

## テーマ: 巨大数論②

【定義】 原始帰納関数 primitive recursive function (原始再帰関数)

\* 以下で関数というときは、定義域、値域ともに自然数 ( $N_0$ ) の範囲のものだけを考える。

\* 全ての原始帰納関数の集合のことを (複雑性クラス) PR と呼ぶ。

\*  $n$  個の引数を取る関数 ( $n$  項演算) のことを  $n$ -ary と略記する。

\* 作用素 operator とは、関数を関数に変換する写像のことである。

1° 定数関数 constant function :  $n$ -ary (0-ary と考える場合もある)

$$\text{cons}^n(x_1, \dots, x_n) = 0 \quad \text{但し } 0 \leq n$$

2° 後者関数 successor function : 1-ary

$$\text{suc}(x) = x' \quad [= x + 1]$$

3° 射影関数 projection function :  $n$ -ary

$$\text{proj}_i^n(x_1, \dots, x_n) = x_i \quad \text{但し } 1 \leq i \leq n$$

4° 合成作用素 composition operator :  $m$ -ary と  $m$  個の  $n$ -ary から  $n$ -ary を作る

$$C : [f(x_1, \dots, x_n), g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n)] \rightarrow [h(x_1, \dots, x_n)] \quad f, g_1, \dots, g_m \in PR$$

$$h(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n)) \quad \text{但し } 0 < m, 0 \leq n$$

5° 原始帰納作用素 primitive recursion operator :  $n$ -ary と  $(n+2)$ -ary から  $(n+1)$ -ary を作る

$$P : [f(x_1, \dots, x_n), g(p, q, x_1, \dots, x_n)] \rightarrow [h(r, x_1, \dots, x_n)] \quad f, g \in PR$$

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

$$h(m+1, x_1, \dots, x_n) = g(h(m, x_1, \dots, x_n), m, x_1, \dots, x_n) \quad \text{但し } 0 \leq n$$

このとき、 $f$  を 基底関数 base function、 $g$  を ステップ関数 recursion step function と呼ぶ。

また、 $g$  の第 1 項を 帰納項 recursive argument、第 2 項を 後退項 regressive argument と呼ぶ。

6° 原始帰納関数の定義

1° ~ 3° の関数は原始帰納的である (これらを総称して 始式 という)。

4° ~ 5° の作用素によって原始帰納関数を有限回変換した関数もまた原始帰納的である。

そして、このような関数のみが原始帰納的である。

\* 合成作用素 (4°) の  $m, n=1$  の場合への特殊化

$$C : [f(x), g(x)] \rightarrow [h(x)] \quad f, g \in PR$$

$$h(x) = f(g(x)) = f \circ g(x) \quad \text{特に } f(x) = g(x) \text{ のとき } h(x) = f^2(x) = g^2(x)$$

\* 原始帰納作用素 (5°) の  $n=1$  の場合への特殊化

$$P : [f(x), g(p, q, x)] \rightarrow [h(r, x)] \quad f, g \in PR$$

$$h(0, x) = f(x)$$

$$h(m+1, x) = g(h(m, x), m, x)$$

【定理】 任意の自然数は原始帰納関数である。

[証明]  $\text{cons}^n(x_1, \dots, x_n) = 0$  は原始帰納関数。  $f(x_1, \dots, x_n) = m \in PR$  と仮定する。

$$C : [\text{suc}(x), f(x_1, \dots, x_n) = m] \rightarrow [g(x_1, \dots, x_n) = \text{suc}(f(x_1, \dots, x_n)) = m + 1] \quad \text{suc}, f \in PR$$

よって  $g(x_1, \dots, x_n) = m + 1 \in PR$  も成立。(Q.E.D.)

