

計算理論 第2回 計算モデル

岡本 吉央
okamotoy@uec.ac.jp

電気通信大学

2020年10月8日

最終更新：2020年10月7日 16:23

岡本 吉央 (電通大)

計算理論 (2)

2020年10月8日

1 / 32

概要

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

岡本 吉央 (電通大)

計算理論 (2)

2020年10月8日

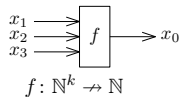
3 / 32

概要

この講義で扱う計算

この講義で扱う計算対象

k 個の自然数を入力として、1 個の自然数を出力する部分関数



定義：部分関数

$f \subseteq A \times B$ が A から B への部分関数であるとは、任意の $a \in A$ に対して、 $(a, b) \in f$ となる $b \in B$ が存在しないか、存在するとしてもただ1つであること

ただし、 A, B は集合

岡本 吉央 (電通大)

計算理論 (2)

2020年10月8日

5 / 32

WHILE プログラム

目次

- 1 WHILE プログラム
- 2 WHILE 計算可能性
- 3 糖衣構文
- 4 WHILE 計算可能な部分関数の例
- 5 今日のまとめ

岡本 吉央 (電通大)

計算理論 (2)

2020年10月8日

7 / 32

概要

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)

岡本 吉央 (電通大)

計算理論 (2)

2020年10月8日

2 / 32

概要

計算モデル

計算モデルとは？ (直感的な定義)

「計算主体」を数学的に抽象化したもの

代表的な計算モデル

- ▶ チューリング機械
- ▶ ランダム・アクセス機械
- ▶ カウンタ機械
- ▶ ポインタ機械
- ▶ タグ・システム
- ▶ 帰納的関数 (μ 再帰関数)
- ▶ ラムダ計算
- ▶ マルコフ・アルゴリズム

この講義 (の前半) では

- ▶ ある「単純化したプログラミング言語」を計算モデルとして用いる
↪ 今回
- ▶ 計算というときは、自然数の組を自然数にうつす部分関数の計算を考察対象とする
↪ 前回

岡本 吉央 (電通大)

計算理論 (2)

2020年10月8日

4 / 32

概要

今日の目標

今日の目標

- ▶ 単純化したプログラミング言語 (WHILE プログラム) を通じて計算モデルの例を理解する
- ▶ WHILE プログラムによるプログラミングができるようになる

重要な概念

- ▶ 計算可能性 (computability)

岡本 吉央 (電通大)

計算理論 (2)

2020年10月8日

6 / 32

WHILE プログラム

WHILE プログラム (1)

機能をそぎ落として、本質的な部分だけを残したプログラミング言語

データ型

- ▶ 自然数 $0, 1, 2, \dots$ (\mathbb{N} の要素)

構成要素

変数

- ▶ x_0, x_1, x_2, \dots

記号

- ▶ $:=, ,, \text{WHILE}, \text{DO}, \text{END}$
- ▶ $\neq 0, +1, -1$

WHILE プログラムの例

```
WHILE  $x_1 \neq 0$  DO
   $x_0 := x_0 + 1;$ 
   $x_1 := x_1 - 1$ 
END;
WHILE  $x_2 \neq 0$  DO
   $x_0 := x_0 + 1;$ 
   $x_2 := x_2 - 1$ 
END
```

岡本 吉央 (電通大)

計算理論 (2)

2020年10月8日

8 / 32

WHILE プログラムの文法 (構文論)

WHILE プログラムは再帰的に定義される

- 1 変数 x に対して, 「 $x := x + 1$ 」は WHILE プログラム
- 2 変数 x に対して, 「 $x := x - 1$ 」は WHILE プログラム
- 3 WHILE プログラム P_1, P_2 に対して, 「 $P_1; P_2$ 」は WHILE プログラム
- 4 変数 x と WHILE プログラム P に対して, 「 $\text{WHILE } x \neq 0 \text{ DO } P \text{ END}$ 」は WHILE プログラム

```
WHILE  $x_1 \neq 0$  DO
   $x_0 := x_0 + 1;$ 
   $x_1 := x_1 - 1$ 
END;
WHILE  $x_2 \neq 0$  DO
   $x_0 := x_0 + 1;$ 
   $x_2 := x_2 - 1$ 
END
```

```
 $x := x + 1$ 
```

