vol. $2^{\aleph_0} = \aleph_{12}$

# The Dark Side of Forcing

vol. $2^{\aleph_0} = \aleph_{12}$

# Wild な数学　ケモノの数学

淡中 圏

グロタンディークは『Esquisse d'un Programme』で**ペアノ曲線**などの**空間充填曲線**や**バナッハ・タルスキーの定理**のような**病的**な例を生み出さないようにトポロジーの基礎を作り替えるプログラムを提唱し、その目標を **tame topology**、すなわち**手懐けられたトポロジー**と名付けた。

それに対して、古典的手法がどうにかこうにか扱ってきたトポロジーは **wild topology**、すなわち**野生のトポロジー**だ、というのだ。

またシェラハは、『On what I do not understand (and have something to say)』という集合論の進むべき道について書いた論文に、**666** という番号を付けている。

これはもちろんヨハネ黙示録の

> ここに、知恵が必要である。思慮のある者は、獣の数字を解くがよい。その数字とは、人間をさすものである。そして、その数字は 666 である。

の引用であると考えられる。

シェラハの意図は明らかだが、この引用ではおそらく「獣の数字」という部分には特に意味は与えられていないのであろう。

しかし、グロタンディークのアイディアと絡めると、なんとも面白い気もする。

数学にはまだまだ野生の対象、ケモノの数学があふれているのだ。tamer として手懐けようとするもよし、naturalist としてその脅威に感嘆するもよし。

さあ、装備を整えて今こそ数学の原野に出発しようではないか！

# 目次

Wild な数学　ケモノの数学 i

第 1 章　フィボナッチ数列と作用の対象化 1
1.1 高校数学の復讐 . . . . . 1
1.2 線形代数による、より一般的な解き方 . . . . . 4
1.3 定義通りの宣言的な再帰関数による計算 . . . . . 7
1.4 線形時間で計算できるフィボナッチ数列と末尾再帰最適化について少し . 18
1.5 行列の冪による計算量 $O(\log n)$ によるフィボナッチ数列 . . . . . 28
1.6 時間計測 . . . . . 30
1.7 まとめ . . . . . 32

参考文献 34

第 2 章　可換環 次元の奇妙な物語 35
2.1 ネーター局所環の次元定理 . . . . . 36
2.2 素イデアルの集合とパラメーター系及び正則列のレシピ . . . . . 39
2.3 次元が有限でないネーター環の例 . . . . . 43
2.4 鎖状でないネーター環の例 . . . . . 44

参考文献 49

第 3 章　束論日記 51
3.1 はじめに . . . . . 51
3.2 冪等モノイドに定まる順序、半束の定義 . . . . . 51
3.3 半束の準同型とイデアル、フィルタ . . . . . 53
3.4 最後に . . . . . 56

参考文献 56

第 4 章　カルマ・モナド 57

参考文献 57

# 第１章

# フィボナッチ数列と作用の対象化

淡中　圏

## 1.1　高校数学の復讐

またフィボナッチ数列か、どうせ黄金比の話でもするのだろう、と思ったかもしれないが、黄金比の話はしないので安心してほしい。

おっと、さっそく嘘をついてしまった。数学が依って立つ標準的な論理学では矛盾からはなんでも証明できるので、この冒頭の文を真だと私が言い張る限り、この文章全体で私は何でも証明できることになる。

なんと幸先良いスタートであろうか。

さてフィボナッチ数列である。それは、

$$\mathrm{fib}_0 = 1, \mathrm{fib}_1 = 1, \mathrm{fib}_{n+2} = \mathrm{fib}_{n+1} + \mathrm{fib}_n \tag{1.1}$$

という式で定義される数列である。高校で習った人も多いかと思う。添え字が 0 から始まるのは単なる趣味なので気にしなくて良い。

フィボナッチは本名はレオナルド・ダ・ピサ（ピサのレオナルド）でフィボナッチは「ボナッチの息子」という意味のあだ名だ。ちなみに「ボナッチ」も父親のあだ名だ。あだ名の漸化式が作れそうである[*1]。

彼は現代では「アラビア数字」と呼ばれる記数法を「インドの方法」という正しい名前でヨーロッパにもたらした。彼自身は商人の家系であった。貿易の仕事でイスラム圏に移住した父親の後について、アルジェリアでアラビア数字を学んだという。その有効性を見抜いた彼は、アラブの数学を学ぶために、エジプト、シリア、ギリシア等を旅したという。そしてヨーロッパに帰って、得た知識を「算盤の書」としてまとめて出版した。商人らしく、アラビア数字の応用例は簿記、単位の変換、利子の計算等であった。

時は１２世紀から１３世紀にかけて。１２世紀ルネサンスという言葉もあるように、十字軍やレコンキスタなどの戦争や、貿易などにより、アラブから進んだ知識がヨーロッパには流れ込んでいた。その中心はイスラム圏に近いスペイン、そしてイタリアだった。イタリア商人は１４世紀から１５世紀にかけて「複式簿記」という世界を変える大発明をし

[*1] 意味不明

ている。これらは今では数学とは縁遠い存在と見られがちだが、この時代に名もなき商人の数学への貢献は計り知れない。複式簿記に関する最古の文献も数学書である。

さて先ほどの「算盤の書」の中でフィボナッチはインドでは 6 世紀ころから知られていた有名な問題について書いた。それがフィボナッチ数列なのだ。

- 1 つがいのウサギは、生まれて 2 ヶ月後から毎月 1 つがいずつのウサギを産む
- 最初の月は 1 つがいとする[*2]
- ウサギが死ぬことはない[*3]

ルールはこれだけ。

すると 0 月目は 1 つがい。

1 月目はまだ一つがい。

2 月目に 1 つがい生まれるので 2 つがい。

3 月目には 1 つがいはまだ生まれたばかりなので、生まれるのは 1 つがいで 3 つがい。

4 月目は 2 つがい生まれるので、5 つがい。

となっていく。

これを式にすると、1.1 になるわけだ。

さて、高校ではこのような式を**三項間漸化式**といい、次のようにして一般項を求める。

まず**特性方程式**という面妖な者が天下りしてくる。

天下りとは、数学界のジャーゴンで、「なぜそういうものが出てくるのか説明せずに導入される概念」についていう。受け入れると補助金がもらえるのかもしれない。

一般の 3 項間漸化式

$$aa_{n+2}n + ba_{n+1} + ca_n = 0 \tag{1.2}$$

の特性方程式は

$$ax^2 + bx + c = 0$$

となる。

特にフィボナッチ数列の場合は $\mathrm{fib}_n - \mathrm{fib}_{n-1} - \mathrm{fib}_{n-2} = 0$ と書き換えれば、

$$x^2 - x - 1 = 0 \tag{1.3}$$

となる。

この方程式は、古代バビロニア人も知っていたと言われる**二次方程式の解の公式**

$$x = \frac{-b \pm \sqrt{b^x - 4ac}}{2a}$$

で解ける。

フィボナッチ数列の場合は

$$x = \frac{1 \pm \sqrt{5}}{2}$$

が特性方程式の解だ。

おっと、あれの話はしないよ。二回も矛盾することはごめんだからね（数学的に一回の矛盾と二回の矛盾が区別できるか、という話は面白いが、これも今回は見送ろう）。

---

[*2] この一つがいはきっと神様が泥をこねて作ったの違いない

[*3] 寂しくても

ここで、この解が重解の場合、すなわち判別式 $\sqrt{b^2-4ac}=0$ の場合は別に考えなくていけないのだが、幸いフィボナッチ数列の特性方程式は重解を持たないので、ここでは重解を持たない場合のみ考えよう。

二つの解を $\alpha$ と $\beta$ とする。二つの解のどちらを $\alpha$ にしてどちらを $\beta$ にしても良い。

すると次の**解と係数の関係**が成り立つ。

$$\alpha+\beta=-\frac{b}{a},\quad \alpha\beta=\frac{c}{a}.$$

フィボナッチ数列の場合、$\alpha+\beta=1,\alpha\beta=-1$ が実際に成り立つことを確かめても良いだろう。

これを使えば、1.2 は

$$\begin{aligned}a_{n+2}-\beta a_{n+1}&=\alpha(a_{n+1}-\beta a_n)\\ a_{n+2}-\alpha a_{n+1}&=\beta(a_{n+1}-\alpha a_n)\end{aligned}$$

と変形できることがわかる。確信が持てなければ、ぜひ展開して整理して確かめてほしい。

上の式は、$a_{n+1}-\alpha a_n$ が初項 $a_1-\alpha a_0$、公比 $\beta$ の等比級数であり、$a_{n+1}-\beta a_n$ が初項 $a_1-\beta a_0$、公比 $\alpha$ の等比級数であることを意味している。

つまり、

$$\begin{aligned}a_{n+1}-\beta a_n&=\alpha^n(a_1-\beta a_0)\\ a_{n+1}-\alpha a_n&=\beta^n(a_1-\alpha a_0)\end{aligned}$$

となる。

$\alpha$ と $\beta$ は違うと仮定しているので、これは連立方程式になり、上の式から下の式を引いて整理することで簡単に解ける。

すなわち

$$a_n=\frac{\alpha^n(a_1-\beta a_0)-\beta^n(a_1-\alpha a_0)}{\alpha-\beta}$$

により、数列の一般項を求めることができた。

これをフィボナッチ数列に適用すれば、

$$\mathrm{fib}_n=\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}-\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right)\tag{1.4}$$

がフィボナッチ数列の一般項であることがわかる。

$$\sqrt{5}\fallingdotseq 2.2360679$$

であることに注意すれば

$$-1<\frac{1-\sqrt{5}}{2}<0<1<\frac{1+\sqrt{5}}{2}$$

がわかるので、1.4 の

$$\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}$$

の部分は 0 に収束し、

$$\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}$$

は無限大に発散する指数関数である。

つまり、

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$$

という数列はどんな多項式もいずれ抜かしてしまいながら増大していくことがわかる。

ウサギの増加の仕方をモデル化したものから導かれた結果は、マルサスが『人口論』で指摘した「人口は指数関数論的に増える」という分析の初期のものと言えるだろう。

実際、培養皿の上で細菌を増やすと、途中までは指数関数的に増える。

しかしもちろん、そのまま増え続けたりはしない[*4]。

実際には、環境の悪化や、その生物を栄養源にしている生物の増加などで、生物は増え続けることはできない。

例えば、環境が受容できる生物数に限りがあり、そこに近づくと環境が悪化して繁殖が弱まる状況をモデル化したものに**ロジスティック方程式**がある。興味があるなら、是非調べてほしい。

## 1.2　線形代数による、より一般的な解き方

ここまでは高校数学の範囲だが、これを大学 1 年で習う線形代数を使って解き直してみよう。

数学という山を登っていると、ある程度の高さまで来て後ろを振り返ると、通ってきた道とはまた違う道が見つかることがある。そしてその道は自分が今いるところよりも高い場所へと繋がっている場合があるのだ。なので、一度解いた問題を、新たに手に入れた道具で解き直すことは、単なる余儀以上の意味を持ちうるのだ。

線形代数は**線形性**という小学校で習った比例を拡張した特徴を持つ変化を扱う。そして、線形性を持つ変化はだいたい（有限次元ならば）行列で表せることが知られている。なので、多くの人には線形代数とは、行列に関する学問に見えているようだ。

世の中の変化の多くは、線形性を満たしていないが、幸いなことに適切な仮定のもとに線形に近似できることが多い。これは意地悪な視線で見れば、我々が現在理解できている事柄の多くは線形近似ができることに過ぎないことの裏返しに過ぎないかもしれないが。

例えば、先ほどの話に戻れば、フィボナッチ数列や指数関数は、線形性を強く持った対象であり、ロジスティック方程式は、非線形な対象である。取り扱いやすいように単純化すれば線形なものが現れるが、現実に近づければ非線形な対象が現れる。とはいえ、ロジスティック方程式やその離散化である**ロジスティック写像**が関係する**カオス**や**フラクタル**などの現象が数学的に解析できるようになったのは、非線形な対象の中の線形な部分を様々な視点から見つけ出す技術が発達したからだとも言える。

さて、それではフィボナッチ数列の漸化式 1.1 を行列で表そう。この式の入力は、$\mathrm{fib}_n$ と $\mathrm{fib}_{n+1}$ である。そして、この両方を $k$ 倍したら $\mathrm{fib}_{n+2}$ も $k$ 倍されるし、$\mathrm{fib}_{n+2} = \mathrm{fib}_{n+1} + \mathrm{fib}_n, \mathrm{fib}'_{n+2} = \mathrm{fib}'_{n+1} + \mathrm{fib}'_n$ とあったら、$(\mathrm{fib}_{n+2} + \mathrm{fib}'_{n+2}) = (\mathrm{fib}_{n+1} + \mathrm{fib}'_{n+1}) + (\mathrm{fib}_n + \mathrm{fib}'_n)$ が成り立つ。これはつまり、この式が線形性を満たしていることを意味する。では出力は何だろうか？　$\mathrm{fib}_{n+2}$ と言いたいところだが、それでは $\mathrm{fib}_{n+3}$ を計算する時

[*4] さもなくば海は牡蠣で一杯になってしまうであろう

に困ってしまう。$\mathrm{fib}_{n+3}$ を計算するためには $\mathrm{fib}_{n+1}$ も必要だ。そこで、$\mathrm{fib}_{n+2}$ と $\mathrm{fib}_{n+1}$ を出力と考えよう。すると、

$$\begin{pmatrix} \mathrm{fib}_{n+2} \\ \mathrm{fib}_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \mathrm{fib}_{n+1} \\ \mathrm{fib}_{n} \end{pmatrix}$$

となる。行列の掛け算の仕方が分かっていれば、簡単に確かめることができるだろう。

行列を使うことによって、この式は等比数列とほぼ同じものに見えるようになってしまった。つまり、等比級数の場合と同様に、

$$\begin{pmatrix} \mathrm{fib}_{n+1} \\ \mathrm{fib}_{n} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} \mathrm{fib}_{1} \\ \mathrm{fib}_{0} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

となる。よって、フィボナッチ数列の一般項を求める問題は、

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

を分かりやすい形に書くことに帰着されるのだ。

こうなれば、もう行列に関する一般論である。

このような問題は、線形代数では、**行列の標準形**を求めることで解かれる。この場合は対角化だ。大学で数学を学んだもので、出会わなかった人はほとんどいないであろう。

対角化をするためには、行列の**固有値**と**固有ベクトル**を求めなくてはいけない。そのためにまず行列の**固有方程式**を求めよう。

この辺りの委細は線形代数学の教科書に当たって欲しい。そうすれば、ここで行われていることが、天下り的にではなく、細部まで理由を整理された形で乗っているはずだ。

行列 $A$ の固有方程式は

$$\det(\lambda I - A) = 0$$

という形になる。この場合は行列

$$\begin{pmatrix} \lambda - 1 & -1 \\ -1 & \lambda \end{pmatrix}$$

の行列式になるので、

$$\lambda^2 - \lambda - 1 = 0$$

となる。

これは 3 項間漸化式の特性方程式と同じものだ。

実際、任意の 3 項間漸化式に同じ手法が適用できて、漸化式の特性方程式は行列の固有方程式に一致する。

この方程式の解が**固有値**と呼ばれるものになる。

あるスカラー値 $\alpha$ が $m$ 次正方行列 $X$ の固有値であるとは、ある 0 でないベクトル $v$ があって、

$$Xv = \alpha v$$

となることである。

この $v$ を $X$ の**固有ベクトル**という。

そして $m$ 次正方行列 $X$ の固有方程式は $m$ 次式であり、これが異なる $m$ 個の解を持つとき、つまり $X$ が異なる $m$ 個の固有値 $\alpha_1, \alpha_2, \ldots, \alpha_m$ を持つとき、それぞれの固有値に

対応する固有ベクトル $v_1, v_2, \ldots, v_m$ を並べた $m$ 次正方行列 $P = \begin{pmatrix} v_1 & v_2 & \ldots & v_m \end{pmatrix}$ は正則行列になり、

$$P \begin{pmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_m \end{pmatrix} = XP$$

が成り立つ。これにより

$$\begin{pmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_m \end{pmatrix} = P^{-1}XP \tag{1.5}$$

が成り立つ。[*5]これを行列の**対角化**という。

この形ができると、$X^n$ が簡単に求まる。

なぜなら、

$$X^n = PP^{-1}X^nPP^{-1} = P(P^{-1}XP)^nP{-}1 = P \begin{pmatrix} \alpha_1^n & 0 & \cdots & 0 \\ 0 & \alpha_2^n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_m^n \end{pmatrix} P^{-1}$$

