原始帰納的函数とアッカーマン函数

藤田 博司

2012 年 2 月 16 日*

概要

アッカーマン函数は原始帰納的でない帰納的函数の例としてよく知られています. ところが, 原始帰納的でないことの証明の載っている (日本語の) 本が, もともと多くないうえどれも近年おいそれと手に入りにくい状況のようですので, ここに証明を書いて公開します.

1 自然数と帰納的定義

本稿では自然数とは 0 以上の整数 0, 1, 2, ... のことだとします. ここでは自然数の組を引数とし自然数の値を返す函数ばかりを扱います. のっけからあまり厳密にやるとアイディアが伝わらないかもしれないので, 最初はおおざっぱにいきますが, **帰納的定義**というのはたとえば階乗 $x! = 1 \times 2 \times \cdots \times x$ を

$$\begin{cases} 0! = 1 \\ (x{+}1)! = (x+1) \times x! \end{cases}$$

と定義するように, 漸化式の形で函数を定義するものです. この式では定義されるべき $x!$ という式が定義の右辺に登場していて, 話が循環しているように見えるのですが,

$$\begin{gathered} 0! = 1, \\ 1! = 1 \times 0! = 1 \times 1 = 1, \\ 2! = 2 \times 1! = 2 \times 1 = 2, \\ 3! = 3 \times 2! = 3 \times 2 = 6, \\ 4! = 4 \times 3! = 4 \times 6 = 24, \\ 5! = 5 \times 4! = 5 \times 24 = 120, \\ \vdots \end{gathered}$$

という具合に, 値を次々に計算していけるので, すべての自然数に対してちゃんと値が定まります. 自然数というものがまずもって数学的帰納法によって特徴づけられるので, 自然数の函数を定めるには, 帰納的定義がまことに具合いがよいのです.

学校の算数では自然数の加法を「寄せ算」でやったあと, 乗法を足し算の繰り返しとして導入します. a を b 個足したものが ab だよ, というわけですが, これも自然数の理論をちゃんと定式化しようとすると, 漸化式

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

* 最終更新: 2012 年 3 月 2 日金曜日, 補題 6 の証明を修正しました.

をみたす函数 f がただ一つ存在するので $f(x,y)=y\cdot x$ と定める, とやります.[*1] それどころか, ここで利用された足し算でさえも, もっと基本的な函数から帰納的定義によって構成されるものと考えるのです.

そこのところをちゃんと書いてみましょう. 自然数論は, もっとも基本的な要素として, 定数 0 と, "次の自然数" を返す函数 $S(\ )$ を考えて, これらについては次にいう**ペアノの公準**が成立しているものと仮定します. それ以外のこと, たとえば 0 とは具体的に何であるか, とか, 神が自然数を造ったのかどうか, とかいった一切のことは捨象して (カッコに入れて) 考えないことにします. ペアノの公準は次の 5 つです.

(P1) 0 は自然数である.
(P2) x が自然数のとき $S(x)$ も自然数である.
(P3) x と y が 異なる自然数のとき $S(x)$ と $S(y)$ も異なる自然数である.
(P4) どんな自然数 x についても $S(x)$ は 0 とは異なる.
(P5) 自然数についての命題 $P(x)$ について, $P(0)$ であることと $P(x)$ から $P(S(x))$ が導かれることを確かめたなら, すべての自然数 x について $P(x)$ が成立することを結論してよい.

このペアノの公準のうち (P1) から (P4) までは, $S(\ )$ が自然数全体から自然数全体への, "全射でない単射" であることと, 定義域にあって値域にない要素としてとくに 0 をとったことを意味しています. (P5) はいわゆる数学的帰納法の原理で, これは

$$0, S(0), S(S(0)), S(S(S(0))), \dots$$

という一連のものが, 自然数のすべてであることを保証しようとしています. そこでひとまず

$$\mathbb{N}=\{\,0,\ S(0),\ S(S(0)),\ S(S(S(0))),\ \dots\,\}$$

とおいて, $\mathbb{N}$ の要素を自然数と呼ぶことにしましょう.

