I216 計算量の理論と離散数学

上原隆平、面和成

I216 Computational Complexity and Discrete Mathematics

by
Prof. Ryuhei Uehara
and
Prof. Kazumasa Omote

4. 計算不能性と対角線論法

4. 計算不能な問題

以下の問題を解くチューリングマシンは存在しない：

停止性判定問題HALT (停止するかどうかを決定する問題)

入力：チューリングマシンTと
それへの入力xを符号化した文字列<T,x>

出力: T に入力 x を与えると，停止するか？

Yes: $T(x)$ は(有限時間内に)停止する

No: 停止しない(無限ループ)

正確に言えば，停止性判定問題を解くチューリングマシン U' は存在しない.

…証明は「対角線論法」を用いて行う

4. Undecidability and Diagonalization

4. Undecidable problem

The following problem cannot be solved by any Turing machine:

The problem HALT (Problem of deciding halting)

input : a code $<T,x>$ of Turing machine T and an input x

output: T will terminates for the input x?

Yes: if $T(x)$ terminates

No: otherwise.

Precisely, we can show that there is no Turing machine U' that computes the halting problem

…Proof is done by “diagonalization” essentially…

4.計算不能性と対角線論法

4．1. 計算不能性の*単純*な証明

[単純(?)な証明]

背理法による: 停止性判定問題HALTを解くチューリングマシン*U*が存在したと仮定

→ *U* は他のチューリングマシンで模倣可能

→ 次のチューリングマシン *X* を構成することができる

```
prog X(input w: Σ*): Σ*;
    label LOOP;
    begin
        if U (w, w) then LOOP: goto LOOP
                    else halt(0) end-if
    end.
```

X(*x*) を実行すると何が起こるか??

プログラム *X* 自身も文字列 *x* で表現できる

- 一つ目の *w* はチューリングマシン*W*を表現する文字列
- 二つ目の *w* はそのマシンへの入力文字列

プログラム *X*(*w*) は…
- *W*(*w*) が停止しないなら停止する
- *W*(*w*) が停止するなら無限ループ

4. Undecidability and Diagonalization

4. 1. A *simple* proof of undecidability

[A simple(?) proof]

By contradiction: Suppose that there is a Turing machine *U* that solves HALT.

→ *U* can be simulated by the other Turing machines.

→ We can design/construct the following Turing machine *X*:

```
prog X(input w: Σ*): Σ*;
    label LOOP;
    begin
        if U (w, w) then LOOP: goto LOOP
                    else halt(0) end-if
    end.
```

What happens on *X*(*x*)??

Program *X* can be encoded by a string *x*

- The first *w* is the code of a Turing machine *W*
- The second *w* is an input string to the machine.

Program *X(w)*

- terminates if *W(w)* does not terminate
- never stop if *W(w)* terminates

4.計算不能性と対角線論法

4. 1.計算不能性の*単純*な証明

[単純な証明]

背理法による: 停止性判定問題HALTを解くチューリングマシン*U*が存在したと仮定

→ *U* は他のチューリングマシンで模倣可能

→ 次のチューリングマシン *X* を構成することができる

プログラム *X(w)* は…

- *W(w)* が停止しないなら停止する
- *W(w)* が停止するなら無限ループ

X(x) を実行すると何が起こるか??

- 結果は2通り; 停止/無限ループ

ケース1: *X(x)* が停止すると仮定

プログラムの構成上、*X(x)* が停止しないときに実行されるはず

→ 仮定に矛盾*!*

論理的には正しそうだが…??
背後に*対角線論法*が隠れている.

ケース2: *X(x)* が停止しないと仮定

プログラムの構成上、*X(x)* が停止しないときに実行されるはず

→ 仮定に矛盾*!*

4. Undecidability and Diagonalization

4. 1. A *simple* proof of undecidability

[A simple proof]

By contradiction: Suppose that there is a Turing machine *U* that solves HALT.

→ *U* can be simulated by the other Turing machines.

→ We can design/construct the following Turing machine *X*:

Program *X(w)*
- terminates if *W(w)* does not terminate
- never stop if *W(w)* terminates

What happens on *X*(*x*)??
- Two choices; terminate/loop

Case 1: Assume *X(x)* terminates.

By the design of the program, *X(x)* does not terminate.

→ *It contradicts the assumption!*

Case 2: Assume *X(x)* does not terminate.

By the design of the program, *X(x)* does terminate.

→ *It contradicts the assumption!*

Logically, it may be true, but…??
Diagonalization is hidden here.

