

情報数学

第2回 原始帰納的関数

萩野 達也

hagino@sfc.keio.ac.jp

スライドURL

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

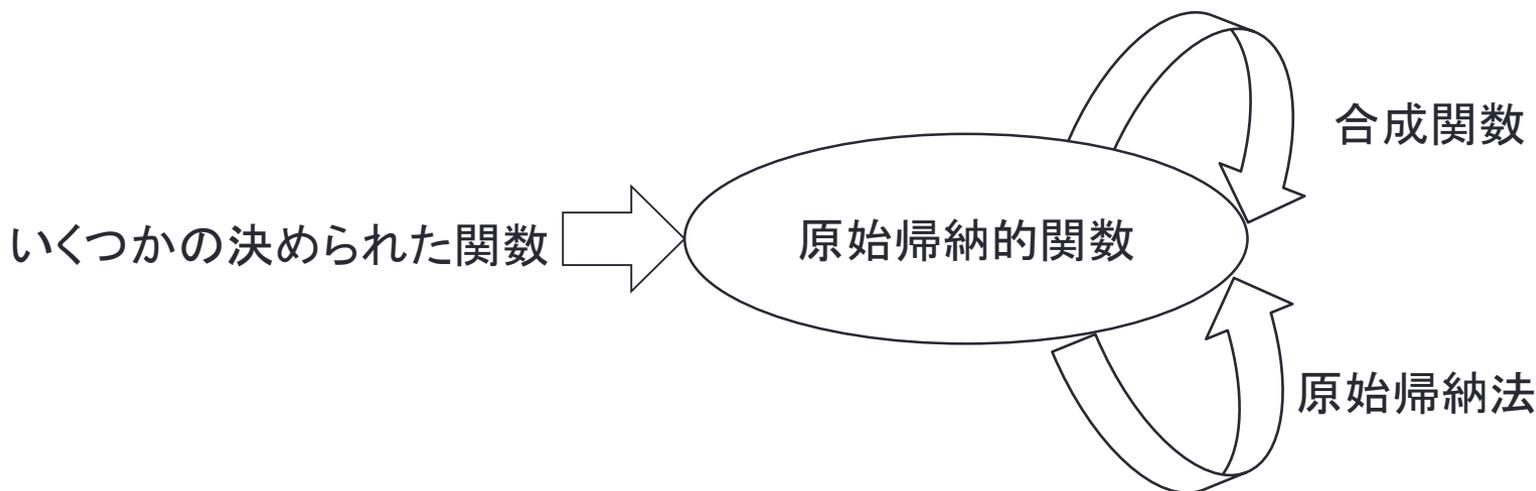
前回

- 計算とは何か？
 - 計算 = コンピュータが計算しているもの
 - 計算可能関数 = コンピュータで計算している数学的な関数
 - 計算可能性 = どのような関数をコンピュータは計算できるのか
- whileプログラムとフローチャートは同じ計算能力がある
- どんなプログラムもwhileが一つだけのプログラムに書き換えることができる

原始帰納的関数

定義:

- **原始帰納的関数** (primitive recursive function) は以下のものからなっている.
 1. いくつかの決められた関数
 2. 原始帰納的関数の**合成関数**
 3. **原始帰納法**によって定義された関数



いくつかの決められた原始帰納的関数

以下の3種類の関数は原始帰納的関数

1. $zero : N^0 \rightarrow N$

- $zero() = 0$
- 常に 0 を返す関数

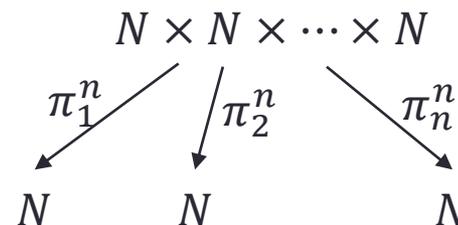
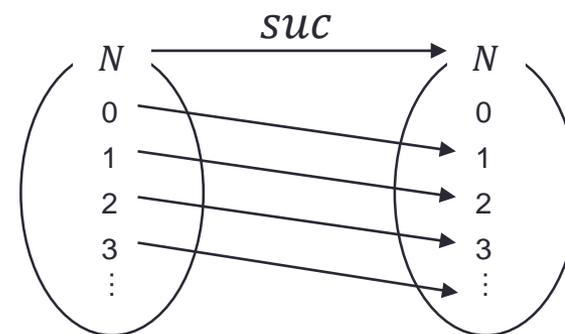
2. $suc : N \rightarrow N$

