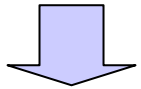
2. セルラ・オートマトン（Cellular Automata）

オートマトン（automata）：　自動人形　→　自動機械，順序機械

[John von Neuman]

自己複製オートマトン：　局所近傍則を備えた自己増殖プログラム，離散系



セルラ・オートマトン

2．1　セルラ・オートマトン（CA)の一般事項

（1）セルラ・オートマトンの定義

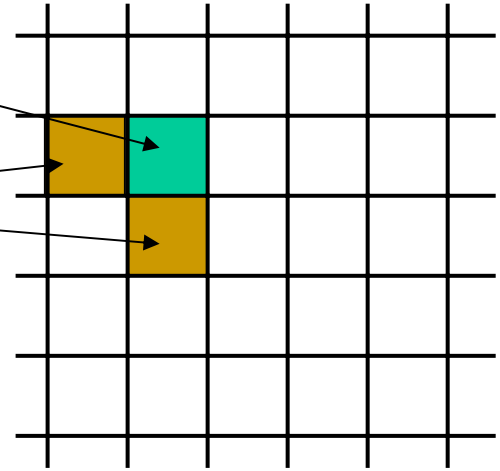
解析空間をセルと称する離散的領域に分割し，セル同士の簡単な規則を与えることで各セルの状態量を時間を追って決定することにより動的なパターンを再現する手法.

創発の概念（全体的な構造形態が局所的活動から生まれる）

① 不連続な系

いろいろな初期値をもつ格子サイト

② いくつかの局所的な隣接サイトの値に基づいた新しい値



⟶ 不連続な時間ステップで発展

⤴ 有限な数の過去の時間ステップを仮定

(2)CA格子

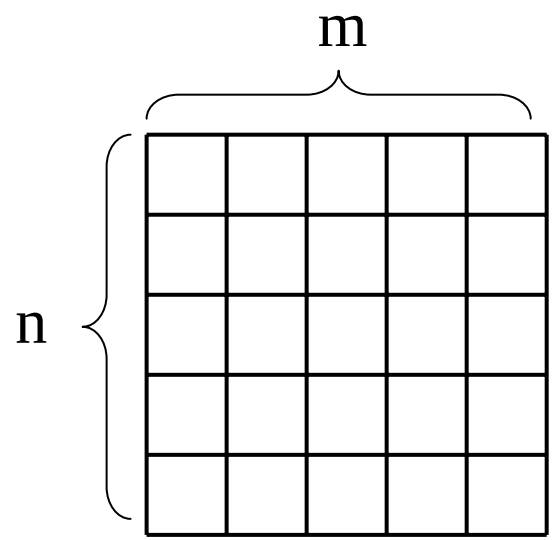
[1次元]

セルラ・オートマトン＝線形リスト　　Table[expr, {i, 1, s}]

