# Friedberg-Muchnik の定理と有限害優先論法

y. (@waidotto)

2019 年 10 月 27 日 @ 第 12 回関西すうがく徒のつどい 2 日目

http://iso.2022.jp/math/tsudoi/12/slide.pdf

# 目次

# 1. 概要

この講演の全体像について.

# 決定問題

決定問題 (decision problem) とは，与えられた入力に対して YES/NO で答えるタイプの計算課題をいう．例えば次のようなもの:

**例 (素数判定問題)**

**Input:** 自然数 $n$

**Question:** $n$ は素数か？

- $7 \mapsto$ YES,
- $12 \mapsto$ NO,
- $20191027 \mapsto$ YES

**定義 (決定可能性)**

決定問題について

- 決定可能 (decidable) $\iff$ 全ての入力に対して正しい答えを返す単一のアルゴリズム (= プログラム) が存在,
- 決定不能 (undecidable) $\iff$ 決定可能でない

と定義する.

# 決定問題

決定問題 (decision problem) とは，与えられた入力に対して YES/NO で答えるタイプの計算課題をいう．例えば次のようなもの:

**例 (素数判定問題)**

**Input:** 自然数 $n$

**Question:** $n$ は素数か？

- $7 \mapsto$ YES,
- $12 \mapsto$ NO,
- $20191027 \mapsto$ YES

**定義 (決定可能性)**

決定問題について

- 決定可能 (decidable) $\iff$ 全ての入力に対して正しい答えを返す単一のアルゴリズム (= プログラム) が存在,
- 決定不能 (undecidable) $\iff$ 決定可能でない

と定義する.

# 数学に自然に現れる決定問題

数学には様々な決定問題が自然に現れる．いくつかの例を挙げる．

## 充足可能性問題 (SAT)

決定可能

**Input:** 命題論理式 $\varphi(x_1, \ldots, x_n)$

**Question:** $\exists a_1, \ldots, a_n \in \{\mathbf{True}, \mathbf{False}\}[\varphi(a_1, \ldots, a_n) = \mathbf{True}]$?

## Post の対応問題 (PCP)

決定不能

**Input:** 有限集合 $\Sigma$ とモノイド準同型 $f, g \colon \Sigma^* \to \{0, 1\}^*$

**Question:** $\exists \sigma \in \Sigma^*[f(\sigma) = g(\sigma)]$?

## Hilbert の第 10 問題 (HTP)

決定不能

**Input:** 多項式 $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, x_2, \ldots]$

**Question:** $\exists a_1, \ldots, a_n \in \mathbb{Z}[f(a_1, \ldots, a_n) = 0]$?

ポイント どれも「〜は存在するか？」という形の質問になっている．

# 数学に自然に現れる決定問題

数学には様々な決定問題が自然に現れる．いくつかの例を挙げる．

## 充足可能性問題 (SAT)

決定可能

**Input:** 命題論理式 $\varphi(x_1, \ldots, x_n)$
**Question:** $\exists a_1, \ldots, a_n \in \{\mathbf{True}, \mathbf{False}\}[\varphi(a_1, \ldots, a_n) = \mathbf{True}]$?

## Post の対応問題 (PCP)

決定不能

**Input:** 有限集合 $\Sigma$ とモノイド準同型 $f, g \colon \Sigma^* \to \{0, 1\}^*$
**Question:** $\exists \sigma \in \Sigma^*[f(\sigma) = g(\sigma)]$?

## Hilbert の第 10 問題 (HTP)

決定不能

**Input:** 多項式 $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, x_2, \ldots]$
**Question:** $\exists a_1, \ldots, a_n \in \mathbb{Z}[f(a_1, \ldots, a_n) = 0]$?

ポイント どれも「〜は存在するか？」という形の質問になっている．

# 数学に自然に現れる決定問題

数学には様々な決定問題が自然に現れる．いくつかの例を挙げる．

## 充足可能性問題 (SAT)

決定可能

**Input:** 命題論理式 $\varphi(x_1, \ldots, x_n)$

**Question:** $\exists a_1, \ldots, a_n \in \{\mathbf{True}, \mathbf{False}\}[\varphi(a_1, \ldots, a_n) = \mathbf{True}]$?

## Post の対応問題 (PCP)

決定不能

**Input:** 有限集合 $\Sigma$ とモノイド準同型 $f, g \colon \Sigma^* \to \{0, 1\}^*$

**Question:** $\exists \sigma \in \Sigma^*[f(\sigma) = g(\sigma)]$?

## Hilbert の第 10 問題 (HTP)

決定不能

**Input:** 多項式 $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, x_2, \ldots]$

**Question:** $\exists a_1, \ldots, a_n \in \mathbb{Z}[f(a_1, \ldots, a_n) = 0]$?

ポイント どれも「〜は存在するか？」という形の質問になっている．

# プログラムの停止問題

重要な決定不能問題としてプログラムの停止問題がある.

**停止問題 (halting problem)**

**Input:** プログラム $P$ と入力 $x$
**Question:** $P(x)$ の計算は停止するか？

**定理**

停止問題は決定不能である.

証明はのちほど.

# 決定不能性の証明法

多くの重要な決定問題 $\mathcal{P}$ について，その決定不能性の証明はどれも

「仮に $\mathcal{P}$ を判定するアルゴリズムが存在したとすると，
それをもとにして停止問題を判定するアルゴリズムが作れてしまう」

という論法によっている．

**Post の問題 (ざっくり版)**

どんな決定不能問題も，その決定不能性を上の方法で証明できるだろうか？

**答 (Friedberg-Muchnik の定理)**

できない．現実は非情である．

すなわち，決定不能ではあるが，その決定不能性を上の方法では証明できないような決定問題が存在する！

この定理を証明することが今回の目標．

# 決定不能性の証明法

多くの重要な決定問題 $\mathcal{P}$ について，その決定不能性の証明はどれも

「仮に $\mathcal{P}$ を判定するアルゴリズムが存在したとすると，
それをもとにして停止問題を判定するアルゴリズムが作れてしまう」

という論法によっている．

**Post の問題 (ざっくり版)**

どんな決定不能問題も，その決定不能性を上の方法で証明できるだろうか？

**答 (Friedberg-Muchnik の定理)**

できない．現実は非情である．

すなわち，決定不能ではあるが，その決定不能性を上の方法では証明できないような決定問題が存在する！

この定理を証明することが今回の目標．

# 1 節のまとめ

- 入力に対して YES/NO で答えるタイプの計算課題を決定問題といい，アルゴリズム (= プログラム) によって解けない決定問題を決定不能問題という．
- 重要な決定問題はたいてい

  「入力に対して，～は存在するか？」

  という形をしている．
- 数学に自然に現れる決定不能問題 $\mathcal{P}$ は

  「$\mathcal{P}$ を判定するプログラムがあったら，
  それをもとに停止問題を判定するプログラムが作れてしまう」

  という論法で決定不能性が証明される．
- しかし，上の方法では決定不能性が証明できない決定不能問題が存在する (Friedberg-Muchnik の定理)．

# 2. 計算論からの準備

ここまでの内容を，計算可能性理論の言葉を用いて数学的にきちんと定式化してみる.

