2019 年度 数理情報学基礎論概論 2・講義ノート [*†]

木原 貴行
名古屋大学 情報学部・情報学研究科

最終更新日: 2019 年 8 月 1 日

目次

* 本講義ノートは，2019 年度春 2 期開講の名古屋大学大学院情報学研究科における講義「数理情報学基礎論概論 2」の内容をまとめたものである．
† 講義のページ： http://www.math.mi.i.nagoya-u.ac.jp/~kihara/teach.html

1　原始再帰関数

1.1　原始再帰法とはなにか

自然数の足し算および掛け算のような基本的な演算から，**原始再帰法** (*primitive recursion*)[*1]と呼ばれる操作によって，様々な自然数上の関数を構成できることを本節では見ていこう．

このアイデアを説明するために，人類が「新しい演算」を創造していく過程をシミュレートしよう[*2]．まず，自然数 x が与えられたとき，「x の次」である数が何であるかを知っているとしよう．すると，「x の次の次」や「x の次の次の次」などを考えることができる．しかし，いくつも「の次」という文字を書くのは億劫なので，x の y 個次の数を $x+y$ と書くことにしよう．こうして，人類は，「足す」という演算を生み出した．

$$x+y = x \text{ の } \underbrace{\text{次の次の次} \ldots \text{ の次の次}}_{y \text{ 個}}$$

このように「足す」という演算を知った人類は，$x+x+x$ や $x+x+x+x+x$ のように x を何度も足す，という演算が有用であることに次第に気づき始める．これを簡潔に表すために，人類は「掛ける」という演算を次のように定義した．

$$x \cdot y = \underbrace{x+x+\cdots+x+x}_{y \text{ 個}}$$

そして，「掛ける」という演算を知った人類は，$x \cdot x \cdot x$ のように x を何度も掛ける，という演算の有用性に気づく．そして「累乗」という演算を次のように定義した．

$$x^y = \underbrace{x \cdot x \cdot \cdots \cdot x \cdot x}_{y \text{ 個}}$$

すると，自然に x^{x^x} や $x^{x^{x^x}}$ のようなものを考える人も現れる．上方向にたくさん添字が付くのは見づらいので，x^y のことを今後は $x \uparrow y$ と書くこととしよう．たとえば，x^{x^x} は $x \uparrow (x \uparrow x)$ であり，$x^{x^{x^x}}$ は $x \uparrow (x \uparrow (x \uparrow x))$ である．また，以下，括弧は省略し，これらの演算は右から順に適用するものとする．さて，累乗でも飽き足らない一部の人類は，「テトレーション」という演算を編み出した．

$$x \uparrow\uparrow y = \underbrace{x \uparrow x \uparrow \ldots \uparrow x \uparrow x}_{y \text{ 個}}$$

[*1] Primitive recursion は伝統的には，原始再帰でなく**原始帰納**と訳されることがある．しかし，再帰理論やその周辺の理論では，inductive と recursive が全く別概念として登場し，たとえば，「recursive ではないが inductive であるような集合が存在する」「Π^1_1-transfinite induction は Π^1_1-transfinite recursion を導かない」というような定理が成立する．したがって，inductive と recursive には異なる訳語を割り当てる必要があるが，ここでは inductive を**帰納**と訳し，recursive を**再帰**と訳す流儀を採用する．

[*2] 本稿の記述は現実の数学史に沿っているとは限らない．

飽くなき人類は，更なる演算「ペンテーション」を定義する．

$$x \uparrow\uparrow\uparrow y = \underbrace{x \uparrow\uparrow x \uparrow\uparrow \ldots \uparrow\uparrow x \uparrow\uparrow x}_{y \text{ 個}}$$

より一般に，クヌースの矢印記法というものは以下によって定義される．

$$x \uparrow^{n+1} y = \underbrace{x \uparrow^{n} x \uparrow^{n} \ldots \uparrow^{n} x \uparrow^{n} x}_{y \text{ 個}}$$

このような再帰的な関数構成を数学的に抽象化したものが，原始再帰法と呼ばれる概念である．

さて，ここまで，何かの演算 $x \diamond y$ を元に，人類が新たな演算 $x \star y$ を創造する過程を見てきた．これらの過程が共有するものとは何であろうか．それは以下の性質である．

$$x \star y = \underbrace{x \diamond x \diamond \cdots \diamond x \diamond x}_{y \text{ 個}}$$

実際に，この値 $x \star y$ を計算する場合には，$x \star 2 = x \diamond x$ を求め，$x \star 3 = x \diamond x \diamond x = x \diamond (x \star 2)$ を求め，$x \star 4 = x \diamond x \diamond x \diamond x = x \diamond (x \star 3)$ を求め，... という手続きを行うこととなるだろう．たとえば，掛け算以降の演算の定義を少し書き直せば，次のようにして定義されていることがわかる．

$$\begin{cases} x \star 1 = x, \\ x \star (y+1) = x \diamond (x \star y). \end{cases}$$

上の定義では曖昧であるが，$x \star 0$ の場合も定義しておくのが自然である．たとえば，$x \cdot 0 = 0$ であるし，$x \uparrow 0 = x^0 = 1$ である．

演習問題 1.1. 以下のようにして，$x \uparrow^{n+1} y$ を定義する．

$$\begin{cases} x \uparrow^{n+1} 0 = 1, \\ x \uparrow^{n+1} (y+1) = x \uparrow^{n} (x \uparrow^{n+1} y). \end{cases}$$

このとき，$x \uparrow^{n+1} 1 = x$ であることを示せ．

さて，ここまでは演算の形式で書いてきたが，これらを関数として見直すこととする．つまり，今までのことを，関数 $h(x,y) = x \diamond y$ から新たな関数 $f(x,y) = x \star y$ を創造する過程を考えてきたものとしよう．また，$g(x) = x \star 0$ は与えられているものとする．これまでの内容を言い直せば，g と h からの新たな関数 f は以下のように生み出された．

$$\begin{cases} f(x,0) = g(x), \\ f(x,y+1) = h(x, f(x,y)). \end{cases}$$

それでは，原始再帰法の厳密な定義に入ろう．上では，f は 2 変数関数であったが，より多変数であってよい．

定義 1.2 (原始再帰法)**.** 関数 $g : \mathbb{N}^n \to \mathbb{N}$ と $h : \mathbb{N}^{n+2} \to \mathbb{N}$ が与えられているとする．さらに，$f : \mathbb{N}^{n+1} \to \mathbb{N}$ が，次のように定義されるとしよう: 任意の $\bar{x} \in \mathbb{N}^n$ と $y \in \mathbb{N}$ について，

$$\begin{cases} f(\bar{x}, 0) = g(\bar{x}), \\ f(\bar{x}, y+1) = h(\bar{x}, y, f(\bar{x}, y)). \end{cases}$$

このとき，f は g と h から**原始再帰法** (*primitive recursion*) によって定義されるという．

本節の導入において，「の次」を初期関数として，原始再帰法を有限回だけ用いて，数多くの関数を構成してきた．このような関数は，原始再帰関数と呼ばれる関数の一種である．より正確には，原始再帰関数の概念を以下のように導入しよう．

定義 1.3 (原始再帰関数)**.** 以下のように，**原始再帰関数** (*primitive recursive function*) を帰納的に定義する．

1. 以下によって定義される後続関数 $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$, 零関数 $\mathsf{zero}^n : \mathbb{N}^n \to \mathbb{N}$ および射影関数 $\mathsf{proj}_i^n : \mathbb{N}^n \to \mathbb{N}$ は原始再帰関数である．

$$\mathsf{succ}(x) = x + 1, \quad \mathsf{zero}^n(x_1, \ldots, x_n) = 0, \quad \mathsf{proj}_i^n(x_1, \ldots, x_n) = x_i.$$

2. 原始再帰関数たちの合成は原始再帰関数である．つまり，$h : \mathbb{N}^m \to \mathbb{N}$ と $g_1, \ldots, g_m : \mathbb{N}^n \to \mathbb{N}$ が原始再帰的ならば，以下のように定義される関数 $f : \mathbb{N}^n \to \mathbb{N}$ もまた原始再帰的である．
$$f(\bar{x}) = h(g_1(\bar{x}), \ldots, g_m(\bar{x})).$$

3. 原始再帰関数 g, h から原始再帰法によって定義される関数は原始再帰的である．

以後，後続関数，零関数，射影関数のことを**初期関数** (*initial function*) と呼ぶ．つまり，原始再帰関数全体の族とは，初期関数を含み合成と原始再帰法で閉じた最小の関数族として与えられる．

例 1.4. 和 $(x, y) \mapsto x + y$, 積 $(x, y) \mapsto x \cdot y$, 冪乗 $(x, y) \mapsto x^y$, 第 n 矢印演算 $(x, y) \mapsto x \uparrow^n y$ はいずれも原始再帰的関数である．

例 1.5. ここでは自然数上の関数を考えるので，引き算は一般には定義されないが，部分的引き算 $x \dot{-} y = \max\{0, x - y\}$ を考えることはできる．これが原始再帰的であることを示そう．まず，「入力が正の数であれば，値を 1 減らす」という演算 pred が原始再帰的であることを確認する．これは，$g = \mathsf{zero}^0$ と $h = \mathsf{proj}_1^2$ から原始再帰法によって定義される関数を考えればよい．

$$\begin{cases} \mathsf{pred}(0) = \mathsf{zero}^0 = 0, \\ \mathsf{pred}(y+1) = \mathsf{proj}_1^2(y, \mathsf{pred}(y)) = y. \end{cases}$$

このとき，部分的引き算 $f(x,y) = x \dot{-} y = \max\{0, x-y\}$ は，$g = \mathsf{proj}_1^1$ と $h = \mathsf{pred} \circ \mathsf{proj}_3^3$ から原始再帰法によって定義される．

$$\begin{cases} x \dot{-} 0 = \mathsf{proj}_1^1(x) = x, \\ x \dot{-} (y+1) = \mathsf{pred}(\mathsf{proj}_3^3(x, y, x \dot{-} y) = \mathsf{pred}(x \dot{-} y). \end{cases}$$

例 1.6. 典型的な原始再帰関数として，総和と総乗がある．

$$f_0(\bar{x}, y) = \sum_{i=0}^{y-1} p(\bar{x}, i), \qquad f_1(\bar{x}, y) = \prod_{i=0}^{y-1} p(\bar{x}, i).$$

ここで，$p : \mathbb{N}^{n+1} \to \mathbb{N}$ は与えられた原始再帰関数とする．このとき，f_0 は $g_0 = \mathsf{zero}^n$ と $h_0(\bar{x}, y, z) = z + p(\bar{x}, y)$ から原始再帰法によって定義される．

$$\begin{cases} f_0(\bar{x}, 0) = \mathsf{zero}^n(\bar{x}) = 0, \\ f_0(\bar{x}, y+1) = f_0(\bar{x}, y) + p(\bar{x}, y). \end{cases}$$

同様に，f_1 は $g_1 = \mathsf{succ} \circ \mathsf{zero}^n$ と $h_1(\bar{x}, y, z) = z \cdot p(\bar{x}, y)$ から原始再帰法によって定義される．

$$\begin{cases} f_1(\bar{x}, 0) = \mathsf{succ}(\mathsf{zero}^n(\bar{x})) = 1, \\ f_1(\bar{x}, y+1) = f_0(\bar{x}, y) \cdot p(\bar{x}, y). \end{cases}$$

さて，新しい原始再帰関数を作るために，場合分けを自由に使用してよいことを示しておくと便利である．

命題 1.7 (原始再帰的場合分け)**.** $g, h, p : \mathbb{N}^n \to \mathbb{N}$ を原始再帰関数とする．このとき，次によって定義される $f : \mathbb{N}^n \to \mathbb{N}$ は原始再帰的である．

$$f(\bar{x}) = \begin{cases} g(\bar{x}) & \text{if } p(\bar{x}) = 0, \\ h(\bar{x}) & \text{if } p(\bar{x}) > 0. \end{cases}$$

Proof. まず，$\mathsf{sgn} : \mathbb{N} \to \mathbb{N}$ を zero^0 と $\mathsf{succ} \circ \mathsf{zero}^2$ から原始再帰法よって定義される関数とする．

$$\begin{cases} \mathsf{sgn}(0) = \mathsf{zero}^0 = 0, \\ \mathsf{sgn}(y+1) = \mathsf{succ}(\mathsf{zero}^2(y, \mathsf{sgn}(y))) = 1. \end{cases}$$

つまり，sgn は入力値が 0 であれば 0 を出力し，入力値が 0 以外であれば 1 を出力する．このとき，f は次のような合成によって定義する．

$$f(\bar{x}) = g(\bar{x}) \cdot (1 \dot{-} \mathsf{sgn}(p(\bar{x}))) + h(\bar{x}) \cdot \mathsf{sgn}(p(\bar{x})).$$

これが求める関数であることを確認しよう．もし $p(\bar{x}) = 0$ であれば，$\mathsf{sgn}(p(\bar{x})) = 0$ かつ $1 \dot{-} \mathsf{sgn}(p(\bar{x})) = 1$ であるから，

$$f(\bar{x}) = g(\bar{x}) \cdot 1 + h(\bar{x}) \cdot 0 = g(\bar{x})$$

である．もし $p(\bar{x}) > 0$ であれば，$\mathsf{sgn}(p(\bar{x})) = 1$ かつ $1 \dot{-} \mathsf{sgn}(p(\bar{x})) = 0$ であるから，

$$f(\bar{x}) = g(\bar{x}) \cdot 0 + h(\bar{x}) \cdot 1 = h(\bar{x})$$

である．よって f が求める関数であることが分かった．また，f は原始再帰関数 $+$, $\cdot$, $\dot{-}$, sgn, g, h, p から合成によって作られているので，原始再帰的である． □

これによって，原始再帰法による関数構成をある種のプログラミングであると見ることができる．つまり，原始再帰法は，いわゆる「for ループ（つまり，繰り返し回数が事前に分かるループ命令）」を表すものであり，命題 1.7 によって，場合分けも利用可能である．これによって，かなり多くの関数が，原始再帰関数構成としてプログラミング可能である．一方で，このような「原始再帰的プログラミング」によっては構成できない関数もまた存在する．たとえば，原始再帰関数では「while ループ（つまり，繰り返し回数が事前に分からないループ命令）」を実装できない．

第 1.2 節では，原始再帰関数のより細かい分類を与える．第 2 節以降では，原始再帰法の一般化を導入し，拡張された意味で原始再帰的であるが本節の意味では原始再帰的でない関数について考察する．

■ **Historical Remark.** 本節の導入のような関数構成を，定義 1.2 のような原始再帰法として抽象化した歴史上最初の人物が誰であるかは不明瞭であるが，原始再帰法概念の具体的な使用例としては，1861 年のヘルマン・グラスマン (Hermann Grassmann; 1809–1877) によるものがあるらしい．ただし，原始再帰法の理論的研究という面では，1888 年のリヒャルト・デデキント (Richard Dedekind; 1831–1916) の研究が始祖であり，このためデデキントを原始再帰法の祖として挙げることが多いようである．

1.2 初等関数とグジェゴルチク階層

原始再帰関数の中にも，構成が簡単なものから難しいものまで様々な関数がある．この原始再帰の内部構造を理解するために，原始再帰関数の階層構造を考察しよう．その基点となる関数のクラスとして，初等関数という概念を導入する．初等関数の族とは，初期関数と加法，部分的減法を含み，合成と総和，総乗で閉じている最小の関数族である．より正確には，以下のように定義される．

定義 1.8. 以下のように，**初等関数** (*elementary function*) を帰納的に定義する．

1. 初期関数，加法 $x + y$, 部分的減法 $x \dot{-} y$ は初等関数である．
2. 初等関数たちの合成は初等関数である．
3. 初等関数から総和, 総乗によって定義される関数は初等関数である．つまり, $p : \mathbb{N}^{n+1} \to \mathbb{N}$

が初等関数ならば，以下のように定義される関数 $f, g : \mathbb{N}^{n+1} \to \mathbb{N}$ もまた初等関数である．

$$f(\bar{x}, y) = \sum_{i=0}^{y-1} p(\bar{x}, i) \qquad g(\bar{x}, y) = \prod_{i=0}^{y-1} p(\bar{x}, i).$$

例 1.9. 任意の $n \in \mathbb{N}$ について，関数 $f(x) = x \uparrow\uparrow n$ は初等関数である．なぜなら，総乗を用いれば，指数関数 x^y が初等関数であることは明らかで，$x \uparrow\uparrow n$ は指数関数の n 回合成として表されるためである．