数値，シンボルまたは両方を評価 ⤴

［2次元］

長方形格子，三角格子，六方格子



n×mの長方形格子

```
Table[expr, {i, 1, n}, {j, 1, m}]
```

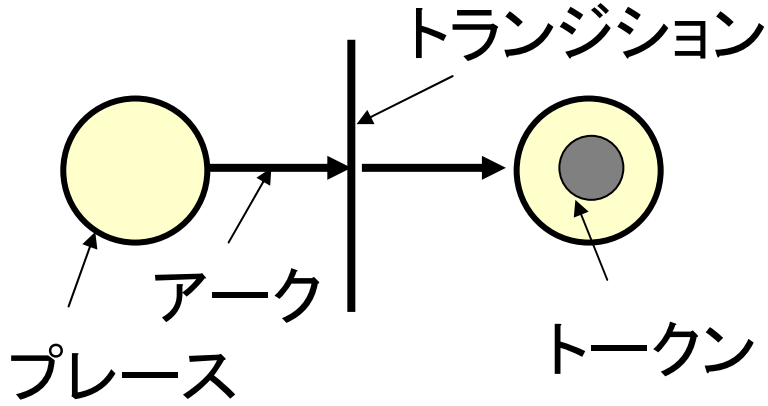
各種2次元セルラ・オートマトン

ライフゲーム，格子ガスオートマトン，
ペトリネット，Lシステム，マルチエージェントシステム

（ペトリネット）

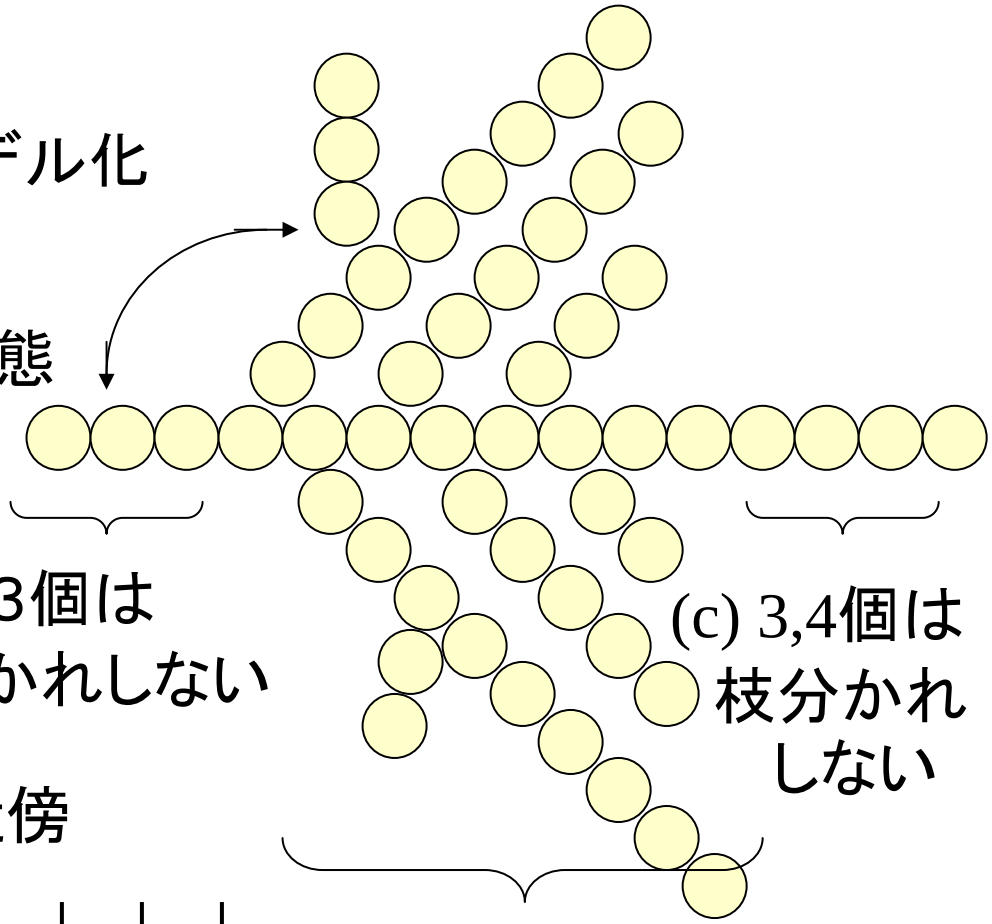
- 離散現象に対するモデル化の一手法，グラフィックモデル
- 並列的，非同期的な振舞をするシステムのモデル化
- 回路，コンピュータOSの設計

前提条件： 入力プレース
事象の生起： トランジション
結果： 出力プレース



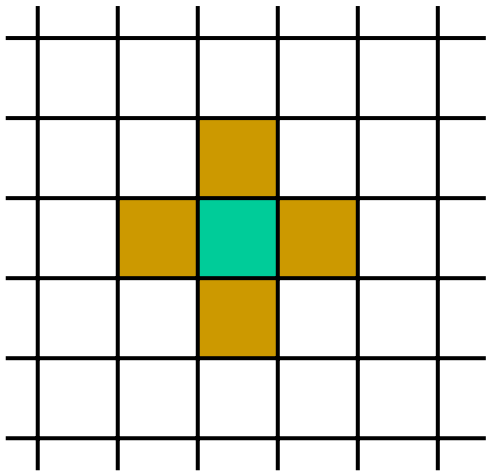
（Lシステム）

- 生物の発生，成長過程のモデル化
- 情報処理系の数学モデル

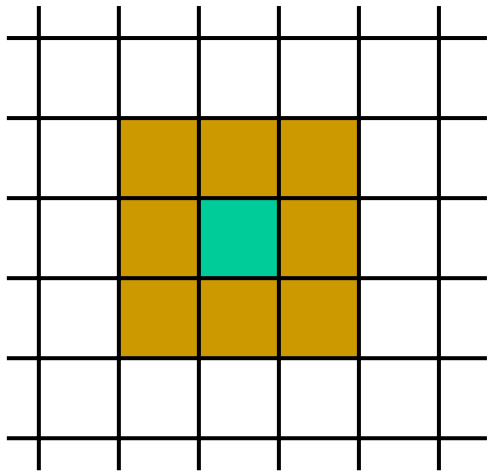


（3） CA近傍

ノイマン近傍



ムーア近傍



(4)境界

1	2	3
4	5	6
7	8	9
10	11	12

単純格子のムーア型近傍

[周期的境界]

12	10	11	
3	**1**	2	3
6	4	5	6
	7	8	9
	10	11	12

[吸収端境界]

0	0	0	0	0
0	**1**	2	3	0
0	4	5	6	0
0	7	8	9	0
0	10	11	12	0
0	0	0	0	0

[斜行境界]

9	10	11	
12	**1**	2	3
3	4	5	6
	7	8	9
	10	11	12

２．２　ライフ･ゲーム

John Horton Conway (Cambridge University)

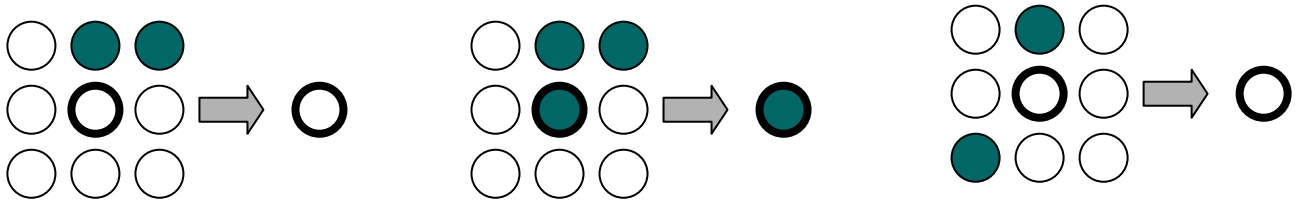
- 人工生命体システムの先駆け
- コンピュータ上での“知的エージェント”
- 世界最初の並列計算機“結合されたマシン”のプログラム
- セルラオートマトンの遷移規則　⇒　遺伝子にコーディング
- 大規模なセルラオートマトン　⇒　神経回路網
- 人間の脳＝数億個の素子からなる回路網 ⇒ 自律的に構成