意味としてはもちろん $S(0)$ とは 1 のことだし, $S(x)$ とは $x+1$ のことですが, なにしろペアノの公準にもとづいてこれから自然数論のすべてを構築しようという段階では "1 を足す" ということ自体がまだ定義されていません. そうしたことすべては, 次の帰納的に定義された函数の存在と一意性 (定理 0) によって保証されるわけです.[*2]

定理 0. 函数 $f:\mathbb{N}^{\nu}\to\mathbb{N}$ と $g:\mathbb{N}^{\nu+2}\to\mathbb{N}$ が与えられているとき,

$$\begin{cases} h(0,y_1,\dots,y_\nu)=f(y_1,\dots,y_\nu) \\ h(S(x),y_1,\dots,y_\nu)=g(x,h(x,y_1,\dots,y_\nu),y_1,\dots,y_\nu) \end{cases}$$

をすべての $x,y_1,\dots,y_\nu\in\mathbb{N}$ についてみたす函数 $h:\mathbb{N}^{\nu+1}\to\mathbb{N}$ が, ただひとつ存在する. □

既知の函数 f と g からこの図式によって得られた函数 h のことを, f と g から**帰納的に定義された**函数, というわけです. (ただし, 最近では**再帰的に定義された**という言い方のほうが通りがよいかもしれません.)

[*1] ここでもっと素直に $f(x,y)=x\cdot y$ としてもいいのですが, この漸化式ではどうも y が "掛けられる数" になっているので日本風に y を左因子にしました. これは決してわたくしが「2 かける 8 ならタコ足 2 本」派だからではありません. 掛け算の定義を順序数にまで拡張したときには, この漸化式によって定義される積 $f(x,y)$ はまさに $y\cdot x$ でなければなりません. それが $f(x,y)=x\cdot y$ という式がわたくしには気持ち悪く感じられる理由です.

[*2] なお, 「その "定理 0" というたときの 0 と $\mathbb{N}$ の要素の 0 は同じものですか」とか「なんで y の添字が 1 から始まるのん?自然数は 0 からやないの?」とか「足し算もこれから定義する言うてるときに "$\nu+2$ 変数函数" とは何ちゅうこっちゃ」いうようなご質問・コメントには, ここではお答えできません. ごめんなさい.

さて, もっとも基本的な函数としての定数 0 と**後者函数** $S(\ )$ から出発することにより, 足し算は

$$\begin{cases} \mathrm{add}(0,y) = y \\ \mathrm{add}(S(x),y) = S\big(\mathrm{add}(x,y)\big) \end{cases}$$

と定義される函数 $\mathrm{add}(x,y)$ だ, ということになります.[*3] このように $\mathrm{add}(x,y)$ が定義されたので, 掛け算もあらためて,

$$\begin{cases} \mathrm{prod}(0,y) = 0 \\ \mathrm{prod}(S(x),y) = \mathrm{add}\big(y,\mathrm{prod}(x,y)\big) \end{cases}$$

と定義できることになります. 冪乗や階乗もそれぞれ

$$\begin{cases} \mathrm{pow}(0,y) = S(0) \\ \mathrm{pow}(S(x),y) = \mathrm{prod}\big(y,\mathrm{pow}(x,y)\big) \end{cases}$$

$$\begin{cases} \mathrm{fact}(0) = S(0) \\ \mathrm{fact}(S(x)) = \mathrm{prod}\big(S(x),\mathrm{fact}(x)\big) \end{cases}$$

のように定義されます.

2 原始帰納的函数

もっとも基本的な函数である定数 0 と後者函数 $S(\ )$ からこのようにして得られる函数を総称する名前をつけます. それが**原始帰納的函数**というものです. きちんと定義しましょう.

定義 1. (0) 定数 0 は (ゼロ変数の) 原始帰納的函数である.

(1) 後者函数 $S(x)$ は 1 変数の原始帰納的函数である.

(2) $1 \le i \le \nu$ のとき $I^i_\nu(x_1,\dots,x_\nu) = x_i$ で定まる函数 I^i_ν は原始帰納的函数である.

(3) $g(y_1,\dots,y_\mu)$ が μ 変数の原始帰納的函数であるとき, その引数 $y_1,\dots,y_\mu$ に ν 変数の原始帰納的函数 $f_1(x_1,\dots,x_\nu),\dots,f_\mu(x_1,\dots,x_\nu)$ をそれぞれ代入して得られる ν 変数函数

$$h(x_1,\dots,x_\nu) = g\big(f_1(x_1,\dots,x_\nu),\dots,f_\mu(x_1,\dots,x_\nu)\big)$$

も原始帰納的函数である.

