アナウンス
中間試験：5月10日（木曜日） チュートリアルアワーに大講義室で実施
スライドのコピー，手書きのノート，筆記用具持ち込み可．

I238 計算の理論

上原 隆平

2018年I-1期（4-5月）

I238 Computation Theory

by

Prof. Ryuhei Uehara

Term I-1, April-May, 2018

計算(量)の理論

- ゴール1:
  - “計算可能な*関数/問題/言語/集合*”
    - 関数には2種類存在する;
      1. 計算不能(!)な関数
      2. 計算可能な関数
- ゴール2:
  - 「問題の困難さ」を示す方法を学ぶ
    - 計算可能な問題であっても、*手におえない*場合がある！
      - 計算に必要な資源(時間・領域)が多すぎるとき

Computation Theory/ Computational Complexity

- Goal 1:
  - "*Computable Function/Problem/Language/Set*"
    - We have two functions;
      1. Functions that are not computable!
      2. Functions that are computable.
- Goal 2:
  - How can you show "*Difficulty of Problem*"
    - There are *intractable* problems even if they are computable!
      - because they require too many resources (time/space)!

4. 計算不能性と対角線論法

4. 計算不能な問題

以下の問題を解くチューリングマシンは存在しない:

停止性判定問題HALT (停止するかどうかを決定する問題)
入力:チューリングマシン T と
それへの入力 x を符号化した文字列$<T,x>$
出力: T に入力 x を与えると，停止するか？
Yes: $T(x)$ は(有限時間内に)停止する
No: 停止しない(無限ループ)

正確に言えば，停止性判定問題を解くチューリングマシン U' は存在しない.

…証明は「対角線論法」を用いて行う

4. Undecidability and Diagonalization

4. Undecidable problem

The following problem cannot be solved by any Turing machine:

The problem HALT (Problem of deciding halting)

input : a code <*T*,*x*> of Turing machine *T* and an input *x*

output: *T* will terminates for the input *x*?

Yes: if *T*(*x*) terminates

No: otherwise.

Precisely, we can show that there is no Turing machine *U'* that computes the halting problem.

…Proof is done by "diagonalization" essentially…

2.5 計算不可能な関数の例

関数の性質についての述語は計算不可能になることが多い.

例2.19. 与えられたプログラムが計算する関数が恒等的に0か?

Zero(a): aはプログラムAのコードで，任意のxに対していつでもA(x)=0

Zeroは計算不可能.

まず，次のプログラム$Y_{(a,x)}(t)$は作成可能:

- 固定されたプログラムコードaとxに対して,
- A(x)がtステップで止まるかどうかを模倣して
  - 止まるなら1を出力
  - 止まらないなら0を出力

「プログラムはいつか止まるか？」という問題は解けないが,
「プログラムはtステップ後に止まるか？」という問題なら解ける.

ここで,

$\mathrm{Halt}(a,x)=\text{“yes”} \Leftrightarrow \exists t\ Y_{(a,x)}(t)=1 \Leftrightarrow \mathrm{Zero}(y_{(a,x)})=\text{“no”}$

$\mathrm{Halt}(a,x)=\text{“no”} \Leftrightarrow \forall t\ Y_{(a,x)}(t)=0 \Leftrightarrow \mathrm{Zero}(y_{(a,x)})=\text{“yes”}$

である. よって,

もしZeroが計算可能なら，Halt(a,x)は次のように計算できる:

- a,xから$Y_{(a,x)}$のプログラムコード $y_{(a,x)}$ を構築する
- $\mathrm{Zero}(y_{(a,x)})$を計算して，その逆を出力する.

Haltは計算不可能だったので，Zeroも計算不可能になる.

2.5 Examples of incomputable functions

Predicates concerning on properties of functions are often incomputable.

Ex.2.19. Does a given program always output 0?

Zero(a): a is a code of program A, and A(x)=0 for any x.

Zero is incomputable.