# すべての (有限な) 概念は自然数である (1/2)



コンピューター内部では，あらゆる情報は $0$ と $1$ からなるビット列によって表現される．




このビット列をある数の $2$ 進展開とみなすことで，計算の対象となるモノ (= 決定問題の入力) はどれもひとつの自然数であると仮定して差し支えない．

よって，以降は決定問題への入力はすべて自然数

$$\omega = \{0, 1, 2, \dots\}$$

であると仮定する．

$0$ を含める

# すべての (有限な) 概念は自然数である (1/2)

コンピューター内部では，あらゆる情報は $0$ と $1$ からなるビット列によって表現される．


このビット列をある数の $2$ 進展開とみなすことで，計算の対象となるモノ ($=$ 決定問題の入力) はどれもひとつの自然数であると仮定して差し支えない．

よって，以降は決定問題への入力はすべて自然数

$$\omega = \{0, 1, 2, \dots\}$$

であると仮定する．

$0$ を含める

# すべての (有限な) 概念は自然数である (2/2)

決定問題への入力を全て自然数であるとみなせることから，決定問題を関数

$$f\colon \omega \to \{0,1\}$$

$0 = \textsc{No}, 1 = \textsc{Yes}$ とする

と同一視することができる．さらにこれは

$$A = \{\, x \in \omega \mid f(x) = 1 \,\}$$

という $\omega$ の部分集合と同一視できる．$f$ を $A$ の特性関数 (characteristic function) と呼び，$\chi_A$ で表す．

すなわち，決定問題 $A \subseteq \omega$ について

$A$ が決定可能 $\iff$ $\chi_A\colon \omega \to \{0,1\}$ を計算するアルゴリズムが存在.

次はこれを形式化する

# すべての (有限な) 概念は自然数である (2/2)

決定問題への入力を全て自然数であるとみなせることから，決定問題を関数

$$f\colon \omega \to \{0,1\}$$

$0 = \textsc{No}, 1 = \textsc{Yes}$ とする

と同一視することができる．さらにこれは

$$A = \{\, x \in \omega \mid f(x) = 1 \,\}$$

という $\omega$ の部分集合と同一視できる．$f$ を $A$ の特性関数 (characteristic function) と呼び，$\chi_A$ で表す．

すなわち，決定問題 $A \subseteq \omega$ について

$$A \text{ が決定可能} \iff \chi_A\colon \omega \to \{0,1\} \text{ を計算するアルゴリズムが存在}.$$

次はこれを形式化する

# すべてのプログラムを並べ上げる

個々のプログラムは有限長の文字列なので，それらを全て一列に並べて自然数で番号付けることができる．

**定義 (計算可能部分関数)**

$e \in \omega$ 番目のプログラムによって計算される部分関数 $\omega \rightharpoonup \omega$ を $\varphi_e$ で表す．

ただし，プログラムによって計算される関数は入力によってはいつまでも計算が停止せず値を返さないかもしれない．そのような “関数” を部分関数 (partial function) という．

**定義 (部分関数)**

$\omega$ のある部分集合から $\omega$ への関数を $\omega$ から $\omega$ への部分関数 (partial function) といい，記号 $\varphi\colon \omega \rightharpoonup \omega$ で表す．

# すべてのプログラムを並べ上げる

個々のプログラムは有限長の文字列なので，それらを全て一列に並べて自然数で番号付けることができる．

**定義 (計算可能部分関数)**

$e \in \omega$ 番目のプログラムによって計算される部分関数 $\omega \rightharpoonup \omega$ を $\varphi_e$ で表す．

ただし，プログラムによって計算される関数は入力によってはいつまでも計算が停止せず値を返さないかもしれない．そのような “関数” を部分関数 (partial function) という．

**定義 (部分関数)**

$\omega$ のある部分集合から $\omega$ への関数を $\omega$ から $\omega$ への部分関数 (partial function) といい，記号 $\varphi\colon \omega \rightharpoonup \omega$ で表す．

# すべてのプログラムを並べ上げる

個々のプログラムは有限長の文字列なので，それらを全て一列に並べて自然数で番号付けることができる．

**定義 (計算可能部分関数)**

$e \in \omega$ 番目のプログラムによって計算される部分関数 $\omega \rightharpoonup \omega$ を $\varphi_e$ で表す．

ただし，プログラムによって計算される関数は入力によってはいつまでも計算が停止せず値を返さないかもしれない．そのような “関数” を部分関数 (partial function) という．

**定義 (部分関数)**

$\omega$ のある部分集合から $\omega$ への関数を $\omega$ から $\omega$ への部分関数 (partial function) といい，記号 $\varphi\colon \omega \rightharpoonup \omega$ で表す．

# 暫定的な計算結果

プログラムの実行が最終的に停止するかどうかだけでなく，「今のところまだ停止していない」といったことを表現できるようにしておくと便利である．

**定義**

入力 $x \in \omega$ に対し，$\varphi_e(x)$ が定義される ($=$ 計算が停止する，$x \in \mathrm{dom}(\varphi_e)$) ことを $\varphi_e(x)\downarrow$ で表し，そうでないことを $\varphi_e(x)\uparrow$ で表す．

また，$\varphi_e(x)$ の計算が $s \in \omega$ ステップ以内に停止することを $\varphi_{e,s}(x)\downarrow$ で表し，そうでないことを $\varphi_{e,s}(x)\uparrow$ で表す．

さらに，$\varphi_e(x)\downarrow$ かつ $\varphi_e(x) = y$ であることを $\varphi_e(x)\downarrow = y$，または単に $\varphi_e(x) = y$ で表す．$\varphi_{e,s}(x)\downarrow = y$ も同様．

