02. Some set-theoretical notions

2016年9月21日 0:15

Last time

Computability Theory < (relative) computability / Randomness > properties of sets of natural numbers

Coding: mapping "objects" to natural numbers

so that we can apply mathematics in the real world.

"Randomness" is a crucial concept

Some set-theoretical notions

$0 = \emptyset$, $1 = \{0\}$, $n+1 = \{0, \cdots, n\}$

$\mathbb{N} = \omega = \{0, 1, 2, \cdots\}$ the set of all natural numbers

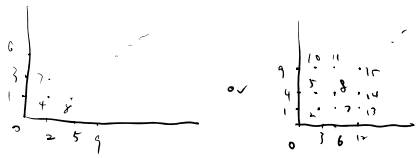
Ordered pair: (n, m) points on the Cartesian Product $\mathbb{N} \times \mathbb{N}$

R is Relation: if $R \subseteq \mathbb{N} \times \mathbb{N}$, function: relation with one value property i.e.

$f \subseteq \omega \times \omega$ is a function iff for any $(n, m_1) \in f$, $(n, m_2) \in f$ we have $m_1 = m_2$

$\mathrm{dom} R = \{n \mid \exists m\ (n, m) \in R\}$

Code $\mathbb{N} \times \mathbb{N}$



Fact $|\mathbb{N}| = |\mathbb{N} \times \mathbb{N}| = |\mathbb{Z}| = |\mathbb{Q}|$ [Hint: integers can be represented by $(i, n, m) \in 2 \times \omega \times \omega$; rationals can be represented by $(n, m) \in \mathbb{Z} \times \mathbb{N}^+$]

Define $A^B = \{f \mid f : B \to A\}$

$= \{f \mid f$ is a function with $\mathrm{dom} f = B$ and $\mathrm{ran} f \subseteq A\}$

Example 2^n is the set of all finite 01-sequence of length n

2^ω is the set of all infinite 01-sequence

if $f \in 2^n$, then $\mathrm{dom} f = n = \{0, \cdots n-1\}$

$f = \{(0, i_0), \cdots (n-1, i_{n-1})\}$, we also write f as a sequence $\langle i_0, \cdots i_{n-1} \rangle$, And the length of f (as a sequence) $= |f| = \mathrm{dom} f$ as a set as a function

Fact $|\{A \mid A \subseteq \omega\}| = |2^\omega| = |\{r \in \mathbb{R} \mid 0 \le r \le 1\}|$

[$\chi_A(n) = \begin{cases} 0 & \text{if } n \in A \\ 1 & \text{else} \end{cases}$]

Notation For $A \subseteq \omega$, we write $A(n) = \chi_A(n) \in 2 = \{0, 1\}$

$r = 0.111\cdots1\cdots \to 111\cdots$

Therefore sets of natural numbers, (infinite) 01 sequences, functions from ω to 2, reals (in $[0, 1]$) are essentially the same

Fact $|\omega| \lneq |2^\omega|$

Proof diagonal argument:

	0	1	2	⋯
f_0	$f_0(0)$	$f_0(1)$		
f_1	$f_1(0)$	$f_1(1)$		
⋮				

Consider $g : \mathbb{N} \to 2$ s.t. $g(i) = 1 - f_i(i)$

f_0 $f_0(0)$ $f_0(1)$ ...

f_1 $f_1(0)$ $f_1(1)$

f_2 ⋮ ⋮

⋮

$g(i) = 1 - f_i(i)$

Defn $2^{<\omega} = \bigcup_{n<\omega} 2^n$, which is the set of all finite sequences

Fact $|\mathbb{N}| = |2^{<\omega}|$

Defn $\sigma <_L \tau$ iff $|\sigma| < |\tau|$ or

$|\sigma| = |\tau|$ and $\sigma(n)=0$, $\tau(n)=1$ for the least n s.t. $\sigma(n) = \tau(n)$

Claim $(2^{<\omega}, <_L) \simeq (\omega, <)$

subclaim1 $(2^{<\omega}, <_L)$ is a well-ordering ie. it is a total order and every subset $A \subseteq 2^{<\omega}$ has a $<_L$ least element

subclaim2 every proper initial segement of $(2^{<\omega}, <_L)$ is finite

Lemma For well-ordering $(W_1, <_1)$, $(W_2, <_2)$ either $W_1 \simeq W_2 \restriction u$ for some $u \in W_2$

or $W_2 \simeq W_1 \restriction v$ for some $v \in W_1$

or $W_1 \simeq W_2$

Therefore $|\omega| = |2^{<\omega}|$

Exe $|\mathbb{N}| = |\omega^{<\omega}|$

$|\omega^\omega| = |2^\omega|$

03. Computabiliy

2016年9月22日 15:26

Computability theory

Computer Program (Turing Machine)

a string of ASCII, which essetially a finite 0,1-string $\subseteq 2^{<\omega}$

Therefore the set of all Program / Machine can be enumerated with natural numbers

$\Phi_0, \Phi_1, \Phi_2 \cdots$

Note, there can be many different enmerations (compilers), we just fix one

Partial Functions and computability

Normaly, when we write $f: \omega \to A$, we mean f is a (total) function on ω

i.e. $\mathrm{dom} f = \omega$

When we say $f: \omega \to A$ is a partial function, we mean $\mathrm{dom} f \subseteq \omega$

(not necessarily equal to)

Every program Φ_e corresponds to a partial function $f: \omega \to \omega$

we can assume to inputs and possible outputs of a program are natural numbers

(if the input is an pair (n,m), or even $(n, \cdots n_k)$, take their code. Note $|2^{<\omega}| = |\omega|$)

We write $\Phi_e(n)\downarrow$ if the program (compiled from) e with input n has an output (halts)

We write $\Phi_e(n)\downarrow = m$, or $\Phi_e(n) = m$ if it halts and output m.

