Gödel Number*

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

General Remark

The set of the programs are countable.

More importantly, every program can be coded up effectively by a number in such a way that a unique program can be recovered from the number.

Denumerability and Enumerability

A set X is denumerable if there is a bijection $f : X \to \mathbb{N}$.

An enumeration of a set X is a surjection $g : \mathbb{N} \to X$;
this is often represented by writing $\{x_0, x_1, x_2, \ldots\}$.
It is an enumeration *without repetitions* if g is injective.

Let X be a set of "finite objects".
Then X is effectively denumerable if there is a bijection $f : X \to \mathbb{N}$ such that both f and f^{-1} are effectively computable functions.

Effective Denumerability

Fact. $\mathbb{N} \times \mathbb{N}$ is effectively denumerable.

Proof. A bijection $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined by

$$\begin{aligned}\pi(m,n) &\stackrel{\text{def}}{=} 2^m(2n+1)-1,\\ \pi^{-1}(l) &\stackrel{\text{def}}{=} (\pi_1(l), \pi_2(l)),\end{aligned}$$

where

$$\begin{aligned}\pi_1(x) &\stackrel{\text{def}}{=} (x+1)_1,\\ \pi_2(x) &\stackrel{\text{def}}{=} ((x+1)/2^{\pi_1(x)}-1)/2.\end{aligned}$$

Effective Denumerability

Fact. $\mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+$ is effectively denumerable.

Proof. A bijection $\zeta : \mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}$ is defined by

$$\begin{aligned}\zeta(m,n,q) &\stackrel{\text{def}}{=} \pi(\pi(m-1,n-1),q-1),\\ \zeta^{-1}(l) &\stackrel{\text{def}}{=} (\pi_1(\pi_1(l))+1, \pi_2(\pi_1(l))+1, \pi_2(l)+1).\end{aligned}$$

Effective Denumerability

Fact. $\bigcup_{k>0} \mathbb{N}^k$ is effectively denumerable.

Proof. A bijection $\tau : \bigcup_{k>0} \mathbb{N}^k \to \mathbb{N}$ is defined by

$$\begin{aligned}\tau(a_1,\ldots,a_k) \stackrel{\text{def}}{=}\ & 2^{a_1} + 2^{a_1+a_2+1} + 2^{a_1+a_2+a_3+2} + \ldots\\ & + 2^{a_1+a_2+a_3+\ldots,a_k+k-1} - 1.\end{aligned}$$

Now given x we can find a unique expression of the form

$$2^{b_1} + 2^{b_2} + 2^{b_3} + \ldots + 2^{b_k}$$

that equals to $x+1$. It is then clear how to define $\tau^{-1}(x)$.

Gödel Encoding

Let $\mathscr{I}$ be the set of all instructions.

Let $\mathscr{P}$ be the set of all programs.

The objects in $\mathscr{I}$, and $\mathscr{P}$ as well, are 'finite objects'.
They must be effectively denumerable.

Gödel Encoding

Theorem. $\mathscr{I}$ is effectively denumerable.

Proof. The bijection $\beta : \mathscr{I} \to \mathbb{N}$ is defined as follows:

$$\begin{aligned}
\beta(Z(n)) &= 4(n-1),\\
\beta(S(n)) &= 4(n-1)+1,\\
\beta(T(m,n)) &= 4\pi(m-1,n-1)+2,\\
\beta(J(m,n,q)) &= 4\zeta(m,n,q)+3.
\end{aligned}$$

The converse β^{-1} is easy.

Gödel Encoding

Theorem. $\mathscr{P}$ is effectively denumerable.

Proof. The bijection $\gamma : \mathscr{P} \to \mathbb{N}$ is defined as follows:

$$\gamma(P) = \tau(\beta(I_1), \ldots, \beta(I_s)),$$

assuming $P = I_1, \ldots, I_s$.

The converse γ^{-1} is obvious.

Gödel Encoding

The number $\gamma(P)$ is called the Gödel number of P.

$$\begin{aligned}
P_n &= \text{the program with Gdel number } n\\
&= \gamma^{-1}(n)
\end{aligned}$$

Gödel Encoding

Let P be the program $T(1,3), S(4), Z(6)$.

$\beta(T(1,3)) = 18$, $\beta(S(4)) = 13$, $\beta(Z(6)) = 20$.

$\gamma(P) = 2^{18} + 2^{32} + 2^{53} - 1$.

Gödel Encoding

Consider P_{4127}.

$4127 = 2^5 + 2^{12} - 1$.

$\beta(I_1) = 4 + 1$, $\beta(I_2) = 4\pi(1, 0) + 2$.

So P_{4127} is $S(2); T(2, 1)$.

Gödel Encoding

We shall fix this particular coding function γ throughout.

Numbering Computable Functions

Suppose $a \in \mathbb{N}$ and $n \geq 1$.

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

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

Numbering Computable Functions

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

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

$$
\begin{aligned}
\phi_{4127}^{(n)}(x_1, \ldots, x_n) &= x_2 + 1, \\
W_{4127}^n &= \mathbb{N}^n, \\
E_{4127}^n &= \mathbb{N}^+.
\end{aligned}
$$

Numbering Computable Functions

Suppose $f = \phi_a$. Then a is an index for f.

There are an infinite number of indexes for f.

Numbering Computable Functions

Theorem. $\mathscr{C}_n$ is denumerable.

Proof

We use the enumeration $\phi_0^{(n)}, \phi_1^{(n)}, \phi_2^{(n)}, \cdots$ (with repetitions) to construct one without repetitions.

