

1/14

3. 計算可能性の分析

3.1. 関数から集合へ
3.2. 枚挙可能集合

- 集合 L が **帰納的** $\Leftrightarrow L$ を次の意味で認識するプログラム A が存在する:
 $x \in L$ なら $A(x) = \text{accept}$
 $y \notin L$ なら $A(y) = \text{reject}$
- 集合 L が **半帰納的** $\Leftrightarrow L$ を次の意味で認識するプログラム A が存在する:
 $x \in L$ なら $A(x) = \text{accept}$
 $y \notin L$ なら $A(y) = \perp$

無限ループ

1/14

3. Analysis of Computability

3.1. From Functions to Sets
3.2. Enumerable sets

- A set L is **recursive** \Leftrightarrow There is a program A that recognizes L in the following sense:
 $x \in L$ implies $A(x) = \text{accept}$
 $y \notin L$ implies $A(y) = \text{reject}$
- A set L is **semi-recursive** \Leftrightarrow There is a program A that recognizes L in the following sense:
 $x \in L$ implies $A(x) = \text{accept}$
 $y \notin L$ implies $A(y) = \perp$

A(y) does not stop

2/14

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。

→ 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。

$e(\epsilon), e(0), e(1), e(00), e(01), e(10), e(11), e(000), \dots$
 は L の要素を「漏れ」なく「重複」なく列挙する。

2/14

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

→ 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 .

We can enumerate all elements in L as
 $e(\epsilon), e(0), e(1), e(00), e(01), e(10), e(11), e(000), \dots$
 without skip and duplicate.

3/14

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 $\Leftrightarrow L$ が枚挙可能

3/14

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 $\Leftrightarrow L$ is enumerable

4/14

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

4/14

Comparison between enumerability and recursiveness

A: recursive set

- ✓ The characteristic predicate $R_A(x)$ is computable
- ✓ For $x \in \Sigma^*$ it is computable whether $x \in A$
- ✓ There is a program that always outputs Yes/No for any $x \in \Sigma^*$

B: enumerable set

- ✓ A function that enumerates B is computable
- ✓ We can enumerate all the elements of B

5/14

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
 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

5/14

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.

6/14

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

6/14

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.

7/14

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
 (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.

7/14

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
 (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 ' $x \in L$ '

8/14

3.3. クラスRECとクラスRE

クラスREC $\equiv \{L: L \text{ は帰納的}\}$: 帰納的集合のクラス

- クラスRECの外側は帰納的でない集合の領域
- 空でないこと程度しか分かっていない(ここまでの議論では)

HALT \notin クラスREC
 ZEROFT \notin クラスREC
 ZEROFTとは, 単純for-timesプログラムが常に0を出力するかどうかを判定する述語を特徴述語とする集合のこと. ただし, for-timesに関する説明は省略した.

目標: RECの外側の領域の構造の解析
 RECの外側で最も扱いやすい集合のクラスは何か?
 → 枚挙可能集合.

8/14

3.3 Class REC and Class RE

Class REC $\equiv \{L: L \text{ is recursive}\}$: a class of recursive sets

- Outside of the class REC is a region for non-recursive sets. It is only known that it is not empty (by the argument so far).

HALT \notin class REC
 ZEROFT \notin class REC
 ZEROFT is a set with characteristic predicate that a simple for-times program always outputs 0.
 (although the explanation for for-times has been omitted.)

GOAL: Analyzing the structure outside REC
 What is the easiest class of sets outside REC?
 → enumerable sets.

9/14

RE $\equiv \{L: L \text{ は枚挙可能}\}$
 co-RE $\equiv \{L: \bar{L} \text{ が枚挙可能}\}$

注: L : 集合
 L が枚挙可能 $\iff L$ が半帰納的
 $\iff L$ を半認識するプログラム A が存在.
 $x \in \Sigma^*, x \in L \iff A(x) = \text{accept}$
 $x \notin L \iff A(x) = \perp$

クラスco-REはクラスREの補クラスREではないことに注意.

例3.8. クラスRE, co-REに入る集合の例.

HALT \in RE, $\overline{\text{HALT}} \in$ co-RE
 $\overline{\text{ZEROFT}} \in$ RE, ZEROFT \in co-RE

9/14

RE $\equiv \{L: L \text{ is enumerable}\}$
 co-RE $\equiv \{L: \bar{L} \text{ is enumerable}\}$

Note: L : set
 L is enumerable $\iff L$ is semi-recursive
 \iff there is a program A that semi-recognizes $L.$
 $x \in \Sigma^*, x \in L \iff A(x) = \text{accept}$
 $x \notin L \iff A(x) = \perp$

Note that the class co-RE is not complementary of the class RE.

Ex.3.8. Examples of sets belonging to class RE and class co-RE.