↪ x の値を 1 だけ増やす

```
 $x := x - 1$ 
```

↪ x の値を 1 だけ減らす

x の値が 0 であるとき, x の値を 0 のままにする

```
 $P_1; P_2$ 
```

↪ P_1 を実行してから, P_2 を実行する

```
WHILE  $x \neq 0$  DO  $P$  END
```

↪ x が 0 でない限り, P の実行を続ける

注

- ▶ 変数 x_1, \dots, x_k は入力の値で初期化される
- ▶ その他の変数は 0 で初期化される
- ▶ 変数 x_0 が出力を表す

入力 $x_1 = 2, x_2 = 1$ として, 実行してみる

```
WHILE  $x_1 \neq 0$  DO
   $x_0 := x_0 + 1;$ 
   $x_1 := x_1 - 1$ 
END;
WHILE  $x_2 \neq 0$  DO
   $x_0 := x_0 + 1;$ 
   $x_2 := x_2 - 1$ 
END
```

- ▶ $x_0 = 0$
- ▶ $x_1 = 2$
- ▶ $x_2 = 1$

∴ 出力は 3

このプログラムは関数 $f(x_1, x_2) = x_1 + x_2$ を計算する

```
WHILE  $x_1 \neq 0$  DO
   $x_0 := x_0 + 1;$ 
   $x_1 := x_1 - 1$ 
END;
WHILE  $x_2 \neq 0$  DO
   $x_0 := x_0 + 1;$ 
   $x_2 := x_2 - 1$ 
END
```

目次

- 1 WHILE プログラム
- 2 WHILE 計算可能性
- 3 糖衣構文
- 4 WHILE 計算可能な部分関数の例
- 5 今日のまとめ

部分関数 $f: \mathbb{N}^k \rightarrow \mathbb{N}$

定義: WHILE 計算可能性

f が WHILE 計算可能であるとは, 次を満たす k 入力 WHILE プログラム P が存在すること

$$f(x_1, \dots, x_k) \downarrow \Rightarrow P \text{ の出力は } f(x_1, \dots, x_k)$$

$$f(x_1, \dots, x_k) \uparrow \Rightarrow P \text{ は停止しない}$$

このとき, P は f を計算するという

先の例より, 次の (部分) 関数 $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ は WHILE 計算可能

$$f(x_1, x_2) = x_1 + x_2$$

今から行うこと

- ▶ WHILE 計算可能な部分関数の例をいくつか見る

例題 1

次の部分関数 $\text{infloop}: \mathbb{N} \rightarrow \mathbb{N}$ は WHILE 計算可能か?

任意の $x_1 \in \mathbb{N}$ に対して, $\text{infloop}(x_1) \uparrow$

解答例: WHILE 計算可能である.

実際, 次の WHILE プログラムが infloop を計算する. □

```
 $x_1 := x_1 + 1;$ 
WHILE  $x_1 \neq 0$  DO
   $x_1 := x_1 + 1$ 
END
```

注意

本来はこのプログラムが確かに infloop を計算すると証明しないといけない

(↪ 『プログラミング言語論』や『アルゴリズム論』の範疇)

目次

- 1 WHILE プログラム
- 2 WHILE 計算可能性
- 3 糖衣構文
- 4 WHILE 計算可能な部分関数の例
- 5 今日のまとめ

WHILE プログラムの利点

- ▶ とてもシンプル
- ▶ プログラムに対して証明を行うことが易しい

WHILE プログラムの欠点

- ▶ 機能が限られる (ように見える)
- ▶ プログラムを構成することが難しい

今から行うこと

欲しい機能を WHILE プログラムとして書く方法を与える

↪ 糖衣構文 (syntax sugar, syntactic sugar)

定数を代入する機能

変数 x_i の値を 3 とする機能を

$$x_i := 3$$

と書くことで実現したい

$$x_i := 3$$

↪

$$\begin{array}{l} x_i := 0; \\ x_i := x_i + 1; \\ x_i := x_i + 1; \\ x_i := x_i + 1 \end{array}$$