We write $\Phi_e(n)\uparrow$ if $\Phi_e(n)$ does not halt

We write $\Phi_{e,s}(n)\downarrow$ / $\Phi_{e,s}(n)\downarrow = m$ / $\Phi_{e,s}(n)\uparrow$ if $\Phi_e(n)$ halts (with output m) / does not halt after s steps

Given a program Φ_e, define partial function $f: \mathbb{N} \to \mathbb{N}$ s.t.

$$f(n) = \begin{cases} \Phi_e(n) & \text{if } \Phi_e(n)\downarrow \\ \text{undefined} & \text{o.w.} \end{cases}$$

Note $\mathrm{dom} f = \{ n \in \omega \mid \Phi_e(n)\downarrow \}$. In this case, we also write Φ_e to indicate the corresponding partial function

we define W_e to be the set

We say a partial function $f: \mathbb{N} \to \mathbb{N}$ is computable, if $f = \Phi_e$ for some $e \in \omega$

Note, for any computable partial function f, there are infinitely many program computes it, i.e.

i.e. $|\{ d \mid \Phi_d = \Phi_e \}| = \omega$

For a set $A \subseteq \mathbb{N}$, we say A is computible, if the (total) function

$X_A = \bar{\Phi}_e$ for some e

Intuitively, a set A is computable, iff there is a effective method to decide whether a given number n belongs to A

The enumeration of all programs, universal program, and unsolvability of halting problem

Fix an enumeration of all programs $\bar{\Phi}_0, \bar{\Phi}_1, \cdots \bar{\Phi}_e \cdots$

We can program a universal machine U as follow:

on input (e,n), U compiles e to get the program $\bar{\Phi}_e$, and run $\bar{\Phi}_e(n)$

Proposition There is no program $\bar{I}$ which (can) given x, y can decide whether $\bar{\Phi}_x(y)\downarrow$ (or equivalently whether $U(x,y)\downarrow$)

Proof (diagonal argument). If $\bar{\Phi}$ exists, consider the following program $\bar{D}$:

on input x, it check if $\bar{\Phi}_x(x)\downarrow$ if no, return 0, if yes return $(\bar{\Phi}_x(x)+1)$

Assume $\bar{\Psi} = \bar{\Phi}_e$, then $\bar{\Phi}_e(e)\uparrow \Rightarrow \bar{\Phi}_e(e)\downarrow = 0$

$\bar{\Phi}_e(e)\downarrow \Rightarrow \bar{\Phi}_e(e) = \bar{\Phi}_e(e)+1$ ↯

	0	1	2	⋯	e	⋯
$\bar{\Phi}_0$	$\bar{\Phi}_0(0)+1$					
$\bar{\Phi}_1$		0				
⋮			⋱			
$\bar{\Phi}_e$					$\bar{\Phi}_e(e) = ?$	

□

Corollary $\{e \mid \bar{\Phi}_e(e)\downarrow\} = 0'$ is not computable

---

Lemma (s-m-n theorem / Parameter Lemma)

Let $\bar{\varphi} : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ be a partial computable function.

Then there is a (total) computable function $s : \mathbb{N}\to\mathbb{N}$ s.t. for all $x, y \in \omega$

$$\bar{\Phi}_{s(x)}(y) = \bar{\varphi}(x,y)$$

Proof s is a function of "rewriting programs" given program $\bar{\Phi}(x,y)$ (with two inputs) it rewrite the program to be $\bar{\Phi}_{s(x)}(y)$ with x predefined

Proposition $Tot = \{e \mid \bar{\Phi}_e$ is a total function$\}$ is not computable

Proof If Tot is computable we show $0' = \{e \mid \bar{\Phi}_e(e)\downarrow\}$ is also computable

Consider the program $\theta(e,x) = \bar{\Phi}_e(e)$. By s-n-m, there is a computable $s : \mathbb{N}\to\mathbb{N}$ s.t.

$$\bar{\Phi}_{s(e)}(x) = \theta(e,x) = \bar{\Phi}_e(e)$$

Then $s(e) \in Tot \iff e \in 0'$ □

Exe $\{e \mid \mathrm{dom}\,\Phi_e \neq \emptyset\}$, $\{e \mid 1 \in \mathrm{dom}\,\Phi\}$ are not computable

Define We say $f_0, f_1, \cdots$ are uniformly (partial) computable functions if there is a (partial) computable function $f: \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ s.t. $f(n,x) = f_n(x)$ for all $n, x \in \mathbb{N}$

Fact $f_0, f_1, \cdots$ are uniformly computible iff there is a total computible function $g: \mathbb{N} \to \mathbb{N}$ s.t. $f_n = \Phi_{g(n)}$

Prof ($\Rightarrow$) By s-n-m theorem

($\Leftarrow$) Define $\Phi(n,x)$: firstly, compile $g(n)$, then run $\Phi_{g(n)}(x)$ ☐