

## Chapter 2: Introduction to Computability

What "Computation" is...

- Difference between "computable" and "incomputable"
  - Basic factor of a "computation" (Done)
  - Proof of "incomputable"...diagonalization (Today)

### 2.1. Studies on recursive functions

recursive function theory

- (1) studies on what is "computation"
- (2) proof of incomputability
- (3) structural studies on a class of incomputable functions
- (4) related mathematics fields

## 2. 計算可能性入門

計算とは何か？

- 「計算できる」と「計算できない」の違い
  - 「計算」の基本要素(前回)
  - 「計算できない」ことの証明...対角線論法(今回)

### 2.1. 帰納的関数論概観

帰納的関数論(recursive function theory)

- ① “計算”とは何かについての研究
- ② 計算不可能性の証明
- ③ 計算不可能な関数のクラスの構造的な研究
- ④ 他の数学との関連分野

## Chapter 2: Introduction to Computability

### (1) Studies on what is computation.

"When do we call a function computable?"

- recursive function theory by Kleene
- Turing machine theory by Turing

→ the whole set of recursive functions  
= the whole set of functions computable by Turing machines

Church's Thesis on the definition of "computability"

## 2. 計算可能性入門

### ① 計算とは何かについての研究

「何をもちて計算可能な関数というか？」

- クリーネが定義した帰納的関数(recursive function)
- チューリングが考えたチューリング機械(Turing machine)

→ 帰納的関数全体 = チューリング機械で計算可能な関数全体  
⇔  
計算可能性の定義...チャーチの提唱(Church's Thesis)

### (2) Proof of incomputability

- Proof of computability is easy: just give a program
- to prove incomputability  
we must prove that no program exists...  
proof tools: diagonalization  
recursive reducibility

Difficult!

### (3) Structural studies on a class of incomputable functions

hierarchical class depending of hardness  
→ structural studies

### (4) Related mathematics fields

mathematical logic

### ② 計算不可能性の証明

- 計算可能性の証明ではプログラムを作ればよい
- 計算不可能性の証明では  
どんなプログラムも作れないことの証明:  
「対角線論法」  
「帰納的還元性」

難しい

### ③ 計算不可能な関数のクラスの構造的な研究

難しさに応じて階層化されたクラス  
→ 構造的な研究

### ④ 他の数学との関連分野

数理論理学(mathematical logic)など

## Chapter 2: Introduction to Computability

4/13

### 2.4. Incomputability Proof and Diagonalization

#### Halting Problem (Problem of deciding whether it halts)

Input: a program  $A$  and an input  $x$  to it.

Output: Whether does it stop if  $x$  is given to  $A$ ?

Here we only consider the problem only for one-input programs, but we can generalize the argument into the cases of multiple inputs.

(Remark) Programs are also encoded into strings on  $\Sigma^*$ .  
That is,  $A$  and  $x$  are also considered as strings on  $\Sigma^*$ .

[A]

## 2. 計算可能性入門

4/13

### 2.4. 計算不可能性の証明と対角線論法

#### 停止問題(停止性判定問題)

入力: プログラム  $A$  とそれへの入力  $x$

出力:  $A$ へ  $x$ を与えて実行させると(いつかは)停止するか?

ここでは1入力プログラムの停止問題のみ考えるが、この結果を多入力の場合に拡張することは可能。

(注意)プログラムも $\Sigma^*$ 上にコード化可能。  
つまり、 $A$ も $x$ も $\Sigma^*$ 上の文字列と考えることができる。

[A]

for  $a, x \in \Sigma^*$

5/13

#### IsProgram( $a$ )

$\Leftrightarrow [a \text{ is a one-input grammatically correct standard program}]$

#### eval( $a, x$ )

$\equiv \begin{cases} f_a(x), & \text{if IsProgram}(a), \\ ?, & \text{otherwise.} \end{cases}$

$f_a(x)$ : output value when an input  $x$  is given to the program represented by the code  $a$

