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'. $e(\varepsilon), e(0), e(1), e(00), e(01), e(10), e(11), e(000), \ldots$
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

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: ∑*;
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

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.

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？
⟶ 枚挙可能集合．

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

クラスco-REはクラス$\overline{\mathrm{RE}}$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.

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

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

REとco-REは同程度の“難しさ”

A: 任意のRE集合

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

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．

定理3.6. すべての集合Lに対し，次の関係が成り立つ.

(1) $L \in \mathrm{REC} \leftrightarrow \overline{L} \in \mathrm{REC}$

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

証明:

(1) $L \in \mathrm{REC}$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，$\overline{L} \in \mathrm{REC}$

(2)はco-REの定義より明らか.

証明終

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

証明: 略

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

定理3.9. RE ≠ co-RE

証明:

RE=co-REと仮定すると, RE=RE ∩ co-RE

定理3.8より, REC＝REとなり, 定理3.7に矛盾.

証明終