

## テーマ: 巨大数論③

## 巨大数生成の構造分析

巨大数の構成という観点から、グラハム数 ( $G$ ) とふいつしゅ数 ver.1 ( $F_1$ ) とを比較する。

グラハム数は  $G = G^{64}(4)$  として定義される。  $G(x) = 3 \uparrow^x 3 = \text{hyper}(3, x+2, 3)$  であるが、3変数ハイパー演算子が2重帰納函数であったことに鑑み、函数  $G$  もまたそうであるように思われる。だが本当だろうか。  $\text{hyper}(x, n, y)$  は原始帰納法の範疇を超えているとしても、そこから2変数を固定した  $G(x) = \text{hyper}(3, x+2, 3)$  までは原始帰納法の範疇を超えている (あらゆる  $f \in PR$  を支配する) とはかぎらないのではないか。

そこで、瑣末な部類には入るけれども、一応このことを証明しておく。

【定理】 グラハム函数  $G$  は原始帰納函数ではない。

[証明] 背理法。いかなる  $f \in PR$  に対しても、ある適当な  $c \in N_0$  が存在して、全ての  $x \in N_0$  に対して

$$f(x) < \text{Ack}(c, x) = \text{hyper}(2, c, x+3) - 3$$

が成立することを利用する。

$G(x) = \text{hyper}(3, x+2, 3) \in PR$  と仮定。すると、ある適当な  $c \in N_0$  が存在して、全ての  $x \in N_0$  に対して

$$\text{hyper}(3, x+2, 3) < \text{hyper}(2, c, x+3) - 3$$

となる筈である。ところで、このような  $c$  は存在しない。なんとすれば、  $c-2 < x$  において

$$\text{hyper}(2, c, x+3) - 3$$

$$< \text{hyper}(2, c, x+3)$$

$$< \text{hyper}(2, c, \text{hyper}(3, x+1, 3)) \quad [\because x+3 < \text{hyper}(3, x+1, 3) \text{ となるのは自明}]$$

$$< \text{hyper}(3, x+1, \text{hyper}(3, x+1, 3)) \quad [\because c \leq x+1]$$

$$< \text{hyper}(3, x+2, 3)$$

となるから。このゆえ仮定は偽。(Q.E.D.)

上掲定理から、函数  $G$  が2重帰納函数であることが確定する。すなわち、  $G^{64}(x)$  とは、1回の2重帰納変換と63回の合成変換とから構成される巨大函数であり、その  $x$  に4を代入することでグラハム数が生成される。

他方、ふいつしゅ数 ver.1 を生成する巨大函数  $F_1$  は、定義より  $f(x) = x+1 \in PR$  に  $S$  変換を相当回数施したものである。1回の  $S$  変換は1回の2重帰納変換に等価である。すると、  $S$  変換を2回適用した時点で、構成される巨大函数は2回の2重帰納変換を施されており、すでに  $G^{64}(x)$  を支配している (変換回数の多寡は端的に支配関係を示すものではないが、この場合  $G(x) \approx \text{Ack}(x, y)$  と考えてよいから比較可能である。具体的計算は次頁)。

$SS$  変換は、  $S$  変換の変換回数を、それまでに構成された巨大函数を利用して指定するものである。  $SS$  変換により、  $S$  変換は数え切れないほどの回数適用されるが、最終的に構成される函数  $F_1$  は  $S$  変換の回数を変数化しておらず、2重帰納的であるに留まる。すなわち、函数  $F_1$  は一定回数の2重帰納変換から構成される巨大函数であり、その  $x$  に巨大定数を代入することでふいつしゅ数 ver.1 が生成される。

※一定回数というのは、正確には以下で示す通りの回数である。すなわち、  $[3, x+1, S]$  を  $n$  回  $SS$  変換するとき、えられる自然数を  $x_n$ 、えられる函数を  $f_n$  と置くと、  $F_1 = f_{63}$  における  $S$  変換の適用回数  $\tau$  は