となるからである。

ここで、フィボナッチ数列に戻ろう。$A$ の固有値を

$$\alpha = (\frac{1+\sqrt{5}}{2}),\ \beta = (\frac{1-\sqrt{5}}{2})$$

とおくと、それぞれに対応する固有ベクトルは、

$$\begin{pmatrix} \alpha \\ 1 \end{pmatrix},\ \begin{pmatrix} \beta \\ 1 \end{pmatrix},$$

であり、よって先ほどの記号をもう一度使うと、

$$P = \begin{pmatrix} \alpha & \beta \\ 1 & 1 \end{pmatrix}, P^{-1} = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 & -\beta \\ -1 & \alpha \end{pmatrix}$$

となる。これにより、$A^n$ は

$$\begin{aligned} \frac{1}{\sqrt{5}} \begin{pmatrix} \alpha & \beta \\ 11 & \end{pmatrix} \begin{pmatrix} \alpha^n & 0 \\ 0 & \beta^n \end{pmatrix} \begin{pmatrix} 1 & -\beta \\ -1 & \alpha \end{pmatrix} &= \frac{1}{\sqrt{5}} \begin{pmatrix} \alpha^{n+1} & \beta^{n+1} \\ \alpha^n & \beta^n \end{pmatrix} \begin{pmatrix} 1 & -\beta \\ -1 & \alpha \end{pmatrix} \\ &= \frac{1}{\sqrt{5}} \begin{pmatrix} \alpha^{n+1} - \beta^{n+1} & -\beta\alpha^{n+1} + \alpha\beta^{n+1} \\ \alpha^n - \beta^n & -\beta\alpha^n + \alpha\beta^n \end{pmatrix} \end{aligned}$$

となる。

これにより、フィボナッチ数列の一般項は

---

[*5] これはつまり、$v_1, v_2, \ldots, v_m$ は $m$ 次元線形空間の基底であり、$P$ は基底の変換行列と呼ばれるもので、1.5 は $A$ をこの基底で表現すると対角行列になる、ということを意味する。それは、$X$ の $v_1, v_2, \ldots, v_m$ への作用がスカラー倍であることから明らかであろうし、これが明らかであることが理解できることを、線形代数の理解の試金石とすることもできるであろう。

$$\begin{pmatrix} a_{n+1} \\ a_n \end{pmatrix} = A^n \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

であり、

$$\alpha + \beta = 1$$

に注意すれば

$$\mathrm{fib}_n = \frac{1}{\sqrt{5}}\left( (\frac{1+\sqrt{5}}{2})^{n+1} - (\frac{1-\sqrt{5}}{2})^{n+1} \right)$$

となって。高校数学での特性方程式による解法の答えと一致する。

この方法の何が嬉しいかは、まずこれは線形ならば、任意の $m$ についての $m$ 項間漸化式

$$c_m a_{n+m} + c_{m-1} a_{n+m-1} + \ldots c_0 a_n = 0$$

に対して適用可能であることだ。

これが線形でない場合、例えば

$$a_n + 1 = a_n^2 + c$$

のような漸化式は、突然挙動が複雑になる。

興味がある方は、**ジュリア集合**および**マンデルブロ集合**について調べてみるといいと思う。

また、ほぼ同じ手法が**線形微分方程式**にも適用可能である。

数列の漸化式は、ある数列の満たす条件（方程式）のことなので、**差分方程式**とも呼ばれる。

差分方程式とは、時間の流れが離散的な世界での状態の変化と考えることができる。そして時間の流れが連続的な場合が微分方程式だ。

ここで説明した方法は、**線形差分方程式**一般に適用できる方法であり、それが線形微分方程式にほぼ同じ形で応用可能なのは、それほど不思議ではないのかもしれない。

そして非線形な差分方程式と同様に、非線形な微分方程式も、たとえ単純なものでも複雑で予測不可能な挙動を示すものがある。

微分および差分においてそれらを研究するのが、**カオス**や**フラクタル**などの研究分野である。

## 1.3 定義通りの宣言的な再帰関数による計算

では、一般項も分かったので、実際に大きなフィボナッチ数列を計算してみよう、とはなかなかならない。

高校の授業でフィボナッチ数列の一般項を使って、１００番目のフィボナッチ数を計算してみよう、などという問題が出たところを見た覚えがない。

それをやろうと思おうと、

$$1+\sqrt{5}, (1+\sqrt{5})^2 = 6+2\sqrt{5}, (1+\sqrt{5})^3 = 16+8\sqrt{5}$$

とやっていくことになるが、実際これは大変である。

こういう大変なことはコンピュータにやらせたいと思うところだが、$\sqrt{5}$ をどう扱うかという問題が発生する。

浮動小数点で $\sqrt{5}$ を計算して行くと少し困ったことが起こる。

ここでは数学的な式を割とそのまま定義にできる **Haskell** でプログラムを書いてみよう。

```
root5 = sqrt 5
alpha = (1 + root5) / 2
beta = (1 - root5) / 2
float_fib n = (alpha^(n+1) - beta^(n+1)) / root5
```

プログラミングも Haskell も知らない方でも、意味は明らかであろうと思う。これを Haskell の対話環境[*6]に読み込んで使ってみると、

```
*Main> float_fib 0
1.0
*Main> float_fib 1
1.0
*Main> float_fib 2
2.0
*Main> float_fib 3
3.0
*Main> float_fib 4
5.0
*Main> float_fib 5
8.0
*Main> float_fib 6
13.0
*Main> float_fib 7
21.0
*Main> float_fib 8
34.0
```

といい調子そうだが、

```
*Main> float_fib 9
54.99999999999999
```

といきなり問題発生である。

浮動小数点には精度の限界があるので、このような丸め誤差が出てしまう。

しかし、そもそもフィボナッチ数列は自然数にしかならないので、こんな誤差に悩まされるのは理にかなわない。

さらに大きな数を入れてみよう。

```
*Main> float_fib 1000
7.033036771142261e208
*Main> float_fib 1100
```

[*6] **GHCi** が有名である。今回もこれで動かしている。

```
5.570663342003133e229
*Main> float_fib 1200
4.41235999181296e250
*Main> float_fib 1300
3.494901684428648e271
*Main> float_fib 1400
2.7682097123728895e292
```

表示された結果だけ見た限りでは、ものすごい誤差が出ていように見える。

そして、最後のとどめが

```
*Main> float_fib 1500
Infinity
```

である。

そもそもこのような計算に浮動小数点を使うことが間違っている。

あなたの上司が業務でフィボナッチ数列が欲しいとなって、あなたの作ったこの関数を使ったときに、妙な誤差のある数字や、明らかに小さな桁がない指数表記や、Infinity などと答えが出たら面食らうのも仕方がないだろう。

一つの解決策としては、$\sqrt{5}$ を二乗したら 5 になる数として、代数的に扱う、つまい

$$(a+b\sqrt{5})(c+d\sqrt{5}) = (ac+5bd)+(ad+bc)\sqrt{5}$$

として計算していくことである。こうすれば最後には $\sqrt{5}$ は綺麗に消えてしまう。

Haskell で代数的実数を作るプロジェクトは過去に存在するので、それを使ってもいいし、環 $\mathbb{Z}[\sqrt{5}] = \{a+b\sqrt{5} | a,b \in \mathbb{Z}\}$ だけを作ることは、それほど難しくない。

ただ、少々フィボナッチ数列を計算する程度のタスクに作り込み過ぎの感がある。

そこで、初心に帰って、素直に自然数の演算でフィボナッチ数列を計算してみよう。

そうすれば Haskell の整数はデフォルトでは多倍長整数、すなわちメモリの許す限り長い整数が使えるので、先ほどのように Infinity になったり、オーバーフローして突然マイナスの値になったりはしない。

まず一番単純な定義通りの実装をしてみよう、Haskell なら、自然な形でそれができる。

```
simple_fib 0 = 1
simple_fib 1 = 1
simple_fib n = simple_fib (n-1) + simple_fib (n-2)
```

このように、関数の定義の中にその関数自体を使うことを、**再帰関数**といい、このような定義を**再帰的定義**という。

これはまさにフィボナッチ数列の漸化式による定義をそのまま書き下ろしただけである。このように、プログラムが**どんな処理をするか**ではなく、**どんな性質の結果が欲しいか**を指定することでプログラムを書くことを、**宣言型プログラミング**という。それに対して、前者のように処理内容を書いてプログラムを書くことを**命令型プログラミング**という。

Haskell は、関数の定義を書くことでプログラムが作れるし、関数の定義の順番も自由なので、[*7]宣言的プログラミングに非常に適した言語である。

[*7] ただし Haskell は定義を上からチェックするので、先ほどの関数の定義の 0 の場合や 1 の場合を下に持ってくると無限ループに陥ってしまう。

では動かしてみよう。

```
*Main> simple_fib 1
1
*Main> simple_fib 2
2
*Main> simple_fib 3
3
*Main> simple_fib 4
5
*Main> simple_fib 5
8
*Main> simple_fib 6
13
*Main> simple_fib 7
21
*Main> simple_fib 8
34
*Main> simple_fib 9
55
*Main> simple_fib 10
89
```

順調そのものに見える。しかし、これを続けているとまたまた暗雲が立ち込め始める。

```
*Main> simple_fib 20
10946
*Main> simple_fib 21

17711
*Main> simple_fib 22


28657
*Main> simple_fib 23



46368
*Main> simple_fib 24




75025
*Main> simple_fib 25
```

```
121393
*Main> simple_fib 26
```

```
196418
*Main> simple_fib 27
```

```
317811
*Main> simple_fib 28
```

```
514229
*Main> simple_fib 29
```

```
832040
*Main> simple_fib 30
```

```
1346269
*Main> simple_fib 31
```

```
2178309
*Main> simple_fib 32
```

```
3524578
*Main> simple_fib 33
```

```
5702887
*Main> simple_fib 34
```

```
9227465
*Main> simple_fib 35
```

14930352

これは紙面では表現できない結果を待つ時の時間経過を表現した冗談であり、筒井康隆の『虚人たち』オマージュである。

冗談はさておき、上司が大きなフィボナッチ数が欲しいときに 35 程度でこんなに待たされたら怒ってしまうであろう。

この遅さの原因は、定義に戻れば明らかで、この関数は

```
simple_fib n = simple_fib (n-1) + simple_fib (n-2)
             = (simple_fib (n-2) + simple_fib (n-3)) + simple_fib (n-2)
```

と展開されていくので、これでは $n-2$ 項が二回計算されている。

計算量を関数の呼び出しの数を数えることにより計算してみよう。

$\mathrm{comp\_fib}_n$ を

```
simple_fib n
```

の $n$ 項目を計算するためにこの関数自身の呼び出しを何回行ったかとして定義する。

すると、$\mathrm{comp\_fib}_0 = \mathrm{com\_fib}_1 = 1$ は明らかである。

そして

```
simple_fib n = simple_fib (n-1) + simple_fib (n-2)
```

より、

$$\mathrm{comp\_fib}_n = \mathrm{comp\_fib}_{n-1} + \mathrm{comp\_fib}_{n-2}$$

が明らかに成り立つ。これにより $\mathrm{comp\_fon}_n$ がフィボナッチ数列そのものであることがわかる。すなわち、前半の結果を使えば、comp_fib は指数関数のオーダーで発散するのだ。

道理で時間がかかるわけだ。

この簡単な関数は、時間がかかると言われる計算量が指数関数的に増加する関数の非常にいい例であり、指数関数的な計算量にどれくらい時間がかかるかを肌で感じるよい教材であるといえよう。

## 1.4　線形時間で計算できるフィボナッチ数列と末尾再帰最適化について少し

ではこれをどうやったら早くできるであろうか。

問題はせっかく $n-1$ 番目のフィボナッチ数を計算する際、せっかくすでに計算していたた $n-2$ 番目の結果を捨ててしまっていることなので、それを覚えておくことにしよう。命令型プログラミング言語では「何かを覚えておく」とは変数の値として、随時それを更新していくことである。それはつまり、プログラムがプログラムを包む**環境**に影響を与えるということである。

Haskell のような**関数型プログラミング言語**は環境に影響を与えないエコロジー志向のプログラミング言語なので、そういうことは極力しない。

その代わりに環境を引数として、しっかり周りから隔離して自分で持ち運ぶことになる。[*8]

では実装してみよう。

```
linear_fib n =
  let
    fib_aux fib_k fib_k_1 m =
      if m <= 1
      then fib_k
      else fib_aux (fib_k+fib_k_1) fib_k (m-1)
  in
    fib_aux 1 1 n
```

これで 5 番目のフィボナッチ数を計算すると、

```
linear_fib 5 = fib_aux 1 1 5
             = fib_aux 2 1 4
             = fib_aux 3 2 3
             = fib_aux 5 3 2
             = fib_aux 8 5 1
             = 8
```

となる。

$n$ 番目のフィボナッチ数を計算しようとする場合、式変形の間、いつでも一つ目の引数はいつでもフィボナッチ数であり、それが $k$ 番目のフィボナッチ数だとすると、二つ目の引数は $k-1$ 番目のフィボナッチ数であり、そして三つ目の引数 $m$ との間には

$$k + m = n + 1 \tag{1.6}$$

という関係が成り立っている[*9]。よって $m$ が 1 になった時に式変形を止めれば、$n$ 番目のフィボナッチ数が計算できるという仕掛けだ。

---

[*8] 同じ計算を何回もしたくない場合の対策として、これまで計算した結果を全て覚えておいて、要求されたらすぐに返せるようにする**メモ化**という手法もよく使われる。しかし、環境へ影響を与えることが簡単な言語では、メモ化は自然に実装できるのであるが、環境への影響が非常に強く制限されている Haskell では一工夫いる。なので今回はメモ化によるフィボナッチ数列計算の高速化は取り上げなかった。

[*9] 数学的帰納法で証明できる

そして、上の式変形からも、この計算が $n$ に比例する計算量でできることは明らかであろう。

この定義にはもう一つ利得がある。

`simple_fib` では、再帰的に呼び出した関数の返り値を加工して返す必要があったが、`linear_fib` では、再帰的に呼び出した関数の返り値はそのまま返している。

このような形を**末尾再帰**という。

末尾再帰だと何が嬉しいかに応えるためには、関数呼び出しの仕組みを少し考える必要がある。

プログラムの途中で関数が呼び出されると、関数の本体の実行を始めないといけない。すると、関数の実行が終わったときに、帰ってくる場所、すなわち、どこでその関数が呼び出されたかを覚えておかなくてはいけない。関数は色々な場所から呼び出されるからである。

そして、関数の中で別の関数や自分自身を呼び出した際には、さらに覚えておかなくてはいけない場所が増える。

それは**スタック**と呼ばれるもので管理される。

スタックは**最後に入れたものが最初に出される（LIFO:Last In First Out）**というデータ構造で、ただの箱と思っても良い[*10]。関数の呼び出しがあったら、その箱に呼び出される前にいた場所を書いた紙を入れる。関数の実行が終わったら、箱の一番上にある紙をとって、そこに書いてある場所に計算結果を携えて帰ればいいのである。関数の中で関数の呼び出しがあっても同じことをすれば、いつでも今いる関数が呼び出される前にいた場所に帰ることができる。

頭のいい仕掛けであるが、重要なことは、再帰関数にはただのループにはない余分の仕事が発生しているということである。またあまりに関数呼び出しの階層が深くなると、スタックが限界まで一杯になってしまう。有名な**スタックオーバーフロー**である。

しかし、末尾再帰ならば、このスタックに積む仕事を省くことができる。元の場所に帰ってきてやることは、さらにもう一つ前の元の場所に帰ることなので、そもそもスタックに何も積まずに、最初の場所に帰ればいいのだ。

これは実は、引数をローカル変数に変えて、ただの命令型プログラミングにおけるループに変換していることになる。この変換を**末尾再帰最適化**と呼び、多くの関数型プログラミング環境でサポートされている。そしてその際 1.6 は、ループの間に変わらない条件なので**ループ不変条件**と呼ばれ、プログラムの正しさを考える際に重要なものである。

ループではなく再帰を中心に考えることは、上でも見たように効率の面から見たら足かせではあるが、より宣言的にプログラミングをすることは、これもまた上で見たように、プログラムの意味を変えずにより効率の良い処理へと変換することや、プログラムの意味の正しさを証明するためには、より良い視点をもたらすことができる[*11]。

さて、御託ばかり述べていても仕方がない。実際に計算してみよう。前回重くなり始めた 30 あたりからスタートすると、

---

[*10] また箱か

[*11] **Erlang** という関数型プログラミング言語では、プログラムがアップデートしたことをプログラム内で検知することができて、その時に、末尾再帰の時に呼ばれる関数を新しい関数にしてしまうことにより、**ホットスワッピング**（プログラムを止めずにアップデートすること）を綺麗にプログラムとして表現している。しかし実際にやろうとすると、他の関数との連携が完璧にできていないといけないので、危険性は大きいままだが。

```
*Fibonacci> linear_fib 30
1346269
```