- $suc(x) = x + 1$
- x の次の数を返す関数
- $suc(2) = 3$
- $suc(100) = 101$

3. $\pi_i^n : N^n \rightarrow N$

- $\pi_i^n(x_1, \dots, x_n) = x_i$
- n 個の引数の i 番目の要素を返す関数
- 射影(projection)
- $\pi_1^2(x, y) = x$
- $\pi_2^2(x, y) = y$

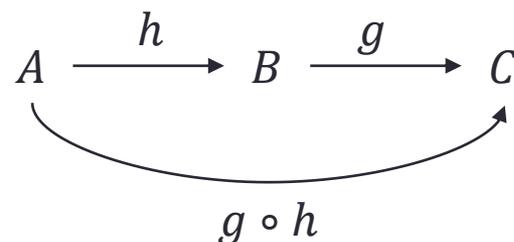
- $N^0 = \{ \cdot \}$
- $N^1 = N$
- $N^2 = N \times N = \{ (x, y) | x \in N, y \in N \}$
- $N^3 = N \times N \times N$
- $N^n = N \times N \times N \times \dots \times N$



原始帰納関数の合成

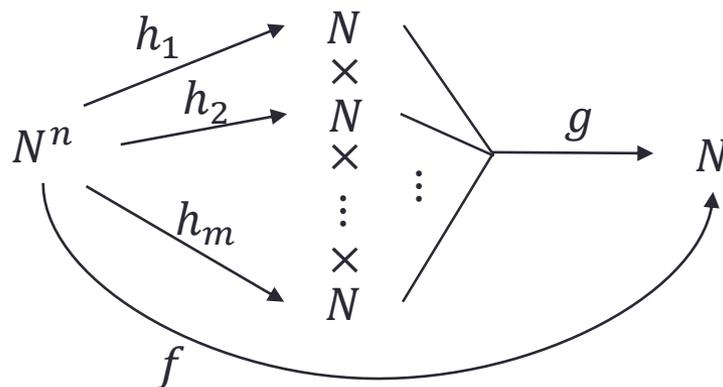
関数の合成

- 2つの関数を結合(合成)する
- $g \circ h(x) = g(h(x))$



原始帰納的関数の合成は原始帰納的関数

- $g : N^m \rightarrow N$ と $h_i : N^n \rightarrow N$ ($i = 1, 2, \dots, m$) が原始帰納的関数のとき, 合成した $f : N^n \rightarrow N$ も原始帰納的関数である.
- $f(x_1, x_2, \dots, x_n) = g(h_1(x_1, x_2, \dots, x_n), \dots, h_m(x_1, x_2, \dots, x_n))$



例1: 原始帰納的関数

- $id(x) = x$ $id : N \rightarrow N$
 - 恒等関数は原始帰納的関数である.
 - $id(x) = \pi_1^1(x)$
- $one() = 1$ $one : N^0 \rightarrow N$
 - 常に 1 を返す関数は原始帰納的関数である.
 - $one() = suc(zero())$
- $two() = 2$ $two : N^0 \rightarrow N$
 - 定数を返す関数は原始帰納的関数である.
 - $two() = suc(suc(zero()))$
- $add2(x) = x + 2$ $add2: N \rightarrow N$
 - 与えられた数に2を加える関数は原始帰納的関数である.
 - $add2(x) = suc(suc(x))$

