

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)
 \Leftrightarrow 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: $\text{EVEN} = \{\lceil n \rceil : n \text{ は偶数}\}$, $\text{EQ} = \{\langle a, b \rangle : a=b\}$
 • 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, $\text{Eq}(x) \leftrightarrow \exists a, b [x = \langle a, b \rangle \wedge a=b]$
 直観的には, $\text{Eq}(\langle a, b \rangle) \leftrightarrow [a=b]$
 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 \rightarrow hardness of a set
 \uparrow
 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 $R_L(x) \Leftrightarrow x \in L$

Ex.3.1 $\text{EVEN} = \{\lceil n \rceil : n \text{ is even}\}$, $\text{EQ} = \{\langle a, b \rangle : a=b\}$
 • 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, $\text{Eq}(x) \leftrightarrow \exists a, b [x = \langle a, b \rangle \wedge a=b]$
 Intuitively $\text{Eq}(\langle a, b \rangle) \leftrightarrow [a=b]$
 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 \rightarrow 集合の難しさ

第1章: 問題 = 関数の計算問題
 第2章: 問題 = Σ^* 上の関数の計算問題
 第3章: 問題 = Σ^* 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. Σ^* 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. $\text{HALT} \equiv \{\langle a, x \rangle : \text{Halt}(a, x)\}$
 HALTが帰納的 \leftrightarrow 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 \rightarrow hardness of a set

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

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

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

3/16

定理3.1. 与えられた Σ^* 上の n 引数関数 f に対し、
 $\text{BIT-}f \equiv \{\langle x_1, \dots, x_n, \lceil i \rceil, d \rangle : f(x_1, \dots, x_n) \text{ の } i \text{ bit 目が } d\}$
 このとき、
 $\text{BIT-}f$ が帰納的 $\Leftrightarrow f$ 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 Σ^* ,
 $\text{BIT-}f \equiv \{\langle x_1, \dots, x_n, \lceil i \rceil, d \rangle : \text{the } i\text{-th bit of } f(x_1, \dots, x_n) \text{ is } d\}$
 Then
 $\text{BIT-}f$ is recursive $\Leftrightarrow f$ is computable

The problem to compute a function
is essentially the same as
the problem to recognize a set.

4/16

2通りの問題記述方法

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

認識プログラム = Σ^* 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: $A(x) = \text{accept}$
 A が入力 x を却下(reject) $\Leftrightarrow A(x)$ が0を出力する
 記法: $A(x) = \text{reject}$
 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

<p>String equivalence(EQ) Input: pair $\langle a, b \rangle$ of strings on Σ^* Question: $a=b?$ Intuitive</p>	<p>Given an x, determine whether “$x \in EQ?$” where, $EQ = \{\langle a, b \rangle : a=b\}$ Formal</p>
--	--

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

For a recognition problem A
 A accepts an input $x \Leftrightarrow A(x)$ outputs 1
 notation: $A(x) = \text{accept}$
 A rejects an input $x \Leftrightarrow A(x)$ outputs 0
 notation: $A(x) = \text{reject}$
 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.
  
```

input $\langle a, b \rangle$: 直観的には文字列の対 $\langle u, v \rangle$ 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 $x = \langle u, v \rangle$ 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 $L = \{a_1, a_2, \dots, a_n\}$.
 このとき、次の自明なプログラムで L, \bar{L} の認識が可能.

<pre> prog L(input x); begin case x of a₁: accept; : a_n: accept end-case; reject end. </pre>	<pre> prog NotL(input x); begin case x of a₁: reject; : a_n: reject end-case; accept end. </pre>
--	---

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.
  
```

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

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

<pre> prog L(input x); begin case x of a₁: accept; : a_n: accept end-case; reject end. </pre>	<pre> prog NotL(input x); begin case x of a₁: reject; : a_n: reject end-case; accept end. </pre>
--	---

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
 $x \in L \Leftrightarrow A(x) = \text{accept}$
 $x \notin L \Leftrightarrow A(x) = \perp$ ($A(x)$ が停止しない)

集合 L は半帰納的 \Leftrightarrow 集合 L を半認識するプログラムが存在

<p>帰納的集合 \subsetneq 半帰納的集合</p> <p>i.e., 認識可能な集合 \subsetneq 半認識可能な集合</p>

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 $x \in \Sigma^*$
 $x \in L \Leftrightarrow A(x) = \text{accept}$
 $x \notin L \Leftrightarrow A(x) = \perp$ ($A(x)$ does not stop)