(4) f が ν 変数の原始帰納的函数, g, が ν+2 変数原始帰納的函数であるとき, これらによって

$$\begin{cases} h(0,y_1,\dots,y_\nu) = f(y_1,\dots,y_\nu) \\ h(S(x),y_1,\dots,y_\nu) = g(x,h(x,y_1,\dots,y_\nu),y_1,\dots,y_\nu) \end{cases}$$

と帰納的に定義される $\nu+1$ 変数の函数 h の原始帰納的函数である.

(5) 以上のことから原始帰納的函数とわかるものだけが原始帰納的函数である. □

ここで, 原始帰納的函数という函数の範囲自体が帰納的に定義されていることに注意しましょう. 定義 1 の (1)〜(4) によると, なにか $\mathbb{N}$ 上の (多変数を含む) 函数のクラス $\mathcal{R}$ が, (0) 定数 0 と (1) 後者函数 $S(x)$ と

[*3] これも, 脚注 1 と同じ理由で $\mathrm{add}(x,y) = x+y$ ではなく $y+x$ と書きます. きっと, 帰納的定義の漸化式を書いたときに, $\nu+1$ 変数函数 h のいちばん右側の引数を動かすように書くべきだったのでしょう. ひとまず慣例どおりに書くと, こういうことになります.

(2) 射影函数 I^i_ν を含み, (3) 代入合成のもとで閉じていて, (4) 帰納的定義のもとでも閉じているならば, $\mathcal{R}$ は原始帰納的函数をすべて含むことになります. 定義 1 の (5) は, そのような条件をみたす $\mathcal{R}$ のうち最小のものが, 原始帰納的函数全体のクラスに一致すると言っています. このことと, ペアノの公準との類似によく注意しておいてください.

— 実例なんかもういいから早く本題に入れ, とお思いの方は次のセクションへどうぞ —

前のセクションで述べたことから, 自然数の加法, 乗法, 冪乗, 階乗などが原始帰納的函数であることがわかります. 原始帰納的函数のクラスが思いのほか広いものであることをわかってもらうために, もう少し例を挙げます.

ひとつ前の自然数を返す函数 $\mathrm{pred}(x)$ は

$$\begin{cases} \mathrm{pred}(0) = 0 \\ \mathrm{pred}(S(x)) = x \end{cases}$$

と定義できます. 0 には前の自然数がないので, ここでは $\mathrm{pred}(0) = 0$ と定義しています.

細かいことを言えば, ここで $\mathrm{pred}(S(x)) = x$ と言ったのでは, 定義 1 の (4) の形にそのままではあてはまらないので, 右辺を $I^1_2(x, \mathrm{pred}(x))$ とでもしなければなりません. ここまでの加法や乗法などの定義でも同様で, このような工夫は随時必要になるのですが, たいていは簡単な話なので, とくに必要のない限りはとりたてて言及しないことにします.

自然数の範囲での引き算は, 大きい数から小さい数を引くことに限定されていますので, 整数の引き算と区別して

$$x \mathbin{\overset{\circ}{-}} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{if } x < y \end{cases}$$

と定めます. この函数は

$$\begin{cases} y \mathbin{\overset{\circ}{-}} 0 = y \\ y \mathbin{\overset{\circ}{-}} S(x) = \mathrm{pred}(y \mathbin{\overset{\circ}{-}} x) \end{cases}$$

をみたすので, 原始帰納的です. この引き算から

$$|x - y| = (x \mathbin{\overset{\circ}{-}} y) + (y \mathbin{\overset{\circ}{-}} x)$$
$$\mathrm{zero}(x) = 1 \mathbin{\overset{\circ}{-}} x$$
$$\mathrm{posi}(x) = \mathrm{zero}(\mathrm{zero}(x))$$

