# 2. 計算可能性入門

## 2.3. for-times計算可能性

省略

## 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.

# Chapter 2: Introduction to Computability

## 2.3. for-times Computability

omitted

## 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 $\Sigma^*$.

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 は計算可能.**

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 represented by the code $a$

**Theorem2.16: IsProgram and eval are computable.**

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($a$, $x$)

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

例2.1 ループを含んでいても停止性を簡単に判定できる場合.

```
prog B(input w: Σ*): Boolean;
label LOOP;
begin
  if w≠ε then LOOP: goto LOOP
         else halt(0) end-if
end.
```

実際のプログラムは
標準形でかかれていると仮定

- 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$\rceil$, $x$)

Bの停止性は
容易に判定できる

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

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

## Definition of a predicate Halt

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

Halt($a$, $x$)

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

**Program described by code *a***

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

```
prog B(input w: Σ*): Boolean;
  label LOOP;
  begin
    if w ≠ ε then LOOP: goto LOOP
             else halt(0) end-if
  end.
```

Assume that the program is written in the standard form

- 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 stops or not.

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

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

## 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Haltが存在する.

そのHalt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 Halt(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Halt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

**Halt：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 **Halt** to compute Halt.

Using the **Halt,** we obtain the following program X.

```
prog X(input w: Σ*): Σ*;
 label LOOP;
 begin                          Assume that it is written in the standard form
    if Halt(w, w) then LOOP: goto LOOP
                  else halt(0) end-if
 end.
```

Using the function **Halt** 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.

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

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Halt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

$x_1 = \lceil X \rceil$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.

(i) の場合
・プログラムがループに入るから, Halt $(x_1 , x_1 ) = true$
・つまり $X(x_1)$ は停止する: 仮定に矛盾

(ii)の場合
・プログラムが終了するから, Halt $(x_1, x_1) = false$
・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
即ち,「Haltは計算可能」という仮定が誤り.
証明終

Halt:プログラム
Halt:述語

Let $x_1 = \lceil X \rceil$ and input $x_1$ 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($x_1$, $x_1$ )
- at the if statement in the program X we have **Halt** ($x_1$ , $x_1$ )=false

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

Case (ii)

- Since it stops, Halt($x_1$, $x_1$) is true.
- at the if statement in the program X we have **Halt** ($x_1$, $x_1$)=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

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

**定理2.18 次の関数 diag は計算不可能**

$\mathrm{diag}(a) = f\_a(a) \# 0$, $\mathrm{Halt}(a, a)$のとき
$= \varepsilon$, 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$F_1$とする.
プログラムのコードは$\Sigma^*$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$a_1, a_2, \ldots, a_k, \ldots$
と並べることができる.(長さ優先の辞書式順序)
$F_1$の関数も$f_a_1, f_a_2, \ldots, f_a_k, \ldots$ と並べることができる.

| | $a_1$ | $a_2$ | $a_3$ | … | $a_k$ |
|---|---|---|---|---|---|
| $f\_a_1$ | 1 | ε | 00 | | 0 |
| $f\_a_2$ | 0 | ⊥ | 1 | | ε |
| $f\_a_3$ | 0 | 11 | 0 | | 11 |
| : | ..... | | | | |
| : | ..... | | | | |
| $f\_a_k$ | ε | ε | 1 | | 0 |

$f\_a_i(a_j)$

$f_a_i$の値

| | $a_1$ | $a_2$ | $a_3$ | … | $a_k$ |
|---|---|---|---|---|---|
| | 10 | ε | 00 | … | … 00 |

$\mathrm{diag}(a_i)$の値

$\mathrm{diag}(a_i) = w\#0$, $f_a_i(a_i)$の値$w$が未定義 ⊥でないとき
$= \varepsilon$, その他のとき

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

$$\mathbf{diag}(a) \begin{cases} = f\_a(a) \# 0, & \text{if } \mathbf{Halt}(a, a) \\ = \varepsilon, & \text{otherwise} \end{cases}$$

Proof:

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

Since a code of a program is an element of $\Sigma^*$,

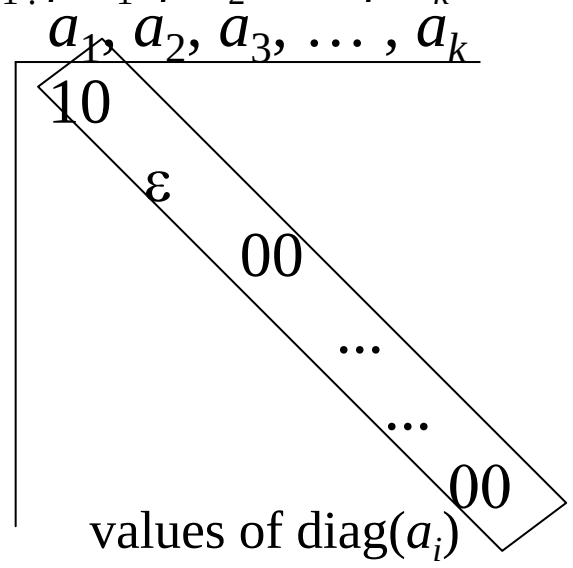
we can enumerate all grammatically correct program codes

$a_1, a_2, \ldots, a_k$ ... in the psuedo-lexicographical order.

