2. 計算可能性入門

計算とは何か？

- 「計算できる」ことと「計算できない」ことの違い
    - ➢ 「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研究
④ 他の数学との関連分野

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

① 計算とは何かについての研究

「何をもっ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)

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<u>作れない</u>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研究

難しさに応じて階層化されたクラス

→構造的研究

④ 他の数学との関連分野

数理論理学(mathematical logic)など

(2) Proof of incomputability

- Proof of computability is easy: just give a program
- to prove incomputability

  must prove that no program exists…

Difficult!

proof tools: diagonalization

recursive reducibility

(3) Structural studies on a class of incomputable functions

hierarchical class depending of hardness

→structural studies

(4) Related mathematics fields

mathematical logic

2. 計算可能性入門

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Σ^*上にコード化可能.

つまり, A も x もΣ^*上の文字列と考えることができる.

今日の暗黙の記法

A	大文字はプログラム名
$\ulcorner A \urcorner$	$\ulcorner\ \urcorner$ はプログラムのコード
a	小文字はプログラムコード

Chapter 2: Introduction to Computability

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 Σ^*.
That is, A and x are also considered as strings on Σ^*.

Implicit Notations

A	Capital means "program name"
$\ulcorner A \urcorner$	$\ulcorner\ \urcorner$ means program code
a	Small means "program code"

各 $a, x \in \Sigma^*$ に対し,

IsProgram(a)

$\Leftrightarrow$ [aは1入力の文法的に正しい標準形プログラムのコード]

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A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

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

IsProgram(*a*)

$\Leftrightarrow$ [*a* 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 A represented by the code *a*

Theorem2.16: 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

述語Haltの定義

コードaが表現するプログラム

各$a, x \in \Sigma^*$に対し

Halt(a, x)

$\Leftrightarrow$ [IsProgram(a) $\wedge$ [入力 x に対し$\lfloor a \rfloor$は停止する.]]

Definition of a predicate Halt

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

Halt(a, x)

$\Leftrightarrow$ [IsProgram(a) $\wedge$ [$\lfloor a \rfloor$ stops for an input x]]

Program described by code *a*

定理2.17 **Halt**は計算不可能

（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➔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: Σ*): Σ*;
 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$\lfloor w \rfloor$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

Theorem 2.17: Halt is incomputable.

(Proof)

By contradiction : Assume that Halt is computable.

Halt is computable ➔ There is a program H to compute Halt.