などが得られ, これらも原始帰納的函数となります. $|x - y|$ は読んで字のごとし. $\mathrm{zero}(x)$ は $x = 0$ のとき 1 を返し, $x > 0$ のとき 0 を返します. $\mathrm{posi}(x)$ は逆に $x = 0$ のとき 0 を返し $x > 0$ のとき 1 を返します. これらを利用すると,

$$f(x) = \begin{cases} f_0(x), & \text{if } g(x) = 0 \\ f_1(x), & \text{if } g(x) \neq 0 \end{cases}$$

のような場合分けで定義される函数を

$$f(x) = f_0 \cdot \mathrm{zero}(g(x)) + f_1(x) \cdot \mathrm{posi}(g(x))$$

のように書けるので, f_0, f_1, g が原始帰納的函数なら f も原始帰納的函数になります. また,

$$[\![x \leq y]\!] = \mathrm{zero}(x \mathbin{\overset{\circ}{-}} y)$$
$$[\![x < y]\!] = \mathrm{posi}(y \mathbin{\overset{\circ}{-}} x)$$

のように, $x \leq y$ (resp. $x < y$) のとき 1 を返しそうでないときゼロを返す函数が定義できて, これも原始帰納的函数になります.

ここまでくれば, 次のように定義される函数 $\mathrm{rem}(x, y)$ が原始帰納的函数であることも推して知るべしでしょう

$$\mathrm{rem}(0, y) = 0,\ \mathrm{rem}(x+1, y) = \begin{cases} \mathrm{rem}(x, y) + 1, & \text{if } \mathrm{rem}(x, y) + 1 < y \\ 0, & \text{otherwise} \end{cases}$$

この函数は $y > 0$ のときは x を y で割った余りを返し, $y = 0$ のときはすべての x について 0 を返します.

$$[\![y|x]\!] = \mathrm{posi}(y) \cdot \mathrm{zero}(\mathrm{rem}(x, y))$$

は $y > 0$ でありかつ x が y で割りきれるときに 1 を返しそれ以外のときに 0 を返します.

関数値の総和

$$\sum_{0 \leq k < x} f(k, y)$$

は

$$\sum_{0 \leq k < 0} f(k, y) = 0, \qquad \sum_{0 \leq k < S(x)} f(k, y) = \left(\sum_{0 \leq k < x} f(k, y) \right) + f(x, y)$$

と帰納的に定義されるので, f が原始帰納的函数のときこれも原始帰納的函数になります. 関数値の積

$$\prod_{0 \leq k < x} f(k, y)$$

についても同様です. これにより, x が素数のとき 1 を返しそうでないとき 0 を返す函数 $\mathrm{isPrime}(x)$ が原始帰納的函数であることもわかります. というのも, 等式

$$\mathrm{isPrime}(x) = [\![1 < x]\!] \cdot \mathrm{posi} \left(\prod_{2 \leq k < x} \mathrm{rem}(x, k) \right)$$

が成立するからです. (ここで k の下限が 0 でなく 2 になっていますがどう処理するかは明らかでしょう.)

もう詳細は省きますが, $p_0 = 2$, $p_1 = 3$, 以下 i 番目の素数 p_i を返す函数 p_i も i の函数として原始帰納的ですし, 正の整数 x の最小素因数を返す函数や, x と i に対して x の素因数分解における i 番目の素数 p_i の指数を返す 2 変数函数なども, 原始帰納的函数になります.

このように, さまざまな数論的函数が原始帰納的函数になり, また, 原始帰納的函数 $f(x_1, \ldots, x_\nu)$ によって

$$R(x_1, \ldots, x_\nu) \iff f(x_1, \ldots, x_\nu) \neq 0$$

と特徴づけできる**原始帰納的述語**も, さきほどの $x \leq y$ や $y|x$ や "x は素数である" のように, かなり多岐にわたることがわかります.

3 アッカーマン函数

原始帰納的函数は, その定義にもとづいて値を計算することができます. 定義 1 の (0) 定数函数, (1) 後者函数, (2) 射影函数については言うまでもありませんし, (3) 代入合成も, f_1 から f_μ までの値を計算できれば, つぎにその値を g に放り込んで計算すればいいので問題ありません. (4) 帰納的定義については, 階乗について例を示したように

$$
\begin{aligned}
h(0,y) &= f(y),\\
h(1,y) &= g(0,h(0,y),y) = g(0,f(y),y),\\
h(2,y) &= g(1,h(1,y),y) = g(1,g(0,f(y),y),y),\\
h(3,y) &= g(2,h(2,y),y) = g(2,g(1,g(0,f(y),y),y),y),\\
&\vdots
\end{aligned}
$$

のように順次計算が進みます. ですから, 定義 1 の直後に述べた注意によると, さしあたっては漠然としているものの, なにか "値を計算する具体的な手続きがある函数のクラス" といったものを定義しようとしたとき, それが原始帰納的函数をすべて含むことになるのは間違いがありません.