4.計算不能性と対角線論法

4. 2. 対角線論法

「対角線論法」はゲオルク・カントールが1873年に考案.
無限集合の大きさを測るという問題に取り組むためのもの

定義:
無限集合の「大きさ」のことを「濃度(cardinality)」と呼ぶ.

ごく自然(?)な疑問:
どんな無限集合も同じ「濃度」を持つのか？
どうやって大きさを比較したらよいのか？
… 1対1対応が見つかったらそれらは同じ濃度とする!

例1. 以下の集合たちはどれも「同じ濃度」である:
自然数 (0,1,2,…), 整数 (…, -2,-1,0,1,2,…),
偶数 (0,2,4,…), 素数 (2,3,5,7,11,13,…),
有理数, チューリングマシン (= 計算可能な関数), …

4. Undecidability and Diagonalization

4. 2. Diagonalization

"Diagonalization" was introduced by Georg Cantor in 1873.
He concerned with the problem of measuring the size of *infinite* sets.

Definition:
The "size" of an infinite set is called "cardinality" of the set.

Natural(?) question:
Any pair of infinite sets have the same "cardinality"?
How can we compare them?
… design a one-to-one mapping!

Ex. 1. The following sets have the *same cardinality*:
Natural numbers (0,1,2,…), integers (…, -2,-1,0,1,2,…),
even numbers (0,2,4,…), primes (2,3,5,7,11,13,…),
rational numbers, Turing machines (= computable functions), …

4.計算不能性と対角線論法

4. 2.対角線論法

定義:
集合が有限であるか、自然数と同じ濃度を持つとき、これを「可算」集合という.
(別の言い方をすれば、「一つ目」「二つ目」と数え上げられる集合が可算集合)

例1.1.
偶数は右の対応付けがあるので
可算集合:

i 番目の偶数を 2*i* とする

0	1	2	3	4
0	2	4	6	8

例1.2.
素数は右の対応付けがあるので
可算集合:

*i*番目の素数

0	1	2	3	4
2	3	5	7	11

観測:
可算集合の部分集合は可算集合

4. Undecidability and Diagonalization

4. 2. Diagonalization

Definition:

A set is *countable* if it is finite or it has the same cardinality of natural numbers.
(In other words, countable set can be enumerated as "1st," "2nd," …

Ex. 1.1.

Even numbers are countable by the one-to-one mapping:

The *i*th even number is 2*i*

0	1	2	3	4
0	2	4	6	8

Ex. 1.2.

Primes are countable by the one-to-one mapping:

The *i*th prime

0	1	2	3	4
2	3	5	7	11

Observation:
Any subset of a countable set is also countable

4.計算不能性と対角線論法

演習問題2: ここで通常の辞書式
順序を使わないのは，なぜか？

4. 2.対角線論法

定義:
集合が有限であるか、自然数と同じ濃度を持つとき、これを「可算」集合という.

例 1.3’.
0/1文字列の集合は長さ優先辞書式順序により可算集合:

ε, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, …

観測:
可算集合の部分集合
は可算集合

例 1.3.
チューリングマシン（で計算できる関数）は、
各マシンが2進文字列で符号化できることから可算集合
(別の言い方をすれば T_0, T_1, T_2,… と列挙できる)

自然な疑問: 可算でない集合なんて存在するのだろうか？

4. Undecidability and Diagonalization

Exercise 2: Why we do not use the ordinary lexicographical ordering?

4. 2. Diagonalization

Definition:

A set is *countable* if it is finite or it has the same cardinality of natural numbers.

Observation:
Any subset of a countable set is also countable

Ex. 1.3'.

The set of 0/1 strings is countable by the lex. ordering:

ε, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, …

Ex. 1.3.

Turing machines (and corresponding functions) are countable because each machine can be represented by a binary string. (In other words, they can be enumerated as T_0, T_1, T_2,…)

Natural question: Is there any uncountable set??

4.計算不能性と対角線論法

4. 2.対角線論法

定理:
実数の集合 $\mathcal{R}$ は非可算集合.

[対角線論法による証明]
集合 $\mathcal{R}$ が可算だったと仮定する; 以下のように列挙可能 $\mathcal{R} = \{ R_0, R_1, R_2, R_3, \ldots \}$

各 R_i は十進表記で $R_i = \ldots r_{i,4}' r_{i,3}' r_{i,2}' r_{i,1}' r_{i,0} . r_{i,1} r_{i,2} r_{i,3} r_{i,4} \ldots$ と書ける.

数 $X = 0. x_1 x_2 x_3 \ldots$ の各桁を次で定義

$$\begin{cases} x_i = 3 \text{ if } r_{i,i} = 1,2,4,5,6,7,8,9, \text{ or } 0 \\ x_i = 1 \text{ if } r_{i,i} = 3 \end{cases}$$

すると X は実数なので、ある i に対して $X=R_i$ と書けるはず.