お、早い！　調子に乗って 10 ずつ増やしていくと、

```
*Fibonacci> linear_fib 40
165580141
*Fibonacci> linear_fib 50
20365011074
*Fibonacci> linear_fib 60
2504730781961
*Fibonacci> linear_fib 70
308061521170129
*Fibonacci> linear_fib 80
37889062373143906
*Fibonacci> linear_fib 90
4660046610375530309
*Fibonacci> linear_fib 100
573147844013817084101
```

サクサク動く！　ようし、今度は１０倍ずつ増やすぞ。

```
*Fibonacci> linear_fib 1000
70330367711422815821835254877183549770181269836358732742604905087154537118196933579742249494562611733487750449241765991088186363265450223647106012053374121273867339111198139373125598767690091902245245323403501
*Fibonacci> linear_fib 10000
54438373113565281338734260993750380135389184554695967026247715841208582865622349017083051547938960541173822675978026317384359584751116241439174702642959169925586334117906063048089793531476108466259072759367899150677960088306597966641965824937721800381441158841042480997984696487375337180028163763317781927941101369262750979509800713596718023814710669912644214775254478587674568963808002962265133111359929762726679441400101575800043510777465935805362502461707918059226414679005690752321895868142367849593880756423483754386342639635970733756260098962462668746112041739819404875062443709868654315626847186195620146126642232711815040367018825205314845875817193533529827837800351902529239517836689467661917953884712441028463935449484614450778762529520961887597272889220768537396475869543159172434537193611263743926337313005896167248051737986306368115003088396749587102619524631352447499505204198305187168321623238859794627245919771454628218399695789223798912199431775469705216131081096559950638297261253848242007897109054754028438149611930465061866170122983288964352733750792786069444761853525144421077928045979904561298129423809156055033032338919609162236698759922782923191896688017718575555520994653320128446502371153715141749290913104897203455577507196645425232862022019506091483585223882711016708433051169942115775151255510251655931888164048344129557038825477521111577395780115868397072602565614824956460538700280331311861485399805397031555727529693399586079850381581446276431
```

```
3858828529535803424850845426446471681531001533180479567436396815653
3261525095711274804119281960221488491482843891241785201745073055389
2871785792350941774338333150689823935442198880542933244037119486721
5543576548565499134519271098919802665184564927827827212957649240235
5075955582056475693653948733176590002063731265706435097094826497100
3873351747771340331902810557566793178947002411880309460403436295347
1997461392274791549730356412633074230824051999996101549784667340458
3268529603883011207656292459981362516523470939630497340464451063653
0416363082366924225776146828846179184322479343440607991788336067684
6711185597501
*Fibonacci> linear_fib 100000

4202692702995154386319005101293915131773915702632234503304716087198
3357314572762266339384772670136609625336617028583291866411622988222
1533373357414726861452220517796036021657629209679553065650253799831
4495026330500620719088898984643619599926476236108318505023749864703
8594910246866212417306827361157235516477242575475023524124687460748
5105335392343870354787001970158627451490394358177801241082646446182
3272924826749362282954004235923662667858166740323769532233540810434
2666616797388659593046520172457610944955660711670543016908957146048
8401367949139456649384464629891207894064459578250799792887873939298
5610180101343882600283820398139200927163512122969924839839463533622
3695998805936245483149559199827314406646068603896747678466307142387
9852006122094836019584282869489411495688799440753318862162470604312
7847588617179645038305434988777154546903665027721615391983839969917
8813439814194436880925011148181979840226116610460731803229235537660
3758649849997454020469379179304796322248613906014741363723033925344
3256608485701564784891053397926289664177821919706286664135844377802
8763525621223893534661028262352892632344883631994041592865608297285
9882120514749356839216074406481815370670988971910860608364879553400
4972817295465584196082602156304071901493325048737897039612705399999
4584598845863245763568504659028006821633602230188113830112087272531
2131480410618363074633391723478815558419942200436973479456746429373
1590299016321476910276047953145737133530620933370013274962876666024
0923781444622479118612427736288472117262538601036584279610764871633
3134720052848864474489786722218427986141051299691235751838042022652
2996351489992015929613369585992284921472840596751824858206083370375
1758706963474517424844808950896189553107994754733303323880745512298
9610511170391807481930946312967652658352325400619801548104415925441
8528609134434118408385912583154979720111735082542883519942050207089
7062913193237059361186997087674052153664968457182466406036383021770
5328485144253249745463099949757758310452101219511581460678590329580
7850171689359081326188293931649613908497264994236206250488864434829
4362951952328518385509235288894849877513810223200882551600336416459
1409282318882848545378075475179304890760002551214147063706640250187
2547952277141425050621431726133512376491993264368219026824677798278
0500122966776813381646679739913465787055127453439469440459707771918
```

79954073954276059158130792852127416568328282956215993174690589582184622213311123743528370969527697376659264209701430460262628089511749784920754475292453555824485037081169662961361437456685569229108353804447871639780202320840236093709222541404770096912820491605767773669730366536498834923648247283567638754132943595918097552972633716970745724835609042802322527268722270064679384555006784008088065427716480324882491803961614497071569285030997191355977454006784666563982465863641444396717248234018026243961781706483360371717605953991771045496130564628772749130877955433893243191369395281413812395013590807898227658488263250802618524850762756185439484575939335126157545942937679620715509540065149312865564443708655439771673542181054558453697618966155551804171330106742714737433118616111666800134742357715495644208822055262531207630247540274901297665161330833084658587180009012735858860387323290477368590430709381480698925172403629520614399575937151757870535045340264867187154085515769299670425865012093497499458084071483820091588806573538923913704024350619657151098348052986098255110975492885336115545257393042442696791158576341900753200960006019671663122779964553274124613827002842207651443127215907411334080034890407625849192846588580412869163865154796429776325124265243357090373279994090608028914324200216278403137569688793319327179281513651857484816495100553151506872893457040698364864908926339042055054701374510871324934497999408637497701279716028106056465342577613367511844757858239618765128089472486851278975083193508109837771994103670102484129663885788768053596123033251579123980971589651437698106274532939294318318049638382537268302829198532880152940714337232086321104061644358354382664864917218438210293517424553745505758090753355882025205602453784317543553853199425081002171431279765462841928691038739447760277789370517385306279744047722489086714748028855582994177874607142469064610509456252022128258329950255642602299905929046841595791167921512858860328296849951271553446508878296327611803906735633266224123093687573746634252135170679694981289617944100773168818654945023743438512318640838968116604957523873700674044602618648483425343480172570393405927238389192205890319989061851984210383082164165286102068738694744762282270671562080897567280776109321283700776536592362521370175610492010490647926435022313362357037672820456385127428485063107497131400081807697966418670920862707517582909049383024204603334049254656583584442952963370887765452655876066062446937832610287823426705006187125841909320161763987566513486927154010085083647269535513915816758710288791204561224527095476616464634423436358506846144301437379657717790829502874496253094716740270948624055817414650104670627976517803754955899088266937466501789934094158017587793129776426891242809581667986926232813625576760050205090619888399123409239688442164270550778132287725184933490420599495679264605659110257760825014193077871965880962249652458201556815237466699463967549092715904279567434739404713888403457003339771465555694947195071035775250811866532575065892797554153301608090613307672021657715401873436452152566925919864525536834568235530705981621300855049045749462125005586100312191

55224527412244517230546007648956874403781163712048894248144245738388122164699392123013082293765621951481198747805928300611916585469727144423076654840725787482060121836224986639372150974752653753967735395851984476957732214244609798081017294862506075465564785057434857467376092365663458265549569450080494662560068300330396979197272897854877289869170023334483811114362442199724834837339381791454083836674113793067254363060731858421388931047329746129464330913406615122087253374620231774863799904011403703504128611269026473639339727381792431574550310718602904646516848966499148495583627521027839809598906735345377174703544236535544574712744792321404060396945172845120264505187739366087820740427979492937157133826232579378916797946108593141509568348629616240187575589413012040737867171243772918848752012191291081032674915409604064875424892753452196207410242529191376694436903213210355093338224539955838775790474551175896508802785493729953647184656663763691900135139867974638850946840528454914075647948605645792064942961782974196194610970362930453091346671623844214882018944958150302976635357970338869992036590532404737312054965564334186262277707438964642317957967457795634132283078491602681354845462336294905591488533788039843697491807133608958002012311099161455612789712862952845820906795219432697689699775212704233642924681873485383474288933098056010335792228867656648020661465244444346177401172126439820562310243124840816832768854775304579417636166590305855258136611924901969916280249645408605332127506284062339817305401939875935121571198730663617841028308977914673241823990233067699411191624958418482129016174188490744574259103285106768945456293864755738899456354943259853902533118320431113938205766012422223175295440659963277854864852984742083179611314290857560405710728212958906402305931907418946110255773607562277250486349227222087557966804673075308805002278223894683614028706639026079344344434686294556396103123818136981228794370785368388506599135026980613330934094961793563094627199853201613193153865108452721786814140586572232010017459153962501788701203805098164119780148853693024087086122573636233064502251961233974992366640735451891044862799462188815582042386580030801106890097238687055452552110829965059161245343415161008354431626917063844116549102496993880762448490819042269804160074405392450174036209847483710949092804116773547479425117696723576173189547660596984813143044906939144736572025112530913402350316579592727107140204745792181312643267980166276213539787318686784516646409043739461341058341866130194515206728071388017201750465345177683567166246451096874716792230890148549731651715426719134701779009646503713855713690614979894889941888457297282875954463903599095148708206058572338514158627674241149795154024155999938386472330121596245377182900950030617269384713599791193454044355416529598060101173170994161856361550265787123369355532596323286384309305409145251652049876541298945371331714297379550668239431873282678293074955266864563208743768540095516367311985558520002873697087726788222240830142818294228089082545681401285437906803150262330031355435708411061568390519438584930325978406002583613274547851416495855586226252253360321671709941618563615502657871233693555325963232863843093054091452516520498765412989453713317142973795506682394318732826782930749552668645632087437685400955163673119855585200028736970877267882222408301428182942280890825456814012854379068031502623300313554357084110615683905194385849303259784060025836132745478514164958555862262522533603216 7

476698162042182733938520659373909688525278714540614267265826217757365504832826736859726921745784565192436195150513429476781385542250131264079646974464975869632268016274093312319995315408701199104881239311172712335296437902042142058375699484961276681889437669707324051106480371714625822888567683321085870260352984651037084497253919485107416824237985733807699179323184889784653349547151396087359431148393232317497136900767515061983191875262963294069878067300582894318636140853643354518251362173336646993432586033590160929285044356195560151952535903838669632931421195319086416029491768186604338108617856893658652762599159576463089057484057012421874561183369156744400271989993859435242606882312306519905393345015702439093202555708305829478488168303432296270227178110341320372986862855706153836192552748513974514717282608166467373122237687789655395096239211535388318181125900905930703765521017233830956585103471622343721470945239512383177898180368057660537081723959944939544511007394960086606314007369726809860353148028035639179259295514035556750901346225565651200500040388964695765286839375894326641359638758981952190977233092636192929286884271506009538217214933792744247421149897522229185157042978227719340975109319968827282928000446381475380613389725119957350959904451122273439971539673040298002820751768235827912095052102735663217352194962362184727464644960226149981751908352203306342021592050720598795035239228195869165158542449929474548106605050117625206972350963702019752601985260677384221896524605299854929995338755313295340940138314907468467139499677720572636120372107374342868057717426582051294080969959563531911282739990097960539607984437523116507734719777809570433883645270239644294309466315623904013251732165782921533830102558013937152598856329347755255719232808344589151922043864806081186297548489573356845798244178775602686253784216073315926315117006546664792964827288742401180295584471128695425297779253841606405927427313508860836696395679355243026261886885752006481760497725961130073922590764478449734883006577227566577741463644369609395375368102520301197046454171341193397573510378873839276937556550684521053667831828394540040197553472129245038333600381738567892128171380274979022554327432617244384438351916592795000171368253675854672346903373369846405951324771607653042856523597989801868226004994815042879710747667645929446038640988204665343974190422108476050665162566607895047007370036487826848781094370412378082086720321102798363733609058654381860297492878115548426614641514458315939170095759152419285874646232579928980821266399189797920309651377003004333255996575709580488051628031829131789228353961180609648076058832439883385192787126630843743665519645114080093624785226304263033325302142263623395355864654556795200038674005641384150331650044319688279186208951204048719083431831872080561388048272209950449676983025946244664338734008755934808078312171484872268533770220526080803825612965371213092814286832673202454734191840954326577075184602797070105815430474260354129521452999063598800011317197504331666954239837411800058621274994165791632889639236400118340323406940428738845184765282641012203360562432312163556897598636974964759651 

028711885857843551016875046246365393245524901540321405698655473042 4
595463633116465552163718390641441530975871281690612142586596351957 2
977922422193998635333872816316775329382888285388337632233238036645 0
894998747380600574088983607539281237219292218879353578281400690656 3
477820288750788181324078262439499226028832489914572737403078219099 1
736334686587324734925180822573968054570812461164995383965527453691 4
873409526301775673649365031050400120837333235076971527125231473234 3
371598238115498290209252432698108537214871783872236284696808583169 3
638384498765389697888945985515666277324765372113023407091352370001 3
328165692932769007814710786430979065784477931789605431432050244595 9
789408721143382595725905067611408377757766694987903449842361683356 5
457642039910618453618935750101710017725620355769563378953758665795 7
853193491667125067776002011167198244820679621382136374799960783889 4
747697685437117948202138934986118174082899084739771156621957573016 6
769531616390028616437744333987950465924649624221361582556979790545 4
982203659191586049120207156356310403155389866280413309663767214109 1
404518203202405716711882043446479392874980381935293814505434327576 4
367252195300947748256245931417922605137017689402364247601489831981 2
498818156593637676255006576938058747631406154481557792125352765262 3
573043368919268525405452626241191101615635779566696006590853227700 2
703275187121877171252803456126830235101667545124754877133773716679 5
514204267034673510427911733846292378010277610416833716861842537265 6
203751373284353363356516322014511878286982312595142110620136454193 0
300041723532889519178318955856530899618414036372766242613462981832 3
667185162276129290644748426426707202283516846142659205167622312078 3
082873649567475314773182809567925095745518320864072003407328047697 7
250429065166775862662137940822216639672500136596904942830895605490 9
512559689879221017746588179487743452315903585260321833096670991125 8
339348544848724784898998655510616338928668999670022545572520119246 8
963399245341930568628287266154554977591171677234870273109994284553 0
662906292309446065059613248595504182779386726685195646351040786178 3
979941387203353282784319564570900789300894702792354810518440182172 7
997576067980745684603001508138923452502105044934874146471722564995 3
671648212368660371399041943343776223988666385192175677262831838538 3
267256743980731998946648066449320332566724813016944238626026093859 5
262185473829829655664167239056097352374183107195265095683795670742 6
467406838497344751696859357892196424161249791143980395906669903749 7
600507857633103610020722599825555066260311254332369339553635821828 9
831122170914531594780311067952757440640496806878780414499789323112 4
540042356279079504960535323207987979466036248168633614842978382950 3
833009643088822457573493742883268731131033564532896600482935896139 8
637161232521561088632564699206653873837378293823902017672262403793 2
326437236496365765501744424019911126611218540395116166122909793543 5
094249668214817688676151993353930237597426829520797054969020159710 8
189357323757477303432824688714733897849865805712867879245604618677 3
275721041013928045946017349464511989767069540269263259770854119190 1
569104599555220735129313818716140549908490955484541943493191734443 0