Using the H, we obtain the following program X.

```
prog X(input w: Σ*): Σ*;
 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 $\lfloor w \rfloor$ 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

X(w)
プログラム$\lfloor w \rfloor$ 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

$x = \lceil \mathrm{X} \rceil$とし，$x$ を
プログラムXに入力
(i) 無限ループに入ってしまう，or
(ii) 0を出力して停止.

(i) を仮定すると…
- プログラムがループに入るから，H $(x, x)=$ *true*
- つまり X(x) 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，H $(x, x)=$*false*
- つまり X(x) 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**

Let $x = \lceil X \rceil$ and input x to the program X

(i) enters an infinite loop, or

(ii) stops normally with the output 0.

Case (i)

- Since it enters infinite loop, $\neg$Halt(x, x)
- at the if statement in the program X we have H (x , x)=false

  So, halt(0) is executed（normal termination）: contradiction

Case (ii)

- Since it stops, Halt(x, x) is true.
- at the if statement in the program X we have H (x, x)=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

定理2.17の別証明(対角線論法による)

証明:
計算可能な(1引数の)関数全体の集合をF_1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

$$a_1, a_2, \dots, a_k, \dots$$

と(長さ優先の辞書式順序で)並べることができる.
よってF_1の関数を $f\_a_1, f\_a_2, \dots, f\_a_k, \dots$ と並べることができ、以下の表をえる。

	a_1	a_2	a_3	$\dots$	a_k
f_a_1	1	ε	00		0
f_a_2	0	⊥	1		ε
f_a_3	0	11	0		11
:	....				
:	....				
f_a_k	ε	ε	1		0

$f_a_i(a_j)$の値

Another proof of Theorem 2.17 (by diagonalization)

Proof:

Let F_1 be a set of all computable functions (with one argument) .

Since each program code is in Σ^*, we can enumerate all grammatically correct program codes

$a_1, a_2, \ldots, a_k \ldots$

in the psuedo-lexicographical order. *Thus, we can also enumerate all the functions in F_1:*

$f\_a_1, f\_a_2, \ldots, f\_a_k, \ldots$

that gives the following table:

	a_1	a_2	a_3	$\ldots$	a_k
f_a_1	1	ε	00		0
f_a_2	0	⊥	1		ε
f_a_3	0	11	0		11
:	....				
:	....				
f_a_k	ε	ε	1		0

The value of $f\_a_i(a_j)$

定理2.17の別証明(対角線論法による)

証明:

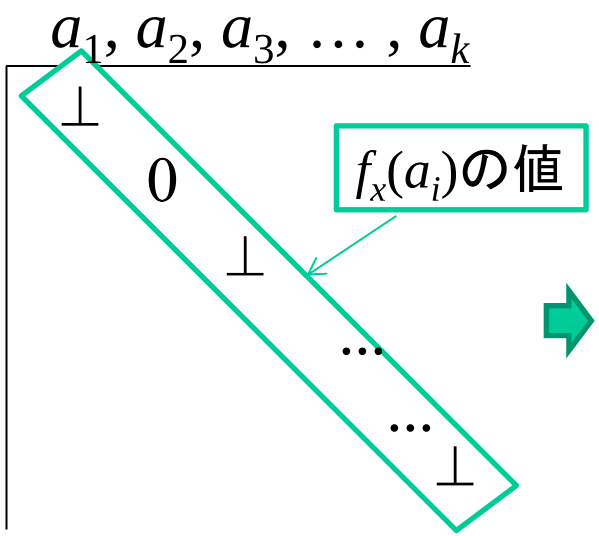
ここで Halt が計算可能なら、それを計算するプログラム H が存在する。
そして H を使うと以下の関数 f_x が計算可能であることがわかる。

$$f_x(a) = \perp, \quad \mathrm{Halt}(a, a)\text{のとき}$$
$$= 0, \quad \text{その他のとき}$$

先の表と照らし合わせると…

	a_1	a_2	a_3	…	a_k
f_a_1	1	ε	00		0
f_a_2	0	⊥	1		ε
f_a_3	0	11	0		11
⋮	………………………				
⋮	………………………				
f_a_k	ε	ε	1		0

f_a_iの値



どんな整数 i に対しても以下が成立:

$$f\_a_i(a_i) \neq f_x(a_i)$$

よって f_x は F_1 の中に現れない!

よって $f_x(a)$ は F_1 の要素ではない。つまり Halt は計算可能ではない。

Another proof of Theorem 2.17 (by diagonalization)

Proof:

If Halt is computable, there exists a program **H** that computes Halt.

Using **H**, we can compute the following function f_x.

$$f_x(a) = \perp, \quad \text{if Halt}(a, a)$$
$$= 0, \quad \text{otherwise}$$

Comparing to the table…

	a_1	a_2	a_3	…	a_k
f_a_1	1	ε	00		0
f_a_2	0	⊥	1		ε
f_a_3	0	11	0		11
:	……………………				
:	……………………				
f_a_k	ε	ε	1		0

Values of f_a_i

a_1	a_2	a_3	…	a_k
⊥				
	0			
		⊥		
			…	
			…	
				⊥

Values of $f_x(a_i)$

For any integer i, we have:
$f\_a_i(a_i) \neq f_x(a_i)$

Thus f_x does not appear in F_1!

Hence $f_x(a)$ is not an element in F_1. Therefore, Halt is not computable.

[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。

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*.

対角線論法

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, ...}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.

Diagonalization

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, ...}.

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

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

定理：実数全体の集合Rは非可算である.

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:

$0.a_{11}a_{12}\,a_{13}\ldots$
$0.a_{21}a_{22}\,a_{23}\ldots$
$0.a_{31}a_{32}\,a_{33}\ldots$
$0.a_{41}a_{42}\,a_{43}\ldots$

$0.a_{k1}a_{k2}\,a_{k3}\ldots$　ただし, $a_{ij}\in\{0,1,\ldots,9\}$

$0.\boxed{a_{11}}a_{12}\,a_{13}\ldots$
$0.a_{21}\boxed{a_{22}}\,a_{23}\ldots$
$0.a_{31}a_{32}\,\boxed{a_{33}}\ldots$
$0.a_{41}a_{42}\,a_{43}\ldots$

$0.a_{k1}a_{k2}\,a_{k3}\ldots\ \boxed{a_{kk}}$

上の並びで対角線上にある数に注目し, 新たな無限小数
$x = 0.b_1b_2b_3\ldots$
を作る. ここで,
if a_{kk}=1 then b_k = 2 else b_k=1
としてb_k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.

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

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:

$0.a_{11}a_{12}\,a_{13}...$
$0.a_{21}a_{22}\,a_{23}...$
$0.a_{31}a_{32}\,a_{33}...$
$0.a_{41}a_{42}\,a_{43}...$

$0.a_{k1}a_{k2}\,a_{k3}...$ where $a_{ij} \in \{0, 1, ... , 9\}$

$$\boxed{\begin{array}{l} 0.a_{11}a_{12}\,a_{13}... \\ 0.a_{21}a_{22}\,a_{23}... \\ 0.a_{31}a_{32}\,a_{33}... \\ 0.a_{41}a_{42}\,a_{43}... \\ \\ 0.a_{k1}a_{k2}\,a_{k3}...\,a_{kk} \end{array}}$$

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

$x = 0.b_1b_2b_3...$

where b_k is defined by

if a_{kk}=1 then $b_k = 2$ else b_k=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.