

情報数学

第3回 帰納的関数

萩野 達也

hagino@sfc.keio.ac.jp

スライドURL

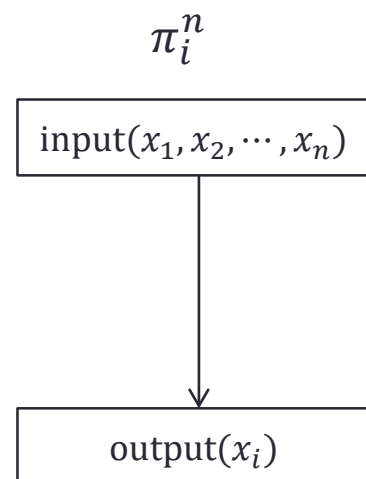
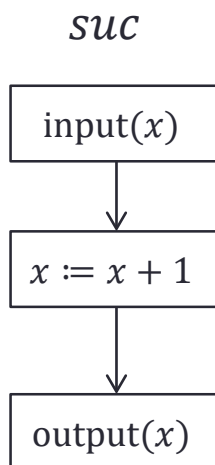
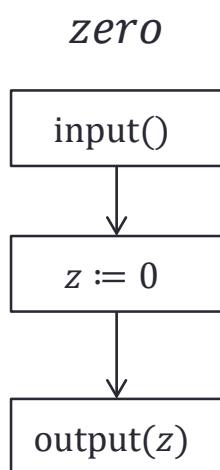
<https://vu5.sfc.keio.ac.jp/slide/>

前回まで

- 計算可能性
 - whileプログラムとフローチャートは同じ計算能力を持つ.
- 原始帰納的関数
 - $zero : N^0 \rightarrow N$ $zero() = 0$
 - $suc : N \rightarrow N$ $suc(x) = x + 1$
 - $\pi_i^n : N^n \rightarrow N$ $\pi_i^n(x_1, \dots, x_n) = x_i$
 - 原始帰納法
 - $f(x_1, \dots, x_n, zero()) = g(x_1, \dots, x_n)$
 - $f(x_1, \dots, x_n, suc(y)) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$
 - 関数合成
 - $f(x_1, \dots, x_n) = g(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n))$
- 原始帰納的関数の例:
 - one, pred, add, sub, mul, div, ...

原始帰納的関数の計算

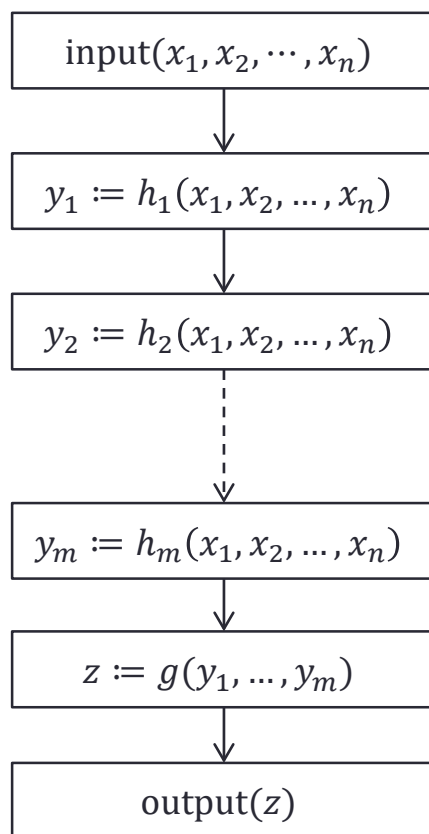
- 定理:
 - 原始帰納的関数は計算可能である.
- 証明:
 - $zero, suc, \pi_i^n$ は計算可能である.