原始帰納法によって定義された関数

- $g : N^n \rightarrow N$ と $h : N^{n+2} \rightarrow N$ が原始帰納的関数のとき, 次のように定義した $f : N^{n+1} \rightarrow N$ も原始帰納的関数
 - $f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$
 - $f(x_1, \dots, x_n, \text{suc}(y)) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$
- **原始帰納法**による関数の定義と呼ばれる
- $n = 0$ の場合, $g : N^0 \rightarrow N$ と $h : N \times N \rightarrow N$ が原始帰納的関数であるとき, $f : N \rightarrow N$ は原始帰納的関数
 - $f(0) = g()$
 - $f(\text{suc}(y)) = h(y, f(y))$
- $f(x)$ を x が 0 である場合と, そうでない場合に分けて定義
 - x が 0 のときは $g()$ で定数
 - x が 0 でない場合は $x = \text{suc}(y)$ と書くことができるので, y と $f(y)$ から $f(x)$ の値を $h(y, f(y))$ で計算
 - $f(y)$ については, y が 0 であれば $g()$ であるが, そうでない場合にはもう一度 $f(\text{suc}(y))$ の場合を適用し, 0 になるまでこれを繰り返す
- $f(1) = f(\text{suc}(0)) = h(0, f(0)) = h(0, g())$
- $f(2) = f(\text{suc}(1)) = h(1, f(1)) = h(1, h(0, g()))$
- $f(3) = f(\text{suc}(2)) = h(2, f(2)) = h(2, h(1, h(0, g())))$

例2: 原始帰納的関数

• $double: N \rightarrow N$ $double(x) = x \times 2$

• 数の2倍する関数を原始帰納法で定義する.

• $double(0) = 0$



原始帰納法による定義($n = 0$)

• $f(0) = g()$

• $f(suc(y)) = h(y, f(y))$

• $double(suc(y)) = suc(suc(double(y)))$

• $g() = zero()$

• $h(y, z) = \pi_1^2(y, suc(suc(z)))$

• $double(1) = double(suc(0))$

$$= suc(suc(double(0))) = suc(suc(0)) = 2$$

• $double(2) = double(suc(suc(0)))$

$$= suc(suc(double(suc(0)))) = suc(suc(2)) = 4$$

例3: 原始帰納的関数

- $pred: N \rightarrow N$ $pred(x) = x - 1$
 - 一つ前の数を返す関数(suc の逆)を原始帰納法で定義する.

- $pred(0) =$



原始帰納法による定義 ($n = 0$)

- $f(0) = g()$
- $f(suc(y)) = h(y, f(y))$

- $pred(suc(y)) =$

- $pred(1) =$

- $pred(5) =$

例4: 原始帰納的関数

- $add: N^2 \rightarrow N$

$$add(x, y) = x + y \quad \text{加算}$$

- $add(x, 0) = x$

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

原始帰納法による定義 ($n = 1$)

- $f(x, 0) = g(x)$

- $f(x, suc(y)) = h(x, y, f(x, y))$

- $add(1, 1) =$

- $add(3, 2) =$

例5: 原始帰納的関数

- $sub: N^2 \rightarrow N$

$$sub(x, y) = x - y \quad \text{減算}$$

- $sub(x, 0) = x$

- $sub(x, suc(y)) =$

原始帰納法による定義 ($n = 1$)

- $f(x, 0) = g(x)$

- $f(x, suc(y)) = h(x, y, f(x, y))$

- $mul: N^2 \rightarrow N$

$$mul(x, y) = x \times y \quad \text{乗算}$$

- $mul(x, 0) =$

- $mul(x, suc(y)) =$

原始帰納的関数

• Definition:

- 次の関数を**原始帰納的関数** (primitive recursive function)
- いくつかの決められた関数は原始帰納的関数である。

- $zero : N^0 \rightarrow N$ $zero() = 0$
- $suc : N \rightarrow N$ $suc(x) = x + 1$ (x の次の数)
- $\pi_i^n : N^n \rightarrow N$ $\pi_i^n(x_1, \dots, x_n) = x_i$