$$\tau = 4 + \sum_{j=1}^{62} \left\{ 4 * \prod_{k=1}^j f_k(x_k) \right\}$$

として表現される。函数  $F_1$  は  $\tau$  回の2重帰納変換から構成される。ここで  $\tau$  が定数であることに注意。

【定義】3重帰納関数 triple recursive function

2重帰納変換を数えあげる作用素を3重帰納作用素と呼ぶ。

3重帰納関数とは、合成、原始帰納、2重帰納、3重帰納の作用素によって始式を有限回変換した関数であって、3重帰納変換を少なくとも1回は含む関数のことである。

【定義】3変数アッカーマン関数 ternary Ackermann function

$$A_T(0,0,z) = z + 1$$

$$A_T(x+1,0,z) = A_T(x,z,z)$$

$$A_T(x,y+1,0) = A_T(x,y,1)$$

$$A_T(x,y+1,z+1) = A_T(x,y,A_T(x,y+1,z))$$

3変数アッカーマン関数は3重帰納関数である

$x=0$ のとき、以下で示される通り、 $A_T(x,y,z) = Ack(y,z)$ が成立する。

$$A_T(0,0,z) = z + 1; A_T(0,y+1,0) = A_T(0,y,1); A_T(0,y+1,z+1) = A_T(0,y,A_T(0,y+1,z))$$

また、 $f(y,z) = A_T(x,y,z)$ ,  $g(y,z) = A_T(x+1,y,z)$ と置くと、

$$g(0,z) = f(z,z); g(y+1,0) = g(y,1); g(y+1,z+1) = g(y,g(y+1,z))$$

と書くことができ、 $g$ は $f$ から2重帰納的である。

以上から、 $A_T(x,y,z)$ は $A_T(0,y,z) = Ack(y,z)$ に対して2重帰納変換を $x$ 回施したものに相当する。

すなわち、変数 $x$ は2重帰納変換を数えあげており、3変数アッカーマン関数は1回の3重帰納変換から構成される3重帰納関数である。

$G \ll g_2(2)$ となることの、3変数アッカーマン関数を介した証明

$f(x) = A_T(n,x,x)$ ,  $g(x) = A_T(n+1,x,x)$ と置くと、 $g$ は $f$ に対して $S$ 変換を1回適用したものである(定義が一致するため)。すなわち、関数 $A_T$ も関数 $F_1$ も同一の2重帰納変換を含んでいる。 $[3,x+1]$ を $n$ 回 $S$ 変換するとき、えられる関数を $g_n(x)$ とすると、 $g_1(x) = Ack(x,x) = A_T(0,x,x)$ だから、 $g_n(x) = A_T(n-1,x,x)$ となる。

ここで、 $G \ll g_2(2)$ であることを示そう。

$$g_2(2) = A_T(1,2,2) = A_T(1,1,A_T(1,1,A_T(1,2,0))) = A_T(1,1,A_T(1,1,A_T(1,1,1))) = A_T(1,1,A_T(1,1,g_2(1)))$$

簡単な計算から、 $g_2(1) = 61$ であることが分かって、以下の等式が成立する。

$$g_2(2) = A_T(1,1,A_T(1,1,61))$$

次に、 $A_T(1,1,x)$ がどんな数になるか考えてみる。

$$A_T(1,1,x+1) = A_T(1,0,A_T(1,1,x)) = A_T(0,A_T(1,1,x),A_T(1,1,x)) = Ack(A_T(1,1,x),A_T(1,1,x))$$

ここで $g_1(x) = Ack(x,x)$ であるから、 $A_T(1,1,x+1) = g_1(A_T(1,1,x))$ と置くことができる。

$\alpha \geq \beta + 4$ のとき  $g_1(\alpha) = hyper(2,\alpha,\alpha+3) - 3 > hyper(3,\beta+2,3) = G(\beta)$ となるであろう [→自由研究] から、

$$A_T(1,1,1) = 61 > 4$$

$$A_T(1,1,2) = g_1(61) > G(4)$$

$$A_T(1,1,3) = g_1^2(61) > G^2(4)$$

などとなって、グラハム数 $G^{64}(4)$ に対しては、以下の不等式が成立する。

$$A_T(1,1,65) = g_1^{64}(61) > G^{64}(4)$$

ここまでの議論から、以下に示す通りの大小関係があることが分かる。

$$G = G^{64}(4) < g_1^{64}(61) = A_T(1,1,65) \ll A_T(1,1,A_T(1,1,61)) = g_2(2) \ll g_2(61) = g_2(g_1(3)) \ll \ll F_1$$

よって、 $G \ll g_2(2)$ である。このことはまた、 $G \ll A_T(1,2,2)$ であることをも意味する。

ふいつしゅ数 ver.1 は、先に示した $S$ 変換の適用回数 $\tau$ を用いて、 $F_1 = g_\tau(g_{\tau-1}(g_{\tau-2}(\dots g_2(g_1(3))\dots)))$ と表現することができるから、むしろグラハム数程度では比較にもならない。

