情報数学
第3回　帰納的関数

萩野　達也
hagino@sfc.keio.ac.jp

Slides 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.

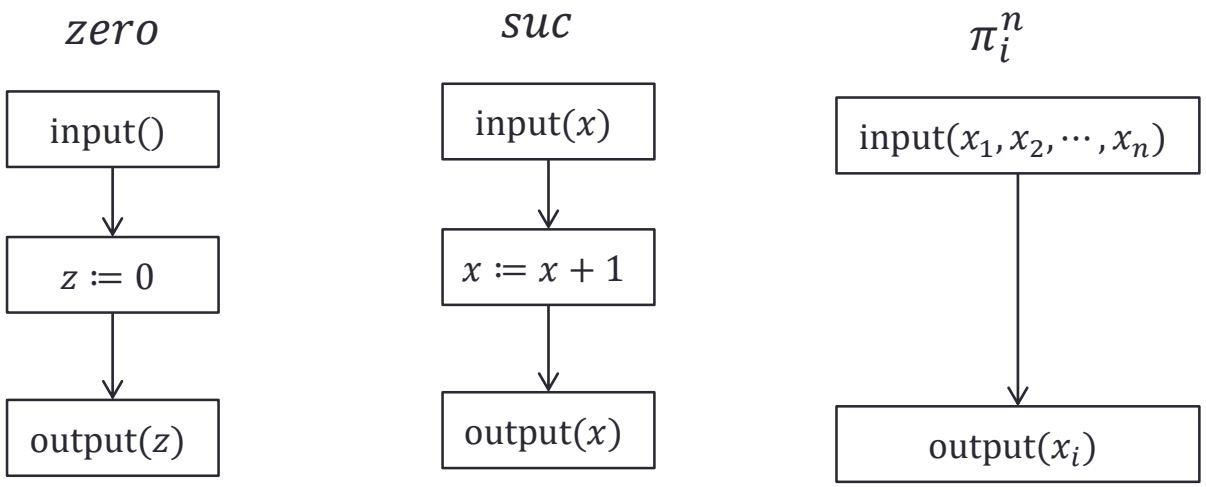
- 原始帰納的関数
  - $zero : N^0 \to N$ $\quad zero() = 0$
  - $suc : N \to N$ $\quad suc(x) = x + 1$
  - $\pi_i^n : N^n \to N$ $\quad \pi_i^n(x_1, \ldots, x_n) = x_i$
  - 原始帰納法
    - $f(x_1, \ldots, x_n, zero()) = g(x_1, \ldots, x_n)$
    - $f\big(x_1, \ldots, x_n, suc(y)\big) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$
  - 関数合成
    - $f(x_1, , \ldots, x_n) = g(h_1(x_1, , \ldots, x_n), \ldots, h_m(x_1, , \ldots, x_n))$

- 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.

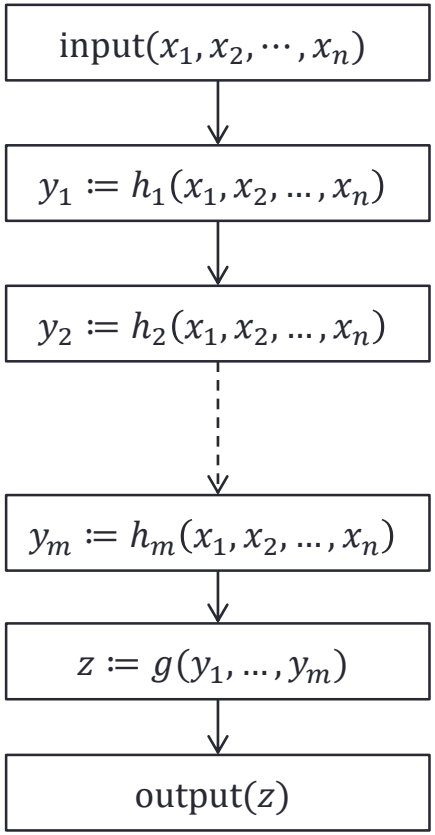
- **証明:**
  - $zero, suc, \pi_i^n$ は計算可能である.