#### Theorem 2.16: IsProgram and eval are computable (programmable).

IsProgram : compiler (lint program)

eval( $a, x$ ) : it suffices to simulate the behavior of the program for a code  $a$  with an input  $x$ , i.e. interpreter or emulator

refer to Section 4.3 for detail

各  $a, x \in \Sigma^*$  に対し,

5/13

#### IsProgram( $a$ )

$\Leftrightarrow [a \text{ is a one-input grammatically correct standard program}]$

#### eval( $a, x$ )

$\equiv \begin{cases} f_a(x), & \text{IsProgram}(a) \text{ のとき,} \\ ?, & \text{その他のとき.} \end{cases}$

$f_a(x)$ : コード  $a$  が表すプログラムに入力  $x$  を加えたときの出力の値. ( $f_a(x)$  は部分関数)

#### 定理 2.16: IsProgram と eval はプログラムで実現可能.

IsProgram : コンパイラ (lint)

eval( $a, x$ ) : コード  $a$  が表すプログラムに  $x$  を入力したときの実行をシミュレートすればよい。  
つまり、インタープリタ。(エミュレータ)

詳細は4.3節

### Definition of a predicate Halt

6/13

for  $a, x \in \Sigma^*$

Halt( $a, x$ )

$\Leftrightarrow [\text{IsProgram}(a) \wedge [a \text{ stops for an input } x]]$

#### Ex. 2.1 Halting is sometimes easily checked even with loops

prog B(input  $w: \Sigma^*$ ): Boolean;

label LOOP;

begin

if  $w \neq \varepsilon$  then LOOP: goto LOOP  
else halt(0) end-if

end.

• Halt( $\lceil B \rceil, \varepsilon$ ): program B stops for an input  $\varepsilon$

•  $\neg$ Halt( $\lceil B \rceil, x$ ) for any  $x \in \Sigma^* - \{\varepsilon\}$

Thus, we can easily check whether B halts or not.

(Remark) eval( $\lceil B \rceil, \varepsilon$ ) = 0 but, for  $x \neq \varepsilon$

eval( $\lceil B \rceil, x$ ) =  $\perp$  (undefined)

Program described by code  $a$

Assume that the program is written in the standard form

### 述語 Halt の定義

6/13

各  $a, x \in \Sigma^*$  に対し

Halt( $a, x$ )

$\Leftrightarrow [\text{IsProgram}(a) \wedge [\text{入力 } x \text{ に対し } \lceil a \rceil \text{ は停止する.}]]$

#### 例 2.1 ループを含んでも停止性を簡単に判定できる場合.

prog B(input  $w: \Sigma^*$ ): Boolean;

label LOOP;

begin

if  $w \neq \varepsilon$  then LOOP: goto LOOP  
else halt(0) end-if

end.

• Halt( $\lceil B \rceil, \varepsilon$ ): 入力  $\varepsilon$  に対しプログラム B は停止.

• 任意の  $x \in \Sigma^* - \{\varepsilon\}$  に対し,  $\neg$ Halt( $\lceil B \rceil, x$ )

(注意) eval( $\lceil B \rceil, \varepsilon$ ) = 0 だが,  $x \neq \varepsilon$  に対しては

eval( $\lceil B \rceil, x$ ) =  $\perp$  (未定義)

コード  $a$  が表現するプログラム

実際のプログラムは標準形でかかれていと仮定

Bの停止性は容易に判定できる

### Theorem 2.17: Halt is incomputable.

7/13

(Proof)

By contradiction: Assume that **Halt** is computable.  
Halt is computable  $\rightarrow$  There is a program **H** to compute **Halt**.  
Using the **H**, we obtain the following program **X**.

```

prog X(input w:  $\Sigma^*$ ):  $\Sigma^*$ ;
label LOOP;
begin
  if H(w, w) then LOOP: goto LOOP
  else halt(0) end-if
end.

```

Assume that it is written in the standard form

Using the function **H** we check whether the program  $[w]$  stops for an input  $w$ . If the answer is "HALT" then the program **X** enters infinite loop, and if it is "DO NOT HALT" then it stops.

**H**: program or function, **Halt**: predicate

### 定理2.17 Haltは計算不可能

7/13

(証明)

背理法: **Halt**が計算可能だと仮定して矛盾を導く。

**Halt**が計算可能  $\rightarrow$  **Halt**を計算するプログラム**H**が存在する。  
その**H**を用いて、次のようなプログラム**X**を作る。

```

prog X(input w:  $\Sigma^*$ ):  $\Sigma^*$ ;
label LOOP;
begin
  if H(w, w) then LOOP: goto LOOP
  else halt(0) end-if
end.