しかし, 値を計算する具体的な手続きがある函数というものが, 原始帰納的函数だけで尽されるわけでないこともわかります. その例として登場するのが, 次に紹介するアッカーマン函数です.[*4]

定義 2 次のように定義される函数 $A(x,y)$ を**アッカーマン函数**という

$$
\begin{cases}
A(0,y) = S(y)\\
A(S(x),0) = A(x,S(0))\\
A(S(x),S(y)) = A(x,A(S(x),y)). \quad \square
\end{cases}
$$

アッカーマン函数は, 後者函数から足し算, 足し算から掛け算, 掛け算から冪乗を定義したプロセスを延長したものと考えることができます. 最初のいくつかを計算してみると,

$$
\begin{aligned}
A(1,y) &= y+2,\\
A(2,y) &= 2y+3,\\
A(3,y) &= 2^{y+3}-3,\\
A(4,y) &= \overbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}^{y+3} -3,\\
&\vdots
\end{aligned}
$$

となり, x がひとつ増えるごとに $A(x,y)$ はとてつもないスピードで増加することになります.

次のセクションで, このアッカーマン函数 $A(x,y)$ が原始帰納的函数にならないことを証明します. ここではそのために必要な範囲で $A(x,y)$ の性質を調べましょう.

補題 3. すべての x と y について $A(x,y) > y$.

[*4] Wilhelm Friedrich Ackermann (29 March 1896 — 24 December 1962)

証明: $x = 0$ のとき $A(0,y) = S(y) > y$. $A(x,y) > y$ がすべての y について成立したとすると, $A(x+1,0) = A(x,1) > 1 > 0$ であるから $A(x+1,0) > 0$ であり, また y に関する帰納法の仮定として $A(x+1,y) > y$ であったとすれば $A(x+1,y+1) = A(x,A(x+1,y)) > A(x+1,y) > y$ となるので $A(x+1,y+1) > y+1$ も言える. □

補題 4. $A(x,y)$ は x ごとに y の狭義増加函数である: $A(x,y) < A(x,y+1)$.
証明: $x = 0$ のときは明らか. すべての y について $A(x,y) < A(x,y+1)$ であるとすると, 補題 3 の結果から

$$A(x+1,y+1) = A(x,A(x+1,y)) > A(x+1,y). \quad \square$$

補題 5. $A(x,y)$ は y ごとに x の狭義増加函数である: $A(x,y) < A(x+1,y)$.
証明: y にかんする帰納法. $y = 0$ とすると $A(x+1,0) = A(x,1) > A(x,0)$. つぎに $A(x+1,y) > A(x,y)$ を仮定すれば

$$A(x+1,y+1) = A(x,A(x+1,y)) > A(x,y+1)$$

である. これは帰納法の仮定 $A(x+1,y) > A(x,y) > y$ より $A(x+1,y) > y+1$ となることと, y にかんする $A(x,y)$ の単調性による. □

補題 6. $A(a,A(b,y)) \le A(\max\{a,b\}+1, y+1)$.
証明: $c = \max\{a,b\}$ とおこう. すると, 補題 3 から 5 までの結果により,

$$A(a,A(b,y)) \le A(c,A(b,y)) \le A(c,A(c+1,y)) = A(c+1,y+1). \quad \square$$

補題 7. $x \ge 3, n \ge 1$ のとき $nA(x,y) \le A(x,y+n-1)$.
証明: $x = 3$ のときは $n \le 2^{n-1}$ と $A(3,y) = 2^{y+3} - 3$ によって直接計算して確かめればよい. あとは, $A(x+1,y)$ が y について狭義増加で $A(x+1,y) + n - 1 \le A(x+1,y+n-1)$ であることから

$$nA(x+1,y+1) = nA(x,A(x+1,y)) \le A(x,A(x+1,y)+n-1) \le A(x,A(x+1,y+n-1)) = A(x+1,y+n)$$

