２．４　ウルフラム･セルラ･オートマトン（複雑系の代表例）

Stephen Wolfram：　Mathematicaの開発者

- １次元セルラ･オートマトン
- 格子サイトは確率論的，均一（一様）な局所的ルールで発展

　→　発展パターンは4タイプに落ち着く

- ２次元セルラ･オートマトンには時間発展に対して多くの興味あるパターン

（例）グライダー，プリンカ，蜂の巣などの安定型

⇩

時間を追ってパターンを比較して推移を見るのでなく一目瞭然とする工夫

⇩

１次元を時間軸　⇒　１次元セルラ･オートマトン

［1次元セルラ･オートマトンの定義］

(1)　N個のセル．周期境界条件

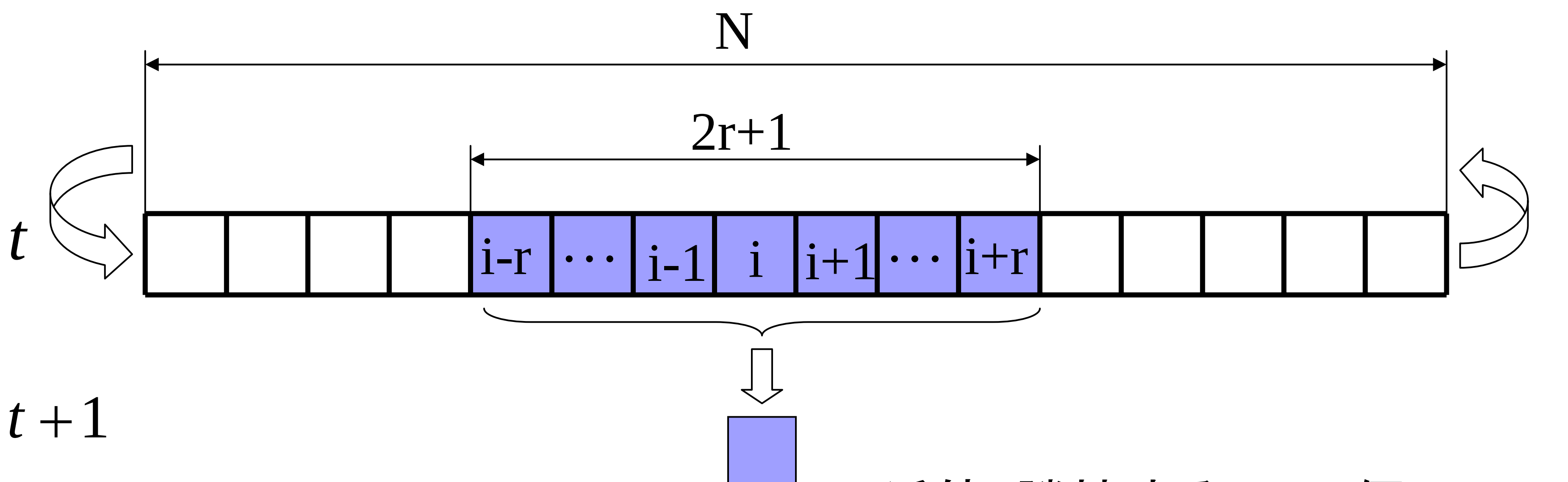
(2)　k個の状態

(3)　“ $x(i,t)$:セル i の時間 t の状態” とすると,

$x(i,t+1)$ はセル i とそのセルの左右に隣接するr個づつ,

合計 2r+1 個のセル状態の並び方により定まる.