The following program $Y_{(a,x)}(t)$ exists:

- For any fixed program code a and x,
- Simulate A(x) t steps, and
  - Output 1 if it halts
  - Output 0 if it does not halt

Now, we have;

$\text{Halt}(a,x)=\text{“yes”} \Leftrightarrow \exists t\ Y_{(a,x)}(t)=1 \Leftrightarrow \text{Zero}(y_{(a,x)})=\text{“no”}$

$\text{Halt}(a,x)=\text{“no”} \Leftrightarrow \forall t\ Y_{(a,x)}(t)=0 \Leftrightarrow \text{Zero}(y_{(a,x)})=\text{“yes”}$

Therefore, if Zero is computable, Halt(a,x) is also computable as follows:

- Construct a program code $y_{(a,x)}$ of $Y_{(a,x)}$ from a,x, and
- Output the opposite of $\text{Zero}(y_{(a,x)})$.

The problem “does a given program halts?” cannot be solved, while the problem “does a given program halts after t steps?” is solvable.

Since Halt is incomputable, Zero is also incomputable.

例2.20 与えられたプログラムは全域的か?

Total(a): a はプログラムAのコードで，すべてのxについてA(x)≠⊥

プログラムAに対して次の作業を考える.
(1)その中のhalt文を探し出す．→halt(*y*)と仮定
(2)「y≠0なら無限ループ，それ以外なら0を出力して停止」と書き換える.
(3)以上のことをすべてのhalt文について行う.
出来上がったプログラムをBとする.
　　プログラムAが0以外を出力すると必ず無限ループに陥る.
　　i.e., Aが常に0を出力しない限り，Bは全域的にならない.
上記の変換は計算可能➔次の関数が計算可能
　　replace(*a*) = *b*, *a*がプログラムコードのとき,
　　　　= *a*, その他のとき.
　　ただし，*b*はプログラムBのコード

一方、$\forall a \in \Sigma^*$[Zero(*a*)⇔Total(replace(*a*))]

よって，Totalが計算可能なら，Zeroも計算可能
　　i.e., Totalは計算不可能

Ex. 2.20 Is a given program total?

Total(a): a is a code of program A, and A(x)≠⊥ for any x.

Given a program A, consider the following computation.
(1) Find all halt statements.→assume that it is halt(*y*).
(2) Rewrite as [if y≠0 then loop, otherwise output y and halt].
(3) For each halt statement, do the above.
Let B be the resulting program.

If the program A outputs other than 0 then it enters infinite loop.
i.e., unless A always outputs 0, B is not total.

The above conversion is computable➔So is the following function

replace(*a*) = *b*, if *a* is a program code,
= *a*, otherwise,
where *b* is a code of the converted program B above.

On the other hand, $\forall a \in \Sigma^*$[Zero(*a*)⇔Total(replace(*a*))]

Therefore, if Total is computable, then Zero is also computable, a contradiction. Total is incomputable.