原始帰納的関数の計算

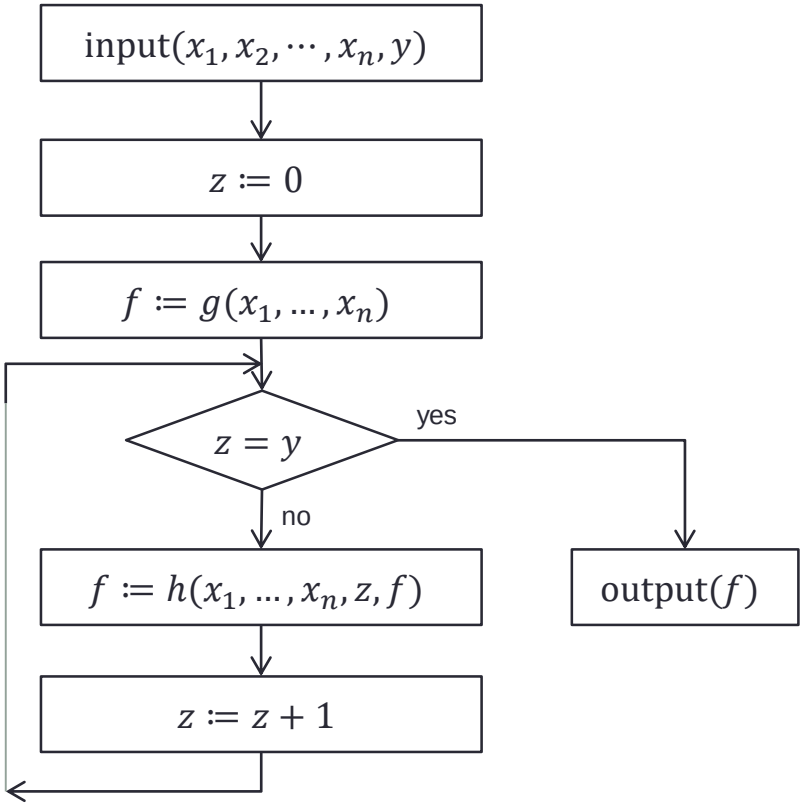
- 原始帰納的関数の合成は計算可能である.

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



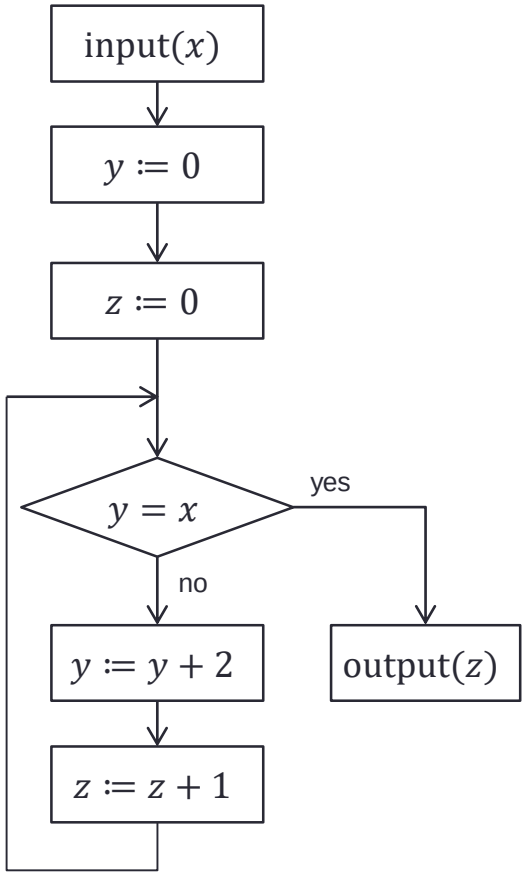
原始帰納的関数の計算

- 原始帰納法は計算可能である.
  - $f(x_1, \ldots, x_n, zero()) = g(x_1, \ldots, x_n)$
  - $f\big(x_1, \ldots, x_n, suc(y)\big) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$