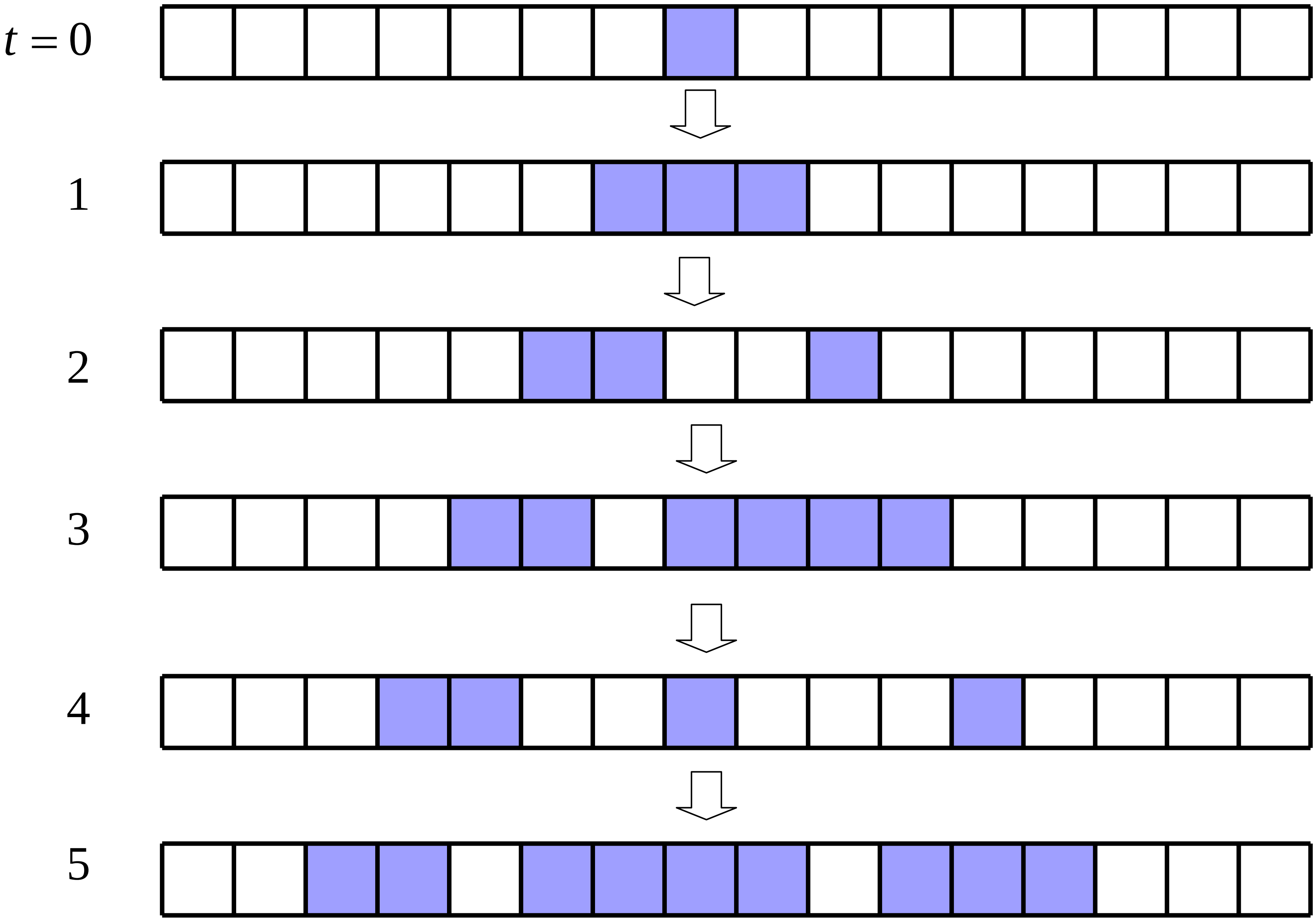
$$\{x(i-r,t),\cdots,x(i,t),\cdots,x(i+r,t)\}\quad\Rightarrow\quad x(i,t+1)$$



近傍: 隣接する 2r+1 個のセル

近傍の大きさ: 2r+1 ,　配列: 近傍中における状態の並び

推移の例



［推移規則］

(a) ２状態で近接サイトに無関係（k = 2, r = 0 (2r+1=1)）

rule 1: 全て0 {{1} -> 0, {0} -> 0}
rule 2: 反転 {{1} -> 0, {0} -> 1}
rule 3: 不変 {{1} -> 1, {0} -> 0}
rule 4: 全て1 {{1} -> 1, {0} -> 1}