522376469392263403392646510051744603644570754463847534733966705557908398431759668907930084657438598561912663124717703955390006624505825330361779064559334274155743297547382435851687558291288934724053570713445458222971933730758205897144472889089781340754148110731186122936859973310507122731462503393873897738479354688284953678332216656064611036606566593132311955013694851138439291211561574498547816003618955409622344616927979949484656636148330699173829550144656063429037359575915469558527253860731802499652816768425443425889739073884352974610591022393415807247569583300801154557691634376230748190087515083604267146590376806793177153184509444555078218804371479017254048737590623414381181091777959894452851110954312451007661884236739962192456077969511122779364407990415119514440661546965008576217977093190909583132987787434522165255034445404702763169714368762469620043258522003356436848692882916179287819142433475654756287226124351664343691962489756968151414727888765534977698006480871154761766945959111922927429454384882497333914219571616022292224906364296597972099547195800025036746754894650280064556031370644217419634227831878046942804047499799250678440344926170781018652689185600559289478435620230506562090798332784054409933754325466268996342381853087817159138132409688240004900823384875229448700461035381883418217015178779468503346298941683080899357908203402596981571019152124322599142297993961017488797887830281192191238727922353623295034097047026384576950518160676657044486620981891268715888390683926816266566045504069411325517726301593005468118625314041176546254147484954209655218329262550835896721811635441831388710685040170614684768874135589248534951236185901475459798978133816784134989257021373591925918938505117120958353354468195902418584460471378852709468342796348047587156573415690896444525395795251837597314216029681638412088153036099966996188639795041806972170297532612770372422177442399757860752244275767990624899121340496474889285918528422833247118694861120447658840271122911279114792252504327153033684366578150172821878605530320299879588079217379263325468352926442085153146379821322093120797035112094626071007302778912072566743084284949202405450391750643121078577065441187219944044713259661653295743027549410956701543326099389775148130009541828402651298935227760199983004106141718321080195789503790459326081276788138994738324227512564849341286531071052868689648644035294619832639219517599262158567853308180854622517432057724041779265437756658172231185312218755767693801703046218276691467762250144758754615144514318255923066943062603732646568363555326160331715437535324349374224250967669972209569182920399221779749244067926548005910615674789500542998017635444037775787063172034013555643683051468017023738859881563516237835944674355662760135149674301440420415171928872374647867382781971827035983886036616187357262700493342349286117240506228275121275481554264228471521735497145802473814776450360970418983518376319628772221549149786891339743058279126510663209664415284380540087275689926332447230094477529152928444134842867829072123694329242420883604363950605506758772124314102728605283875613848102406526257247712999775694643500056233401551

```
22088652827186955482430003765644790848939010005368410575223632803708388974966754934111198962884134376481657627555162493858197998761905934105965887729537505921354802824063847997251499460206485037420745024330993731647827119117031718785024921562817625090920957441290270496650891858244409518490868362448613027462096347537932263188167152611078126073281106201153449776581573049877867738275450173165896627327221710231020444689102004159483035267539082000099323700359160202697273589672875968000924313510899759597717396707377055456353397877628307096526435678705918396664011715402562042282810600679441091796539416853650052727243245369973451155161758421415997019097554830734246349425709110977253628963700518310446086702859049154897854518616404859671218321879126032244468656745720029096300907327964193810873683005770084885498763264242012257756113517921120697060985848063753001433032062979008560113174534602168985499355344812429007061967707035833467717189296096500460819064538137446492703889306501956180604625773009506220644692348349311631968718394606882313994056149928968895536885444260159789175948852213048333203788826561558879638240876906542931930466777690810720489086401931338095243898717372372530055217475244159479221137666528058884383960541475674869873371673551661045430428991319934308209916501596476562084563243630817817126022170163477755641114418035921930884712892557953936428778244536540481659503232237881780602859055827000867386848837503377149809171644446930603083902259623799462727132897689299367245365532334824657942127340270705653885119362157626562988646844353708235718480842471383975257783101287855534457080267659017382060818735190509984996309659107934323668138539702028213949494061344899524539088309035457877956639633534630661764226022021057756219237819574946297164368153492108570727063164273284535774985139946371594189537208267671471332050925700993356729526111636280008058244097546017085175764114440793957202152484441164215559578287487454242260118448659067156641108028682905182106128604044547505114389225444391434868194881337928254493717003965533900634644833101037271446752357724260351165699515773967642913572637535873501670470613312527450847505213359793783861950188821929180325547521785176580032177426333023524832481849348447808439788218713542427529133485690077625718966956228062966168347006680212942597861705301218647978947398135250300005080179027045518797793505294699210802788329649160385012184940049568696199707027524548531041667993720684674742262714439058761479088895841824160373748908700112817860631624151944290572786670975635303518603594047587697599520048169851304461861034981185133385903366316421187501308774815870676228749311300865732973446153743975659991079304994892588234011350360387511421993302025047776808575499810068887160787732953050111241465547976334636931551115834357110038285979669707537501
```

お、少しタイムラグを感じた（そういう問題ではない気がする。紙面的に）。しかしここで止めたら上司が廃るというものだ（意味不明）。

さらにいくぞと思うと、

```
*Fibonacci> linear_fib 100000
```

（省略）

3160476873866898734458419122053091354986341086926062241724721085805
3402584856937058789023065222252698184719361701572155753620355245606
8969866014863500715572554770147092754408476591962812151867770833598
8936062847837947512676558578687034983009115928520465917989984761427
9118928432164912845312007451390682346145808954626634866477461203823
0972762871588830264026154213890485679867290129263139050605467325382
1878403819632133936040342593917008325204814033167410342007459834688

（省略）

1217318186394872967387668120301497349768866780529458909899936441987
5512357488154033507026858713867017225248753435025802794318888601679
6269381830160385478113646266046526438745647943332427830612881315921
1879681425442010157485889660057486428152755110234293217903136427199
5883868830622612262198090806737563006621563898332865559917649355629
4452650649511393459616089461578449813330453556710264100580535142277
359244926937501

とかなり待たされる[*12]。

これには上司も不満顔である。もっと大きく、もっと長く、もっと速く、という人類とその上司の願いを叶えるためにはどうしたらいいのであろうか。

## 1.5　行列の冪による計算量 $O(\log n)$ によるフィボナッチ数列

というわけで最後に、フィボナッチ数列を $O(\log n)$ で計算する方法を実装してみよう。

そのためには実は、フィボナッチ数列の次の数を計算することが行列を作用させることだということをもう一度思い出せば良いのだ。

$$\begin{pmatrix} \mathrm{fib}_{n+1} \\ \mathrm{fib}_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

よって、フィボナッチ数列は

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

を計算すれば良いことになる。

同じ元の冪を効率よく計算していくためには、できるだけ二乗をすることで済ますことだ。

話を簡単にするために、$2 \times 27$ を計算することを考えてみると、

[*12] 待ち時間と表示を紙面で表現しようとすると、**暗黒通信団**さんの丸パクリになってしまうので諦めた。

$$
\begin{aligned}
2 \times 27 &= 2 \times 26 + 2 \\
&= 4 \times 13 + 2 \\
&= 4 \times 12 + 4 + 2 \\
&= 8 \times 6 + 4 + 2 \\
&= 16 \times 3 + 4 + 2 \\
&= 16 \times 2 + 16 + 4 + 2 \\
&= 32 \times 1 + 16 + 4 + 2 \\
&= 32 + 16 + 4 + 2 \\
&= 54
\end{aligned}
$$

となる。これは**ロシア農民の掛け算**と呼ばれ、ロシアの農民はこれを使って、×2 と ÷2 だけを覚えて掛け算をしていたという。これは我々が通常 ×2 ～ ×9 と ÷10 だけで、掛け算をしていることに対応する、と考えればそれほど不自然ではない。つまりこれは、2 進法の筆算をしているのだ[*13]。そして、筆算のいいところは、ただ $2 \times 27$ を 2 を 27 回足すよりも、大幅に計算量が減ることである。例えば $2 \times 54$ は $2 \times 27$ よりも一回しか計算量が増えない。すなわち、この方法の計算量は $O(\log n)$ になっているのだ。

そしてこれはそのまま冪乗にも使えることは明らかであろうと思う。

実装してみよう（あえてここではしていないが、ループ不変条件を考えて、末尾再帰にすることもそれほど難しくはない）。

```
matrix_product (a,b,c,d) (e,f,g,h) =
  (a * e + b * g, a * f + b * h, c * e + d * g, c * f + d * h)
matrix_squared a = matrix_product a a
matrix_power a n =
  if n == 0
  then (1,0,0,1)
  else if n `mod` 2 == 0
  then matrix_power (matrix_squared a) (n `div` 2)
  else matrix_product a (matrix_power (matrix_squared a) (n `div` 2))
fibonacci_matrix = (1,1,1,0)
matrix_fib n =
  let (_,_,a,b) = matrix_power fibonacci_matrix n
  in a + b
```

二次行列の積を定義し、ロシア農民の方法で冪乗を効率よく計算している。プログラミングも Haskell も知らない人にもそれほど難しくはないと期待したいところだが、どうであろうか。

さて、それでは試してみよう。もう紙面で表現することはやめるが、1000000 番目のフィボナッチ数も一瞬で計算できる。10000000 番目も一瞬のタイムラグがあるが、あまり気にならない。100000000 番目はかなり待たされる。しかし、長い数字の表示がいつまでも終わらないことに比べたらいかほどのものでもない。

[*13] ×2 と ÷2 はコンピュータ上では**ビットシフト**と呼ばれるものである。おそらくロシア農民はビットシフトをネイティブでサポートしてたんだね。

## 1.6　時間計測

さあ、最後に真面目に時間を測ってみよう。GHCi で

```
*Fibonacci> :set +s
```

とすると、実行時間が表示される。ただし、長い数字の画面表示自体に時間がかかってしまうため、次の関数を定義して計算結果を虚無への供物とした。

```
import Control.Exception(evaluate)

devnull f x = do
  evaluate $ f x
  return ()
```

`Control.Exception.evaluate` という関数が必要なのは、Haskell は**遅延評価**という、必要になるまで、関数を評価しない仕組みがあるので、これがないと、`fx` は計算されないままになってしまい、実行時間が計測できないからである。

これを使って、`simple_fib` の実行時間をはかると、

```
*Fibonacci> devnull simple_fib 21
(0.03 secs, 12,292,184 bytes)
*Fibonacci> devnull simple_fib 22
(0.05 secs, 19,854,152 bytes)
*Fibonacci> devnull simple_fib 23
(0.09 secs, 32,083,992 bytes)
*Fibonacci> devnull simple_fib 24
(0.13 secs, 51,875,400 bytes)
*Fibonacci> devnull simple_fib 25
(0.21 secs, 83,897,752 bytes)
*Fibonacci> devnull simple_fib 26
(0.34 secs, 135,708,328 bytes)
*Fibonacci> devnull simple_fib 27
(0.55 secs, 219,546,040 bytes)
*Fibonacci> devnull simple_fib 28
(0.86 secs, 355,193,208 bytes)
*Fibonacci> devnull simple_fib 29
(1.38 secs, 574,676,976 bytes)
*Fibonacci> devnull simple_fib 30
(2.23 secs, 929,807,672 bytes)
*Fibonacci> devnull simple_fib 31
(3.61 secs, 1,504,423,360 bytes)
*Fibonacci> devnull simple_fib 32
(5.87 secs, 2,434,169,464 bytes)
*Fibonacci> devnull simple_fib 33
(9.46 secs, 3,938,531,736 bytes)
*Fibonacci> devnull simple_fib 34
(15.31 secs, 6,372,640,296 bytes)
*Fibonacci> devnull simple_fib 35
```

```
(24.70 secs, 10,311,109,264 bytes)
*Fibonacci> devnull simple_fib 36
(39.85 secs, 16,683,688,344 bytes)
```

わかるだろうか。計算にかかった時間がだいたいフィボナッチ的な数列になっていることが。

また、

```
*Fibonacci> 15.31 / 9.46
1.6183932346723044
(0.01 secs, 85,392 bytes)
*Fibonacci> 39.85 / 24.70
1.6133603238866399
```

というように m 前後の比もだいたい黄金比になっている。

続いて `linear_fib` の実行時間を計測すると、

```
*Fibonacci> devnull linear_fib 50000
(0.17 secs, 131,766,504 bytes)
*Fibonacci> devnull linear_fib 60000
(0.26 secs, 184,408,592 bytes)
*Fibonacci> devnull linear_fib 70000
(0.35 secs, 245,830,712 bytes)
*Fibonacci> devnull linear_fib 80000
(0.49 secs, 316,047,352 bytes)
*Fibonacci> devnull linear_fib 90000
(0.53 secs, 395,012,840 bytes)
*Fibonacci> devnull linear_fib 100000
(0.67 secs, 482,778,216 bytes)
```

おお、一次関数っぽい。

最後に、`matrix_fib` も計測してみよう。

```
*Fibonacci> devnull matrix_fib 1000000
(0.05 secs, 7,040,848 bytes)
*Fibonacci> devnull matrix_fib 2000000
(0.12 secs, 13,270,520 bytes)
*Fibonacci> devnull matrix_fib 4000000
(0.21 secs, 26,431,832 bytes)
*Fibonacci> devnull matrix_fib 8000000
(0.45 secs, 52,752,672 bytes)
*Fibonacci> devnull matrix_fib 16000000
(1.02 secs, 105,391,576 bytes)
```

ううむ、どう見ても線形だなあ。なんでだろう。たまたまこの辺りが線形なだけで、値が大きくなっていったら傾斜が緩くなるのかな？

```
*Fibonacci> devnull matrix_fib 32000000
(2.28 secs, 210,671,608 bytes)
*Fibonacci> devnull matrix_fib 64000000
(5.08 secs, 426,766,400 bytes)
```

```
*Fibonacci> devnull matrix_fib 128000000
(11.67 secs, 842,311,272 bytes)
*Fibonacci> devnull matrix_fib 256000000
(25.81 secs, 1,684,500,672 bytes)
*Fibonacci> devnull matrix_fib 512000000
(57.63 secs, 3,368,878,784 bytes)\
```

ううん、それどころか、線形よりわずかに速く大きくなってる。なんでだ？

正直私には分かりませんでした。詳しい人に教えを請いたい。

でも、多倍長、すなわち無限に桁が大きくなる整数を扱うのはオーバーヘッドがでかいというし、色々と遅くなる原因はあるのではなかろうか？

なんか釈然としない終わり方だけど、時間切れなので今回はここまでで、まとめに入ろう。

## 1.7　まとめ

前半にフィボナッチ数列の一般項を代数的に解き、後半では数値的に解いたが、どちらもフィボナッチ数列の次の数を求める操作を行列による作用と考えることが、役に立った。

重要なのは作用と、**作用素**として、それ自体が対象であると考える抽象化のプロセスである。

普通我々がまず注目するのは作用されるものである。作用するものを注目する、と自分では思っている場合も、多くは作用そのものではない。

作用そのものを対象とすることは、思考の訓練が必要であると思われる。

実際、歴史の中で「対称性を持ったもの」はおそらく歴史以前から注目されているにも関わらず、「対称性＝ある作用のクラスに対する普遍性」を取り出すために必要だったもの、すなわち「作用そのもの」に注目し始めたのは、19 世紀のガロア以降でしかないのだ。

この抽象化によって、数学は多くのものを得た。

**群**とはある図形や集合に作用している作用を集めたものと考えることが出来る。

そして群を調べる最も重要な方法は、群を何かに作用させてみることである。

そうすると群の**表現**と呼ばれるものが現れる。例えば群を線形空間に作用させると、群の**線形表現**という最もメジャーな表現が現れるのだ。

作用そのものを考えていない状態、というのはいわば、群の表現を一つしか考えていない状態だ。

作用そのものを考えることにより、作用させられるものの選択肢は大きく広がり、すると最終的には、最初にこの作用が作用していたものへの理解も深まることになる。

例えば、群を考えただけで、最低 2 つは作用させられるものが増える。

一つはその群自身である。

群自身に片側からその群を作用させることにより、ラグランジュの定理、すなわち有限群の部分群の位数は、元々の群の位数をわりきる。

という定理がわかる。

そしてもう一つの例もその群自身であり、自己共役作用により、群は自分にまた別の仕方で作用することが出来る。

この考察から**正規部分群**という非常に重要な概念が現れてくるし、シローの定理のエレ

ガントな証明も見えてくる。

こうして、作用自体を考えることにより、代数学の様々な理論が深められてくる。

それによって現在では、それほど深いことを考えなくてもある変化があった時に、その変化の中から作用自体を取り出して、その代数がどんなものか考えるだけでも、代数学の様々な知見が応用できるようになっている。

行列の対角化もまた、行列の作用が単純（固有値倍）になるベクトル（固有ベクトル）を探そうとすることで導入できる概念だ。そこまで分かれば、三項間漸化式において、$a_{n+1} - \alpha a_n, a_{n+1} - \beta a_n$ ($\alpha, \beta$は特性方程式の解) の数列を考えてこれが等比数列になることから解く、という発想も自然に思えるであろう。

素晴らしい。

計算数学の方からも、作用自体を取り出すことの意味を考えてみよう、

フィボナッチ数列の計算において、数列自体に注目し続けていたら、$O(n)$ より早い計算方法はなかなか見つからなかったであろう。どうしても順々に計算していくように考えてしまうからである。

数列の変化から、作用自体を見つけることにより、$O(\log n)$ への道が開けたのだ。

そしてこれは、他の作用に広く応用ができる。

どこまで広く応用ができるのか。そう考えることで、代数との接点もできる。

今回の手法が適用ができたのは、単位行列という単位元があり、式のどこから計算していってもいい、すなわち**結合律**が成り立つからである。このような代数を**モノイド**という。もし結合律が成り立たないのに、この手法を適用しようとしていたら危険があることがわかる[*14]。

今回は同じ元の冪乗だったが、これが異なる元の積だったらどうだろう。前から順にやるしかないであろうか？

もし、同時に計算できる装置があったら話が別になるだろう。4 つ同時に計算できるなら、列を 4 つに分けて、それぞれ別に計算して最後に計算をまとめることが出来る。そして、それが可能なのもやはり結合律が成り立つからである。

もし結合律が成り立たないならば、どれだけ同時に計算ができる能力を持っていたって、前から順番に計算していくしかないであろう（結合律を弱めた何らかの法則があるなら別だが)。昔ながらの計算機業界の例えを使うと「妊婦が１０人いても、１ヶ月で子供を埋めたりしない」というやつだ。世の中には並列化できないものが結構あるのだ。

逆にもし、この計算が可換だったらどうだろう。可換ではなかったら、列の分け方や結果の集め方に注意が必要だろう。しかし可換なら、そこを雑にできるので、さらに早く計算ができるかもしれない。

このように、変化を作用と考え、その作用の代数がどんな性質を満たすかは、計算のためにも結構重要なのだ。

数学もプログラミングも、それぞれ抽象化を重要な道具として使っているが、実際見てみると、目的が違うので、抽象化の仕方や使い方も結構違う。

しかし作用をそれ自体対象としてみることは、数学では代数の根幹として、プログラミングでもデザインパターンにおけるコマンドパターンなどで、よく現れる現象なので、共

[*14] プログラムに証明を添付して、モノイドである証明があったらより良い最適化を自動でしてくれたり、モノイドである証明がないのにこの最適化をしようとしたら警告を発したりしてくれたらいいですね。

通部分として、少し考えてみたい、と考えたのだ。

# 参考文献

[1] エイブルソン, ハロルド/サスマン, ジェラルド・ジェイ. 和田英一 訳. 計算機プログラムの構造と解釈 第 2 版. 翔泳社. 2014.