【定義】 コンウェイの矢印チェーン表記 Conway's chained arrow notation

\*チェーン  $v_1 \rightarrow \dots \rightarrow v_k$  を  $C$  とし、 $n-1$  を  $n^-$  とし略記する。

$$x \rightarrow y \rightarrow z = x \uparrow^z y$$

$$C \rightarrow 1 = C$$

$$C \rightarrow 1 \rightarrow x = C$$

$$\begin{aligned} C \rightarrow x \rightarrow y &= C \rightarrow (C \rightarrow x^- \rightarrow y) \rightarrow y^- \\ &= C \rightarrow (C \rightarrow (\dots (C \rightarrow (C \rightarrow y^-) \dots) \rightarrow y^-) \rightarrow y^- \quad (x \text{ は右辺に現れる } C \text{ の数}) \end{aligned}$$

【練習問題】

問 1  $5 \rightarrow 4 \rightarrow (5 \uparrow^{100} 4)$  と  $5 \rightarrow 4 \rightarrow 3 \rightarrow 2$  との大小を述べよ。

問 2  $2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2$  と  $3 \rightarrow 2 \rightarrow 2 \rightarrow 4 \rightarrow 3$  との大小を述べよ。

問 3  $a \rightarrow b \rightarrow c \rightarrow 2$  がどんな数になるか考えよ。

問 4 問 3 を踏まえ、 $3 \rightarrow 3 \rightarrow 3 \rightarrow 3$  がどんな数になるか考えよ。

チェーン表記とグラハム数

$G(x) = 3 \uparrow^x 3 = 3 \rightarrow 3 \rightarrow x$  であるから、

$$3 \rightarrow 3 \rightarrow 1 \rightarrow 2 = 3 \rightarrow 3 \rightarrow 1 = G(1) = 27$$

$$3 \rightarrow 3 \rightarrow 2 \rightarrow 2 = 3 \rightarrow 3 \rightarrow (3 \rightarrow 3 \rightarrow 1) = G^2(1) = G(27)$$

$$3 \rightarrow 3 \rightarrow 3 \rightarrow 2 = 3 \rightarrow 3 \rightarrow (3 \rightarrow 3 \rightarrow 2 \rightarrow 2) = G^3(1) = G^2(27)$$

などとなって、けっきょく

$$3 \rightarrow 3 \rightarrow n \rightarrow 2 = G^n(1) = G^{n-1}(27)$$

と書くことができる。このことから、グラハム数  $G^{64}(4)$  に対して以下の不等式が成立する。

$$3 \rightarrow 3 \rightarrow 64 \rightarrow 2 = G^{64}(1) < G^{64}(4) < G^{64}(27) = 3 \rightarrow 3 \rightarrow 65 \rightarrow 2$$

チェーン表記は 2 重帰納変換を含む

矢印チェーン表記の函数表式  $C$  を、 $C(x_1, \dots, x_k) = x_1 \rightarrow \dots \rightarrow x_k$  とし定義する。

チェーン函数の漸化式は以下の通り。但し  $\vec{v} = (v_1, \dots, v_k)$  である。

$$C(x, y, z) = x \uparrow^z y$$

$$C(\vec{v}, 1) = C(\vec{v}, 1, x) = C(\vec{v})$$

$$C(\vec{v}, x, y) = C(\vec{v}, C(\vec{v}, x-1, y), y-1)$$

$C(x, y, z) = x \uparrow^z y = x \uparrow^{z-1} (x \uparrow^z (y-1)) = C(x, C(x, y-1, z), z-1)$  であるから、変数が 3 箇の場合でも第 3 式を適用できることが分かる (すなわち、第 1 式を  $C(x, y, 1) = x^y$  に置き換えたとしても well-defined である)。

ここで  $f(x) = C(\vec{v}, x, y)$ ,  $g(x) = C(\vec{v}, x, y+1)$  と置くと、アッカーマン函数の場合 [→kms002; p.5] と同様の議論をすることができて、変数  $y$  が原始帰納変換の回数を数えあげていることが分かる。すなわち、チェーン函数は 2 重帰納変換を含んでいる。

