

1/13

## 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:  $A$  を  $x$  と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。

1/13

## 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  $\Sigma^*$ .

2/13

各  $a, x \in \Sigma^*$  に対し、

$\text{IsProgram}(a)$   
 $\Leftrightarrow [a \text{ は 1 入力の文法的に正しい標準形プログラムのコード}]$   
 $\text{eval}(a, x)$   
 $\equiv \begin{cases} f_a(x), & \text{IsProgram}(a) \text{ のとき,} \\ ?, & \text{その他のとき.} \end{cases}$

$f_a(x)$ : 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:  $\text{IsProgram}$  と  $\text{eval}$  は計算可能.

$\text{IsProgram}$ : コンパイラ(lint)  
 $\text{eval}(a, x)$ : 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

2/13

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

$\text{IsProgram}(a)$   
 $\Leftrightarrow [a \text{ is a one-input grammatically correct standard program}]$   
 $\text{eval}(a, x)$   
 $\equiv \begin{cases} f_a(x), & \text{if IsProgram}(a), \\ ?, & \text{otherwise.} \end{cases}$

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

Theorem 2.16:  $\text{IsProgram}$  and  $\text{eval}$  are computable.

$\text{IsProgram}$ : compiler(lint program)  
 $\text{eval}(a, x)$ : 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

3/13

### 述語 Halt の定義

各  $a, x \in \Sigma^*$  に対し

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

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  $\neq$   $\epsilon$  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っていると仮定

- $\text{Halt}(\lceil B \rceil, \epsilon)$ : 入力  $\epsilon$  に対しプログラム  $B$  は停止.
- 任意の  $x \in \Sigma^* - \{\epsilon\}$  に対し、 $\neg \text{Halt}(\lceil B \rceil, x)$

(注意)  $\text{eval}(\lceil B \rceil, \epsilon) = 0$  だが、 $x \neq \epsilon$  に対しては  $\text{eval}(\lceil B \rceil, x) = \perp$  (未定義)

3/13

### Definition of a predicate Halt

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

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

Ex. 2.1 Halting is sometimes easily checked even with loops

```

prog B(input w:  $\Sigma^*$ ): Boolean;
label LOOP;
begin
  if w  $\neq$   $\epsilon$  then LOOP: goto LOOP
  else halt(0) end-if
end.
  
```

Assume that the program is written in the standard form

- $\text{Halt}(\lceil B \rceil, \epsilon)$ : program  $B$  stops for an input  $\epsilon$
- $\neg \text{Halt}(\lceil B \rceil, x)$  for any  $x \in \Sigma^* - \{\epsilon\}$   
 Thus, we can easily check whether  $B$  halts or not.

(Remark)  $\text{eval}(\lceil B \rceil, \epsilon) = 0$  but, for  $x \neq \epsilon$   
 $\text{eval}(\lceil B \rceil, x) = \perp$  (undefined)

