

## 普遍チューリングマシン

全てのチューリングマシンの動作を模倣する

- 入力： $(\langle M \rangle, w)$ 
  - ★  $\langle M \rangle$ ：機械  $M$  の符号化（プログラムに相当）
  - ★  $w$ ： $M$  に与える入力データ
  
- 出力：機械  $M$  が入力  $w$  を受理するかどうか

## 普遍チューリングマシン

普遍チューリングマシンとは、  
言語

$$A_{\text{TM}} = \left\{ (\langle M \rangle, w) \mid \begin{array}{l} \langle M \rangle : \text{TM } M \text{ の符号化} \\ M \text{ が入力 } w \text{ を受理} \end{array} \right\}$$

を認識するチューリングマシン

普遍チューリングマシンが存在

$\iff A_{\text{TM}}$  がチューリングマシンで認識可能

## 定理

### 言語

$$A_{\text{TM}} = \left\{ (\langle M \rangle, w) \mid \begin{array}{l} \langle M \rangle : \text{TM } M \text{ の符号化} \\ M \text{ が入力 } w \text{ を受理} \end{array} \right\}$$

は認識可能だが、判定可能ではない。

証明は一種の対角線論法による  
(Russell のパラドックス風)

## $A_{TM}$ の判定不可能性

$A_{TM}$  を判定する TM  $U$  があったとする。

入力  $\langle M \rangle$  に対し、

- $M$  が  $\langle M \rangle$  を受理するなら拒否
- $M$  が  $\langle M \rangle$  を拒否するなら受理

となる TM  $D$  が ( $U$  を使って) 作れる。

これに、入力  $\langle D \rangle$  を喰わせよ。

対角線論法が出てきたついでに...

## 対角線論法の例：冪集合の濃度

集合  $X$  の冪集合 (power set)

$$\mathcal{P}(X) = \{S \mid S \subset X\}$$

について、

$$\#X \not\leq \#\mathcal{P}(X)$$

## 集合の濃度

集合  $A, B$  に対し、

$$A \prec B \iff \exists \iota : A \longrightarrow B : \text{単射}$$

$$\iff \exists \pi : B \longrightarrow A : \text{全射}$$

( $\iff$  には選択公理が必要)

$$A \sim B \iff \exists \varphi : A \longrightarrow B : \text{全単射}$$

$$\iff A \prec B \text{ かつ } B \prec A$$

(**Bernstein** の定理)

$\sim$  は “集合全体の集まり” の上の “同値関係”

## 集合の濃度

A の属する “同値類” : A の濃度 (cardinality)  
( $\#A, |A|, \text{card}(A)$  等と書く)

- $\aleph_0 = \aleph := \#\mathbb{N}$  : 可算濃度 (可付番濃度)  
(countable, enumerable)
- $\aleph = \mathfrak{c} := \#\mathbb{R}$  : 連続体濃度 (continuum)

濃度の比較 :  $\#A \leq \#B \iff A \prec B$

- $\#A \leq \#A$
  - $\#A \leq \#B, \#B \leq \#A \implies \#A = \#B$
  - $\#A \leq \#B, \#B \leq \#C \implies \#A \leq \#C$
- ( $\leq$  は濃度の間 “順序関係” である)

## 対角線論法の例：冪集合の濃度

集合  $X$  の冪集合 (power set)

$$\mathcal{P}(X) = \{S \mid S \subset X\}$$

について、

$$\#X \not\leq \#\mathcal{P}(X)$$

応用：

$$\begin{aligned} \#\mathbb{N} = \#\mathbb{Q} = \aleph_0 & \text{ (可算濃度) だが、} \\ \#\mathbb{R} = \#\mathcal{P}(\mathbb{N}) = \aleph & \not\leq \aleph_0 \text{ (連続体濃度)} \end{aligned}$$

注： $\aleph$  は  $\aleph_0$  の次の大きさ、とは言えない  
(連続体仮説)

## 対角線論法の例：冪集合の濃度

集合  $X$  の冪集合 (power set)

$$\mathcal{P}(X) = \{S \mid S \subset X\}$$

について、

$$\#X \precneq \#\mathcal{P}(X)$$

応用：

$$\begin{aligned} \#\mathbb{N} = \#\mathbb{Q} = \aleph_0 & \text{ (可算濃度) だが、} \\ \#\mathbb{R} = \#\mathcal{P}(\mathbb{N}) = \aleph & \succneq \aleph_0 \text{ (連続体濃度)} \end{aligned}$$

注： $\aleph$  は  $\aleph_0$  の次の大きさ、とは言えない  
(連続体仮説)

