1/16

## 3. 計算可能性の分析

3.1. 関数から集合へ
関数の難しさ→集合の難しさ
⇧
構造的解析

集合$L \subseteq \Sigma^*$の認識問題または決定問題(recognition/decision problem)
⇔与えられた文字列 $x$ が $L$ に属するかどうかを判定
$L$の特徴述語 $R_L(x) \leftrightarrow x \in L$

例3.1: EVEN = {$\lceil n \rceil$: $n$ は偶数}, EQ = {$\langle a, b \rangle$: $a=b$}
・EQの認識問題:
「与えられた文字列が$\langle a,b \rangle$という形をしていて，かつ$a=b$か？」
「2つの文字列は等しいか？」
正式には，Eq($x$) ⟷ $\exists a,b[x=\langle a,b \rangle \wedge a=b]$
直観的には， Eq'($a$, $b$)⟷$[a=b]$
EqとEq'の難しさには殆ど差がないので，同じと思ってよい．

1/16

## 3. Analysis of Computability

**3.1. From Functions to Sets**
**Hardness of a function→hardness of a set**
⇧
**structural analysis**

Problem of recognizing a set（or decision problem）
$L \subseteq \Sigma^*$, recognition problem of a set $L$
Given a string $x$, decide whether $x$ belongs to $L$ or not.
Characteristic predicate of $L$ RL($x$) ↔$x \in L$
Ex.3.1 EVEN = {$\lceil n \rceil$: $n$ is even}, EQ = {$\langle a, b \rangle$: $a=b$}
・Recognition of EQ: “Given a string, is it of the form $\langle a,b \rangle$ and $a=b$?” or “Are two strings equal to each other?”
Formally, Eq($x$) ⟷$\exists a,b[x=\langle a,b \rangle \wedge a=b]$
Intuitively Eq'($a$, $b$)⟷$[a=b]$
There is little difference in hardness between Eq and Eq',
so they are considered the same.

2/16

以下では，
集合の認識問題の難しさ → 集合の難しさ

第1章： 問題＝関数の計算問題
第2章： 問題＝$\Sigma^*$上の関数の計算問題
第3章： 問題＝$\Sigma^*$上の集合の認識問題

Yes/Noタイプのプログラムが存在する

定義3.1. $\Sigma^*$上の集合は，その特徴述語が計算可能であるとき帰納的(recursive)であるという．

例3.2. HALT≡ {$\langle a,x \rangle$: Halt($a$, $x$)}
HALTが帰納的⟷ Haltが計算可能
よって，HALTは帰納的でない．

帰納的集合
計算可能集合
認識可能集合
決定可能集合

2/16

Hereafter,
hardness of recognition problem of a set → hardness of a set

Chapter 1： Problem ＝Problem of computing a function
Chapter 2： Problem ＝Problem of computing a function on $\Sigma^*$
Chapter 3： Problem＝Problem of recognizing a set on $\Sigma^*$

Def. 3.1: A set $S$ is ***recursive*** if its characteristic predicate is computable.

Ex. 3.2. HALT ≡{$\langle a,x \rangle$: Halt($a$, $x$)}
HALT is recursive ⟷Halt is computable
Thus, HALT is not recursive.

3/16

定理3.1. 与えられた$\Sigma^*$上の $n$ 引数関数 $f$ に対し、
BIT-$f$ ≡ {$\langle x_1,\ldots,x_n,\lceil i \rceil,d \rangle$ : $f(x_1,\ldots,x_n)$の $i$ bit 目が $d$}
このとき、
BIT-$f$ が帰納的 ⇔ $f$ は計算可能

関数を計算する問題と
集合を認識する問題とは
本質的に同じ

3/16

Theorem 3.1. Given function $f$ with $n$ parameters over $\Sigma^*$,
BIT-$f$ ≡ {$\langle x_1,\ldots,x_n,\lceil i \rceil,d \rangle$ : the $i$-th bit of $f(x_1,\ldots,x_n)$ is $d$}
Then
BIT-$f$ is recursive ⇔ $f$ is computable