原始帰納的関数の計算

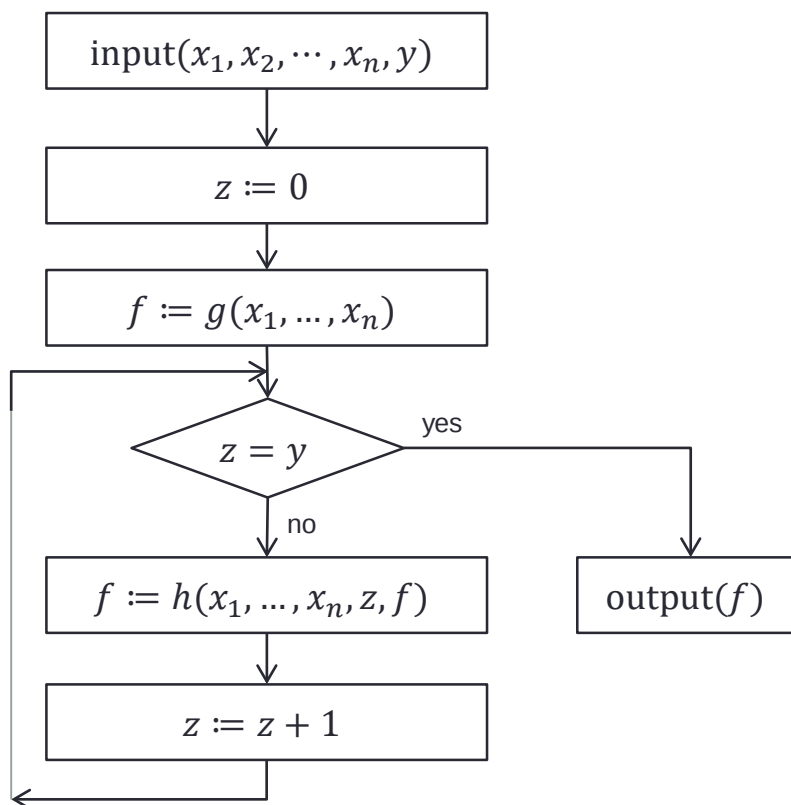
- 原始帰納的関数の合成は計算可能である.

$$f(x_1, x_2, \dots, x_n) = g(h_1(x_1, x_2, \dots, x_n), \dots, h_m(x_1, x_2, \dots, x_n))$$



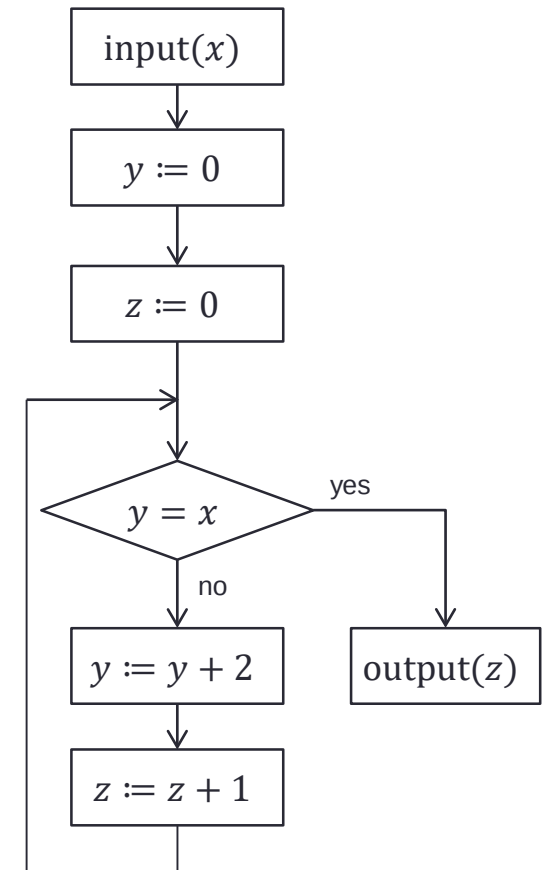
原始帰納的関数の計算

- 原始帰納法は計算可能である.
- $f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$
- $f(x_1, \dots, x_n, \text{suc}(y)) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$



計算可能な関数は原始帰納的関数か？

- 原始帰納的関数は全域的関数である.
 - **全域的** = 入力に対して, 出力が必ずある.
- コンピュータが計算する関数は全域的関数とは限らない.
 - 部分的関数であることもある.
 - **部分的** = 入力に対して出力がないことがある.
- **計算可能な関数は原始帰納的関数より広い.**
- 全域的でコンピュータで計算可能だが, 原始帰納的でない関数もある.
 - **アッカーマン関数** $A: N^2 \rightarrow N$
 - $A(0, y) = \text{suc}(y)$
 - $A(\text{suc}(x), 0) = A(x, \text{suc}(0))$
 - $A(\text{suc}(x), \text{suc}(y)) = A(x, A(\text{suc}(x), y))$