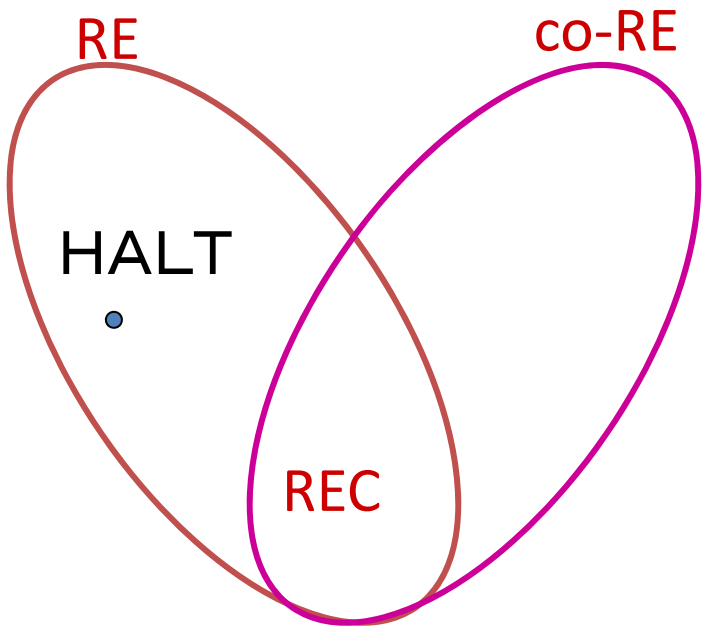
定理3.9. RE ≠ co-RE

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



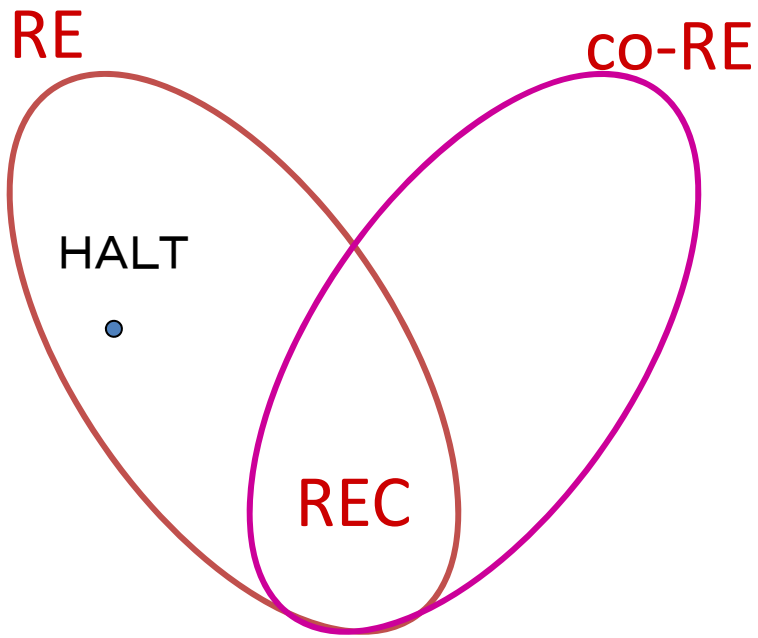
Theorem 3.9. RE ≠ co-RE

Proof:

If we assume RE=co-RE, we have RE=RE ∩ co-RE.

Hence, by Theorem 3.8 we have REC＝RE, contradicts to Theorem 3.7.

Q.E.D.



3.4. 還元可能性と完全性

- 問題の還元可能性
  …問題の相対的な難しさを測る方法
- 問題のあるクラスに関する完全性
  …そのクラス内で最も難しいことを示す方法

クラスREに属している集合の“難しさ”の比較

*A*は帰納的だが*B*は帰納的でないとき，
*B*は*A*より難しいと言える．

では，*A*と*B*が共に帰納的でない場合は？
← 帰納的還元性による比較

A, *B* : 集合
*A*を*B*へ還元する ← *A*の認識問題を*B*の認識問題に言い換えること．
（*A*は*B*へ還元可能）

3.4. Reducibility and Completeness

- Reducibility of a problem
  …Measure of relative hardness of the problem
- Completeness of a problem in a class
  …Most difficult problem in the class

Comparison of sets in the class RE by their "hardness"

If *A* is recursive but *B* is not recursive, then we can say that *B* is *harder* than *A*.

Then, what about if neither *A* nor *B* is recursive?

← comparison based on reducibility

A, B : sets

Reduce *A* to *B* ← Replace the recognition problem of *A* with the recognition problem of *B*.

(*A* is reducible to *B*)

定義3.4:

A, *B* : 任意の集合

(1) 次の条件を満たす関数 *h* を*A*から*B*への帰納的還元という.

(a) *h* は Σ^* からΣ^* への関数(全域的)

(b) $\forall x \in \Sigma^* [x \in A \leftrightarrow h(x) \in B]$

(c) *h* は計算可能

(2) *A*から*B*への帰納的還元が存在するとき,

*A*は*B*へ帰納的に還元可能という.

なお, *A*が*B*へ帰納的還元可能であることを$A \leq_m B$と記述する.

(*m* は, recursive **m**any-one reduction の *m*)

Definition 3.4:

A, B : arbitrary sets

(1) A function h is recursive reduction from A to B if

(a) h is a total function from Σ^* to Σ^*

(b) $\forall x \in \Sigma^* [x \in A \leftrightarrow h(x) \in B]$

(c) h is computable.

(2) If there is a recursive reduction from A to B,
we say that A is recursively reducible to B.