4/16

## 2通りの問題記述方法

| 文字列等価性判定問題(EQ) | 与えられた$x$に対し |
|---|---|
| 入力：Σ上の文字列の組$<a,b>$ | "$x \in$ EQ?"を判定する問題 |
| 質問：$a=b$? | ただし，EQ=$\{<a,b>:a=b\}$ |
| 直観的 | 正式 |

認識プログラム＝ $\Sigma^*$型の入力変数をもつ1入力のプログラムで
どんな入力に対しても1または0を出力するもの．

認識プログラム$A$に対して，
- $A$が入力$x$を受理(accept) $\Leftrightarrow$ $A(x)$が1を出力する
  記法：$A(x)$ = accept
- $A$が入力$x$を却下(reject) $\Leftrightarrow$ $A(x)$が0を出力する
  記法：$A(x)$ = reject
- $A$が集合$L$を認識(recognize) $\Leftrightarrow$ $L=\{x: A(x)=\text{accept}\}$
- $L$が帰納的 $\Leftrightarrow$ **$L$を認識するプログラムがある**

4/16

## Two different ways of problem descriptions

| String equivalence(EQ) | Given an $x$, determine whether |
|---|---|
| Input: pair $<a, b>$ of strings on Σ | "$x \in$ EQ?" |
| Question: $a=b$? | where, EQ=$\{<a,b>:a=b\}$ |
| Intuitive | Formal |

Recognition program＝ a program of one input on $\Sigma^*$
which outputs 1 or 0 for any input.

For a recognition problem $A$
- $A$ accepts an input $x$ $\Leftrightarrow$ $A(x)$ outputs 1
  notation: $A(x)$ = accept
- $A$ rejects an input $x$ $\Leftrightarrow$ $A(x)$ outputs 0
  notation: $A(x)$ = reject
- $A$ recognizes a set $L$ $\Leftrightarrow$ $L=\{x: A(x)=\text{accept}\}$
- $L$ is ***recursive*** $\Leftrightarrow$ **There is a program recognizing $L$**

5/16

例3.3. 次に示すのは集合EQを認識するプログラム．

```
prog Eq(input <a, b>);
begin
   if a=b then accept else reject end-if
end.
```
(halt(1): accept, halt(0): reject)

input $<a,b>$: 直観的には文字列の対$<u,v>$を入力と考えていることを表す．
正確には「$x$が入力されると，それが $x=<u, v>$ の形になっているか調べ，そうならば変数 $a,b$ に$u,v$を代入して実行をはじめる．」

例3.4. $L$: 有限集合 $L=\{a_1, a_2, \ldots, a_n\}$.
このとき，次の自明なプログラムで $L, \overline{L}$ の認識が可能．

```
prog L(input x);
begin
   case x of
   a1: accept;
    :
   an: accept
   end-case;
   reject
end.
```

```
prog NotL(input x);
begin
   case x of
   a1: reject;
    :
   an: reject
   end-case;
   accept
end.
```

5/16

Ex.3.3 The following program recognizes the set EQ.

```
prog Eq(input <a, b>);
begin
   if a=b then accept else reject end-if
end.
```
(halt(1): accept, halt(0): reject)

input $<a,b>$: intuitively implies that input is a pair of strings $<u, v>$.
Formally, "if $x$ is input, first check whether it is of the form $x = <u, v>$ and if so start the execution after substituting $u$ and $v$ into $a$ and $b$."

Ex.3.4. $L$: finite set $L=\{a_1, a_2, \ldots, a_n\}$.
Then, the following simple program can recognize $L$ and $\overline{L}$

```
prog L(input x);
begin
   case x of
   a1: accept;
    :
   an: accept
   end-case;
   reject
end.
```

```
prog NotL(input x);
begin
   case x of
   a1: reject;
    :
   an: reject
   end-case;
   accept
end.
```

6/16

## 3.2. 枚挙可能集合

Yes/Noタイプのプログラム

帰納的でない集合を認識するプログラムは存在しない．
but 弱い意味での"認識"を考えると話は別

プログラムAが集合$L$を半認識する
すべての $x \in \Sigma^*$ で
$x \in L$ ↔ A($x$)=accept
$x \notin L$ ↔ A($x$)= ⊥ (A($x$)が停止しない)