[2] Lippvača, Miran. 田中英行/村主崇行 訳. すごい Haskell たのしく学ぼう！. オーム社. 2012.

[3] 平岡和幸/堀玄. プログラミングのための線形代数. オーム社. 2004.

# 第 2 章

# 可換環 次元の奇妙な物語

龍 孫 江

## はじめに

現代可換環論を「切り拓いた」のは, 誰だろうか？ この問いに答えるには, いくつかの論点を整理する必要がある.

可換環を代数系, すなわち代数構造 (然るべき性質を充たす演算) が定義された集合として最初に捉えたのは Dedekind であろう. Dedekind は **既知の集合の部分集合として対象の「要素」を再規定する** という発想[*1]に基づいて, 可換環のイデアルを定義し, Kummer の謎めいた理想数の理論を見通しよく整備してみせた. イデアルを定義するにはその器となる環を捉えている必要があるわけで, 可換環論は Dedekind に始まると言っても誤りではあるまい.

個人的には,「可換環論」という理論の水源は D. Hilbert にあると考えたい. Hilbert は, 現在その名を冠して呼ばれる 2 つの定理 (**基底定理**[*2]と **零点定理**) によって, 可換環の精査が代数幾何及び不変式論においても重要であることを明らかにした. とりわけ, 不変式環の構成方法については, 未だ以て Hilbert の示した基本思想の範疇から出たものはほとんど存在しないと言ってよい. その意味でも, Hilbert は未だに可換環論を専門とする上で乗り越えるべき壁なのである.

Hilbert に続いて抽象代数の旗手の役割を担ったのは E. Nöther であった. Nöther は, イデアルの昇鎖律[*3]が成り立つ環において, 総てのイデアルが有限個の準素イデアルの交叉として表されることを示した. この業績を称えて, 現在, 我々は昇鎖律を充たす可換環のことを **ネーター環** と呼んでいる.

しかし, である. Hilbert, Nöther が重要性を明らかにした可換環であったが, これをひとつの分野として確立したのは Nöther の高弟 W. Krull だろう. Hilbert や Nöther が

---

[*1] 最もよく知られ成功した例は, 有理数の切断による実数の構成であろう.

[*2] まったく個人的な話であるが, Hilbert の基底定理は筆者が一番好きな定理であり, 可換環論を専攻する切っ掛けともなった定理である.

[*3] この昇鎖律という性質は Hilbert が不変式論の第一・第二基本問題を同時に解決すべく導入したものであった. 当時の不変式論の基本戦略は **不変式系とその関係式を膨大な計算によって導出する** (結果として有限個しかないことが解る) というもので, 斯様な抽象的な性質から不変式系や関係式の有限性を導き出した Hilbert の仕事は, 不変式論の帝王 Gordan から「神学」と言われた. なお, この「神学」という表現は, 最初は非難として, 次に賞賛として 2 度用いられたそうである.

明らかにしたのは, **可換環の考察が代数幾何や不変式論などに重要な役割を果たす** ことである. これに対し, Krull は, 局所化, 完備化をはじめとして, 現代の可換環論が考察の対象とする概念や手法となる操作を整備した. これらは多項式環や (代数体の) 整数環の場合に既に創案されていた概念を改めて抽象的な可換環に拡張したもので, 言うなれば Krull は **可換環論という分野の基礎をほぼ独力で建設した** のである. Nöther を 抽象代数学の母 と称えるならば, さしづめ Krull は 可換環論の父 と比定されよう. そして筆者は, 冒頭の問いの答えとして Krull を挙げたいと思う.

この Krull の限りない可換環論への貢献の中で, 歴史に特筆大書されるべき第一のものは, ネーター (局所) 環の次元論の確立であろう. これは, 当時の考えられる技術の粋を結集したイデアル論の大輪の華なのである.

## 2.1　ネーター局所環の次元定理

さて, 次元である. と偉そうに始めたが, 筆者は, そのあらゆる用法に共通する明確に言語化された定義を持ち合わせているわけではない. しかしながら, それが何がしかの「空間の広がり方／大きさ」を示す尺度であることは相違あるまい. したがって, 可換環の「次元」を規定するには, **ある程度評価の確定した確固たる対象から尺度となりうる値を抽出する方法を見出し, それを一般化する** のが筋道となろう.

可換環論において, もっとも典型的な, かつ重要な例が (体に係数を持つ) 多項式環である. 整数環はこれに比べると, (可換環論の方法と文脈においては) 少し綺麗過ぎる[*4]. 幸い, Hilbert の零点定理によって, 多項式環 $S := \mathbb{C}[x_1, \ldots, x_d]$ はアファイン $n$ 次元空間 $\mathbb{C}^d$ と対応することが知られている. $\mathbb{C}[x_1, \ldots, x_d]$ の極大イデアルは

$$(x_1 - c_1, \ldots, x_d - c_d), \text{ ここで } c_i \in \mathbb{C}$$

と表され[*5], これを $\mathbb{C}^n$ の点 $(c_1, \ldots, c_d)$ と対応付けることで

$$\mathbb{C}[x_1, \ldots, x_d] \text{ の極大イデアルの全体 } \simeq \mathbb{C}^d$$

が得られる. $\mathbb{C}^d$ の次元は (測り方にもよるだろうが) 概ね $d$ と規定してよいだろうから, $d$ 変数多項式環 $\mathbb{C}[x_1, \ldots, x_d]$ から (多項式環に特有の事情を極力使わずに) $d$ を導出する操作があれば, それによって一般の可換環の次元をも導出できると期待されるのである.

### 2.1.1　クルル次元

「$\mathbb{C}^d$ の次元は $d$ である」という主張について, 数学科の大学生は初年度の線型代数でみっちり躾けられる. この場合, 次元を規定するのは基底である.

1 組の基底を固定する. 何でもいいので, 簡単のために標準基底 $\{e_1, \ldots, e_d\}$ を考えよう. つまり $e_i$ は, $i$ 番目の成分が 1, 他の成分が 0 というベクトルである. 順番も込めて,

[*4] 数学において綺麗過ぎる対象は, 話がうまく行き過ぎて本質を見失う場合が多い. さりとて難しい対象では難しすぎて値の候補すら見つからなかったりする. 玄妙微妙な例を見出すのも難しいのである.

[*5] この形のイデアルが極大イデアルとなることは容易い. 一方, 代数閉体上の多項式環の極大イデアルがこの形のイデアルで尽くされていることを保証するのが Hilbert の零点定理である.

この基底を指定することは, 概ね部分空間の列[*6]

$$\{0\} \subset \mathbb{C}e_1 \subset \mathbb{C}e_1 + \mathbb{C}e_2 \subset \cdots \subset \mathbb{C}^n$$

を指定することと言えよう[*7]. 一方で, この旗は多項式環の素イデアルの系列

$$(x_1, \ldots, x_d) \supset (x_2, \ldots, x_d) \supset (x_3, \ldots, x_d) \supset \cdots \supset 0$$

に対応する. この素イデアルの階段を一段下るごとに空間の次元は 1 ずつ上昇していることを鑑みると, 次の定義が得られる.

**定義 1 (クルル次元)** $R$ をネーター環とする. $R$ の素イデアル鎖

$$P_0 \subset P_1 \subset \cdots \subset P_r$$

の **長さ** を $r$ とする. $R$ の素イデアル鎖の長さの上限を $R$ の **クルル次元** といい $\dim R$ と表す.

クルル次元は文字通り素イデアルのみによって定義される概念なので, 素スペクトル (素イデアルの全体) に関する記号や用語をいくつか準備しておく. ネーター環 $R$ に対して, $R$ の素イデアルの全体を $R$ の **素スペクトル** といい $\operatorname{Spec} R$ と表す. 同様に極大イデアルの全体を $\operatorname{Max} R$ と表す.

$R$ の素イデアル $P, Q$ が $P \subset Q$ を充たすとき, $P$ に始まり $Q$ に終わる素イデアル鎖

$$P = P_0 \subset P_1 \subset \cdots \subset P_r = Q$$

が **飽和している**, またはこの鎖が **飽和鎖** であるとは, 間に素イデアルを加えて鎖を伸ばせない, すなわち各 $i$ に対して $P_i$ を包み $P_{i+1}$ に包まれる素イデアルが存在しないことをいう. さらに, $R$ の素イデアル鎖が **極大鎖** であるとは, その鎖に含まれないいかなる $R$ の素イデアルを添加しても鎖にならないことをいう. 極大鎖とは, ある極小素イデアルに始まりある極大イデアルに終わる飽和鎖に他ならず[*8], この用語に従えば $R$ のクルル次元とは

$$R \text{ の極大鎖の長さの上限}$$

と比較的簡単に表現できる.

### 2.1.2 特性多項式の次数

多項式環の「大きさ」を測ろうと思うとき, おそらく真っ先に思いつくのが **単項式の個数を数えよう** という試みではなかろうか. 実際, 多項式環において単項式の全体は係数体の上のベクトル空間としての基底をなすのだから, その濃度は大きさの手掛かりになるに違いない.

むろん, 総ての単項式を一度にまとめて相手にはできないので, 漸近的な振る舞いを見る. すなわち, 次数 $n$ を固定した場合に, 次数 $n$ の単項式の総数を求めよう. これは問題

[*6] しばしばこのような系列を旗といい, 代数幾何では旗の全体を多様体と見做して考察したりする. これを旗多様体といい, 旗多様体の特殊な場合としてグラスマン多様体が, さらに特殊な場合として射影空間がある.

[*7] 若干だが, 基底の方が精度の要求が高い.

[*8] さらっと述べたが, ここにはネーター性の仮定が強く効いている.

としては「$d$ 個のものから重複を許して $n$ 個のものを選ぶ組合せの総数を求めよ」と言い換えられる. 高校の 順列及び組合せ において見られる重複組合せである. 公式にあてはめて

$$\binom{d+n-1}{d-1}$$

を得るが, これを $n$ の多項式と考えればその次数は $d-1$ となる.

ところで, 変数が生成する極大イデアルを

$$\mathfrak{m} = (x_1, \ldots, x_d)$$

とするとき, $n$ 次の単項式の全体は $k$ ベクトル空間 $\mathfrak{m}^n/\mathfrak{m}^{n+1}$ の次元に等しい. $\dim_{\mathbb{C}} \mathfrak{m}^n/\mathfrak{m}^{n+1}$ が $n$ の $d-1$ 次の多項式をなすならば, それらの和分である

$$\ell\left(\mathbb{C}[x_1, \ldots, x_n]/\mathfrak{m}^{n+1}\right) = \dim_{\mathbb{C}} S/\mathfrak{m} + \dim_{\mathbb{C}} \mathfrak{m}/\mathfrak{m}^2 + \cdots + \dim_{\mathbb{C}} \mathfrak{m}^n/\mathfrak{m}^{n+1}$$

は $d$ 次の多項式になる.

こうなると一般化は比較的容易で, ネーター局所環 $(R, \mathfrak{m}, k)$ においても $\dim_k \mathfrak{m}^n/\mathfrak{m}^{n+1}$ やその和分

$$\ell\left(R/\mathfrak{m}^{n+1}\right) = \sum_{j=0}^{n} \dim_k \mathfrak{m}^j/\mathfrak{m}^{j+1}$$

を考えることができ, さらに $n$ が小さい (誤差の影響が一定しない) 部分を除けば, $n$ の多項式 $\chi_R(n)$ で $\ell\left(R/\mathfrak{m}^{n+1}\right) = \chi_R(n)$ を充たすものが存在することもわかる. この多項式 $\chi_R(n)$ の次数 $d(R)$ もまた, 環の大きさを表す尺度の候補として充分な資格を持つと言えよう.

### 2.1.3　準素イデアルとパラメーター系

第 3 の候補のヒントは, 極大イデアル $\mathfrak{m} = (x_1, \ldots, x_d)$ が $d$ 個の要素で生成されているという事実であろう. 実際, $\mathbb{C}[x_1, \ldots, x_n]/\mathfrak{m}$ は体であって, 先述のクルル次元は 0 である.

**補題 2** $\mathbb{C}[x_1, \ldots, x_n]$ のイデアル $I$ に対し, 以下は同値である :

1. 剰余環 $\mathbb{C}[x_1, \ldots, x_n]/I$ のクルル次元は 0 である ;
2. 剰余環 $\mathbb{C}[x_1, \ldots, x_n]/I$ は $\mathbb{C}$ ベクトル空間として有限次元である ;
3. $\sqrt{I} = \mathfrak{m}$ ;
4. $I$ は各変数の冪を含む.

$\mathfrak{m}$ の冪など, $\mathfrak{m}$ 準素イデアルであっても生成系の濃度は一意的には定まりそうにない. しかし, 上記補題 2 からも解るように, 各変数の冪が生成するイデアルが準素イデアルとなるのである. そこで, 前 2 候補よりは (今のところは) 若干根拠薄弱ながらも, ネーター局所環 $(R, \mathfrak{m})$ に対して

$$\mathfrak{m} \text{ 準素イデアルの生成系の濃度の最小値 } \delta(R)$$

を次元の候補としたい. このとき, $\delta(R)$ 個の要素からなる準素イデアルの生成系を $R$ の **パラメーター系** とよぶ. 体の上の多項式環において, 変数列がパラメーター系をなすことは明らかである.

### 2.1.4 結実：ネーター局所環の次元定理

前節までに紹介された 3 つの値が等しいと主張するのが, Krull によるネーター局所環論の金字塔・次元定理である：

**定理 3 (Krull の次元定理)** ネーター局所環 $(R, \mathfrak{m})$ における以下の値は互いに等しい：

- $\dim R$： $R$ の素イデアルの極大鎖の長さの上限,
- $d(R)$： $R$ の特性多項式 $\chi_R(n) = \ell_R(R/\mathfrak{m}^n)$ の次数,
- $\delta(R)$： $R$ の $\mathfrak{m}$ 準素イデアルの生成系の濃度の最小値.

この定理ではあまり見えにくいが, いずれにも通底するのは **パラメーター系の暗躍** である. 次節ではパラメーター系の構成方法をおさらいしつつ, その暗躍の様子を白日の下に晒すべく試みたい.

## 2.2 素イデアルの集合とパラメーター系及び正則列のレシピ

前節の次元定理 3 により, ネーター局所環の次元とは パラメーター系の濃度 なのであった. とはいえ, そもそもパラメーター系とは然るべき $\mathfrak{m}$ 準素イデアルの生成系 として定義されたのだったが, 如何にすればパラメーター系が得られるのかはこの小文内ではまだ述べられていない.

次の補題は基本的であるが, パラメーター系の構成においては本質的な役割を果たす補題である.

**補題 4 (Prime Avoidance)** 可換環 $R$ の有限個の素イデアル $P_1, \ldots, P_r$ をとる. イデアル $I$ がどの $P_i$ にも包まれないならば, $I$ の要素 $x$ でどの $P_i$ にも属さないものが存在する.