By $A \leq_m B$ we express that A is recursively reducible to B.
(the m in the suffix indicates recursive **m**any-one reduction)

例3.10

EVEN＝{ $\lceil n \rceil$: nは偶数}，　ODD＝{ $\lceil n \rceil$: nは奇数}

$\lceil n \rceil$ はnの2進表記（n: 自然数）

$$h_1(x) \equiv \begin{cases} \lceil n+1 \rceil & x = \lceil n \rceil \text{となっているとき} \\ x, & \text{その他のとき} \end{cases}$$

この h_1 は明らかに全域的かつ計算可能．また，

$$\forall x \in \Sigma^*[x \in \mathrm{EVEN} \leftrightarrow h_1(x) \in \mathrm{ODD}]$$

よって，h_1 はEVENからODDへの帰納的還元

$$\therefore \mathrm{EVEN} \leq_m \mathrm{ODD}$$

同じ h_1 がODDからEVENへの帰納的還元にもなっている．

$$\forall x \in \Sigma^*[x \in \mathrm{ODD} \rightarrow h_1(x) \in \mathrm{EVEN}]$$

$$\forall x \in \Sigma^*[h_1(x) \in \mathrm{EVEN} \rightarrow \exists n \geq 0[h_1(x) = \lceil n+1 \rceil \in \mathrm{EVEN}]]$$

$$\rightarrow \exists n \geq 1[h_1(x) = \lceil n+1 \rceil \in \mathrm{EVEN}]]$$

$$\rightarrow \exists n \geq 1[x = \lceil n \rceil \in \mathrm{ODD}]] \rightarrow [x \in \mathrm{ODD}]$$

$$\therefore \mathrm{ODD} \leq_m \mathrm{EVEN}$$

Ex.3.10

EVEN = { $\lceil n \rceil$: n is even }, ODD = { $\lceil n \rceil$: n is odd }

$\lceil n \rceil$ is binary representation of n (n : natural number)

$$h_1(x) \equiv \begin{cases} \lceil n+1 \rceil & \text{if } x = \lceil n \rceil \\ x, & otherwise \end{cases}$$

This h_1 is obviously total and computable. Also,

$$\forall x \in \Sigma^* [x \in \text{EVEN} \leftrightarrow h_1(x) \in \text{ODD}]$$

Therefore, h_1 is a recursive reduction from EVEN to ODD.

$$\therefore \text{EVEN} \leq_m \text{ODD}$$

The same h_1 is also a recursive reduction from ODD to EVEN.

$$\forall x \in \Sigma^* [x \in \text{ODD} \rightarrow h_1(x) \in \text{EVEN}]$$

$$\forall x \in \Sigma^* [h_1(x) \in \text{EVEN} \rightarrow \exists n \geq 0 [h_1(x) = \lceil n+1 \rceil \in \text{EVEN}]]$$

$$\rightarrow \exists n \geq 1 [h_1(x) = \lceil n+1 \rceil \in \text{EVEN}]]$$

$$\rightarrow \exists n \geq 1 [x = \lceil n \rceil \in \text{ODD}]] \rightarrow [x \in \text{ODD}]$$

$$\therefore \text{ODD} \leq_m \text{EVEN}$$

EVENからODDへのもっと単純な還元

$$h_2(x) \equiv \begin{cases} 1 & x \in \text{EVEN}\text{のとき} \\ 10 & \text{その他のとき} \end{cases}$$

自然数の偶奇が判定可能なので，h_2 は計算可能

$1 \in \text{ODD}, 10 \notin \text{ODD}$だから

$x \in \text{EVEN} \rightarrow h_2(x) = 1 \in \text{ODD}$

$x \notin \text{EVEN} \rightarrow h_2(x) = 10 \notin \text{ODD}$

$\therefore x \in \text{EVEN} \leftrightarrow h_2(x) \in \text{ODD}$

Simpler reduction from EVEN to ODD

$$h_2(x) \equiv \begin{cases} 1 & x \in \text{EVEN} \\ 10 & \text{otherwise} \end{cases}$$

Since odd-evenness of a natural number is computable, so is h_2.

Since $1 \in \text{ODD}$, $10 \notin \text{ODD}$

$x \in \text{EVEN} \rightarrow h_2(x) = 1 \in \text{ODD}$

$x \notin \text{EVEN} \rightarrow h_2(x) = 10 \notin \text{ODD}$

$\therefore x \in \text{EVEN} \leftrightarrow h_2(x) \in \text{ODD}$

定理3.12: $A \leq_m B$という関係にある任意の集合A,Bを考える.
このとき，Bが帰納的→Aも帰納的.