集合$L$は半帰納的 ↔ 集合$L$を半認識するプログラムが存在

帰納的集合 $\subsetneq$ 半帰納的集合

i.e., 認識可能な集合 $\subsetneq$ 半認識可能な集合

6/16

## 3.2 Enumerable set

There is no program for recognizing a non-recursive set,
but we have a different story if we consider weak "recognition"

Program A ***semi-recognizes*** a set $L$
for every $x \in \Sigma^*$
$x \in L$ ↔ A($x$)=accept
$x \notin L$ ↔ A($x$)= ⊥ (A($x$) does not stop)

A set $L$ is semi-recursive ↔ semi-recognizing program of a set $L$

Recursive sets $\subsetneq$ semi-recursive sets

i.e., recognizable sets $\subsetneq$ semi-recognizable sets

7/16

例3.5. Haltは帰納的ではないが，半帰納的

（方針：プログラムとそれへの入力が与えられたとき，その
停止性を調べるために実際に実行してみる．
停止する場合には有限時間内に判明する．）

```
prog HALT(input <a, x>);
var t: num;
begin
  t:=0;
  while true do
    if HaltInTime(a, x, t) then accept end-if;
    t:=t+1;
  end-ehile
end.
```

文字列 $a$ が表すプログラムに $x$ を入力すると $t$ ステップ以内に停止する

上記のプログラムはHALTを半認識する．

もし Halt($a$,$x$) なら、あるステップ数 $t$ が存在して、
HALT(<a,x>) は HaltInTime(a,x,t)を実行した時点で accept.

7/16

**Ex.3.5. Halt is not recursive but it is semi-recursive**

(Strategy：given a program and an input to it, execute it to
determine whether its stops or not.
If it stops, we know it within finite time.)

```
prog HALT(input <a, x>);
var t: num;
begin
  t:=0;
  while true do
    if HaltInTime(a, x, t) then accept end-if;
    t:=t+1;
  end-ehile
end.
```

when $x$ is input to a program represented by a string $a$, it halts within $t$ steps.

This program semi-recognizes HALT,
since HALT(<a,x>) will accept for some t
when the program execute HaltInTime(a,x,t) if HALT($a$,$x$).

8/16

(準備) $RANGE(g)$: 関数 $g$ の値域．関数 $g$ を計算するプログラムの出力の集合

定理3.2. 集合$L$を空でない任意の集合とする．このとき，次の2条件は同値である：
(a) $L$は半帰納的
(b) $L = RANGE(g)$となるような計算可能関数 $g$ が存在する．

(a)→(b)の証明
$L$は半帰納的 ➡ $L$を半認識するプログラム $A_L$が存在
$c_1$: $L$の任意の要素 ($L \neq \Phi$より存在)

$A_L$, $c_1$ より、プログラムGを構成

```
prog G(input w: Σ*): Σ*;
var x: Σ* ; t: num;
begin
  if w ∉ Σ*× N then halt(c1) end-if;
  x:=1st(w); t:=2nd(w);
  if HaltInTime(⌈A_L⌉, x, t) then halt(x)
                             else halt(c1) end-if
end.
```

プログラムGが計算する関数を $g$ とし，$g$ が(b)を満たすことを示す．

8/16

(Note) $RANGE(g)$: range of a function $g$, i.e.,
set of all outputs of a program computing a function $g$

**Theorem3.2** Let $L$ be an arbitrary non-empty set. Then, the following two conditions are equivalent:
(a) $L$ is semi-recursive.
(b) There is a computable function $g$ such that $L = RANGE(g)$.