また，例 1.6 より，任意の初等関数は原始再帰的である．一方，$f(x) = x \uparrow\uparrow x$ は初等関数ではない．よって，初等関数でないような原始再帰的関数が存在する:

$$\text{初等関数全体の族} \subsetneq \text{原始再帰的関数全体の族}.$$

したがって，初等関数の構成は，ある制限された原始再帰と考えることができる．これをより明快に理解するために，以下の概念を導入しよう．

定義 1.10 (有界原始再帰法)**.** 関数 $g : \mathbb{N}^n \to \mathbb{N}$, $h : \mathbb{N}^{n+2} \to \mathbb{N}$ および $t : \mathbb{N}^{n+1} \to \mathbb{N}$ が与えられているとする．さらに，$f : \mathbb{N}^{n+1} \to \mathbb{N}$ が，次のように定義されるとしよう: 任意の $\bar{x} \in \mathbb{N}^n$ と $y \in \mathbb{N}$ について，

$$\begin{cases} f(\bar{x}, 0) = g(\bar{x}), \\ f(\bar{x}, y+1) = h(\bar{x}, y, f(\bar{x}, y)), \\ f(\bar{x}, y) \leqslant t(\bar{x}, y) \end{cases}$$

このとき，f は g, h, t から**有界原始再帰法** (*bounded primitive recursion*) によって定義されるという．

初等関数は，有界原始再帰法によって特徴づけることができる．このために，まず，素数判定などは初等関数によって行うことができ，したがって列のコーディングなども初等関数によって行うことができることに注意する．以下の定理の証明の前半の議論は，原始再帰によるコーディングの議論に慣れていないと難しいので，飛ばしてしまっても構わない．

定理 1.11. 初等関数全体の族は，以下を満たす最小の関数族と正確に一致する．

- 初期関数と指数関数 x^y を含む．
- 合成と有界原始再帰法で閉じている．

Proof. 初等関数全体の族が有界原始再帰法で閉じていることを示す．f が初等関数 g, h, t から有

界原始再帰法で定義されていると仮定する．まず

$$m = \langle f(\bar{x},0),\ldots,f(\bar{x},y)\rangle \iff lh(m) = y+1 \ \wedge\ (m)_0 = g(\bar{x}) \\ \wedge\ (\forall i < y)\ (m)_{i+1} = h(\bar{x},i,(m)_i) \tag{1}$$

であり，上で述べたように，(1) の右式の真偽判定は初等関数によって行うことができる．また，$f(\bar{x},y) \leqslant t(\bar{x},y)$ であるから，$t^+(\bar{x},y) = \sum_{i\leqslant y} t(\bar{x},i)$ とすると $f(\bar{x},i) \leqslant t^+(\bar{x},y)$ である．よって，$q(\bar{x},y) = (p_y^{t^+(\bar{x},y)})^{y+1}$ とすれば $\langle f(\bar{x},0),\ldots,f(\bar{x},y)\rangle \leqslant q(\bar{x},y)$ であることが分かる．いま，q は明らかに初等関数である．

以上より，$f(\bar{x},y)$ は (1) の条件を満たす $m \leqslant q(\bar{x},y)$ について $(m)_y$ として得られる．より正確には，

$$h(\bar{x},y,m) = 1 \iff h(\bar{x},y,m) \neq 0 \iff m \text{ は (1) の条件を満たす}$$

と h を定義すると，h は初等関数であり，

$$f(\bar{x},y) = \sum_{m\leqslant q(\bar{x},y)} h(\bar{x},m)\cdot(m)_y$$

が成立する．よって，f は初等関数である．

逆方向については，総和と総乗が有界原始再帰法によって定義できることを示せばよい．例 1.6 において総和と総乗が原始再帰法によって定義できることを示したので，これらの有界性をみればよい．これについては，

$$\sum_{z\leqslant y} p(\bar{x},z) \leqslant (y+1)\cdot\max_{z\leqslant y} p(\bar{x},z),$$
$$\prod_{z\leqslant y} p(\bar{x},z) \leqslant (\max_{z\leqslant y} p(\bar{x},z))^{y+1}$$

であるから，$q(\bar{x},y) = \max_{z\leqslant y} p(\bar{x},z)$ が有界原始再帰法によって構成できることを示せばよい．命題 1.7 で原始再帰的に構成した場合分け関数の特殊なケースである次の関数

$$\texttt{[if } z \texttt{ is zero then } x \texttt{ else } y\texttt{]} = \begin{cases} x & \text{if } z = 0, \\ y & \text{if } z \neq 0. \end{cases}$$

は $x+y$ で有界であるから，有界原始再帰法によって定義できる．$\{p(\bar{z})\}_{z\leqslant y}$ の中から最大値を達成する z を探索する関数 p^* を次の原始再帰法によって定義する．

$$p^*(\bar{x},0) = 0,$$
$$p^*(\bar{x},z+1) = \begin{cases} p^*(\bar{x},z) & \text{if } p(\bar{x},z+1) \leqslant p(\bar{x},p^*(\bar{x},z)) \\ z+1 & \text{otherwise.} \end{cases}$$
$$= \texttt{[if } p(\bar{x},z+1) \dot{-} p(\bar{x},p^*(\bar{x},z)) \texttt{ is zero then } p^*(\bar{x},z) \texttt{ else } z+1\texttt{]}.$$

明らかに $p^*(\bar{x},z) \leqslant z$ であるから，有界原始再帰法によって p^* を構成できる．このとき，明らかに $\max_{z\leqslant y} p(\bar{x},z) = p(\bar{x},p^*(\bar{x},y))$ であるから，定理は示された． □

有界原始再帰で初等関数を構成できることを示した．原始再帰関数にどのようなものがあるかを理解するために，原始再帰関数を構成に必要な原始再帰法の利用回数によって分類し，その各レベルがどのような関数を含むかを分析しよう．具体的には，次の関数族を考える．

$$\mathcal{PR}_n = \text{初等関数を始点とし，高々 } n \text{ 回の原始再帰法の適用で構成できる関数族.}$$

より正確には，以下のように定義する．

定義 1.12. 各 $n \in \mathbb{N}$ について，原始再帰関数の族 $\mathcal{PR}_n$ を以下のように帰納的に定義する．

1. $\mathcal{PR}_0$ を初等関数全体の族とする．
2. $\mathcal{PR}_{n+1}$ を $\mathcal{PR}_n$ から任意回の合成と高々 1 回の原始再帰法の適用で構成できる関数全体の族とする．

例 1.13. 演習問題 1.1 で定義を与えたクヌースの矢印表記 $\uparrow^n$ について，定義より指数関数 $\uparrow^1$ は初等関数である．よって，$\uparrow^{n+1}$ は $\mathcal{PR}_n$ に属す．一方，$\uparrow^{n+2}$ は $\mathcal{PR}_n$ に属さない (補題 1.15)．

以後，$\mathcal{PR}_n$ に属す関数を，**階数 n の原始再帰関数**と呼ぶこととする．$\mathcal{PR}_n$ の性質を調べるために，次のような初等関数の相対化を考えよう．

定義 1.14. 関数 $g : \mathbb{N} \to \mathbb{N}$ が与えられているとする．このとき，以下のように，**g-初等関数** (*g-elementary function*) を帰納的に定義する．

1. 初期関数，加法 $x + y$, 部分的減法 $x \dot{-} y$, および g は g-初等関数である．
2. g-初等関数たちの合成は g-初等関数である．
3. g-初等関数から総和，総乗によって定義される関数は g-初等関数である．

g-初等関数全体の族を $\mathcal{E}(g)$ と書こう．ここで主に取り扱うものは $\mathcal{E}(\uparrow^n)$ である．この階層は，**グジェゴルチク階層** *(Grzegorczyk hierarchy)* と呼ばれる．まず，グジェゴルチク階層が巨大関数の階層を与えることを示そう．以下，$\uparrow^n(x)$ によって $x \uparrow^n x$ を表す．関数 $f, g : \mathbb{N} \to \mathbb{N}$ について，g が f を**支配** (*dominate*) するとは，次が成立することである．

$$(\exists d \in \mathbb{N})(\forall x \geqslant d)\ f(x) \leqslant g(x).$$

補題 1.15. $n \geqslant 1$ とする．任意の関数 $f \in \mathcal{E}(\uparrow^n)$ について，$\uparrow^n$ のある定数 c 回合成 $(\uparrow^n)^{(c)}$ が f を支配する．

Proof. 証明はさほど難しくないので, 興味があったら, たとえば Odifreddi [3, Theorem VIII.8.10] を参考にしてほしい． □

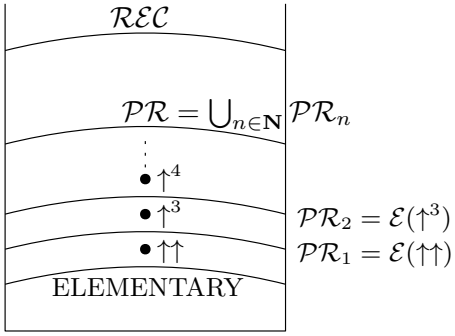

図 1　原始再帰関数のグジェゴルチク階層

定理 1.16. 任意の $n \in \mathbb{N}$ について，$\mathcal{PR}_n = \mathcal{E}(\uparrow^{n+1})$ が成立する．つまり，

階数 n の原始再帰関数 $= (\uparrow^{n+1})$-初等関数.

Proof. まず，$\uparrow^{n+1} \in \mathcal{PR}_n$ であるから，$\mathcal{E}(\uparrow^{n+1}) \subseteq \mathcal{PR}_n$ であることは明らかである．逆向きの包含関係を帰納法により証明する．$\mathcal{PR}_n = \mathcal{E}(\uparrow^{n+1})$ は既に示されていると仮定する．$\mathcal{PR}_{n+1} = \mathcal{E}(\uparrow^{n+2})$ を示すためには，$\mathcal{E}(\uparrow^{n+1})$ の関数から 1 回の原始再帰法の適用で定義できる関数が $\mathcal{E}(\uparrow^{n+2})$ に属すことを示せばよい．$g, h \in \mathcal{E}(\uparrow^{n+1})$ から原始再帰法によって f が構成されていると仮定する．f の構成過程をコードする関数 t を次によって定義する．

$$t(\langle \bar{x}, n, z\rangle) = \langle \bar{x}, n+1, h(\bar{x}, n, z)\rangle.$$

このとき，

$$t^{(y)}(\langle \bar{x}, 0, g(\bar{x})\rangle) = \langle \bar{x}, y, f(\bar{x}, y)\rangle.$$

が成立するから，$(x, y) \mapsto t^{(y)}(x)$ から合成を用いて f を定義できる．これより，関数 $(x, y) \mapsto t^{(y)}(x)$ が $\mathcal{E}(\uparrow^{n+2})$ に属すことを示せばよい．定理 1.11 と同様にして，$\mathcal{E}(\uparrow^{n+2})$ が有界原始再帰法で閉じていることは示せるので，ある $s \in \mathcal{E}(\uparrow^{n+2})$ が存在して，$t^{(y)}(x) \leqslant s(x, y)$ であることを示せば十分である．補題 1.15 より，ある定数 c が存在して，十分大きな x について，$t(x) \leqslant (\uparrow^{n+1})^{(c)}(x)$ が成立する．よって，十分大きな x, y について，

$$t^{(y)}(x) \leqslant (\uparrow^{n+1})^{(c+y)}(x) \leqslant \uparrow^{n+2} (x + y).$$

よって，$t^{(y)}(x)$ は $\mathcal{E}(\uparrow^{n+2})$ の関数で抑えられることが示された．以上より，$\mathcal{PR}_{n+1} = \mathcal{E}(\uparrow^{n+2})$ が結論付けられる．□

以上のようにして，原始再帰関数のクラスは，構成に必要な原始再帰法の利用回数，あるいはク

ヌースの矢印表記によって階層分けすることができる (図 1).

$$\mathcal{E} = \mathcal{PR}_0 \subsetneq \mathcal{E}(\uparrow\uparrow) = \mathcal{PR}_1 \subsetneq \cdots \subsetneq \mathcal{E}(\uparrow^{n+1}) = \mathcal{PR}_n \subsetneq \mathcal{E}(\uparrow^{n+2}) = \mathcal{PR}_{n+1} \subsetneq \cdots$$
$$\cdots \subsetneq \mathcal{PR} = \bigcup_{n\in\mathbb{N}} \mathcal{PR}_n \subsetneq \cdots \subsetneq \mathcal{REC}.$$

ここで，$\mathcal{REC}$ は計算可能関数全体のクラスを表す.

2 原始再帰的でない関数

2.1 高階汎関数を用いた原始再帰とアッカーマン関数

デデキントの原始再帰法は，与えられた $\mathbb{N}$ 上の関数たちから別の $\mathbb{N}$ 上の関数を作る手続きであった．このアイデアは，より一般の集合上の関数の構成に対しても適用できる．たとえば，以下の抽象的な原始再帰法を考えるのは自然である.

定義 2.1. X と Y を任意の集合とし，関数 $g\colon X \to Y$ と $h\colon X \times \mathbb{N} \times Y \to Y$ が与えられているとする．さらに，$f\colon X \times \mathbb{N} \to Y$ が，次のように定義されるとしよう: 任意の $x \in X$ と $y \in \mathbb{N}$ について,

$$\begin{cases} f(x,0) = g(x), \\ f(x,y+1) = h(x,y,f(x,y)). \end{cases}$$

このとき，f は g と h から**集合上の原始再帰法**によって定義されるという.

この他にも色々な形式の原始再帰法があり得る．このような原始再帰の変種を利用すれば，ふつうの原始再帰法では構成できない関数を定義できるかもしれない.

例 2.2. ダフィット・ヒルベルト (David Hilbert; 1862–1943) が 1925 年に言及したように，拡張された原始再帰法を用いれば，与えられた 2 項演算 $*$ の累積を行う汎関数 `iterate` を定義できる：

$$\texttt{iterate}(*,a,n) = \underbrace{a * a * a * \cdots * a}_{n\text{ 個}}$$

より具体的には，集合 A 上の 2 項演算 $*$ は 2 変数関数 $\nu\colon A \times A \to A$ として扱う．ここで，$*$ は右単位元 $e \in A$ を持つ，つまり，任意の $a \in A$ に対して $\nu(a,e) = a * e = a$ を満たすと仮定する．このとき，$g(\nu,a) = e$ および $h(\nu,a,y,z) = \nu(a,z)$ に対して，集合上の原始再帰法を適用すると,

$$\begin{cases} f(\nu,a,0) = e, \\ f(\nu,a,n+1) = \nu(a, f(\nu,a,n)) \end{cases}$$

となる f を得る．どのような集合 X, Y 上の原始再帰法を考えているか，念のために注意しておくと，$X = A^{A\times A} \times A$ かつ $Y = A$ である．このとき，$\texttt{iterate} = f$ が成立していることは容易に確認できる.