ポイント 各 $e, s, x$ 毎に $\varphi_{e,s}(x)\downarrow$ は有限ステップで判定可能な条件である．

# 停止問題再訪

ここまでで定義した記号を用いると，停止問題は

$$K_0 := \{ \langle e, x \rangle \in \omega \mid \varphi_e(x)\downarrow \}$$

と書ける．

$2$ つの自然数 $(e, x)$ をひとつの自然数に符号化したもの

**$K_0$ の決定不能性の証明.**

仮に $\chi_{K_0}$ が計算可能関数だったとすると，

$$f(e) = \begin{cases} \varphi_e(e) + 1 & (\chi_{K_0}(\langle e, e \rangle) = 1 \text{ のとき}), \\ 0 & (\chi_{K_0}(\langle e, e \rangle) = 0 \text{ のとき}) \end{cases}$$

も 全域的 な計算可能関数である．よって $f = \varphi_i$ となる $i \in \omega$ があるが，

$\mathrm{dom}(f) = \omega$

$$\chi_{K_0}(\langle i, i \rangle) = 1 \implies \varphi_i(i) = f(i) = \varphi_i(i) + 1,$$
$$\chi_{K_0}(\langle i, i \rangle) = 0 \implies \varphi_i(i)\uparrow \iff f(i)\uparrow$$

となり矛盾．

$f$ の全域性に矛盾

□

ここまでで定義した記号を用いると，停止問題は

$$K_0 := \{ \langle e, x \rangle \in \omega \mid \varphi_e(x)\downarrow \}$$

と書ける．

$\langle e, x \rangle$ : 2 つの自然数 $(e, x)$ をひとつの自然数に符号化したもの

**$K_0$ の決定不能性の証明.**

仮に $\chi_{K_0}$ が計算可能関数だったとすると，

$$f(e) = \begin{cases} \varphi_e(e) + 1 & (\chi_{K_0}(\langle e, e \rangle) = 1 \text{ のとき}), \\ 0 & (\chi_{K_0}(\langle e, e \rangle) = 0 \text{ のとき}) \end{cases}$$

も 全域的 な計算可能関数である．よって $f = \varphi_i$ となる $i \in \omega$ があるが，

$\mathrm{dom}(f) = \omega$

$$\chi_{K_0}(\langle i, i \rangle) = 1 \implies \varphi_i(i) = f(i) = \varphi_i(i) + 1,$$
$$\chi_{K_0}(\langle i, i \rangle) = 0 \implies \varphi_i(i)\uparrow \iff f(i)\uparrow$$

$f$ の全域性に矛盾

となり矛盾． □

# c.e. 集合とは

「入力に対し，〜は存在するか？」という形の決定問題を形式化したものを c.e. 集合という．

**定義 (c.e. 集合)**

集合 $A \subseteq \omega$ が c.e. 集合 (computably enumerable set, c.e. set) であるとは，ある決定可能な集合 $R \subseteq \omega$ が存在して，任意の $x \in \omega$ に対し

$$x \in A \iff \exists y \in \omega\, [\langle x, y \rangle \in R]$$

となることをいう．

**例 ($K_0$ は c.e. 集合)**

実際，

$$\begin{aligned} \langle e, x \rangle \in K_0 &\iff \varphi_e(x)\downarrow \\ &\iff \exists s \in \omega [\varphi_{e,s}(x)\downarrow] \end{aligned}$$

なので c.e. 集合である．

判定可能な条件になっている

# c.e. 集合とは

「入力に対し，〜は存在するか？」という形の決定問題を形式化したものを c.e. 集合という．

**定義 (c.e. 集合)**

集合 $A \subseteq \omega$ が c.e. 集合 (computably enumerable set, c.e. set) であるとは，ある決定可能な集合 $R \subseteq \omega$ が存在して，任意の $x \in \omega$ に対し

$$x \in A \iff \exists y \in \omega\, [\langle x, y \rangle \in R]$$

となることをいう．

**例 ($K_0$ は c.e. 集合)**

実際，

$$\begin{aligned} \langle e, x \rangle \in K_0 &\iff \varphi_e(x)\downarrow \\ &\iff \exists s \in \omega [\varphi_{e,s}(x)\downarrow] \end{aligned}$$

なので c.e. 集合である．

判定可能な条件になっている

# c.e. 集合の要素の並べ上げ (1/2)

**定理 (Listing theorem)**

$A \subseteq \omega$ に対し，

$A$ が c.e. 集合 $\iff$ $A = \emptyset$ または，$A$ はある計算可能関数 $f$ の像.

**証明.**

$A \neq \emptyset$ とする.

**($\Longrightarrow$)** $a \in A$ をひとつ固定し，

$$f(\langle x, y \rangle) = \begin{cases} x & (\langle x, y \rangle \in R \text{ のとき}), \\ a & (\langle x, y \rangle \notin R \text{ のとき}) \end{cases}$$

とおけばよい.

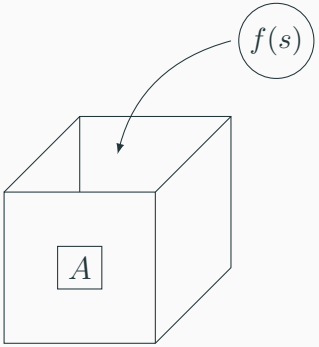
**($\Longleftarrow$)** $R = \{ \langle x, y \rangle \in \omega \mid f(y) = x \}$ とおけばよい. □

# c.e. 集合の要素の並べ上げ (2/2)

Listing Theorem から，c.e. 集合 $A = \mathrm{Im}(f)$ とは，$\boxed{A}$というラベルが貼られた空箱に順番に自然数

$$f(0), f(1), f(2), \ldots, f(s), \ldots$$

を投入してできる集合だと思うことができる．



# 神託計算 (1/2)

「仮に $\chi_B$ が計算可能だったら，それを利用して $\chi_A$ を計算するプログラムを作ることができる」ということをきちんと定式化しよう．

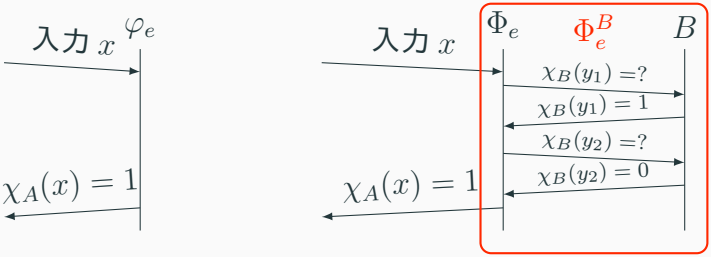
もし $\chi_B$ を計算するアルゴリズムがあれば，あらゆる自然数 $y \in \omega$ について $\chi_B(y)$ の値を知ることができるということになる．よって，$\chi_A(x)$ の値を計算するために，任意の自然数 $y \in \omega$ について $\chi_B(y)$ の値を好きなだけ参照してよい．