最小解関数

- 定義:

- 述語 $P: N^{n+1} \rightarrow \{T, F\}$ に対して

$$f(x_1, \dots, x_n) = \min(\{y \mid P(x_1, \dots, x_n, y) \text{ が } T\})$$

$f(x_1, \dots, x_n)$ は $P(x_1, \dots, x_n, y)$ が真となる最小の y

- $f(x_1, \dots, x_n)$ は述語 $P(x_1, \dots, x_n, y)$ の最小解関数とよばれ, 次のように書かれる.

$$\mu_y(P(x_1, \dots, x_n, y))$$

- μ は最小解演算子とよばれる.

- 例:

- $f(x) = \mu_y(x = y \times 2)$
 $f(2) =$
 $f(3) =$
- $g(x) = \mu_y(x = y^2)$
 $g(4) =$
 $g(5) =$

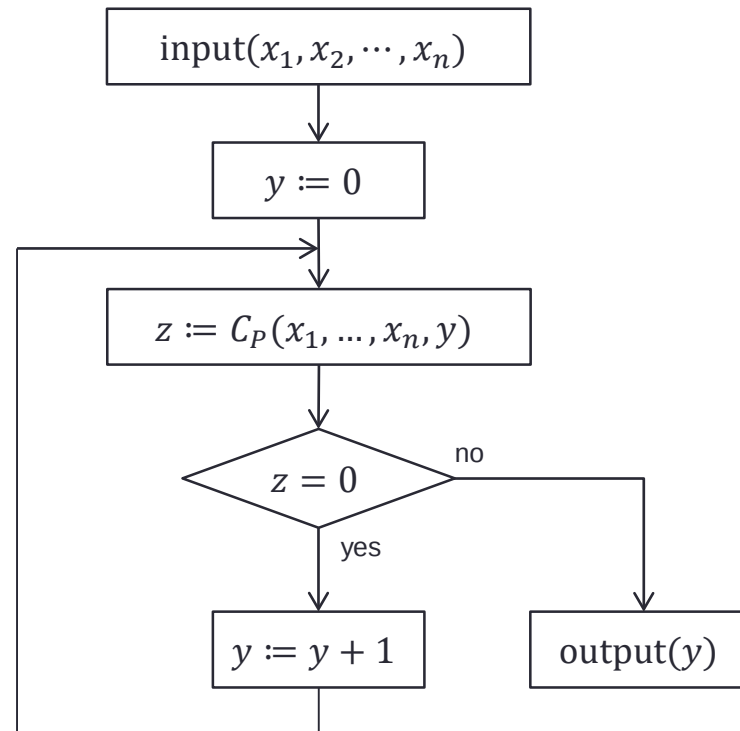
帰納的関数

- 帰納的関数 (recursive function)
 - 原始帰納的関数
 - 原始帰納的述語に対する最小解関数
 - 帰納的関数の合成
- 帰納的関数は
 - 原始帰納的関数 + 最小解演算子

帰納的 $\Rightarrow$ 計算可能

- 定理: 帰納的関数は計算可能である.
- 証明:
 - 原始帰納的関数は計算可能なので, 最小解関数の場合だけを示せばよい.

$$f(x_1, \dots, x_n) = \mu_y(P(x_1, \dots, x_n, y))$$



Gödel関数

- **Gödel関数** $G: N^n \rightarrow N$ とその逆関数 $G_1: N \rightarrow N, \dots, G_n: N \rightarrow N$ は次の性質を持たなくてはならない.
 - G は1対1関数(単射)
 - $G_i(G(x_1, \dots, x_n)) = x_i$
 - $G, G_1, \dots, G_n$ は原始帰納的関数
- $G(x_1, \dots, x_n)$ を $x_1, \dots, x_n$ の**Gödel数**とよぶ.
- 例:
 - $G(x_1, x_2, \dots, x_n) = 2^{x_1} \times 3^{x_2} \times \dots \times p_n^{x_n}$ (ここで p_n は n 番目の素数)
 - $G_1(x) = x - \mu_{y < x}(\text{divisible}(x, 2^{x-y}))$
 - $G_2(x) = x - \mu_{y < x}(\text{divisible}(x, 3^{x-y}))$
 - $\vdots$
 - $G_n(x) = x - \mu_{y < x}(\text{divisible}(x, p_n^{x-y}))$