関数の呼び出し

x_i, x_j を入力して, $x_i + x_j$ を出力する機能を $\text{add}(x_i, x_j)$ で書くとき

$$x_3 := \text{add}(x_4, x_5)$$

のように書くことを可能にしたい

$$x_3 := \text{add}(x_4, x_5)$$

↪

$$\begin{array}{l} x_3 := x_4; \\ y := x_5; \\ \text{WHILE } y \neq 0 \text{ DO} \\ \quad x_3 := x_3 + 1; \\ \quad y := y - 1 \\ \text{END} \end{array}$$

ただし, y はプログラムに出てこない変数

注意: この糖衣構文では、「関数の再帰呼び出し」はできない

条件分岐の機能

WHILE プログラム P_1, P_2 と変数 x に対して,

$$\text{IF } x \neq 0 \text{ THEN } P_1 \text{ ELSE } P_2 \text{ END}$$

と書いて, $x \neq 0$ ならば P_1 を実行し, $x = 0$ ならば P_2 を実行するという機能を実現したい

$$\begin{array}{l} \text{IF } x \neq 0 \text{ THEN} \\ \quad P_1 \\ \text{ELSE} \\ \quad P_2 \\ \text{END} \end{array}$$

↪

$$\begin{array}{l} y := 1; \\ \text{IF } x \neq 0 \text{ THEN} \\ \quad P_1; y := 0 \\ \text{END;} \\ \text{IF } y \neq 0 \text{ THEN} \\ \quad P_2 \\ \text{END} \end{array}$$

ただし, y はプログラムに出てこない変数

初期化の機能

変数 x_i の値を 0 とする機能を

$$x_i := 0$$

と書くことで実現したい

$$x_i := 0$$

↪

$$\begin{array}{l} \text{WHILE } x_i \neq 0 \text{ DO} \\ \quad x_i := x_i - 1 \\ \text{END} \end{array}$$

代入の機能

変数 x_i の値を別の変数 x_j に代入する機能を

$$x_j := x_i$$

と書くことで実現したい

$$x_j := x_i$$

↪

$$\begin{array}{l} x_j := 0; \\ \text{WHILE } x_i \neq 0 \text{ DO} \\ \quad y := y + 1; y' := y' + 1; x_i := x_i - 1 \\ \text{END;} \\ \text{WHILE } y \neq 0 \text{ DO } x_j := x_j + 1; y := y - 1 \text{ END;} \\ \text{WHILE } y' \neq 0 \text{ DO } x_i := x_i + 1; y' := y' - 1 \text{ END} \end{array}$$

ただし, y, y' はプログラムに出てこない変数

条件分岐の機能

WHILE プログラム P と変数 x に対して,

$$\text{IF } x \neq 0 \text{ THEN } P \text{ END}$$

と書いて, $x \neq 0$ ならば P を実行するという機能を実現したい

$$\begin{array}{l} \text{IF } x \neq 0 \text{ THEN} \\ \quad P \\ \text{END} \end{array}$$

↪

$$\begin{array}{l} y := x; \\ \text{WHILE } y \neq 0 \text{ DO} \\ \quad P; \\ \quad y := 0 \\ \text{END} \end{array}$$

ただし, y はプログラムに出てこない変数

- 1 WHILE プログラム
- 2 WHILE 計算可能性
- 3 糖衣構文
- 4 WHILE 計算可能な部分関数の例
- 5 今日のまとめ

例題 2

次の部分関数 $\text{mult}: \mathbb{N}^2 \rightarrow \mathbb{N}$ は WHILE 計算可能か？

任意の $x_1, x_2 \in \mathbb{N}$ に対して, $\text{mult}(x_1, x_2) = x_1 x_2$

解答例：WHILE 計算可能である。

実際、次のプログラムが mult を計算する。

```
WHILE  $x_1 \neq 0$  DO
   $x_0 := \text{add}(x_0, x_2)$ ;
   $x_1 := x_1 - 1$ 
END
```