*We can also enumerate all the functions of* $F_1$ *:* $f\_a_1, f\_a_2, \ldots, f\_a_k$,...

| | $a_1$, $a_2$, $a_3$, … , $a_k$ |
|---|---|
| $f\_a_1$ | 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 $f\_a_i$




values of diag($a_i$)

diag($a_i$) = $w$#0, if the value $w$ of ($f\_a_i$ , $a_i$ ) is not undefined ⊥ .

ε, otherwise

diagはどの$f_a_i$とも異なる.

理由:diag()と$f_a_i$()は，対角線の所で必ず異なる.

$$\mathrm{diag}(a_i) \neq f\_a_i(a_i)$$

$$\mathrm{diag} \notin F_1$$

つまり，関数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。

diag is different from any $f\_a_i$.

Why: 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 diag is not computable.

End of proof

**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：

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

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

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

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

上の並びで対角線上にある数に注目し，新たな無限小数
$x = 0.b_1b_2b_3...$
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:

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

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

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

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

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.

## 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

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

*f_X*: プログラムXが計算する関数

$f\_a_i(a_i) = \perp$ のとき, $\neg\mathrm{Halt}(a_i, a_i)$

$\therefore\ f\_\mathrm{X}(a_i) = 0$

$f\_a_i(a_i) \neq \perp$ のとき, $\mathrm{Halt}(a_i, a_i)$

$\therefore\ f\_\mathrm{X}(a_i) = \perp$

つまり, $f\_X = f\_a_i$ となる $f\_a_i$ は
計算可能な関数の集合 $F_1$ 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

## Ex.2.17 Program X used in the proof of incomputablity of Halt

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

*f*_X: function computed by the program X

if $f\_a_i(a_i) = \perp$ then $\neg\text{Halt}(a_i, a_i)$

$\therefore f\_\text{X}(a_i) = 0$

if $f\_a_i(a_i) \neq \perp$ then, $\text{Halt}(a_i, a_i)$

$\therefore f\_\text{X}(a_i) = \perp$

That is, there is no function $f\_a_i$ in the set $F_1$ of functions such that $f\_\text{X} = f\_a_i$.

## 2.5 計算不可能な関数の例

関数の性質についての述語は計算不可能になることが多い.

例2.19. 与えられたプログラムが計算する関数が恒等的に0か?

$$\mathrm{Zero}(a) \Leftrightarrow \mathrm{IsProgram}(a) \wedge \forall x[f\_a(x) = 0]$$

Zeroは計算不可能.

```
prog X a,x(input w: Σ*): Σ*;
const c1=a; c2=x;
begin
   if HaltInTime (c1,c2,|w|) then halt(1)
                             else  halt(0) end-if
end.
```

ただし,

$$\mathrm{HaltInTime}(a, x, t)$$

$$\Leftrightarrow [\mathrm{IsProgram}(a) \wedge (\lfloor a \rfloor(x)\text{が}t\text{ステップ以内に停止})]$$

これは計算可能

## 2.5 Examples of incomputable functions

Predicates concerning on properties of functions are often incomputable.

Ex.2.19. Does a given program always output 0?

$$\text{Zero}(a) \Leftrightarrow \text{IsProgram}(a) \wedge \forall x[f\_a(x) = 0]$$

Zero is incomputable.

```
prog X a,x(input w: Σ*): Σ*;
const c1=a; c2=x;
begin
   if HaltInTime (c1,c2,|w|) then halt(1)
                             else  halt(0) end-if
end.
```

where,

$$\text{HaltInTime}(a, x, t)$$

$$\Leftrightarrow [\text{IsProgram}(a) \wedge (\lfloor a \rfloor(x) \text{ stops within } t \text{ steps})]$$

**Computable!!**

すべての $a,\ x \in \Sigma^*$ で

$$\mathrm{Halt}(a,\ x) \leftrightarrow \exists t[\mathrm{HaltInTime}(a, x, t)]$$

$$\leftrightarrow \exists w[\mathrm{X}_{a,x}(w)\text{が}1\text{を出力}]$$

$$\leftrightarrow \neg\mathrm{Zero}(\lceil \mathrm{X}_{a,x} \rceil).$$

つぎのようにして, $\mathrm{Halt}(a, x)$か否か判定可能

(1) $a$と$x$からプログラム$\lceil \mathrm{X}_{a,x} \rceil$のコードを求める.