A set L is semi-recursive \Leftrightarrow semi-recognizing program of a set L

<p>Recursive sets \subsetneq semi-recursive sets</p> <p>i.e., recognizable sets \subsetneq semi-recognizable sets</p>

例3.5. Haltは帰納的ではないが、半帰納的 7/16

(方針: プログラムとそれへの入力を与えられたとき、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 $\text{Halt}(a,x)$ なら、あるステップ数 t が存在して、 $\text{HALT}(\langle a,x \rangle)$ は $\text{HaltInTime}(a,x,t)$ を実行した時点で accept.

Ex.3.5. Halt is not recursive but it is semi-recursive 7/16

(Strategy: given a program and an input to it, execute it to determine whether it 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 $\text{HALT}(\langle a,x \rangle)$ will accept for some t when the program execute $\text{HaltInTime}(a,x,t)$ if $\text{HALT}(a,x)$.

(準備) $\text{RANGE}(g)$: 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 8/16

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) $L = \text{RANGE}(g)$ となるような計算可能関数 g が存在する.

(a) \rightarrow (b) の証明
 L は半帰納的 $\Rightarrow L$ 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 ($L \neq \emptyset$ より存在)

```

prog G(input w:  $\Sigma^*$ ;  $\Sigma^*$ ;
var x:  $\Sigma^*$ ; t: num;
begin
  if  $w \notin \Sigma^* \times \mathbb{N}$  then halt( $c_1$ ) end-if;
  x:=1st(w); t:=2nd(w);
  if HaltInTime( $\lceil A_L \rceil$ , x, t) then halt(x)
  else halt( $c_1$ ) end-if
end.

```

A_L, c_1 より、プログラム G を構成

プログラム G が計算する関数を g とし、 $g \models (b)$ を満たすことを示す。

(Note) $\text{RANGE}(g)$: range of a function g , i.e., set of all outputs of a program computing a function g 8/16

Theorem 3.2 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 $L = \text{RANGE}(g)$.

Proof: (a) \rightarrow (b)
 L is semi-recursive \Rightarrow a program (A_L) that semi-recognizes L exists
 c_1 : any element of L

```

prog G(input w:  $\Sigma^*$ ;  $\Sigma^*$ ;
var x:  $\Sigma^*$ ; t: num;
begin
  if  $w \notin \Sigma^* \times \mathbb{N}$  then halt( $c_1$ ) end-if;
  x:=1st(w); t:=2nd(w);
  if HaltInTime( $\lceil A_L \rceil$ , x, t) then halt(x) else halt( $c_1$ ) end-if
end.

```

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

プログラム G : 入力 $\langle x, t \rangle$ に対して $\text{HaltInTime}(\lceil A_L \rceil, x, t)$ なら $\text{halt}(x)$ 、それ以外なら $\text{halt}(c_1)$ を実行する 9/16

(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
 $L(x) = \text{accept} \rightarrow \exists t \in \mathbb{N} [\text{HaltInTime}(\lceil A_L \rceil, x, t)]$
 $\rightarrow \exists t \in \mathbb{N} [G \text{ は } \langle x, t \rangle \text{ に対して } x \text{ を出力}]$
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 $L \subseteq \text{RANGE}(g)$.

(3) 一方、どんな $y \notin L$ に対しても A_L は停止しないから、
 $L(y) = \perp \rightarrow \forall t \in \mathbb{N} [\neg \text{HaltInTime}(\lceil A_L \rceil, y, t)]$
 $\rightarrow \forall t \in \mathbb{N} [G \text{ は } \langle y, t \rangle \text{ に対して } c_1 \text{ を出力}]$
つまり、 L のどの要素 y も G の出力として現れない。よって
 $\bar{L} \subseteq \text{RANGE}(g)$ つまり $\text{RANGE}(g) \subseteq \bar{L}$

$\Rightarrow (2), (3)$ より $L = \text{RANGE}(g)$

g is computable and total 9/16

• Since any $x \in L$ is accepted by A_L ,
 $L(x) = \text{accept} \rightarrow \exists t \in \mathbb{N} [\text{HaltInTime}(\lceil A_L \rceil, x, t)]$
 $\rightarrow \exists t \in \mathbb{N} [G \text{ outputs } x \text{ for an input } \langle x, t \rangle]$
 $\rightarrow \exists w (= \langle x, t \rangle) \in \Sigma^* [g(w) = x]$
that is, $L \subseteq \text{RANGE}(g)$
every element of L appears as an output of G .