さて，問題となり得る点は，$h(\nu, a, y, z) = \nu(a, z)$ となる h が何らかの意味で原始再帰的に定義できるかという部分である．この h を定義するためには，たとえば $\mathsf{eval}(v, x) = v(x)$ となる写像 $\mathsf{eval}\colon Y^X \times X \to Y$ が定義できれば十分である．これについては後で議論することにするが，もし評価写像 eval さえ定義できれば，2 項演算の累積を行う関数は，集合上の原始再帰法によって定義されたこととなる．

さて，特に重要なのは，$A = \mathbb{N}$ のときである．この場合，上で作った `iterate` は，$\mathbb{N}$ 上の関数などを入力として，自然数を返す「型 2」の**高階汎関数**である．本稿では，以下のように有限型を定義することにする．

- 自然数は，型レベル 0 の対象である．
- 型 n 以下の対象たちの組は，型レベル n 以下である．
- 型 n 以下の対象を入力として，型レベル $n+1$ 以下の対象を出力とする関数は型レベル $n+1$ 以下である．

ただし，以後，用語の簡易化のために，型レベル n を単に型 n と言ってしまうことにする．

例 2.3. $\mathbb{N}$ の元は型 0 対象であり，$\mathbb{N}^{\mathbb{N}}$ の元は型 1 対象であり，$\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$ の元は型 2 対象である．同様に，$\mathbb{N} \times \mathbb{N}$ の元は型 0 対象であるから，$\mathbb{N}^{\mathbb{N}\times\mathbb{N}}$ の元は型 1 対象で，$\mathbb{N}^{\mathbb{N}\times\mathbb{N}} \times \mathbb{N}^2$ の元は型 1 対象である．よって，$\mathtt{iterate}\colon \mathbb{N}^{\mathbb{N}\times\mathbb{N}} \times \mathbb{N}^2 \to \mathbb{N}$ は型 2 対象である．

通常の数学の文脈では，X から Y への関数全体の集合を Y^X と書く．しかし，高階汎関数を考える際にこの記法を用いると，記号が縦に積み重なっていき，スペースを取ってしまうので，Y^X の代わりに $[X \to Y]$ とも書くことにする．

高階汎関数を扱う際には，λ-記法が便利である．たとえば，$\lambda x.f(x, y)$ と書いた場合，x を入力として $f(x, y)$ を出力する 1 変数関数を表す．同様に，$\lambda y.f(x, y)$ と書いた場合は，y を入力として $f(x, y)$ を出力する 1 変数関数である．そして，$\lambda xy.f(x, y)$ と書けば，x, y を入力とする 2 変数関数 f を表している．**λ-抽象** (λ-*abstraction*) とは，変数 $\overline{x}$ を持つ項 $t(\overline{x})$ に対して，項 $\lambda\overline{x}.t(\overline{x})$ を割り当てる手続きである．

1925 年，ヒルベルトは，例 2.2 の型 2 汎関数 `iterate` を用いて，以下のような自然数上の関数の階層を考察した．

$$\begin{aligned}
\varphi_1(a, b) &:= a + b \\
\varphi_2(a, b) &:= \mathtt{iterate}(\varphi_1, a, b) = a \cdot b \\
\varphi_3(a, b) &:= \mathtt{iterate}(\varphi_2, a, b) = a^b \\
\varphi_4(a, b) &:= \mathtt{iterate}(\varphi_3, a, b) = a \uparrow\uparrow b \\
&\vdots \\
\varphi_{n+1}(a, b) &:= \mathtt{iterate}(\varphi_n, a, b) = a \uparrow^{n-1} b
\end{aligned}$$

ヒルベルトが注目したことは，関数 $A(n, a, b) = \varphi_{n+1}(a, b)$ を λ-抽象と型 2 汎関数 `iterate` を

用いたある種の原始再帰によって定義できるということであった[*3].

$$\begin{cases} A(0,a,b) = a+b, \\ A(n+1,a,b) = \texttt{iterate}(\lambda xy.A(n,x,y),a,b). \end{cases}$$

この関数 $A\colon \mathbb{N}^3 \to \mathbb{N}$ は，現在，**アッカーマン関数** (*Ackermann function*) として知られるものの原型である．つまり，ヒルベルトは，

アッカーマン関数 A は型 2 汎関数を用いた原始再帰法によって定義できる

ということを示したのである．そして，ヒルベルトは，関数 A は型 1 汎関数しか用いない原始再帰法によっては定義できないと予想した．つまり，現代的な用語を用いれば，A は原始再帰的関数ではないと予想したのである．ヒルベルトの目的は，「高い型を持つ汎関数を利用すれば，低い型を持つ汎関数しか利用しないよりも，真に多くの $\mathbb{N}$ 上の関数を定義できる」ということを示すことであった．このようにして，再帰的定義のために必要な汎関数の型のランクによって，$\mathbb{N}$ 上の関数たちを分類しようとしたのである．1927 年頃に，ヒルベルトの問題に解決を与え，A が原始再帰的でないことを示したのが，ヒルベルトの 2 人の弟子，ヴィルヘルム・アッカーマン (Wilhelm Ackermann; 1896–1962) とガブリエル・スーダン (Gabriel Sudan; 1899–1977) であり，このために A はアッカーマン関数と呼ばれることとなった．

定理 2.4 (スーダン, アッカーマン)**.** アッカーマン関数 A は原始再帰的ではない．

ただし，現在，アッカーマン関数として巷で最も広く知られているものは，上で定義した関数 A ではなく，ロザ・ペーター (Rózsa Péter; 1905–1977) による別の関数である．ペーターは，アッカーマン関数の定義から高階汎関数の概念を除去し，その定義を単純化した．あまりにもペーターの定義が広く流通したため，アッカーマン関数が高階汎関数の研究の過程で生まれたものであるということは，ほとんど忘れ去られてしまったようである．数学的にはあまり重要ではないが，ペーターによるアッカーマン関数 $A'\colon \mathbb{N}^2 \to \mathbb{N}$ の定義は以下によって与えられる．

$$\begin{aligned} A'(0,x) &= x+1, \\ A'(n+1,0) &= A'(n,1), \\ A'(n+1,x+1) &= A'(n,A'(n+1,x)). \end{aligned}$$

ペーターのアッカーマン関数が元のアッカーマン関数と異なるということについては，それなりに認知されており，ペーターの関数 A' は 2 変数関数であり，元の関数 A は 3 変数関数であるという点が違いとして強調されることがあるようである．ただ，アッカーマン関数の定義が 2 変数か 3 変数かという点は，とてつもなくどうでもいいことである．真に強調すべき点は，ヒルベルトの

[*3] 一応，注意しておくと，ヒルベルトの原論文においては λ 記法は用いられていない．

アッカーマン関数（アッカーマンが原論文で用いた関数）は「**型 2 再帰**」によって定義され，ペーターのアッカーマン関数は「**型 1 再帰**」によって定義される，という点であろう．

同種の関数を低い型の再帰で作れるならばそれに越したことはないと思うかもしれない．しかし，たとえば，第 3 節ではアッカーマン関数よりもさらに複雑な関数を考察するが，この場合，型 1 多重再帰を用いるとかなり見通しが悪くなる．それに比べると高階原始再帰はかなり透き通った構造を持つので，アッカーマン関数を高階再帰として理解しておくことにより，アッカーマン関数よりも先の世界が広く見渡せるようになるように思う．

さて，ここまでは，高階原始再帰関数の定義を若干曖昧にしていた．厳密な定義を述べる前に，アッカーマン関数の定義にどのように λ-抽象が使われていたかに注目しよう．まず，型 $X \to Y$ の自由変数 $\dot{\nu}$ と型 X の自由変数 $\dot{x}$ に対して，型 Y の項 $\dot{\nu}(\dot{x})$ を定義することは，基本構成として認められているとしよう．ここで，自由変数であることを強調するために，記号の上にドットを付けている．このとき，λ-抽象によって，

$$\mathsf{eval} = \lambda\dot{\nu}\dot{x}.\dot{\nu}(\dot{x})$$

という型 $[Y^X \times X \to Y]$ の項が定義される．次に，$h(\nu, a, z) = \mathsf{eval}(\nu, (a, z))$ に対して，以下の原始再帰法によって f を定義する．

$$\begin{cases} f(\nu, a, 0) = e, \\ f(\nu, a, n+1) = h(\nu, a, f(\nu, a, n)). \end{cases}$$

このとき $\mathtt{iterate} = f$ と定義する．いま，$h(\dot{u}, \dot{v}, t(\dot{a}, \dot{b})) = \mathtt{iterate}(\dot{u}, \dot{v}, \lambda\dot{a}\dot{b}.t(\dot{a}, \dot{b}))$ を考えよう．このとき，アッカーマン関数 A は以下のように定義される．

$$\begin{cases} A(0, \dot{a}, \dot{b}) = \dot{a} + \dot{b}, \\ A(n+1, \dot{a}, \dot{b}) = h(\dot{a}, \dot{b}, A(n, \dot{a}, \dot{b})). \end{cases}$$

このように書くと，λ-抽象によって型 2 汎関数が現れるものの，見かけ上は，ごく普通の原始再帰法のようである．しかし，$A(n+1, \dot{a}, \dot{b})$ を求める過程で，自由変数 $\dot{a}, \dot{b}$ をもつ $A(n, \dot{a}, \dot{b})$ にアクセスしているから，別の自由変数 $\dot{x}, \dot{y}$ に取り替えて $A(n, \dot{x}, \dot{y})$ を考えているのと本質的に変わらない．つまり，上の “原始再帰法” は，実質的には多重再帰の構造を持っている．

そういうわけで，高階汎関数を考える上で重要なのは，自由変数の取り扱いである．このため，関数本体を直接取り扱うよりも，まずは自由変数および基本演算と原始再帰法の組み合わせによって作られる項を考える．記法の簡便さのために，カリー化・非カリー化は暗黙に行われているとし，$[X \to [Y \to Z]]$ と $[X \times Y \to Z]$ を同一視する．

定義 2.5. 有限型原始再帰における**項** (*term*) とは，以下のように帰納的に定義されるものである．

- 型 X の自由変数 $\dot{x}$ は，型 X の項である．

- $0 \in \mathbb{N}$ は，型 $\mathbb{N}$ の項である．
- 後続関数 $n \mapsto n+1$ は，型 $[\mathbb{N} \to \mathbb{N}]$ の項である．
- v が型 $[X \to Y]$ の項で，x が型 X の項ならば，$v(x)$ は型 Y の項である．
- g が型 Y の項で，h が型 $[\mathbb{N} \times Y \to Y]$ の項ならば，

$$\begin{cases} f(0) = g \\ f(y+1) = h(y, f(y)) \end{cases}$$

によって定義される f は，型 $[\mathbb{N} \to Y]$ の項である．

一応，注意しておくと，項は自由変数を含みうるので，その自由変数を明示すれば，たとえば，実際には上の原始再帰法において，

$$\begin{cases} f(\dot{x}, 0) = g(\dot{x}) \\ f(\dot{x}, y+1) = h(\dot{x}, y, f(\dot{x}, y)) \end{cases}$$

のような項 f の構成も許容される．したがって，通常の原始再帰法で作られる関数はいずれも項として実現できる．

定義 2.6. 関数 $f\colon X_1 \times X_2 \times \cdots \times X_n \to Y$ が**有限型原始再帰的** (*finite type primitive recursive*) とは，型 $[X_1 \times X_2 \times \cdots \times X_n \to Y]$ の項 $t(\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n)$ が存在して，任意の $a_1 \in X_1, a_2 \in X_2, \ldots, a_n \in X_n$ に対して，$t(a_1, a_2, \ldots, a_n)$ を求めた結果が $f(a_1, a_2, \ldots, a_n)$ と等しいことを意味する．また，そのような項 t の構成において，型レベル n 以下の項しか現れない場合，f を**型 n 原始再帰的** (*type n primitive recursive*) と呼ぶことにする．

この定義の下で，たとえばアッカーマン関数 A が型 2 原始再帰的であることは，先程の構成に倣って証明できる．このようにして，高階汎関数を利用した原始再帰法は，原始再帰関数の外側の階層を作っていく (図 2, ここで型 n 原始再帰的な型 1 関数全体のクラスを Type n-$\mathcal{PR}$ と書く)．

■ **Historical Remark.** 数学基礎論の歴史の流れを見ると，1924 年，アッカーマンは，ヒルベルトの指導下で執筆した博士論文において，2 階原始再帰的算術の無矛盾性証明を行っていた．そういうわけで，1925 年のオリジナルのアッカーマン関数が，型 1 多重再帰ではなく型 2 汎関数による高階原始再帰法によって定義されたことは，数学基礎論の研究の流れから見れば，かなり自然だったようである．

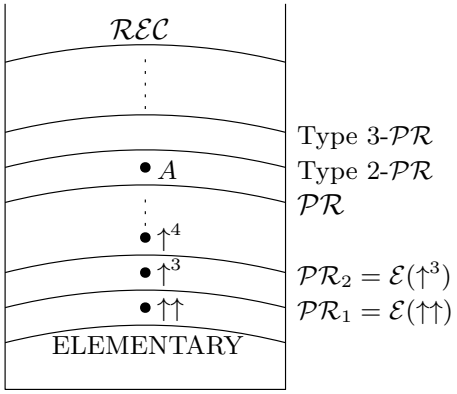

図 2　高階原始再帰関数の階層

2.2　弱グッドスタイン列とアッカーマン関数

アッカーマン関数はいかなる原始再帰関数よりも急増大である．そのような急増大関数が，一体どのような状況下で現れるのだろうか．本節では，ここまでと少し異なる観点から，アッカーマン関数程度の増大度を持つ関数を紹介しよう．

定義 2.7 (弱グッドスタイン列)**.** いま，この世界には，無数に多くの国が存在している．そして，それぞれの国は数値の表記に異なる位取り記数法を用いているとしよう．たとえば，我々のように，数字の表記に 10 進法を用いている国もあれば，2 進法や 16 進法を用いている国もある．自然数 $b \geqslant 2$ について，数字の表記に b 進法を用いている国を「b 進国」と呼ぶことにする．

初期値 $x \in \mathbb{N}$ の**弱グッドスタイン列** (*weak Goodstein sequence*) とは，次のような列である．

- 2 進国の住民が自然数 x を自国の言葉で記し，3 進国の住民に伝える．
- 3 進国の住民は，それを自国の言葉として理解し，その数から 1 引いて書き直す．そして，これを 4 進国の住民に伝える．
- 4 進国の住民は，それを自国の言葉として理解し，その数から 1 引いて書き直す．そして，これを 5 進国の住民に伝える．
- これを延々と続ける．

例 2.8. 具体例として，初期値 13 の弱グッドスタイン列は次のように計算できる．

- 2 進国では 13 のことを 1101 と書く．
- 3 進国では 1101 は 37 を表す．これを 1 引くと 1100 である．
- 4 進国では 1100 は 80 を表す．これを 1 引くと 1033 である．
- 5 進国では 1033 は 143 を表す．これを 1 引くと 1032 である．

つまり，弱グッドスタイン列は $13 \to 37 \to 80 \to 143 \to \ldots$ とつづく．

上記の例を見ると，弱グッドスタイン列は増大列のように見えるが，あるステップから下降を始め，いつか 0 に到達することが知られている (系 2.11)．このとき，$\mathrm{WGood}(x)$ を次によって定義する．

$$\mathrm{WGood}(x) = \text{“初期値 } x \text{ の弱グッドスタイン列が 0 に到達するまでのステップ数”}$$

この関数 WGood はどれくらいの増大度を持つだろうか．この分析のために，数学における順序構造の概念を導入しよう．いま，2 項関係 $\preceq$ に関する以下の性質を考える．

- 反射律: $x \preceq x$
- 推移律: $(x \preceq y \mathbin{\&} y \preceq z) \ \to \ x \preceq z$
- 反対称律: $(x \preceq y \mathbin{\&} y \preceq x) \ \to \ x = y$
- 比較可能律: $x \preceq y$ or $y \preceq x$

反射律と推移律を満たす 2 項関係 $\preceq$ を**擬順序** (*quasi-order*) または**前順序** (*preorder*) と呼ぶ．反対称律を満たす擬順序は，**半順序** (*partial order*) と呼ばれる．さらに，比較可能律を満たす半順序を**全順序** (linear order) という．擬順序 $\preceq$ が**整礎** (*well-founded*) であるとは，無限下降列 $x_1 \succ x_2 \succ x_3 \succ \ldots$ を持たないことを意味する．整礎な全順序を**整列順序** (*well-order*) と呼ぶ．そして，**順序数** (*ordinal*) とは，整列順序の順序型（順序同型による同値類）のことである．

順序数については，歴史的経緯から「数の無限への拡張」として語られることが多いようだが，むしろ，この「整礎性」という一種の《**有限性**》こそが順序数の本質であることを強調しておく．もう一点，順序数について注意しておくと，計算論などの関わる文脈では，整列順序本体ではなくその順序型を取り扱うことは，表記の簡便さを除くとデメリットしかない．したがって，あくまで順序数といっても，その背後にある整列順序を念頭に置いておいたほうがよいということも強調しておこう．

整列順序の重要な例のひとつが，以下の辞書式順序である．

例 2.9. 自然数の m 組全体の集合 $\mathbb{N}^m$ 上の**辞書式順序** $\leqslant_{\mathrm{lex}}$ を次によって定義する：自然数の m 組 $\bar{x} = (x_1, x_2, \ldots, x_m)$ および $\bar{y} = (y_1, y_2, \ldots, y_m)$ について，

$$\bar{x} <_{\mathrm{lex}} \bar{y} \iff (\exists i \leqslant m)(\forall k < i)\,[x_k = y_k \ \wedge \ x_i < y_i]$$

とし，$\bar{x} <_{\mathrm{lex}} \bar{y}$ または $\bar{x} = \bar{y}$ であるとき，$\bar{x} \leqslant_{\mathrm{lex}} \bar{y}$ と書く．

