計算理論　第 6 回
停止性問題

岡本 吉央
okamotoy@uec.ac.jp

電気通信大学

2020 年 11 月 12 日

最終更新：2020 年 11 月 16 日　00:09

この講義の主題

計算理論 (Theory of Computation)

- 計算可能性理論 (Computability Theory)
- 計算複雑性理論 (計算量理論) (Complexity Theory)

講義の進め方

- 前半：計算可能性理論 (担当：岡本)
- 後半：計算複雑性理論 (担当：垂井先生)

スケジュール 前半 (予定)

1. 計算とは何か？ (10/1)
2. 計算モデル (10/8)
3. チャーチ・チューリングの定立 (10/15)

⋆ 休み (体育祭) (10/22)

4. コード化 (10/29)
5. 計算可能性 (11/5)
6. 停止性問題 (11/12)
7. 再帰定理 (11/19)
8. 前半のまとめ (11/26)

注意：予定の変更もありうる

前回の話しと今日の話

前回の話：この講義のハイライト (1)

- WHILE 計算不可能な部分関数が存在することの証明 (対角線論法)
- 万能プログラム (インタプリタ) の設計

今日の話：この講義のハイライト (2)

- WHILE 計算不可能な「意義深い」関数の具体例 (停止性)
- プログラムの性質を問う関数の WHILE 計算不可能性
  - s-m-n 定理の利用

目次

1. 停止性問題
2. s-m-n 定理
3. 計算不可能性の証明法
4. 今日のまとめ

(復習) 万能関数

前回，万能関数を考察した

万能関数とは $\mathtt{univ}\colon \mathbb{N}^2 \nrightarrow \mathbb{N}$ で，

$$\mathtt{univ}(x_1, x_2) = \begin{cases} P(a_1, a_2, \ldots, a_k) & (x_1, x_2 \text{ が } \boxed{\text{条件} *} \text{ を満たすとき}) \\ \text{定義されない} & (\text{そうではないとき}) \end{cases}$$

$\boxed{\text{条件} *}$

- ある自然数 k に対して
  x_1 が k 入力 GOTO プログラム P のコードであり，
  x_2 が長さ k のリスト $a = (a_1, \ldots, a_k)$ のコードであり，
  P に a を入力したときに，P が停止する

前回，証明したこと

万能関数 univ は WHILE 計算可能

停止性関数

定義：停止性関数

停止性関数 (停止関数) とは $\mathtt{isHalting}\colon \mathbb{N}^2 \nrightarrow \mathbb{N}$ で，

$$\mathtt{isHalting}(x_1, x_2) = \begin{cases} 1 & (x_1, x_2 \text{ が } \boxed{\text{条件} *} \text{ を満たすとき}) \\ 0 & (\text{そうではないとき}) \end{cases}$$

$\boxed{\text{条件} *}$

- ある自然数 k に対して
  x_1 が k 入力 GOTO プログラム P のコードであり，
  x_2 が長さ k のリスト $a = (a_1, \dots, a_k)$ のコードであり，
  P に a を入力したときに，P が停止する

定義：停止性問題

プログラム P が停止性関数を計算するとき，
P は停止性問題 (停止問題) を解くという

停止性関数の計算不可能性

定理

停止性関数 `isHalting` は WHILE 計算不可能である

証明：`isHalting` が WHILE 計算可能であると仮定する

- `isHalting` を用いて，次の 1 入力 WHILE プログラム Q を作れる

```
x2 := isHalting(x1, x1);
IF x2 = 0
   THEN x0 := 0
   ELSE infloop(x1)
END
```

- Q は次の部分関数 $g\colon \mathbb{N} \nrightarrow \mathbb{N}$ を計算する

$$g(x_1) = \begin{cases} 0 & (\texttt{isHalting}(x_1, x_1) = 0 \text{ のとき}) \\ \text{定義されない} & (\text{それ以外のとき}) \end{cases}$$

停止性関数の計算不可能性：イメージ

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	$\cdots$
P_0								
P_1								
P_2								
P_3								
P_4								
P_5								