簡単のために、今後 $zero()$ を 0 と書く

- 原始帰納的関数の合成は原始帰納的関数である。

- $g : N^m \rightarrow N$ と $h_i : N^n \rightarrow N$ ($i = 1, 2, \dots, m$) が原始帰納的関数のとき、合成した $f : N^n \rightarrow N$ も原始帰納的関数である。

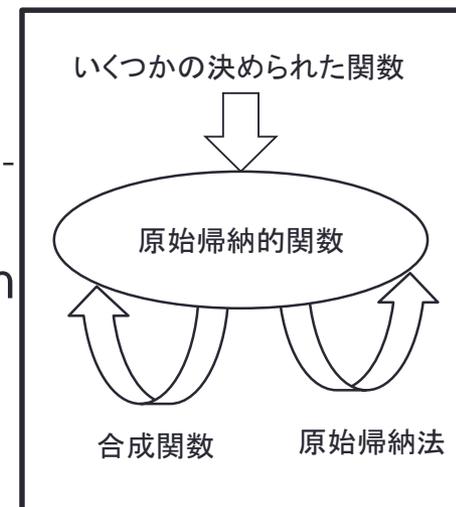
$$f(x_1, x_2, \dots, x_n) = g(h_1(x_1, x_2, \dots, x_n), \dots, h_m(x_1, x_2, \dots, x_n))$$

- **原始帰納法**による関数の定義

- $g : N^n \rightarrow N$ と $h : N^{n+2} \rightarrow N$ が原始帰納的関数のとき、次のように定義した $f : N^{n+1} \rightarrow N$ も原始帰納的関数である。

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$$

$$f(x_1, \dots, x_n, suc(y)) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$



和と積

補題:

- $f(x_1, \dots, x_n, y)$ が原始帰納的関数であれば, 次の2つは原始帰納的関数である.

$$g(x_1, \dots, x_n, y) = \sum_{z=0}^{y-1} f(x_1, \dots, x_n, z)$$

$$h(x_1, \dots, x_n, y) = \prod_{z=0}^{y-1} f(x_1, \dots, x_n, z)$$

- $n = 0$ の場合

- $g(y) = f(0) + f(1) + \dots + f(y-1)$
- $h(y) = f(0) \times f(1) \times \dots \times f(y-1)$

証明:

- $g(x_1, \dots, x_n, 0) = 0$
- $g(x_1, \dots, x_n, \text{suc}(z)) = \text{add}(f(x_1, \dots, x_n, z), g(x_1, \dots, x_n, z))$
- $h(x_1, \dots, x_n, 0) = 1$
- $h(x_1, \dots, x_n, \text{suc}(z)) = \text{mul}(f(x_1, \dots, x_n, z), g(x_1, \dots, x_n, z))$

原始帰納的述語

定義:

- 述語 $P(x_1, \dots, x_n): N^n \rightarrow \{T, F\}$ は, その特性関数 $C_P(x_1, \dots, x_n): N^n \rightarrow N$ 原始帰納的関数であるとき, 原始帰納的述語とよぶ.
 - $C_P(x_1, \dots, x_n) = 1$ ($P(x_1, \dots, x_n)$ が T のとき)
 - $C_P(x_1, \dots, x_n) = 0$ ($P(x_1, \dots, x_n)$ が F のとき)

例:

- 述語「 $x = 0$ 」は原始帰納的述語である.
 - $C_{=0}(x) = 1 - x$
- 述語「 $x \leq y$ 」は原始帰納的述語である.
 - $C_{\leq}(x, y) = C_{=0}(x - y) = 1 - (x - y)$
- 述語「 $x = y$ 」は原始帰納的述語である.
 - $C_{=}(x, y) = C_{=0}((x - y) + (y - x)) = 1 - ((x - y) + (y - x))$

論理式

補題:

- $P(x_1, \dots, x_n), Q(x_1, \dots, x_n), R(x_1, \dots, x_n, z)$ が原始帰納的述語であれば, 次の述語も原始帰納的述語である.
 - $P(x_1, \dots, x_n) \wedge Q(x_1, \dots, x_n)$
 - $P(x_1, \dots, x_n) \vee Q(x_1, \dots, x_n)$
 - $\neg P(x_1, \dots, x_n)$
 - $\forall z < y (R(x_1, \dots, x_n, z)) \equiv \forall z (z < y \Rightarrow R(x_1, \dots, x_n, z))$
 - $\exists z < y (R(x_1, \dots, x_n, z)) \equiv \exists z (z < y \wedge R(x_1, \dots, x_n, z))$

証明:

- $C_{P \wedge Q}(x_1, \dots, x_n) = C_P(x_1, \dots, x_n) \times C_Q(x_1, \dots, x_n)$
- $C_{\neg P}(x_1, \dots, x_n) = 1 - C_P(x_1, \dots, x_n)$
- $C_{P \vee Q}(x_1, \dots, x_n) = C_{\neg(\neg P \wedge \neg Q)}(x_1, \dots, x_n) = 1 - (1 - C_P(x_1, \dots, x_n)) \times (1 - C_Q(x_1, \dots, x_n))$
- $C_{\forall R}(x_1, \dots, x_n, y) = \prod_{z=0}^{y-1} C_R(x_1, \dots, x_n, z)$
- $C_{\exists R}(x_1, \dots, x_n, y) = C_{\neg \forall \neg R}(x_1, \dots, x_n, y) = 1 - \prod_{z=0}^{y-1} (1 - C_R(x_1, \dots, x_n, z))$

例

- 「 $x > y$ 」は原始帰納的述語である.
 - $(x > y) \equiv$
- $divisible(x, y) \equiv$ 「 x は y で割り切れる」は原始帰納的述語である.
 - $divisible(x, y) \equiv$
- $prime(x) \equiv$ 「 x は素数である」は原始帰納的述語である.
 - $prime(x) \equiv$

最小値

補題:

- $P(x_1, \dots, x_n, z)$ が原始帰納的述語であるとき, 次の関数は原始帰納的関数である.

$$\mu_{z < y} P(x_1, \dots, x_n, z) = \min(\{z \mid P(x_1, \dots, x_n, z)\} \cup \{y\})$$

y 以下で $P(x_1, \dots, x_n, z)$ が成り立つ最小の z を求める.

証明:

- $g(x_1, \dots, x_n, z)$ を $\forall v < z (\neg P(x_1, \dots, x_n, v))$ の特性関数とすると,

$$\mu_{z < y} P(x_1, \dots, x_n, z) = \sum_{z=0}^{y-1} g(x_1, \dots, x_n, z + 1)$$

$n = 1$ の場合
 $\mu_{z < y} P(x, z)$

z	0	1	2	3	4	5	6	...	$y - 1$	y
$P(x, z)$	F	F	F	T	F	T	T	...	F	T
$\neg P(x, z)$	T	T	T	F	T	F	F	...	T	F
$\forall v < z (\neg P(x, v))$	T	T	T	T	F	F	F		F	F
$g(x, z)$	1	1	1	1	0	0	0	...	0	0

$$\mu_{z < y} P(x, z) = \sum_{z=0}^{y-1} g(x, z + 1) = \sum_{z=1}^y g(x, z) = 3$$

系

• **系:** $div(x, y) = x \div y$ は原始帰納的関数である.

- $div(x, y) = \mu_{z < x} (x < (y \times (z + 1)))$

- $div(5, 2) =$

• **定理:** 四則演算 (加算, 減算, 乗算, 除算) は原始帰納的関数である.

系: $pr(x) =$ 「 x 番目の素数」は原始帰納的関数である.

- $pr(0) = 2$

- $pr(suc(x)) = \mu_{y < pr(x)!+2} (pr(x) < y \wedge prime(y))$

まとめ

- 原始帰納的関数
 - *zero*
 - *suc*
 - π_i^n
 - 関数合成
 - 原始帰納法
- 原始帰納的述語
- 四則演算は原始帰納的関数である.