$ such that $\phi_{f(n(z),z)} = \phi_{n(z)}$ for all z.

Proof.

By S-m-n Theorem there is an injective primitive recursive function $s(x, z)$ such that

$$\phi_{f(\phi_x(x),z)} = \phi_{s(x,z)}.$$

By the same theorem there is an injective primitive recursive function $m(z)$ such that $s(x, z) = \phi_{m(z)}(x)$. So

$$\phi_{f(\phi_x(x),z)} = \phi_{\phi_{m(z)}(x)}.$$

We are done by letting $x = m(z)$ and $n(z) = \phi_{m(z)}(m(z))$.

It is clear that n is an injective primitive recursive function. □

Basic recursion theory can be developed from

S-m-n Theorem and Enumeration Theorem.

Advanced recursion theory makes frequent use of

Recursion Theorem.