HALT \in RE, $\overline{\text{HALT}} \in$ co-RE
 $\overline{\text{ZEROFT}} \in$ RE, ZEROFT \in co-RE

10/14

REとco-REは同程度の“難しさ”

A: 任意のRE集合
 $x \in \Sigma^* [(x \in A \rightarrow X(x) = \text{accept}) \wedge (x \notin A \rightarrow X(x) = \perp)]$
 となるプログラムXが作れる

B: 任意のco-RE集合
 $x \in \Sigma^* [(x \in B \rightarrow X(x) = \perp) \wedge (x \notin B \rightarrow X(x) = \text{accept})]$
 となるプログラムXが作れる

上記の2つのプログラムはよく似ており、難しさに差がつけられない。

10/14

RE and co-RE are equally “hard”

A: arbitrary RE set
 we can write a program X such that
 $x \in \Sigma^* [(x \in A \rightarrow X(x) = \text{accept}) \wedge (x \notin A \rightarrow X(x) = \perp)]$

B: arbitrary co-RE set
 we can write a program X such that
 $x \in \Sigma^* [(x \in B \rightarrow X(x) = \perp) \wedge (x \notin B \rightarrow X(x) = \text{accept})]$

The above two programs are similar, and there is no difference.

11/14

定理3.6. すべての集合Lに対し、次の関係が成り立つ。
 (1) $L \in \text{REC} \leftrightarrow \bar{L} \in \text{REC}$
 (2) $L \in \text{RE} \leftrightarrow \bar{L} \in \text{co-RE}$

証明:
 (1) $L \in \text{REC}$ とすると、Lを認識するプログラムがある。
 accept \rightarrow reject, reject \rightarrow accept
 と変更すると、 \bar{L} を認識するプログラムを得る。
 よって、 $\bar{L} \in \text{REC}$

(2)はco-REの定義より明らか。 証明終

11/14

Theorem 3.6. For every set L, the followings hold:
 (1) $L \in \text{REC} \leftrightarrow \bar{L} \in \text{REC}$
 (2) $L \in \text{RE} \leftrightarrow \bar{L} \in \text{co-RE}$

Proof:
 (1) $L \in \text{REC}$, then there is a program that recognizes L.
 If we exchange accept with reject
 accept \rightarrow reject, reject \rightarrow accept
 then, the resulting program recognizes \bar{L} .
 So, $\bar{L} \in \text{REC}$

(2) is obvious from the definition of co-RE. Q.E.D.

12/14

定理3.7. (1) $\text{REC} \subsetneq \text{RE}$ (2) $\text{REC} \subsetneq \text{co-RE}$

証明: 略

12/14

Theorem 3.7. (1) $\text{REC} \subsetneq \text{RE}$ (2) $\text{REC} \subsetneq \text{co-RE}$

Proof: Omitted.

13/14

定理3.8. $REC = RE \cap co-RE$

証明:
 定理3.7より, $REC \subseteq RE \cap co-RE$
 任意の $L \in RE \cap co-RE$ について, $L \in REC$ を示したい.
 仮定より, $L \in RE$ かつ $\bar{L} \in RE$
 $\rightarrow L$ を半認識するプログラム A_1 と
 \bar{L} を半認識するプログラム A_2 が存在.
 このとき, 次のプログラムBは L を認識する.

```

prog B(input x);
var t: num;
begin
  for t:=0 to do
    if HaltInTime([A1], x, t) then accept end-if;
    if HaltInTime([A2], x, t) then reject end-if;
  end-for
end.
  
```

$x \in L$ のとき,
 A_1 が先に停止して
 acceptとなる.
 $x \notin L$ のとき,
 A_2 が先に停止して
 rejectとなる.

証明終

13/14

Theorem 3.8 $REC = RE \cap co-RE$

Proof:
 By Theorem 2.7 we have $REC \subseteq RE \cap co-RE$
 We want to show that $L \in REC$ for any $L \in RE \cap co-RE$.
 By the assumption, $L \in RE$ and $\bar{L} \in RE$
 \rightarrow there are a program A_1 that semi-recognizes L and
 a program A_2 that semi-recognizes \bar{L} .
 Then, the following program B recognizes L .

```

prog B(input x);
var t: num;
begin
  for t:=0 to do
    if HaltInTime([A1], x, t) then accept end-if;
    if HaltInTime([A2], x, t) then reject end-if;
  end-for
end.
  
```

if $x \in L$,
 A_1 stops before A_2
 and accepts x .
 if $x \notin L$,
 A_2 stops before A_1
 and rejects x .

Q.E.D.

14/14

定理3.9. $RE \neq co-RE$

証明:
 $RE = co-RE$ と仮定すると, $RE = RE \cap co-RE$
 定理3.8より, $REC = RE$ となり, 定理3.7に矛盾.
 証明終

14/14

Theorem 3.9. $RE \neq co-RE$

Proof:
 If we assume $RE = co-RE$, we have $RE = RE \cap co-RE$.
 Hence, by Theorem 3.8 we have $REC = RE$, contradicts to
 Theorem 3.7.
 Q.E.D.