Proof: (a)→(b)
$L$ is semi-recursive ➡ a program ($A_L$) that semi-recognizes $L$ exists
$c_1$: any element of $L$

```
prog G(input w: Σ*): Σ*;
var x: Σ* ; t: num;
begin
  if w ∉ Σ*× N then halt(c1) end-if;
  x:=1st(w); t:=2nd(w);
  if HaltIn Time(⌈A_L⌉, x, t) then halt(x) else halt(c1) end-if
end.
```

Let $g$ be a function computed by this program.
Then, we prove that $g$ satisfies (b) as follows:

9/16

プログラムG: 入力<$x$,$t$> に対して HaltInTime(⌈$A_L$⌉, $x$, $t$)ならhalt($x$)、
それ以外なら halt($c_1$) を実行する

(1) $g$ は計算可能で全域的

(2) すべての $x \in L$ は $A_L$ で受理されるから
$L(x)$=accept → $\exists t \in N$ [ HaltInTime(⌈$A_L$⌉, $x$, $t$)]
→ $\exists t \in N$ [Gは<$x$,$t$>に対して $x$ を出力]
よって、$L$のすべての要素は G の出力として現れる。
つまり $L \subseteq RANGE(g)$.

(3) 一方，どんな $y \notin L$ に対してもA$_L$は停止しないから，
$L(y) = \perp$ → $\forall t \in N$ [¬HaltInTime(⌈$A_L$⌉, $y$, $t$)]
→ $\forall t \in N$ [G は入力<$y$, t>に対して $c_1$ を出力]
つまり，$\overline{L}$のどの要素 $y$ もGの出力として現れない．よって
$\overline{L} \subseteq \overline{RANGE(g)}$ つまり $RANGE(g) \subseteq L$

⇒(2),(3)より $L = RANGE(g)$

9/16

・$g$ is computable and total
・Since any $x \in L$ is accepted by $A_L$,
$L(x)$ = accept → $\exists t \in N$ [HaltInTime(⌈$A_L$⌉, x, t);
→ $\exists t \in N$ [G outputs x for an input <x, t>]
→ $\exists w$ (=<x, t>) $\in \Sigma^*$ [g(w) = x]
that is, $L \subseteq RANGE(g)$
every element of $L$ appears as an output of G.
・On the other hand, since $L$ does not halt for any $y \notin L$,
$L(y) = \perp$ → $\forall t \in N$ [¬HaltInTime(⌈$A_L$⌉, y t)]
→ $\forall t \in N$ [G outputs $c_1$ for an input <y, t>]
→ $\forall y' \in \Sigma^*, \forall t \in N$ [g(<$y'$, t>) $\neq y$]
$y \notin L$, $c_1 \in L$ thus $y \neq c_1$
→ $\forall w \in \Sigma^*$ [$g(w) \neq y$]
that is, no element $y$ of $\overline{L}$ appears as an output of G.
$\overline{L} \subseteq \overline{RANGE(g)}$
$L \subseteq RANGE(g) \wedge \overline{L} \subseteq \overline{RANGE(g)}$
$L = RANGE(g)$

10/16

**(b)→(a)の証明:** $L = RANGE(g)$ となるような計算可能関数 $g$ が存在するなら $L$ は半帰納的

$g$ は計算可能 ⇒ $g$ を計算するプログラム G が存在
G を用いて次のプログラムBを作る.

```
prog B(input x);
var w: Σ*;
begin
  w:= ε ;
  while true do
    if G(w) = x then accept end-if;
    w:=next(w)
  end-while
end.
```

next は長さ優先辞書式順序で次の元を求める関数

- $\Sigma^*$ の元をすべて列挙して調べている.
- $G(w)=x$となる $w \in \Sigma^*$ が存在すれば, $B(x)$=accept
  (存在しなければ停止しない)

よってプログラムBは$L$を半認識する.

（証明終）

10/16

**Proof: (b)→(a)**

i.e., there is a computable function g such that $L$ = RANGE($g$) → $L$ is semi-recursive

$g$ is computable ⇒ there is a program G that computes $g$.
Using this, we have the following program B.

```
prog B(input x);
var w: ;
begin
  w:= ε ;
  while true do
    if G(w) = x then accept end-if;
    w:=next(w)
  end-while
end.
```

**next is a function that computes the next element in the pseudo-lexicographic order**

all the elements of $\Sigma^*$ are checked in order.
if there is a $w \in \Sigma^*$ such that $G(w)=x$, then $x \in L$.
The above program semi-recognizes $L$.