【定理】 加法  $plus(x, y) = x + y$  は原始帰納函数である。

[証明] 以下で定義される函数  $g(x_1, x_2, x_3)$ 、 $plus(x, y)$  は原始帰納函数である。

$$\begin{aligned} g(x_1, x_2, x_3) &= suc(proj_3^3(x_1, x_2, x_3)) \\ plus(m, 0) &= proj_1^1(m) \\ plus(m, n+1) &= g(m, n, plus(m, n)) \end{aligned}$$

$plus(x, y) = x + y$  となることの数学的帰納法による証明。

$$\begin{aligned} plus(x, 0) &= proj_1^1(x) = x = x + 0 \\ plus(x, k) &= x + k \text{ と仮定 (} k \text{ は任意の自然数)。} \\ plus(x, k+1) &= g(x, k, plus(x, k)) \end{aligned}$$

仮定を用いて、

$$= g(x, k, x+k) = suc(proj_3^3(x, k, x+k)) = (x+k)' = x + (k+1)$$

以上から、任意の自然数  $x$  に対して  $plus(x, y) = x + y$ 。(Q.E.D.)

【定理】 ハイパー演算子の 2 変数函数表式  $hyper[n](x, y)$  は原始帰納函数である (但し  $1 \leq n, 0 \leq x, 2 \leq y$ )。

[証明] 数学的帰納法による。

$$\begin{aligned} hyper[1](x, y) &= plus(x, y) \in PR \\ hyper[k](x, y) &\in PR \text{ と仮定。 } g(x, y) = proj_1^2(x, y) = x \text{ と定義する (} g \in PR \text{)。} \\ hyper[k+1](x, y) &= hyper[k](x, hyper[k](x, hyper[k](\dots hyper[k](x, x)\dots))) \\ &= hyper[k](g(x, x), hyper[k](g(x, x), hyper[k](\dots hyper[k](g(x, x), hyper[k](x, x))\dots))) \end{aligned}$$

これは合成作用素によって原始帰納函数を有限回変換した函数だから、

$$hyper[k+1](x, y) \in PR$$

以上から、 $hyper[n](x, y) \in PR$ 。(Q.E.D.)

※このゆえ、全てのハイパー演算 (乗法、冪乗、超冪、……) は原始帰納函数である。

《さまざまの原始帰納函数》

\* 乗法  $times(x, y) = xy$

$$\begin{aligned} times(m, 0) &= cons^1(m) \\ times(m, n+1) &= g(m, n, times(m, n)) = plus(m, times(m, n)) \end{aligned}$$

\* 冪乗  $power(x, y) = x^y$

$$\begin{aligned} power(m, 0) &= suc(cons^1(m)) \\ power(m, n+1) &= g(m, n, power(m, n)) = times(m, power(m, n)) \end{aligned}$$

\*  $k$  番目のハイパー演算子の 2 変数函数表式  $h_k(x, y) = hyper[k](x, y)$  但し  $2 \leq k, 0 \leq x, 2 \leq y$  の場合

$$\begin{aligned} h_k(m, 2) &= hyper[k-1](m, m) \\ h_k(m, n+1) &= g(m, n, h_k(m, n)) = hyper[k-1](m, h_k(m, n)) \end{aligned}$$

\* 階乗  $bikkuri(x) = x!$

$$\begin{aligned} bikkuri(0) &= suc(cons^0()) \\ bikkuri(n+1) &= g(n, bikkuri(n)) = times(suc(n), bikkuri(n)) \end{aligned}$$

\* 超階乗  $sugokubikkuri(x) = x\$$

$$sugokubikkuri(x) = hyper[4](bikkuri(x), bikkuri(x))$$

《変換回数の変数化》

原始帰納がいかなる操作であるのかを考える。

$$h(m+1, x) = g(h(m, x), m, x)$$

かかる原始帰納の漸化式において、 $B(m) = h(m, x)$ 、 $C(y) = g(y, m, x)$ と置けば、

$$B(m+1) = C(B(m))$$

$B(1) = C(B(0))$ 、 $B(2) = C(B(1)) = C(C(B(0))) = C^2(B(0))$ 、 $B(3) = C^3(B(0))$ などとなって、けっきょく

$$B(m) = C^m(B(0))$$

と書くことができる。すなわち、原始帰納とは、合成変換の回数を変数とする函数を作り出すような操作のことである。変換回数を変数化することを「数えあげ」と表現する。原始帰納は合成変換を数えあげる。

