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

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)

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

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

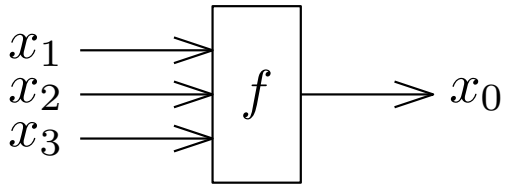
この講義 (の前半) では

- 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
  $\rightsquigarrow$ 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
  $\rightsquigarrow$ 前回

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



$$f : \mathbb{N}^k \nrightarrow \mathbb{N}$$

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

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) を通し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)

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 (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 $0, 1, 2, \ldots$ ($\mathbb{N}$ の要素)

構成要素

変数

- $x_0, x_1, x_2, \ldots$

記号

- $:=$, ;, WHILE, DO, END
- $\neq 0$, $+1$, -1

WHILE プログラムの例

```
WHILE x_1 ≠ 0 DO
    x_0 := x_0 + 1;
    x_1 := x_1 − 1
END;
WHILE x_2 ≠ 0 DO
    x_0 := x_0 + 1;
    x_2 := x_2 − 1
END
```

WHILE プログラム (2) : 文法 (構文論)

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,
   「WHILE $x \neq 0$ DO P END」は
   WHILE プログラム

```
WHILE x_1 ≠ 0 DO
    x_0 := x_0 + 1;
    x_1 := x_1 − 1
END;
WHILE x_2 ≠ 0 DO
    x_0 := x_0 + 1;
    x_2 := x_2 − 1
END
```

WHILE プログラム (4)：意味論

$\boxed{x := x + 1}$

$\rightsquigarrow$ x の値を 1 だけ増やす

$\boxed{x := x - 1}$

$\rightsquigarrow$ 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

$\boxed{P_1; P_2}$

$\rightsquigarrow$ P_1 を実行してから，P_2 を実行する

$\boxed{\texttt{WHILE } x \neq 0 \texttt{ DO } P \texttt{ END}}$

$\rightsquigarrow$ x が 0 でない限り，P の実行を続ける

注

- 変数 $x_1, \ldots, x_k$ 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

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO ←
    x0 := x0 + 1;
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

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

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1; ←
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

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

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1 ←
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

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

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END; ←
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

- $x_0 = 1$
- $x_1 = 1$
- $x_2 = 1$

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;   ←
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

- $x_0 = 1$
- $x_1 = 1$
- $x_2 = 1$

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1 ←
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

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

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END; ←
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

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

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO ←
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

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

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1; ←
    x2 := x2 − 1
END
```

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

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1 ←
END
```

- $x_0 = 3$
- $x_1 = 0$
- $x_2 = 1$

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END ←
```

- $x_0 = 3$
- $x_1 = 0$
- $x_2 = 0$

WHILE プログラム (5)：例の実行

入力 $x_1 = 2, x_2 = 1$ として，実行してみる

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

- $x_0 = 3$
- $x_1 = 0$
- $x_2 = 0$

$\therefore$ 出力は 3

和を計算する WHILE プログラム

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 x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 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

WHILE 計算可能性

部分関数 $f\colon \mathbb{N}^k \rightharpoonup \mathbb{N}$

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

$$\begin{aligned} f(x_1,\ldots,x_k)\downarrow &\quad\Rightarrow\quad P \text{ の出力は } f(x_1,\ldots,x_k) \\ f(x_1,\ldots,x_k)\uparrow &\quad\Rightarrow\quad P \text{ は停止しない} \end{aligned}$$

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 $f\colon \mathbb{N}^2 \rightharpoonup \mathbb{N}$ 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

WHILE 計算可能性：例題 1

例題 1

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

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

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． □

$$
\begin{array}{l}
x_1 := x_1 + 1; \\
\mathtt{WHILE}\ x_1 \neq 0\ \mathtt{DO} \\
\quad x_1 := x_1 + 1 \\
\mathtt{END}
\end{array}
$$

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
($\rightsquigarrow$『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欠点

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

$\rightsquigarrow$ 糖衣構文 (syntax sugar，syntactic sugar)

初期化

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

$x_i := 0$

$\rightsquigarrow$

```
WHILE x_i ≠ 0 DO
    x_i := x_i − 1
END
```

定数の代入

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

$x_i := 3$ $\rightsquigarrow$

$$\begin{aligned} &x_i := 0; \\ &x_i := x_i + 1; \\ &x_i := x_i + 1; \\ &x_i := x_i + 1 \end{aligned}$$

定数の代入

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

$x_i := 3$ $\rightsquigarrow$