$\uparrow$ $\mathtt{isHalting}(\mathsf{enc}(P_i), \mathsf{enc}(P_j))$

停止性関数の計算不可能性：イメージ

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	$\cdots$
P_0	0	1	0	0	1	1	0	$\cdots$
P_1	0	0	1	0	0	1	0	$\cdots$
P_2	1	0	1	1	0	0	1	$\cdots$
P_3	0	0	0	0	1	0	1	$\cdots$
P_4	1	0	0	0	1	1	0	$\cdots$
P_5	0	0	1	0	1	0	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$

$\uparrow$ $\texttt{isHalting}(\mathsf{enc}(P_i), \mathsf{enc}(P_j))$

停止性関数の計算不可能性：イメージ

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	$\cdots$
P_0	0	1	0	0	1	1	0	$\cdots$
P_1	0	0	1	0	0	1	0	$\cdots$
P_2	1	0	1	1	0	0	1	$\cdots$
P_3	0	0	0	0	1	0	1	$\cdots$
P_4	1	0	0	0	1	1	0	$\cdots$
P_5	0	0	1	0	1	0	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$

$\uparrow$ $\mathtt{isHalting}(\mathsf{enc}(P_i), \mathsf{enc}(P_j))$

停止性関数の計算不可能性：イメージ

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	$\cdots$
P_0	0	1	0	0	1	1	0	$\cdots$
P_1	0	0	1	0	0	1	0	$\cdots$
P_2	1	0	1	1	0	0	1	$\cdots$
P_3	0	0	0	0	1	0	1	$\cdots$
P_4	1	0	0	0	1	1	0	$\cdots$
P_5	0	0	1	0	1	0	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	$\cdots$
P_0	$\downarrow$							
P_1		$\downarrow$						
P_2			$\uparrow$					
P_3				$\downarrow$				
P_4					$\uparrow$			
P_5						$\downarrow$		
$\vdots$							$\ddots$	

$\uparrow$ $\texttt{isHalting}(\mathsf{enc}(P_i), \mathsf{enc}(P_j))$ $g(\mathsf{enc}(P_i))$ $\uparrow$

停止性関数の計算不可能性 (続き)

- ここで，$x_1 = \mathsf{enc}(Q)$ とすると
  - $\texttt{isHalting}(\mathsf{enc}(Q), \mathsf{enc}(Q)) = 0$
    $\Leftrightarrow g(\mathsf{enc}(Q))\downarrow$ (g の定義)
    $\Leftrightarrow Q$ に $\mathsf{enc}(Q)$ を入力できて，停止する (Q と g の対応)
    $\Leftrightarrow \texttt{isHalting}(\mathsf{enc}(Q), \mathsf{enc}(Q)) = 1$ ($\texttt{isHalting}$ の定義)
- つまり，矛盾が導かれる $\square$

停止性関数の計算不可能性 (続き)

- ここで，$x_1 = \mathsf{enc}(Q)$ とすると
  - $\mathtt{isHalting}(\mathsf{enc}(Q), \mathsf{enc}(Q)) = 0$
    $\Leftrightarrow g(\mathsf{enc}(Q)) \downarrow$ (g の定義)
    $\Leftrightarrow Q$ に $\mathsf{enc}(Q)$ を入力できて，停止する (Q と g の対応)
    $\Leftrightarrow \mathtt{isHalting}(\mathsf{enc}(Q), \mathsf{enc}(Q)) = 1$ ($\mathtt{isHalting}$ の定義)
- つまり，矛盾が導かれる $\square$

$$g(x_1) = \begin{cases} 0 & (\mathtt{isHalting}(x_1, x_1) = 0 \text{ のとき}) \\ \text{定義されない} & (\text{それ以外のとき}) \end{cases}$$

停止性関数の計算不可能性 (続き)

- ここで，$x_1 = \mathsf{enc}(Q)$ とすると
  - $\mathtt{isHalting}(\mathsf{enc}(Q), \mathsf{enc}(Q)) = 0$
    $\Leftrightarrow g(\mathsf{enc}(Q))\downarrow$ (g の定義)
    $\Leftrightarrow Q$ に $\mathsf{enc}(Q)$ を入力できて，停止する (Q と g の対応)
    $\Leftrightarrow \mathtt{isHalting}(\mathsf{enc}(Q), \mathsf{enc}(Q)) = 1$ ($\mathtt{isHalting}$ の定義)
- つまり，矛盾が導かれる $\square$

$$\boxed{\begin{array}{l} x_2 := \mathtt{isHalting}(x_1, x_1); \\ \mathtt{IF}\ x_2 = 0\ \mathtt{THEN}\ x_0 := 0\ \mathtt{ELSE}\ \mathtt{infloop}(x_1)\ \mathtt{END} \end{array}}$$