命題 2.10. $\mathbb{N}^m$ 上の辞書式順序は整列順序である．

$\mathbb{N}^m$ 上の辞書式順序の順序型は，ω^m と表記される．この辞書式順序に注目しているときは，各 $(x_0, x_1, \ldots, x_{m-1}) \in \mathbb{N}^m$ を形式的に

$$\omega^{m-1} \cdot x_{m-1} + \cdots + \omega \cdot x_1 + x_0$$

と表し，しばしば $\leqslant_{\mathrm{lex}}$ を単に $\leqslant$ と書く．

さて，弱グッドスタイン列の分析に戻ろう．例 2.8 で見たように，初期値 13 の弱グッドスタイン列は $13 \to 37 \to 80 \to 143 \to \dots$ であった．実際の数値ではなく，各国の言語での表記の列として見ると，$1101 \to 1100 \to 1033 \to 1032 \to \dots$ という列である．各ステップの 4 桁の数字を自然数の 4 組と考えると，これは辞書式順序における下降列になっていることが分かる．

$$1101 >_{\mathrm{lex}} 1100 >_{\mathrm{lex}} 1033 >_{\mathrm{lex}} 1032 >_{\mathrm{lex}} \dots$$

実際，いかなる初期値の弱グッドスタイン列も辞書式順序における下降列を与えることは容易に確認できる．したがって，命題 2.10 の帰結として，以下を得る．

系 2.11. 任意の自然数 $x \in \mathbb{N}$ について，初期値 x の弱グッドスタイン列はいつか 0 に到達する．したがって，$\mathrm{WGood}(x)$ は有限の値を持つ．

さて，関数 WGood の増大度の分析のため，表記上の弱グッドスタイン列 $\alpha_2 >_{\mathrm{lex}} \alpha_3 >_{\mathrm{lex}} \alpha_4 >_{\mathrm{lex}} \dots$ に再び着目しよう．ここで，α_2 は初期値の 2 進表記であり，α_n は第 n ステップ目の n 進表記である．表記上の弱グッドスタイン列について，桁数は非増大であるから，初期値の 2 進表記 α_2 の桁数が m 桁以下であれば，いずれの α_n も m 桁以下である．つまり，$(\alpha_n)_{n\in\mathbb{N}}$ は，$\mathbb{N}^m$ の辞書式下降列である．いま，n 進表記であることから，数値の表記上は，α_n のどの桁にも n 未満の数字しか現れないということにも注目しよう．

弱グッドスタイン列の場合，初期値によって桁数は変化していたが，桁数を固定しておいたほうが分析が容易なので，弱グッドスタイン列を少し修正した「m 桁弱グッドスタイン列」を考えよう．まず，初期値 x を 2 進国が受け取るのではなく，入力 x に対して，$(x+1)$ 進国の住民に「最も大きな m 桁の数字」を書いてもらい，それを初期値とする．たとえば，$m = 6$ かつ $x = 9$ ならば，初期値は 999999 である．それ以降のステップは，定義 2.7 と全く同様であり，これを m 桁弱グッドスタイン列と呼ぶことにする．

この場合，いかなる入力を考えても，どのステップも m 桁以下の数字で表記されるから，議論がすべて $\mathbb{N}^m$ の中で収まる．また，先ほどと同様に，m 桁弱グッドスタイン列の表記上の列を見れば，$\mathbb{N}^m$ の辞書式順序の下降列になっているから，いつか値が 0 に到達する．したがって，$\mathrm{WGood}_m(x)$ を以下のように定義できる．

$$\mathrm{WGood}_m(x) = \text{“入力 } x+1 \text{ の } m \text{ 桁弱グッドスタイン列が 0 に到達するまでのステップ数”}$$

まず，WGood と WGood_m の比較として，十分大きな任意の $x \in \mathbb{N}$ について，初期値 x の弱グッドスタイン列 $\alpha_2 >_{\mathrm{lex}} \alpha_3 >_{\mathrm{lex}} \dots$ が $(x+1)$ 進国のステップに辿り着いたときの数の $(x+1)$ 進表記 α_{x+1} は m 桁を越えている．これより $\mathrm{WGood}(x) > \mathrm{WGood}_m(x)$ がただちに導かれるから，つまり次の命題を得る．

命題 2.12. 任意の $m \in \mathbb{N}$ について，WGood_m は WGood に支配される．

関数 WGood_m がどの程度の増大度を持つかを分析するために，関数の急増大度の指標となる急増加関数の階層を導入しよう．第 2.1 節で定義したヒルベルトのアッカーマン関数 $A(n,a,b) = \varphi_{n+1}(a,b)$ は，関数の階層 $(\varphi_{n+1})_{n\in\mathbb{N}}$ を伴うものであった．以後，この関数の階層を φ_{n+1} の代わりに F_n で表すものとする．つまり，

$$\begin{aligned} F_0(x) &= x+1, \\ F_{n+1}(x) &= \underbrace{F_n \circ F_n \circ \cdots \circ F_n}_{x \text{ 個}}(x). \end{aligned}$$

命題 2.12 より，WGood に比べると，m 桁制限版 WGood_m の増大度は小さい．それにも関わらず，以下に示すように，WGood_{m+1} の増大度はかなりのもので，アッカーマン関数の第 m レベルの急増大関数 F_m 以上に増大することがわかる．

定理 2.13. 任意の $n \in \mathbb{N}$ について，$F_m(n) \leqslant \mathrm{WGood}_{m+1}(n)$ が成立する．

証明. $h_m = \mathrm{WGood}_m$ とおく．まず，$h_1(n)$ を求める．入力 n の桁 1 弱グッドスタイン列は，明らかに

$$n > n-1 > n-2 > \cdots > 1 > 0$$

であるから，その長さは $h_1(n) = n+1 = F_0(n)$ である．

与えられた $n \in \mathbb{N}$ について，$h_{m+1}(n)$ を求めよう．入力 n の桁 $(m+1)$ 弱グッドスタイン列の初期値は

$$\alpha_0 := \omega^m \cdot n + \omega^{m-1} \cdot n + \cdots + \omega \cdot n + n$$

である．第 2 項以降の項からなる多項式 β_0 は，入力 n の桁 m 弱グッドスタイン列の初期値である．よって，入力 n の桁 m 弱グッドスタイン列の長さは $h_m(n)$ であることから，そのような長さ $h_m(n)$ の辞書式下降列 $\beta_0 > \beta_1 > \cdots > \beta_{h_m(n)-1} = 0$ を考えると，

$$\begin{aligned} \alpha_0 = \omega^m \cdot n + \beta_0 > \alpha_1 := \omega^m \cdot n + \beta_1 > \ldots \\ \cdots > \alpha_{h_m(n)-1} := \omega^m \cdot n + \beta_{h_m(n)-1} = \omega^m \cdot n \end{aligned}$$

というように桁 $(m+1)$ 弱グッドスタイン列は進行していく．このとき，

$$\alpha_{h_m(n)} := \omega^m \cdot (n-1) + \omega^{m-1} \cdot h_m(n) + \cdots + h_m(n)$$

が次のステップの値となる．先程と同様に，第 2 項以降の項からなる多項式は，入力 $h_m(n)$ の桁 m 弱グッドスタイン列の初期値である．よって，入力 $h_m(n)$ の桁 m 弱グッドスタイン列の長さは

$h_m \circ h_m(n)$ であることから，上の議論と同様にして，$\alpha_{h_m(n)+h_m\circ h_m(n)-1} = \omega^m \cdot (n-1)$ を得る．これを繰り返せば，

$$h_{m+1}(n) \geqslant \underbrace{h_m \circ h_m \circ \cdots \circ h_m}_{n\text{ 個}}(n)$$

を求めることができる．よって，帰納的に，任意の $x \geqslant m$ について $h_{m+1}(x) \geqslant F_m(x)$ を得る． □

以上より，定義 2.7 の弱グッドスタイン列から得られた WGood という極めて単純な関数が，如何なる原始再帰関数よりも急増大である，すなわちアッカーマン的に増大するということが示された．

系 2.14. 任意の m について，WGood は F_m を支配する．特に，WGood は任意の原始再帰関数を支配する．

2.3 「スーダン関数」の謎*

(※本節の内容は現代的にはあまり重要でないトピックなので，飛ばしても問題ない．)

ガブリエル・スーダンは，ルーマニアの数学者である．ゲッティンゲンのヒルベルトの下で学位を取った後は，ずっとルーマニアで生活していたようだ．スーダンの主要業績は，「超限順序数 ω^ω について (Sur le nombre transfini ω^ω)」という題でフランス語で執筆され，ルーマニアのあまり知られていない論文誌において 1927 年に出版された．スーダンの成し遂げた仕事の紹介は，ほとんどがルーマニアの研究者によってなされ，その一部はルーマニア語で書かれているようなので，その全貌を知るのは難しい．

2019 年 7 月現在，ウィキペディアで 7 ヵ国語で書かれている「**スーダン関数** (*Sudan function*)」という記事がある．これは，以下のような多重再帰によって定義された関数だそうである．

$$\begin{cases} F_0(x,y) = x+y, \\ F_{n+1}(x,y+1) = F_n(F_{n+1}(x,y), F_{n+1}(x,y)+y+1). \end{cases}$$

どの言語の記事を見ても，このような単純な「原始再帰的でない多重再帰関数」を，あたかもアッカーマン以前にスーダンが導入していたかのように紹介されている．しかし，ペーターのアッカーマン関数 A' がアッカーマンによって定義されたものではないのと同様に，スーダン関数もまたスーダンによって定義されたものではなさそうである．

2019 年 7 月現在，上記のような「スーダン関数」について記載された英字のまともな学術的論文は見つけることはできなかった（上記の「スーダン関数」ではなく，スーダンが実際に定義した本物のスーダン関数について書かれた論文はいくつかある）．論文以外の学術的文献を含めるならば，ルーマニアの計算機科学者 Cristian Calude によって執筆された計算複雑性の教科書 [2] が，英文で「スーダン関数」に触れた唯一のまともな文献であると思われる．Calude の教科書によれば，

「スーダン関数」は 1981 年に Nelu Dima によるルーマニア語で書かれた論文において定義されたもののようである．したがって，上の関数は，**Dima のスーダン関数** (Dima's Sudan function) とでも呼ぶほうが適切であると思われる．

スーダンは 1927 年の論文中で様々な関数を再帰的定義の例として挙げているが，その中で「Dima のスーダン関数」に最も近いものは原論文 13 頁において定義された関数 φ だと思われる．上の「Dima のスーダン関数」に近い記法で記述すれば，これは以下の**順序数値**関数である．

$$\begin{cases} F_0(x,y) = x + y, \\ F_{n+1}(x, y+1) = \sup_{z<\omega} F_n(F_{n+1}(x,y), z). \end{cases}$$

正確には，スーダンの原論文の関数は，n も変数として動く 3 変数関数である．スーダンは，これ以外にも様々な自然数上の関数と順序数値関数を取り上げているが，自然数上の関数については，いずれも Dima のスーダン関数からは程遠い．

それでは，スーダンはどのような文脈で，このような関数を考えたのだろうか．前節で集合上の原始再帰的定義を導入したが，ヒルベルトが主として考えていたものは，$\mathbb{N}$ を基礎型とする高階汎関数の集合と，そして順序数の集合である．ここでは，原始再帰法による順序数の具体的構成を考察してみよう．いま，順序数の後続を取る関数 $\alpha \mapsto \alpha + 1$ は初期関数として含まれているとする．また，順序数値関数 $\alpha\colon \mathbb{N} \to \mathsf{Ord}$ が与えられているとき，上限 $\sup_n \alpha(n)$ を取るという構成もまた基本構成として許容されるものと仮定しよう．

順序数値関数の原始再帰的構成で難しい部分は，各原始再帰法においては ω ステップで構成を終える必要があるという点である．このために，2 つの順序数の和ですら，どのように定義すればよいか分からない．しかし，順序数 α と自然数 b について，$\psi_1(\alpha, b) = \alpha + b$ を与える関数 ψ_1 であれば，原始再帰法で容易に定義できる．これを用いると，たとえば，以下のような関数の階層を考えることができる．

$$\begin{cases} \psi_{n+1}(\alpha, 0) = \alpha, \\ \psi_{n+1}(\alpha, b+1) = \sup_c \psi_n(\psi_{n+1}(\alpha, b), c). \end{cases}$$

具体的に計算してみると，

$$\psi_2(\alpha, b+1) = \sup_c \psi_1(\psi_2(\alpha, b), c) = \sup_c(\psi_2(\alpha, b) + c) = \psi_2(\alpha, b) + \omega = \alpha + \omega \cdot (b+1),$$
$$\psi_3(\alpha, b+1) = \sup_c \psi_2(\psi_3(\alpha, b), c) = \sup_c(\psi_3(\alpha, b) + \omega \cdot c) = \psi_3(\alpha, b) + \omega^2 = \alpha + \omega^2 \cdot (b+1).$$

これを繰り返すことにより，一般に $\psi_n(\alpha, b) = \alpha + \omega^{n-1} \cdot b$ となることが計算できる．したがって，原始再帰的な方法で ω^ω 未満の順序数を作ることができる．たとえば，$\omega^{31} \cdot 4 + \omega^{15} \cdot 9 + \omega^2 \cdot 6$ であれば，以下のように書ける．

$$\psi_2(\psi_{15}(\psi_{31}(0,4), 9), 6) = \omega^{31} \cdot 4 + \omega^{15} \cdot 9 + \omega^2 \cdot 6.$$

あまり標準的な用語ではないが，原始再帰的な手続きのみで作ることができる順序数を**原始再帰的算出可能順序数**[*4]と呼ぶことにする．より正確には，$\mathbb{N}$ と Ord を基礎型とし，高階汎関数を用い

[*4] 可算順序数 α が**計算可能**あるいは**再帰的**であるとは，次のような $\mathbb{N}$ 上の整列順序 $\prec$ が存在することを意味する：

ない原始再帰法によって定義される関数 $\psi\colon \mathbb{N} \to \mathsf{Ord}$ が存在して，$\psi(0) = \alpha$ となるとき，α は原始再帰的算出可能であるとする．上記の議論より，以下の命題を得る．

命題 2.15 (スーダン 1927)**.** ω^ω 未満の順序数は原始再帰的算出可能である．

スーダンの主定理は，この順序数 ω^ω が原始再帰法にとっての超えられない壁であるというものであった．つまり，スーダンが示したことは以下である．

定理 2.16 (スーダン 1927)**.** ω^ω は原始再帰的算出可能でない．

スーダンによって指摘されたように，上で述べた順序数上の関数の階層 (ψ_n) は，次のような型 2 汎関数 $\mathtt{iterate}'$ を用いて作ることも可能である：関数 $f\colon A \to A$ と $x \in A$ および $y \in \mathbb{N}$ について，

$$\mathtt{iterate}'(f, x, y) = \underbrace{f \circ f \circ f \circ \cdots \circ f}_{y\ 個}(x).$$

この型 2 汎関数は，以下のように集合上の原始再帰法によって構成できる．

$$\begin{cases} \mathtt{iterate}'(f, x, 0) = x, \\ \mathtt{iterate}'(f, x, y+1) = f(\mathtt{iterate}'(f, x, y)). \end{cases}$$

型 2 汎関数 $\mathtt{iterate}'$ を用いれば，関数の階層 (ψ_n) は以下のように定義できる：順序数 α と自然数 b について，

$$\begin{cases} \Psi(0, \alpha, b) = \alpha + b, \\ \Psi(n+1, \alpha, b) = \mathtt{iterate}'(\lambda x.\, \sup_c \psi(n, x, c), \alpha, b). \end{cases}$$

実際に計算してみれば，$\Psi(n, \alpha, b) = \psi_{n+1}(\alpha, b) = \alpha + \omega^n \cdot b$ となることは容易に分かる．つまり，型 2 汎関数を用いた原始再帰法によって Ψ を定義することができ，よって，$\sup_n \Psi(n, 0, 1) = \omega^\omega$ もまた定義できる．したがって，順序数 ω^ω は型 2 原始再帰的算出可能であるが，型 1 原始再帰的算出可能ではない．

$(\mathbb{N}, <)$ の順序型が α であり，さらに "$\chi_<(x, y) = 1 \iff x < y$" によって定義される関数 $\chi_<\colon \mathbb{N}^2 \to \{0, 1\}$ が計算可能である．

構成的順序数という用語も，これと同値な概念を指す際に用いられる．この $\chi_<$ が原始再帰的であるとしても，計算可能順序数であることと等しくなるため，原始再帰的順序数という用語を用いると混乱を招く恐れがある．このため，本稿では，原始再帰的算出可能という用語を用いることにした．