《対角化 diagonalization》

函数  $c \in PR$  に対して合成函数の列  $b_n$  を  $b_n(x) = c(c(c(\dots c(x)\dots))) = c^n(x)$  として定義する。ここで  $b_n \in PR$  となるのは自明。各  $n, x$  の値に対する  $b_n(x)$  の値を table 形式で表示する。

$$b_n(x) = \left\{ \begin{array}{c|cccc} & 1 & 2 & \dots & x & \dots \\ \hline 1 & c(1) & c(2) & \dots & c(x) & \dots \\ 2 & c^2(1) & c^2(2) & \dots & c^2(x) & \dots \\ \vdots & \vdots & \vdots & & \vdots & \\ n & c^n(1) & c^n(2) & \dots & c^n(x) & \dots \\ \vdots & \vdots & \vdots & & \vdots & \end{array} \right\}$$

次に、函数  $b$  を  $b(x) = b_x(x) = c^x(x)$  として定義すると、 $b(1) = c(1)$ 、 $b(2) = c^2(2)$ 、 $b(3) = c^3(3)$  などとなって、この函数は上掲 table の対角線上の値を拾いあげたものに相当する。 $c^m(x) < c^{m+1}(x)$  であれば、 $n < x$  において  $b_n(x) = c^n(x) < c^x(x) = b(x)$  となるから、 $b$  は全ての  $b_n$  よりも急増加する函数である（対角線論法）。

函数  $b$  は函数  $c$  に対する合成変換の回数を数えあげている。かように、変換回数を数えあげるとは、ある函数列  $f_n$  の対角線上の値を拾いあげて、 $f_n$  の支配函数  $f$  を構成することにほかならないから、これを対角化と表現する。

※上記の方法で構成される、函数列  $b_n$  の支配函数  $b$  は、しかしながら合成変換の回数を数えあげているわけだから、やはり原始帰納法の範疇であることに変わりはない。次に、原始帰納を対角化して、原始帰納変換の回数を数えあげるような函数を構成し、そのような函数が原始帰納的でありうるかどうかを考察する。

【定義】 原始帰納函数の列  $\varphi_n \in PR$  を以下の通り定義する。

$$\varphi_0(x) = suc(x)$$

$$\varphi_{n+1}(x) = \varphi_n^x(x)$$

$\varphi_{n+1}$  は  $\varphi_n$  に対する合成変換の回数を数えあげており、 $\varphi_n$  に原始帰納変換を 1 回施したものに相当する。

$$Ex. \varphi_1(x) = \varphi_0^x(x) = suc^x(x) = 2x$$

$$\varphi_2(x) = \varphi_1^x(x) = 2^x \cdot x$$

【定義】 支配 domination ( $\geq^*$ )

1-ary 函数  $f(x)$  が n-ary 函数  $g(x_1, \dots, x_n)$  を支配する ( $f \geq^* g$ ) とは、

$$\forall (x_1, \dots, x_n) \in N_0, \exists \xi \in N_0, [\xi < x_1, \dots, x_n \Rightarrow g(x_1, \dots, x_n) \leq f(\max(x_1, \dots, x_n))]$$

となることをいう。特に  $n=1$  の場合は以下の通り。

$$\forall x \in N_0, \exists \xi \in N_0, [\xi < x \Rightarrow g(x) \leq f(x)]$$

【補題】 (1)  $x < \varphi_n(x)$

(2)  $(n < m, 0 < x) \Rightarrow \varphi_n(x) < \varphi_m(x)$

(3)  $x \leq y \Rightarrow \varphi_n(x) \leq \varphi_n(y)$

[証明] 全て略。おおむね数学的帰納法による。(Q.E.D.)