となるからよい. □

補題 8. $A(x,y+1) \le A(x+1,y)$.
証明: $y = 0$ のときは定義 2 の第 2 式により等号が示される. $y > 0$ のときは $y = z+1$ として

$$A(x,y+1) = A(x,z+2) = A(x,A(1,z)) \le A(x,A(x+1,z)) = A(x+1,z+1) = A(x+1,y). \quad \square$$

補題 9. $x \ge 3, n \ge 1$ のとき $nA(x,y) \le A(x+n-1,y)$.
証明: 補題 7 と補題 8 による. □

補題 10. $x \ge 1$ のとき $A(x+1,y) > A(x,2y)$.
証明: $y = 0$ のときは $A(x+1,0) = A(x,1) > A(x,0)$. つぎに $A(x+1,y) > A(x,2y)$ とすると

$$A(x+1,y+1) = A(x,A(x+1,y)) > A(x,A(x,2y)) \ge A(x,2y+2)$$

ここでの $>$ は帰納法の仮定により, $\ge$ は $x \ge 1$ と $A(1,y) = y+2$ であることによる. □

4　アッカーマン函数は原始帰納的でない

アッカーマン函数 $A(x,y)$ の第一の引数 x をなんらかの定数 c に固定しておけば, y の函数としての $A(c,y)$ は原始帰納的函数であることは (c にかんする帰納法で) 容易にわかります. ところが x を独立変数にして 2 変数函数と思ってしまうと $A(x,y)$ は原始帰納的でないのです. その証明の鍵になるのは, やはり $A(x,y)$ の増加のスピードです.

定理 11. 任意の原始帰納的函数 $f(x_1,\ldots,x_\nu)$ に対して定数 c をうまくとって, すべての $x_1,\ldots,x_\nu$ について

$$f(x_1,\ldots,x_\nu) \le A\big(c, x_1+\cdots+x_\nu\big)$$

となるようにできる.

証明: 原始帰納的函数の構成ステップにかんする帰納法で証明する. 定義 1 の (0),(1),(2) については

$$\begin{gathered} 0<1\le A(0,x),\\ S(y)=A(0,y),\\ I^i_\nu(x_1,\ldots,x_\nu)=x_i<A(0,x_1+\cdots+x_\nu)\end{gathered}$$

であるから $c=0$ とすればよい.

(3) の代入合成については,

$$\begin{gathered} f_1(x_1,\ldots,x_\nu)\le A(c_1,x_1+\cdots+x_\nu)\\ \vdots\\ f_\mu(x_1,\ldots,x_\nu)\le A(c_\mu,x_1+\cdots+x_\nu)\\ g(y_1,\ldots,y_\mu)\le A(d,y_1+\cdots+y_\mu)\end{gathered}$$

となる定数 $c_1,\ldots,c_\mu$ と d がそれぞれ取れたとして,

$$\begin{aligned} g(f_1(x_1,\ldots,x_\nu),\ldots,f_\mu(x_1,\ldots,x_\nu)) &\le A(d,f_1(x_1,\ldots,x_\nu)+\cdots+f_\mu(x_1,\ldots,x_\nu))\\ &\le A(d,A(c_1,x_1+\cdots+c_\nu)+\cdots+A(c_\mu,x_1+\cdots+x_\nu))\\ &\le A(d,\mu A(\max\{c_1,\ldots,c_\mu\},x_1+\cdots+x_\nu)\\ &\le A(d,A(\max\{c_1,\ldots,c_\mu,3\}+\mu-1,x_1+\cdots+x_\nu))\\ &\le A(\max\{d,c_1,\ldots,c_\mu,3\}+\mu,\ x_1+\cdots+x_\nu+1)\\ &\le A(\max\{d,c_1,\ldots,c_\mu,3\}+\mu+1,\ x_1+\cdots+x_\nu).\end{aligned}$$

ここで, 4 つめの $\le$ で補題 9, 5 つめの $\le$ で補題 6, 最後の $\le$ では補題 8 を使っている. この式から $c=\max\{d,c_1,\ldots,c_\mu,3\}+\mu+1$ とおけば

$$g(f_1(x_1,\ldots,x_\nu),\ldots,f_\mu(x_1,\ldots,x_\nu))\le A(c,\ x_1+\cdots+x_\nu)$$

となる.