3　急増加関数の階層

3.1　急増加階層と順序数

第 2.1 節で定義したヒルベルトのアッカーマン関数 $A(n,a,b) = \varphi_{n+1}(a,b)$ は，関数の階層 $(\varphi_{n+1})_{n\in\mathbb{N}}$ を伴うものであった．第 2.2 では，この関数の階層を φ_{n+1} の代わりに F_n で表した．この節以降では，さらに，アッカーマン関数に相当する急増加関数を F_ω で表す．つまり，

$$\begin{aligned} F_0(x) &= x+1, \\ F_{n+1}(x) &= \underbrace{F_n \circ F_n \circ \cdots \circ F_n}_{x\text{ 個}}(x), \\ F_\omega(x) &= F_x(x). \end{aligned}$$

この発想は，**急増加階層** (*fast-growing hierarchy*) というものに拡張される．たとえば，各 $n \leqslant \omega$ について，以下のような関数 $F_{\omega+n}\colon \mathbb{N} \to \mathbb{N}$ を定義できることは明らかであろう．

$$\begin{aligned} F_{\omega+n+1}(x) &= \underbrace{F_{\omega+n} \circ F_{\omega+n} \circ \cdots \circ F_{\omega+n}}_{x\text{ 個}}(x), \\ F_{\omega+\omega}(x) &= F_{\omega+x}(x). \end{aligned}$$

これを繰り返して，可算順序数 α に対して，急増加関数 $F_\alpha\colon \mathbb{N} \to \mathbb{N}$ を定義していこう，というものが急増加階層の発想である．ただし，これはこの言葉通りに都合よく行くわけではない．実際には，可算順序数 α ではなく**可算整礎木** $T \subseteq \mathbb{N}^*$ に対して，急増加関数 $F_T\colon \mathbb{N} \to \mathbb{N}$ を定義できるだけである，という点に注意しておこう．たとえば，十分小さい順序数 α, たとえば $\alpha < \varepsilon_0$ については，基本列 $(\alpha[n])_{n\in\mathbb{N}}$ というものを割り当てる標準的な方法があり，ここから可算整礎木 $T(\alpha)$ が定義されるから，$F_\alpha = F_{T(\alpha)}$ の標準的な定義がある．しかし，一般の可算順序数 α に対して，急増加関数 F_α が定義されるわけではない．

順序数 α の**基本列** (*fundamental sequence*) とは，α 未満の順序数の増大列

$$\alpha[0] \leqslant \alpha[1] \leqslant \alpha[2] \leqslant \cdots \leqslant \alpha[n] \leqslant \alpha[n+1] \leqslant \cdots < \alpha$$

であって，$\{\alpha[n] : n \in \mathbb{N}\}$ が α で共終である，つまり，任意の $\beta < \alpha$ について $\beta \leqslant \alpha[n]$ となる $n \in \mathbb{N}$ が存在することである．後者の条件は，$\alpha = \sup_{n\in\mathbb{N}}(\alpha[n]+1)$ となることと同値である．

例 3.1. ε_0 未満の順序数には，以下の標準的な方法によって，基本列を割り当てることができる．

1. 後続順序数 $\alpha = \beta + 1$ に対しては，定数列 $\alpha[n] = \beta$ を基本列として割り当てる．
2. 既に基本列が割り当てられている順序数 $\beta > 0$ と自然数 c について $\alpha = \omega^\beta \cdot (c+1)$ となっている場合，$\alpha[n]$ を以下によって定義する．

$$\alpha[n] = \begin{cases} \omega^\beta \cdot c + \omega^{\beta[n]} \cdot n & (\beta \text{ が後続順序数の場合}) \\ \omega^\beta \cdot c + \omega^{\beta[n]} & (\beta \text{ が極限順序数の場合}) \end{cases}$$

3. 一般に，$\alpha < \varepsilon_0$ の場合，α を**カントール標準形** (*Cantor normal form*) によって表せば，

$$\alpha = \omega^{\beta_0} \cdot c_0 + \omega^{\beta_1} \cdot c_1 + \cdots + \omega^{\beta_\ell} \cdot c_\ell$$

となる順序数の下降列 $\beta_0 > \beta_1 > \cdots > \beta_\ell$ と自然数列 $c_0, c_1, \ldots, c_\ell > 0$ が存在する．もし $\alpha \neq 0$ であれば，$\alpha[n]$ を以下のように帰納的に定義する．

$$\alpha[n] = \omega^{\beta_0} \cdot c_0 + \omega^{\beta_1} \cdot c_1 + \cdots + (\omega^{\beta_\ell} \cdot c_\ell)[n]$$

個々の順序数 α に対して無数の基本列 $(\alpha[n])$ が有り得るが，このように十分小さい順序数 α に対しては標準的な基本列 $(\alpha[n])_{n\in\mathbb{N}}$ が選択されていると仮定しよう．本稿では ε_0 未満の順序数しか用いないので，この仮定は常に適用できる．この仮定を適用可能であるような十分小さい順序数 α に対して，以下の階層を考えよう．

定義 3.2. 急増加関数 $F_\alpha \colon \mathbb{N} \to \mathbb{N}$ を以下によって帰納的に定義する．

$$F_{\alpha+1}(x) = \underbrace{F_\alpha \circ F_\alpha \circ \cdots \circ F_\alpha}_{x \text{ 個}}(x),$$
$$F_\alpha(x) = F_{\alpha[x]}(x). \qquad (\alpha \text{ が極限順序数の場合})$$

例 3.1 によって，$\alpha < \varepsilon_0$ については，関数 F_α が定義された．このように，小さい可算順序数 α に対して，なんとか F_α を整合的に扱うことができる．このように定義された関数の階層 (F_α) は，**急増加階層** (*fast-growing hierarchy*) と呼ばれる．しかし，何度も繰り返すが，大きな可算順序数 α に対しては，一般には F_α は定義されていないと思っておいた方が安全である．

もちろん，$\omega \leqslant \alpha < \varepsilon_0$ に対して，F_α の定義は原始再帰法によるものではない．関数の定義に順序数を伴うので，これは超限再帰の一種である．しかし，順序数の整礎性より，定義に従って $F_\alpha(x)$ の値を求める手続きは有限ステップで停止する．このような種類の再帰は**整礎再帰** (*well-founded recursion*), **後退再帰** (*backward recursion*), あるいは**バー再帰** (*bar recursion*) のように呼ばれることがある．

特に，F_α のように初期関数や再帰的定義が計算可能な方法で与えられているときには，この手法は**実効超限再帰** (*effective transfinite recursion*) とも呼ばれる，この場合は，後退再帰の各ステップが有限時間で計算できることと後退ステップ数が有限であることから，どんな入力 x が与えられても，$F_\alpha(x)$ の値は有限時間で求めることができる……と簡単に思うかもしれないが，証明を試みるとそこまで自明ではないことが分かる．数学的には，

「実効超限再帰によって定義される任意の関数は計算可能である」

という事実の証明は，**クリーネの再帰定理** (*Kleene's recursion theorem*) の代表的な応用例のひとつであり，計算可能性理論の基本的な定理のひとつとして知られる．

注意. 順序数の基本列の指定と可算整礎木構造との関連性を注意しておこう．木の根には，与えられた順序数 α をラベル付ける．この木は一般には無限分岐木であり，最初の分岐の直後のノードの左から順に，$\alpha[0], \alpha[1], \ldots, \alpha[n], \alpha[n+1], \ldots$ とラベル付けていく．$\alpha[n] \neq 0$ ならば，これをラベル付けたノードもまた無限分岐である．この分岐の直後のノードの左から順に，$\alpha[n][0], \alpha[n][1], \ldots, \alpha[n][m], \alpha[n][m+1], \ldots$ とラベル付ける．この手続を繰り返していく．ラベルの値が 0 となったノードは葉となり，その先にノードが伸びることはない．つまり，木 $T \subseteq \mathbb{N}^*$ の各ノードは自然数列 $(n_0, n_1, n_2, \ldots, n_\ell) \in \mathbb{N}^*$ であるが，これと順序数 $\alpha[n_0][n_1][n_2]\ldots[n_\ell]$ が対応付いていると考える．このように，順序数の基本列を指定するという行為は，順序数の背後にある木構造を明示する行為と思える．一般に，順序数に関する議論は，木構造に関する議論だと思った方がよいことがある．

ここでは，一般的な設定で整礎原始再帰を導入しておこう．上では，急増加関数 F_α という，見かけ上は可算順序数に沿って定義されているような関数を導入した．ここで，**可算順序数** (*countable ordinal*) とは，$\mathbb{N}$ 上の整列順序の順序型を意味する．ただし，何度でも強調するが，「順序型」に対する定義というのは絶対に上手くいかないので，実際には F_α というものが定義されているわけではなく，$\mathbb{N}$ 上の整列順序 $\preceq$ そのものに対する $F_\preceq$ あるいは可算整礎木 $T \subseteq \mathbb{N}^*$ に対する F_T しか定義されていない[*5]．

一方で，$\mathbb{N}$ 上の整列順序や整礎木が与えられたとき，それに対する整礎再帰を考えることができる．$\mathbb{N}$ 上の整礎関係 $\preceq$ に対して，$x \prec y$ を $x \preceq y$ かつ $x \neq y$ であることの略記とする．関数 $\theta\colon \mathbb{N}^{n+1} \to \mathbb{N}$ が $\preceq$-後退関数であるとは，任意の $\bar{x} \in \mathbb{N}^n$ と $a \in \mathbb{N}$ に対して $\theta(\bar{x}, a) \preceq a$ であることを意味する．一般に，整礎原始再帰と呼ばれる以下の関数構成を導入できる．

定義 3.3. $\preceq$ を $\mathbb{N}$ 上の整礎関係とし，θ を $\preceq$-後退関数であるとする．いま，与えられた関数 $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ と $h\colon \mathbb{N}^{n+2} \to \mathbb{N}$ から，関数 $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ を以下のように定義する．

$$f(\bar{x}, a) = \begin{cases} g(\bar{x}, a) & \text{if } \theta(\bar{x}, a) = a \\ h(\bar{x}, a, f(\bar{x}, \theta(\bar{x}, a))) & \text{if } \theta(\bar{x}, a) \prec a \end{cases}$$

このとき，f は g と h から θ に沿う $\preceq$ 上の**整礎原始再帰法**によって定義されるという．

初期関数を含み，合成と $\preceq$-整礎原始再帰法で閉じた最小の関数族に属す関数を $\prec$-原始再帰関数と呼ぶ．順序型 α 以下の原始再帰的整礎関係 $\preceq$ に対して $\prec$-原始再帰的であるような関数を α-原始再帰関数と呼ぶ．たとえば F_α は ω^α-原始再帰的である．一応，注意しておけば，定義 3.3 には自然数上の関数しか現れていない（たとえば，$\mathbb{N}$ 上の関係 $\preceq$ は，$\mathbb{N}^2$ 上の 2 値関数と同一視される）から，一般に，α-原始再帰は自然数上のごく普通の再帰の一種にすぎない．

[*5] 記法の簡便化のために，順序数 α に対して何か X_α を定義しているように見せかける，という議論はよくあるのだが，実際には，順序数ではなく順序数記法あるいは木構造 T に対して X_T を定義している場合がある．これは急増加階層だけでなく，無矛盾性述語の超限累積，極限計算のエルショフ階層など他にも様々な例がある．これはあくまで記法の簡便性のためであり，諸々の必要な性質を満たす $T \mapsto X_T$ は存在しても，同様の性質を満たす well-defined な $\alpha \mapsto X_\alpha$ は存在しないと数学的に証明されていることがしばしばあるため，この点には注意しておこう．

3.2 型 2 原始再帰とランク $< \omega^\omega$ の急増加関数

型 2 原始再帰によって，アッカーマン関数，したがって F_ω が定義可能なことは，第 2.1 節で見たとおりである．具体的には，高階原始再帰によって，型 2 汎関数 $\texttt{iterate}'\colon \mathbb{N}^\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ を定義することができた：

$$\texttt{iterate}'(f, x) = \underbrace{f \circ f \circ \cdots \circ f}_{x \text{ 個}}(x).$$

この汎関数は，急増加階層を 1 ランク持ち上げるので，以後は F'_{+1} と書くことにしよう．つまり，$F'_{+1}(f, x) = \texttt{iterate}'(f, x)$ であり，$F'_{+1}(F_\alpha, x) = F_{\alpha+1}(x)$ となる．ただし，2 変数汎関数 $F'_{+1}\colon \mathbb{N}^\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ よりも，関数 F_α を $F_{\alpha+1}$ に変換する 1 変数汎関数 $F_{+1}\colon \mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}$ を考えた方が以後の議論の見通しがよい．つまり，$F_{+1}(f) = \lambda x.F'_{+1}(f, x)$ と定義し，以後はこれを主に用いることにしよう．

さて，アッカーマン関数は第 ω ランクの急増加関数に相当するが，この構成に倣えば，急増加階層を ω ランク持ち上げる型 2 汎関数 $F_{+\omega}\colon \mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}$ を次によって定義できる：

$$\begin{cases} F^*_{+\omega}(0, f) = f, \\ F^*_{+\omega}(n+1, f) = F_{+1}(F^*_{+\omega}(n, f)), \\ F_{+\omega}(f) = \lambda x.F^*_{+\omega}(x, f)(x). \end{cases}$$

言い換えれば，これは以下の性質を持つ汎関数である．

$$F_{+\omega}(f)(x) = \left(\underbrace{F_{+1} \circ F_{+1} \circ \cdots \circ F_{+1}}_{x \text{ 個}}(f)\right)(x)$$

実際にこれが急増加ランクを ω 持ち上げることを確認しよう．

命題 3.4. $\alpha < \varepsilon_0$ について，$F_{+\omega}(F_\alpha) = F_{\alpha+\omega}$ である．

Proof. 実際に計算してみると，$F^*_{+\omega}(0, F_\alpha) = F_\alpha$ であり，

$$F^*_{+\omega}(1, F_\alpha) = F_{+1}(F^*_{+\omega}(0, F_\alpha)) = F_{+1}(F_\alpha) = F_{\alpha+1}$$

を得る．これを繰り返せば，$F^*_{+\omega}(n, F_\alpha) = F_{\alpha+n}$ であると分かる．よって，$F_{+\omega}(F_\alpha)(x) = F_{\alpha+x}(x) = F_{\alpha+\omega[x]}(x) = F_{\alpha+\omega}(x)$ を得る． □

このアイデアを一般化して，急増加階層を ω^k ランク持ち上げる型 2 汎関数 $F_{+\omega^k}\colon \mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}$ を定義しよう．ここで，$\omega^0 = 1$ なので，$F_{+\omega^0}$ と書いた場合，F_{+1} を意味する．

$$\begin{cases} F^*_{+\omega^{k+1}}(0, f) = f, \\ F^*_{+\omega^{k+1}}(n+1, f) = F_{+\omega^k}(F^*_{+\omega^{k+1}}(n, f)), \\ F_{+\omega^{k+1}}(f)(x) = \lambda x.F^*_{+\omega^{k+1}}(x, f)(x). \end{cases}$$

この型 2 原始再帰構成も意味としては単純で，以下の性質を持つ汎関数を与える．

$$F_{+\omega^{k+1}}(f)(x) = \left(\underbrace{F_{+\omega^k} \circ F_{+\omega^k} \circ \cdots \circ F_{+\omega^k}}_{x \text{ 個}}(f)\right)(x)$$

となっている（下の命題 3.5 の証明も参考にしてみよう）．実際に，これが急増加ランクを ω^k 持ち上げることを確認しよう．

命題 3.5. $\alpha < \varepsilon_0$ について，$F_{+\omega^k}(F_\alpha) = F_{\alpha+\omega^k}$ である．

Proof. k 上の数学的帰納法による．既に $F_{+\omega^k}(F_\alpha) = F_{\alpha+\omega^k}$ は証明されているとしよう．先ほどと同じように，$F^*_{+\omega^{k+1}}(0, F_\alpha) = F_\alpha$ であり，

$$F^*_{+\omega^{k+1}}(1, F_\alpha) = F_{+\omega^k}(F^*_{+\omega^{k+1}}(0, F_\alpha)) = F_{+\omega^k}(F_\alpha) = F_{\alpha+\omega^k}$$

を得る．これを繰り返せば，$F^*_{+\omega^{k+1}}(n, F_\alpha) = F_{\alpha+\omega^k \cdot n}$ であると分かる．よって，$F_{+\omega^{k+1}}(F_\alpha)(x) = F_{\alpha+\omega^k \cdot x}(x) = F_{\alpha+\omega^{k+1}[x]}(x) = F_{\alpha+\omega^{k+1}}(x)$ を得る． □

型 1 原始再帰では，$n < \omega$ までの急増加関数 F_n しか作れなかった．しかし，型 2 原始再帰を用いると，命題 3.5 より，$\alpha < \omega^\omega$ までの急増加関数 F_α を作ることができるのである．

