0.3 濃度の話

それでは、もう少し濃度について語ることにしよう。

偶数や奇数は自然数と一対一の対応がつく、という理由から濃度が等しいことを説明した。今度は分数、すなわち有理数の濃度についてだが、結論を先に言えば、有理数の濃度は自然数の濃度に等しい。つまり有理数は自然数と一対一の対応がつくのだ。表を見てもらいたい。

$b \backslash a$	1	2	3	4	5	…
1	$\frac{1}{1}$ →	$\frac{2}{1}$	$\frac{3}{1}$ →	$\frac{4}{1}$	$\frac{5}{1}$ →	…
	↙	↗	↙	↗		
2	$\frac{1}{2}$	$(\frac{2}{2})$	$\frac{3}{2}$	$(\frac{4}{2})$	$\frac{5}{2}$	…
	↓ ↗	↙	↗			
3	$\frac{1}{3}$	$\frac{2}{3}$	$(\frac{3}{3})$	$\frac{4}{3}$	$\frac{5}{3}$	…
	↙	↗				
4	$\frac{1}{4}$	$(\frac{2}{4})$	$\frac{3}{4}$	$(\frac{4}{4})$	$\frac{5}{4}$	…
	↓ ↗					
5	$\frac{1}{5}$	$\frac{2}{5}$	$\frac{3}{5}$	$\frac{4}{5}$	$(\frac{5}{5})$	…
⋮	⋮	⋮	⋮	⋮	⋮	⋱

( ) と矢印を無視しておけば、ここにはすべての有理数があることが分かるだろう。もっとも 0 と負の有理数を省いているが、この先の話の本質を変えることはないし、簡単のために正の有理数だけで説明を続けるとしよう。

有理数の濃度が自然数の濃度に等しいことを示すには、表にあるすべての有理数に 1, 2, 3, … と番号が振れることを示せばよいわけである。それには、表の有理数を矢印の順に沿って番号を付けるだけで済む。( ) で囲った数は、すでに番号が振られている有理数と同じ値なので、飛ばしておく。すると有理数が自然数に一対一に対応していることが分かるはずだ。えー？　そうかな？　このような番号の振り方では、対角線の左上に番号が付いても、それと同じ量の番号の付いてない数が右下に残るんじゃないの？　そうだね。たしかにそんな気がするかもしれない。でも、それはわれわれが無限をよく理解できないからである。この番号の付け方で、もれなく有理数には番号が付くのだ。いくら右下に無尽蔵の有理数があっても、自然数だって無尽蔵にあるんだから。

すると今度は小数の番である。小数は有理数と違い、自然数によって番号を振ることができない。カントールは次のように説明している。

いま、すべての小数にもれなく番号を振ることができたと仮定しよう。そしてすべての小数を番号順に

$$\begin{aligned} d_1 &= 0.\alpha_1\alpha_2\alpha_3\alpha_4\dots \\ d_2 &= 0.\beta_1\beta_2\beta_3\beta_4\dots \\ d_3 &= 0.\gamma_1\gamma_2\gamma_3\gamma_4\dots \\ d_4 &= 0.\delta_1\delta_2\delta_3\delta_4\dots \\ &\vdots \end{aligned}$$

と、並べたとする。そして、これらの数をもとに、次の操作で新たな小数 $\dot{x}$ を作る。$\dot{x}$ の小数第 1 位の数字は、d_1 の小数第 1 位の数字と違う数字 $\dot{\alpha}_1$ を選ぶ。また、小数第 2 位の数字は、d_2 の小数第 2 位の数字と違う数字 $\dot{\beta}_2$ を選ぶ。このように、$\dot{x}$ の小数第 n 位の数字には、n 番の番号が付いた数の小数第 n 位とは違う数字を選ぶのである。すると新しい数

$$\dot{x} = 0.\dot{\alpha}_1\dot{\beta}_2\dot{\gamma}_3\dot{\delta}_4\dots$$

が出来上がるが、$\dot{x}$ はこれまでに番号を振られた数の中にない数である。

ちょっと変だぞ。はじめの仮定では、すべての小数にもれなく番号が振られたはずなのに、いま出来上がった $\dot{x}$ は番号を振られた数とは明らかに違っている。この矛盾が生じた理由は、最初にすべての小数に番号を付けることができると仮定したからだ。仮定は間違っている。すなわち、小数には自然数と同じ番号を振ることができない。つまり、小数の集合と自然数の集合では濃度は違うのだ。それも小数のほうが濃い濃度を持っていることになる。

集合の濃度といっても無数に**要素**を含んでいるので、その量を数値で表すことなどできない。そこでわれわれは、自然数の集合の濃度を $\aleph_0$[1)] で表すことにしよう。すると、これまでに分かったことから、偶数の集合と奇数の集合の濃度も、さらには有理数の集合の濃度も $\aleph_0$ である。しかし小数の集合の濃度は $\aleph_0$ ではない。