４つのルール

(b) ２状態３近傍（k = 2, r = 1 (2r+1=3)）および一般の場合

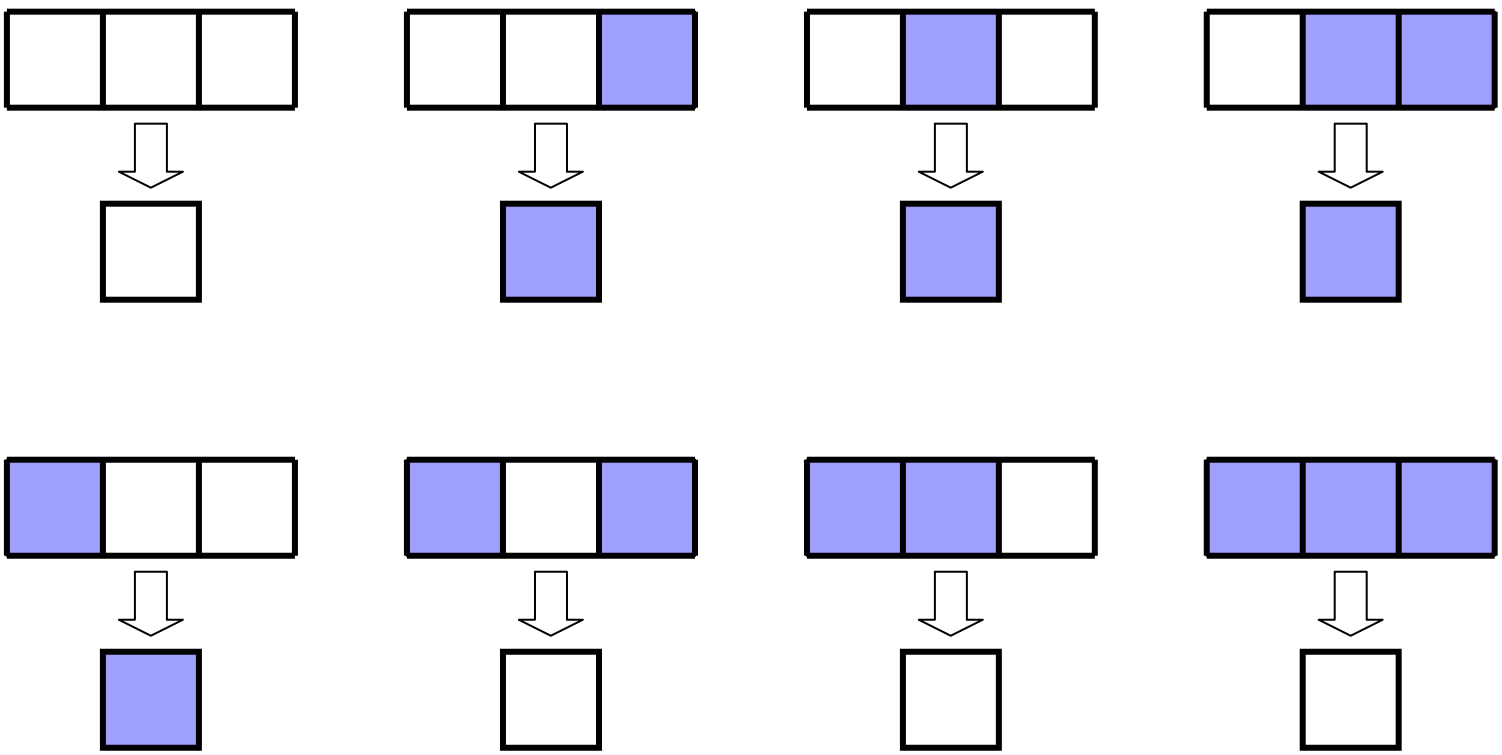
可能な近傍の値

{{1, 1, 1}, {1, 1, 0}, {1, 0, 1}, {1, 0, 0},
{0, 1, 1}, {0, 1, 0}, {0, 0, 1}, {0, 0, 0}}

の８（＝2^3）組

可能なルールの１つ

{{1, 1, 1} -> 1, {1, 1, 0} -> 1, {1, 0, 1} -> 1, {1, 0, 0} -> 0,
{0, 1, 1} -> 1, {0, 1, 0} -> 0, {0, 0, 1} -> 0, {0, 0, 0} -> 1}



１つの組に対し２つ　⇒　８組に対し2^8＝256個のルールセット

一般に，k 状態 (2r + 1) 近傍に対して　　$k^{k^{(2r+1)}}$

２状態５近傍(k = 2, r = 2)に対して，

$$2^{2^5} = 2^{32} = 4{,}294{,}967{,}296$$ (約43億）

３状態３近傍(k = 3, r = 1)に対して

7,625,597,484,987（約7兆600億）

状態（サイトがとりうる値）の数k

近傍（相互作用に含まれるサイト）の数(2r + 1)

増加

ルール・セットの数は劇的に増加

○　各ルールの番号付け(rule number)

各組に対して割当てた１，０を２進数表示し，それを10進数に直す.