定理 3.6. 任意の順序数 $\alpha < \omega^\omega$ に対し，急増加関数 $F_\alpha \colon \mathbb{N} \to \mathbb{N}$ は型 2 原始再帰的である．

Proof. 命題 3.5 より，関数 F_α は，F_0 と型 2 汎関数 $F_{+\omega^k}$ たちの合成によって表すことができる（具体な構成は，次節の例 3.11 を参考にせよ）．F_0 は原始再帰的であり，$F_{+\omega^k}$ が型 2 原始再帰的であることから，F_α も型 2 原始再帰的であることが従う． □

以上をまとめると，型 1 関数を反復合成する型 2 汎関数 F_{+1} の有限回の反復合成によって有限個の型 2 汎関数 $F_{+\omega^k}$ を作り，これに初期関数 F_0 を適用することによって，急増加関数 F_α を作ることができた．つまり，急増加関数 F_α の構成から超限再帰を除去し，型 2 汎関数による有限的な再帰によって置き換えたと考えられる．

3.3 型 3 原始再帰とランク $< \omega^{\omega^\omega}$ の急増加関数

それでは，ω^ω 以上のランクの急増加関数を作るにはどうすればよいだろうか．その答えが**型 3 原始再帰**である．次の型 3 汎関数 $\mathcal{I} \colon [\mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}] \to [\mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}]$ を考えよう：

$$\begin{cases} \mathcal{I}^*(0, F)(f) = f, \\ \mathcal{I}^*(n+1, F)(f) = F(\mathcal{I}^*(n, F)(f)), \\ \mathcal{I}(F)(f) = \lambda x. \mathcal{I}^*(x, F)(f)(x). \end{cases}$$

少し分かりにくいかもしれないが，これは `iterate` の型 3 版である．つまり，`iterate` が与えられた型 1 関数を反復合成する型 2 汎関数であるのに対し，$\mathcal{I}$ は与えられた型 2 汎関数を反復合成する型 3 汎関数である．つまり，以下が成立する．

$$\mathcal{I}(F)(f)(x) = \left(\underbrace{F \circ F \circ \cdots \circ F}_{x \text{ 個}}(f)\right)(x).$$

この型 3 手続きは，前節の型 2 汎関数 $F_{+\omega^k}$ から新たな型 2 汎関数 $F_{+\omega^{k+1}}$ を作る原始再帰法を型 3 汎関数 $\mathcal{I}$ として表したものでもある．したがって，$\mathcal{I}(F_{+\omega^k}) = F_{+\omega^{k+1}}$ となることが確認できる．

この $\mathcal{I}$ が急増加関数のランクをどれくらい持ち上げるかについて考察しよう．これを明示化するために，$\mathcal{I}$ のことを $F_{+\bullet\cdot\omega}$ と書くことにしよう．以下，順序数 α として，関数 F_α が定義されているようなものを取る．また，順序数 β として，$F_{+\beta}(F_\alpha) = F_{\alpha+\beta}$ となる型 2 汎関数 $F_{+\beta}$ が定義されているようなものとする．記法 $F_{+\bullet\cdot\omega}$ が意味するものは，以下の性質である．

命題 3.7. $F_{+\bullet\cdot\omega}(F_{+\beta})(F_\alpha) = F_{+\beta\cdot\omega}(F_\alpha) = F_{\alpha+\beta\cdot\omega}$ が成立する．

Proof. 前のように，$\mathcal{I}^*(0, F_{+\beta})(F_\alpha) = F_\alpha$ であり，

$$\mathcal{I}^*(1, F_{+\beta})(F_\alpha) = F_{+\beta}(\mathcal{I}^*(0, F_\alpha)) = F_{+\beta}(F_\alpha) = F_{\alpha+\beta}$$

を得る．これを繰り返せば，$\mathcal{I}^*(n, F_{+\beta})(F_\alpha) = F_{\alpha+\beta\cdot n}$ であると分かる．よって，$\mathcal{I}^*(F_{+\beta})(F_\alpha)(x) = F_{\alpha+\beta\cdot x}(x) = F_{\alpha+\beta\cdot\omega[x]}(x) = F_{\alpha+\beta\cdot\omega}(x)$ を得る． □

これを用いて，F_{ω^ω} が型 3 原始再帰的であることを示してみよう．このために，型 3 汎関数 $F_{+\bullet\cdot\omega}$ を用いた以下の型 3 原始再帰を考えよう．

$$\begin{cases} \mathcal{G}^*(0, F) = F, \\ \mathcal{G}^*(n+1, F) = F_{+\bullet\cdot\omega}(\mathcal{G}^*(n, F)), \\ \mathcal{G}(F) = \lambda f.\lambda x.\mathcal{G}^*(x, F)(f)(x). \end{cases}$$

この式もごちゃごちゃしてややこしいかもしれないが，与えられた型 2 汎関数 F に型 3 反復合成汎関数 $\mathcal{I} = F_{+\bullet\cdot\omega}$ を有限回反復合成した結果を返す型 3 汎関数である．つまり，以下のように書き直せる．

$$\mathcal{G}(F)(f)(x) = \left(\underbrace{F_{+\bullet\cdot\omega} \circ F_{+\bullet\cdot\omega} \circ \cdots \circ F_{+\bullet\cdot\omega}}_{x \text{ 個}}(F)\right)(f)(x).$$

この書き換えについては，以下の命題 3.8 の証明も参考にするとよい．この型 3 汎関数を用いて，以下を証明する．

命題 3.8. F_{ω^ω} は型 3 原始再帰的である．

Proof. このために, $\mathcal{G}(F_{+1})(F_\alpha)$ を計算してみよう．以後, 記法の簡易化のために $\mathcal{G}^*_n = \mathcal{G}^*(n, F_{+1})$ とおく．まず $\mathcal{G}^*(0, F_{+1}) = F_{+1}$ であるから,

$$\mathcal{G}^*(1, F_{+1})(F_\alpha) = F_{+\bullet\cdot\omega}(F_{+1})(F_\alpha) = F_{+\omega}(F_\alpha) = F_{\alpha+\omega}$$

となる．つまり，$\mathcal{G}^*_1(F_\alpha) = F_{\alpha+\omega}$ となるから，$\mathcal{G}^*$ は $F_{+\omega}$ の持つべき性質を満たす．したがって，命題 3.7 より，

$$\mathcal{G}^*(2, F_{+1})(F_\alpha) = F_{+\bullet\cdot\omega}(\mathcal{G}^*_1, F_\alpha) = F_{+\omega\cdot\omega}(F_\alpha) = F_{\alpha+\omega\cdot\omega} = F_{\alpha+\omega^2}$$

を得る．これを繰り返せば，$\mathcal{G}^*(n, F_{+1})(F_\alpha) = F_{\alpha+\omega^n}$ を導くことができる．以上より，$\mathcal{G}(F_{+1})(F_0)(x) = F_{\omega^x}(x) = F_{\omega^\omega[x]}(x) = F_{\omega^\omega}(x)$ が導かれた． □

上で定義した型 3 汎関数 $\mathcal{G}$ について，$\mathcal{G}(F_{+\beta})(F_\alpha) = F_{\alpha+\beta\cdot\omega^\omega}$ が成立するので，$\mathcal{G}$ のことを $F_{+\bullet\cdot\omega^\omega}$ と書くのは妥当であろう．このアイデアを一般化すると，以下のような型 3 汎関数 $\mathcal{G}_k = F_{+\bullet\cdot\omega^{\omega^k}} : [\mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}] \to [\mathbb{N}^\mathbb{N} \to \mathbb{N}^\mathbb{N}]$ を定義することができる：

$$\begin{cases} \mathcal{G}^*_{k+1}(0, F) = F, \\ \mathcal{G}^*_{k+1}(n+1, F) = \mathcal{G}_k(\mathcal{G}^*_{k+1}(n, F)), \\ \mathcal{G}_{k+1}(F) = \lambda f.\lambda x.\mathcal{G}^*_{k+1}(x, F)(f)(x). \end{cases}$$

ここで，$\omega^{\omega^0} = \omega^1 = \omega$ であるから，$\mathcal{G}_0 = F_{\bullet+\bullet\cdot\omega^{\omega^0}} = F_{\bullet+\bullet\cdot\omega}$ であることに注意する．これまでのように，$\mathcal{G}_{k+1} = F_{+\bullet\cdot\omega^{\omega^{k+1}}}$ の構成は以下のように書き直せる．

$$\mathcal{G}_{k+1}(F)(f)(x) = \left(\underbrace{\mathcal{G}_k \circ \mathcal{G}_k \circ \cdots \circ \mathcal{G}_k}_{x \text{ 個}}(F)\right)(f)(x).$$

命題 3.9. $F_{+\bullet\cdot\omega^{\omega^k}}(F_{+\beta})(F_\alpha) = F_{+\beta\cdot\omega^{\omega^k}}(F_\alpha) = F_{\alpha+\beta\cdot\omega^{\omega^k}}$ が成立する．

Proof. k 上の数学的帰納法による．$k = 0$ のときは命題 3.7 による．任意の適切な順序数 γ について，既に $\mathcal{G}_k(F_{+\gamma})(F_\alpha) = F_{\alpha+\gamma\cdot\omega^{\omega^k}}$ は証明されていると仮定しよう．

以後，記法の簡易化のために $\mathcal{G}^*_{k+1,n} = \mathcal{G}^*_{k+1}(n, F_{+\beta})$ とおく．まず $\mathcal{G}^*_{k+1}(0, F_{+\beta}) = F_{+\beta}$ であるから，帰納的仮定より，

$$\mathcal{G}^*_{k+1}(1, F_{+\beta})(F_\alpha) = \mathcal{G}_k(F_{+\beta})(F_\alpha) = F_{\alpha+\beta\cdot\omega^{\omega^k}}(x)$$

となる．つまり，$\mathcal{G}^*_{k+1,1}(F_\alpha) = F_{\alpha+\beta\cdot\omega^{\omega^k}}$ が成立する．言い換えれば，$\mathcal{G}^*_{k+1,1}$ は，$F_{+\beta\cdot\omega^{\omega^k}}$ としての条件を満たす．したがって，帰納的仮定に $\gamma = \beta\cdot\omega^{\omega^k}$ を適用すると，

$$\mathcal{G}^*_{k+1}(2, F_{+\beta})(F_\alpha) = \mathcal{G}_k(\mathcal{G}^*_1)(F_\alpha) = F_{\alpha+\beta\cdot\omega^{\omega^k}\cdot\omega^{\omega^k}}(x) = F_{\alpha+\beta\cdot\omega^{\omega^k}\cdot 2}(x)$$

を得る．これを繰り返せば，$\mathcal{G}^*_{k+1}(n, F_{+\beta})(F_\alpha) = F_{\alpha+\beta\cdot\omega^{\omega^k\cdot n}}$ を導くことができる．以上より，

$$\mathcal{G}_{k+1}(F_{+\beta})(F_\alpha)(x) = F_{\alpha+\beta\cdot\omega^{\omega^k\cdot x}}(x) = F_{\alpha+\beta\cdot\omega^{\omega^{k+1}[x]}}(x) = F_{\alpha+\beta\cdot\omega^{\omega^{k+1}}}(x)$$

が導かれた． □

定理 3.10. 任意の順序数 $\alpha < \omega^{\omega^\omega}$ に対し，急増加関数 $F_\alpha\colon \mathbb{N} \to \mathbb{N}$ は型 3 原始再帰的である．

Proof. 命題 3.5 より，順序数 $\alpha < \omega^{\omega^\omega}$ について，関数 F_α は，F_0 と型 2 汎関数 F_{+1} および型 3 汎関数 $F_{+_{\bullet\cdot\omega^{\omega^k}}}$ たちの合成によって表すことができる（具体な構成は，下の例 3.11 を参考にせよ）．F_0 は原始再帰的であり，F_{+1} は型 2 原始再帰的であり，そして $F_{+_{\bullet\cdot\omega^{\omega^k}}}$ が型 3 原始再帰的であることから，F_α も型 3 原始再帰的であることが従う． □

以上より，$\omega \uparrow\uparrow 3$ 未満の順序数 α について，急増加関数 F_α は型 3 原始再帰的であることが示された．同様の議論を繰り返せば，$\omega \uparrow\uparrow n$ 未満の順序数 α について，急増加関数 F_α が型 n 原始再帰的であることを示すことができる．たとえば，次のステップとしては，与えられた型 3 汎関数を有限回反復合成する型 4 汎関数 $\mathfrak{I}$ を考える．つまり，

$$\mathfrak{I}(\mathcal{F})(F)(f)(x) = \left(\underbrace{\mathcal{F}\circ\mathcal{F}\circ\cdots\circ\mathcal{F}}_{x\text{ 個}}(F)\right)(f)(x).$$

となる型 4 汎関数 $\mathfrak{I}$ を型 4 原始再帰によって定義し，これまでと同様の構成を進めればよい．これは，型 3 汎関数 $\mathcal{G}_k$ から $\mathcal{G}_{k+1}$ を得る原始再帰的構成を型 4 汎関数として表したものでもある．つまり，$\mathfrak{I}(\mathcal{G}_k) = \mathcal{G}_{k+1}$ が成立する．

一応，これがどのような順序数演算を与えるかだけ述べておこう．まず，$\mathfrak{I}(F_{+_{\bullet\cdot\gamma}})(F_{+\beta})(F_\alpha) \approx F_{\alpha+\beta\cdot\gamma^\omega}$，つまり $\mathfrak{I}(F_{+_{\bullet\cdot\gamma}}) \approx F_{+_{\bullet\cdot\gamma^\omega}}$ となることは予想が付くと思われる．したがって，この $\mathfrak{I}$ は $F_{+_{\bullet^\omega}}$ とでも呼ぶべき型 4 作用素と考えられる．以前と同様に，$\omega^{\omega^0} = \omega^1 = \omega$ であるから，$F_{+_{\bullet^{\omega^{\omega^0}}}}$ と書いてもよい．そして，型 4 原始再帰を用い，汎関数 $F_{+_{\bullet^{\omega^{\omega^k}}}}$ の累積として $F_{+_{\bullet^{\omega^{\omega^{k+1}}}}}$ を定義できる．この型 4 汎関数について，

$$F_{+_{\bullet^{\omega^{\omega^k}}}}(F_{+_{\bullet\cdot\omega}})(F_{+1})(F_0) \approx F_{0+1\cdot\omega^{\omega^{\omega^k}}} = F_{\omega^{\omega^{\omega^k}}}$$

となることが自然に想定される．この性質さえ示せば，任意の $\alpha < \omega \uparrow\uparrow 4$ について F_α が型 4 原始再帰的であることは直ちに導かれる．この発想をさらに高い型に持ち上げることにより，ε_0 未満のランクの急増加関数は，有限型の高階原始再帰的定義に用いる高階汎関数の型のランクによって階層分けできることが分かる．

以上の構成をまとめよう．ここまでにたくさんの高階汎関数を構成してきたと感じたかもしれない．しかし，実を言うと，**本質的にひとつしか高階汎関数を使っていない**ということに気づいただろうか．その汎関数とは，「与えられた型 n 汎関数を反復合成する型 $n+1$ 汎関数」である．ここ

までの議論を統一的に見るために，この型 $n+1$ 汎関数を $\mathtt{it}_{n+1}$ と書くことにしよう．たとえば，$\mathtt{it}_2 = F_{+1}$ である．本節の汎関数構成は，以下のように図示できる．

$$F_0 \overset{\mathtt{it}_2}{\longmapsto} \mathtt{it}_2(F_0) \overset{\mathtt{it}_2}{\longmapsto} \mathtt{it}_2 \circ \mathtt{it}_2(F_0) \overset{\mathtt{it}_2}{\longmapsto} \mathtt{it}_2 \circ \mathtt{it}_2 \circ \mathtt{it}_2(F_0) \overset{\mathtt{it}_2}{\longmapsto} \dots$$
$$\mathtt{it}_2 \overset{\mathtt{it}_3}{\longmapsto} \mathtt{it}_3(\mathtt{it}_2) \overset{\mathtt{it}_3}{\longmapsto} \mathtt{it}_3 \circ \mathtt{it}_3(\mathtt{it}_2) \overset{\mathtt{it}_3}{\longmapsto} \mathtt{it}_3 \circ \mathtt{it}_3 \circ \mathtt{it}_3(\mathtt{it}_2) \overset{\mathtt{it}_3}{\longmapsto} \dots$$
$$\mathtt{it}_3 \overset{\mathtt{it}_4}{\longmapsto} \mathtt{it}_4(\mathtt{it}_3) \overset{\mathtt{it}_4}{\longmapsto} \mathtt{it}_4 \circ \mathtt{it}_4(\mathtt{it}_3) \overset{\mathtt{it}_4}{\longmapsto} \mathtt{it}_4 \circ \mathtt{it}_4 \circ \mathtt{it}_4(\mathtt{it}_3) \overset{\mathtt{it}_4}{\longmapsto} \dots$$