ところがこのとき x_i の値は… 3? あるいは 1?… 決定できない。
これは矛盾!
したがって $\mathcal{R}$ は可算ではない!!

例.
$R_0 = 123.\underline{4}56\ldots$
$R_1 = 0.1\underline{3}1313\ldots$
$R_2 = 555.55\underline{5}5555\ldots$
$R_3 = 3.141\underline{5}92\ldots$
…

$X = 0.3133\ldots$

4. Undecidability and Diagonalization

4. 2. Diagonalization

Theorem:
The set $\mathcal{R}$ of real numbers is *not* countable.

[Proof by *diagonalization*]
Assume that $\mathcal{R}$ is countable; i.e., they are enumerated as $\mathcal{R}= \{ R_0, R_1, R_2, R_3, \ldots \}$

Each R_i is in the form of $R_i = \ldots r_{i,4}' \, r_{i,3}' \, r_{i,2}' \, r_{i,1}' \, r_{i,0} \,.\, r_{i,1} \, r_{i,2} \, r_{i,3} \, r_{i,4} \ldots$ in decimal.

We define a number $X = 0.\, x_1\, x_2\, x_3 \ldots$ by

$$\begin{cases} x_i = 3 \text{ if } r_{i,i} = 1,2,4,5,6,7,8,9, \text{ or } 0 \\ x_i = 1 \text{ if } r_{i,i} = 3 \end{cases}$$

Then X is a real number, so it will appear as $X=R_i$ for some i.

But x_i is… 3? or 1?... we cannot decide it,
which is a contradiction!
Therefore $\mathcal{R}$ is not countable!!

Ex.
$R_0 = 123.\underline{4}56\ldots$
$R_1 = 0.1\underline{3}1313\ldots$
$R_2 = 555.55\underline{5}5555\ldots$
$R_3 = 3.141\underline{5}92\ldots$
…

$X = 0.\,3133\ldots$

4.計算不能性と対角線論法

4. 3.対角線論法による計算不能性の証明

[定理] 停止性判定問題HALTは決定不能である.

[対角線論法による証明]

計算可能な(1入力)関数すべてからなる集合を $\mathcal{F}$ とする.
集合 $\mathcal{F}$ の各要素は一つのチューリングマシンに対応し、
それはΣ^*の2進文字列で表現される.
これらの2進文字列は辞書式順序で

$$b_1, b_2, \dots , b_k \ \dots$$

と列挙できる.
したがって$\mathcal{F}$のすべての関数は次のように列挙できる:

$$f_1, f_2, \dots , f_k, \dots$$

簡単にいえば $\mathcal{F}$ は可算!

4. Undecidability and Diagonalization

4. 3. Proof of undecidability via Diagonalization

[Theorem] The problem HALT is undecidable.

[Proof by diagonalization]

Let $\mathcal{F}$ be a set of all computable functions (with one argument) .

Each element in $\mathcal{F}$ corresponds to a Turing machine, that can be represented in a binary string in Σ^*.

Thus we can enumerate all corresponding binary strings as

$$b_1, b_2, \dots , b_k \ \dots$$

in the lexicographical order.

Thus, we can also enumerate all the functions in $\mathcal{F}$:

$$f_1, f_2, \dots , f_k, \dots$$

In other words, the set $\mathcal{F}$ is countable!

4.計算不能性と対角線論法

4. 3.対角線論法による計算不能性の証明

[定理] 停止性判定問題HALTは決定不能である.

[対角線論法による証明]
計算可能な(1入力)関数すべてからなる集合を $\mathcal{F}$ とする.
$\mathcal{F}$のすべての関数は次のように列挙できる: $f_1, f_2, \ldots, f_k, \ldots$
(文字列 $b_1, b_2, \ldots, b_k, \ldots$ に対応付けられている)
これらの文字列と関数に対して次の表 $f_i(b_j)$ を考える;

	b_1	b_2	b_3	…	b_k
f_1	1	ε	00		0
f_2	0	⊥	1		ε
f_3	0	11	0		11
:	……	……	……	……	……
:	……	……	……	……	……
f_k	ε	ε	1		0

各要素は$f_i(b_j)$の値
⊥は「無限ループ」

表を利用して新しい関数を定義する

	b_1	b_2	b_3	…	b_k
f_1	⊥	ε	00		0
f_2	0	0	1		ε
f_3	0	11	⊥		11
:	……	……	……	……	……
:	……	……	……	……	……
f_k	ε	ε	1		⊥

4. Undecidability and Diagonalization

4. 3. Proof of undecidability via Diagonalization

[Theorem] The problem HALT is undecidable.