２進数　11101001_2　⇒　10進数　233

この10進数を番号とする.

［Mathematica によるウルフラムCAプログラムの作成］

(1)組のリスト

```
In[7]:= tuplets[k_, r_] :=
            Reverse[Flatten[Apply[Outer,
                    Prepend[Table[Range[0, k – 1],
                    {(2r + 1)}], 2r]]
```

（例）

```
In[8]:= typlets[2, 1]
Out[8]= {{1, 1, 1}, {1, 1, 0}, {1, 0, 1}, {1, 0, 0},
         {0, 1, 1}, {0, 1, 0}, {0, 0, 1}, {0, 0, 0}}
```

(2) 更新値（update value）リスト

２状態３近傍，128番以上は

```
In[9]:= IntegerDigits[233, 2]
Out[9]= {1, 1, 1, 0, 1, 0, 0, 1}
```

でよい．

ところが，127番以下では

```
In[10]:= IntegerDigits[24, 2]
Out[10]= {1, 1, 0, 0, 0}
```

となり，組の数８と要素数５が合わない．リストの先頭に0, 0, 0を加えるため，

```
In[11]:= fillCode[k_, r_, w_] :=
            Join[Table[0, {k^(2r + 1) – Length[#]}], #]&
                [IntegerDigits[w, k]]

In[12]:= fillCode[2, 1, 24]
Out[12]= {0, 0, 0, 1, 1, 0, 0, 0}
```

○ k 状態 (2r + 1) 近傍，ルール番号 w のルール・セット

```
In[13]:= ruleSet[k_, r_, w_] := Module[{fillCode, tuplets},
            fillCode[x_, y_, z_] :=
              Join[Table[0, {x^(2y + 1) – Length[#]}], #]&
               [IntegerDigits[z, x]];
            tuplets[p_, s_] :=
               Reverse[Flatten[Apply[Outer,
                     Prepend[Table[Range[0, p – 1],
                     {(2s + 1)}]List]], 2s]];
            MapThread[Rule,
                     {tuplets[k, r], fillCode[k, r, w]}]
             ]

In[14]:= ruleSet[2, 1, 233]
Out[14]= {1, 1, 1} -> 1, {1, 1, 0} -> 1, {1, 0, 1} -> 1, {1, 0, 0} -> 0,
         {0, 1, 1} -> 1, {0, 1, 0} -> 0, {0, 0, 1} -> 0, {0, 0, 0} -> 1
```