```

実際には標準形で書かれていると仮定。

プログラム $[w]$ に $w$ を入力したとき停止するかどうかをプログラム**H**を呼び出して判定し、  
答が *true* なら無限ループに入り、  
答が *false* なら0を出力して停止する、というプログラム

**H**: プログラム, **Halt**: 述語

Let  $x_1 = [X]$  and input  $x_1$  to the program **X**  
(i) enters an infinite loop, or  
(ii) stops normally with the output 0.

$X(w)$  determines if program  $w$  halts on the input  $w$  or not using the program **H**, and it does not halt if the answer is *true*, and it halts if the answer is *false*.

Case (i)

- Since it enters infinite loop,  $\neg \text{Halt}(x_1, x_1)$
- at the if statement in the program **X** we have  $\text{H}(x_1, x_1) = \text{false}$   
So,  $\text{halt}(0)$  is executed (normal termination): contradiction

Case (ii)

- Since it stops,  $\text{Halt}(x_1, x_1)$  is true.
- at the if statement in the program **X** we have  $\text{H}(x_1, x_1) = \text{true}$   
So, it enters an infinite loop: contradiction

In either case we have a contradiction.

That is, the assumption that "**Halt** is computable" is wrong.  
End of proof

**H**: program or function, **Halt**: predicate

$x_1 = [X]$  とし,  $x_1$  を  
プログラム**X**に入力

- (i) ループに入ってしまう, or
- (ii) 0を出力して停止.

$X(w)$   
プログラム $[w]$ に $w$ を入力したとき停止するか  
どうかをプログラム**H**を呼び出して判定し、  
答が *true* なら無限ループに入り、  
答が *false* なら0を出力して停止する

8/13

(i) を仮定すると...

- プログラムがループに入るから,  $\text{H}(x_1, x_1) = \text{true}$
- つまり  $X(x_1)$  は停止する: 仮定に矛盾

(ii) を仮定すると...

- プログラムが終了するから,  $\text{H}(x_1, x_1) = \text{false}$
- つまり  $X(x_1)$  は停止しない: 仮定に矛盾

どちらの場合も矛盾を生じる。  
したがって「**Halt**は計算可能」という仮定は誤り。

証明終

**H**: プログラム  
**Halt**: 述語

### Diagonalization

9/13

#### Enumerable infinite set:

a set with one-to-one correspondence with the set of all natural numbers

**Enumerable set:** finite or enumerable infinite set.

that is, a set whose elements are enumerable one by one.

**Ex. 1:** The set  $E$  of all even positive integers is enumerable infinite.

one-to-one correspondence between an element  $i$  of the set of all natural numbers and an element  $2i$  of the set  $E$

**Ex. 2:** The set  $Z$  of all integers is enumerable infinite.

We can enumerate them as  $Z = \{0, 1, -1, 2, -2, 3, -3, \dots\}$ .

**Ex. 3:** The set  $R$  of all rational numbers is enumerable infinite. (Why?)

**Theorem:** The set  $R$  of all real numbers is not enumerable.

### 対角線論法

9/13

**可算無限集合:** 自然数全体の集合との間に1対1対応がある集合のこと。

**可算集合:** 有限または可算無限である集合のこと。

つまり、1つずつ要素を取り出してきて、もれなく書き並べられるもの

**例1.** 正の偶数全体の集合  $E$  は可算無限である。

自然数全体の集合  $N$  の要素  $i$  と,  $E$  の要素  $2i$  を対とする1対1対応がある。

**例2.** 整数全体の集合  $Z$  は可算無限である。

1対1対応がある。または,  $Z = \{0, 1, -1, 2, -2, 3, -3, \dots\}$  と列挙できる。

**例3.** 有理数全体の集合は可算無限である。(なぜか?)

**定理:** 実数全体の集合  $R$  は非可算である。

**Theorem: The set R of all real numbers is not enumerable.**

10/13

Using the diagonalization we prove that the set S of all real numbers between 0 and 1 is not enumerable. By contradiction, we assume that it is enumerable:

0.a<sub>11</sub>a<sub>12</sub>a<sub>13</sub>...  
0.a<sub>21</sub>a<sub>22</sub>a<sub>23</sub>...  
0.a<sub>31</sub>a<sub>32</sub>a<sub>33</sub>...  
0.a<sub>41</sub>a<sub>42</sub>a<sub>43</sub>...