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 $A: N^2 \to N$
    - $A(0, y) = suc(y)$
    - $A(suc(x), 0) = A(x, suc(0))$
    - $A(suc(x), suc(y)) = A\big(x, A(suc(x), y)\big)$



最小解関数

- **定義:**
  - 述語 $P: N^{n+1} \to \{\mathrm{T}, \mathrm{F}\}$ に対して

$$f(x_1, \ldots, x_n) = \min(\{y \mid p(x_1, \ldots, x_n, y) \text{ is True}\})$$

- $f(x_1, \ldots, x_n)$ は $P(x_1, \ldots, x_n, y)$ が真となる最小の y
  - $f(x_1, \ldots, x_n)$ は述語 $P(x_1, \ldots, x_n, y)$ の最小解関数とよばれ，次のように書かれる.

$$\mu_y\big(p(x_1, \ldots, x_n, y)\big)$$

  - 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

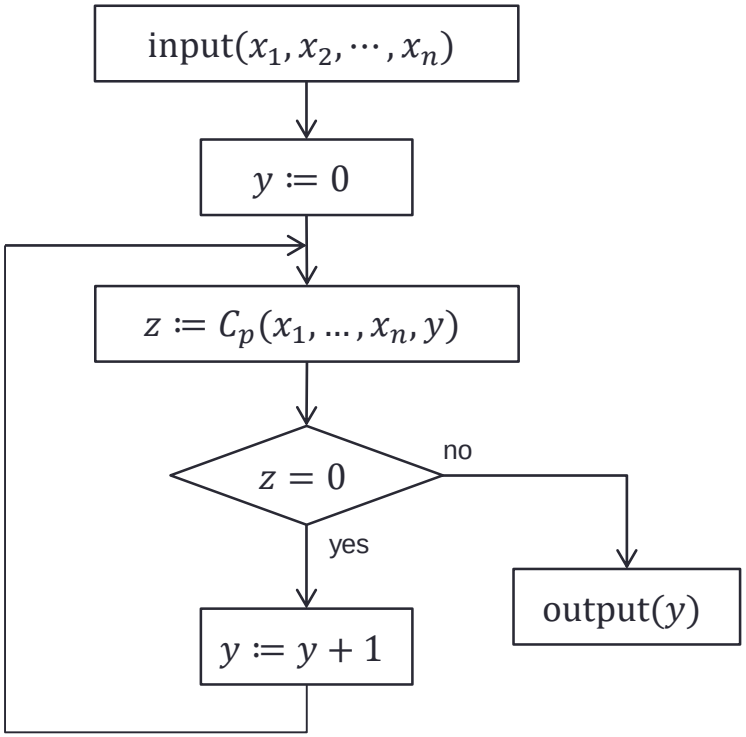
- 帰納的関数は
  - 原始帰納的関数 + 最小解演算子

帰納的 ⇒ 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．

$$f(x_1, \ldots, x_n) = \mu_y\big(p(x_1, \ldots, x_n, y)\big)$$



Gödel関数

- Gödel関数 $G: N^n \to N$ とその逆関数 $G_1: N \to N, \dots, G_n: N \to N$ は次の性質を持たなくてはいけない.
  - G は1対1関数（単射）
  - $G_i\big(G(x_1, \dots, x_n)\big) = x_i$
  - $G, G_1, \dots, G_n$ 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:
  - $G(x_1, x_2, \dots, x_n) = 2^{x_1} \times 3^{x_2} \times \cdots \times p_n^{x_n}$ (ここで p_n は n番目の素数)
  - $G_1(x) = x - \mu_{y<x}\big(\mathrm{divisible}(x, 2^{x-y})\big)$
  - $G_2(x) = x - \mu_{y<x}\big(\mathrm{divisible}(x, 3^{x-y})\big)$
  - $\vdots$
  - $G_n(x) = x - \mu_{y<x}\left(\mathrm{divisible}\big(x, p_n^{x-y}\big)\right)$

計算可能 ⇒ 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($x_1, \dots, x_n$);
$a := 1$;
while ($a - k = 0$) {
    if ($a = 1$) P_1;
    else if ($a = 2$) P_2;
    else if ($a = 3$) P_3;
      ⋮
    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 $a_1, \dots, a_n$ 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「 $a_m := f(a_1, \dots, a_n); a_1 := l$ 」の場合
  - $b := G\left(l, G_2(b), \dots, f\left(G_1(b), \dots, G_n(b)\right), \dots, G_n(b)\right)$