一応，注意しておくと，$\mathtt{it}_{n+1}$ は「型 n 汎関数を反復合成して新たな型 n 汎関数を作る型 n 原始再帰的構成」を型 $n+1$ 汎関数として表したものであるから，$\mathtt{it}_{n+1}$ の単独の適用はあくまで型 n 原始再帰法によるものである．

さて，これらの単なる「反復合成」という手続きが急増加階層のランクをどれだけ持ち上げるかということを明示したのが，本節での議論であった．つまり，$\mathtt{it}_2 = F_{+1}$ であり，$\mathtt{it}_3 = F_{+\bullet\cdot\omega}$ であり，$\mathtt{it}_4 = F_{+\cdot\bullet^\omega}$ であった．したがって，上記の反復合成は，以下のように書き直せる．

$$F_0 \overset{F_{+1}}{\longmapsto} F_1 \overset{F_{+1}}{\longmapsto} F_2 \overset{F_{+1}}{\longmapsto} F_3 \overset{F_{+1}}{\longmapsto} \dots$$
$$F_{+1} \overset{F_{+\bullet\cdot\omega}}{\longmapsto} F_{+\omega} \overset{F_{+\bullet\cdot\omega}}{\longmapsto} F_{+\omega^2} \overset{F_{+\bullet\cdot\omega}}{\longmapsto} F_{+\omega^3} \overset{F_{+\bullet\cdot\omega}}{\longmapsto} \dots$$
$$F_{+\bullet\cdot\omega} \overset{F_{+\cdot\bullet^\omega}}{\longmapsto} F_{+\bullet\cdot\omega^\omega} \overset{F_{+\cdot\bullet^\omega}}{\longmapsto} F_{+\bullet\cdot\omega^{\omega^2}} \overset{F_{+\cdot\bullet^\omega}}{\longmapsto} F_{+\bullet\cdot\omega^{\omega^3}} \overset{F_{+\cdot\bullet^\omega}}{\longmapsto} \dots$$

結論を述べると，ε_0 までの超限再帰によって構成された急増加関数 F_α は，いずれも初期関数 $F_0 = \mathsf{succ}$ と有限型汎関数の反復合成演算 $\mathtt{it}_n$ という単純な基本演算の**有限個**の組み合わせによって作れる，ということである．以下で，いくつか具体的を見てみよう．

例 3.11. たとえば急増加関数 $F_{\omega^3\cdot 2+\omega\cdot 4+5}$ は，以下のように初期関数と反復合成演算の有限個の組み合わせによって書ける．

$$\begin{aligned}&\mathtt{it}_2^{(5)} \circ (\mathtt{it}_3(\mathtt{it}_2))^{(4)} \circ \left(\mathtt{it}_3^{(3)}(\mathtt{it}_2)\right)^{(2)} (\mathtt{succ})\\ =&F_{+1}^{(5)} \circ F_{+\omega}^{(4)} \circ F_{+\omega^3}^{(2)}(F_0) = F_{\omega^3\cdot 2+\omega\cdot 4+5}.\end{aligned}$$

もう少し複雑な例としては，たとえば $\omega^{\omega^3\cdot 2+4} + \omega^{\omega^2}\cdot 7 + 6$ ランクの急増加関数を具体的に高階汎関数の組み合わせによって記述してみよう．まずは最初の項は，以下のように表せる．

$$\mathtt{it}_3^{(4)} \circ \left(\mathtt{it}_4^{(3)}(\mathtt{it}_3)\right)^{(2)} (\mathtt{it}_2) = F_{+\bullet\cdot\omega}^{(4)} \circ F_{+\bullet\cdot\omega^{\omega^3}}^{(2)}(F_{+1}) = F_{+(\omega^{\omega^3})^2\cdot\omega^4} = F_{+\omega^{\omega^3\cdot 2+4}}.$$

次の項については，以下の通りである．

$$\left(\mathtt{it}_4^{(2)}(\mathtt{it}_3)(\mathtt{it}_2)\right)^{(7)} = \left(F_{+\bullet\cdot\omega^{\omega^2}}(F_{+1})\right)^{(7)} = \left(F_{+\omega^{\omega^2}}\right)^{(7)} = F_{+\omega^{\omega^2}\cdot 7}.$$

以上をまとめると，急増加関数 $F_{\omega^{\omega^3\cdot 2+4}+\omega^{\omega^2}\cdot 7+6}$ は以下のように有限的に書けることが分かる．

$$\mathtt{it}_2^{(6)} \circ \left(\mathtt{it}_4^{(2)}(\mathtt{it}_3)(\mathtt{it}_2)\right)^{(7)} \circ \left(\mathtt{it}_3^{(4)} \circ \left(\mathtt{it}_4^{(3)}(\mathtt{it}_3)\right)^{(2)} (\mathtt{it}_2)\right) (\mathtt{succ}).$$

■ **Historical Remark.** 歴史的には，これは 1925 年のミュンスター会議においてヒルベルトの報告したプロジェクトと関係しているようだ．つまり，第 1 ステップとして「自然数上の関数の構成から超限再帰を除去して，高階汎関数を用いた有限的な再帰に置き換える」ということ，そして，第 2 ステップとして「再帰的定義に必要な型のランクによって自然数上の関数を分類する」というものである．ただし，ヒルベルトは，有限型の汎関数に留まらず，超限順序数型の高階汎関数を主な考察対象としている．どうやら，超限型の汎関数による再帰まで考えることによって，実質的に，自然数上のすべての関数の分類が可能であると考えていたようである．このように自然数上の関数とそれを再帰的定義するのに必要な型という順序数を対応付けることによって，自然数上の関数の個数を数え上げる—それが当時ヒルベルトの思い描いていた「連続体仮説」を解くための戦略であった．

3.4 有限型原始再帰の限界と ε_0

前節では，ε_0 未満のランクの急増加関数が有限型原始再帰的であることを示した．本節では，この ε_0 が限界であること，つまり有限型原始再帰関数は ε_0 の壁を越えられないことを示そう．このためには，任意の有限型原始再帰を ε_0 未満の整礎原始再帰によってシミュレートできることを示せばよい．しかし，ε_0-原始再帰はあくまで自然数上の再帰なので，有限型原始再帰を自然数上の何らかの再帰に還元する必要がある．そこで重要なことは，有限型原始再帰関数の構成手続きは，いずれも有限長の記号列で記述できるという点である．そして，有限長の記号列は 1 つの自然数でコードすることができる．これによって，

「ある関数を元にして新しい関数を構成する」

という手続き（合成，関数適用，λ-抽象，原始再帰法など）を

「関数のコードを入力として新しい関数のコードを出力する」

という自然数上の（原始再帰的）関数に置き換えるのである．このようにして，有限型原始再帰関数の構成の各ステップは $\mathbb{N}$ 上の原始再帰に還元される．しかし，これは 1 ステップの原始再帰性であるから，次に保証しなければならないことは，再帰の深さが ε_0 未満であるという点である．

最終的に，自然数上の関数，すなわち型 1 の関数に対して，「深度」と呼ぶ順序数を割り当てたい．この過程で，型 2 汎関数には，順序数から順序数への関数，つまり順序数を基礎型とする型 1 関数が「深度」として割り当てられる．一般に，型 $n+1$ 汎関数の「深度」は，順序数を基礎型とする型 n 汎関数となる．実際に，まずは具体例で，この深度について考えてみよう．

例 3.12. 1. 型 1 汎関数を反復合成する型 2 汎関数 $\mathtt{it}_2$ を思い出そう．深度 α の関数 f を反復合成することを考える．$\mathtt{it}_2^*(0,f,x)=f(x)$ を求めるには，深度 α の f に自然数 x を適用する 1 ステップが加わるので，深度 $\alpha+1$ に達する．これを繰り返し，$\mathtt{it}_2^*(n,f,x)=f^{(n)}(x)$ の深度は $\alpha+n$ となる．よって，$\mathtt{it}_2(f)(x)=f^{(x)}(x)$ は深度 $\alpha+\omega$ の再帰で求められる．以上より，$\mathtt{it}_2$ の深度は $\alpha\mapsto\alpha+\omega$ と考える．

2. 型 2 汎関数を反復合成する型 3 汎関数 $\mathtt{it}_3$ を思い出そう．深度 $\xi \mapsto \beta(\xi)$ の型 2 汎関数 F を反復合成し，深度 α の関数 f を適用することを考える．$\mathtt{it}_3^*(0, F, f) = F(f)$ を求めるには，深度 $\xi \mapsto \beta(\xi)$ の F に深度 α の f を適用した $F(f)$ の深度は $\beta(\alpha)$ である．これを繰り返し，$\mathtt{it}_3^*(n, F, f) = F^{(n)}(f)$ の深度は $\beta^{(n)}(\alpha)$ となる．よって，$\mathtt{it}_3(F)(f)(x) = F^{(x)}(f)(x)$ は深度 $\sup_n \beta^{(n)}(\alpha)+2$ の再帰で求められる．以上より，$\mathtt{it}_3$ の深度は $(\alpha, \beta) \mapsto \sup_n \beta^{(n)}(\alpha)+2$ と考える．したがって，たとえば $\mathtt{it}_3 \circ \mathtt{it}_2$ の深度はおよそ $\alpha \mapsto \alpha + \omega + \omega + \omega + \ldots$ の上限なので $\alpha \mapsto \alpha + \omega^2$ であり，$\mathtt{it}_3 \circ \mathtt{it}_3 \circ \mathtt{it}_2$ の深度はおよそ $\alpha \mapsto \alpha + \omega^2 + \omega^2 + \omega^2 + \ldots$ なので $\alpha \mapsto \alpha + \omega^3$ となる．

一般に，各型の自由変数の深度は 0 とし，合成や関数適用後の深度は，対応する関数の深度の合成や関数適用によって得る．また，λ-抽象 $\lambda x.t$ の深度は t の深度を $+1$ したものとする．原始再帰法では，初期値 $f_0 = g$ から，h を繰り返し適用して f_n から f_{n+1} を作り，最終的に f を得るが，f_n までの深度は上記の方法で求め，f の深度はその上限によって与えられることとする．

それでは，構成中に型 n 以下の高階汎関数しか伴わないような有限型原始再帰関数の深度が $\omega \uparrow\uparrow n$ 未満であることを示そう．

定理 3.13. 型 2 原始再帰的な型 1 関数の深度は ω^ω 未満である．

Proof. 型 2 原始再帰的な型 2 汎関数の深度は，ある $\theta < \omega^\omega$ について，$\alpha \mapsto \alpha + \theta$ より劣ることを示す．関数の構成に関する帰納法による．関数の合成によって関数を構成した場合，$\theta_1, \theta_2 < \omega^\omega$ ならば $\theta_1 + \theta_2 < \omega^\omega$ であることから主張は導かれる．型 2 原始再帰法の場合も，深度 $+\theta_1 < \omega^\omega$ の型 2 汎関数 G を起点にして，深度 $+\theta_2 < \omega^\omega$ の型 2 汎関数 H の適用によって型 2 汎関数 F を構成した場合を考える．ある $\ell < \omega$ について $\theta_1, \theta_2 < \omega^\ell$ となることから，汎関数 F の深度はおよそ $+\theta_1 + \theta_2 \cdot \omega < \omega^\ell \cdot \omega = \omega^{\ell+1} < \omega^\omega$ を得る． □

定理 3.14. 型 3 原始再帰的な型 1 関数の深度は ω^{ω^ω} 未満である．

Proof. 型 3 原始再帰的な型 3 汎関数の深度は，ある $\theta < \omega^{\omega^\omega}$ について，$(\alpha, +\beta) \mapsto \alpha + \beta \cdot \theta$ より劣ることを示す．関数の構成に関する帰納法による．関数の合成によって関数を構成した場合，$\theta_1, \theta_2 < \omega^{\omega^\omega}$ ならば $\theta_1 \cdot \theta_2 < \omega^{\omega^\omega}$ であることから主張は導かれる．型 3 原始再帰法の場合も，深度 $\bullet \cdot \theta_1 < \omega^{\omega^\omega}$ の型 3 汎関数を起点にして，深度 $\bullet \cdot \theta_2 < \omega^{\omega^\omega}$ の型 3 汎関数の適用によって型 3 汎関数 $\mathcal{F}$ を構成した場合を考える．ある $\ell < \omega$ について $\theta_1, \theta_2 < \omega^{\omega^\ell}$ となることから，汎関数 $\mathcal{F}$ の深度はおよそ $\bullet \cdot \theta_1 \cdot \theta_2^\omega < (\omega^{\omega^\ell})^\omega = \omega^{\omega^{\ell+1}} < \omega^{\omega^\omega}$ を得る． □

同様の方法によって，一般に，型 n 原始再帰的な型 1 関数の深度は $\omega \uparrow\uparrow n$ 未満となることが示せる．あとはコーディング手続きを厳密に形式化することによって，任意の有限型原始再帰的な型

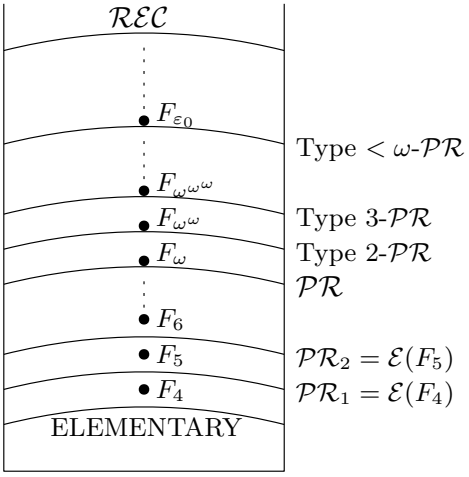

図 3　有限型原始再帰関数と ε_0-原始再帰

1 関数は，ある $\alpha < \varepsilon_0$ について α-原始再帰的であることが示せる．ここまでの議論をまとめると，以下のような計算可能関数の階層を得られたこととなる．

$$\mathcal{E} = \mathcal{PR}_0 \subsetneq \mathcal{E}(\uparrow\uparrow) = \mathcal{PR}_1 \subsetneq \cdots \subsetneq \mathcal{E}(\uparrow^{n+1}) = \mathcal{PR}_n \subsetneq \mathcal{E}(\uparrow^{n+2}) = \mathcal{PR}_{n+1} \subsetneq \ldots$$
$$\cdots \subsetneq \mathcal{PR} = \bigcup_{n\in\mathbb{N}} \mathcal{PR}_n \subsetneq \text{Type 2-}\mathcal{PR} \subsetneq \text{Type 3-}\mathcal{PR} \subsetneq \ldots$$
$$\cdots \subsetneq \text{Type } n\text{-}\mathcal{PR} \subsetneq \text{Type } (n+1)\text{-}\mathcal{PR} \subsetneq \cdots \subsetneq \text{Type} < \omega\text{-}\mathcal{PR} = \mathcal{REC}_{<\varepsilon_0} \subsetneq \cdots \subsetneq \mathcal{REC}.$$

ここで $\mathcal{REC}_{<\varepsilon_0}$ は ε_0 未満の θ について θ-原始再帰的であるような関数全体のクラスを表す．

4　ゲーデルのダイアレクティカ解釈

最後に，高階の原始再帰と関わる数学基礎論の話題として，ゲーデルのダイアレクティカ解釈を紹介しよう．これは，自然数論における論理式の正しさを，高階原始再帰関数によって保証できるかを議論するものである．形式的には，ゲーデルは，有限型の原始再帰汎関数の理論である**体系** T (*System* T) を導入し，これによって，自然数論の形式化として知られる一階**ペアノ算術** (*Peano arithmetic*) の無矛盾性を有限型原始再帰の体系 T のそれへと還元したのである．

この具体的な方法を解説する．算術の論理式 φ が与えられたとき，これを帰納的に，有限型汎関数パラメータの列 $\bar{g}, \bar{x}$ を含む**量化なし**論理式 $\varphi_D(\bar{g} \mid \bar{x})$ に変形しよう．ここで，$\bar{g}$ と $\bar{x}$ の間の縦線は，パラメータの列 $\bar{g}$ と $\bar{x}$ の役割が異なることを強調するためだけのものであり，特に数学的な意味はない．

定義 4.1 (ゲーデル)**.** 算術の論理式 φ に対し，有限型の汎関数の列を自由変数として持つ量化なし論理式 φ_D を以下のように定義する．まず，φ が原始論理式ならば，$\varphi_D \equiv \varphi$ とする．もし φ