（ルール）

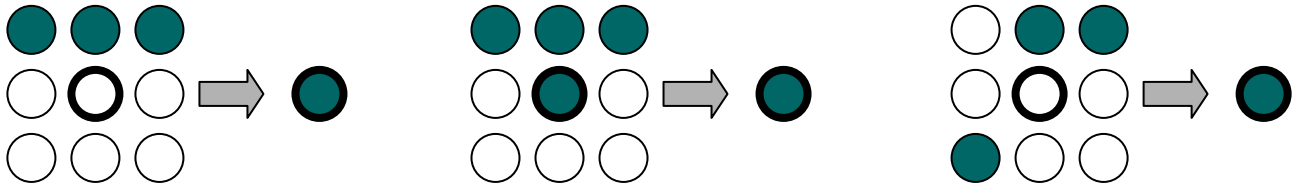
- 周期境界条件，２次元正方格子
- ムーア近傍（周囲の８個のセル）の状態で次の時刻のそのセルの状態を決定
- 周囲に仲間がいない場合も，多すぎる場合も死滅し，ちょうど適当な仲間がいるのが健全

（1）1が2個なら変化なし

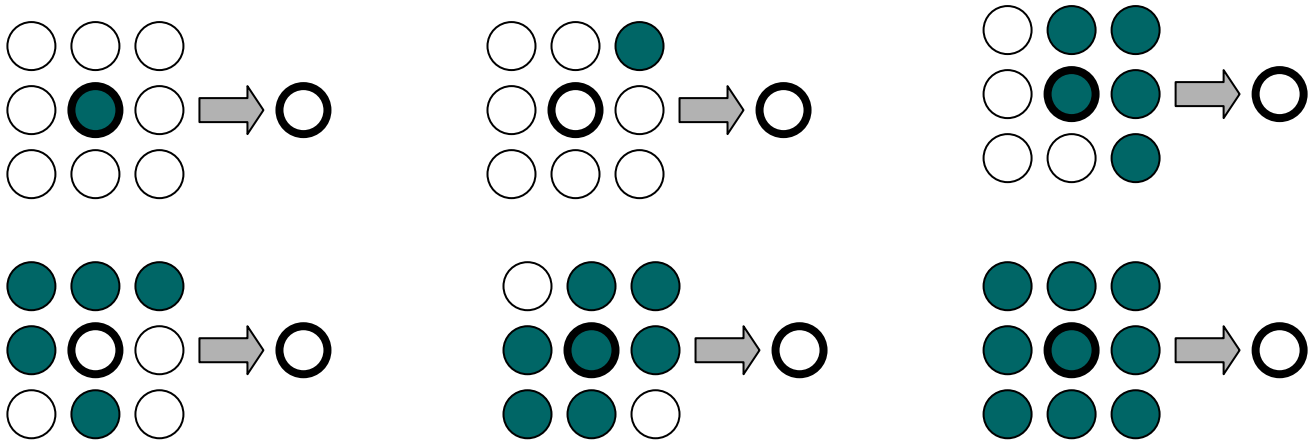
1（●），0（○）



（2）1が3個なら1

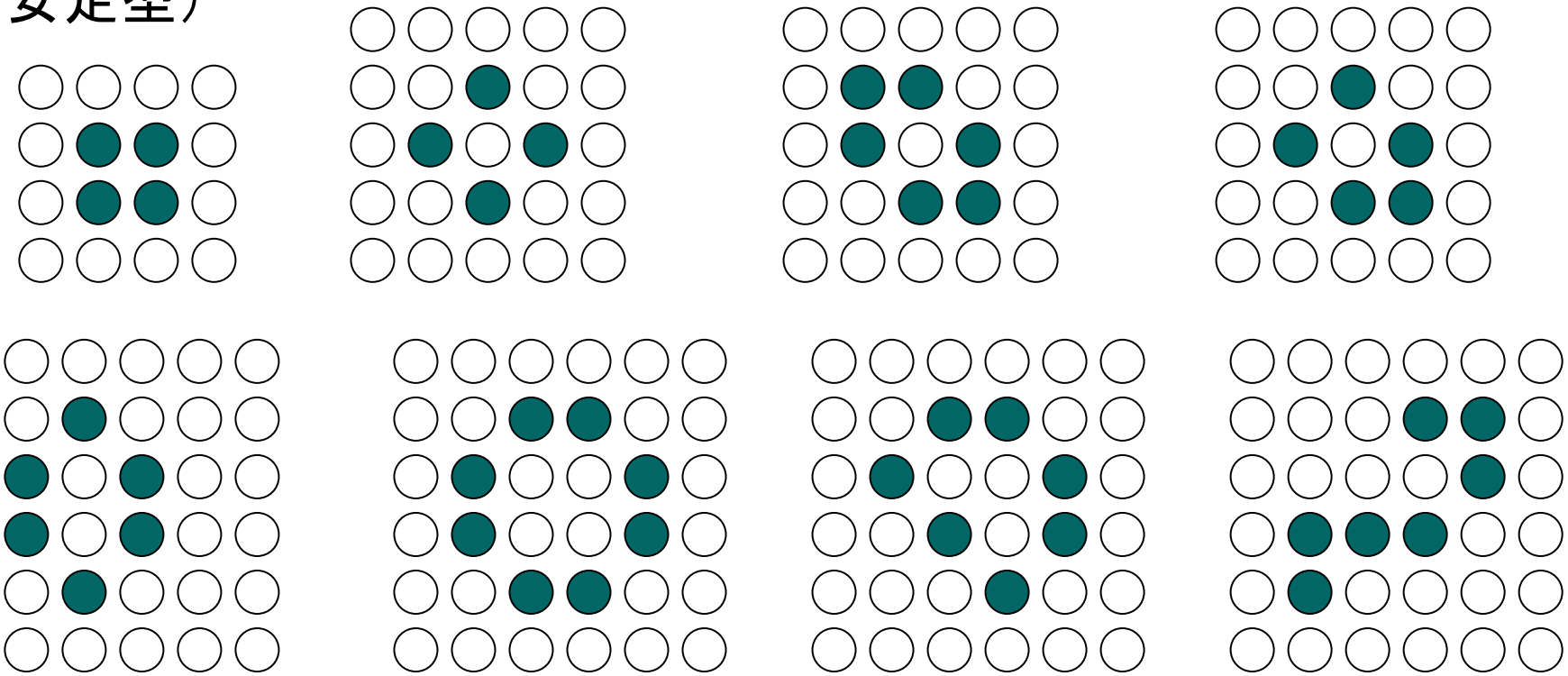


（3）（1）（2）以外は0

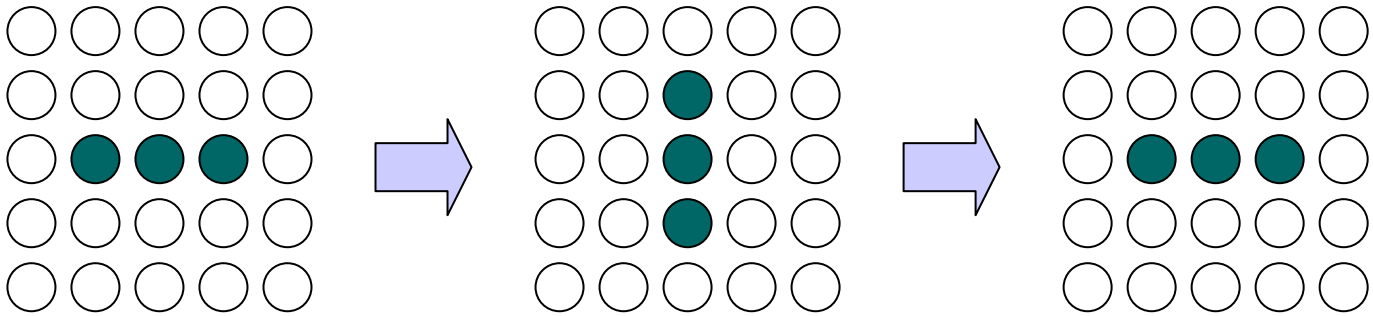


［各種パターン］

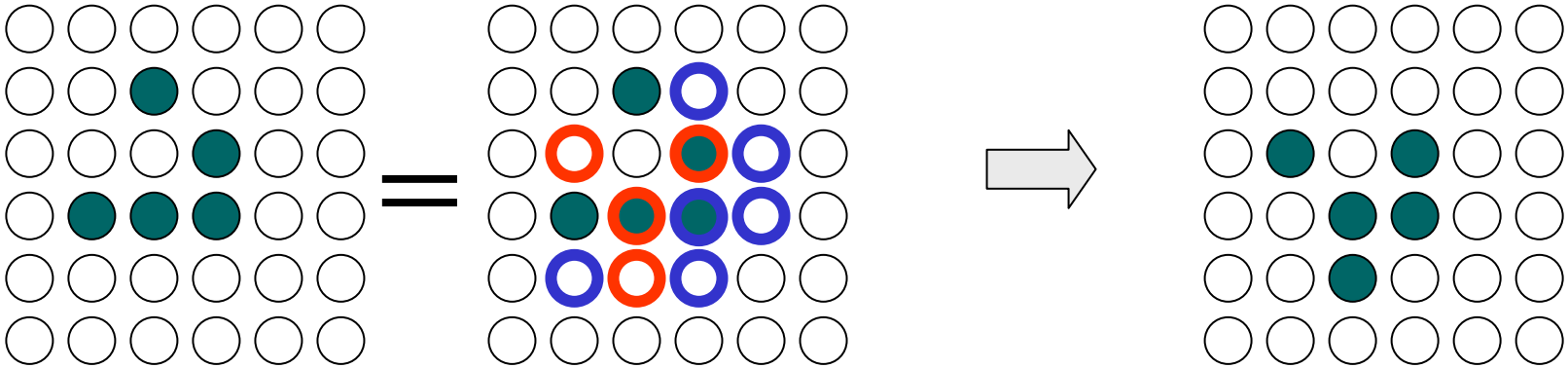
（安定型）



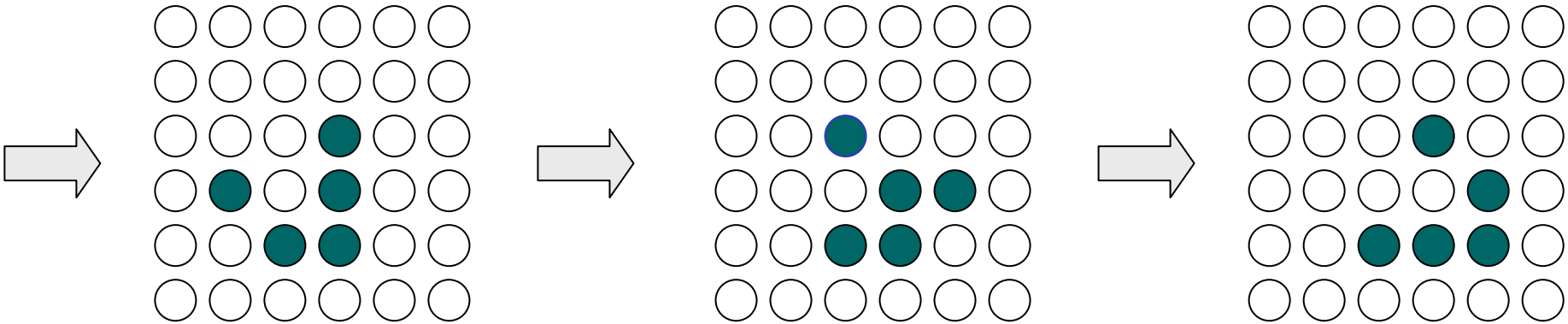
（振動型）
プリンカ



（移動型）グライダ



周辺の1の数
○ 3　○ 2



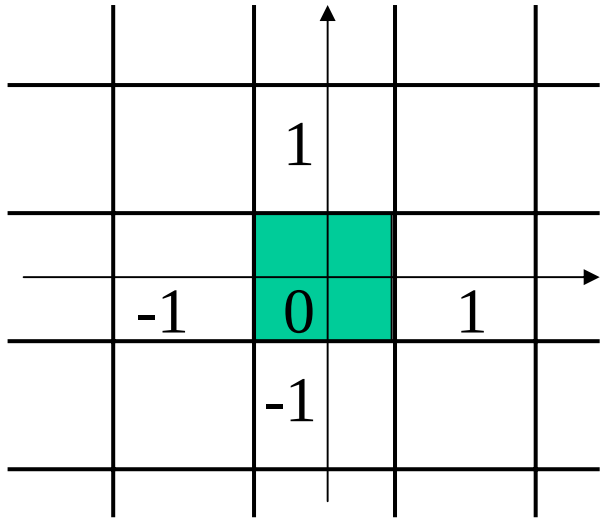
アルゴリズム（mathematica）

(1)初期配置

```
initconfig = Table[Random[Integer], {s}, {s}]
```

(2)要素が生きている最近接サイトの数を格子マトリクスに割当てる

```
In[1] := livingNghbrs[mat_] := Apply[Plus,