次もよく似た, また便利な補題である.

**補題 5** 可換環 $R$ の素イデアル $P$ とイデアル $I_1, \ldots, I_s$ に対し, $\bigcap I_j \subset P$ ならば, ある $j$ に対し $I_j \subset P$ である. さらに $\bigcap I_j = P$ ならば, ある $j$ に対し $I_j = P$ である.

### 2.2.1 準素分解と素イデアルの集合

次元論において ネーター環 という仮定の恩恵はいろいろある. 第一に, **個々の素イデアルの極大鎖の長さは有限に収まる** 点がある. また第二には, Nöther による準素分解可能性が挙げられよう.

**定理 6 (Nöther の分解可能性定理)** ネーター環の任意のイデアルは準素分解を持つ, すなわち有限個の準素イデアルの交わりに表される.

後に必要になる素イデアルの部分集合について整理しておく.

**定義 7** 可換環 $R$ のイデアル $I$ に対し, $I$ の無駄のない準素分解[*9] を

$$I = Q_1 \cap Q_2 \cap \cdots \cap Q_r$$

とする.

1. $\mathrm{Ass}_R(R/I) = \{\sqrt{Q_i} \mid 1 \leq i \leq r\}$ の要素を $I$ の **素因子** という.
2. 包含関係により $\mathrm{Ass}_R(R/I)$ を順序集合と見做した場合の極小な要素を $I$ の **極小素因子** といい, その全体を $\mathrm{Min}_R(R/I)$ と表す.
3. $I$ を包む素イデアルの集合を $V(I)$ と表す[*10]. $\mathrm{Assh}_R(R/I) := \{P \in V(I) \mid \dim R/P = \dim R/I\}$ とおく.

**補題 8** ネーター環 $R$ のイデアル $I$ に対し, 包含関係による $V(I)$ の極小元の全体は $\mathrm{Min}_R(R/I)$ に等しい.

**証明** $I$ の無駄のない準素分解 $I = Q_1 \cap \cdots \cap Q_r$ をひとつとる. $P \in V(I)$ に補題 5 を適用すると, $P$ はある $Q_j$ を包む. 特に $\sqrt{Q_j} \subset P$ である. したがって, $P$ が $V(I)$ の極小元ならば $P = \sqrt{Q_j} \in \mathrm{Ass}_R(R/I)$. □

次の定理は, イデアル論とホモロジー代数的手法との架け橋となる重要な命題である[*11].

**定理 9** ネーター環 $R$ のイデアル $I$ に対し, $R$ 加群 $R/I$ の零因子[*12]の全体を $\mathrm{ZD}_R(R/I)$ とすると,

$$\mathrm{ZD}_R(R/I) = \bigcup_{P \in \mathrm{Ass}_R(R/I)} P.$$

この定理の本質は次の言い換えにある：

**補題 10** ネーター環 $R$ のイデアル $I$ と素イデアル $P$ に対し, 以下は同値である：
(1)
$P$ は $I$ の素因子である；
(2) $R$ 加群の単射 $R/P \to R/I$ が存在する, すなわち ある $x \in R/I$ で $P = \mathrm{ann}_R(R/I)$ なるものが存在する.

**証明** $R$ を $R/I$ で置き換えて $I = 0$ としてよい. $0 = \bigcap Q_i$ を無駄のない準素分解とし, $P_i = \sqrt{Q_i}$ とする. 無駄はないとしたので $I_i := \bigcap_{j \neq i} Q_i \neq 0$.
(1)⇒(2). $I_i$ の特徴づけとして, $y \in I_i$ に対し

$$y = 0 \iff y \in Q_i$$

が成り立つことに注意する. $x \in I_i$, $x \neq 0$, をとれば $x \notin Q_i$ である. $r \in R$ が $rx = 0$ を充たせば $rx \in Q_i$, ゆえに $Q_i$ の準素性から $r \in P_i$ でなければならない. すなわち, $\mathrm{ann}_R(x) \subset P_i$ である.

---

[*9] イデアル $I$ の準素分解 $I = Q_1 \cap \cdots \cap Q_r$ が **無駄がない** とは, (1) どの $Q_i$ を除くことができず, かつ (2) 各根基 $\sqrt{Q_i}$ が互いに異なる, の 2 条件をを充たすことをいう. イデアルの準素分解が与えられたならば, そこから無駄がない準素分解を得ることは比較的容易である.

[*10] これは, 剰余環 $R/I$ の素イデアルの全体 Spec $R/I$ と同一視できる.

[*11] 重要性のわりにこれと言った呼び名がないことを筆者は不思議に思っている. 知らないだけかもしれないが.

[*12] $r \in R$ が $R$ 加群 $M$ の **零因子**とは, ある $m \in M \setminus \{0\}$ で $rx = 0$ を充たすものをいう.

逆に $Q_i$ は $P_i$ 準素イデアルで基礎環はネーター的なので, 充分大きな自然数 $m$ をとれば $P_i^m \subset Q_i$ である.

$$P_i^m I_i \subset P_i^m \cap I_i \subset Q_i \cap I_i = 0$$

なので, $P_i^m I_i = 0$ となる最小の $m$ が存在する. $P_i^{m-1} I_i \neq 0$ なのでその 0 でない要素 $x$ を取れば, $\mathrm{ann}_R(x) \supset P_i$ を充たす. 前段の結論と合わせて $\mathrm{ann}_R(x) = P_i$ である.

(2)⇒(1). $x \in R$ を $P = \mathrm{ann}_R(x)$ は素イデアルとなるものとする. イデアル商と交叉の可換性から

$$P = \mathrm{ann}_R(x) = \left(\bigcap Q_i :_R x\right) = \bigcap (Q_i :_R x),$$

補題 5 により, ある $j$ が存在して $P = (Q_j :_R x)$ である. 両辺の根基を取って $P = P_j$ を得る. □

**定理 9 の証明** $R$ のイデアルの集合 $\mathcal{I} = \{\mathrm{ann}_R(x) := (0 :_R x) \mid x \in (R/I) \setminus \{0\}\}$ を考えよう. 定義により

$$\mathrm{ZD}_R(R/I) = \bigcup_{x \in (R/I)\setminus\{0\}} \mathrm{ann}_R(x)$$

である.

$\mathrm{ann}_R(1) = \{0\}$ なので $\mathcal{I} \neq \emptyset$, ネーター性により $\mathcal{I}$ は極大元 $P = \mathrm{ann}_R(x)$ を持つ. $P$ が素イデアルとなることを示そう.

$a, b \in R$ が $ab \in P$, $a \notin P$ を充たすとする. このとき, $abx = 0$ かつ $ax \neq 0$ である. 一般に $\mathrm{ann}_R(ax) \supset \mathrm{ann}_R(x) = P$ であり, $b \in \mathrm{ann}_R(ax)$ である. $P$ の極大性と合わせて $b \in \mathrm{ann}_R(ax) = P$ である. □

零因子に関する補題 9 が得られたところで, イデアルによる局所化を定義しておこう.

**定義 11** ネーター環 $R$ のイデアル $I$ に対し, $R/I$ の非零因子の全体 $S = R \setminus \left(\bigcup_{P \in \mathrm{Ass}_R(R/I)} P\right)$ による局所化を $R$ の $I$ による **局所化** といい, $R_I$ と表す.

### 2.2.2 パラメーター系の構成

では, パラメーター系が如何にして次元, すなわち素イデアルの系列を統制するかを検めたい.

$d$ 次元のネーター局所環 $(R, \mathfrak{m})$ のパラメーター系とは, $d$ 個の要素の列 $x_1, \ldots, x_d$ で $Q = (x_1, \ldots, x_d)$ が $\mathfrak{m}$ 準素イデアルとなるものであった. $\mathfrak{m}$ 準素イデアルは **剰余環のクルル次元が 0 となる**イデアル とも言い換えられるから, 次の条件を充たすように要素の列を取ることでパラメーター系が得られる :

$$\text{条件 : 各 } i = 1, \ldots, d \text{ に対し } \dim R/(x_1, \ldots, x_i) \leq d - i.$$

次第に話は複雑化していくわけだが, まず, 素イデアルの全体が鎖である場合, すなわち次のような素スペクトルを持つネーター環 $R$ が存在したとしよう[*13] :

$$\mathrm{Spec}\ R = \{P_0 \subset P_1 \subset \cdots \subset P_d\}.$$

[*13] とは言うものの, 次元 $d$ が 1 より大きい場合にはこのようなネーター環はない.

このとき, $R$ のクルル次元は $d$ である. 一方, 各 $i$ に対し

$$x_i \in P_i \setminus P_{i-1}$$

を充たすように $x_i$ をとる.

この系列についてパラメーター系の条件を考える. $I_i = (x_1, \ldots, x_i)$ とおく. $x_j \notin P_{j-1}$ から $V(x_j) \subset \{P_0, \ldots, P_{j-1}, P_{j+1}, \ldots, P_d\}$ である. これを合わせると

$$V(I_i) \subset \{P_i, \ldots, P_d\}$$

であり, $\dim R/I_i \leq d - i$ が成り立つ.

これは大変シンプルな系列であったから議論は明快であった[*14]が, 一般の環の場合でも議論はこれと同じである. ただし, その前に脚注で述べたようにこの例がまやかしであることも明かしておこう.

**定理 12 (Krull の標高定理)** ネーター環 $R$ のイデアル $I$ が $n$ 個の要素で生成されるならば, $I$ の極小素因子の高さは $n$ 以下である.

次元定理を認めれば, 標高定理は次のように証明される[*15]. $I$ の極小素因子 $P$ に対し, $IR_P$ は $n$ 個の要素で生成される $PR_P$ 準素イデアルなので

$$\mathrm{ht}\ P = \dim R_P = \delta(R_P) \leq n.$$

さて, 標高定理を単項イデアル $(x_2)$ に適用すれば, その極小素イデアルの高さは 1 以下のはずである. しかしこの例では, $x_2$ は $P_0$ にも $P_1$ にも属さないので, $(x_2)$ を包む素イデアルは高さ 2 以上のものしかないことになる. これは矛盾であり, このような素スペクトルを持つネーター環は存在しない.

しかし, この考察は全く無駄というわけではない. 実際に, $(x_1, \ldots, x_j)$ が $P_{j-1}$ に包まれないように $x_j$ を選ぶことで, パラメーター系は構成される.

**定理 13 (パラメーター系のレシピ)** ネーター局所環 $(R, \mathfrak{m})$ に対して, 次のように帰納的に構成した列 $x_1, \ldots, x_d$ は $R$ のパラメーター系をなす :
(構成法) $x_j$ まで構成されたとし, $I_j = (x_1, \ldots, x_j)$ とする. このとき

$$x_{j+1} \in \mathfrak{m} \setminus \left( \bigcup_{P \in \mathrm{Assh}_R(R/I_j)} P \right)$$

ととる.

**証明** この構成法によって条件が充たされることを示す. $V(I_j)$ の極大鎖で長さが $\dim R/I_j$ に等しいもの

$$P_s \subset P_{s+1} \subset \cdots \subset P_d$$

をとる. このとき $P_s \in \mathrm{Assh}_R(R/I_j)$ ゆえ $x \notin P_s$, したがって $V(I_{j+1})$ に属しうるのは $P_{s+1}$ から $P_d$ のみであるから, $\dim R/I_{j+1} \leq \dim R/I_j - 1$ である. 帰納法の仮定と合わせて, $\dim R/I_j \leq d - j$ を得る.

[*14] むしろ話がうまく行き過ぎて気味が悪い. 実際には存在しないのだから当然である.
[*15] 実際には, Krull は標高定理を先に証明し, 次元定理への足掛かりとしたことで知られている.

以上によって, 素スペクトルの状況を調べる手法がおぼろげながら見えてきただろうか. すると, もう少し素スペクトルの形について知りたくなるのは人情である[*16]. 以下, 次の２つの問題を考える：

**問題 14** ネーター環 $R$ は次の性質を充たすか？
(1) クルル次元は有限か？
(2) $R$ は鎖条件を充たすか, すなわち 2 つの素イデアル $P, Q$ が $P \subset Q$ を充たすとき, $P$ に始まり $Q$ に終わる飽和鎖の長さは一定になるか？

実はいずれも否定的であることが知られている. 残りの節を使って, 反例を紹介したい.

## 2.3 次元が有限でないネーター環の例

素スペクトルの様子を観察することが少しずつ面白くなってきたところで, ネーター環であって次元が有限でない例を構成しておこう. もちろん, 極大イデアルの個数が有限であればそのような例は存在しない[*17]. 素スペクトルの様子の観察法とその操作に慣れてくると, 何をしているか, 何をしたいかがよく解ると思う.

ただし, 素スペクトルの様子だけを観察していてもネーター性を導出することは難しい[*18]. そこで, ネーター性を導出する鍵を与えておく.

**補題 15** 環 $R$ が以下の 2 条件を充たせば ネーター的である. :

1. $R$ の極大イデアル $\mathfrak{m}$ に対し $R_{\mathfrak{m}}$ はネーター的である ;
2. $R$ の非零元 $f$ に対し, $f$ を包む極大イデアルは高々有限個である.

**証明** $R$ のイデアル $I \neq 0$ が有限生成であることを証明する. $I$ を包む素イデアルは $I$ のある非零元を含むので, 条件 *2.* により高々有限個である. それらを $\{\mathfrak{m}_i\}$ とするとき, その各々に対し $IR_{\mathfrak{m}_i}$ を生成する $I$ の部分集合 $G_i$ をとる. 条件 *1.* により $G_i$ は有限集合にとれる. 和集合 $\bigcup G_i$ が生成するイデアル $J$ を考えると, $J \subset I$ かつ総ての極大イデアル $\mathfrak{m}$ に対して $JR_{\mathfrak{m}} = IR_{\mathfrak{m}}$ が成り立つ. *Local Property* により $I = J$, 特に $I$ は有限生成である.□

**例 16** 体 $K$ 上の可算個の独立変数 $X_s$, $s \in \mathbb{N}$, をとる. 自然数の単調増大列 $\{m_t\}_{t\in\mathbb{N}}$ を, 階差数列 $\{m_{t+1} - m_t\}$ が単調増大列となるようにとる. ここで, 多項式環 $R = K[X_s \mid s \in \mathbb{N}]$ の素イデアル $P_t := (X_u \mid m_t \leq u < m_{t+1})$ を考え, どの $P_t$ にも属さない要素の全体がなす積閉集合を $S$ とおくと, 局所化 $R_S$ はネーター環で $\dim R_S = \infty$ である.

**証明** $P_t$ に属する変数列が生成する素イデアルの列を考えれば, $P_tR_S$ の高さの上限は無限大である. ネーター性は補題 *15* による. $R_S$ の極大イデアルは $P_tR_S$ で尽くされることに注意しよう. 各要素 $f$ に現れる変数は高々有限個なので, $f$ を包む極大イデアルは

[*16] ちょっと強引な展開である. 筆者にも自覚はあるが, 時間がないのでご勘弁願いたい.
[*17] ネーター環 $R$ の素イデアル $P$ に対し, $R_P$ の次元は $P$ の生成系の濃度以下であるから有限である. $R$ の次元は極大イデアルによる局所化の上限であるから, 極大イデアルが有限個ならば次元は有限である.
[*18] 素スペクトルが 1 点集合となる非ネーター環はいくらでもある. 例えば, 無限変数の多項式環 $S := K[x_n \mid n \in \mathbb{N}]$ の極大イデアル $\mathfrak{m} = (x_n \mid n \in \mathbb{N})$ の冪による剰余環 $S/\mathfrak{m}^s$ は総てそのような例を与える.

