計算の理論 II
帰納的関数2

月曜4校時
大月美佳

今日の講義内容

1. 復習
   1. 原始帰納的関数の定義
2. 原始帰納的でない関数
   1. Ackermann関数
3. 原始帰納的な集合･述語
4. 部分帰納的関数と帰納的関数

原始帰納的関数
(primitive recursive)

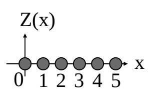
- **定義**

  初期関数(1), (2), (3)に
  操作(I), (II)を
  有限回(0回以上)適用して
  得られた関数。

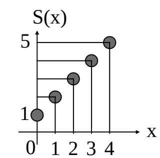
初期関数

- **原始帰納的関数の素。**

(1) $Z(x)=0$
　どんなxも0にしてしまう。



(2) $S(x)=x+1$
　xに1を加える。



(3) $U_n^i(x_1, \ldots, x_i, \ldots, x_n)=x_i$
　i番目のx_iを取り出す。

これらに操作を加えて原始帰納的関数を作成する。

合成と原始帰納

- 初期関数に加える操作

(I) 合成

r変数の関数hとr個のn変数関数g_i(1≦i≦r)から、
n変数の関数fを以下の操作で作ること。

$f(x_1, \ldots, x_n)=h(g_1(x_1, \ldots, x_n), \ldots, g_r(x_1, \ldots, x_n))$

(II) 原始帰納 (primitive recursion)

n-1変数の関数gとn+1変数の関数hから、
n変数の関数fを以下の操作で作ること。

$f(x_1, \ldots, x_n)=g(x_1, \ldots, x_{n-1})$ (x_n=0のとき)

$f(x_1, \ldots, x_n)=h(x_1, \ldots, x_{n-1}, x_n-1, f(x_1, \ldots, x_n-1))$ (x_n＞0のとき)

原始帰納的でない関数

計算可能だが原始帰納的ではない関数

F(x, n)

任意の1変数の原始帰納的関数f(x)に対して
f(x)=F(n, x)
となる自然数が存在するような関数。

Ackerman関数

原始帰納的でない関数

A(x, y)

1. A(0, y) = y+1
2. A(x+1, 0) = A(x, 1)
3. A(x+1, y+1) = A(x, A(x+1, y))

計算に手間のかかる関数→小レポート
類似品：たらいまわし関数

計算量と数の発散

(x, y)	結果	再帰計算回数
(3, 1)	13	106
(3, 2)	29	541
(3, 3)	61	2432
(3, 4)	125	10307
(3, 5)	253	42438
(3, 6)	509	172233
(3, 7)	1021	693964

$$A(0,y)=y+1$$
$$A(1,y)=y+2$$
$$A(2,y)=2y+3$$
$$A(3,y)=2^{y+3}-3$$
$$A(4,y)=\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{y+3}-3$$

(2の2乗を(y+3)繰り返したもの引く3)

原始帰納的でないことの証明

原始帰納的と仮定→矛盾（背理法）

A(x, y)を原始帰納的と仮定すると、

A(x, x)は1変数の原始帰納関数。

次ページの定理より、

$A(x, x) \leqq A(c, \max(x)) = A(c, x)$

となるcが存在する。ここでx=c+1とおくと、

$A(c+1, c+1) \leqq A(c, c+1)$

となり、次ページの補題3に反する。

ゆえに、A(x, y)は原始帰納的ではない。

証明の基礎

補題：

1. $A(x+1, y) \geqq y+2$
2. $A(x, y) \geqq y+1$
3. $A(x, y) < A(x, y+1) \leqq A(x+1, y)$
4. 任意のc_1, c_2に対して$A(c_1, A(c_2, x)) \leqq A(c_3, x)$となる$c_3$が存在する