証明:

$A \leq_m B$➔ AからBへの帰納的還元 h が存在する.

よって，$x \in A$という判定問題 ➔ $h(x) \in B$？

つまり，次のプログラムはAを認識する.

```
prog A(input x);
begin
    if h(x)∈ B then accept else reject end-if
end.
```

Bが帰納的なら，Bを認識するプログラムが存在する.

→$h(x) \in B$を判定するプログラム

これで上記のプログラムAが完成.

よって，Aは帰納的.　　証明終

Theorem 3.12: Consider any sets A and B such that $A \leq_m B$.
Then, B is recursive → A is also recursive.

Proof:

$A \leq_m B$ ➔ there is a recursive reduction h from A to B.

So, the decision problem of $x \in A$ ➔ $h(x) \in B$?

That is, the following program recognizes A.

```
prog A(input x);
begin
   if h(x) ∈ B then accept else reject end-if
end.
```

If B is recursive, there is a program that recognizes B.

→a program that determines $h(x) \in B$

Now, we have a complete program A.

Thus, A is recursive. Q.E.D.

与えられた集合が“手に負えない”ことを示すための方法を示唆

(i) $A \leq_m B$ かつ
(ii) Aは帰納的でない.

→ このような集合Aを示せれば，Bは帰納的でない

IsProgram(a): a はプログラム A のコードか？
これは計算可能な関数である.

例3.11:

$$ZERO \equiv \{a: IsProgram(a) \wedge \forall x[\, A_{\iota}(x) = 0]\}$$

$$TOTAL \equiv \{a: IsProgram(a) \wedge \forall x[\, A_{\iota}(x) \neq \perp\,]\}$$

まとめると

関係　　したがって,

$\overline{HALT} \leq_m ZERO$　　$ZERO \notin REC$　（$\overline{HALT} \notin REC$より）

$ZERO \leq_m TOTAL$　　$TOTAL \notin REC$　（$ZERO \notin REC$より）

It suggests a method to show that a given set is “intractable”

(i) $A \leq_m B$ and

(ii) A is not recursive.

If we can show such a set A, then B is not recursive.

IsProgram(a): Is a the program code of A?
This problem is computable.

Ex.3.11:

$$\text{ZERO} \equiv \{a: \text{IsProgram}(a) \wedge \forall x[\, A\, \iota(x) = 0]\}$$

$$\text{TOTAL} \equiv \{a: \text{IsProgram}(a) \wedge \forall x|\, A\, (x) \neq \perp\,]\}$$

Summarizing,

relation	what follows
$\overline{\text{HALT}} \leq_m \text{ZERO}$	$\text{ZERO} \notin \text{REC}$ (by $\overline{\text{HALT}} \notin \text{REC}$)
$\text{ZERO} \leq_m \text{TOTAL}$	$\text{TOTAL} \notin \text{REC}$ (by $\text{ZERO} \notin \text{REC}$)

定理3.13. $A \leqq_m B$という関係にある任意の集合A, Bを考える.
このとき、次のことが成り立つ。
(1) $B \in$ RE $\longrightarrow$ $A \in$ RE　(Bが枚挙可能 $\longrightarrow$ Aも枚挙可能)
(2) $B \in$ co-RE $\longrightarrow$ $A \in$ co-RE

(補注) 対偶を考えると、
(1) $A \notin$ RE $\rightarrow$ $B \notin$ RE
(2) $A \notin$ co-RE $\rightarrow$ $B \notin$ co-RE

例3.11, 定理3.13 → ZERO、TOTALは
REにもco-REにも属さない。

性質	理由
ZERO $\notin$ RE	$\overline{\text{HALT}} \notin$ RE、$\overline{\text{HALT}} \leqq_m$ ZERO
ZERO $\notin$ co-RE	HALT $\notin$ co-RE、HALT $\leqq_m$ ZERO
TOTAL $\notin$ RE	ZERO $\notin$ RE、ZERO $\leqq_m$ TOTAL
TOTAL $\notin$ co-RE	ZERO $\notin$ co-RE、ZERO $\leqq_m$ TOTAL