高々有限個である．また極大イデアルによる局所化は全てネーター的であるから，$R_S$ はネーター的である．□

この環の素スペクトルは，多項式環の素スペクトルの直和集合に等しい．気分的には，どんどん長くなる鎖をかき集めている[*19]ようなものである．これに倣えば「クルル次元が有限でない」環を見つけることは難しくはない．実際，環の直積の素スペクトルは各々の素スペクトルの直和集合であり，同じ素スペクトルを得られる．しかしながら，ネーター性を保ちつつ素スペクトルが直和集合となるような工夫が必要なのである．

## 2.4　鎖状でないネーター環の例

ネーター環の部分環でネーター的でない例を作る方法として，次のアイデアがある：

**例 17** $\mathbb{Q}$ を有理数体，$\mathbb{R}$ を実数体とする．$t$ を $\mathbb{R}$ 上の変数とし，

$$A := \mathbb{Q} + t \cdot \mathbb{R}[t] = \{a_0 + a_1 t \cdots + a_s t^s \mid a_0 \in \mathbb{Q}, a_i \in \mathbb{R} \ \ (\forall i > 0)\}$$

とおくと，$A$ はネーター的ではない．

この例では，基礎体 $\mathbb{Q}$ と，極端に大きい極大イデアル $t\mathbb{R}[t]$ との和を取ることでネーター性を崩した．このように，**奇怪なイデアルと基礎体との和を取って奇妙な環を作る** 方法はなかなか有効である．今回の例もこのように構成されるが，鎖状でないネーター環を作ろうというのだから少し工夫が必要ではある．

**補題 18** $R$ をネーター半局所環，$\{\mathfrak{m}_i\}_{1\le i\le s}$ をその極大イデアルの全体とする．$R$ はネーター局所環 $(K, \mathfrak{m})$ を支配し[*20]，かつ $R/\mathfrak{m}_i$ は $K/\mathfrak{m}$ の有限次代数拡大とする．$\mathfrak{a} = \bigcap \mathfrak{m}_i$ とし，$S = K + \mathfrak{a}$ とおく．このとき $S$ はネーター局所環で，$R$ は $S$ 加群として有限生成である．

**証明** $R$ は $K$ を支配するので，各 $\mathfrak{m}_i$ は $\mathfrak{m}$ を包む，ゆえに $\mathfrak{a}$ は $\mathfrak{m}$ を包む．ここから $x \in \mathfrak{a}$ に対し $1 + x$ は可逆[*21]であり，$S$ は $\mathfrak{a}$ を唯一の極大イデアルとする局所環である．

$R/\mathfrak{a}$ は $K/\mathfrak{m}$ ベクトル空間として有限生成である[*22]．$e_1, \ldots, e_t \in R$ を，その $R/\mathfrak{a}$ への像が $K/\mathfrak{m}$ ベクトル空間としての基底を与えるようにとる．

---

[*19] 実際にはもう少し複雑であるが．

[*20] 環 $R$ が環 $S$ を支配するとは，(1) $S \subset R$ かつ両者は単位元を共有する，(2) $J \subset S$ がイデアルならば $JR \neq R$，(3) 各 $\mathfrak{m} \in \mathrm{Max}\, R$ に対し $\mathfrak{m} \cap S$ は $S$ の極大イデアルである，の 3 条件を充たすことをいう．

[*21] $\mathfrak{a} = \mathrm{rad}\ (R)$ なので $1 + x$ は $R$ 内で可逆である．逆元 $y$ に対し $1 - y$ を考えると

$$(1 + x) \times (1 - y) = (1 + x) \times 1 - (1 + x) \times y = (1 + x) - 1 = x \in \mathfrak{a},$$

$1 + x$ は可逆ゆえ $1 - y \in \mathfrak{a}$，ゆえに $y \in 1 + \mathfrak{a} \subset S$.

[*22] イデアルの系列

$$\mathfrak{a} = \mathfrak{a}_s \subset \mathfrak{a}_{s-1} \subset \cdots \subset \mathfrak{a}_1 \subset R, \quad \mathfrak{a}_i = \bigcap_{j=1}^{i} \mathfrak{m}_j$$

の各因子 $\mathfrak{a}_{i-1}/\mathfrak{a}_i$ は $R/\mathfrak{m}_i$ 加群として有限生成である．$R/\mathfrak{m}_i$ は $K/\mathfrak{m}$ ベクトル空間として有限次元なので，上の系列は各因子が有限次元 $K/\mathfrak{m}$ ベクトル空間となる部分加群の系列であり，$R/\mathfrak{a}$ も有限次元 $K/\mathfrak{m}$ ベクトル空間となる．

$R/\mathfrak{a}$ は $K/\mathfrak{a}$ 上整[*23]なので, 各 $x \in R$ に対し, $c_1, \cdots, c_n \in K$ で

$$x^n + c_1 x^{n-1} + \cdots + c_n \in \mathfrak{a}$$

を充たすものが存在する. このとき, $x$ は $S$ 上整である. したがって, $S[e_1, \ldots, e_t, x]$ は $S$ 加群として有限生成である. $R = \mathfrak{a} + \sum S \cdot e_i$ から

$$S[e_1, \ldots, e_t, x] = \mathfrak{a} S[e_1, \ldots, e_t, x] + \sum S \cdot e_i$$

を得る. 中山の補題により, $S[e_1, \ldots, e_t, x] = \sum S \cdot e_i$ である. $x$ の任意性から $R = \sum S \cdot e_i$. *Eakin*-永田の定理[*24]により, $R$ のネーター性から $S$ のネーター性を得る.□

今回の例がネーター的となることを保証する補題を紹介する.

**補題 19** 局所環 $(R, \mathfrak{m})$ が条件 (1) $\bigcap \mathfrak{m}^n = 0$, 及び (2) $\mathfrak{m}$ は単項イデアルである, を充たすならば ネーター的である.

**証明** $\mathfrak{m}$ の生成元を $p$ とする. $R$ のイデアル $I \neq R$, 0 に対し, $I \subset \mathfrak{m}^s$, $I \not\subset \mathfrak{m}^{s+1}$ を充たす自然数 $s$ が存在する. このとき $(I :_R p^s) \not\subset pR$, ゆえに $(I :_R p^s) = R$. ここから $I = p^s R$ を得る. □

体 $K$ 上の 1 変数形式冪級数環 $K[[X]]$ を考え, その $s$ 個の要素 $z_1, \ldots, z_s$ で, 分数体 $K(X)$ 上代数的に独立なものをとる. これらを

$$z_i = a_{i,0} + a_{i,1} X + a_{i,2} X^2 + \cdots, \quad a_{i,j} \in K$$

と表し,

$$z_{i,j} = \frac{z_i - \sum_{k<j} a_{i,k} X^k}{X^{j-1}} = a_{i,j} X + a_{i,j+1} X^2 + a_{i,j+2} X^3 + \cdots$$

とおく. このとき, $z_{i,j} = X a_{i,j} + X z_{i,j+1}$ が成り立つ.

さらに変数列 $Y_1, \ldots, Y_m$ により

$$T_0 = K[X, z_{i,j} \mid 1 \leq i \leq s,\ j \geq 1]$$
$$T_m = T_0[Y_1, \ldots, Y_m]$$

とおく. このとき, 先の関係式 $z_{i,j} = X a_{i,j} + X z_{i,j+1}$ から $X \cdot K[[X]] \cap T_0 = X T_0$ が成り立つ. したがって, この極大イデアル $XT_0$ による局所化 $U = (T_0)_{XT_0}$ は $XU$ を極大イデアルとする局所環で, $K[[X]]$ に支配される. $K[[X]]$ においては $\bigcap_n (Xk[[X]])^n = 0$ なので $\bigcap_n (XT_0)^n = 0$, したがって補題 19 により $U$ はネーター的である.

$T_m$ においては, $\mathfrak{M} = XT_m + (Y_1, \ldots, Y_m)$ が極大イデアルで, $V_m := (T_m)_{\mathfrak{M}}$ はネーター環 $U[Y_1, \ldots, Y_m]$ の局所化となる. 特に $V_m$ は $m+1$ 次元の正則局所環である.

また, $T_0[X^{-1}] = K[X^{\pm 1}, z_1, \ldots, z_s]$ ゆえこれはネーター的である. このとき, $T_m$ において $\mathfrak{N} = (X - 1, z_1, \ldots, z_s, Y_1, \ldots, Y_m)$ なる極大イデアルを考えると, $W_m = (T_m)_{\mathfrak{N}}$ は次元 $s + m + 1$ の正則局所環である.

---

[*23] 環拡大 $A \subset B$ が与えられたとき, $b \in B$ が $A$ 上 **整** とは, $b$ を根に持つ $A$ 係数のモニック多項式が存在することをいう. $B$ が $A$ 加群として有限生成ならば $B$ の任意の要素は $A$ 上整である.

[*24] 環拡大 $A \subset B$ が与えられ, $B$ が $A$ 加群として有限生成かつネーター的ならば $A$ もネーター的である.

$\mathfrak{M}$ と $\mathfrak{N}$ の違いは, $z_t$ が $X$ では割り切れるが $X-1$ では割り切れない点にある. これによって, 整域内で高さの異なる極大イデアルを作ることに成功している. あとは, これをうまく貼り合わせる.

$R_m = (T_m)_{\mathfrak{M}\cap\mathfrak{N}}$ とおくと, $R_m$ は $\mathfrak{M}R_m$ 及び $\mathfrak{N}R_m$ をただ 2 つの極大イデアルとする半局所環である. $(R_m)_{\mathfrak{M}R_m} = (T_m)_{\mathfrak{M}}$, $(R_m)_{\mathfrak{N}R_m} = (T_m)_{\mathfrak{N}}$ はともにネーター的で, 補題 15 により $R_m$ はネーター的である.

ここで $\mathfrak{a}_m = \mathfrak{M}R_m \cap \mathfrak{N}R_m$ とおき, $S_m = K + \mathfrak{a}_m$ とおく. 補題 18 により, $S_m$ はネーター局所環で, $R_m$ は $S_m$ 加群として有限生成である. $\mathfrak{a}_m \subset S_m$ ゆえ $Q(R_m) = Q(S_m)$ である. $V_m$, $W_m$ はともに正規ゆえ $R_m$ も正規であり, $R_m$ は $S_m$ の整閉包である.

$R_m$ には極大イデアルが 2 個しか存在しないので, 極大鎖のある要素が一方の極大イデアルに包まれなければ, その鎖の最大元は自然に他方に決まる. そしてその高さは既に解っているので, 極大鎖の長さが計算できるという仕組みになっている.

そして, $R_m$ の 2 個の極大イデアルを強引に 1 つにすぼめてしまったのが $S_m$ というわけだ. $S_m$ の素スペクトルの様子は局所的には $R_m$ の素スペクトルの様子と似ている[*25] ので, その極大鎖の長さは $R_m$ 内の鎖と比較して求めることができる. このとき, $S_m$ の素イデアルの鎖が $R_m$ の 2 個の極大イデアルのいずれを最大元とする鎖に由来するかによって, 極大鎖の長さが変化するという仕掛けである. すなわち, 次が成り立つ.

**補題 20** (1) $S_0$ においては鎖条件は成り立つが, $S_0$ の整閉包 $R_0$ は高さ 1 の極大イデアル $\mathfrak{m}R_0$ と高さ $s+1$ の極大イデアル $\mathfrak{n}R_0$ を持つ.
(2) $m \geq 1$ のとき, $S_m$ は鎖条件を充たさない.

**証明** (1) $S_0$ の整閉包 $R_0$ の極大イデアルは $\mathfrak{M}R_0$, $\mathfrak{N}R_0$ の 2 個であり, $\mathfrak{M}R_0$ の高さは 1 である. $\mathrm{Spec}\, R_0 \setminus \{\mathfrak{M}R_0\}$ は $\mathrm{Spec}\, W_0$ と包含関係を保って一対一に対応する. ゆえに $\mathfrak{N}R_0$ の高さは $s+1$ である.
(2) $Q = XR_m \cap S_m$ とおくと, $Q = XR_m \cap \mathfrak{N}R_m$ ゆえ $QW_m = \mathfrak{N}W_m$. ここから, $Q$ 上に乗っている $W_m$ の素イデアルは $\mathfrak{n}R_m$ に包まれないので, $\mathrm{ht}\, Q = \mathrm{ht}\, XR_m = 1$. $Q$ を含む素イデアルの極大鎖

$$0 \subset Q \subset Q_2 \subset \cdots \subset Q_v$$

をとり, 上昇定理を用いて $R_m$ の素イデアルの鎖

$$0 \subset Q_1' \subset Q_2' \subset \cdots \subset Q_v'$$

をとれば $Q_1' \not\subset \mathfrak{N}R_m$ ゆえ $Q_v' = \mathfrak{M}R_m$, したがって $v = m+1$. ここから $S_m$ における極大鎖で長さが $m+1$ のものが存在する. 一方 $\mathrm{ht}\, \mathfrak{a}_m = s+m+1$ ゆえ $S_m$ は鎖状でない. □

**定理 21** 体 $K$ 上可算変数の多項式環は鎖状でない.

**証明** 上記の鎖状でないネーター環の構成を精査すると, 体 $K$ 上可算個の変数で生成される代数である. その素スペクトルは 体 $K$ 上の可算変数の多項式環の素スペクトルの部分集合と見做せるので, 前者が鎖状でない以上, 後者も鎖状ではありえない. □

[*25] ある環とその整閉包の間には素スペクトルの様子が「似ている」ことを保証する一連の定理群 (全射性定理, 比較不能定理, 上昇定理, 下降定理など) がある.

この注意は Hutchins [2] に述べられていたものであるが, この本を読んで筆者が一番驚いた主張でもある. まだまだ身近なところにも知らないことがいくらでもあるのだと当たり前のことを思い知らされ, また数学が好きになったものである.

# 参考文献

[1] M. Atiyah and I. G. MacDonald, “Introduction to Commutative Algebra”, Addison-Wesley, 1969.

[2] H. Hutchins, “Examples of Commutative Rings”, Polygonal Publishing House, 1984.

[3] 松村英之, “復刊 可換環論”, 共立出版, 2000.

[4] 永田雅宜, “可換環論”, 紀伊國屋数学叢書, 1974.

# 第 3 章

# 束論日記

木村 允哉

## 3.1 はじめに

前から Peter T. Johnstone って人の『Stone spaces』[1] って本が読みたかったんですよ。ただ何となく手が付かないまま時間が過ぎて、これまた何となくガロア理論の本 [2] 読んでたら Stone Duality の話が出てきた。特に急ぎで読まなきゃいけない訳でもなし、ちょっと寄り道するか。半分読むくらい集中力が続けばいいな、くらいに構えてやり始めました。

ところがどっこい、最初の Preliminaries で真面目に束やるかーという段で疑問が出てくる出てくる。いつの間にか Stone Space をやろうとしていたことを忘れて束のことばっか考えるようになってた。

この日記はその時の疑問と考察をできるだけそのまま書いたものです。だから束論を触ったことのある人は知ってる話が多いだろうし、この日記の時点では肝心要の Stone space は出てきません。悪しからず。

束に詳しい人は「そんなとこ気になる？」とか「あーそれ私も気になった、懐かし」とか、

詳しくない人は「そう考えるんだ」とか「こうは考えないのか？」とか、

そんな感じで読んでください。

## 3.2 冪等モノイドに定まる順序、半束の定義

まず、最初に半束の定義から入りました。有限部分集合に対して上界 (または下界) が定まる、という順序集合的な定義に対して代数的な定義もあるというのはどの教科書でも触れられていること (らしい) ですが、きちんとした主張として見たのは初めてだった気がします。

**Theorem 3.2.1** *[1, P.2]* $A$ は演算 $(\vee) : A \times A \longrightarrow A$ を持つ可換モノイド *(*単位元付き可換半群*)* で単位元は $0 \in A$ とする。任意の $a \in A$ に対して冪等律

$$a \vee a = a$$

が成立するとき、$A$ に次を満たす半順序構造が一意的に定まる。

1. 任意の $a, b \in A$ に対して $a \vee b = \sup\{a, b\}$ となる。
2. $0 = \min A$ となる。

この性質を使って半束を定義します。

**Definition 3.2.2** 上の定理で可換冪等モノイドに一意的に定まる半順序構造を半束、特に上半束や結び半束 *(join semi-lattice)* などと呼ぶ。演算 $(\vee)$ を結び *(join)* と呼ぶ。これの双対として、可換冪等モノイド $(A, \wedge, 1)$ に対して $a \leq b$ を $a \wedge b = a$ で定めた順序については、演算 $(\wedge)$ を交わり *(meet)*、半順序構造を下半束または交わり半束 *(meet semi-lattice)* と呼ぶ。

ここでいう上半束は下に有界、つまり最小元があるものですが、単位元なしの冪等可換半群に対して同様の定理が成り立つであろうことは想像がつきます。この場合、空でない有限部分集合に対して上界が定まる順序集合ということになりますね。

半束ではなく束の代数的な特徴づけといったら結びと交わりの間の性質、吸収律があります。

思えば以前少しだけ束に触れたのは某可換環入門書で *Stone* 対応 *(*ブール環とブール束の対応、または同値性*)* についての演習問題を解こうとしたときでした。あの時はブール束の定義がとても複雑に見えたものです。

まず束って何よ、から始めた僕は当時参考にした教科書 *(*タイトルは忘れました*)* で束の代数的特徴づけについて触れられた文章を見て疑問符を頭上に浮かべていたような気がします。さらっと「吸収律」って書いてあるけどこいつは何者だ？

その教科書では「こういう定義の仕方もあるよ」という程度の書き方で、吸収律がどんな風に生きてくるのかも特に書いていなかったのであの時の僕は潔く投げ出しました。ようするにじゅんじょしゅうごうだろう？