4/13

**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;           実際に標準形で書かれていると仮定。
begin
  if Halt(w, w) then LOOP: goto LOOP
  else halt(0) end-if
end.

```

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

4/13

**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
  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

5/13

$x_1 = [X]$ とし、 $x_1$ をプログラムXに入力

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

(i) 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、 $\text{Halt}(x_1, x_1) = \text{true}$
- 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、 $\text{Halt}(x_1, x_1) = \text{false}$
- 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

Halt: プログラム  
Halt: 述語

5/13

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.

Case (i)

- Since it enters infinite loop,  $\neg \text{Halt}(x_1, x_1)$
- at the if statement in the program X we have  $\text{Halt}(x_1, x_1) = \text{false}$   
So, 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  $\text{Halt}(x_1, x_1) = \text{true}$   
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

Halt: program or function, Halt: predicate

6/13

**定理2.18 次の関数 diag は計算不可能**

$\text{diag}(a) = f_a(a) \neq 0$ , 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 $a_1, a_2, \dots, a_k, \dots$   
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 $f_{a_1}, f_{a_2}, \dots, f_{a_k}, \dots$ と並べることができる。

$f_{a_1}$	1	$\varepsilon$	00	0
$f_{a_2}$	0	$\perp$	1	$\varepsilon$
$f_{a_3}$	0	11	0	11
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$f_{a_k}$	$\varepsilon$	$\varepsilon$	1	0

$f_{a_i}$ の値

$a_1$	$a_2$	$a_3$	$\dots$	$a_k$
0	$\varepsilon$	00	$\dots$	00
$\varepsilon$	00	$\dots$	$\dots$	00
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\varepsilon$	$\varepsilon$	1	$\dots$	0

$\text{diag}(a_i)$ の値

$\text{diag}(a_i) = w \neq 0$ ,  $f_{a_i}(a_i)$ の値 $w$ が未定義  $\perp$ でないとき  
 $\varepsilon$ , その他のとき

6/13

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

$\text{diag}(a) = f_a(a) \neq 0$ , if Halt( $a, a$ )  
 $= \varepsilon$ , otherwise

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  
 $a_1, a_2, \dots, a_k, \dots$  in the psuedo-lexicographical order.  
We can also enumerate all the functions of  $F_1$ ,  $f_{a_1}, f_{a_2}, \dots, f_{a_k}, \dots$

$f_{a_1}$	1	$\varepsilon$	00	0
$f_{a_2}$	0	$\perp$	1	$\varepsilon$
$f_{a_3}$	0	11	0	11
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$f_{a_k}$	$\varepsilon$	$\varepsilon$	1	0

values of  $f_{a_i}$

$a_1$	$a_2$	$a_3$	$\dots$	$a_k$
0	$\varepsilon$	00	$\dots$	00
$\varepsilon$	00	$\dots$	$\dots$	00
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\varepsilon$	$\varepsilon$	1	$\dots$	0

values of  $\text{diag}(a_i)$

$\text{diag}(a_i) = w \neq 0$ , if the value  $w$  of  $(f_{a_i}, a_i)$  is not undefined  $\perp$ .  
 $\varepsilon$ , otherwise

7/13

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。  
 $\downarrow$   $\text{diag}(a_i) \neq f_{-a_i}(a_i)$   
 $\text{diag} \notin F_1$   
 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。

7/13

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  $f_1(x) \neq f_2(x)$ )  
 $\downarrow$   
 $\text{diag} \notin F_1$   
 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  $G$ .

8/13

**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、 $Z = \{0, 1, -1, 2, -2, 3, -3, \dots\}$  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。**

8/13

**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  $Z = \{0, 1, -1, 2, -2, 3, -3, \dots\}$ .  
**Ex.3.** The set  $R$  of all rational numbers is enumerable infinite. (Why?)

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

9/13

**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:

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

$0.a_{k1}a_{k2}a_{k3}\dots$  ただし、 $a_{ij} \in \{0, 1, \dots, 9\}$   
 上の並びで対角線上にある数に注目し、新たな無限小数  
 $x = 0.b_1b_2b_3\dots$   
 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。

9/13

**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:

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

$0.a_{k1}a_{k2}a_{k3}\dots$  where  $a_{ij} \in \{0, 1, \dots, 9\}$   
 Define a new real number  $x$  by collecting those digits in the diagonal  
 $x = 0.b_1b_2b_3\dots$   
 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.

10/13

**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,  $\neg \text{Halt}(a_i, a_i)$   
 $\therefore f_X(a_i) = 0$

$f_{a_i}(a_i) \neq \perp$  のとき,  $\text{Halt}(a_i, a_i)$   
 $\therefore f_X(a_i) = \perp$

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

10/13

**Ex.2.17 Program X used in the proof of incomputability 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  $\neg \text{Halt}(a_i, a_i)$   
 $\therefore f_X(a_i) = 0$

if  $f_{a_i}(a_i) \neq \perp$  then,  $\text{Halt}(a_i, a_i)$   
 $\therefore f_X(a_i) = \perp$

That is, there is no function  $f_{a_i}$  in the set  $F_1$  of functions  
 such that  $f_X = f_{a_i}$ .

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

11/13

**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?

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

**Zeroは計算不可能.**

```

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

**これは計算可能**

11/13

**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_{ax}(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,  
 $\text{HaltInTime}(a, x, t)$  **Computable!!**  
 $\Leftrightarrow [\text{IsProgram}(a) \wedge \lfloor a \rfloor(x) \text{ stops within } t \text{ steps}]$

12/13

すべての  $a, x \in \Sigma^*$  で

$\text{Halt}(a, x) \Leftrightarrow \exists t [\text{HaltInTime}(a, x, t)]$   
 $\Leftrightarrow \exists w [X_{ax}(w) \text{が} 1 \text{を出力}]$   
 $\Leftrightarrow \neg \text{Zero}(\lfloor X_{ax} \rfloor)$ .

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

(1)  $a$  と  $x$  からプログラム  $\lfloor X_{ax} \rfloor$  のコードを求める.

(2)  $\text{Zero}(\lfloor X_{ax} \rfloor)$  を判定する.

(3)  $\neg \text{Zero}(\lfloor X_{ax} \rfloor) \Rightarrow \text{Halt}(a, x)$   
 $\text{Zero}(\lfloor X_{ax} \rfloor) \Rightarrow \neg \text{Halt}(a, x)$

```

prog X_{ax}(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  $\text{code}(a, x) \equiv \lfloor X_{ax} \rfloor$  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

12/13

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

$\text{Halt}(a, x) \Leftrightarrow \exists t [\text{HaltInTime}(a, x, t)]$   
 $\Leftrightarrow \exists w [X_{ax}(w) \text{ outputs } 1]$   
 $\Leftrightarrow \neg \text{Zero}(\lfloor X_{ax} \rfloor)$ .

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

(1) Given  $a$  and  $x$ , obtain a code of a program  $\lfloor X_{ax} \rfloor$ .

(2) Check  $\text{Zero}(\lfloor X_{ax} \rfloor)$ .

(3)  $\neg \text{Zero}(\lfloor X_{ax} \rfloor) \Rightarrow \text{Halt}(a, x)$   
 $\text{Zero}(\lfloor X_{ax} \rfloor) \Rightarrow \neg \text{Halt}(a, x)$

Here, the function  $\text{code}(a, x) \equiv \lfloor X_{ax} \rfloor$  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?

13/13

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

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 ≠ 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

$$\text{replace}(a) = b, \text{ IsProgram}(a) \text{ のとき,} \\ = a, \text{ その他のとき.}$$

ただし、bは「a」を上のように変換したプログラムのコード

一方、 $\forall a \in \Sigma^* [\text{Zero}(a) \Leftrightarrow \text{Total}(\text{replace}(a))]$

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?

13/13

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

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 ≠ 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

$$\text{replace}(a) = b, \text{ if IsProgram}(a), \\ = a, \text{ otherwise,}$$

where b is a code「a」 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.