逆に，計算の途中で任意の $y \in \omega$ に対する $\chi_B(y)$ の値を参照することができる状況で $\chi_A$ が計算できるなら，「$\chi_B$ が計算可能ならば $\chi_A$ が計算可能である」と言ってよいだろう．

以上を定式化したものがオラクルプログラム (oracle program) である．

# 神託計算 (2/2)

オラクルプログラムとは，通常のプログラムの機能に加えて，計算の途中でオラクル (oracle) に自然数が属しているか否かを尋ねることができるようなプログラム．集合 $B$ をオラクルをした場合の動作は下図の通り．




普通のプログラム $\varphi_e$　オラクルプログラム $\Phi_e$ とオラクル $B$

※ただし，オラクル $B$ に $\chi_B(y)$ の値を尋ねるには少なくとも $y$ ステップだけかかるものと約束しておく．

ポイント　オラクルプログラム $\Phi_e$ そのものは有限の文字列で記述できる．

# Turing 還元

**定義 (Turing 還元)**

集合 $A, B \subseteq \omega$ に対し，あるオラクルプログラム $\Phi_e$ が存在して $\chi_A = \Phi_e^B$ となるとき，$A$ は $B$ に Turing 還元可能 (Turing reducible) あるいは $A$ は $B$ 計算可能 ($B$-computable) であるといい，記号 $A \leq_{\mathrm{T}} B$ で表す．

直感的には，

$A \leq_{\mathrm{T}} B$ “$\Longleftrightarrow$” $A$ より $B$ の方が難しい

“$\Longleftrightarrow$” $A$ より $B$ の方が多くの情報を持っている

ということである．

**例**

決定可能な集合 (例えば $\emptyset$ や $\omega$) はオラクルを用いなくても計算できるので，
任意の集合 $B \subseteq \omega$ に対し $\emptyset \leq_{\mathrm{T}} B$ が成り立つ．

また，$A \leq_{\mathrm{T}} \emptyset$ は $A$ が決定可能であることの言い換えである．

# Post の問題のきちんとした定式化

Turing 還元を用いると，Post の問題を正確に定式化することができる．

**命題**

任意の c.e. 集合 $A$ について，$\emptyset \leq_{\mathrm{T}} A \leq_{\mathrm{T}} K_0$.

**証明.**

$\emptyset \leq_{\mathrm{T}} A$ は明らか．$x \in A \iff \exists y[\langle x, y\rangle \in R]$ なる決定可能集合 $R \subseteq \omega$ をとり，$f(x)$ を「入力を無視し，$y = 0, 1, 2, \ldots$ のそれぞれについて $\langle x, y\rangle \in R$ かどうかを調べ，成り立った瞬間に停止する」ようなプログラムとする．このとき $x \in A \iff \langle f(x), 0\rangle \in K_0$ となるので $A \leq_{\mathrm{T}} K_0$ である． □

これを踏まえると，

**Post の問題**

$\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} K_0$ となるような c.e. 集合 $A \subseteq \omega$ は存在するか？

# Post の問題のきちんとした定式化

Turing 還元を用いると，Post の問題を正確に定式化することができる．

**命題**

任意の c.e. 集合 $A$ について，$\emptyset \leq_{\mathrm{T}} A \leq_{\mathrm{T}} K_0$.

**証明.**

$\emptyset \leq_{\mathrm{T}} A$ は明らか．$x \in A \iff \exists y[\langle x, y\rangle \in R]$ なる決定可能集合 $R \subseteq \omega$ をとり，$f(x)$ を「入力を無視し，$y = 0, 1, 2, \ldots$ のそれぞれについて $\langle x, y\rangle \in R$ かどうかを調べ，成り立った瞬間に停止する」ようなプログラムとする．このとき $x \in A \iff \langle f(x), 0\rangle \in K_0$ となるので $A \leq_{\mathrm{T}} K_0$ である． □

これを踏まえると，

**Post の問題**

$\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} K_0$ となるような c.e. 集合 $A \subseteq \omega$ は存在するか？

# Turing 次数

$\leq_{\mathrm{T}}$ は $\mathcal{P}(\omega)$ 上の反射的かつ推移的な二項関係 (前順序) である．よって

$$A \equiv_{\mathrm{T}} B \ :\Longleftrightarrow\ A \leq_{\mathrm{T}} B \text{ かつ } B \leq_{\mathrm{T}} A$$

とおくと $\equiv_{\mathrm{T}}$ は $\mathcal{P}(\omega)$ 上の同値関係になり，$\leq_{\mathrm{T}}$ は $\mathcal{D} := \mathcal{P}(\omega)/\equiv_{\mathrm{T}}$ 上の半順序になる．集合 $A \subseteq \omega$ の $\equiv_{\mathrm{T}}$ による同値類を $\deg(A)$ で表し，$A$ の Turing 次数 (Turing degree) と呼ぶ．Turing 次数は決定問題の難しさを表す量である．

$(\mathcal{D}, \leq_{\mathrm{T}})$ が次の性質を持つことはすぐにわかる．

- 各 Turing 次数は可算個の元からなる (オラクルプログラムが可算個しかないから)．よって $\mathcal{D}$ は連続体濃度を持つ．
- $\mathbf{0} = \deg(\emptyset)$ は決定可能問題の全体に一致し，$\mathcal{D}$ の最小元を与える．
- $\mathbf{0} <_{\mathrm{T}} \deg(K_0)$.

# Turing 次数

$\leq_{\mathrm{T}}$ は $\mathcal{P}(\omega)$ 上の反射的かつ推移的な二項関係 (前順序) である．よって

$$A \equiv_{\mathrm{T}} B \ :\Longleftrightarrow\ A \leq_{\mathrm{T}} B \text{ かつ } B \leq_{\mathrm{T}} A$$

とおくと $\equiv_{\mathrm{T}}$ は $\mathcal{P}(\omega)$ 上の同値関係になり，$\leq_{\mathrm{T}}$ は $\mathcal{D} := \mathcal{P}(\omega)/\equiv_{\mathrm{T}}$ 上の半順序になる．集合 $A \subseteq \omega$ の $\equiv_{\mathrm{T}}$ による同値類を $\deg(A)$ で表し，$A$ の Turing 次数 (Turing degree) と呼ぶ．Turing 次数は決定問題の難しさを表す量である．

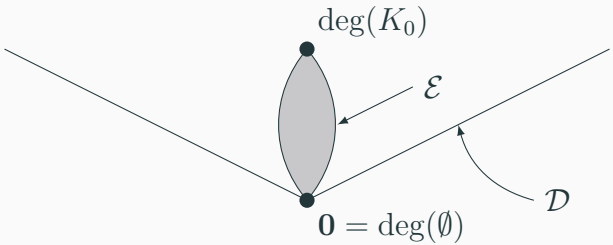
$(\mathcal{D}, \leq_{\mathrm{T}})$ が次の性質を持つことはすぐにわかる．

- 各 Turing 次数は可算個の元からなる (オラクルプログラムが可算個しかないから)．よって $\mathcal{D}$ は連続体濃度を持つ．
- $\mathbf{0} = \deg(\emptyset)$ は決定可能問題の全体に一致し，$\mathcal{D}$ の最小元を与える．
- $\mathbf{0} <_{\mathrm{T}} \deg(K_0)$.