0.a<sub>11</sub>a<sub>12</sub>a<sub>13</sub>...  
0.a<sub>21</sub>a<sub>22</sub>a<sub>23</sub>...  
0.a<sub>31</sub>a<sub>32</sub>a<sub>33</sub>...  
0.a<sub>41</sub>a<sub>42</sub>a<sub>43</sub>...

0.a<sub>k1</sub>a<sub>k2</sub>a<sub>k3</sub>... a<sub>kk</sub>

0.a<sub>k1</sub>a<sub>k2</sub>a<sub>k3</sub>... where a<sub>ij</sub> ∈ {0, 1, ..., 9}

Define a new real number x by collecting those digits in the diagonal

x = 0.b<sub>1</sub>b<sub>2</sub>b<sub>3</sub>...

where b<sub>k</sub> is defined by

if a<sub>kk</sub>=1 then b<sub>k</sub> = 2 else b<sub>k</sub>=1

The number x defined above is obviously between 0 and 1, but it is different from any number listed above since it is different at its diagonal position.

That is, x does not belong to S, which is a contradiction.

Therefore, our assumption that S is enumerable is wrong.

**定理: 実数全体の集合Rは非可算である.**

10/13

0以上1未満の実数全体の集合Sが非可算であることを対角線論法で証明する。可算であると仮定すると、すべての要素を書き並べることができる:

0.a<sub>11</sub>a<sub>12</sub>a<sub>13</sub>...  
0.a<sub>21</sub>a<sub>22</sub>a<sub>23</sub>...  
0.a<sub>31</sub>a<sub>32</sub>a<sub>33</sub>...  
0.a<sub>41</sub>a<sub>42</sub>a<sub>43</sub>...

0.a<sub>11</sub>a<sub>12</sub>a<sub>13</sub>...  
0.a<sub>21</sub>a<sub>22</sub>a<sub>23</sub>...  
0.a<sub>31</sub>a<sub>32</sub>a<sub>33</sub>...  
0.a<sub>41</sub>a<sub>42</sub>a<sub>43</sub>...

0.a<sub>k1</sub>a<sub>k2</sub>a<sub>k3</sub>... a<sub>kk</sub>

0.a<sub>k1</sub>a<sub>k2</sub>a<sub>k3</sub>... ただし, a<sub>ij</sub> ∈ {0, 1, ..., 9}

上の並びで対角線にある数に注目し, 新たな無限小数

x = 0.b<sub>1</sub>b<sub>2</sub>b<sub>3</sub>...

を作る。ここで,

if a<sub>kk</sub>=1 then b<sub>k</sub> = 2 else b<sub>k</sub>=1

としてb<sub>k</sub>を定める。

このように作られた無限小数は明らかに0と1の間の実数である。

しかし, 作り方から, 上に列挙したどの要素とも等しくない(対角線の所で必ず異なる)。

つまり, xはSに属さないことになり, 矛盾である。

したがって, Sが可算であるという仮定に誤りがある。

**Ex.2.17 Program X used in the proof of incomputability of Halt**

11/13

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

f<sub>X</sub>: function computed by the program X

if f<sub>X</sub>(a<sub>i</sub>) = ⊥ then ¬Halt(a<sub>i</sub>, a<sub>i</sub>)

∴ f<sub>X</sub>(a<sub>i</sub>) = 0

if f<sub>X</sub>(a<sub>i</sub>) ≠ ⊥ then, Halt(a<sub>i</sub>, a<sub>i</sub>)

∴ f<sub>X</sub>(a<sub>i</sub>) = ⊥

That is, there is no function f<sub>a</sub> in the set F<sub>1</sub> of functions such that f<sub>X</sub> = f<sub>a</sub>.