(4) の帰納的定義については次のとおり. 函数 $h(x,y_1,\ldots,y_\nu)$ が

$$\begin{cases} h(0,y_1,\ldots,y_\nu)=f(y_1,\ldots,y_\nu)\\ h(S(x),y_1,\ldots,y_\nu)=g(x,h(x,y_1,\ldots,y_\nu),y_1,\ldots,y_\nu)\end{cases}$$

と定義されたとする. ここで f と g について定数 b と d が

$$f(y_1,\dots,y_\nu) \le A(b,\max\{y_1,\dots,y_\nu\}+1)$$
$$g(x,z,y_1,\dots,y_\nu) \le A(d,\max\{x,z,y_1,\dots,y_\nu\}+1)$$

となるようにとれている仮定しよう. $c=\max\{b,d+2\}$ とおくと, 帰納的に定義された h について

$$h(x,y_1,\dots,y_\nu) \le A(c,x+y_1+\cdots+y_\nu)$$

となることを示そう. $x=0$ のときは

$$h(0,y_1,\dots,y_\nu)=f(y_1,\dots,y_\nu)\le A(b,y_1+\cdots+y_\nu)\le A(c,0+y_1+\cdots+y_\nu).$$

つぎに, x に関する帰納法の仮定として,

$$h(x,y_1,\dots,y_n) \le A(c,x+y_1+\cdots+y_\nu)$$

だったとすると,

$$\begin{aligned}
h(x+1,y_1,\dots,y_\nu) &= g(x,h(x,y_1,\dots,y_\nu),y_1,\dots,y_\nu)\\
&\le A(d,x+h(x,y_1+\cdots+y_\nu)+y_1+\cdots+y_\nu)\\
&\le A(d,A(c,x+y_1+\cdots+y_\nu)+x+y_1+\cdots+y_\nu)\\
&\le A(d,2A(c,x+y_1+\cdots+y_\nu))\\
&\le A(d+1,A(c,x+y_1+\cdots+y_\nu))\\
&\le A(c-1,A(c,x+y_1+\cdots+y_\nu))\\
&= A(c,x+y_1+\cdots+y_\nu+1)
\end{aligned}$$

となる. (4 つめの $\le$ は補題 10 による.) この不等式から,

$$h(x+1,y_1,\dots,y_\nu) \le A(c,(x+1)+y_1+\cdots+y_\nu)$$

となるので, 帰納法が成立する.

以上のことから, ある定数 c が存在してすべての $x_1,\dots,x_\nu$ について

$$f(x_1,\dots,x_\nu) \le A(c,x_1+\cdots+x_\nu)$$

となるような函数 f の全体のクラスは, 定数函数 0 と後者函数 $S(x)$ と射影函数を含み, 代入合成と帰納的定義のもとで閉じている. そのようなクラスは原始帰納的函数をすべて含む. これが証明したかったことであった. □

さて, $A(x,y)$ が原始帰納的函数でありえないことはもはや明白です. というのも, もしも $A(x,y)$ が原始帰納的なら, 1 変数函数としての $A(x,x)$ も原始帰納的なので, 定理 11 によってある定数 c について

$$A(x,x) \le A(c,x)$$

となるはずですが, この x に $c+1$ を代入すれば補題 5 によって

$$A(c,c+1) < A(c+1,c+1) \le A(c,c+1)$$

となって矛盾するからです. したがって $A(x,x)$ は原始帰納的でなく, $A(x,y)$ も原始帰納的でないことになります.

しかしながら, 各 x ごとに $A(x,y)$ を計算する方法はあるので, アッカーマン函数は “値を計算する手続きをもつ函数” には違いありません. つまり, 原始帰納的函数の範囲は, 計算手続きをもつ函数をすべて網羅するほどには広くないわけです.

参考文献

前原昭二 『数学基礎論入門』朝倉書店 (1977 年/2006 年)

篠田寿一 『帰納的関数と述語』河合文化教育研究所 (1997 年)