# 5 論理演算

集合を基本として数を構成してみたものの、いまひとつしっくりこない体系になってしまった。そこで今度は、計算（演算）に主眼を置いて数を構成してみよう。

旅のコースから多少はずれるかもしれないが、集合から少し離れてすでに数があるものとしてみたい。まず、0 である。0 の概念は “何もない” ことだから、集合では $\emptyset$ にあたるだろう。何もないと話が進まないので、集合の概念で用いた $\{\emptyset\}$ の代用である 1 も用意しよう。0 と 1 を用いてうまく演算ができれば、それを拡張して一般的な計算に発展させられるだろう。

まずは **PowerShell** に演算をしてもらうことにする。

[ps script]
```
PS C:\Users\Yours > @"
>> 0 -and 0
>> 0 -and 1
>> 1 -and 0
>> 1 -and 1
>> "@ > test.ps1
>>
PS C:\Users\Yours > ./test
>> False
>> False
>> False
>> True
```

ヒア文字列の機能を使って、4 つの演算をまとめて行った。これだけでは何をやっているか分からないだろうから、演算の仕組みについて説明しておこう。**PowerShell** は基本的に**論理演算**で処理をしている。論理演算とは、**真**か**偽**で判定できることがらを組み合わせて処理することで、処理した結果も真か偽となる演算である。これは **PowerShell** に備わった機能というより、コンピュータにできる唯一の処理方法なのである。コンピュータは電気で動作する機械なので、コンピュータに分かることは電気が流れたか流れないかとか、電圧がかかったかかからないかという状態を二者択一するしかないのだ。**PowerShell** はそれを `True` と `False` で表すのである。

ただし、私たちは `True` と `False` で日常の計算を行っていないので、コンピュータに計算を委ねるときは、私たちが `True`, `False` で思考するか、コンピュータに `1`, `0` で処理してもらうしかない。当然、コンピュータに歩み寄ってもらわなくては困る。そこで私たちが 0, 1 を使えば、コンピュータがそれらを `False`, `True` と解釈してくれる仕様である。

さあ、それで-and 演算である。-and 演算を日常の言葉で言えば**かつ**である。A -and B とは A かつ B を意味する。日常の感覚と同じく **PowerShell** においても、-and 演算が真になる場合は、A と B がともに真であるときに限る。これが、先の結果だ。

**PowerShell** が行う基本的なもうひとつの論理演算子に-or がある。-or 演算を日常の言葉で言えば**または**である。A -or B とは A または B を意味する。

[ps script]

```
PS C:\Users\Yours > @"
>> 0 -or 0
>> 0 -or 1
>> 1 -or 0
>> 1 -or 1
>> "@ > test.ps1
>>
PS C:\Users\Yours > ./test
>> False
>> True
>> True
>> True
```

ここでも日常感覚にしたがって、-or 演算が偽になる場合は、A と B がともに偽であるときに限る。一覧表にまとめてみよう。

| -and | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| -or | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

表をじっくり見ると-and は $\times$（掛ける）に置き換えられることに気づく。つまり、-and 演算子は掛け算の代用になるということだ。-or 演算子はどうだろうか。一見、$+$（足す）に置き換えられそうに思える。だが、残念ながら 1 -and 1 がうまくない。ここは 2 になってもらいたいところだが、この時点で 2 を定義できていないので 2 にならないのはやむを得ない。ならば、1 -and 1 は 1 であると定義するのが自然だろうか。

しかし、そのように定義したのでは、先ほど $\emptyset$ と集合の概念から数を構成したものと同じである。この体系は私たちが望むものではないのだ。私たちが望むのは $1+1=0$ である。なぜなら、いま使える数が 0 と 1 しかない状態だから、$1+1$ には新たな考えが必要となる。それは、新しい数の導入か新しい表記の導入であろう。

新しい数の導入では、数を構成したことにならない。2 を、0 と 1 から作ってこそ構成と言える。現状で処理できないことが出るたびに新たな数や記号を作ったのでは、何でもありの無法状態と変わるところはなく、

論理に基づいた考えとは言えないからだ。

0 と 1 しか使えないなら、新しい表記を導入するのがよい。桁を増やすのである。$1+1=10$ なら問題ないし、これで数を

$$0, 1, 10, 11, 100, 101, 110, 111, 1000, \ldots$$

の順で構成できる。いわゆる 2 進数であるが、旅の目的からすると本末転倒というところだろう。2 進数は数が構成できている前提の上に、その表記が成り立っているのだから。