— vordenker-archive —

Rudolf Kaehr
(1942-2016)

Title

Aspects of Complexity Reduction for morphoCAs – The Deutero Approach

Archive-Number / Categories

3_41 / K12, K09, K12

Publication Date

2014

Keywords / Topics

Morphograms, Celluilar Automata, Semiotics

Disciplines

Computer Science, Artificial Intelligence and Robotics, Logic and Foundations of Mathematics, Cybernetics, Theory of Science

Abstract

One important motivation for a introduction of deutero-structural based morphic cellular automata is to reduce computational complexity by saving its structural complexity.

Deutero-structures of the morphogrammatic system shall be introduced as such a further strategy of complexity reduction for morphogrammatic based cellular automata.

The focus of the previous papers on morphoCAs had been on the trito-structure of morphogrammatics. This paper turns its focus on the deutero-structure of morphogrammatics with the aim to introduce a further strategy of complexity reduction for morphoCAs.

Aspects of Complexity Reduction for morphoCAs - The deutero approach

Dr. phil Rudolf Kaehr

ISSN 2041-4358
(work in progress, vs. 0.1, Jan. 2015)

Reduction of the complexity of morphoCAs by deutero-abstraction

One important motivation for a introduction of deutero-structural based morphic cellular automata is to reduce computational complexity by saving its structural complexity.

Deutero-structures of the morphogrammatic system shall be introduced as such a further strategy of complexity reduction for morphogrammatic based cellular automata.

The focus of the previous papers on morphoCAs had been on the *trito-structure* of morphogrammatics. This paper turns its focus on the *deutero-structure* of morphogrammatics with the aim to introduce a further strategy of complexity reduction for morphoCAs.

> ABSTRACTION FROM THE ORDER AND IDENTITY OF ATOMS: HEAPS OF KENOMS
> *"In his development of the theory of kenograms, Gotthard Gunther introduced three layers of abstraction, and called them* "Trito Structure", "Deutero Structure" *and* "Proto Structure". *The Trito Structure coincides with what we have called "strings of kenoms". The Deutero Structure was derived from Trito Structure by abstracting from the* order *in which the kenoms occur. The Proto Structure was derived from Deutero Structure by excluding patterns in which more than one atom occur repeatedly.*
>
> *- heaps of atoms are stucturally very* similar *to multisets."* (R. Matzka)
>
> http://www.rudolf-matzka.de/dharma/semabs.pdf
>
> "*Deutero-Structure* results from the assumption that maximal repetition is allowed for individual kenograms. As for the rest, the placing of the symbol still remains irrelevant" (Gunther 1980, p. 111).

Deutero-sets

If we abstract in this model of tritosets from the *order* of the occurrences of the elements (kenograms), we get a new class or type of languages, the languages based on deutero-sets.

Hence, the deutero-sets (aab), (aba), (bba) and (bab), are deutero-equal, equally [aaa] and [bbb], while (aaa) and (aab) are not deutero-equal. Deutero-sets are measured by the sum of partitions: P(n, m). Therefore, [aaa] and [bbb] are d-equal because they have the same number of partitions, i.e. one partition of itself.

In contrast, albeit multisets {a,a,a} and {b,b,b} have the same partitions, they differ in their elements, "a" and "b", a ≠ b.

That defines the difference between deuteroCAs and *indicational* CAs, indCAs. Indicational CAs are based mathematically on multisets. The famous Calculus of Indication is based on multsets with just 2 elements, Mark and Nil.

Hence, deutero-sets are not just abstracting from the order (position) of identitive elements of a set, but from the order of kenograms in trito-sets. Kenograms are not identitive elements in contrast to the elements (atoms) of sets and multisets as they are defined in set theory.

Multisets are well presented by the paper:
http://www.emis.de/journals/NSJOM/Papers/37_2/NSJOM_37_2_073_092.pdf

System of abstractions

types	set	mset	tset	dset	pset	list
locus	–	–	+	–	–	+
multiplicity	–	+	+	+	+	+
occurrence	+	+	–	–	–	+
order	–	–	+	+	–	+

Abstraction from msets to deutero - psets and to deutero - sets:

mset	1	2	3	4	num {–}
1	*a*	*a*	*a*	*a*	1^4
2	*b*	*b*	*b*	*b*	2^4
3	*a*	*a*	*a*	*b*	$1^3 2^1$
4	*a*	*a*	*b*	*b*	$1^2 2^2$
5	*a*	*b*	*b*	*b*	$1^1 2^3$

⟹

dpset	□	□	□	□	num {–}
1	*a*	*a*	*a*	*a*	1^4
2	*b*	*b*	*b*	*b*	2^4
3	*a*	*a*	*a*	*b*	$1^3 2^1$
4	*a*	*a*	*b*	*b*	$1^2 2^2$

⟹

dset	□	□	□	□	num {–}
1	*a*	*a*	*a*	*a*	1^4
2	*a*	*a*	*a*	*b*	$1^3 2^1$
3	*a*	*a*	*b*	*b*	$1^2 2^2$

$$\text{mset} = \binom{n+m-1}{n}$$
ident – commutative

$$\text{dpset} = \sum_{k=1}^{M} P(m,\ n) \binom{m}{n}$$
deutero – partitive

$$\text{dset} = \sum_{k=1}^{M} P(n,\ k)$$
deutero

Reduction from sets to tritograms

sets = $m^n : 2^4 = 16$ ⟹

tset	1	2	3	4	num (–)
1	*a*	*a*	*a*	*a*	1^4
2	*a*	*a*	*a*	*b*	$1^3 2^1$
3	*a*	*a*	*b*	*a*	$1^2 2^2$
4	*a*	*b*	*a*	*a*	$1^1 2^1 1^2$
5	*a*	*a*	*b*	*b*	$1^2 2^2$
6	*a*	*b*	*a*	*b*	$1^1 2^1 1^1 2^1$
7	*a*	*b*	*b*	*a*	$1^1 2^2 1^1$
8	*a*	*b*	*b*	*b*	$1^1 2^3$

$$\text{tset} = \sum_{k=1}^{M} S(n,\ k)$$
trito

Hence, deutero-sets are in a strict sense not sets and also not multi-sets but kenomic aggregations of kenograms that are abstracting from the order of the (not-identive) kenograms of a trito-structure. Multisets are abstracting from the order of their elements too, but the elements of a multiset are identive, while the kenograms of a deuterogram are not identive.