定理：

任意の原始帰納関数$f(x_1, \ldots, x_n)$に対して

$f(x_1, \ldots, x_n) \leqq A(c, \max(x_1, \ldots, x_n))$

となるcが存在する。

原始帰納的な集合と述語

特徴関数C_S, C_p

集合$S \in N^n$

$C_S(x_1, \ldots, x_n)=0$ （$(x_1, \ldots, x_n) \in S$のとき）

$C_S(x_1, \ldots, x_n)=1$ （$(x_1, \ldots, x_n) \notin S$のとき）

が原始帰納的であるとき、Sは原始帰納的集合。

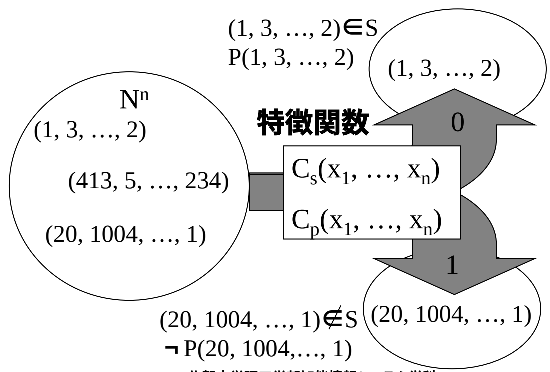
述語$P(x_1, \ldots, x_n)$

$C_p(x_1, \ldots, x_n)=0$ （$P(x_1, \ldots, x_n)$のとき）

$C_p(x_1, \ldots, x_n)=1$ （$\neg P(x_1, \ldots, x_n)$のとき）

が原始帰納的であるとき、Pは原始帰納的述語。

イメージ図



原始帰納的述語と関数の例 1

(13) **述語** $x=y$
(14) **述語** $x<y$
(15) **述語** $x\leqq y$
(16) **関数** $\max(x, y)$
(17) **関数** $\min(x, y)$
(18) **関数** $\max(x_1, \ldots, x_n)$
(19) **述語** $x|y$ (x **は** y **を割り切る**)
(20) **述語** $P_r(x)$ (x **は素数である**)

述語x=yの特徴関数

$C_p=sg(|x-y|)$

何故なら、

$x=y$ **のとき** 0

$x>y$ **または** $x<y$ **のとき** 1

となる。

述語x<yの特徴関数

$C_p=1-sg(y\dot{-}x)$

何故なら、

$x<y$ **のとき** 0

$x=y$ **または** $x>y$ **のとき** 1

となる。

原始帰納的述語と関数の例 2

(21) **述語** x/y (x **を** y **で割ったときの商**)
(22) **第** $n+1$ **番目の素数を表す関数** p_n
(23) **関数**

$l(a)=a$ **の素因数分解における** 0 **でない数の個数**
($a\neq 0$ **のとき**)
$l(a)=0$ ($a=0$ **のとき**)

(24) a **と** i **の関数**

$(a)_i=a$ **の素因数分解における** p_i **のべき数** ($a\neq 0$ **のとき**)
$(a)_i=0$ ($a=0$ **のとき**)

原始帰納的述語と関数の例 3

(25) **関数**

$$x \circ y = x \bullet \prod_{i<l(y)} p_{l(x)+i}{}^{(y)_i}$$

原始帰納的集合の性質

集合S, R∈N^n**が原始帰納的であれば、**

$\overline{S}=N^n - S$

S∪R

S∩R

も原始帰納的。

原始帰納的述語の性質 1

$P(x_1, \ldots, x_r)$**を原始帰納的述語とし、**
$h_1, \ldots, h_r$**を**n**変数の原始帰納的関数とする。**

このとき述語
$P(h_1(x_1, \ldots, x_n), \ldots, h_r(x_1, \ldots, x_n))$
は原始帰納的。

合成による特徴関数

$h_1, \ldots, h_r$**は**n**変数の原始帰納的関数であるので、**$P(x_1, \ldots, x_r)$**の特徴関数**$C_p(x_1, \ldots, x_r)$**に**$h_1, \ldots, h_r$**を代入した**
$C_p(h_1(x_1, \ldots, x_n), \ldots, h_r(x_1, \ldots, x_n))$**も**
特徴関数とすることができる

例：$x<y$**と**$a+b$, $c-d$
新しい述語は、$(a+b) < (c-d)$
特徴関数は、$1-sg((a+b)\dot{-}(c-d))$

原始帰納的述語の性質 2

$g_1, \ldots, g_{m+1}$をn変数の原始帰納的関数とし、
$P_1, \ldots, P_m$をn変数の原始帰納的述語で、
各$(x_1, \ldots, x_n)$に対して高々1個の$P_i(x_1, \ldots, x_n)$が真になるものとする。

