3. 計算可能性の分析

3.2. 枚挙可能集合

集合 L は枚挙可能:
1. L は有限集合
2. L を枚挙する計算可能な関数 e が存在する

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

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

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,

$$\begin{aligned} L &= \{x:\ \exists w \in \Sigma^* [e(w) = x]\} \\ &= \{x:\ \exists w \in \Sigma^* [R(x,w)]\} \end{aligned}$$

e is computable $\longrightarrow$ 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.

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

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: ∑*;
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.

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.

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 '$x \in L$'

3.3. クラスRECとクラスRE

クラスREC≡ $\{L: L$ 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？
$\longrightarrow$ 枚挙可能集合.

3.3 Class REC and Class RE

Class REC ≡ {L: L 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?
$\longrightarrow$ enumerable sets.

RE $\equiv \{L: L$ は枚挙可能$\}$

co-RE $\equiv \{L: \overline{L}$ が枚挙可能$\}$

注 : L : 集合

Lが枚挙可能 $\Leftrightarrow L$ が半帰納的

$\Leftrightarrow L$ を半認識するプログラムAが存在.

$$x \in \Sigma^*, \quad x \in L \Leftrightarrow \mathrm{A}(x) = \mathrm{accept}$$

$$x \notin L \Leftrightarrow \mathrm{A}(x) = \perp$$

クラスco-REはクラスREの補クラス$\overline{\text{RE}}$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

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

注 : L : set

L is enumerable $\Leftrightarrow L$ is semi-recursive

$\Leftrightarrow$ there is a program A that semi-recognizes L.

$x \in \Sigma^*$, $x \in L \Leftrightarrow \mathrm{A}(x) = \text{accept}$

$x \notin L \Leftrightarrow \mathrm{A}(x) = \perp$

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.

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

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

REとco-REは同程度の“難しさ”

A: 任意のRE集合

$$x \in \Sigma^*[(x \in A \rightarrow \mathrm{X}(x)=\mathrm{accept}) \wedge (x \notin A \rightarrow \mathrm{X}(x)=\perp)]$$

となるプログラムXが作れる

B: 任意のco-RE集合

$$x \in \Sigma^*[(x \in B \rightarrow \mathrm{X}(x)=\perp) \wedge (x \notin B \rightarrow \mathrm{X}(x)=\mathrm{accept})]$$

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．

RE and co-RE are equally "hard"

A: arbitrary RE set

we can write a program X such that

$$\forall x \in \Sigma^*[(x \in A \longrightarrow \mathrm{X}(x)=\text{accept}) \wedge (x \notin A \longrightarrow \mathrm{X}(x)=\perp)]$$

B: arbitrary co-RE set

we can write a program X such that

$$\forall x \in \Sigma^*[(x \in B \longrightarrow \mathrm{X}(x)=\perp) \wedge (x \notin B \longrightarrow \mathrm{X}(x)=\text{accept})]$$

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

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

証明:
(1) $L \in$ RECとすると, L を認識するプログラムがある.

accept →reject, reject → accept

と変更すると, $\overline{L}$を認識するプログラムを得る.

よって, $\overline{L} \in$ REC

(2)はco-REの定義より明らか.

証明終

Theorem 3.6. For every set L, the followings hold:

(1) $L \in \text{REC} \longleftrightarrow \overline{L} \in \text{REC}$

(2) $L \in \text{RE} \longleftrightarrow \overline{L} \in \text{co-RE}$

Proof:

(1) $L \in \text{REC}$, then there is a program that recognizes L.

If we exchange accept with reject

accept → reject, reject → accept

then, the resulting program recognizes $\overline{L}$.

So, $\overline{L} \in \text{REC}$

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

Q.E.D.

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

証明: 略

$\boxed{\textbf{Theorem 3.7. } (1)\ \mathsf{REC} \subsetneqq \mathsf{RE} \qquad (2)\ \mathsf{REC} \subsetneqq \text{co-}\mathsf{RE}}$

Proof : Omitted.

定理3.8. REC＝RE∩co-RE

証明:

定理3.7より，REC⊆RE∩co-RE

任意の $L \in$ RE∩ co-REについて，$L \in$ RECを示したい．

仮定より，$L \in$ RE かつ $\overline{L} \in$ RE

→L を半認識するプログラムA_1と

$\overline{L}$ 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

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 $\overline{L}$ $\in$ RE

→there are a program A_1 that semi-recognizes L and
a program A_2 that semi-recognizes $\overline{L}$.

Then, the following program B recognizes L.

prog B(input x);
var t: num;
begin
    for t:=0 to   do
        if HaltInTime($\lceil A_1 \rceil$, x, t) then accept end-if;
        if HaltInTime($\lceil A_2 \rceil$, 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.

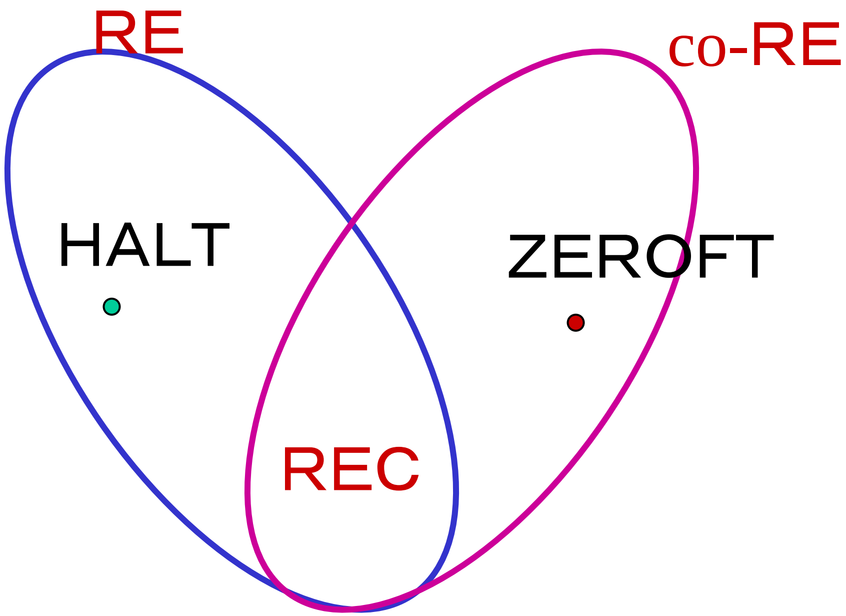
定理3.9. RE $\neq$ co-RE

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



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.