• On the other hand, since L does not halt for any $y \notin L$,
 $L(y) = \perp \rightarrow \forall t \in \mathbb{N} [\neg \text{HaltInTime}(\lceil A_L \rceil, y, t)]$
 $\rightarrow \forall t \in \mathbb{N} [G \text{ outputs } c_1 \text{ for an input } \langle y, t \rangle]$
 $\rightarrow \forall y' \in \Sigma^*, \forall t \in \mathbb{N} [g(\langle y', t \rangle) \neq y]$
 $y \notin L, c_1 \in L$ thus $y \neq c_1$
 $\rightarrow \forall w \in \Sigma^* [g(w) \neq y]$
that is, no element y of L appears as an output of G .
 $\bar{L} \subseteq \text{RANGE}(g)$
 $L \subseteq \text{RANGE}(g) \wedge \bar{L} \subseteq \text{RANGE}(g)$
 $L = \text{RANGE}(g)$

10/16

(b)→(a)の証明: $L = \text{RANGE}(g)$ 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 $\Rightarrow g$ 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:  $\Sigma^*$ ;
begin
w :=  $\epsilon$  ;
while true do
if  $G(w) = x$  then accept end-if;
w := next(w)
end-while
end.

```

\rightarrow next は長さ優先辞書式順序で次の元を求める関数

$\triangleright \Sigma^*$ の元をすべて列挙して調べている.
 $\triangleright G(w)=x$ となる $w \in \Sigma^*$ が存在すれば, $B(x)=\text{accept}$ (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 $L = \text{RANGE}(g)$
 $\rightarrow L$ is semi-recursive

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

```

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

```

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

all the elements of Σ^* 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) $L = \text{RANGE}(e)$ 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) $L = \text{RANGE}(g)$ 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

Theorem 3.3. 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 $L = \text{RANGE}(e)$.

↕ cf

Theorem 3.2 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 $L = \text{RANGE}(g)$.

Proof of Theorem 3.3 is omitted.

12/16

定理3.3
 \rightarrow 半帰納的集合 L には
 $L = \{e(\epsilon), e(0), e(1), e(00), e(01), e(10), e(11), e(000), \dots\}$
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 $L = \text{RANGE}(e)$ 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 $\iff L$ が枚挙可能

12/16

Theorem 3.3
 \rightarrow for a semi-recursive set L there exists a computable one-to-one function such that
 $L = \{e(\epsilon), e(0), e(1), e(00), e(01), e(10), e(11), e(000), \dots\}$
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 $L = \text{RANGE}(e)$.

Theorem 3.4 For any set L we have
 L is semi-recursive $\iff L$ 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 \iff the characteristic predicate $R_A(x)$ is computable
 That is, for $x \in \Sigma^*$ it is computable whether $x \in A$

B: enumerable set \iff 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) \rightarrow (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 $\rightarrow e$ 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

Theorem 3.5. 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) \rightarrow (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 \rightarrow 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) \rightarrow (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:  $\Sigma^*$ ;
begin
  w :=  $\epsilon$ ;
  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

Theorem 3.5 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) \rightarrow (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:  $\Sigma^*$ ;
begin
  w :=  $\epsilon$ ;
  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, $x \in L \iff \exists w \in \Sigma^*[R(x, w)].$ 」

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
(\mathcal{RE} 論理式)

Q をこの \mathcal{RE} 論理式の核(kernel)という.

L の \mathcal{RE} 論理式: 枚挙可能集合 L に対する \mathcal{RE} 論理式

L の \mathcal{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 $x \in L \iff \exists w \in \Sigma^*[R(x, w)].$ ”

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
(\mathcal{RE} predicate)

Q is a kernel of the \mathcal{RE} predicate.

\mathcal{RE} predicate for L : the \mathcal{RE} predicate for an enumerable set L

If the \mathcal{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 ' $x \in L$ '