```
WHILE x_i ≠ 0 DO
    x_i := x_i − 1
END;
x_i := x_i + 1;
x_i := x_i + 1;
x_i := x_i + 1
```

代入

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

$x_j := x_i$ $\rightsquigarrow$

$x_j := 0;$
WHILE $x_i \neq 0$ DO
$\quad y := y + 1;\ y' := y' + 1;\ x_i := x_i - 1$
END;
WHILE $y \neq 0$ DO $x_j := x_j + 1;\ y := y - 1$ END;
WHILE $y' \neq 0$ DO $x_i := x_i + 1;\ y' := y' - 1$ END

ただし，y, y' はプログラムに出てこない変数

関数の呼び出し

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 $\mathsf{add}(x_i, x_j)$ で書くとき

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

のように書くことを可能にしたい

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

$x_3 := x_4;$
$y := x_5;$
WHILE $y \neq 0$ DO
　$x_3 := x_3 + 1;$
　$y := y - 1$
END

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，「関数の再帰呼出」はできない

条件分岐 (1)

条件分岐の機能

WHILE プログラム P と変数 x に対して，

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

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

```
IF x ≠ 0 THEN
   P
END
```

$\rightsquigarrow$

```
y := x;
WHILE y ≠ 0 DO
   P;
   y := 0
END
```

ただし，y はプログラムに出てこない変数

条件分岐 (2)

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，

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

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

```
IF x ≠ 0 THEN
   P1
ELSE
   P2
END
```

$\rightsquigarrow$

```
y := 1;
IF x ≠ 0 THEN
   P1; y := 0
END;
IF y ≠ 0 THEN
   P2
END
```

ただし，y はプログラムに出てこない変数

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 計算可能性：例題 2

例題 2

次の部分関数 $\mathtt{mult}\colon \mathbb{N}^2 \nrightarrow \mathbb{N}$ は WHILE 計算可能か？

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

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 ≠ 0 DO
    x_0 := 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 □

WHILE 計算可能性：例題 2 (WHILE プログラム)

次の WHILE プログラムが mult を計算する

```
WHILE x_1 ≠ 0 DO
   WHILE x_1 ≠ 0 DO
      x_3 := x_3 + 1; x_4 := x_4 + 1; x_1 := x_1 − 1
   END;
   WHILE x_3 ≠ 0 DO x_5 := x_5 + 1; x_3 := x_3 − 1 END;
   WHILE x_4 ≠ 0 DO x_2 := x_2 + 1; x_4 := x_4 − 1 END;
   WHILE x_5 ≠ 0 DO x_0 := x_0 + 1; x_5 := x_5 − 1 END;
   x_1 := x_1 − 1
END
```

WHILE 計算可能性：例題 3

例題 3

次の部分関数 $\mathtt{isEven}\colon \mathbb{N} \nrightarrow \mathbb{N}$ は WHILE 計算可能か？

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

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． □

```
x0 := 1;
WHILE x1 ≠ 0 DO
   x1 := x1 − 1;
   IF x1 ≠ 0 THEN
      x1 := x1 − 1;
      IF x1 ≠ 0 THEN
         x2 := x2 + 1
      ELSE
         x0 := 1
      END;
   ELSE
      x0 := 0
   END
END
```

WHILE 計算可能性：例題 4

例題 3

次の部分関数 $\mathtt{half}\colon \mathbb{N} \nrightarrow \mathbb{N}$ は WHILE 計算可能か？
任意の $x_1 \in \mathbb{N}$ に対して，

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

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． □

```
x2 := isEven(x1);
IF x2 ≠ 0 THEN
   WHILE x1 ≠ 0 DO
      x0 := x0 + 1;
      x1 := x1 − 1
   END
ELSE
   x0 := infloop(x1)
END
```

WHILE 計算可能性：注意

注意

部分関数 $f\colon \mathbb{N}^k \rightharpoonup \mathbb{N}$ が WHILE 計算可能であるとき，
f を計算する WHILE プログラムは無限に存在する

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

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END
```

```
WHILE x1 ≠ 0 DO
    x0 := x0 + 1;
    x1 := x1 − 1
END;
WHILE x2 ≠ 0 DO
    x0 := x0 + 1;
    x2 := x2 − 1
END;
x1 := x1 + 1
```

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

今日のまとめ と 次回の予告

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) を通し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

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