（Q.E.D.）

11/16

**定理3.3.** 任意の無限集合$L$に対し, 次の2条件は同値.
*(a)* $L$は半帰納的
*(b)* $L=RANGE(e)$となるような計算可能で1対1の関数 $e$ が存在する.

⇕ cf

**定理3.2.** 集合$L$を空でない任意の集合とする. このとき, 次の2条件は同値.
*(a)* $L$は半帰納的
*(b)* $L=RANGE(g)$となるような計算可能関数 $g$ が存在する.

定理3.3の証明は省略

11/16

**Theorem3.3.** For any infinite set $L$, the following two conditions are equivalent:
*(a)* $L$ is semi-recursive.
(b) There is a computable one-to-one function $e$ such that $L=RANGE(e)$.

⇕ cf

**Theorem3.2** Let $L$ be an arbitrary non-empty set. Then, the following two conditions are equivalent:
*(a)* $L$ is semi-recursive.
(b) There is a computable function $g$ such that $L=RANGE(g)$.

Proof of Theorem 3.3 is omitted.

12/16

定理3.3
➔ 半帰納的集合 $L$ には
$L=\{e(\varepsilon), e(0), e(1), e(00), e(01), e(10), e(11), e(000), ...\}$
となるような1対1の計算可能関数 $e$ が存在する.

関数 $e$ は $L$ を枚挙 (enumerate)する

定義3.2. 集合 $L$ は次のいずれかが成り立つとき, (帰納的に)
枚挙可能であるという(recursively enumerable).
(a) $L$ は有限集合
(b) $L$ を枚挙する関数で計算可能なものが存在.

注:有限集合 $L$ に対しては $L = RANGE(e)$となるような1対1の全域関数 $e$ などあり得ないので, 例外的に扱っている.

定理3.4 すべての集合 $L$ に対し,
$L$ が半帰納的 ⟺ $L$が枚挙可能

12/16

Theorem3.3
➔ for a semi-recursive set $L$ there exists a computable one-to-one function such that
$L=\{e(\varepsilon), e(0), e(1), e(00), e(01), e(10), e(11), e(000), ...\}$
We say the function $e$ ***enumerates*** $L$.

**Def.3.2** A set $L$ is **(recursively) enumerable** if
(a) $L$ is a finite set, or
(b) there is a computable function that enumerates $L$.

Remark: Finite sets are exceptional, since for any finite set $L$ there is no total on-to-one function e such that $L$ = RANGE($e$).

**Theorem3.4** For any set $L$ we have
$L$ is semi-recursive ⟺ $L$ is enumerable

13/16

## 枚挙可能性と帰納性の比較

$A$: 帰納的集合
- ✓ $A$の特徴述語 $R_A(x)$ が計算可能.
- ✓ $x \in \Sigma^*$に対し、$x \in A$かどうか判定可能
- ✓どんな入力 $x \in \Sigma^*$に対しても、
  いつも停止してYes/Noを答えてくれるプログラムが存在

$B$: 枚挙可能集合
- ✓ $B$ を枚挙する関数が計算可能
- ✓すべての$B$の要素を順番に出力するプログラムが存在

13/16

## Comparison between enumerability and recursiveness

$A$: recursive set ⇒ the characteristic predicate $R_A(x)$ is computable
That is, for $x \in \Sigma^*$ it is computable whether $x \in A$

$B$: enumerable set ⇒ a function that enumerates $B$ is computable
that is, we can enumerate all the elements of $B$

14/16

定理3.5. すべての集合 $L$ に対し，次の条件は同値
(a) $L$ は枚挙可能.
(b) 適当な計算可能述語 $R$ に対し，$L=\{x: \exists w \in \Sigma^* [R(x, w)]\}$

(a)→(b)の証明
$L$ は枚挙可能だから，$L$ を枚挙する計算可能関数 $e$ が存在する.
$R(x,w) \equiv [e(w) = x]$と定義
$e$ が $L$ の枚挙関数なので，
$L = \{x: \exists w \in \Sigma^* [e(w) = x]\}$
$= \{x: \exists w \in \Sigma^* [R(x,w)]\}$
$e$ は計算可能関数⟶ $e$ を計算するプログラムが存在
しかも $e$ は全域的なので，そのプログラムは必ず停止して答を出力
よって，述語 $R$ は計算可能