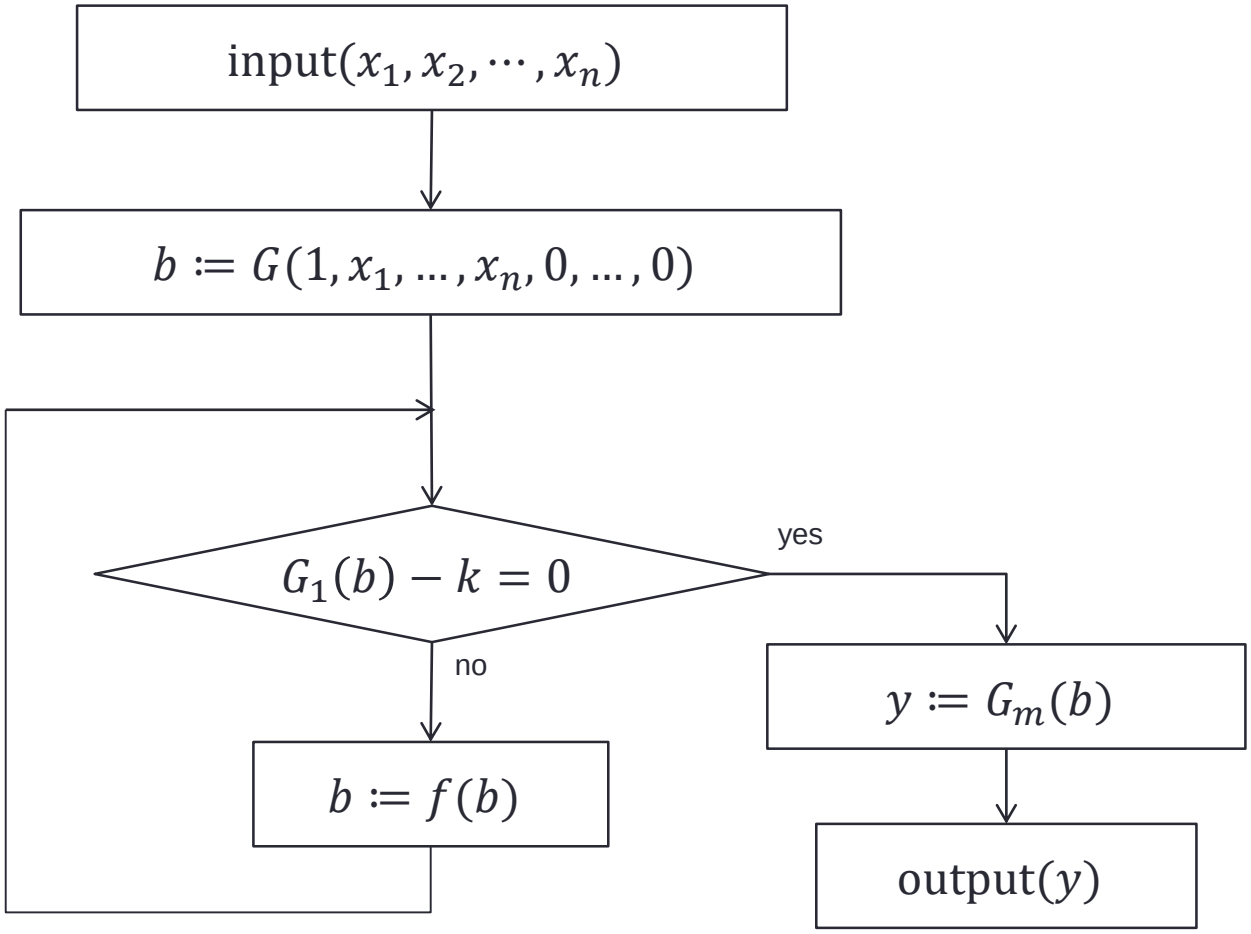
- P_i が条件文「 if $(P(a_1, \dots, a_n))$ $a_1 := l$ else $a_1 := m$ 」の場合
  - $b := G\left(C_P\left(G_1(b), \dots, G_n(b)\right) \times l + \left(1 - C_P\left(G_1(b), \dots, G_n(b)\right)\right) \times m, G_2(b), \dots, G_n(b)\right)$

- 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.)