このとき関数

$f(x_1, \ldots, x_n)=g_1(x_1, \ldots, x_n)$	$(P_1(x_1, \ldots, x_n)$のとき)
…	
$f(x_1, \ldots, x_n)=g_m(x_1, \ldots, x_n)$	$(P_m(x_1, \ldots, x_n)$のとき)
$f(x_1, \ldots, x_n)=g_{m+1}(x_1, \ldots, x_n)$	(それ以外のとき)

は原始帰納的。→証明

原始帰納的述語の性質 3

$P(x_1, \ldots, x_n), Q(x_1, \ldots, x_n)$を
原始帰納的述語とすれば、述語

$\neg P(x_1, \ldots, x_n)$
$P(x_1, \ldots, x_n) \vee Q(x_1, \ldots, x_n)$
$P(x_1, \ldots, x_n) \wedge Q(x_1, \ldots, x_n)$
$P(x_1, \ldots, x_n) \rightarrow Q(x_1, \ldots, x_n)$
$P(x_1, \ldots, x_n) \leftarrow Q(x_1, \ldots, x_n)$

は原始帰納的。→証明

原始帰納的述語の性質 4

$P(x_1, \ldots, x_n, y)$を原始帰納的述語とすれば、述語

$(\exists y)_{<z} P(x_1, \ldots, x_n, y)$
$\Leftrightarrow P(x_1, \ldots, x_n, 0) \vee \ldots \vee P(x_1, \ldots, x_n, z-1)$
$(\forall y)_{<z} P(x_1, \ldots, x_n, y)$
$\Leftrightarrow P(x_1, \ldots, x_n, 0) \wedge \ldots \wedge P(x_1, \ldots, x_n, z-1)$

は原始帰納的。→証明

部分帰納的関数と帰納的関数

部分帰納的(partial recursive)関数
初期関数(1), (2), (3)に操作(I), (II), **(III)**を
有限回適用して定義された関数。

新操作

帰納的(recursive)関数
(一般帰納的(general recursive)関数)
初期関数(1), (2), (3)に操作(I), (II), **(III')**を
有限回適用して定義された関数。

2つの新操作

(III) **全域的であることが保証されない。**

関数$g(x_1, \ldots, x_n, y)$**から関数**$f(x_1, \ldots, x_n)$**を以下の操作で作る。**

$$f(x_1, \ldots, x_n)= \mu y(g(x_1, \ldots, x_n, y)=0)$$

(III´) **全域的であることが保証される。**

正則関数$g(x_1, \ldots, x_n, y)$**から関数**$f(x_1, \ldots, x_n)$**を以下の操作で作る。**

$$f(x_1, \ldots, x_n)= \mu y(g(x_1, \ldots, x_n, y)=0)$$

μ作用素
(μ-operator)

$n+1$**変数の述語から**n**変数の関数を作る操作**

定義

述語$P(x_1, \ldots, x_n, y)$**に対して**

$\mu yP(x_1, \ldots, x_n, y)$

$=\min\{y|P(x_1, \ldots, x_n, y)\}$

($(\exists y)P(x_1, \ldots, x_n, y)$**のとき**)

$\mu yP(x_1, \ldots, x_n, y)$

=**無定義**

($\neg(\exists y)P(x_1, \ldots, x_n, y)$**のとき**)

正則とは

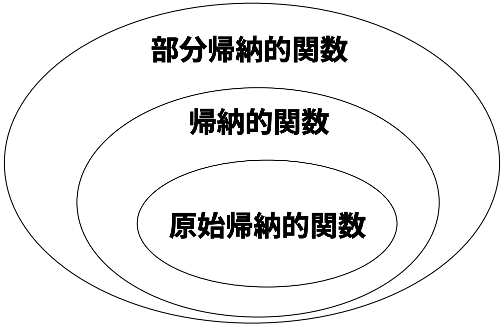
述語$P(x_1, \ldots, x_n, y)$**が正則**

＝ 任意の$(x_1, \ldots, x_n)$**に対して**$P(x_1, \ldots, x_n, y)$**を真とする**y**が存在する。**

関数$g(x_1, \ldots, x_n, y)$**が正則**

＝任意の$(x_1, \ldots, x_n)$**に対して**$g(x_1, \ldots, x_n, y)=0$**となる**y**が存在する。**

各関数の関係



最後に

- **次回は、**
  - Turing**機械**?