Theorem 3.13. Consider any sets A and B such that $A \leqq_m B$. Then, we have:

(1) $B \in$ RE $\rightarrow A \in$ RE (B is enumerable $\rightarrow$ so is A)

(2) $B \in$ co-RE $\rightarrow A \in$ co-RE

(Remark) Their contrapositions:

(1) $A \notin$ RE $\rightarrow B \notin$ RE

(2) $A \notin$ co-RE $\rightarrow B \notin$ co-RE

Ex.3.11, Theorem 3.13 → Neither ZERO or TOTAL belongs to RE or co-RE.

property	reason
ZERO $\notin$ RE	$\overline{\text{HALT}} \notin$ RE、$\overline{\text{HALT}} \leqq_m$ ZERO
ZERO $\notin$ co-RE	HALT $\notin$ co-RE、HALT $\leqq_m$ ZERO
TOTAL $\notin$ RE	ZERO $\notin$ RE、ZERO $\leqq_m$ TOTAL
TOTAL $\notin$ co-RE	ZERO $\notin$ co-RE、ZERO $\leqq_m$ TOTAL

還元可能性 ： 難しさを比較する手段

$A \leqq_m B$ → Aの認識問題をBの認識問題に変換できる。

↓

Aの難しさ ≦ Bの難しさ

（Bを認識するプログラムがあればAの認識に使える。）

定理3.14.

任意に与えられた集合 A, B, C に対し、次の関係が成り立つ

(1) $A \leqq_m A$

(2) $A \leqq_m B$ かつ $B \leqq_m C$ ならば $A \leqq_m C$

$A \equiv_m B \overset{\text{def}}{\Leftrightarrow} A \leqq_m B$ かつ $B \leqq_m A$

$\equiv_m$ は同値関係（同程度の難しさ）

$A \equiv_m B$ のとき、A と B は $\equiv_m$-同値という。

Reducibility : a means of comparing hardness

$A \leqq_m B$ → We can convert the recognition problem of A into that of B.

↓

hardness of $A \leq$ hardness of B

（A program recognizing B can be used to recognize A.）

Theorem 3.14. For any given sets A, B, C, we have

（1） $A \leqq_m A$

（2） $A \leqq_m B$ and $B \leqq_m C$ implies $A \leqq_m C$

$A \equiv_m B \overset{\text{def}}{\leftrightarrow} A \leqq_m B$ and $B \leqq_m A$

$\equiv_m$ is an equivalence relation (equal hardness)

If $A \equiv_m B$, we say that A and B are $\equiv_m$-equivalent.

例3.13.

ZERO $\notin$ RE　　$\therefore$ ZERO $\not\leq_m$ HALT

（$\because$ ZERO $\leq_m$ HALTとすると、HALT $\in$ REなので
ZERO $\in$ REとなり矛盾）

一方、HALT $\leq_m$ ZERO

$\therefore$ ZEROはHALTより真に難しい。

例3.14.

すべての帰納的集合は互いに帰納的に同値。

たとえば、EVEN（偶数の集合）とPRIME（素数の集合）は帰納的に同値

EVEN $\equiv_m$ PRIME

（両方とも帰納的という意味で同程度の難しさ）

どちらも計算できるという意味で同程度に難しい

Ex. 3.13.

$ZERO \notin RE$ $\therefore ZERO \not\leq_m HALT$

（$\because$ if $ZERO \leq_m HALT$ we have $HALT \in RE$ and $ZERO \in RE$, a contradiction）

On the other hand, $HALT \leq_m ZERO$

$\therefore ZERO$ is strictly harder than HALT.

Ex. 3.14.

All the recursive sets are recursively equivalent to each other.

For example, EVEN（set of even numbers）and PRIME（set of primes）are recursively equivalent

$EVEN \equiv_m PRIME$

（both of them are equally hard in the sense that they are recursive.）

both computable

“クラスREの中で最も難しい集合”の定義

（one of the most difficult sets in RE）

定義3.5.