▶ ただし、これは糖衣構文を含むので、WHILE プログラムではない
→ 糖衣構文を書き換えて、WHILE プログラムにできる □

例題 3

次の部分関数 $\text{isEven}: \mathbb{N} \rightarrow \mathbb{N}$ は WHILE 計算可能か？

任意の $x_1 \in \mathbb{N}$ に対して, $\text{isEven}(x_1) = \begin{cases} 1 & (x_1 \text{ が偶数のとき}) \\ 0 & (x_1 \text{ が奇数のとき}) \end{cases}$

解答例：WHILE 計算可能である。

実際、次のプログラムが isEven を計算する。

```
 $x_0 := 1$ ;
WHILE  $x_1 \neq 0$  DO
   $x_1 := x_1 - 1$ ;
  IF  $x_1 \neq 0$  THEN
     $x_0 := x_0 - 1$ ;
  IF  $x_1 \neq 0$  THEN
     $x_2 := x_2 + 1$ 
  ELSE
     $x_0 := 1$ 
  END;
  ELSE
     $x_0 := 0$ 
  END
END
```

注意

部分関数 $f: \mathbb{N}^k \rightarrow \mathbb{N}$ が WHILE 計算可能であるとき、 f を計算する WHILE プログラムは無数に存在する

例：関数 $\text{add}(x_1, x_2) = x_1 + x_2$ を計算する 2 つの WHILE プログラム

```
WHILE  $x_1 \neq 0$  DO
   $x_0 := x_0 + 1$ ;
   $x_1 := x_1 - 1$ 
END;
WHILE  $x_2 \neq 0$  DO
   $x_0 := x_0 + 1$ ;
   $x_2 := x_2 - 1$ 
END

WHILE  $x_1 \neq 0$  DO
   $x_0 := x_0 + 1$ ;
   $x_1 := x_1 - 1$ 
END;
WHILE  $x_2 \neq 0$  DO
   $x_0 := x_0 + 1$ ;
   $x_2 := x_2 - 1$ 
END;
 $x_1 := x_1 + 1$ 
```

今日の目標

- ▶ 単純化したプログラミング言語 (WHILE プログラム) を通じて計算モデルの例を理解する
- ▶ WHILE プログラムによるプログラミングができるようになる

重要な概念

- ▶ 計算可能性 (computability)

次回の予告

- ▶ 別の計算モデル (簡単なプログラミング言語の例) を紹介する
- ▶ 計算モデルによって計算可能性が変わるか／変わらないか考察する

次の WHILE プログラムが mult を計算する

```
WHILE  $x_1 \neq 0$  DO
  WHILE  $x_1 \neq 0$  DO
     $x_3 := x_3 + 1$ ;  $x_4 := x_4 + 1$ ;  $x_1 := x_1 - 1$ 
  END;
  WHILE  $x_3 \neq 0$  DO  $x_5 := x_5 + 1$ ;  $x_3 := x_3 - 1$  END;
  WHILE  $x_4 \neq 0$  DO  $x_2 := x_2 + 1$ ;  $x_4 := x_4 - 1$  END;
  WHILE  $x_5 \neq 0$  DO  $x_0 := x_0 + 1$ ;  $x_5 := x_5 - 1$  END;
   $x_1 := x_1 - 1$ 
END
```

例題 3

次の部分関数 $\text{half}: \mathbb{N} \rightarrow \mathbb{N}$ は WHILE 計算可能か？

任意の $x_1 \in \mathbb{N}$ に対して,

$\text{half}(x_1) = \begin{cases} x_1/2 & (x_1 \text{ が偶数のとき}) \\ \text{定義されない} & (x_1 \text{ が奇数のとき}) \end{cases}$

解答例：WHILE 計算可能である。

実際、次のプログラムが half を計算する。

```
 $x_2 := \text{isEven}(x_1)$ ;
IF  $x_2 \neq 0$  THEN
  WHILE  $x_1 \neq 0$  DO
     $x_0 := x_0 + 1$ ;
     $x_1 := x_1 - 1$ ;
  ELSE
     $x_1 := x_1 - 1$ 
  END
ELSE
   $x_0 := \text{infloop}(x_1)$ 
END
```

- 1 WHILE プログラム
- 2 WHILE 計算可能性
- 3 糖衣構文
- 4 WHILE 計算可能な部分関数の例
- 5 今日のまとめ