計算可能 $\Rightarrow$ 帰納的

- 定理：計算可能な関数は帰納的関数である.
- 証明：
 - 任意のwhileプログラムは次の形に書き換えることができる.

```
「 input( $x_1, \dots, x_i$ );  
「  $a := 1$ ;  
「 while ( $a - k = 0$ ) {  
  「 if ( $a = 1$ )  $P_1$ ;  
  「 else if ( $a = 2$ )  $P_2$ ;  
  「 else if ( $a = 3$ )  $P_3$ ;  
  「  $\vdots$   
  「 else if ( $a = k$ )  $P_k$ ;  
  }  
  output( $y$ )
```

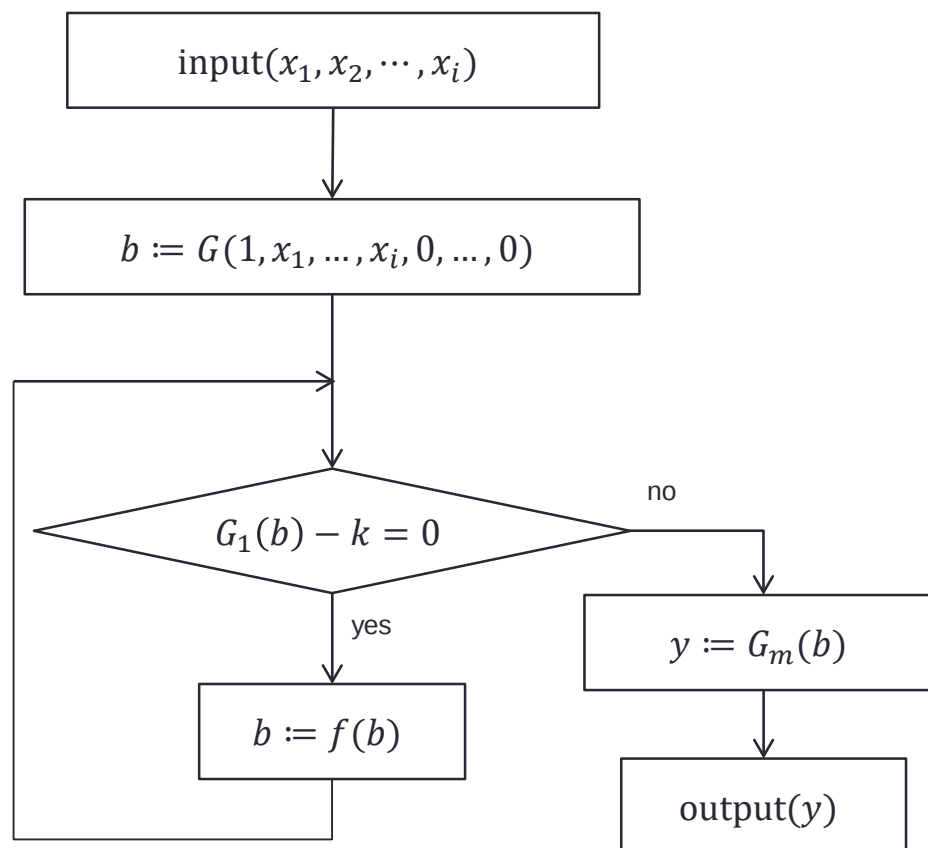
ここで P_i は代入文か条件文

証明 (cont.)

- プログラムに出てくるすべての変数を $a_2, \dots, a_n$ とする.
 - a_1 を箱の番号を表す変数とする.
 - 個々の変数の代わりに $b = G(a_1, \dots, a_n)$ を使う.
- P_i が **代入文** 「 $a_m := f(a_2, \dots, a_n); a_1 := l$ 」の場合
 - $b := G(l, G_2(b), \dots, f(G_2(b), \dots, G_n(b)), \dots, G_n(b))$
- P_i が **条件文** 「 if ($P(a_2, \dots, a_n)$) $a_1 := l$ else $a_1 := m$ 」の場合
 - $b := G(C_P(G_2(b), \dots, G_n(b)) \times l + (1 - C_P(G_2(b), \dots, G_n(b))) \times m, G_2(b), \dots, G_n(b))$
- どちらの場合も, P_i は一つの代入文となる.
 - $b := f_i(b)$
 - ここで f_i は原始帰納的関数.
- a の値に従って P_i を選ぶ部分も一つの代入文として書くことができる.
 - $b := \sum_{i=1}^k C_=(G_1(b), i) \times f_i(b)$

証明 (cont.)