［ウルフラムCAアルゴリズム］

- 記号

m ＝行あたりサイト数, n ＝世代数, w ＝ルール番号

k ＝状態数, 2r + 1 ＝近傍数（相互作用するサイト数）

(1) CA の開始

(a) 中央のサイトを除いてすべてがゼロ

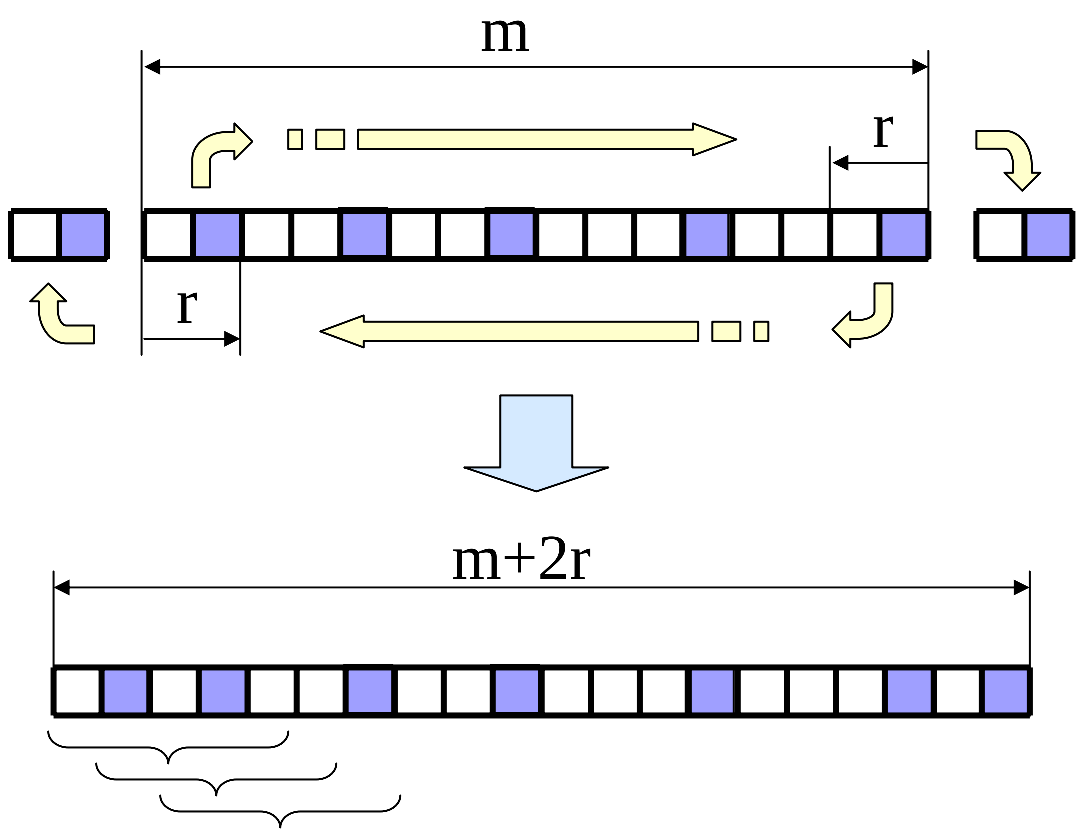
```
row = ReplacePart[Table[0, {m}],
                  Random[Integer, {0, k-1}]
                  Ceiling[m/2]]
```

(b) ランダム分布

```
row = Table[Random[Integer, {0, k-1}], {m}]
```

(2) 近傍の作成

(a) Join[Take[row, -r], row, Take[row, r]]



(b) Partition[Join[Take[row, -r], row, Take[row, r]], (2r + 1), 1]

(3) 更新サイトの値

```
Partition[Join[Take[row, -r], row, Take[row, r]], (2r + 1),
        1]/. ruleset
```

(4) シミュレーション終了

連続した行の値に変化がなくなる
　または　　最大の n 行に達する

```
CArows =
  FixedPointList[(Partition[Join[Take[#, -r],
                  #, Take[#, r]], (2r + 1), 1] /. ruleset) &,
                row, n]
```

ウルフラムCAプログラム

```
CA[w_, k_, r_, m_, n_] := Module[{ruleSet, row},
    ruleSet = MapThread[Rule, {Reverse[Flatten[Apply[Outer,
                    Prepend[Table[Range[0, k, -1], {(2r + 1)}], List]], 2r]],
                Join[Table[0, {k^(2r + 1) – Length[#]}], #]&
                    [IntegerDigits[w, k]]}];
    row = Table[Random[Integer, {0, k-1}], {m}];
    FixedPointList[(Partition[Join[Take[#, -r],
                        #, Take[#, r]], (2r + 1), 1] /. ruleset) &,
                    row, n]
```

関数WolframCA

［標準値を持つオプション］

CASize（初期配置の長さ）, CATimeSteps（時間ステップ数）,
InitialStage（初期状態）

```
In[1] := Options[WolframCA] = {CASize -> 100, CATimeSteps -> 20,
                               InitialState -> “Disordered”};

In[2]:= WolframCA[w_, k_, r_, opts_ _ _Rule] :=
        Module[{size, steps, state, row, ruleSet},
            size = CASize /. {opts} /. Options[WolframCA];
            steps = CATimeSteps /. {opts} /. Options[WolframCA];
            state = InitialState /. {opts} /. Options[WolframCA];
```

```
row = If [state = = = “SingleSeed”,
        ReplacePart[Table[0, {size}],
                k-1, Ceiling[size/2]],
     Table[Random[Integer, {0, k-1}], {size}]
ruleSet = MapThread[Rule, {Reverse[Flatten[Apply[Outer,
        Prepend[Table[Range[0, k, -1], {(2r + 1)}],
              List]], 2r]],
        Join[Table[0, {k^(2r + 1) – Length[#]}], #]&
           [IntegerDigits[w, k]]}];
FixedPointList[(Partition[Join[Take[#, -r],
          #, Take[#, r]], (2r + 1), 1] /. ruleSet) &,
          row, steps]
```