## 対角線論法の例：冪集合の濃度

集合  $X$  の冪集合 (power set)

$$\mathcal{P}(X) = \{S \mid S \subset X\}$$

について、

$$\#X \not\leq \#\mathcal{P}(X)$$

応用：

$$\begin{aligned} \#\mathbb{N} = \#\mathbb{Q} = \aleph_0 & \text{ (可算濃度) だが、} \\ \#\mathbb{R} = \#\mathcal{P}(\mathbb{N}) = \aleph & \not\leq \aleph_0 \text{ (連続体濃度)} \end{aligned}$$

注： $\aleph$  は  $\aleph_0$  の次の大きさ、とは言えない  
(連続体仮説)

## 定理

チューリングマシンで認識可能でない言語が存在する。

- チューリングマシン全体の集合
- 言語全体の集合

の濃度とを比較せよ

## 定理

チューリングマシンで認識可能でない言語が存在する。

- チューリングマシン全体の集合
- 言語全体の集合

の濃度とを比較せよ

さて、本講義最後の話題は、

## 計算量

について

問題の難しさを如何に計るか？

さて、本講義最後の話題は、

## 計算量

について

問題の難しさを如何に計るか？

## Church-Turing の提唱（再掲）

「全てのアルゴリズム（計算手順）は、  
チューリングマシンで実装できる」

（アルゴリズムと呼べるのは  
チューリングマシンで実装できるものだけ）

… 「アルゴリズム」の定式化

## 計算量 (complexity)

- **時間計算量**：計算に掛かるステップ数  
(TM での計算の遷移の回数)
- **空間計算量**：計算に必要なメモリ量  
(TM での計算で使うテープの区画数)

通常は、決まった桁数の四則演算 1 回を  
1 ステップと数えることが多い

入力データ長  $n$  に対する  
増加のオーダー (Landau の  $O$ -記号) で表す

## 計算量 (complexity)

- **時間計算量**：計算に掛かるステップ数  
(TM での計算の遷移の回数)
- **空間計算量**：計算に必要なメモリ量  
(TM での計算で使うテープの区画数)

通常は、決まった桁数の四則演算 1 回を  
1 ステップと数えることが多い

入力データ長  $n$  に対する  
増加のオーダー (Landau の  $O$ -記号) で表す

## 計算量 (complexity)

- **時間計算量**：計算に掛かるステップ数  
(TMでの計算の遷移の回数)
- **空間計算量**：計算に必要なメモリ量  
(TMでの計算で使うテープの区画数)

通常は、決まった桁数の四則演算 1 回を  
1 ステップと数えることが多い

入力データ長  $n$  に対する  
増加のオーダー (Landau の  $O$ -記号) で表す

## Landau の O-記号・o-記号

$f, g : \mathbb{N} \longrightarrow \mathbb{R}_{>0}$  に対し、

$$f = O(g) \iff \exists N \in \mathbb{N}, \exists C > 0 : \forall n \in \mathbb{N} : \\ (n \geq N \implies f(n) \leq Cg(n))$$

$$f = o(g) \iff \frac{f(n)}{g(n)} \longrightarrow 0 \quad (n \rightarrow \infty) \\ \iff \forall \varepsilon > 0 : \exists N \in \mathbb{N} : \forall n \in \mathbb{N} : \\ (n \geq N \implies f(n) \leq \varepsilon g(n))$$

## Landau の O-記号・o-記号

$f, g : \mathbb{N} \longrightarrow \mathbb{R}_{>0}$  に対し、

$$f = O(g) \iff \exists N \in \mathbb{N}, \exists C > 0 : \forall n \in \mathbb{N} : \\ (n \geq N \implies f(n) \leq Cg(n))$$

$$f = o(g) \iff \frac{f(n)}{g(n)} \longrightarrow 0 \quad (n \rightarrow \infty) \\ \iff \forall \varepsilon > 0 : \exists N \in \mathbb{N} : \forall n \in \mathbb{N} : \\ (n \geq N \implies f(n) \leq \varepsilon g(n))$$

## 計算量 (complexity)

問題を解くアルゴリズムによって決まる

… アルゴリズムの計算量

→ アルゴリズムの効率の評価

問題の計算量：

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

= どうしてもどれ位必要か

= どれ位難しい問題か

→ 問題の難しさの評価

## 計算量 (complexity)

問題を解くアルゴリズムによって決まる

… アルゴリズムの計算量

→ アルゴリズムの効率の評価

問題の計算量：

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

= どうしてもどれ位必要か

= どれ位難しい問題か

→ 問題の難しさの評価

## 計算量 (complexity)

問題を解くアルゴリズムによって決まる

… アルゴリズムの計算量

→ アルゴリズムの効率の評価

問題の計算量：

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

= どうしてもどれ位必要か

= どれ位難しい問題か

→ 問題の難しさの評価

## 計算量 (complexity)

問題を解くアルゴリズムによって決まる

… アルゴリズムの計算量

→ アルゴリズムの効率の評価

問題の計算量：

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

= どうしてもどれ位必要か

= どれ位難しい問題か

→ 問題の難しさの評価