(2)$\mathrm{Zero}(\lceil \mathrm{X}_{a,x} \rceil)$を判定する.

(3)$\neg\mathrm{Zero}(\lceil \mathrm{X}_{a,x} \rceil) \Rightarrow \mathrm{Halt}(a, x)$

$\mathrm{Zero}(\lceil \mathrm{X}_{a,x} \rceil) \Rightarrow \neg\mathrm{Halt}(a, x)$

```
prog X a,x(input w: Σ*): Σ*;
const c1=a; c2=x;
begin
   if HaltInTime (c1,c2,|w|) then halt(1)
                             else  halt(0) end-if
end.
```

ここで, 関数 $\mathrm{code}(a, x) \equiv \lceil \mathrm{X}_{a,x} \rceil$は計算可能.

**よって, 上の(1)は計算可能. したがって, もしZeroが計算可能なら Haltも計算可能となり, 矛盾. すなわち, Zeroは計算不可能**

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

$$\mathrm{Halt}(a,\ x) \leftrightarrow \exists t[\mathrm{HaltInTime}(a, x, t)]$$
$$\leftrightarrow \exists w[\mathrm{X}_{a,x}(w) \text{ outputs } 1]$$
$$\leftrightarrow \neg\mathrm{Zero}(\ulcorner \mathrm{X}_{a,x} \urcorner).$$

We can decide whether $\mathrm{Halt}(a, x)$ or not as follows:

(1) Given $a$ and $x$, obtain a code of a program $\ulcorner \mathrm{X}_{a,x} \urcorner$.

(2)Check $\mathrm{Zero}(\ulcorner \mathrm{X}_{a,x} \urcorner)$.

(3)$\neg\mathrm{Zero}(\ulcorner \mathrm{X}_{a,x} \urcorner) \Rightarrow \mathrm{Halt}(a, x)$

$\mathrm{Zero}(\ulcorner \mathrm{X}_{a,x} \urcorner) \Rightarrow \neg\mathrm{Halt}(a, x)$

Here, the function $\mathrm{code}(a, x) \equiv \ulcorner \mathrm{X}_{a,x} \urcorner$ is computable.

Thus, (1) is computable. Therefore, if Zero is computable, then Halt is also computable, a contradiction. That is, Zero is incomputable.

## 例2.20 与えられたプログラムは全域的か?

$$\mathrm{Total}(a) \Leftrightarrow [\mathrm{IsProgram}(a) \wedge \forall x[\, f\_a(x) \neq \perp\, ]]$$

与えられたプログラムAに対し，次の作業を考える.
(1)その中のhalt文を探し出す. →halt($y$)と仮定
(2)halt($y$)→if $y \neq 0$ then T: goto T else halt($y$) end-if　と書き換える.
(3)以上のことをすべてのhalt文について行う.
出来上がったプログラムをBとする.

プログラムAが0以外を出力すると必ず無限ループに陥る.
i.e., Aが常に0を出力しない限り，$f$_Bは全域的にならない.

上記の変換は計算可能➔次の関数が計算可能

replace($a$) = $b$,　IsProgram($a$)のとき,
= $a$,　その他のとき.

ただし，$b$は$\lfloor a \rfloor$ を上のように変換したプログラムのコード

一方、$\forall a \in \Sigma^*$[Zero($a$)⇔Total(replace($a$))]

よって，Totalが計算可能なら，Zeroも計算可能
i.e., Totalは計算不可能

## Ex. 2.20 Is a given program total?

$$\text{Total}(a) \Leftrightarrow [\text{IsProgram}(a) \wedge \forall x[f\_a(x) \neq \perp]]$$

Given a program A, consider the following computation.
(1) Find all halt statements.→assume that it is halt($y$).
(2) Rewrite: halt($y$)→if $y \neq 0$ then T: goto T else halt($y$) end-if.
(3) For each halt statement, do the above.
Let B be the resulting program.

If the program A outputs other than 0 then it enters infinite loop.
i.e., unless A always outputs 0, $f$_B is not total.

The above conversion is computable➔So is the following function

replace($a$) = $b$, if IsProgram($a$),
= $a$, otherwise,
where $b$ is a code$\lfloor a \rfloor$of the converted program.

Thus, (1) is computable. Therefore, if Zero is computable, then Halt is also computable, a contradiction. That is, Zero is incomputable.