集合 A が次の条件を満たすとき、それを（$\leqq_m$ のもとで）RE-完全（RE-complete）という。

（a）$\forall L \in \mathrm{RE}$ ［$L \leqq_m A$］

(A より真に難しいものはREには存在しない)

（b）$A \in \mathrm{RE}$

集合Aが上記の条件（a）だけを満たすとき、RE-困難（RE-Hard）という。

（すべてのRE集合より難しい集合のこと）

Definition of "**the hardest sets in the class RE**"

Def. 3.5.
A set A is called RE-complete (under $\leqq_m$) if the following conditions hold

(a) $\forall L \in \mathrm{RE} \quad [L \leqq_m A]$
(no element of RE is strictly harder than A).

(b) $A \in \mathrm{RE}$

If a set A satisfies only (a) above, it is called RE-hard.
(meaning sets harder than any RE set)

定理3.15: HALTはRE-完全

（証明）

HALT ∈ REなので、条件（b）はOK。

L：任意のRE集合とする。

→Lを半認識するプログラム L が存在する

すべての $x \in \Sigma^*$ に対し、

$$x \in L \iff \text{Halt}(\langle L, x \rangle) \iff \langle L, x \rangle \in \text{HALT}$$

よって、$h(x) \stackrel{\text{def}}{=} \langle L, x \rangle$ は L からHALTへの帰納的還元。

（証明終）

Theorem 3.15 HALT is RE-complete.

(Proof)

Since HALT $\in$ RE, the condition (b) is satisfied.

L : any RE set.

→ a program L that semi-recognizes L.

for any $x \in \Sigma^*$

$$x \in L \Leftrightarrow \text{Halt}(\langle L, x \rangle) \Leftrightarrow \langle L, x \rangle \in \text{HALT}$$

Thus, $h(x) \stackrel{\text{def}}{=} \langle L, x \rangle$ is a recursive reduction from L to HALT.

Q.E.D.

定理3.16: A, B を任意の集合とする。
（1）[AがRE-困難] かつ [$A \leqq_m B$] ならば B はRE-困難
（2）A がRE-困難　$\leftrightarrow \overline{A}$ がco-RE-困難

例3.15. 定理3.16 を用いて、いろいろな集合の
困難性（完全性）を示す。

集合	難しさ	主な理由
HALT	RE-完全	定理３．１５
$\overline{\text{HALT}}$	co-RE完全	HALTがRE-困難、$\overline{\text{HALT}} \in$ co-RE
ZERO	RE-困難、co-RE困難	HALT $\leq_m$ZERO、
$\overline{\text{TOTAL}}$	RE-困難、co-RE困難	ZERO $\leq_m$TOTAL

Theorem 3.16: Let A and B be arbitrary sets.

(1) [A is RE-hard and $A \leqq_m B$] implies B is RE-hard.

(2) A is RE-hard $\leftrightarrow \overline{A}$ is co-RE-hard.

Ex.3.15 Using Theorem 3.16, we can show hardness of various sets.

Sets	hardness	reasons
$\overline{HALT}$	RE-complete	Theorem3.15
HALT	co-RE complete	HALT is RE-hard, HALT$\in$ co-RE
$\overline{ZERO}$	RE-hard、co-RE hard	HALT $\leq_m$ ZERO、
TOTAL	RE-hard、co-RE hard	ZERO $\leq_m$TOTAL

H: RE-完全集合の集合

H: REの中で“最も難しい集合”

REC：REの中で“最もやさしい集合”

還元 $\leqq_m$ のもとで

定理3.17.

（1）$\mathrm{REC} \cap H = \phi$

（2）$\mathrm{RE} - (\mathrm{REC} \cup H) \neq \phi$

（1）$\mathrm{REC} \subsetneqq \mathrm{RE}$

RECは同値関係 $\equiv_m$ のもとで閉じている。

（2）の証明は複雑なので省略。

H: an RE-complete set

H: "hardest set" in RE

REC: "easiest set" in RE

Under the reduction $\leqq_m$

Theorem 3.17.

(1) $REC \cap H = \phi$

(2) $RE - (REC \cup H) \neq \phi$

(1) $REC \subsetneqq RE$

REC is closed under the equivalence relation $\equiv_m$.

(2) The proof is complicated, and so omitted.