★The number of programs is enumerable, while the number of functions is not.

**例2.17 Haltの計算不可能性の証明の中で用いたプログラムX**

11/13

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

f<sub>X</sub>: プログラムXが計算する関数

f<sub>X</sub>(a<sub>i</sub>) = ⊥ のとき, ¬Halt(a<sub>i</sub>, a<sub>i</sub>)

∴ f<sub>X</sub>(a<sub>i</sub>) = 0

f<sub>X</sub>(a<sub>i</sub>) ≠ ⊥ のとき, Halt(a<sub>i</sub>, a<sub>i</sub>)

∴ f<sub>X</sub>(a<sub>i</sub>) = ⊥

つまり, f<sub>X</sub> = f<sub>a</sub>となるf<sub>a</sub>は計算可能な関数の集合F<sub>1</sub>の中に存在しない。

★プログラムの個数は可算無限だが、関数の個数は非可算無限

**Theorem 2.18 The following function diag is incomputable.**

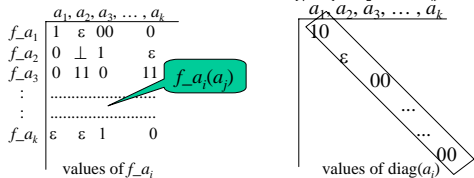
12/13

diag(a) = f<sub>a</sub>(a) ≠ 0, if Halt(a, a)  
ε, otherwise

Proof:

Let F<sub>1</sub> be a set of all computable functions (with one argument). Since a code of a program is an element of Σ\*, we can enumerate all grammatically correct program codes a<sub>1</sub>, a<sub>2</sub>, ..., a<sub>k</sub> ... in the psuedo-lexicographical order.

We can also enumerate all the functions of F<sub>1</sub>: f<sub>a1</sub>, f<sub>a2</sub>, ..., f<sub>ak</sub>, ...



diag(a<sub>i</sub>) = w ≠ 0, if the value w of (f<sub>a<sub>i</sub></sub>, a<sub>i</sub>) is not undefined ⊥.  
ε, otherwise

**定理2.18 次の関数 diag は計算不可能**

12/13

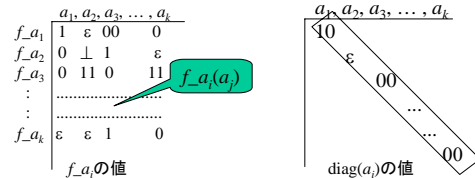
diag(a) = f<sub>a</sub>(a) ≠ 0, Halt(a, a)のとき  
= ε, その他のとき

証明:

計算可能な(1引数の)関数全体の集合をF<sub>1</sub>とする。プログラムのコードはΣ\*の元だから、

“文法的に正しいプログラムのコード”を小さい順にa<sub>1</sub>, a<sub>2</sub>, ..., a<sub>k</sub> ... と並べることができる。(長さ優先の辞書式順序)

F<sub>1</sub>の関数もf<sub>a1</sub>, f<sub>a2</sub>, ..., f<sub>ak</sub>, ... と並べることができる。



diag(a<sub>i</sub>) = w ≠ 0, f<sub>a<sub>i</sub></sub>(a<sub>i</sub>)の値wが未定義 ⊥でないとき  
ε, その他のとき

diag is different from any  $f_{a_i}$ .

**Why:**  $\text{diag}()$  is different from  $f_{a_i}()$  at its diagonal position.

$$\text{diag}(a_i) \neq f_{a_i}(a_i)$$

*(two functions  $f_1()$  and  $f_2()$  are different if there exists an input  $x$  such that  $f_1(x) \neq f_2(x)$ .)*



$$\text{diag} \notin F_1$$

That is, the function  $\text{diag}$  is not computable.

End of proof

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  $G$ .

$\text{diag}$ はどの  $f_{a_i}$ とも異なる。

**理由:**  $\text{diag}()$ と  $f_{a_i}()$ は、対角線の所で必ず異なる。

$$\text{diag}(a_i) \neq f_{a_i}(a_i)$$

$$\text{diag} \notin F_1$$

つまり、関数  $\text{diag}$ は計算可能でない。

証明終

[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。