が原始論理式でないならば，φ_D を以下のように帰納的に定義する．

$$
\begin{aligned}
(\varphi \wedge \psi)_D(\bar{g}, \bar{h} \mid \bar{x}, \bar{y}) &\equiv \varphi_D(\bar{g} \mid \bar{x}) \wedge \psi_D(\bar{h} \mid \bar{y}) \\
(\varphi \vee \psi)_D(i, \bar{g}, \bar{h} \mid \bar{x}, \bar{y}) &\equiv [i = 0 \wedge \varphi_D(\bar{g} \mid \bar{x})] \vee [i = 1 \wedge \psi_D(\bar{h} \mid \bar{y})] \\
(\forall z \varphi(z))_D(f \mid a, \bar{x}) &\equiv \varphi(a)_D(f(a) \mid \bar{x}) \\
(\exists z \varphi(z))_D(a, \bar{g} \mid \bar{x}) &\equiv \varphi(a)_D(\bar{g} \mid \bar{x}) \\
(\varphi \to \psi)_D(h, k \mid \bar{g}, \bar{x}) &\equiv [\varphi_D(\bar{g} \mid h(\bar{g}, \bar{x})) \to \psi_D(k(\bar{g}) \mid \bar{x})]
\end{aligned}
$$

このようにして，φ から量化なし論理式 $\varphi_D(\bar{g} \mid \bar{x})$ を作る．高階汎関数の列 $\bar{g}$ が任意の $\bar{x}$ に対して $\varphi_D(\bar{g} \mid \bar{x})$ を満たすとき，$\bar{g}$ によって φ の正しさが保証されていると考える．このように，高階汎関数を用いた φ の正しさの保証を表す論理式

$$\varphi^D \equiv \exists \bar{g} \forall \bar{x} \varphi_D(\bar{g} \mid \bar{x})$$

を φ の**ダイアレクティカ解釈** (*Dialectica interpretation*) と呼ぶ．

注意. 論理記号 $\to$ の部分だけ意味が不明瞭かもしれないので，これを説明しよう．まず，h が $\bar{g}$ に依存しないような場合は，「φ_D の解を知っていれば ψ_D の解も得られる」あるいは「ψ_D の解を見つける問題を φ_D の解を見つける問題に還元している」と理解できる．つまり，問題設定としては，$\bar{x}$ が与えられているとき，ψ_D の $\bar{x}$-成分 $\psi_D(* \mid \bar{x})$ の解を見つけたい．このために，φ_D を解く神託に，ひとつの成分の解が何であるかを尋ねることができる．今回の場合には，φ_D の $h(\bar{x})$-成分 $\varphi_D(* \mid h(\bar{x}))$ の解 $\bar{g}$ が神託から与えられている．この解 $\bar{g}$ を用いることにより，$\psi_D(* \mid \bar{x})$ の解 $k(\bar{g})$ を見つけることができる．つまり，$(\varphi \to \psi)_D$ の解は，神託へのクエリを作る汎関数 h と，φ_D の解を元に ψ_D の解を作る汎関数 k の対である[*6]．

しかし，実際には汎関数 h は変数として $\bar{g}$ を取ることもできる．この場合は，つまり，与えられた $\bar{g}$ から $k(\bar{g})$ を作るのであるが，$\bar{g}$ が φ_D の正しい解ならば $k(\bar{g})$ は ψ_D の正しい解であってほしい．この対偶を取れば，

もし $k(\bar{g})$ が ψ_D の解でないならば，$\bar{g}$ は φ_D の解ではない．

もう少し細かく見ると，$k(\bar{g})$ が ψ_D の解でないということは，正確には，$k(\bar{g})$ が解とならない x-成分 $\psi_D(* \mid x)$ が存在するということである．このとき，$\bar{g}$ が解とならない成分が存在するので，x と $\bar{g}$ の情報を利用して，その反例を見つけてくるのが h である．つまり，$\bar{g}$ は $h(\bar{g}, \bar{x})$-成分 $\varphi_D(* \mid h(\bar{g}, \bar{x}))$ の解にならない．このように，$(\varphi \to \psi)_D$ の解は，φ_D の解の候補 $\bar{g}$ を元に ψ_D の解の候補 $k(\bar{g})$ を作る汎関数 k と，$k(\bar{g})$ の ψ_D-反例から $\bar{g}$ の φ_D-反例を作る汎関数 h の対である．

例 4.2 (数学的帰納法のダイアレクティカ解釈)**.** 上記のようにして，複雑な量化の入れ子構造を持

[*6] 計算可能解析学では，解の探索問題の還元にこれと類似のアイデアを用いており，これを**強 Weihrauch 還元** (*strong Weihrauch reduction*) と呼ぶ．また，計算量理論における探索問題の間の Levin 還元も，Weihrauch 還元のアイデアに比較的近い．

ち得る算術的論理式は，有限型汎関数を変数として持つ**量化なし**論理式に翻訳される．この解釈の下で，数学的帰納法の正当性は，有限型の原始再帰に還元されるということを見てみよう．数学的帰納法は，形式的には以下の論理式によって表される．

$$[\varphi(0) \wedge \forall n(\varphi(n) \to \varphi(n+1))] \to \forall n\varphi(n).$$

論理式 φ は大量の量化記号を含み得ることに注意する．いま，数学的帰納法の前提の正しさがダイアレクティカ解釈によって保証されていると仮定する．つまり，ある $\bar{f}$ について，任意の $\bar{x}$ で $(\varphi(0))_D(\bar{f} \mid \bar{x})$ が成立しており，さらに，ある h, k について，任意の $\bar{y}$ で以下が成立している．

$$(\varphi(n))_D(\bar{g} \mid h(n, \bar{g}, \bar{y})) \to (\varphi(n+1))_D(k(n, \bar{g}) \mid \bar{y}).$$

以下のような高階原始再帰で，$(\forall n\varphi(n))_D$ の解を見つけられる．

$$\begin{cases} K(0) = \bar{f} \\ K(n+1) = k(n, K(n)) \end{cases}$$

これが実際に $(\forall n\varphi(n))_D$ の解となっていることを，**量化なし**論理式に対する数学的帰納法によって保証しよう．いま，K の定義より，

$$(\varphi(0))_D(K(0) \mid \bar{x}) \ \wedge \ \big[(\varphi(n))_D(K(n) \mid h(n, K(n), \bar{y})) \to (\varphi(n+1))_D(K(n+1) \mid \bar{y})\big]$$

が成立している．与えられた $m \in \mathbb{N}$ について，高階原始再帰によって以下のように H を定義する．

$$\begin{cases} H(0, \bar{y}) = \bar{y} \\ H(n+1, \bar{y}) = h(m \dot{-} n, K(m \dot{-} n), H(n, \bar{y})) \end{cases}$$

これを利用すると，**量化なし**論理式に対する数学的帰納法によって，式 $(\varphi(m))_D(K(m) \mid \bar{y})$ の正しさを $(\varphi(0))_D(K(0) \mid H(m, \bar{y}))$ の正しさに還元できる．後者の式は，前提によって保証されているから，以上により数学的帰納法の帰結のダイアレクティカ解釈 $(\forall n\varphi(n))_D$ が正しいことが確認された．

以上より，数学的帰納法の正当性が有限型原始再帰関数によって保証されることが確認できた．より一般に，直観主義論理上の一階算術として知られる**ハイティング算術** (*Heyting arithmetic*) がある算術的文を証明するならば，ダイアレクティカ解釈より，有限型原始再帰関数がその正しさを保証してくれることを証明できる．これは，無矛盾性証明の観点からは，重要な帰結を持つ．もしハイティング算術が矛盾するならば，ハイティング算術による矛盾の証明の正しさを有限型原始再帰関数が保証してしまう．これを言い換えれば，ハイティング算術が矛盾するならば，有限型原始再帰算術の体系 T も矛盾している，ということに他ならない．つまり，ゲーデルのダイアレクティカ解釈の帰結として，

有限型原始再帰の体系 T が無矛盾ならば，ハイティング算術も無矛盾である

という定理が得られる．一方で，ゲーデル-ゲンツェンの二重否定解釈というものを用いると，ペアノ算術の無矛盾性をハイティング算術の無矛盾性へと容易に還元できる．したがって，上の定理

は，実際には，「有限型原始再帰の体系 T が無矛盾ならば，ペアノ算術も無矛盾である」ということを導く．

ペアノ算術やハイティング算術が公理として認めている**数学的帰納法**は，とてつもなく複雑な量化の入れ子構造を伴う論理式に対しても適用可能であった．たった 1 つの量化記号ですら，計算不可能な関数や述語を容易に定義できてしまうし，複数の量化記号を用いると想像を絶するような状況が起こるかもしれない．このように，とてつもなく複雑な量化を伴った数学的帰納法の無矛盾性を，ダイアレクティカ解釈によって，有限型の高階原始再帰（あるいは現代的な観点からは，原始的な高階プログラミングと呼べるものかもしれない）という，量化のない世界に還元した．

さらに，第 3.4 節の議論によれば，有限型の原始再帰は，ε_0 未満の順序型の整礎原始再帰によってシミュレートできる．ここで注意すれば，順序数とは，整礎性というある種の有限性を持つ順序構造（の同値類）に過ぎず，そういう意味では ε_0 などは自然数概念を超えるものではないとも言える．具体的には，ε_0 未満の順序型は，あくまで $\mathbb{N}$ 上の辞書式順序に過ぎない．つまり，ε_0 未満の順序型は，極めて単純な算術的論理式によって定義できる $\mathbb{N}$ 上の（原始再帰的）関係である．また，有限型の原始再帰を ε_0 未満の順序型の整礎原始再帰へ還元する過程において，本質的な量化は一切現れない．つまるところ，ペアノ算術という極めて複雑な体系の無矛盾性は，最終的に，本質的な量化を伴わない ε_0 上の整礎帰納法という比較的単純な操作によって保証されるのである．

$$\boxed{\text{ペアノ算術の無矛盾性}} \Longleftarrow \boxed{\text{有限型原始再帰算術 } T \text{ の無矛盾性}} \Longleftarrow \boxed{\varepsilon_0 \text{ の整礎性}}$$

量化記号という概念が如何に複雑怪奇なものであるか，ということをペアノ算術を階層化することによっても理解できる．ペアノ算術は，任意の算術的論理式に対する数学的帰納法を公理として保有していた．算術的論理式のうち，量化記号を高々 n 個しか用いず，その先頭の量化記号が $\exists$ である式を Σ_n-論理式と呼ぶ．体系 $\mathrm{I}\Sigma_n$ とは，ペアノ算術における数学的帰納法の公理を Σ_n-論理式に制限したものである．

$$\mathrm{I}\Sigma_1 \subset \mathrm{I}\Sigma_2 \subset \mathrm{I}\Sigma_3 \subset \cdots \subset \mathrm{I}\Sigma_n \subset \mathrm{I}\Sigma_{n+1} \subset \cdots \subset \text{ペアノ算術}$$

体系 $\mathrm{I}\Sigma_n$ の無矛盾性証明のためには，$\omega \uparrow\uparrow (n+1)$ の整礎性があれば十分であることが知られている．一方で，$\mathrm{I}\Sigma_n$ は，$\omega \uparrow\uparrow (n+1)$ 未満の任意の順序数の整礎性を証明できる．したがって，体系 $\mathrm{I}\Sigma_{n+1}$ によって $\mathrm{I}\Sigma_n$ の無矛盾性を証明できる．ここでゲーデルの第 2 不完全性定理より，$\mathrm{I}\Sigma_n$ が無矛盾ならば，$\mathrm{I}\Sigma_n$ によって $\mathrm{I}\Sigma_n$ の無矛盾性は証明できないことに注意しよう．これによって，$(n+1)$ 量化論理式に対する数学的帰納法の体系 $\mathrm{I}\Sigma_{n+1}$ が n 量化論理式に対する数学的帰納法の体系 $\mathrm{I}\Sigma_n$ より遥かに強いことが理解できる．このようにして，数学的帰納法を適用する式に現れる量化記号の数によって，その複雑さは著しく変化するのである．

5　参考文献

以下，本稿を執筆するにあたって参考にした主な文献を紹介する．

第 1 節の内容については，Odifreddi の “Classical Recursion Theory” 第 2 巻 [3] にまとまっ

ている．第 1.1 節に書かれている内容は任意のテキストに載っているが，執筆にあたって特に参考にした文献は存在しない．和書としては篠田の教科書 [9] がある．第 1.2 節については，主に Odifreddi 本 [3] の第 VIII.7 節と VIII.8 節を参考にしている．Rose の教科書 [4] も参考になると思われる．

第 2 節の内容について，まず第 2.1 節については，van Heijenoort [5] に収録されているヒルベルトの原論文の英訳 “On the infinite (1925)” とアッカーマンの原論文の英訳 “On Hilbert’s construction of the real numbers (1928)” を参考にしている．第 2.2 節の執筆にあたって参考にした文献はないが，田中編著「数学基礎論講義 [7]」に本質的に同じ内容が書かれていることに気づいたため，弱グッドスタイン列という名称はそこから借用した．第 2.3 節は，スーダンの仏語の原論文 “Sur le nombre transfini ω^ω (1927)” を参考にしている．

第 3 節と多少関係あるテーマは，Odifreddi 本 [3] の第 VIII.9 節や Rose の教科書 [4] に載っている．第 3.1 節の内容については，それ以外にも多くのテキストで触れられている．第 3.2 節と第 3.3 節の内容について書かれた文献は，和文でも英文でも見つけることができなかった[*7]．これは，$< \varepsilon_0$-再帰関数が有限型の高階原始再帰法で定義できることを，通常はダイアレクティカ解釈を経由して間接的に証明するためであると思われる．このため，第 3.2 節と第 3.3 節のような直接証明を試みた人が少ないのかもしれない．しかし，ダイアレクティカ解釈を経由しないことによって，$< \varepsilon_0$-再帰関数と PA-可証全域再帰関数との同値性を経由する必要がなくなり，$< \varepsilon_0$-再帰関数が有限型の高階原始再帰によってどのように構成されるかが具体的に明示されるという恩恵がある．そういうわけで，第 3.2 節と第 3.3 節のような直接証明にはそれなりにメリットがあると思われる．第 3.4 節のコーディングの厳密な議論は，たとえば Rose 本 [4] にも載っているので，そちらを参考にしてほしい．

第 4 節のダイアレクティカ解釈については，証明論ハンドブック内の Avigad-Feferman の記事 [1] が主な参考文献である．和文であれば，田中編「ゲーデルと 20 世紀の論理学」の第 3 巻「不完全性定理と算術の体系 [6]」内の白旗氏の記事が詳しい．

最後に，筆者による数学セミナーの記事 [8] では，本稿で触れなかった整列擬順序 (WQO) の理論の観点からのアッカーマン関数や急増加関数の役割についてなども議論している．ぜひ本稿の補完的な記事として活用してほしい．

参考文献

[1] Jeremy Avigad and Solomon Feferman. Gödel’s functional (“Dialectica”) interpretation. In *Handbook of proof theory*, volume 137 of *Stud. Logic Found. Math.*, pages 337–405. North-Holland, Amsterdam, 1998.

[2] Cristian Calude. *Theories of computational complexity*, volume 35 of *Annals of Discrete*

[*7] さほど入念に探したわけではないので，もしこの辺の直接証明が書かれた文献をご存知の方がいたら教えてください．

Mathematics. North-Holland Publishing Co., Amsterdam, 1988.

[3] P. G. Odifreddi. *Classical recursion theory. Vol. II*, volume 143 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1999.

[4] H. E. Rose. *Subrecursion: functions and hierarchies*, volume 9 of *Oxford Logic Guides*. The Clarendon Press, Oxford University Press, New York, 1984.

[5] Jean van Heijenoort, editor. *From Frege to Gödel*. Harvard University Press, Cambridge, MA, 2002. A source book in mathematical logic, 1879–1931, Reprint of the third printing of the 1967 original.

[6] 田中 一之. **ゲーデルと *20* 世紀の論理学 *3*　不完全性定理と算術の体系**. 東京大学出版会, 2007.

[7] 田中 一之, 鹿島 亮, 角田 法也, and 菊池 誠. **数学基礎論講義—不完全性定理とその発展**. 日本評論社, 1997.

[8] 木原 貴行. アッカーマン関数とヒルベルト. In **数学セミナー *2019* 年 *7* 月号**, pages 22–27. 日本評論社, 2019.

[9] 篠田 寿一. **帰納的関数と述語**. 数学基礎論シリーズ. 河合文化教育研究所, 1997.