[Proof by diagonalization]

Let $\mathcal{F}$ be a set of all computable functions (with one argument) .

All the functions in $\mathcal{F}$ is enumerable as $f_1, f_2, \ldots , f_k, \ldots$

with corresponding strings $b_1, b_2, \ldots, b_k, \ldots$

For the strings and functions, we consider the table of $f_i(b_j)$ as follows;

	b_1	b_2	b_3	…	b_k
f_1	1	ε	00		0
f_2	0	⊥	1		ε
f_3	0	11	0		11
:	………………………				
:	………………………				
f_k	ε	ε	1		0

From the table,
we design a new function

The value of $f_i(b_j)$
⊥ means “loop”

	b_1	b_2	b_3	…	b_k
f_1	⊥	ε	00		0
f_2	0	0	1		ε
f_3	0	11	⊥		11
:	………………………				
:	………………………				
f_k	ε	ε	1		⊥

4.計算不能性と対角線論法

4. 3.対角線論法による計算不能性の証明

[定理] 停止性判定問

[対角線論法による証
 計算可能な(1入力
 $\mathcal{F}$のすべての関数に
 (文字列 $b_1, b_2, \ldots,$
 これらの文字列と関

するとg は関数である.
もしg が $\mathcal{F}$ の要素なら，ある i に対して f_i となるはずである.
しかしこのとき $f_i(b_i)$ の値は定義できない.
よってg は $\mathcal{F}$ の要素ではない.
つまりg は計算可能ではない!!
この関数g は先に考えたプログラムXで計算される関数そのものである!!

	b_1	b_2	b_3	…	b_k
f_1	1	ε	00		0
f_2	0	⊥	1		ε
f_3	0	11	0		11
:	………				
:	………				
f_k	ε	ε	1		0

表を利用して
新しい関数 g を定義する

$$g(b_i) = \begin{cases} 0 & \text{if } f_i(b_i) = \perp \\ \perp & \text{if } f_i(b_i) \neq \perp \end{cases}$$

	b_1	b_2	b_3	…	b_k
f_1	⊥	ε	00		0
f_2	0	0	1		ε
f_3	0	11	⊥		11
:	………				
:	………				
f_k	ε	ε	1		⊥

4. Undecidability and Diagonalization

4. 3. Proof of undecidability via Diagonalization

[Theorem] The problem HALT is undecid

[Proof by diagonalization]

Let $\mathcal{F}$ be a set of all computable functi

All the functions in $\mathcal{F}$ is enumerable as

with corresponding strings b_1, b_2, …, b

For the strings and functions, we consid

Then, g is a function.

If g is in $\mathcal{F}$, it will appears as f_i for some i.

But $f_i(b_i)$ is not defined properly.

Thus, g is not in $\mathcal{F}$.

That is, g is not computable!!

This function g is exactly the same as the function computed by the program X!!

	b_1	b_2	b_3	…	b_k
f_1	1	ε	00		0
f_2	0	$\perp$	1		ε
f_3	0	11	0		11
:	………………………				
:	………………………				
f_k	ε	ε	1		0

From the table,
we design a new function g

$$g(b_i)=\begin{cases} 0 \text{ if } f_i(b_i)=\perp \\ \perp \text{ if } f_i(b_i)\neq\perp \end{cases}$$

	b_1	b_2	b_3	…	b_k
f_1	$\perp$	ε	00		0
f_2	0	0	1		ε
f_3	0	11	$\perp$		11
:	………………………				
:	………………………				
f_k	ε	ε	1		$\perp$

[対角線論法による証明(続き)]
もし HALT が計算可能なら，プログラムXと同様の構成により，関数 g(x) も計算可能である．ところが g(x) は計算可能ではない．
したがって HALT は計算可能ではない．　証明終わり

結論: 停止性判定問題 HALT はコンピュータでは解けない．

[関数]の個数は[計算できる関数]の
個数よりも``多い”

対角線論法:
ある要素が無限集合に属さないことを示すための論法。
ある関数の集合 G が与えられたとき，その集合に属さない関数 g を構成する方法を与えている。
こうして構成した g は、対角成分がつねに異なるため、関数集合 G には属さない。

[Proof by diagonalization (continued)]

If HALT is computable, we can compute the function g(x), as in the same manner of the program X. However, g(x) is not computable. Therefore, HALT is not computable. Q.E.D.

Our conclusion: The problem HALT is not computable.

The number of *functions* is “greater” than the number of *computable functions*.

Diagonalization

Given a set *G* of functions, construct a function *g* which does not belong to *G*.