# c.e. 次数

ある c.e. 集合の Turing 次数になっているものを c.e. 次数 (c.e. degree) といい，その全体を $\mathcal{E} \subseteq \mathcal{D}$ で表す．$\mathbf{0} = \deg(\emptyset), \deg(K_0) \in \mathcal{E}$ で $\mathbf{0} \neq \deg(K_0)$ だから，$\mathcal{E}$ は少なくとも $2$ つの元を持つ．

よって Post の問題は「$|\mathcal{E}| > 2$ か？」という問いと同値である．



# 2 節のまとめ

- 決定問題は自然数の集合 $A \subseteq \omega$ や，その特性関数 $\chi_A \colon \omega \to \{0, 1\}$ と同一視できる.
- 計算可能な戦略に沿って数を次々に投入していくことで作られる集合を c.e. 集合という (「入力に対し，〜は存在するか？」という決定問題に対応).
- $\chi_B$ を用いて $\chi_A$ を計算できること (あるオラクルプログラム $\Phi_e$ について $\chi_A = \Phi_e^B$ となること) を $A \leq_{\mathrm{T}} B$ で表す.
- 任意の c.e. 集合 $A$ に対し，$\emptyset \leq_{\mathrm{T}} A \leq_{\mathrm{T}} K_0$.
- Post の問題とは，

  $$\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} K_0 \text{ なる c.e. 集合 } A \text{ は存在するか？}$$

  という問いである.

## 3. 定理の証明

Friedberg-Muchnik の定理を有限害優先論法によって証明する.

# 比較不能な c.e. 集合を構成する

**定理 (Friedberg-Muchnik)**

$A \not\leq_{\mathrm{T}} B$ かつ $B \not\leq_{\mathrm{T}} A$ となるような 2 つの c.e. 集合 $A, B \subseteq \omega$ が存在する.

特に，この $A$ について $\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} K_0$ が成り立つ.

**やるべきこと**

構成したい $A, B$ は c.e. 集合だから，空箱に自然数を次々と投入していくことで構成すればよい．つまり，構成の第 $s$ ステージにおける $A$ を $A_s$ と書くと，

- $A_0 = \emptyset$ , ← 空箱からスタート
- 各 $s$ について $A_s \subseteq A_{s+1}$ , ← 一度投入した自然数は取り出せない
- 各 $A_s$ は 有限集合 , ← 各ステージでは有限個 (実際には高々 1 個) しか投入しない
- $A = \bigcup_{s \in \omega} A_s$.

$B, B_s$ についても同様.

# 比較不能な c.e. 集合を構成する

**定理 (Friedberg-Muchnik)**

$A \not\leq_{\mathrm{T}} B$ かつ $B \not\leq_{\mathrm{T}} A$ となるような 2 つの c.e. 集合 $A, B \subseteq \omega$ が存在する.

特に，この $A$ について $\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} K_0$ が成り立つ.

**やるべきこと**

構成したい $A, B$ は c.e. 集合だから，空箱に自然数を次々と投入していくことで構成すればよい．つまり，構成の第 $s$ ステージにおける $A$ を $A_s$ と書くと，

- $A_0 = \emptyset$ , 空箱からスタート
- 各 $s$ について $A_s \subseteq A_{s+1}$ , 一度投入した自然数は取り出せない
- 各 $A_s$ は 有限集合 , 各ステージでは有限個 (実際には高々 1 個) しか投入しない
- $A = \bigcup_{s \in \omega} A_s$.

$B, B_s$ についても同様.

# 困難は分割せよ

- $A \not\leq_{\mathrm{T}} B$ かつ $B \not\leq_{\mathrm{T}} A$ であることは，各 $e \in \omega$ に対し要件 (requirement)

  $\mathcal{R}_e^0$: $\chi_A \neq \Phi_e^B$, ← $A$ は $B$ から $\Phi_e$ によっては計算されない

  $\mathcal{R}_e^1$: $\chi_B \neq \Phi_e^A$ ← $B$ は $A$ から $\Phi_e$ によっては計算されない

  が成立することと同値.

- $\mathcal{R}_e^0$ が成立することは，ある $x \in \omega$ に対して $\Phi_e^B(x)\!\uparrow$ または $\chi_A(x) \neq \Phi_e^B(x)\!\downarrow$ となることと同値．この $x$ を $\mathcal{R}_e^0$ の成立の証拠 (witness) という．$\mathcal{R}_e^1$ についても同様.

- 全ての要件を成立させるような戦略を一足飛びに考えるのは大変なので，個々の要件 $\mathcal{R}_e^i$ を成立させるための小さな戦略 ($\mathcal{R}_e^i$ 戦略) を作り，それらを並列に稼動させることを考える.

# 困難は分割せよ

- $A \not\leq_{\mathrm{T}} B$ かつ $B \not\leq_{\mathrm{T}} A$ であることは，各 $e \in \omega$ に対し要件 (requirement)

$$\mathcal{R}_e^0 \colon \chi_A \neq \Phi_e^B, \qquad \mathcal{R}_e^1 \colon \chi_B \neq \Phi_e^A$$

  $A$ は $B$ から $\Phi_e$ によっては計算されない

  $B$ は $A$ から $\Phi_e$ によっては計算されない

  が成立することと同値．

- $\mathcal{R}_e^0$ が成立することは，ある $x \in \omega$ に対して $\Phi_e^B(x)\!\uparrow$ または $\chi_A(x) \neq \Phi_e^B(x)\!\downarrow$ となることと同値．この $x$ を $\mathcal{R}_e^0$ の成立の証拠 (witness) という．$\mathcal{R}_e^1$ についても同様．

- 全ての要件を成立させるような戦略を一足飛びに考えるのは大変なので，個々の要件 $\mathcal{R}_e^i$ を成立させるための小さな戦略 ($\mathcal{R}_e^i$ 戦略) を作り，それらを並列に稼動させることを考える．

# 困難は分割せよ

- $A \not\leq_{\mathrm{T}} B$ かつ $B \not\leq_{\mathrm{T}} A$ であることは，各 $e \in \omega$ に対し要件 (requirement)

$$\mathcal{R}_e^0 \colon \chi_A \neq \Phi_e^B, \qquad \mathcal{R}_e^1 \colon \chi_B \neq \Phi_e^A$$

  $A$ は $B$ から $\Phi_e$ によっては計算されない

  $B$ は $A$ から $\Phi_e$ によっては計算されない

  が成立することと同値.