このことは、しかしチェーン表記が 2 重帰納函数であるということだけをただちに意味するわけではない。もしかすると、どこかで 2 重帰納変換の回数が数えあげられているかも知れないからである。……大雑把な議論をするなら、原始帰納変換の回数を数えあげている変数  $y$  とは、チェーンの終端の変数のことである。計算の進行にともない、第 2 式によって、チェーンは末尾から削られてゆき、そのたびに新たな終端が現れる。つまり、そのつど原始帰納変換の回数が数えあげられる。このことは、大雑把には、チェーンの長さが 2 重帰納変換の回数に相当することを意味する。

以下で、もう少し厳密な議論を試みる。

チェーン表記は 2 重帰納関数である

チェーン表記においては、チェーン長が 2 重帰納変換の回数に相当するのではないかとこの予想をわれわれは立てた。この予想が正しい場合、チェーン長を変数化（数えあげ）すれば、3 重帰納関数を構成できる筈である。そこで、関数  $C[a]$  を以下の通り定義する。

$$C[a](x, y, z) = (a+1) \rightarrow \cdots \rightarrow (a+1) \rightarrow (z+1) \rightarrow (y+1) \quad (x+1 \text{ は右辺に現れる } a+1 \text{ の数})$$

変数  $x$  はチェーンの長さを変数化している。はたして関数  $C[a]$  は 3 重帰納関数であろうか？

定義より、関数  $C[a]$  に以下で示す漸化式を与えることができる。

$$C[a](0, y, z) = (a+1) \uparrow^{y+1} (z+1)$$

$$C[a](x+1, 0, z) = C[a](x, z, a)$$

$$C[a](x+1, y, 0) = C[a](x, 0, a)$$

$$C[a](x, y+1, z+1) = C[a](x, y, C[a](x, y+1, z) - 1)$$

ここで、 $f(y, z) = C[a](x, y, z)$ ,  $g(y, z) = C[a](x+1, y, z)$  と置くと、

$$g(0, z) = f(z, a)$$

$$g(y, 0) = f(0, a)$$

$$g(y+1, z+1) = g(y, g(y+1, z) - 1)$$

となる。もしも、 $g$  が  $f$  から 2 重帰納的であったとすると、変数  $x$  は 2 重帰納変換の回数を数えあげていることになり、関数  $C[a]$  が 3 重帰納関数であることが示される。では、 $g$  は  $f$  から 2 重帰納的であろうか？

ここで、再び  $h(z) = g(y, z)$ ,  $i(z) = g(y+1, z)$  と置くと、

$$i(z) = g(y+1, z) = g(y, g(y+1, z-1) - 1) = h(i(z-1) - 1)$$

ゆえに、

$$i(z) = h(h(h(\cdots h(i(0) - 1) - 1 \cdots) - 1) - 1) \quad (z \text{ は右辺に現れる } h \text{ の数})$$

となつて、 $i$  は  $h$  に対する合成変換の回数を数えあげている。したがって、 $i$  は  $h$  から原始帰納的である。

すなわち、 $g$  は  $f$  に対する原始帰納変換の回数を数えあげている。したがって、 $g$  は  $f$  から 2 重帰納的である。

以上から、変数  $x$  は 2 重帰納変換の回数を数えあげており、関数  $C[a]$  が 3 重帰納関数であることが示された。

変数  $x$  とはチェーンの長さを変数化したものだった。  $C[a](x+1, y, z)$  が  $C[a](x, y, z)$  から 2 重帰納的であるということは、チェーン長が 2 重帰納変換の回数に相当するということを意味する。

したがって、チェーン関数  $C$  は約チェーン長回の 2 重帰納変換から構成される巨大関数である。

※「約」というのは、厳密には  $C[a](x, y, z)$  におけるチェーン長は  $x+3$  であつて、 $x$  ではないからである。

## 巨大数生成の構造分析 2

ふいつしゅ函数 ver.1 ( $F_1$ ) と 3 変数アッカーマン函数 ( $A_\tau$ ) とを比較してみると、関数  $F_1$  は 2 重帰納関数であつて関数  $A_\tau$  は 3 重帰納関数であつてみれば、 $F_1 \leq^* A_\tau$  が成立する。もっとも、 $g_n(x) = A_\tau(n-1, x, x)$  であることは先に確認した通りであり、 $n < \tau - 1$  の範囲では  $A_\tau(n, x, x) < F_1(x)$  となるだろう。