                Map[RotateRight[mat, #]&,
                  {{-1, -1}, {-1, 0}, {-1, 1}, {0, -1},
                   {0, 1}, {1, -1}, {1, 0}, {1, 1}}]]
```



(example)

In(2) := (board = Table[Random[Integer], {4}, {4}]) //MatrixForm

の出力が

Out[2]//MatrixForm =

0	1	1	1
0	1	0	1
0	0	0	0
1	0	1	0

の場合に，出力 Out[3]//MatrixForm および Out[4]//MatrixForm を求めよ.

ただし，各関数を以下とする.

```
In[3] := (livingNghbrs[board]) //MatrixForm

In[4] := bc = Join[{Last[#]}, #, {First[#]}]&;
                    Partition[bc[Map[bc, board]],
                                {3, 3}, {1, 1}]//MatrixForm
```

board =

0	1	1	1
0	1	0	1
0	0	0	0
1	0	1	0

```
livingNghbrs[mat_] := Apply[Plus, Map[RotateRight[mat, #]&,
   {{-1, -1}, {-1, 0}, {-1, 1}, {0, -1},{0, 1}, {1, -1}, {1, 0}, {1, 1}}]]
```

```
In[3] := (livingNghbrs[board]) //MatrixForm
```

Out[3]//MatrixForm =

5	4	5	4
4	2	5	2
3	3	3	3
2	4	3	4

```
In[4] := bc = Join[{Last[#]}, #, {First[#]}]&;
                Partition[bc[Map[bc, board]],{3, 3}, {1, 1}]//MatrixForm
```

Out[4]//MatrixForm =

0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	1	1	1	1	1	1	0
1	0	1	0	1	0	1	0	1	0	1	0
1	0	1	0	1	1	1	1	1	1	1	0
1	0	1	0	1	0	1	0	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	1	1	1	1	1	1	0

(3)生死のルール

```
update[1, 2] := 1
update[_, 3] := 1
update[_, _] := 0
```

exampleの正方格子に対して，次の関数の出力
Out[5]//MatrixForm, Out[6]//MatrixForm を求めよ.