【定理】 いかなる  $f \in PR$  に対しても、ある適当な  $n \in N_0$  が存在して、全ての  $\bar{x} \in N_0$  に対して  $f(\bar{x}) \leq \varphi_n(\max(\bar{x}))$  が成立する。このとき  $\varphi_n \geq^* f$  も成立している。但し  $\bar{x} = (x_1, \dots, x_k)$  である。

[証明]  $f$  の構成にかんする数学的帰納法。

1°  $f$  が始式するとき

$f$  が始式であるなら  $\varphi_0 \geq^* f$  となることは自明。

2°  $f$  が  $g, h_1, \dots, h_m \in PR$  から合成されているとき

$f(\bar{x}) = g(h_1(\bar{x}), \dots, h_m(\bar{x}))$  として、 $\varphi_a \geq^* g, \varphi_b \geq^* h_1, \dots, \varphi_e \geq^* h_m$  と仮定するとき、 $\varphi_v \geq^* f$  も成立することを示す。じっさい、 $\max(a, b, \dots, e) = n$  と置くと、

$$\varphi_n(\max(\bar{y})) \geq g(\bar{y}), \varphi_n(\max(\bar{x})) \geq h_1(\bar{x}), \dots, \varphi_n(\max(\bar{x})) \geq h_m(\bar{x})$$

となるから、

$$\begin{aligned} f(\bar{x}) &= g(h_1(\bar{x}), \dots, h_m(\bar{x})) \\ &\leq \varphi_n(\max(h_1(\bar{x}), \dots, h_m(\bar{x}))) \\ &\leq \varphi_n(\max(\varphi_n(\max(\bar{x})), \dots, \varphi_n(\max(\bar{x})))) = \varphi_n(\varphi_n(\max(\bar{x}))) = \varphi_n^2(\max(\bar{x})) \\ &\leq \varphi_n^{(\max(\bar{x}))}(\max(\bar{x})) \quad [\because 2 \leq \max(\bar{x}) \text{ として補題 (1)}] \\ &= \varphi_{n+1}(\max(\bar{x})) \end{aligned}$$

ということが示され、 $\varphi_{n+1} \geq^* f$  が成立する。

3°  $f$  が  $g, h \in PR$  から原始帰納されているとき

$f(0, \bar{x}) = g(\bar{x}), f(y+1, \bar{x}) = h(f(y, \bar{x}), y, \bar{x})$  とする。証明の第1段階として、

$$\varphi_n(\max(\bar{x})) \geq g(\bar{x}), \varphi_n(\max(z, y, \bar{x})) \geq h(z, y, \bar{x})$$

と仮定すると、 $T(m) : \varphi_n^{m+1}(\max(m, \bar{x})) \geq f(m, \bar{x})$  が成立することを示す (数学的帰納法)。

$$\begin{aligned} f(0, \bar{x}) &= g(\bar{x}) \\ &\leq \varphi_n(\max(\bar{x})) = \varphi_n^1(\max(0, \bar{x})) \end{aligned}$$

よって  $T(0)$  は成立。次に、 $T(k)$  を仮定すると  $T(k+1)$  も成立する。じっさい、

$$\begin{aligned} f(k+1, \bar{x}) &= h(f(k, \bar{x}), k, \bar{x}) \\ &\leq \varphi_n(\max(f(k, \bar{x}), k, \bar{x})) \\ &\leq \varphi_n(\max(\varphi_n^{k+1}(\max(k, \bar{x})), k, \bar{x})) \quad [\because \text{仮定 } T(k), \text{ 補題 (3)}] \\ &\leq \varphi_n(\varphi_n^{k+1}(\max(k+1, \bar{x}))) \quad [\because \text{補題 (1)}] \\ &= \varphi_n^{k+2}(\max(k+1, \bar{x})) \end{aligned}$$

以上から、任意の自然数  $m$  に対して  $T(m)$  が成立することが示された。

このゆえ、 $\varphi_v \geq^* f$  も示される。じっさい、

$$\begin{aligned} f(m, \bar{x}) &\leq \varphi_n^{m+1}(\max(m, \bar{x})) \quad [\because T(m)] \\ &= \varphi_n(\varphi_n^m(\max(m, \bar{x}))) \\ &\leq \varphi_n(\varphi_n^{\max(m, \bar{x})}(\max(m, \bar{x}))) = \varphi_n(\varphi_{n+1}(\max(m, \bar{x}))) \\ &\leq \varphi_{n+1}(\varphi_{n+1}(\max(m, \bar{x}))) = \varphi_{n+1}^2(\max(m, \bar{x})) \\ &\leq \varphi_{n+1}^{\max(m, \bar{x})}(\max(m, \bar{x})) = \varphi_{n+2}(\max(m, \bar{x})) \end{aligned}$$

となつて、 $\varphi_{n+2} \geq^* f$  が成立する。

4° 以上から、数学的帰納法により、任意の原始帰納函数に対して定理が成立する。(Q.E.D.)