[ウルフラムCAプログラムの実行]

```
In[3]:= ShowCA[list_List, opts_ _ _] :=
        Module[{rule, max = Max[list], colorRule},
            colorRule = Flatten[Map[{# -> Hue[#/(max+1)]}&,
                                Range[0, max] ]];
            Show[Graphics[
                RasterArray[Reverse[list] /. colorRule],
                opts, AspectRatio -> Automatic]]
            ]
```

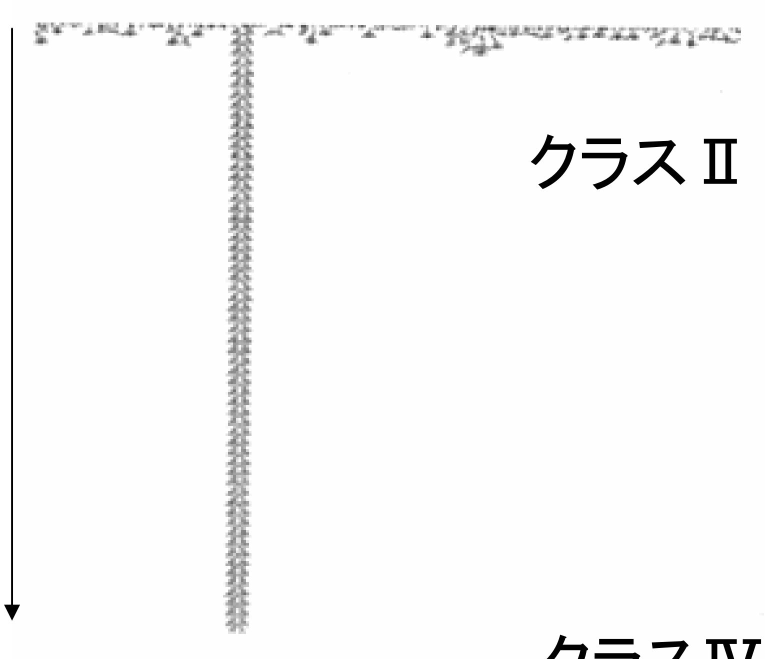
ウルフラムCAのクラス分類（図）

クラスⅠ

時間

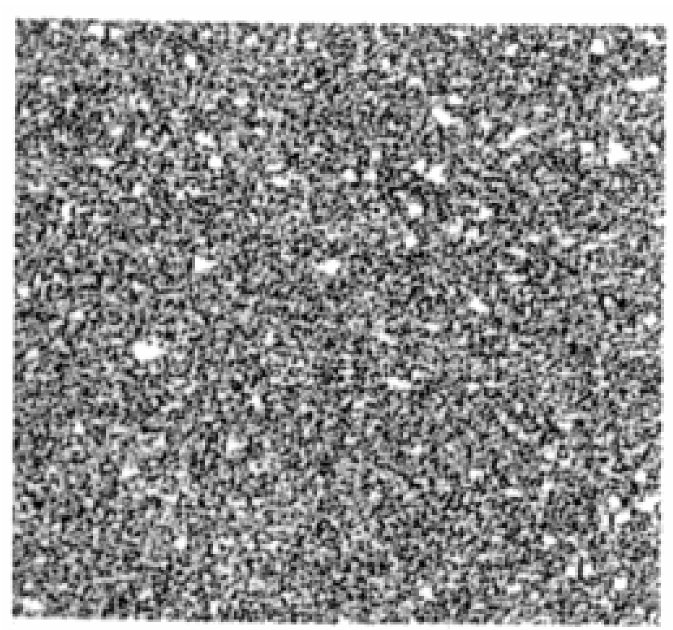
クラスⅡ

時間



クラスⅢ

時間



クラスⅣ

時間

ウルフラムCAのクラス分類

Ⅰ. 均一状態:　集積点(limit point)に似ている

真っ黒か真っ白

Ⅱ. 周期的構造:　極限軌道(limit cycle)に似ている.

