1.3 濃度と集合

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		$\cdots$
1	$\frac{1}{1}$	$\rightarrow$	$\frac{2}{1}$		$\frac{3}{1}$	$\rightarrow$	$\frac{4}{1}$		$\frac{5}{1}$	$\rightarrow$	$\cdots$
		$\swarrow$		$\nearrow$		$\swarrow$		$\nearrow$			
2	$\frac{1}{2}$		$(\frac{2}{2})$		$\frac{3}{2}$		$(\frac{4}{2})$		$\frac{5}{2}$		$\cdots$
	$\downarrow$	$\nearrow$		$\swarrow$		$\nearrow$					
3	$\frac{1}{3}$		$\frac{2}{3}$		$(\frac{3}{3})$		$\frac{4}{3}$		$\frac{5}{3}$		$\cdots$
		$\swarrow$		$\nearrow$							
4	$\frac{1}{4}$		$(\frac{2}{4})$		$\frac{3}{4}$		$(\frac{4}{4})$		$\frac{5}{4}$		$\cdots$
	$\downarrow$	$\nearrow$									
5	$\frac{1}{5}$		$\frac{2}{5}$		$\frac{3}{5}$		$\frac{4}{5}$		$(\frac{5}{5})$		$\cdots$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\ddots$

( ) 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 $1, 2, 3, \ldots$ 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。確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

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

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

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

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。始めの仮定では、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 $\aleph_0$—$\aleph$ はヘブライ文字で “アレフ” と読む—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。

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 $\dfrac{1}{3} = 0.333\cdots$ 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、足を踏み入れてみるとよい。

濃度の話から少し離れてしまうかもしれないが、数学において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。**Haskell** は数の集合をどのようにとらえているのだろう。簡単な例で見ておこう。

たとえば数学では、1 以上 10 以下の整数の集合は

$$\{x \mid 1 \leqq x \leqq 10、x \text{ は整数}\} \quad \text{または} \quad \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\}$$

のように記述する。前者は**内包的記述**などと呼ばれ

$$x \text{ の集合、それは、} 1 \leqq x \leqq 10 \text{ である整数 } x \text{ を用いる}$$

といった意味合いの書き方である。後者は**外延的記述**などと呼ばれ、早い話、具体的な**要素**を列挙する書き方である。**Haskell** は内包的記述を与えると、外延的記述を返してくれる。

(ghci env.)

```
Prelude> [x | x <- [1..10]]
[1,2,3,4,5,6,7,8,9,10]
```

`{ }`は `[ ]` で代用されているが、`[x | ...]` の書き方はまさに数学の記述そのものである。**Haskell** が内包的記述をするときは整数を列挙することが前提であるから、実数の集合を意味するような、単に $1 \leqq x \leqq 10$ とする書き方は実情に合わない。実際、数学においては「x は整数」を補足する必要がある。このことから **Haskell** は `x <- [1..10]` と書くことで、1 から 10 までの整数から x を取り出す、という意味になるのである。

これは信じられないぐらい便利な記法である。それを

$$\{(a,\ b,\ c) \mid a^2 + b^2 = c^2、0 < a < b < c < 20\}$$

を例にとって示すことにしよう。この集合は、斜辺の長さを c とする直角三角形のうち、3 辺とも長さが 20 未満のものの組を表している。もちろん、ここで **Haskell** が処理できる辺の値は整数値に限るけど。また、使った条件は**三平方の定理の逆**である。一般に、三平方の定理とは

$$\text{直角三角形において、} a^2 + b^2 = c^2 \text{ が成り立つ}$$

という方向で使う。いま使った条件はこの逆の方向で、$a^2 + b^2 = c^2$ が成り立つ三角形は、直角三角形である、なのだ。細かいことだが注意したい。

さて、**Haskell** はさっきの条件を次のように処理する。

(ghci env.)

```
Prelude> [(a,b,c) | c <- [1..19], b <- [1..c], a <- [1..b], a^2+b^2 == c^2]
[(3,4,5),(6,8,10),(5,12,13),(9,12,15),(8,15,17)]
```

1 行のスクリプトで結果が表示されることにはびっくりだろうが、スクリプトの記述は数学の記述とぴったり同じとはいかない。**Haskell** にはそれなりの事情があるのだ。

まず、a, b, c の値をまとめて取り出す記述はできない。必ずひとつずつ `c <- [1..19]` のように書く。ただし、条件では $a < b < c$ であったから b の値は `[1..19]` から取り出すのではなく、`[1..c]` から取り出せばよいことになる。しかし、$b < c$ であって $b \leqq c$ ではないのだから、本当は `[1..c]` ではなく `[1..c-1]` と書くべきなのだが、庭いじりを始めたばかりで細部にこだわることはやめておいた。まあ、そのため結果的に $0 < a \leqq b \leqq c < 20$ という条件にすり替わったことになったんだけどね。

$0 < a < b < c < 20$ と同等な一連の条件と $a^2 + b^2 = c^2$ の条件が、例に示した数学の記述とは逆の順に書いたことに注意してほしい。先に `a^2+b^2 == c^2` を記述すると、**Haskell** には `a`, `b`, `c` をいくつに決めてよいか分からなくて、“Variable not in scope: a :: Integer” などのエラーを発する。したがってこの例の場合は、`c` は何よりも先に値を決めてやらなくてはいけないため、条件の先頭に書くのである。よって、同じ理由で `b`、`a`、`a^2+b^2 == c^2` の条件はこの順番以外にあり得ないことになる。

ちょっと気難しい面もあるが、それより、わずか 1 行のスクリプトで求めたい結果が列挙されることに感心しよう。結果から、(6, 8, 10) と (9, 12, 15) は単に (3, 4, 5) の組の整数倍だから、辺の長さが 20 未満の整数値直角三角形は、本質的に (3, 4, 5)、(5, 12, 13)、(8, 15, 17) だけであることが分かるのである。