《原始帰納変換の対角化》

上で定義した原始帰納関数の列  $\varphi_n \in PR$  について、各  $n, x$  の値に対する  $\varphi_n(x)$  の値を table 形式で表示する。

$$\varphi_n(x) = \left( \begin{array}{c|cccc} & 0 & 1 & \cdots & x & \cdots \\ \hline 0 & \varphi_0(0) & \varphi_0(1) & \cdots & \varphi_0(x) & \cdots \\ 1 & \varphi_1(0) & \varphi_1(1) & \cdots & \varphi_1(x) & \cdots \\ \vdots & \vdots & \vdots & & \vdots & \\ n & \varphi_n(0) & \varphi_n(1) & \cdots & \varphi_n(x) & \cdots \\ \vdots & \vdots & \vdots & & \vdots & \end{array} \right)$$

次に、関数  $\varphi$  を  $\varphi(x) = \varphi_x(x)$  として定義すると、この関数は上掲 table の対角線上の値を拾いあげたものに相当し、原始帰納を対角化している。すなわち、関数  $\varphi$  は原始帰納変換の回数を数えあげている。

対角線論法により、関数  $\varphi$  が全ての  $\varphi_n$  よりも急増加することが分かる。然り而して、以下の定理が示される。

【定理】 関数  $\varphi$  は原始帰納関数ではない。

[証明] 背理法による。

$\varphi$  が原始帰納関数であると仮定する。すると、ある適当な  $n$  が存在して、 $\varphi_n \geq^* \varphi$  となる筈である。

ところで、 $\varphi \geq^* \varphi_n$  である。なんとなれば、 $n < x$  において

$$\varphi_n(x) < \varphi_x(x) = \varphi(x) \quad [\because \text{補題 (2)}]$$

となるから、 $\varphi \neq \varphi_n$  であるのに  $\varphi_n \geq^* \varphi$  かつ  $\varphi \geq^* \varphi_n$ 。これは矛盾。(Q.E.D.)

すなわち、原始帰納を対角化して、原始帰納変換の回数を数えあげるような関数を構成すると、そのような関数はあらゆる原始帰納関数の支配関数となり、一方でいかなる原始帰納関数も原始帰納関数によって支配されるから、かかる関数はもはや原始帰納的ではありえないのである。

【定理】 アッカーマン関数は原始帰納関数ではない。

[証明] アッカーマン関数が原始帰納変換の回数を数えあげていることを示せば必要充分。

アッカーマン関数の漸化式は以下の通り。

$$Ack(0, n) = n + 1$$

$$Ack(m + 1, 0) = Ack(m, 1)$$

$$Ack(m + 1, n + 1) = Ack(m, Ack(m + 1, n))$$

ここで  $f(n) = Ack(m, n)$ ,  $g(n) = Ack(m + 1, n)$  と置くと、

$$g(n + 1) = Ack(m + 1, n + 1) = Ack(m, Ack(m + 1, n)) = f(g(n))$$

$$g(n) = f^n(g(0)) = f^n(Ack(m + 1, 0)) = f^n(Ack(m, 1)) = f^n(f(1)) = f^{n+1}(1)$$

となつて、 $g$  は  $f$  に対する合成変換の回数を数えあげている。合成変換の回数を数えあげるのは原始帰納。

したがって、 $g$  は  $f$  に原始帰納変換を 1 回施したものの。

このことは、かえりみて、 $Ack(m + 1, n)$  が  $Ack(m, n)$  に原始帰納変換を 1 回施したものであることを示している。すなわち、変数  $m$  は原始帰納変換の回数を変数化したものである。

以上から、アッカーマン関数は原始帰納変換の回数を数えあげている。(Q.E.D.)