ある安定した周期に落ち着く

Ⅲ. カオス的パターン:　奇妙なアトラクタ(strange attractor)に似ている.

黒白がいつまでもばらばらに現れるカオス

Ⅳ. 複雑な局所構造:　万能コンピュータ

以上のクラスに分類されない不思議な動き

クラスⅠ

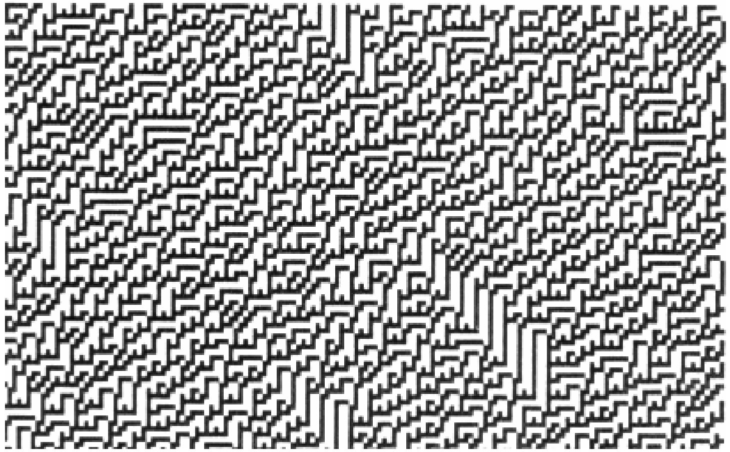
In[1]:=ShowCA[WolframCA[136, 2,1]]

クラスⅡ

In[2]:=ShowCA[WolframCA[4, 2,1]]

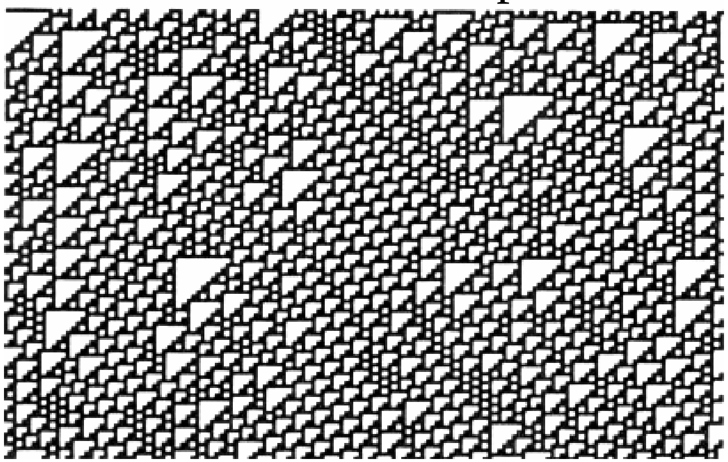
クラスⅢ

In[1]:=ShowCA[WolframCA[45, 2,1]
CASize ->150,
CATimeSteps ->80]]



クラスⅣ

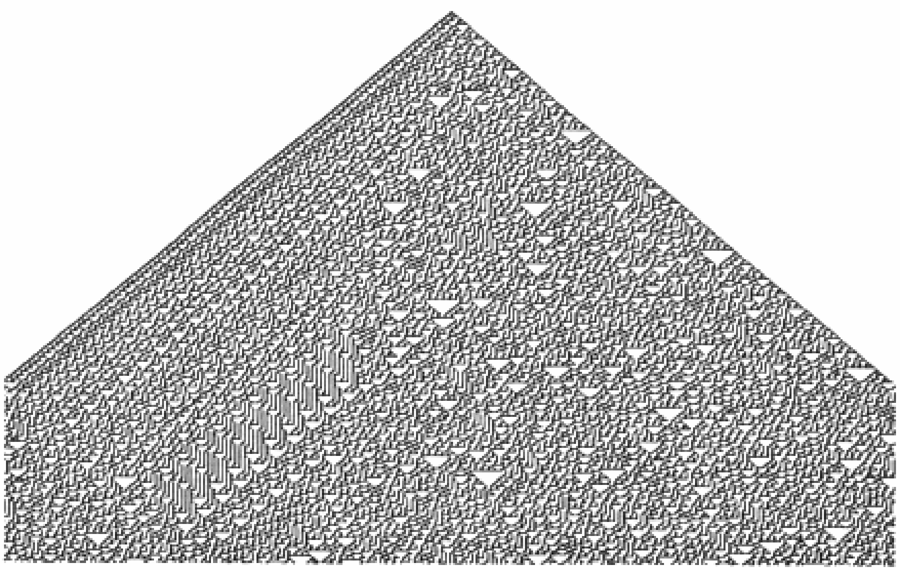
In[1]:=ShowCA[WolframCA[110, 2,1]
CASize ->150,
CATimeSteps ->60]]