（関数）

```
In[5] := Attributes[g] = Listable;
         g[board, livingNghbrs[board]] //MatrixForm

In[6] := update[1, 2] := 1
         update[_, 3] := 1
         update[_, _] := 0
         Attributes[update] =Listable;
         update[board, livingNghbrs[board]] //MatrixForm
```

Out[5]//MatrixForm =

g[0, 5]	g[1, 4]	g[1, 5]	g[1, 4]
g[0, 4]	g[1, 2]	g[0, 5]	g[1, 2]
g[0, 3]	g[0, 3]	g[0, 3]	g[0, 3]
g[1, 2]	g[0, 4]	g[1, 3]	g[0, 4]

Out[6]//MatrixForm =

0	0	0	0
0	1	0	1
1	1	1	1
1	0	1	0

```
update[1, 2] := 1
update[_, 3] := 1
update[_, _] := 0
```

(4) 格子に変化がなくなるか，t 回のステップまで更新される

```
evolution = FixedPointList[update[#, livingNghbrs[#]]&,
                           initconfig, t]
```

ライフ･ゲームのプログラム

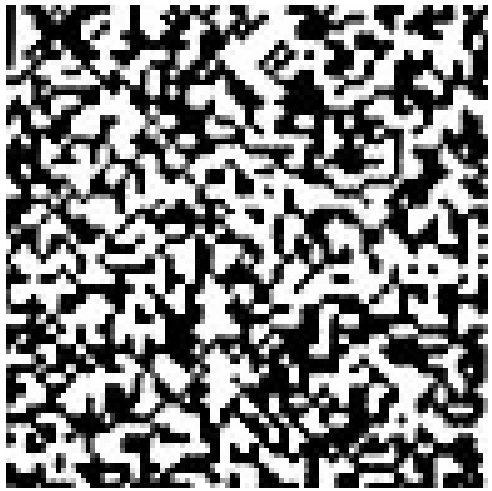
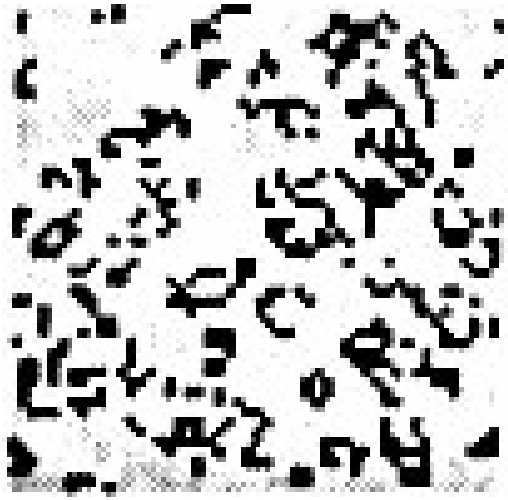
```
In[1] := LifeGame[s_, t_] :=
            Module[{initconfig, livingNghbrs, update},
                initconfig = Table[Random[Integer], {s}, {s}];
                livingNghbrs[mat_] :=
                  Apply[Plus, Map[RotateRight[mat, #]&,
                                  {{-1, -1}, {-1, 0}, {-1, 1}, {0, -1},
                                   {0, 1}, {1, -1}, {1, 0}, {1, 1}}]];
                  update[1, 2] := 1;
                  update[_, 3] := 1;
                  update[_, _] := 0;
                  Attributes[update] =Listable;
                  FixedPointList[update[#, livingNghbrs[#]]&,
                                 initconfig, t]
                ]
```

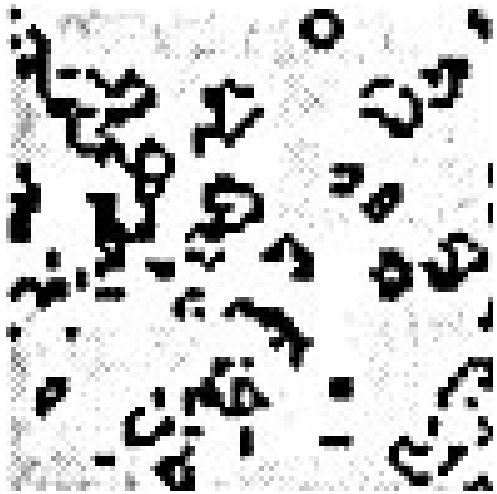
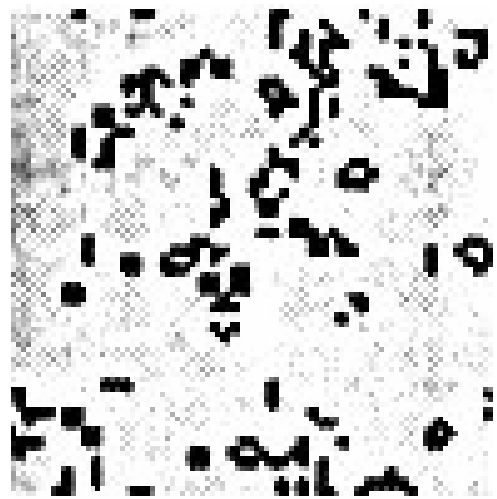
（グラフィック表示）

```
In[3] := Showlife[list_, opts_ _ _Rule] :=
                Map[(Show[Graphics[RasterArray[
                        Reverse[list[[#]]/.
                                {1 -> RGBColor[1, 0, 0],
                                 0 -> RGBColor[0, 0, 0]}]],
                        AspectRatio ->Automatic,
                        opts]])&,
                    Range[Length[list]]]
```

