Chapter 12

セル・オートマトン

Mathematica の開発者スティーブン・ウルフラム (Stephen Wolfram) はセル・オートマトン研究で若くして名を馳せた人物である．*Mathematica* にも当然のように，セル・オートマトンを生成する環境が整っている．そのための組込み関数も存在するが，本章ではまずセル・オートマトンの仕組みを理解した上で，自作することから始めてみよう．

12.1　セル・オートマトンとは

セル・オートマトン (cellular automaton) とは何か．それを一言で定義するのは難しいので，とにかく具体例を見てみよう．

まず**セル** (cell, 細胞) とよばれる正方形 n 個が一列に並んでいる．ここで n は十分大きな自然数で，たとえば 100 ぐらいの具体的な数をイメージしておこう．各セルはそれぞれ白か黒のふたつの**状態** (state) を持っており，時間とともに状態は変化する (図 12.1 左)．

図 12.1　左：セルの配置．右：両端のセルはつながっていると考える．

さて時刻 0 からスタートして，時刻 $t = 0, 1, 2, \ldots$ における i 番目 $(1 \leq i \leq n)$ のセルの状態 $c_i(t)$ を

$$\text{白のとき：}\ c_i(t) = 0 \qquad \text{黒のとき：}\ c_i(t) = 1$$

と表すことにする．さらに，状態 $c_i(t)$ を決めるひとつの原則として，

各セルの時刻 $t+1$ における状態は，時刻 t における自分自身，および隣り合うセルの状態のみで決定される

とする．すなわち，

$$c_i(t+1) \text{ は } c_{i-1}(t),\, c_i(t),\, c_{i+1}(t) \text{ のみで決定される}$$

のである．ただし（あまり本質的でない，技術的な仮定として）$c_0(t) = c_n(t)$, $c_{n+1}(t) = c_1(t)$ と約束しておく．この条件から，1 番目と n 番目のセルが隣り合っていて，セル全体が図 12.1 右のように円環状に並んでいると考えることができる．

ここではより具体的に，漸化式

$$c_i(t+1) \equiv c_{i-1}(t) + c_i(t) + c_{i+1}(t) \mod 2$$

が成り立っていると仮定しよう．また $n = 101$ とし，時刻 0 の時点でちょうど真ん中にあたる 51 番目のセルだけが黒，あとは白であったとする：

時刻 0：　$\cdots$□□□□□□□□□□■□□□□□□□□□□$\cdots$

このセルたちの状態は，時間とともにどう変化していくのだろうか？

式に基づいて計算していってもかまわないが，あらかじめ隣り合う 3 つのセルの状態として考えられるものを列挙して，その結果を計算しておくほうがよいだろう．具体的には，次のような表を作成すればよい：

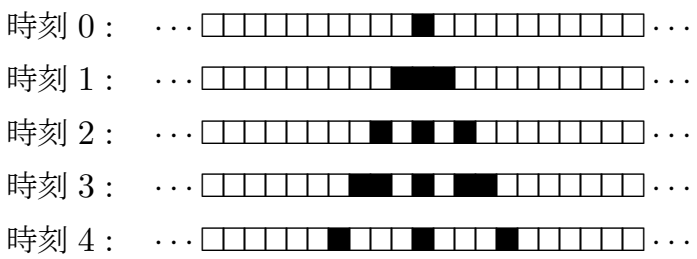
$c_{i-1}(t)$	$c_i(t)$	$c_{i+1}(t)$	$c_i(t+1)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

$\Longleftrightarrow$

t	$t+1$
■■■	■
■■□	□
■□■	□
■□□	■
□■■	□
□■□	■
□□■	■
□□□	□

とくに左側の表のことを便宜的に「ルール表」とよぶことにする．このルール表に基

づいて計算していくと，次のように状態が変化していくことがわかる：



この作業を延々と続けていったとき，いったいどのようなパターンが現れるのだろうか？

一般に，「複数の状態をもつセル」の集合に，「セルの状態が近傍のセルの状態だけで決定される」ような一定の**ルール**を付与して得られる系（システム）が**セル・オートマトン**とよばれるものである．この例の場合，セルは 1 次元的に並んでいるので，1 次元オートマトンともよばれる．2 次元，3 次元のオートマトンがどんなものかも，容易に想像がつくだろう．セル・オートマトンを用いると，銀河形成から人工生命まで，さまざまな現象（とくに，組織化をともなう現象）をモデル化できるという．

本章では *Mathematica* によって 1 次元オートマトンの作業を自動化し，具体例を観察してみよう．

12.2 ArrayPlot による行列の表現

セル・オートマトンを表現（図示）するには，組込み関数 `ArrayPlot` を用いるのが便利である．`ArrayPlot` は行列の各要素を正方形のセル（もしくはピクセル）で表現する．ただし，ここでいう「行列」とは，「リストのリスト」のことである．

[1] 0 と 1 によるランダムな 5 × 10 行列：

```
In[1]:= m = Table[RandomInteger[], {5}, {10}];
```