- $\mathcal{R}_e^0$ が成立することは，ある $x \in \omega$ に対して $\Phi_e^B(x)\!\uparrow$ または $\chi_A(x) \neq \Phi_e^B(x)\!\downarrow$ となることと同値．この $x$ を $\mathcal{R}_e^0$ の成立の証拠 (witness) という．$\mathcal{R}_e^1$ についても同様.
- 全ての要件を成立させるような戦略を一足飛びに考えるのは大変なので，個々の要件 $\mathcal{R}_e^i$ を成立させるための小さな戦略 ($\mathcal{R}_e^i$ 戦略) を作り，それらを並列に稼動させることを考える.

$\mathcal{R}_e^0$ 戦略がどのようになっているべきかを考える．($\mathcal{R}_e^1$ についても同様)

**$\mathcal{R}_e^i$ 戦略 (理想)**

1. 証拠の候補 $x \in \omega$ を選ぶ．
2. $\Phi_e^{B_s}(x)\downarrow$ かどうかをチェックし，もしそうなら $\chi_{A_s}(x) \neq \Phi_e^{B_s}(x)$ となるように $A_s$ を変更する．

$B_s$ を変更しても $\Phi_e^{B_s}(x)$ の計算結果が変わるとは限らないので．

$\mathcal{R}_e^0$ 戦略がどのようになっているべきかを考える．($\mathcal{R}_e^1$ についても同様)

**$\mathcal{R}_e^i$ 戦略 (理想)**

1. 証拠の候補 $x \in \omega$ を選ぶ．
2. $\Phi_e^{B_s}(x)\downarrow$ かどうかをチェックし，もしそうなら $\chi_{A_s}(x) \neq \Phi_e^{B_s}(x)$ となるように $A_s$ を変更する．

ところが，$\Phi_e^{B_s}(x)\downarrow$ かどうかは ($K_0$ の決定不能性より) 有限ステップでは判定できない条件である．よって，「今のところ $\Phi_{e,s}^{B_s}(x)\downarrow$ であるかどうか」ということしか知ることはできない．$B_s$ が有限集合であることから $\Phi_{e,s}^{B_s}(x)\downarrow$ は実際に有限ステップで判定可能な条件である．

$\mathcal{R}_e^0$ 戦略がどのようになっているべきかを考える．($\mathcal{R}_e^1$ についても同様)

**$\mathcal{R}_e^i$ 戦略 (修正版 1)**

1. 証拠の候補 $x \in \omega$ を選ぶ．
2. $\Phi_{e,s}^{B_s}(x)\downarrow$ かどうかをチェックし，もしそうなら $\chi_{A_s}(x) \neq \Phi_e^{B_s}(x)$ となるように $A_s$ を変更する．そうでないなら，計算を続ける．

ところが，$\Phi_e^{B_s}(x)\downarrow$ かどうかは ($K_0$ の決定不能性より) 有限ステップでは判定できない条件である．よって，「今のところ $\Phi_{e,s}^{B_s}(x)\downarrow$ であるかどうか」ということしか知ることはできない．$B_s$ が有限集合であることから $\Phi_{e,s}^{B_s}(x)\downarrow$ は実際に有限ステップで判定可能な条件である．

$\mathcal{R}_e^0$ 戦略がどのようになっているべきかを考える．($\mathcal{R}_e^1$ についても同様)

**$\mathcal{R}_e^i$ 戦略 (修正版 1)**

1. 証拠の候補 $x \in \omega$ を選ぶ．
2. $\Phi_{e,s}^{B_s}(x)\downarrow$ かどうかをチェックし，もしそうなら $\chi_{A_s}(x) \neq \Phi_e^{B_s}(x)$ となるように $A_s$ を変更する．そうでないなら，計算を続ける．

また，一度 $A_s$ に投入した自然数を取り出すことはできないので，$\chi_{A_s}(x) = 1$ を $\chi_{A_{s+1}}(x) = 0$ に変更することはできない．つまり，$x \notin A_s$ だったものを $x \in A_{s+1}$ にする，という変更しかできない．よって証拠の候補 $x$ は $x \notin A_s$ となるように選ぶ必要がある．$A_s$ は有限集合なので，このような $x$ を必ず取ることができる．

$\mathcal{R}_e^0$ 戦略がどのようになっているべきかを考える．($\mathcal{R}_e^1$ についても同様)

**$\mathcal{R}_e^i$ 戦略 (修正版 2)**

1. 証拠の候補 $x \in \omega$ を $x \notin A_s$ となるように選ぶ．
2. $\Phi_{e,s}^{B_s}(x)\downarrow$ かどうかをチェックし，もしそうなら $\Phi_e^{B_s}(x) = 0$ のときは $x$ を $A_s$ に投入する．そうでないなら，計算を続ける．

また，一度 $A_s$ に投入した自然数を取り出すことはできないので，$\chi_{A_s}(x) = 1$ を $\chi_{A_{s+1}}(x) = 0$ に変更することはできない．つまり，$x \notin A_s$ だったものを $x \in A_{s+1}$ にする，という変更しかできない．よって証拠の候補 $x$ は $x \notin A_s$ となるように選ぶ必要がある．$A_s$ は有限集合なので，このような $x$ を必ず取ることができる．

# 無限個の戦略をひとつにまとめる

いま定義した $\mathcal{R}_e^i$ 戦略たちをひとつの戦略にまとめよう．

無限個の $\mathcal{R}_e^i$ 戦略を同時に動かすことはできないので，第 $s$ ステージでは $e \leq s$ なる $\mathcal{R}_e^i$ 戦略たちだけを稼動させることにする．