また、 $n$  変数チェーン関数を  $C_n$  とし、関数  $F_1$  と関数  $C_n$  とを比較してみると、 $F_1$  は  $\tau$  回という、数え切れないほどの 2 重帰納変換から構成されており、約  $n$  回の 2 重帰納変換から構成されている  $C_n$  に対しては、 $n$  が  $\tau$  クラスの巨大数を取らないかぎり、 $C_n \leq^* F_1$  となる（もちろん、 $F_1 \leq^* C_n$  となるような  $n$  もありうるのだが）。

こうしてながめてくると、ふいつしゅ数 ver.1 やふいつしゅ函数 ver.1 が、いかに巨大な（函）数であるかが分かるだろう。2 重帰納関数から生成される巨大数で、ふいつしゅ数 ver.1 を超えるものを構成することは難しいだろうと思われる。それというのも、ふいつしゅ数 ver.1 は、実質的には 3 重帰納関数から生成されたものと考えられるからである。  $A_\tau$  が 3 重帰納関数であったことを思いおこそう。  $g_n(x) = A_\tau(n-1, x, x)$  と書けるから、ふいつしゅ数 ver.1 は構成の過程で 3 重帰納関数  $A_\tau$  を利用しているといえる。たまさか、関数  $F_1$  は 2 重帰納変換の回数を変数化してはいないものの、巨大数のクラスとしては 2 重帰納法のレベルをはるかに超えている。

【定義】 ふいっしゅ数バージョン2 Fish number ver.2 ( $F_2$ )

$$1^\circ \quad S : [m, f(x)] \rightarrow [g(m), g(x)] \quad m, g(m) \in N$$

$$B(0, n) = f(n)$$

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

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

$$g(x) = B(x, x)$$

$$2^\circ \quad SS : [m, f(x), S] \rightarrow [n, g(x), S'] \quad m, n \in N$$

$$S' = S^{f(m)}$$

$$S' : [m, f(x)] \rightarrow [n, p(x)]$$

$$(S')^x : [m, f(x)] \rightarrow [q(x), g(x)]$$

$$3^\circ \quad SS^{63} : [3, x+1, S] \rightarrow [F_2, F_2(x), S']$$

※  $S$  変換の定義や、 $SS$  変換の  $n$  の定義などは  $F_1$  と同様。  $SS$  変換における  $g(x)$  は、  $[m, f(x)]$  に  $S^{f(m)}$  変換を  $x$  回適用したもの。  $q(x)$  は、その性質上  $x$  を変数とする函数として表示されているだけで、特に意味はない。

ふいっしゅ函数 ver.2 は 3 重帰納函数である

$S^k : [m, f(x)] \rightarrow [q(k), g_k(x)]$  として、各  $k, x$  の値に対する  $g_k(x)$  の値を table 形式で表示する。

$$g_k(x) = \left\{ \begin{array}{c|cccc} & 1 & 2 & \cdots & x & \cdots \\ \hline 1 & g_1(1) & g_1(2) & \cdots & g_1(x) & \cdots \\ 2 & g_2(1) & g_2(2) & \cdots & g_2(x) & \cdots \\ \vdots & \vdots & \vdots & & \vdots & \\ k & g_k(1) & g_k(2) & \cdots & g_k(x) & \cdots \\ \vdots & \vdots & \vdots & & \vdots & \end{array} \right\}$$

ここで  $g(x) = g_x(x)$  と置くと、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し、 $S$  変換を対角化している。 $S$  変換は 2 重帰納変換に相当するから、 $g$  は 2 重帰納変換を対角化しており、 $f$  に対して 3 重帰納変換を 1 回施したものに相当する。このとき、いかなる定数  $k$  にかんしても  $g(x) \geq^* g_k(x)$  となることは自明。 $k = \tau$  であってもそうだから、もちろん  $g(x) \geq^* F_1(x) = g_\tau(x)$  である。

ふいっしゅ数 ver.2 における  $SS$  変換は、 $S$  変換でなく  $S'$  変換に対して上記の対角化を行なっているが、 $S' = S^{f(m)}$  として定義されており、合成の回数は無視してかまわないから、本質的には  $S$  変換を対角化したものと考えることができる。すなわち、ふいっしゅ数 ver.2 における 1 回の  $SS$  変換は、1 回の 3 重帰納変換に相当する。

したがって、函数  $F_2$  は 63 回の 3 重帰納変換 から構成される巨大函数であり、その  $x$  に巨大定数を代入することでふいっしゅ数 ver.2 が生成される。

【自由研究】

$1 \leq x, 3 \leq y$  のとき  $hyper(3, x, y) < hyper(2, x+2, y)$  となることの証明法を考えてみること。