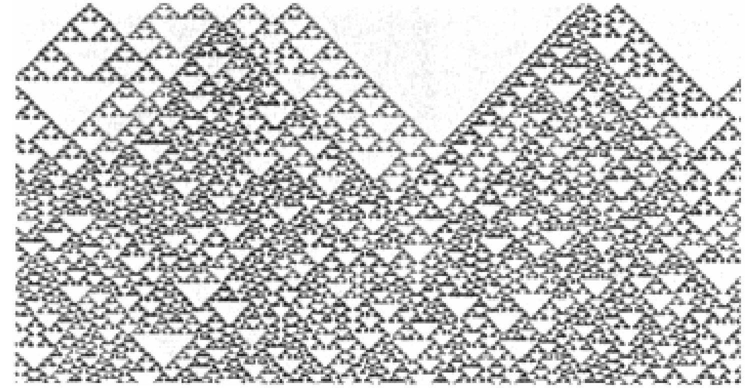


初期状態の相違

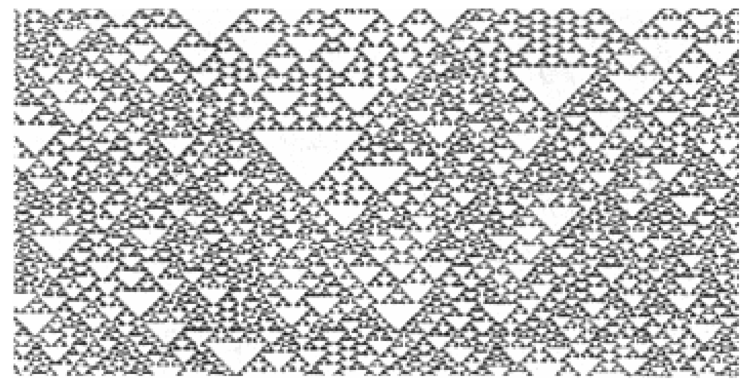
単一セルからのスタート

```
In[1]:=ShowCA[WolframCA[110, 2,1]
                    CASize ->150,
                    CATimeSteps   ->60]]
                    InitialState -> “SingleSeed”]]
```

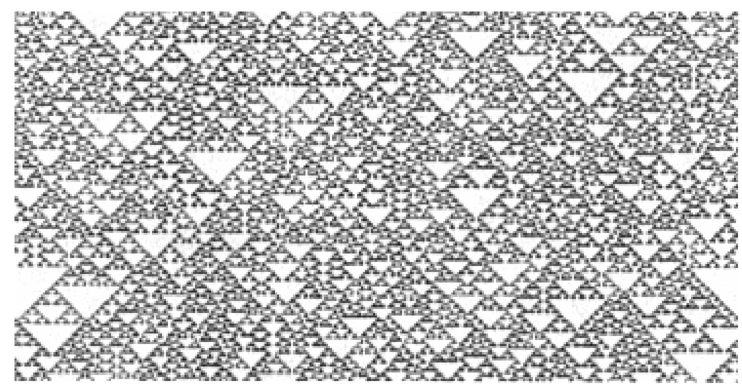




初期比率　0.1



0.5



0.9

状態推移のクラス

推移規則により定まり,
初期状態により変わることはない

２状態３近傍にはないパターン

２状態５近傍

３状態３近傍

配列型推移規則では, k, r の増加につれて
ルール・セットは劇的に増加

ライフ・ゲームのアナロジー

近傍における状態の総和が等しいものは同一とする

合計型推移規則

２状態３近傍では {0, 1, 2, 3} の４通り ⇒ ルールの数＝2^4＝16

一般に $k^{(k-1)(2r+1)+1}$

	合計型	配列型
２状態３近傍	16	256
２状態５近傍	64	4,294,967,296
３状態３近傍	128	7,625,597,484,987