停止性関数の計算不可能性 (続き)

- ここで，$x_1 = \mathsf{enc}(Q)$ とすると
  - $\mathtt{isHalting}(\mathsf{enc}(Q), \mathsf{enc}(Q)) = 0$
    $\Leftrightarrow g(\mathsf{enc}(Q))\downarrow$ （g の定義）
    $\Leftrightarrow Q$ に $\mathsf{enc}(Q)$ を入力できて，停止する （Q と g の対応）
    $\Leftrightarrow \mathtt{isHalting}(\mathsf{enc}(Q), \mathsf{enc}(Q)) = 1$ （$\mathtt{isHalting}$ の定義）
- つまり，矛盾が導かれる $\square$

$$\mathtt{isHalting}(x_1, x_2) = \begin{cases} 1 & (x_1, x_2 \text{ が } \boxed{\text{条件} *} \text{ を満たすとき}) \\ 0 & (\text{そうではないとき}) \end{cases}$$

停止万能関数の計算不可能性

次の関数 $\mathtt{univ}' \colon \mathbb{N}^2 \nrightarrow \mathbb{N}$ を考える

$$\mathtt{univ}'(x_1, x_2) = \begin{cases} P(a_1, \ldots, a_k) + 1 & (x_1, x_2 \text{ が } \boxed{\text{条件 } *} \text{ を満たすとき}) \\ 0 & (\text{そうではないとき}) \end{cases}$$

$\boxed{\text{条件 } *}$

- ある自然数 k に対して
  x_1 が k 入力 GOTO プログラム P のコードであり，
  x_2 が長さ k のリスト $a = (a_1, \ldots, a_k)$ のコードであり，
  P に a を入力したときに，P が停止する

定理

関数 $\mathtt{univ}'$ は WHILE 計算不可能である

証明のアイディア：$\mathtt{univ}'$ が計算できると，isHalting も計算できる
(つまり矛盾)

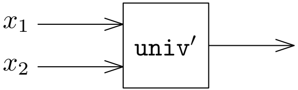
停止万能関数の計算不可能性：証明

証明：univ′ が計算できると仮定する

- このとき，次の 2 入力 WHILE プログラム P を構成できる

```
x3 := univ′(x1, x2);
IF x3 = 0 THEN x0 := 0 ELSE x0 := 1 END
```

- P は isHalting を計算する
  - $P(x_1, x_2)$ の出力が 0
    $\Leftrightarrow$ $\mathtt{univ}'(x_1, x_2) = 0$
    $\Leftrightarrow$ x_1, x_2 が $\boxed{\text{条件} *}$ を満たさない
    $\Leftrightarrow$ $\mathtt{isHalting}(x_1, x_2) = 0$ □



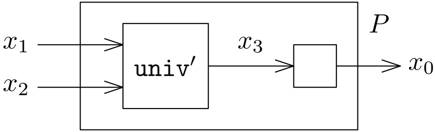
停止万能関数の計算不可能性：証明

証明：$\mathtt{univ}'$ が計算できると仮定する

- このとき，次の 2 入力 WHILE プログラム P を構成できる

$$\boxed{\begin{array}{l} x_3 := \mathtt{univ}'(x_1, x_2); \\ \mathtt{IF}\ x_3 = 0\ \mathtt{THEN}\ x_0 := 0\ \mathtt{ELSE}\ x_0 := 1\ \mathtt{END} \end{array}}$$

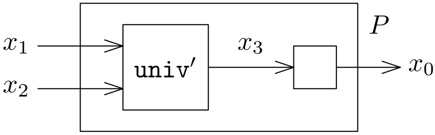
- P は isHalting を計算する
  - $P(x_1, x_2)$ の出力が 0
    $\Leftrightarrow \mathtt{univ}'(x_1, x_2) = 0$
    $\Leftrightarrow x_1, x_2$ が $\boxed{\text{条件} *}$ を満たさない
    $\Leftrightarrow \mathtt{isHalting}(x_1, x_2) = 0$ □