1) $\aleph$ はヘブライ文字で「アレフ」と読む。

おおまかに言って、小数で表すことができる数は**実数**と呼ばれている。もちろん有理数は実数である。なぜなら $\frac{1}{3} = 0.333\cdots$ や $5 = 5.0$ のように、あらゆる有理数は小数表記にできるからだ。さて、実数の集合の濃度は $\aleph_0$ ではないので、今度はそれを $\aleph$ で表すことにしよう。これまでの話では、どうやら $\aleph_0 < \aleph$ であるようだ。

すると、ちょっとした疑問が湧いてくるだろう。$\aleph$ より濃い濃度を持つ集合はどんな集合だろう。また、$\aleph_0$ と $\aleph$ の間の濃度を持つ集合はあるのだろうか。残念なことに、その話題を続ける力量が私にはない。この先の大秘境に興味がある人は、足を踏み入れてみてはいかが？

ところで冒頭の有理数のすべてを一覧にした表だけど、これは Lua の仕業である。正直言って、ここは Lua の出番ではない。でも誘惑に勝てなかったんだ。コードのすべてを晒（さら）してもよいのだけど、さすがに一部だけにしておこう。ほかの部分は `b = 2` の一連の処理と少し違う程度なので察してもらいたい。

```
\directlua{
    tex.print(\asluastring{$\displaystyle{ \atop b~}\backslash{a\atop }$} %
                                                                    .. '&')
    for a = 1, 5 do tex.print(a .. '&&') end
    tex.print(\asluastring{$\dots$ \\ \hline})

    b = 1 （以下略）

    b = 2 tex.print(b .. '&')
    for a = 1, 5 do
        if (a == 2) or (a == 4) then
            tex.print(\asluastring{$\left(} .. a %
                 .. \asluastring{\over} .. b .. \asluastring{\right)$ &&})
        else
            tex.print('$' .. a .. \asluastring{\over} .. b .. '$ &&')
        end
    end tex.print(\asluastring{$\dots$ \\})
    tex.print(\asluastring{& %
                    $\downarrow$ & $\nearrow$ && $\swarrow$ && $\nearrow$ \\})

    b = 3 （以下略）
}
```

でも、Lua は TEX と合わせて使うことに意味があるので、少しばかりコードについて話しておきたい。ここではコンピュータプログラムにおける、重要な分岐処理を理解してほしい。コンピュータは繰り返し同じことをするのに苦痛を感じないが、思い通りの処理を自動的にしてくれるほど

気が利いているわけではない。われわれの希望どおりに選択処理をさせるためには `if` 文が使われる。

`if` 文は言語により書き方に特徴がある。Lua は `if`〜`then`〜`end` を用い、必要に応じ `else` をはさむ。Excel VBA と同じ書き方かな。でも Excel と違うのは**比較演算子**が `==` であることだ。Lua もご多分にもれず `=` を代入に、`==` を比較に使う。Excel の方がおかしいのだ。

注目してほしいのは `for` 文の繰り返し回数である。表の主部は 5×5 なのだから `for i = 1, 5` は当然に見えるかもしれない。実は TEX で作る表は 10×10 にしている。矢印を入れたいからだ。これを単にプログラミング言語だけで書くなら繰り返しは 10 回にし、奇数回目と偶数回目で分けて分数と矢印を出力するはずだ。しかし、そこは TEX である。

TEX は、`tabular` 環境で列の区切りは `&` で与えるので、`A & B` で `A` と `B` はとなりどうしの列に収まる。それを `A && B` とすれば、`A` と `B` の間が 1 列空く。そのため、繰り返し回数の `a`, `b` をそのまま分子, 分母に使えるのだ。原則的に TEX は順に文字を組む必要があるので、数と矢印、または `(  )` の有無が交互に出力されるように `if` 文や `or` 演算子が必要になった。

もっとも今回の表を作るなら、面倒であるけど TEX で一文字ずつ入力するのが最善だと思うけどね。プログラミングだけとか LuaTEX で効率的に書くなら、たとえば九九の表なんかが適している。

$*$	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

100 個の数と `&` をせっせと入力するのは根気がいるだろう。でも、この九九の表を出力したスク

リプトは『

```
\directlua{
    tex.print('$*$') for i = 1, 9 do tex.print('&' .. i) end
    tex.print(\asluastring{\\ \hline})
    for b = 1, 9 do
    tex.print(b)
        for a = 1, 9 do
            tex.print('&' .. a*b)
        end
        tex.print(\asluastring{\\})
    end
}
```

』である。手作業で表を作るより早いはずだし正確でもある。`for` 文を**入れ子**にして、内側の `for` を抜けるたびに改行すれば表は完成だ。

回り路しすぎたかな。濃度の話だったっけ。数学において、濃度の考えのもとである一対一の対応は重要な概念である。有理数の濃度が自然数の濃度と同じというのは、有理数が 1 ずつ増える自然数の性質を継承している表れでもある。だから有理数は自然数と一対一に対応する。しかし実数はそうではない。では、実数の本質はなんだろう。ここで詳しく述べられないが、**連続性**という性質が関わっている。興味が湧いたら少し調べてみるとよい。