ところが，単に並列に稼動させるだけでは次のような問題が発生しうる．

**例**

1. $\Phi_{0,s}^{B_s}(x)\downarrow = 0$ となったので $\mathcal{R}_0^0$ 戦略が $x$ を $A_s$ に投入し，$\mathcal{R}_0^0$ が成立．
2. $\Phi_{0,s}^{A_{s+1}}(y)\downarrow = 0$ となったので $\mathcal{R}_0^1$ 戦略が $y$ を $B_{s+1}$ に投入し，$\mathcal{R}_0^1$ が成立．
3. ところが，$B_{s+1}$ に $y$ を投入したことによって，$\Phi_{0,s}^{B_{s+1}}(x)\downarrow = 1$ に変化してしまった！これにより要件 $\mathcal{R}_0^0$ の成立が害されて (injured) しまった．
4. $\mathcal{R}_0^0$ 戦略は再び証拠の候補 $x'$ を選び，$\Phi_{0,s}^{B_{s+1}}(x')\downarrow$ を待つ．

# 無限個の戦略をひとつにまとめる

いま定義した $\mathcal{R}_e^i$ 戦略たちをひとつの戦略にまとめよう．

無限個の $\mathcal{R}_e^i$ 戦略を同時に動かすことはできないので，第 $s$ ステージでは $e \leq s$ なる $\mathcal{R}_e^i$ 戦略たちだけを稼動させることにする．

ところが，単に並列に稼動させるだけでは次のような問題が発生しうる．

**例**

1. $\Phi_{0,s}^{B_s}(x)\downarrow = 0$ となったので $\mathcal{R}_0^0$ 戦略が $x$ を $A_s$ に投入し，$\mathcal{R}_0^0$ が成立．
2. $\Phi_{0,s}^{A_{s+1}}(y)\downarrow = 0$ となったので $\mathcal{R}_0^1$ 戦略が $y$ を $B_{s+1}$ に投入し，$\mathcal{R}_0^1$ が成立．
3. ところが，$B_{s+1}$ に $y$ を投入したことによって，$\Phi_{0,s}^{B_{s+1}}(x)\downarrow = 1$ に変化してしまった！これにより要件 $\mathcal{R}_0^0$ の成立が害されて (injured) しまった．
4. $\mathcal{R}_0^0$ 戦略は再び証拠の候補 $x'$ を選び，$\Phi_{0,s}^{B_{s+1}}(x')\downarrow$ を待つ．

したがって，単純に $\mathcal{R}_e^i$ 戦略を稼動させるだけでは，

1. $\mathcal{R}_0^0$ 戦略が元を投入し，$\mathcal{R}_0^0$ が成立
2. $\mathcal{R}_0^1$ 戦略が元を投入し，$\mathcal{R}_0^1$ が成立するも，$\mathcal{R}_0^0$ が害される
3. 再び $\mathcal{R}_0^0$ 戦略が元を投入し，$\mathcal{R}_0^0$ が成立するも，$\mathcal{R}_0^1$ が害される
4. ……

といったことが無限に繰り返され，最終的に要件が成立しないかもしれない．

このように，たった $2$ つの要件を考えただけでも，互いに相手の成立を害し合い続けて要件が成立しない可能性がある．

そこで，一度停止した計算が壊れないように保護 (protect) することを考える．

**観察**

$\Phi_{e,s}^{B_s}(x)\downarrow$ となるとき，計算時には $B_s$ は $\{0,\dots,s\}$ の範囲しか参照されていない．したがって，$B_s$ の $s+1$ 以降での値を変更しても $\Phi_{e,s}^{B_s}(x)$ の計算結果は変化しない．

そこで，ある $\mathcal{R}_e^0$ 戦略が $\Phi_e^{B_s}(x)\downarrow = 0$ を検出して要件を成立させたとき，

「これ以降 $B_s$ に $s$ 以下の自然数を投入しないでください」

という要請を他の戦略に通知し，まだ要件が成立していない戦略たちは証拠の候補を $s+1$ 以上になるように取り直す．

このように戦略を変更すれば，一度成立した要件が害されることがなくなるため，目的は達成されたかのように思える．しかし，この変更によって証拠を永遠に投入できない戦略が発生することがある．

**例**

1. $\mathcal{R}_0^0$ 戦略が $\Phi_{0,s}^{B_s}(x)\!\downarrow$ を待つ．
2. $\mathcal{R}_0^1$ 戦略が $\Phi_{0,s}^{A_s}(y_0)\!\downarrow = 0$ を検出し，$A_s$ に $s$ 以下の元を投入しないように通達する．$\mathcal{R}_0^0$ 戦略は証拠の候補を $x' > s$ となるように取り直す．
3. $\mathcal{R}_1^1$ 戦略が $\Phi_{1,s}^{A_{s+1}}(y_1)\!\downarrow = 0$ を検出し，$A_{s+1}$ に $s+1$ 以下の元を投入しないように通達する．$\mathcal{R}_0^0$ 戦略は証拠の候補を $x'' > s+1$ となるように取り直す．

このようにして，他の要件の成立を害さないように証拠の候補を取り直し続けた結果，最終的に要件成立の証拠を得られないことが起こりうる．

# 優先論法

そこで Friedberg と Muchnik が独立に考え出したのが，要件 $\mathcal{R}_e^i$ たちの間に優先度 (priority) を導入するというアイデアである．

優先論法 (priority argument) においては，要件たちの間に

$$\mathcal{R}_0^0 < \mathcal{R}_0^1 < \mathcal{R}_1^0 < \mathcal{R}_1^1 < \mathcal{R}_2^0 < \mathcal{R}_2^1 < \cdots < \mathcal{R}_e^0 < \mathcal{R}_e^1 < \cdots$$

という優先度が入る (左にある方が優先度が高い)．各 $\mathcal{R}_e^i$ 戦略には優先度によって次の制約が付く．

- 優先度が高い要件の戦略は，優先度の低い要件の戦略を害してもよい．
- 優先度が低い要件の戦略は，優先度の高い要件の戦略を害してはならない．

これにより，害し合い続けるイタチごっこや，譲り続けて証拠を投入できない事態を回避しようというというのが優先論法のアイデアである．

ポイント どの要件についても，自身より優先度の高い要件は有限個しかない．

# $\mathcal{R}_e^i$ 戦略の再定義 (1/2)

ここまでの考察に基づいて，$\mathcal{R}_e^i$ 戦略をきちんと定義しなおす．

**定義**

$\mathcal{R}_e^0$ 戦略について，「第 $s$ ステージ時点での証拠の候補」を $x^A(e,s)$ で表し，「第 $s$ ステージ時点で，これ以下の自然数を投入しないでほしいという値」を $r^B(e,s)$ で表す．$r^B(e,s)$ を束縛 (restraint) 関数という．