※同様に、ハイパー演算子の 3 変数関数表式  $hyper(x, n, y)$  も原始帰納関数ではない。変数  $n$  が原始帰納変換の回数を数えあげているからである。アッカーマン関数は全ての原始帰納関数を支配する。すなわち、いかなる  $f \in PR$  に対しても、 $f(\bar{x}) < Ack(c, \max(\bar{x}))$  となるような定数  $c$  が存在する。

### 《原始帰納函数と一般帰納函数》

原始帰納函数は全ての一般帰納函数 **general recursive function** の集合（複雑性クラス **R**）の真部分集合である。一般帰納函数（または、たんに帰納函数）とは、原始帰納函数の条件  $1^\circ \sim 5^\circ$  に  $\mu$  作用素（最小化作用素 **minimization operator**）を加えて定義されるもので、 $\mu$  再帰函数とも呼ばれる。

チャーチ=チューリングの提唱とは、一般帰納函数のクラスを（チューリング）計算可能函数 **computable function** のクラスと同一視しようというものである。この提唱を受け入れるならば、**algorithm** を示せる函数であるということが、すなわち一般帰納函数であるということになる。

複雑性クラス **R** は、合成、原始帰納、 $\mu$  の各作用素において閉じている。すなわち、計算可能函数にこれらの作用素をどんな順番で適用しても、**計算不能函数 non-computable function** を構成することはできない。

【定理】 アッカーマン函数は一般帰納函数である。

〔証明〕 チャーチ=チューリングの提唱を受け入れるならば、計算手順の存在を示せば必要充分。

そして、計算手順が存在することは自明。（**Q.E.D.**）

### 【定義】 2重帰納函数 **double recursive function**

原始帰納変換を数えあげる作用素を **2重帰納作用素** と呼ぶ。

2重帰納函数とは、合成、原始帰納、2重帰納の作用素によって始式を有限回変換した函数であって、2重帰納変換を少なくとも1回は含む函数のことである。

### 《アッカーマン函数は2重帰納函数である》

アッカーマン函数は原始帰納変換を数えあげているから、2重帰納函数である。

アッカーマン函数の漸化式をながめてみよう。

$$Ack(0, n) = n + 1$$

$$Ack(m + 1, 0) = Ack(m, 1)$$

$$Ack(m + 1, n + 1) = Ack(m, Ack(m + 1, n))$$

第3式は、 $Ack(m + 1, n)$  の値から  $Ack(m + 1, n + 1)$  の値を求めている。すなわち、 $n$  についての原始帰納法が用いられている。次に第2式を変形してみると、

$$Ack(m + 1, 0) = Ack(m, 1) = Ack(m - 1, Ack(m, 0))$$

となつて、この式が  $Ack(m, 0)$  の値から  $Ack(m + 1, 0)$  の値を求めていることが分かる。すなわち、 $m$  についての原始帰納法が用いられている。

かように、アッカーマン函数の定義は、まず  $n$  についての原始帰納法を用い、次に各  $n$  に対して  $m$  についての原始帰納法を用いている。このことから、アッカーマン函数が2重帰納函数であるということが分かる。

### \*References

テキスト「巨大数論」以外に参照したものを挙げる。下記のほか多数（おもに Web 上の PDF ファイル）。

特に函数列  $\varphi_n$  の構成とその対角化は、全面的に[3]の記述に与るものであり、筆者 TaniR は殆んど手を加えていない。

[1] 前原昭二『数学基礎論入門』朝倉書店、1977年

[2] Wikipedia 日本語版「原始再帰関数」「 $\mu$ 再帰関数」など（いずれも2009年9月中旬の版）

[3] 照井一成「再帰的関数論」2005年 <http://www.kurims.kyoto-u.ac.jp/~terui/mita05.pdf>

[4] 和手正道「帰納的関数論」2007年 <http://members3.jcom.home.ne.jp/9668pgrt/text/rec.pdf>

[5] 講義資料 <http://www.kurims.kyoto-u.ac.jp/~cs/lecture2009/lecture09computability.pdf>

[6] 講義資料 <http://www.goto.info.waseda.ac.jp/~goto/infomath/infomath-5-induction.pdf>