アニメーション・セル

50×50格子

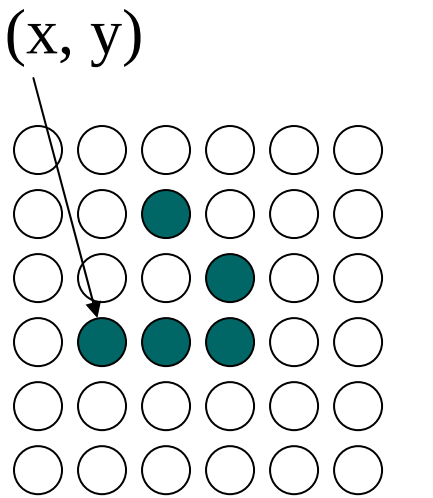
（生命体，life-forms）

グライダ（glider）

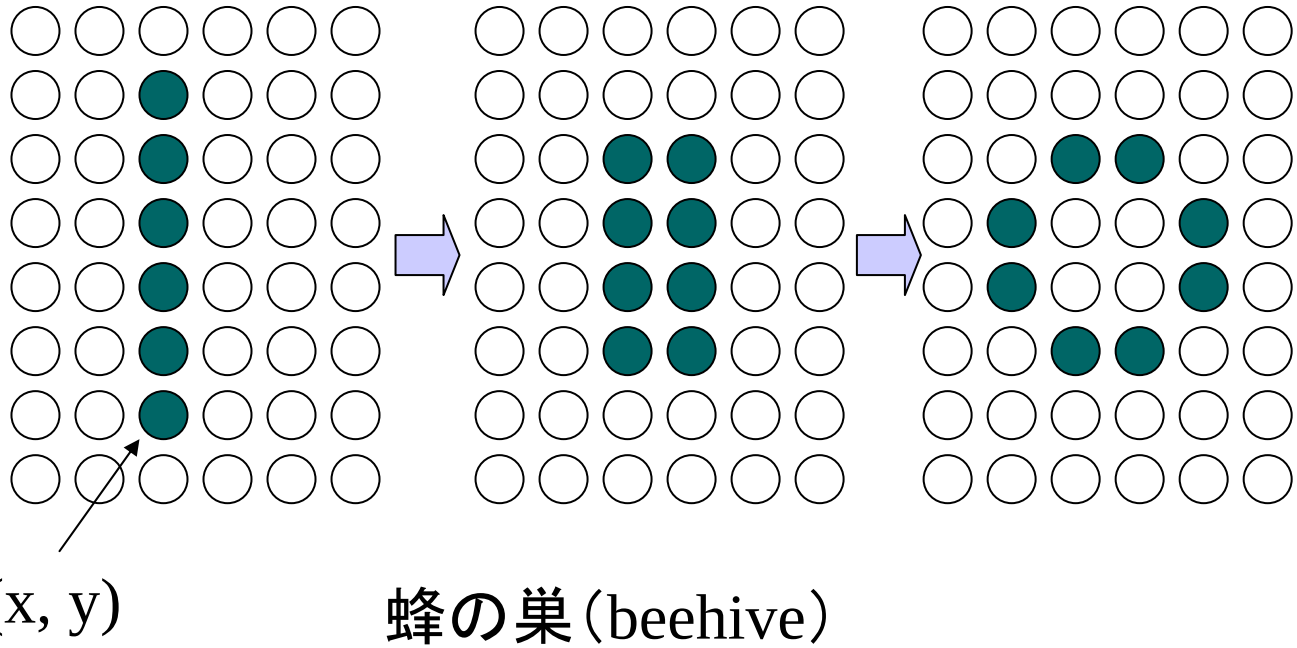
```
glider[x_, y_] :={{x, y}, {x+1, y}, {x+2, y},
                  {x+2, y+1}, {x+1, y+2}}
```

蜂の巣（beehive）

```
beehive[x_, y_] :={{x, y}, {x, y +1}, {x, y +2}, {x, y +3},
                   {x, y+4}, {x, y+5}, {x, y+5}}
```