$\mathcal{R}_e^1$ 戦略については $A$ と $B$ の役割を入れ換えて定義する．

各 $e$ に対し，$x^A(e,0)\uparrow, r^B(e,0)=0$ と定義する．

$x^A(e,s), r^B(e,s)$ は無限にあるのでコンピューターでは扱えないように思えるが，各ステージ $s$ において，$1$ 以上の値になるものは有限個なので実際には有限の情報である．

# $\mathcal{R}_e^i$ 戦略の再定義 (2/2)

$\mathcal{R}_e^0$ 戦略についてのみ示す．$\mathcal{R}_e^1$ 戦略についても同様．

1. 証拠の候補 $x^A(e,s)$ を，
   - これまでにどの戦略の証拠の候補にも選ばれたことがなく，
   - $\max_{i<e} \max\{r^A(i,s), r^B(i,s)\}$ より大きい

   ようなものの中で最小のものと定める．
2. $\Phi_{e,s}^{B_s}(x^A(e,s))\downarrow$ を待つ．待っている間は $x^A(e,s+1) = x^A(e,s)$，$r^B(e,s+1) = r^B(e,s)$ のままにする．
3. $\Phi_{e,s}^{B_s}(x^A(e,s))\downarrow$ となったら，以下の行動 (action) を行う．
   - $\Phi_{e,s}^{B_s}(x^A(e,s))\downarrow = 0$ なら，$x^A(e,s)$ を $A_s$ に投入する．
   - $r^B(e,s+1) = s$ にする．
   - $j \geq e$ に対し，
     - 束縛が $r^A(e,s) \geq x^A(e,s)$ なる $\mathcal{R}_j^1$，
     - 証拠の候補 $x^B(j,s)$ が $r^B(e,s+1) = s$ 以下である $\mathcal{R}_j^1$

     をすべてリセット (reset) し (有限個しかない)，証拠の候補選びからやり直させる．

# 全ての要件が成立すること

以上の構成法のもとで，全ての要件 $\mathcal{R}_e^i$ が成立することを示す．

**証明.**

優先度に関する帰納法．$\mathcal{R}_0^0$ について，$\Phi_{0,s}^{B_s}(x^A(0,s))\downarrow$ となる $s$ が存在するかどうかで場合分けする．存在する場合，$s$ ステップ目で要件が成立し，$\mathcal{R}_0^0$ は最高の優先度を持つため以降害されることはない．存在しない場合，実際には $0$ ステップ目で要件が成立しており，以降他の要件を害することはない．

$\mathcal{R}_e^i$ について考えると，帰納法の仮定から，$\mathcal{R}_e^i$ より優先度の高い要件が全て成立しているようなステージ $t$ が存在する．$\Phi_{e,s}^{X_s}(x^A(0,s))\downarrow$ となる $s \geq t$ が存在するかどうかで場合分けする ($X_s$ は $A_s$ または $B_s$)．存在する場合，$s$ ステップ目で要件が成立し，$\mathcal{R}_e^i$ より優先度の高い要件は既に成立しているため他の要件を害さないことから，特に $\mathcal{R}_e^i$ も害されない．存在しない場合，実際には $t$ ステップ目で要件が成立しており，以降他の要件を害することはない． □

よって $A \not\leq_{\mathrm{T}} B, B \not\leq_{\mathrm{T}} A$ が成り立つ．

# 全ての要件が成立すること

以上の構成法のもとで，全ての要件 $\mathcal{R}_e^i$ が成立することを示す．

**証明.**

優先度に関する帰納法．$\mathcal{R}_0^0$ について，$\Phi_{0,s}^{B_s}(x^A(0,s))\downarrow$ となる $s$ が存在するかどうかで場合分けする．存在する場合，$s$ ステップ目で要件が成立し，$\mathcal{R}_0^0$ は最高の優先度を持つため以降害されることはない．存在しない場合，実際には $0$ ステップ目で要件が成立しており，以降他の要件を害することはない．

$\mathcal{R}_e^i$ について考えると，帰納法の仮定から，$\mathcal{R}_e^i$ より優先度の高い要件が全て成立しているようなステージ $t$ が存在する．$\Phi_{e,s}^{X_s}(x^A(0,s))\downarrow$ となる $s \geq t$ が存在するかどうかで場合分けする ($X_s$ は $A_s$ または $B_s$)．存在する場合，$s$ ステップ目で要件が成立し，$\mathcal{R}_e^i$ より優先度の高い要件は既に成立しているため他の要件を害さないことから，特に $\mathcal{R}_e^i$ も害されない．存在しない場合，実際には $t$ ステップ目で要件が成立しており，以降他の要件を害することはない． □

よって $A \not\leq_{\mathrm{T}} B, B \not\leq_{\mathrm{T}} A$ が成り立つ．

# 3 節のまとめ

- $\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} K_0$ なる c.e. 集合 $A$ の存在を示すために，比較不能な c.e. 集合 $A, B$ を構成したい．
- 無限個の要件 $\mathcal{R}_e^0 \colon \chi_A \neq \Phi_e^B, \mathcal{R}_e^1 \colon \chi_B \neq \Phi_e^A$ を成立させる必要がある．
- 各 $\mathcal{R}_e^i$ 戦略が好き勝手に証拠を投入していくと，お互いに成立を害し合い続けて永遠に要件が成立しないかもしれない．
- 一方で，「一旦誰かが要件を成立させたらその計算を壊さないようにする」という戦略では，他人に譲り続けて永遠に証拠を投入できない要件が出てくるかもしれない．
- そこで要件たちの間に優先度を導入し，下位の要件は上位の要件成立を保証する計算を壊してはいけないが，上位の要件は下位の要件を害してもよいとした．
- これにより各要件は有限回しか害されず，すべての要件が成立する．

## 4. 参考文献

おしまい．ご清聴ありがとうございました．

# 参考文献


- M. Sipser (太田和夫・田中圭介 監訳, 阿部正幸・植田広樹・藤岡淳・渡辺治訳), 計算理論の基礎 [原著第 2 版] 2. 計算可能性の理論, 共立出版, 2008.
- R. I. Soare, *Turing Computability: theory Theory and Applications*, Springer, 2016.
- S. B. Cooper, *Computability Theory*, CRC Press, 2004.
- y., 決定不能問題の話, 第 10 回関西すうがく徒のつどい, 2017, http://iso.2022.jp/math/tsudoi/10/slide.pdf.
- y., 2018 年の決定不能問題ギャラリーを振り返る, 第 3 回関東すうがく徒のつどい, 2017, http://iso.2022.jp/math/tsudoi/kanto3/slide.pdf.
- 決定不能問題ギャラリー, http://iso.2022.jp/math/undecidable-problems/.