Let $\begin{cases} f(0) = 0; \\ f(m+1) = \mu z(\phi_z^{(n)} \neq \phi_{f(0)}^{(n)}, \cdots, \phi_{f(m)}^{(n)}), \end{cases}$

Then $\phi_{f(0)}^{(n)}, \phi_{f(1)}^{(n)}, \phi_{f(2)}^{(n)}, \cdots$ is an enumeration of $\mathscr{C}_n$ without repetitions.

Corollary

Corollary: $\mathscr{C}$ is denumerable.

Proof: Since $\mathscr{C} = \bigcup_{n\geq 1} \mathscr{C}_n$, the corollary follows from the fact that a denumerable union of denumerable sets is denumerable.

Explicitly, for each n let f_n be the function to give an enumeration of $\mathscr{C}_n$ without repetitions. Let π be the bijection $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Define $\theta : \mathscr{C} \to \mathbb{N}$ by

$$\theta\left(\phi_{f_n(m)}^{(n)}\right) = \pi(m, n-1),$$

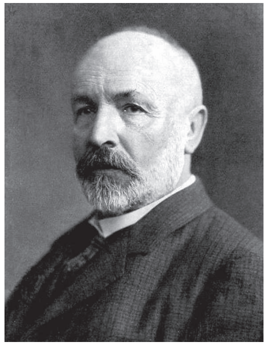
then θ is a bijection.

Cantor's Diagonal Argument

In set theory, Cantor's diagonal argument, also called the diagonalisation argument, the diagonal slash argument or the diagonal method, was published in 1891 by Georg Cantor.

It was proposed as a mathematical proof for uncountable sets.

It demonstrates a powerful and general technique that has been used in a wide range of proofs.



Georg Cantor
1845-1918

The Diagonal Method

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

Proof. Suppose $\phi_0, \phi_1, \phi_2, \ldots$ is an enumeration of $\mathscr{C}_1$. Define

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

The function $f(n)$ is not computable.

Example of uncomputable function

Consider again the construction of f to construct a total uncomputable function. Complete details of the functions $\phi_0, \phi_1, \cdots$ can be represented by the following infinite table:

	0	1	2	3	4
ϕ_0	$\phi_0(0)$	$\phi_0(1)$	$\phi_0(2)$	$\phi_0(3)$	...
ϕ_1	$\phi_1(0)$	$\phi_1(1)$	$\phi_1(2)$	$\phi_1(3)$	...
ϕ_2	$\phi_2(0)$	$\phi_2(1)$	$\phi_2(2)$	$\phi_2(3)$	...
ϕ_3	$\phi_3(0)$	$\phi_3(1)$	$\phi_3(2)$	$\phi_3(3)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Diagonal Method

We suppose that in this table the word 'undefined' is written whenever $\phi_n(m)$ is not defined.

The function f was constructed by taking the diagonal entries on the table $\phi_0(0), \phi_1(1), \phi_2(2), \cdots$ and systematically changing them, obtaining $f(0), f(1), \cdots$ such that $f(n)$ differs from $\phi_n(n)$, for each n.

Note that there was considerable freedom in choosing the value of $f(n)$ (just differ from $\phi_n(n)$). Thus

$$g(n) = \begin{cases} \phi_n(n) + 27^n & \text{if } \phi_n(n) \text{ is defined,} \\ n^2 & \text{if } \phi_n(n) \text{ is undefined,} \end{cases}$$

is another non-computable total function.

Cantor's Diagonal Method

Suppose that $\chi_0, \chi_1, \cdots$ is an enumeration of objects of a certain kind (functions or sets of natural numbers), then we can construct an object χ of the same kind that is different from every χ_n, using the following motto:

'Make χ and χ_n differ at n.'

The interpretation of the phrase *differ at n* depends on the kind of object involved.

Diagonal Construction on Sets

Suppose that $A_0, A_1, \cdots$ is an enumeration of subsets of $\mathbb{N}$. We can define a new set B using the diagonal motto, by

$$n \in B \text{ if and only if } n \notin A_n.$$

Clearly, for each n, $B \neq A_n$.

Note that $B \subseteq 2^{\mathbb{N}}$, so $2^{\mathbb{N}}$ is not a denumerable set.

The s-m-n Theorem

Given a computable binary function $f(x, y)$ (not necessarily total), we get a unary computable function $f(a, y)$ by fixing a value a for x.

We can use a unary computable function $g_a(y) \simeq f(a, y)$ to represent $f(a, y)$, then there is an index e for $f(a, y)$.

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

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

The s-m-n Theorem, simple form

Theorem. Suppose that $f(x, y)$ is a computable function. There is a total computable function $k(x)$ such that

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

The s-m-n Theorem

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

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

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.

Notation

The s-m-n theorem is also called Parametrization Theorem because it shows that an index for a computable function (such as g_a) can be found effectively from a parameter (such as a) on which it effectively depends.

Examples

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.

Let $f(x,y) = \begin{cases} y, & \text{if } y \text{ is a multiple of } x, \\ \text{undefined}, & \text{otherwise}. \end{cases}$.
Then $\phi_{k(n)}(y)$ is defined if and only if y is a multiple of n.

The s-m-n Theorem

Theorem. For m, n, there is a total computable $(m+1)$-function $s_n^m(\_, \mathbf{x})$ such that for all e the following holds:

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

The s-m-n Theorem

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

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

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