使っている手法：帰着 (還元)

univ′ の計算不可能性の証明では次の手法を使っている

- univ′ の計算をすることで，isHalting の計算をする
- isHalting の計算をするために，univ′ の計算を使う



これを次のように言うことがある

- isHalting を univ′ に帰着 (還元) する

目次

s-m-n 定理

目標

いろいろな (部分) 関数が WHILE 計算不可能であることを証明する

そのための道具として「s-m-n 定理」を用いる

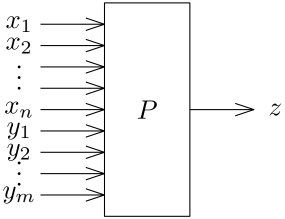
s-m-n 定理

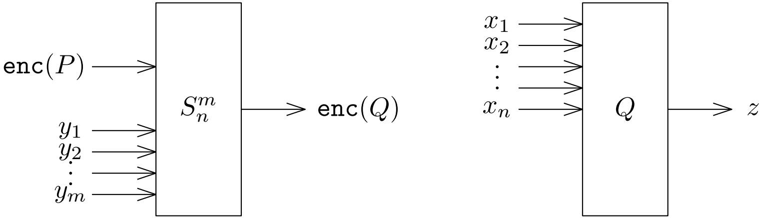
任意の自然数 m, n に対して，
次の性質を持つ $(m+1)$ 変数 WHILE 計算可能関数 $S^m_n \colon \mathbb{N}^{m+1} \to \mathbb{N}$ が存在

1. 任意の自然数 $e, y_1, \ldots, y_m$ に対して
$S^m_n(e, y_1, \ldots, y_m)$ は n 入力 GOTO プログラムのコードである
2. 任意の自然数 $e, x_1, \ldots, x_n, y_1, \ldots, y_m$ に対して
$\mathtt{univ}(S^m_n(e, y_1, \ldots, y_m), \mathsf{enc}((x_1, \ldots, x_n)))$
$= \mathtt{univ}(e, \mathsf{enc}((x_1, \ldots, x_n, y_1, \ldots, y_m)))$

s-m-n 定理：イメージ

$e = \mathrm{enc}(P)$ とする





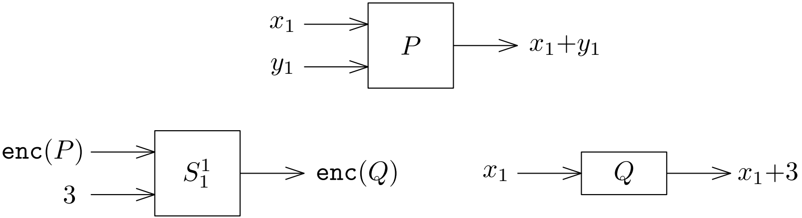
s-m-n 定理：イメージの例

- プログラム P が $\mathtt{add}\colon \mathbb{N}^2 \to \mathbb{N}$ を計算するとする

$$\mathtt{add}(x_1, y_1) = x_1 + y_1$$

- $S^1_1(\mathsf{enc}(P), 3)$ は次の関数 $\mathtt{add}_3$ を計算するプログラム Q のコード

$$\mathtt{add}_3(x_1) = x_1 + 3$$



s-m-n 定理：証明のイメージ

add を計算するプログラム

```
L1: IF x1 = 0 THEN GOTO L5;
L2: x0 := x0 + 1;
L3: x1 := x1 - 1;
L4: GOTO L1;
L5: IF x2 = 0 THEN GOTO L9;
L6: x0 := x0 + 1;
L7: x2 := x2 - 1;
L8: GOTO L5;
L9: HALT
```

$S^1_1(\mathsf{enc}(\mathsf{add}), y_1)$ をコードとするプログラム

```
L1: IF y1 = 0 THEN GOTO L5;
L2: x2 := x2 + 1;
L3: y1 := y1 - 1;
L4: GOTO L1;
L5: IF x1 = 0 THEN GOTO L9;
L6: x0 := x0 + 1;
L7: x1 := x1 - 1;
L8: GOTO L5;
L9: IF x2 = 0 THEN GOTO L13;
L10: x0 := x0 + 1;
L11: x2 := x2 - 1;
L12: GOTO L9;
L13: HALT
```