グライダ（glider）

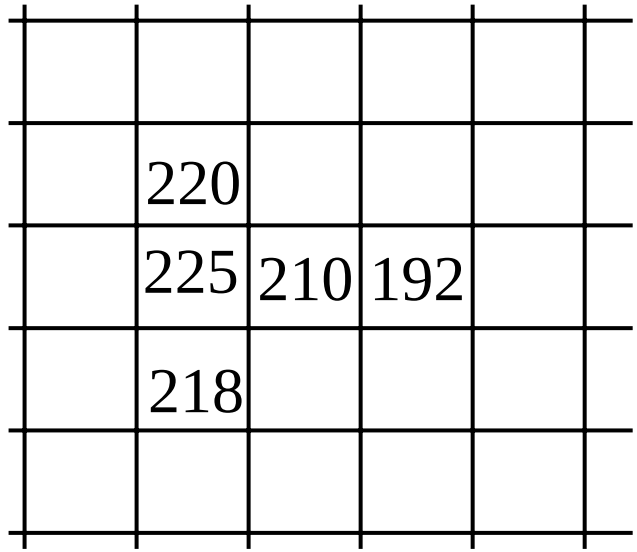


蜂の巣（beehive）

（拡散）

空間中を分子が広がる様子，
物質中の熱伝導のモデル

r =0~255
rは濃度，温度を表す．



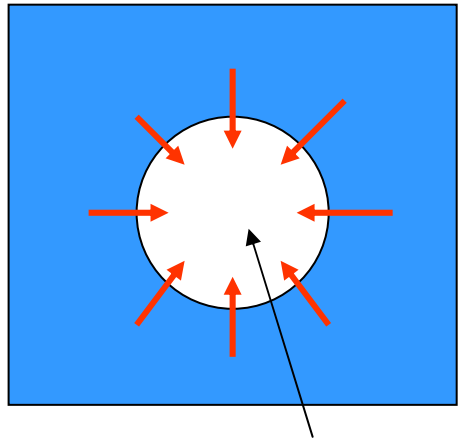

```
Melt[r_, s_, t_] := Module[{init, ngbrsAve},
    init = Table[Random[Integer, {0, r}], {s}, {s}];
    ngbrsAve[mat_] := Floor[Apply[Plus,
                        Map[RotateRight[mat, #]&,
                            {{-1, -1}, {-1, 0}, {-1, 1}, {0, -1},
                              {0, 1}, {1, -1}, {1, 0}, {1, 1}}]]/8];
    NestList[ngbrsAve, init, t]]
```

（沸騰）

液体から気体への変態をモデル化

- 更新されるサイトの値 $\begin{cases} x < r & \Rightarrow \quad x \\ x \geq r & \Rightarrow \quad 0 \end{cases}$

x = 8つの最近接サイトの平均足す1




```
Rug[r_, s_, t_] := Module[{init, ngbrsAve},
    init = Table[Random[Integer, {0, r-1}], {s}, {s}];
    ngbrsAve[mat_] := Floor[Apply[Plus,
                        Map[RotateRight[mat, #]&,
                            {{-1, -1}, {-1, 0}, {-1, 1}, {0, -1},
                             {0, 1}, {1, -1}, {1, 0}, {1, 1}}]]/8];
    NestList[ngbrsAve[#] +1], r]&, init, t]]
```

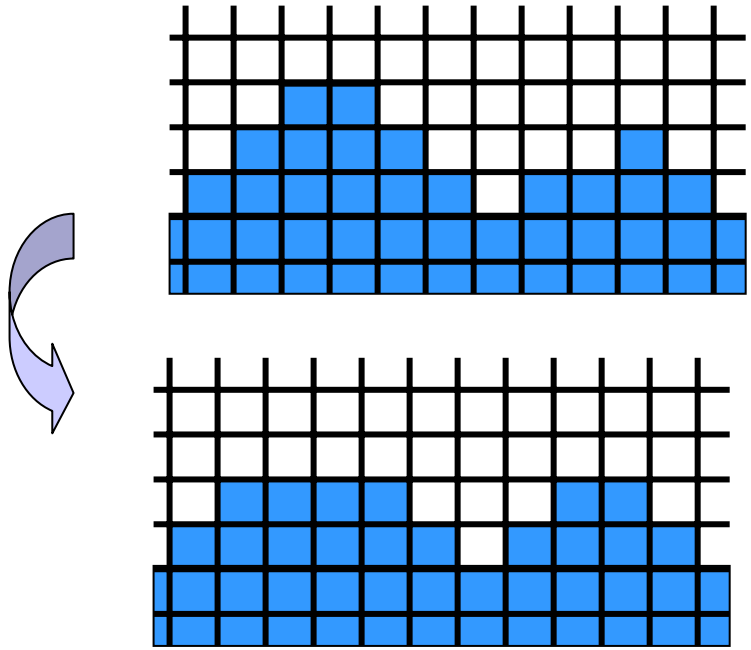
（風化）

- 凹凸端の“平滑化”をモデル化
- サイトの値＝0または1

更新値＝近傍9個の内の多い値

ただし，5個では　→　0

4個では　→　1



```
In[1]:= Print[“          The CA Vote Rule Table”];
        TableForm[{Range[0, 9],
          {0, 0, 0, 0, 1, 0, 1, 1, 1, 1}},
         TableHeadings ->
           {{“Some over neighborhood”, “New cell value”},
             None}]
```

CA Vote Rule Table

Sum over neighborhood	0	1	2	3	4	5	6	7	8	9
New cell value	0	0	0	0	1	0	1	1	1	1

・ プログラム

```
VoteNearCallsToLosers[s_, t_] :=
  Module[{rule, init, ngbrhdTotal},
   init = Table[Random[Integer], {s}, {s}];
   ngbrhdTotal[mat_] := Apply[Plus, Map[RotateRight[mat, #]&,
                          {{-1, -1}, {-1, 0}, {-1, 1}, {0, -1}, {0, 1},
                           {0, 0}, {1, -1}, {1, 0}, {1, 1}}]];
   rule[5] := 0;
   rule[x_] := Floor[x/4];
   Attributes[rule] = Listable;
   NestList[rule[ngbrhdTotal[#]]&, init, t]]
```