- プログラムは下記のように変換することができる.



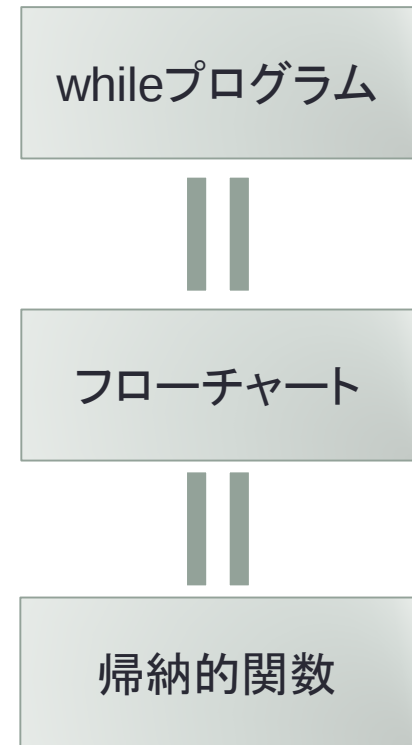
証明 (cont.)

- $f^\#(b, n) = f\left(f\left(f(\cdots f(b))\right)\right)$ とする.
 - f を b に n 回適用したもの.
 - 原始帰納法で定義することができる:
 - $f^\#(b, 0) = b$
 - $f^\#(b, \text{succ}(n)) = f\left(f^\#(b, n)\right)$
- プログラムの繰り返しは $G_1(b)$ が k 以下の間続くので, 繰り返し回数は, 最小解オペレータを使って表すことができる. $b = G(1, x_1, \dots, x_n, 0, \dots, 0)$ としたとき,
 - $\mu_n\left(G_1\left(f^\#(b, n)\right) > k\right)$
- したがって, プログラムの計算する関数は,
 - $G_m\left(f^\#\left(G(1, x_1, \dots, x_n, 0, \dots, 0), \mu_n\left(G_1\left(f^\#(G(1, x_1, \dots, x_n, 0, \dots, 0), n)\right) > k\right)\right)\right)$
- これは帰納的関数. (QED)
- 系: 任意の帰納的関数は原始帰納的関数 f と原始帰納的述語 P を使って次のように表すことができる.

$$f\left(x_1, \dots, x_n, \mu_y\left(P(x_1, \dots, x_n, y)\right)\right)$$

まとめ

- 原始帰納的関数：
 - 和と積
 - 原始帰納的述語
 - 四則演算は原始帰納的
- 帰納的関数：
 - 原始帰納的関数
 - 最小解演算子
- 帰納的関数は計算可能である.
- 計算可能な関数は帰納的関数である.



数学的帰納法

- 自然数 x に関する述語 $P(x)$ を証明するとき, 以下の2つを示す.
 - (基底) $x = 0$ のときに成り立つことを示す
 - (帰納) $x = n$ のときに成り立つと仮定して, $x = n + 1$ のときにも成り立つことを示す.
 - 数学的帰納法**(mathematical induction)により, すべての自然数 x に対して $P(x)$ は成り立つ.

$$\frac{P(0) \quad P(n) \supset P(n+1)}{\forall x P(x)}$$

- 定理: $add(0, x) = x$
- 証明:
 - (基底) $x = 0$ のとき, 定義から $add(0, 0) = 0$ なので成り立つ
 - (帰納) $x = n$ のときに成り立つと仮定すると, $add(0, n) = n$.
 $x = n + 1$ のとき,
 $add(0, n + 1)$
 $= add(0, suc(n))$
 $= suc(add(0, n))$ (add の定義から)
 $= suc(n)$ (帰納法の過程から)
 $= n + 1$

数学的帰納法により, すべての自然数 x に対して, $add(0, x) = x$ である.

宿題

- 数学的帰納法を使って, add が可換であることを証明しなさい.

$$add(x, y) = add(y, x)$$

- 提出方法
 - SOLの課題提出から, PDF形式で提出しなさい。
 - 締め切り: 10月31日土曜日