目次

全域性問題の計算不可能性

定義：全域性関数

全域性関数とは $\mathtt{isTotal}\colon \mathbb{N} \nrightarrow \mathbb{N}$ で,

$$\mathtt{isTotal}(x_1) = \begin{cases} 1 & (x_1 \text{ が全域関数を計算する GOTO プログラムの} \\ & \quad \text{コードであるとき}) \\ 0 & (\text{そうではないとき}) \end{cases}$$

証明すること

関数 isTotal は WHILE 計算可能ではない

証明：WHILE 計算可能であると仮定して矛盾を導く

全域性関数の計算不可能性：s-m-n 定理の利用

- 次の 2 入力プログラム P を考える

$$\boxed{\texttt{univ}(x_1, x_2)}$$

- 次の 2 入力プログラム Q を考える

$$
\begin{array}{|l|}
\hline
x_3 := S^2_0(\mathsf{enc}(P), x_1, x_2); \\
x_4 := \texttt{isTotal}(x_3); \\
\texttt{IF } x_4 = 0 \texttt{ THEN } x_0 := 0 \texttt{ ELSE } x_0 := 1 \texttt{ END} \\
\hline
\end{array}
$$

- このとき，Q は $\texttt{isHalting}$ を計算する

$\because$ Q の出力が 1
$\Leftrightarrow$ $\texttt{isTotal}(x_3) = 1$
$\Leftrightarrow$ $S^2_0(\mathsf{enc}(P), x_1, x_2)$ をコードとするプログラムは全域関数を計算する
$\Leftrightarrow$ $\texttt{univ}(x_1, x_2)$ の計算が停止する
$\Leftrightarrow$ $\texttt{isHalting}(x_1, x_2) = 1$ $\square$

2 倍判定問題の計算不可能性

次の 1 変数関数 double: $\mathbb{N} \to \mathbb{N}$ を考える

$$\texttt{double}(x_1) = 2x_1$$

そして，次の 1 変数関数 isDouble: $\mathbb{N} \to \mathbb{N}$ を考える

$$\texttt{isDouble}(x_1) = \begin{cases} 1 & (x_1 \text{ が double を計算するプログラムの} \\ & \quad \text{コードであるとき}) \\ 0 & (\text{そうではないとき}) \end{cases}$$

証明すること

関数 isDouble は WHILE 計算可能ではない

証明：WHILE 計算可能であると仮定して矛盾を導く

2 倍判定問題の計算不可能性：s-m-n 定理の利用

- 次の 3 入力プログラム P を考える

$$\boxed{\texttt{univ}(x_1, x_2);\ x_0 := \texttt{double}(x_3)}$$

- 次の 2 入力プログラム Q を考える

$$\boxed{\begin{array}{l} x_4 := S^2_1(\mathsf{enc}(P), x_1, x_2); \\ x_5 := \texttt{isDouble}(x_3); \\ \texttt{IF}\ x_5 = 0\ \texttt{THEN}\ x_0 := 0\ \texttt{ELSE}\ x_0 := 1\ \texttt{END} \end{array}}$$

- このとき，Q は `isHalting` を計算する

  $\because$ Q の出力が 1

  $\Leftrightarrow$ $\texttt{isDouble}(x_4) = 1$

  $\Leftrightarrow$ $S^2_1(\mathsf{enc}(P), x_1, x_2)$ をコードとするプログラムは `double` を計算する

  $\Leftrightarrow$ $\texttt{univ}(x_1, x_2)$ の計算が停止する

  $\Leftrightarrow$ $\texttt{isHalting}(x_1, x_2) = 1$ $\square$

目次

今日のまとめ と 次回の予告

今日の話：この講義のハイライト (2)

- WHILE 計算不可能な「意義深い」関数の具体例 (停止性)
- プログラムの性質を問う関数の WHILE 計算不可能性
  - s-m-n 定理の利用

次回の予告

- 再帰定理：WHILE プログラムで再帰を実現する方法
- 再帰定理の応用

目次

1. 停止性問題
2. s-m-n 定理
3. 計算不可能性の証明法
4. 今日のまとめ