- $f^{\#}(b,n) = f\left(f\left(f\left(\cdots f(b)\right)\right)\right)$ 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:
    - $f^{\#}(b,0) = b$
    - $f^{\#}(b,suc(n)) = f\left(f^{\#}(b,n)\right)$
- 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.

$$h(b) = f^{\#}\left(b, \mu_n\left(G_1\left(f^{\#}(b,n)\right) > k\right)\right)$$

- したがって，プログラムの計算する関数は，
  - $G_m\left(h\left(G(1,x_1,\ldots,x_n,0,\ldots,0)\right)\right)$
- 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.

$$f\left(x_1, \ldots, x_n, \mu_y\left(p(x_1, \ldots, x_n, y)\right)\right)$$

- μ は1つだけでかまわない
- 他はすべて原始帰納的関数

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，$x = suc(n)$ 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．

$$\frac{P(0) \qquad P(n) \supset P\big(suc(n)\big)}{\forall x \in N \;\; P(x)}$$

- たとえば，次のことを証明することができる．

定理：$add(x, y) = add(y, x)$

補題 $add(0,x) = x$

補題: $add(0,x) = x$

Definition of add
- $add(x,0) = x$
- $add\big(x, suc(y)\big) = suc\big(add(x,y)\big)$

- 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，$add(0,n) = n$.

$x = suc(n)$ のとき，

$$
\begin{aligned}
\text{左辺} &= add\big(0, suc(n)\big) \\
&= suc\big(add(0,n)\big) && (\because add \text{ の定義}) \\
&= suc(n) && (\because \text{帰納法の仮定}) \\
&= \text{右辺}
\end{aligned}
$$

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.

補題 $add(suc(x), y) = suc\big(add(x, y)\big)$

補題：$add(suc(x), y) = suc\big(add(x, y)\big)$

- 証明：

（基底）$y = 0$ のとき，

左辺 $= add(suc(x), 0)$

$= suc(x)$ ($\because$ add の定義)

$= suc\big(add(x, 0)\big)$ ($\because$ add の定義)

$=$ 右辺

Definition of add
- $add(x, 0) = x$
- $add\big(x, suc(y)\big) = suc\big(add(x, y)\big)$

（帰納）$y = n$ のときに成り立つと仮定すると，$add(suc(x), n) = suc\big(add(x, n)\big)$.

$y = suc(n)$ のとき，

左辺 $= add\big(suc(x), suc(n)\big)$

$= suc\big(add(suc(x), n)\big)$ ($\because$ add の定義)

$= suc\Big(suc\big(add(x, n)\big)\Big)$ ($\because$ 帰納法の仮定)

$= suc\Big(add\big(x, suc(n)\big)\Big)$ ($\because$ add の定義)

$=$ 右辺

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

定理 $add(x,y) = add(y,x)$

定理：$add(x,y) = add(y,x)$

- 証明：

（基底）$y = 0$ のとき，

左辺 $= add(x,0)$

$= x$ （∵ add の定義）

$= add(0,x)$ （∵ 2つ前の補題）

$=$ 右辺

Definition of add
- $add(x,0) = x$
- $add\big(x,suc(y)\big) = suc\big(add(x,y)\big)$

（帰納）$y = n$ のときに成り立つと仮定すると，$add(x,n) = add(n,x)$.

$y = suc(n)$ のとき，

左辺 $= add\big(x,suc(n)\big)$

$= suc\big(add(x,n)\big)$ （∵ add の定義）

$= suc\big(add(n,x)\big)$ （∵ 帰納法の仮定）

$= add(suc(n),x)$ （∵ 1つ前の補題）

$=$ 右辺

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

まとめ

- 原始帰納的関数:
  - 和と差と積
  - 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.