14/16

**Theorem3.5.** For any set $L$, the following conditions are equivalent.
(a) $L$ is enumerable.
(b) For some computable predicate $R$, we have
$L=\{x: \exists w \in \Sigma^* [R(x, w)]\}$

Proof : (a)→(b)
$L$ is enumerable, so there is a computable function e enumerating $L$.
Define $R(x,w) \equiv [e(w) = x]$
Since $e$ is a function enumerating $L$,
$L = \{x: \exists w \in \Sigma^* [e(w) = x]\}$
$= \{x: \exists w \in \Sigma^* [R(x,w)]\}$
$e$ is computable ⟶ there is a program that computes $e$
Moreover, $e$ is total, and thus the program always stops and outputs an answer. Thus, the predicate $R$ is computable.

15/16

定理3.5. すべての集合 $L$ に対し，次の条件は同値
(a) $L$ は枚挙可能.
(b) 適当な計算可能述語 $R$ に対し，$L=\{x: \exists w \in \Sigma^* [R(x, w)]\}$

(b)→(a)の証明
条件(b)を満たす述語を計算する関数 R(x,w) を使って，
$L$ を半認識するプログラムCが作れる.

```
prog C(input x);
var w: Σ*;
begin
  w:=ε;
  while true do
    if R(x, w) then accept end-if;
    w:=next(w)
  end-while
end.
```

したがって，$L$ は半帰納的，つまり枚挙可能.
証明終

15/16

**Theorem3.5** For any set $L$, the following conditions are equivalent.
(a) $L$ is enumerable.
(b) For some computable predicate $R$, $L=\{x: \exists w \in \Sigma^* [R(x, w)]\}$

Proof: (b)→(a)
Using a program that computes a predicate satisfying the condition (b), we have a program that semi-recognizes $L$.

```
prog C(input x);
var w: Σ*;
begin
  w:=ε;
  while true do
    if R(x, w) then accept end-if;
    w:=next(w)
  end-while
end.
```

Therefore, $L$ is semi-recursive. That is, it is enumerable.
Q.E.D.

16/16

どんな枚挙可能集合 $L$ にも次の関係を満たす計算可能な述語$R$が存在

「すべての $x \in \Sigma^*$に対し, $x \in L \Longleftrightarrow \exists w \in \Sigma^*[R(x, w)]$. 」

$L$の認識問題を $\exists w[R(x, w)]$ という形の論理式で判定可能.
逆に, そのような形で認識問題を判定できる集合が枚挙可能集合.

$\exists w[Q(x, w)]$ という形の論理式: 枚挙可能集合のための論理式
（RE論理式）

$Q$をこのRE論理式の核(kernel)という.

$L$のRE論理式: 枚挙可能集合$L$に対するRE論理式

$L$のRE論理式が $\exists w[R(x,w)]$ のとき,
各$x \in L$に対し, $R(x, w_x)$となるような$w_x \in \Sigma^*$ が存在する.
この$w_x$を‘$x \in L$’の証拠（witness)と呼ぶ.

16/16

For any enumerable set $L$ there is a computable predicate $R$ satisfying
"for any $x \in \Sigma^*$, we have $x \in L \Longleftrightarrow \exists w \in \Sigma^*[R(x, w)]$.

The problem of recognizing $L$ can be determined by the predicate of the form $\exists w[R(x, w)]$.
Conversely, sets whose recognition problem can be determined in this way are enumerable sets.

predicate of the form $\exists w[Q(x, w)]$: predicate for enumerable sets
（RE predicate）
$Q$ is a kernel of the RE predicate.
RE predicate for $L$: the RE predicate for an enumerable set $L$

If the RE predicate of $L$ is $\exists w[R(x,w)]$,
for each $x \in L$ there is $w_x \in \Sigma^*$ such that $R(x, w_x)$ is true.
Such $w_x$ is called a **witness** for ‘$x \in L$’