[2] `TableForm` で表現：

```
In[2]:= m //TableForm
```

```
Out[2]//TableForm=
0  1  0  1  1  1  0  0  0  0
0  1  0  0  0  1  0  1  0  0
1  1  0  1  0  1  0  1  1  0
0  0  1  1  1  0  1  0  0  1
0  0  1  0  0  1  1  0  1  0
```

[3] `ArrayPlot` で表現：

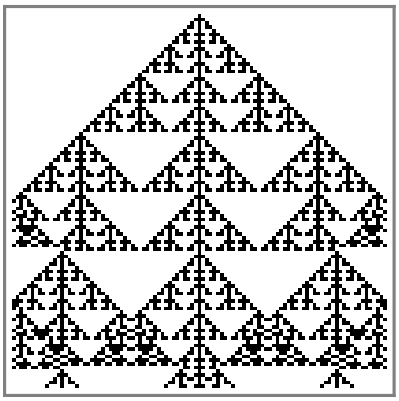
```
In[3]:= ArrayPlot[m]
```

ある．前章で学んだ組込み関数 NestList で一気に作業を自動化しよう．

[20] NestList を用いて，時刻 0 の c0（初期状態）から時刻 100 までの時間発展を縦方向に並べる：

```
In[20]:= ArrayPlot[ NestList[next, c0, 100] ]
```

Out[20]=



このように，意外と複雑なパターンが現れる．

問題 12.3（その他のルール） 以上を応用して，次のように変化させよ．

(1) 同じ初期状態 c0 を用い，[14] の関数 f を次の g に変える：

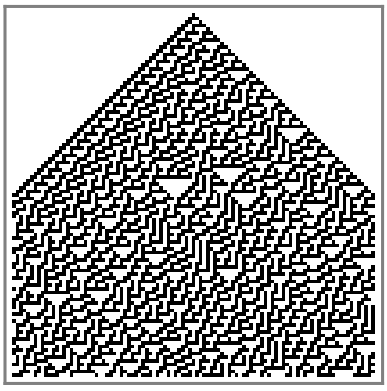
```
g[x_, y_, z_] := Mod[x + y + z + y*z, 2];
```

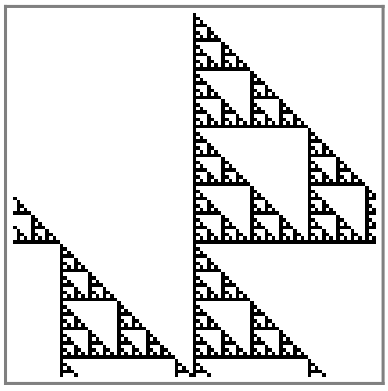
(2) 同様に，関数 f を次の h に変える：

```
h[x_, y_, z_] := Mod[x + y, 2];
```

(3) (1) の初期状態 c0 を，0 と 1 からなるランダムな列 d0 に変える．

【解答】 (1)(2) は指示通りにやるだけである．結果は次のようになる：





(1) は漸化式

$$c_i(t+1) \equiv c_{i-1}(t) + c_i(t) + c_{i+1}(t) + c_i(t)c_{i+1}(t) \mod 2$$

が生成するパターンである．じつはこのパターンによく似た模様がイモ貝の模様として現れることが知られている（図 12.2）．

一方，(2) は漸化式

$$c_i(t+1) \equiv c_{i-1}(t) + c_i(t) \mod 2$$

が生成するパターンである．最初のほうはパスカルの三角形とよく似たパターンが現れるが，これらは実際同じものなのである．その理由は，二項係数がみたす式

$$_{n+1}\mathrm{C}_k \equiv {}_n\mathrm{C}_{k-1} + {}_n\mathrm{C}_k \mod 2$$

を考えればすぐにわかるだろう．またこのパターンから，セルが円環状に配置されていることも観察できる．

ほかにもルールを自作して，どのようなパターンが得られるか試してみるのも面白いだろう．

(3) 次のように初期値 d0 を定めればよい．

```
In[ ]:=  d0 = Table[RandomInteger[], {101}]
```

あとは同じなので，結果は省略する．こちらもぜひ試してみてほしい． ■

図 12.2 イモ貝の殻（出典：Wikipedia）

12.6 組込み関数によるセル・オートマトン

今度は楽をして，組込み関数 CellularAutomaton で 1 次元セル・オートマトンを生成しよう．そのまえに，1 次元セル・オートマトンの「整理番号」について説明しておく必要がある．

いま，時刻 $t+1$ における i 番目のセルの状態 $c_i(t+1)$ は自分自身と隣り合うセルの状態 $c_{i-1}(t)$, $c_i(t)$, $c_{i+1}(t)$ のみで決定される，と仮定していた．このとき，考えうるルールは全部で何通りあるだろうか？

12.1 節のルール表のように，$c_{i-1}(t)$, $c_i(t)$, $c_{i+1}(t)$ の可能な組み合わせ（8 通りある）に対して $c_i(t+1)$ の値（0 もしくは 1）を好き勝手に決めたとすると，$2^8 = 256$ 通りのルールが存在することになる．