今、思い返してみるとブール束とブール環の対応こそ、代数的な束の定義が生きてくるところですね。人生ってうまくいかないものです。

丁寧にステートメントにすると次のように表せるようです。

**Theorem 3.2.3** *[1]* 集合 $A$ が二つの半束構造 $(A, \wedge, 1), (A, \vee, 0)$ を持ち、それぞれについて定まる半順序を $\leq_{MSL}$、$\leq_{JSL}$ とする[*1]。ただし $(\wedge, \leq_{MSL})$ が下半束、$(\vee, \leq_{JSL})$ が上半束を定めるとする。

この二つの順序が一致することは、任意の $a, b \in A$ に対して吸収律と呼ばれる次の性質が成立することに同値である。

$$a \wedge (a \vee b) = a$$
$$a \vee (a \wedge b) = a$$

[*1] MSL は Meet Semi Lattice、JSL は Join Semi Lattice の略。

後で使うので最後に分配束について定義しておきます。

**Definition 3.2.4** 分配律と呼ばれる次の関係式を満たす束を分配束と呼ぶ*:*

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

この関係式とその双対、どちらか一方について満たしていればもう一方の関係式も満たされることが知られています。

**Proposition 3.2.5** *[1, P.3]*
$A$ は束とする。次の二つは同値である。

1. 全ての $a, b, c \in A$ に対して $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ が成立する。
2. 全ての $a, b, c \in A$ に対して $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ が成立する。

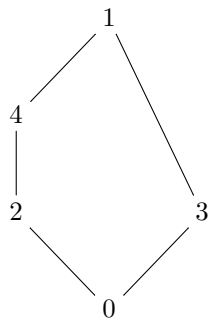
## 3.3 半束の準同型とイデアル、フィルタ

代数的な構造があれば準同型を考えてあげるのが世の情けです。

**Lemma 3.3.1** 上半束の間のモノイド準同型[*2] $f : A \longrightarrow B$ は順序を保存する。

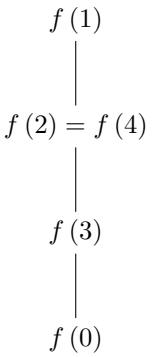
ここでは上半束準同型をモノイド準同型で定義します。でもこれって単なる順序保存写像よりも強い条件でしょうか？

**Example 3.3.2** 結論からいうと半束間の順序保存写像であってもモノイド構造を保つとは限りません。次の上半束を考えます。




[*2] ここでいうモノイド準同型は半群構造と単位元を保つ写像のことです。

これを順序保存写像 $f$ で次のような半束に写すとしましょう。




このとき、$f(3) \vee f(4) = f(4) \neq 1$ ですが $f(3 \vee 4) = f(1) = 1$ です。従って半群構造が保たれていません。

準同型が出てきたら準同型定理を考えてあげるのが世の理です。教科書ではまずイデアルとフィルタについて定義され、それによる準同型定理が紹介されていました。

**Definition 3.3.3** 上半束 $A$ の空でない部分集合 $I \subset A$ は

1. 下方に閉じている *($a \leq b$ かつ $b \in I$ ならば $a \in I$)*
2. 有限個の元の結びで閉じている *(*有限集合 $F \subset I$ に対して $\sup F \in I$ である*)*

を満たすとき、$A$ のイデアルと呼ぶ。特に $\sup \emptyset = 0 \in I$ であるからイデアルは常に最小元を含むことに注意。

また上半束に対するイデアルの双対として、下半束に対するフィルタを定義する。つまり上方に閉じている部分モノイドである。

**Lemma 3.3.4** *[1, P.11]* $A, B$ は上半束、$f : A \longrightarrow B$ は上半束準同型とする。

1. $f$ の核と呼ばれる集合 $\ker f := \{a \in A \,|\, f(a) = 0\}$ は $A$ のイデアルをなす。
2. $A$ のイデアル $I$ について、ある上半束 $C$ と準同型 $g : A \longrightarrow C$ が存在して $\ker g = I$ となる。
3. さらに $A$ が分配束ならば、$C$ として分配束、$g$ として束の準同型[*3]を取れる。

まあ正確に言うと準同型定理のステートメントではありませんが似たようなものですよね。$C$ は普通 $A/I$ などと書くと思われます。

こういう場合、イデアル $I$ に因んだ何らかの同値関係であって演算と両立するものを考えるとその剰余がうまいこと半束になるよ、というストーリーがテンプレです。

演算と両立する同値関係を合同関係とよく呼びますが、今回使う合同関係は次のようなものです。

$$a \equiv_I b \underset{dfn}{\Leftrightarrow} \exists i, j \in I, a \vee i = b \vee j$$

僕にはこれが唐突に見えました。こいつはどこからやってきた？その答えを探す為、我々は密林へと飛びました。

[*3] 上半束準同型かつ下半束準同型であるもののこと。

*[3]* には、圏 $\mathcal{C}$ において、射の関係のクラス $R = \left\{R_{A,B} \subset \mathcal{C}(A,B)^2\right\}_{A,B\in\mathcal{C}}$ があるとき、各 $R_{A,B}$ が潰れてくれるような合同関係 *(*射の同値関係であって合成と両立する*)* を生成できるよ、という話が載っています[*4]。生成というだけあってこの合同関係のクラスは $R$ を含むものとして最小であり、圏と函手について準同型定理ができるようなものです。

今、半束がモノイドであるのですからこの話が使えます。つまり、イデアル $I$ がうまく潰れてくれるような圏としての合同関係 $(\sim_I)$ が構成できます。

結論だけ書くと次が成り立ちます。

**Theorem 3.3.5** $(\equiv_I) = (\sim_I)$

この定理の意味するところは簡潔で、$(\equiv_I)$ は $I$ を潰す合同関係の中で最小ということです。この合同関係のお気持ちは、$I$ の元を同一視する上での最低限の条件ってなんだろう、ということですね。.....

.....でもこれって要するに準同型定理が成り立つということですね[*5]。$(\equiv_I)$ の定義がどこから現れたのか、の答えとしては的を外している気がします。

よくわからんですが今は先に進みましょう。

実際に上記の定理等を示そうとするとわかりますが、実は上半束の合同関係に関してはイデアルの下方閉性は使用しません。

**Lemma 3.3.6** $A$ は上半束、$B$ はその部分上半束 *(*部分モノイド*)* とする。このとき $B$ を潰す最小の合同関係 $(\sim_B) \subset A \times A$ について次が成立する。

$$a \sim_B b \Leftrightarrow \exists \alpha, \beta \in B, a \vee \alpha = b \vee \beta$$

準同型の核がイデアルとなるのは確かです。でもそれって重要なことでしょうか？部分半束で成り立つのだから部分半束でいいんじゃない？わざわざ下方閉性を採用する理由は？君を雇用する上での弊社のメリットは？

いえ、なんでもないです。学生の方は就活頑張ってくださいね。

話を続けましょう。加群の準同型の核などは 0 写像とのイコライザという言い回しがあります。ちょっとその視点で考えて見ましょう。

**Lemma 3.3.7** 平行する上半束準同型の *(*集合の写像としての*)* イコライザは始域の部分束となる。

イデアルとなることを期待してこの視点にしたんですが、部分束でいいじゃんという説を後押しする形になってしまいました。もしかしてイデアルの役割を全部部分束で置き換えられるのでは？

[*4] path category と conditional graph のお話です。

[*5] この日記を書いていて気づきました。

ところがそうはいきませんでした。分配束での合同関係生成を、これまでと同様に

$$\exists \alpha, \beta \in I, a \vee \alpha = b \vee \beta$$

の形にしたいならば $I$ には下方閉性が求められます。疑ってごめんよ。

## 3.4 最後に

中途半端なところで終わりますがまだまだ疑問がたくさんあります。

1. 完備束の代数的な特徴づけってないだろうか *(なんか前に見かけた気がする)*
2. 完備束 $L$ に対して $L \simeq I(A)$ となる半束 $A$ は常に存在するか？ *(ただし $I(A)$ は $A$ のイデアル全体が成す完備束)*
3. 結局 $(\equiv_I)$ はどこから来たの？
4. 反例作りがぱっとできない。
5. 束についてはイデアルによる合同関係とフィルタによる合同関係が作られるけれど、これらについて素イデアルなどの場合はその補集合の成すフィルタとの間に何か関係がありそうじゃない？
6. *etc., etc. ...*

*[1]* では多分この辺の細かい、基本的な疑問にスラスラ答えられることが求められている気がするので、素直に力不足を認めて束の入門書読もう、というのが最近の感想です。がんばろ。

おわり

# 参考文献

*[1] Peter T. Johnstone. Stone Spaces. Cambridge University Press. 1982.*

*[2] Francis Borceux, George Janelidze. Galois Theories. Cambridge University Press. 2001.*

*[3] Francis Borceux. Handbook of Categorical Algebra, volume.1. Cambridge University Press. 1994.*

# 第 4 章

# カルマ・モナド

淡中 圏

仏陀は深い瞑想の中に潜った。輪廻の連鎖を断ち切らなければいけない。そこで仏陀は輪廻の連鎖をつぶさに眺めた。仏陀が見たのは、循環しながら、一巡りの最後にカルマを自らに引数に受け渡して末尾再帰する存在だった。そうしながら自らも少しずつ変化させ、時には大きくホットスワッピングする。それこそ輪廻だ。

仏陀は出家する前に、計算数学について多少学んでいたので、それが何かわかった。$A$ から $B$ への単なる関数 $A \to B$ ではなく、様々な余分のものを帯びながら、それでも $A \to B$ と $B \to C$ の合成ができる概念。それはすなわち**モナド**だ。

仏陀は、一切生者が**カルマ・モナド**であることを悟ったのだ。

仏陀は続いてカルマの中に潜り込んだ。カルマは**継続モナド**の一種のように見えた。すなわちカルマの中には、これから起こるであろうことの情報が陽に入っていて、通常はそれが起こるのだが、輪廻の進展の仕方によってそれが変化するのだ。

仏陀はそこに活路を見つけた。ただの関数ならば $A \to B$ という型を持っていれば、$A$ の次は $B$ にならなければいけない。しかし継続モナドならば、未来に影響を与えることができるのだ。

仏陀はカルマの中で継続をハックし、輪廻の過程を中断させることに成功した。$A \to B$ の流れの中で、$B$ にはいかず、全ての関数呼び出しの大元の元へと帰ってしまうのだ。これすなわち**例外**である。これこそ衆生の救いだ。

救われたい衆生は皆、例外を投げて、トップレベルまで上がってスタックトレースを吐き出すのだ。

# 参考文献

*[1]* ライプニッツ, ゴッドフリート. 清水富雄/竹田篤司/飯塚勝久 訳. モナドロジー　形而上学叙説. 中央公論社〈中公クラシックス〉. *2005.*

*[2] Dee, John. The Hieroglyphic Monad. Red Wheel/Weiser; New edition editon.*

*2001.*

*[3] Bruno, Giordano. De monade, numero, et figura. lacobum Fischerum. 1614.*

**淡中 圏**　本名：田中健策　うえーい、仕事が忙しい時にする数学の同人誌原稿はサイコーだわーい。来年もわけのわからないことをたくさんやって、世人を困惑させたい。計画は色々あるが、なんせ時間と自分の体が足らなすぎる。
よく分からない自作サイト　*https://tannakaken.xyz/*

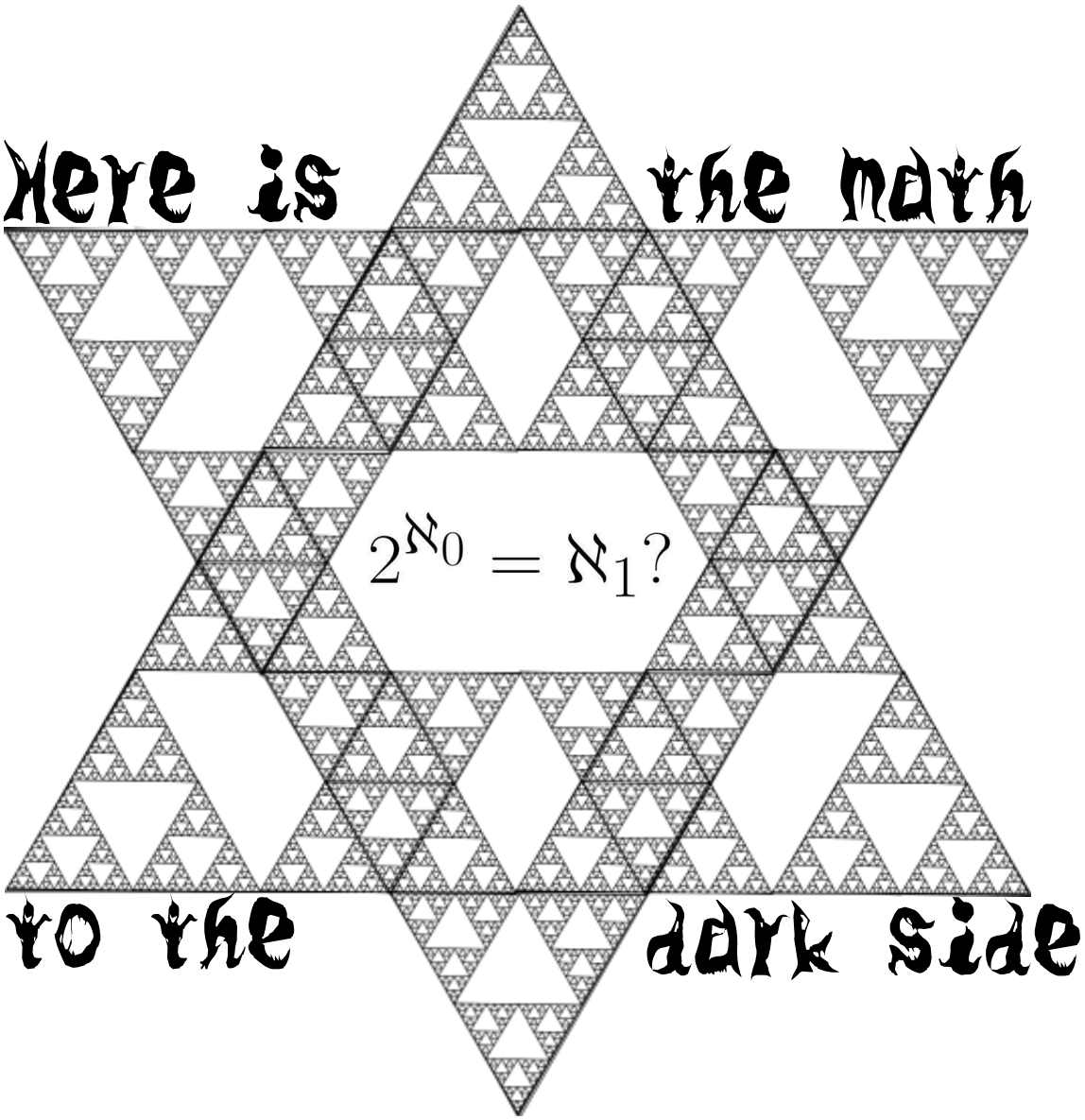
**木村 允哉**　*Gratzer* という人の本が良さそう。読も。指摘があれば *Twitter(mskm9926)* へお願いします。

**サショー☆シロカミ**　表紙をせっせと書くだけになってしまいました。オオカミの頭に乗っかっているのは巨大ナメクジです。オオカミはベール被って思わせぶりな顔してますけど、きっと心の中ではひいこら数学を考えてます。ナメクジ君はそんなオオカミ君の必死な脳みその権化……なのかもしれない。

**龍孫江**　職業柄盆暮れは忙しいことが多いのですが, お盆は締め切り後のお盆当日が忙しいのに対して, 暮れは締め切り直前が忙しいことに気づきました. ちょっとクラクラしながら好きなことを書きました. いつもながら, ありがとうございます.

編集後記*:* とりあえず、前回は私一人しか書かなかったので、今回は私一人じゃなくて良かった、ってことをまず言いたいです。でも最近、ループ量子重力理論の人と微分ガロア・差分ガロアの人に声をかけて（誰のことなのか特定できる人が多そうだな）、特にループ量子重力理論の人からはいい返事がもらえているので、来年の夏か冬あたり、またマニアックな誌面を作ることができるかもしれません。乞うご期待！　と共に、俺ならもっとマニアックにできる！　という人も募集中でっす！ *(*追伸：今回の執筆陣には入ってませんが、まともな *Makefile* を書いてくれた才川隆文さんに感謝。やはり良い *Makefile* がないと雑誌は作れませんね*)* 【淡中　圏】

発行者 ： The dark side of Forcing
連絡先 ： http://forcing.nagoya
発行日 ： 2018 年 12 月 31 日


Here is the math
$2^{\aleph_0} = \aleph_1?$
to the dark side