The computational domain of multisets had been studied as indictional cellular automata, *indCA*, in conection to the Calculus of Indication of George Spencer Brown.

http://memristors.memristics.com/IndCA/Indicational%20CA.html

Concept graph

tritogram
↙ ↘
multiset → deuterogram

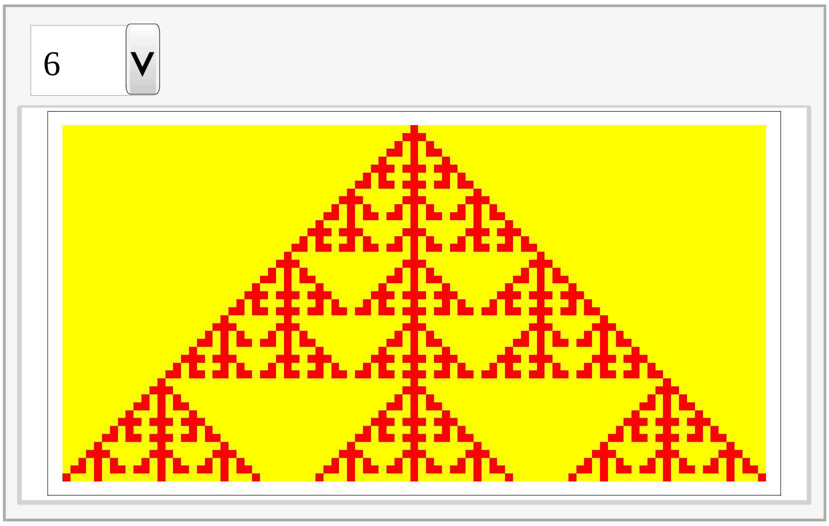
multiset:

mset	1	2	3	4	num{–}
1	*a*	*a*	*a*	*a*	1^4
2	*b*	*b*	*b*	*b*	2^4

⟹ **deuterogram**:

dset	□	□	□	□	num{–}
1	*a*	*a*	*a*	*a*	1^4

Multiset system of indCA$^{(3,2)}$

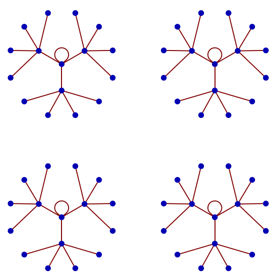


Equivalence of ruleCI[{1, 6, 3, 8}] and ruleMD3[{1, 4, 2}]

(Debug) In[27]:=
```
ArrayPlot[CellularAutomaton[
   ruleCI[{1, 6, 3, 8}],
   {{1}, 0}, 44],
  ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

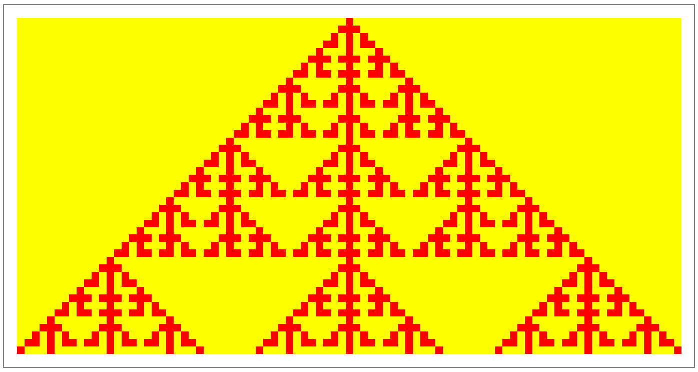
(Debug) In[28]:=
```
GraphPlot[# -> CellularAutomaton[ruleCI[{1, 6, 3, 8}], #] & /@ Tuples[{0, 1}, 6]]
```

(Debug) Out[28]=



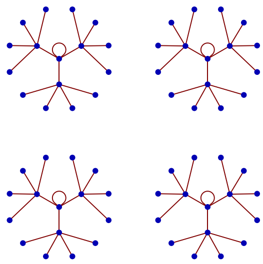
(Debug) In[23]:=
```
ArrayPlot[CellularAutomaton[
  ruleMD3[{1, 4, 2}],
  {{1}, 0}, 44],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[23]=



(Debug) In[30]:=
```
GraphPlot[# -> CellularAutomaton[ruleMD3[{1, 4, 2}], #] & /@ Tuples[{0, 1}, 6]]
```

(Debug) Out[30]=



Systematic locus of deuteroCAs

The emphasis is on the comparison of morphic, indcational and deutero CAs. Indicational CAs, indCAs, had been studied in a previous paper. They are based on multisets. As a contrast, deuteroCAs are based on partitions, while morphoCAs are framed by the Stirling numbers of the second kind. Multisets are, like sets, defined by the identity of their elements. Trito- and deutero structures are structured by non-identive signs, i.e. kenograms.

It follows that the deutero rules are defined by the *trito-abstraction* for the morphograms and by *permutations* defining the deutero-structure of morphograms.

Indicational graph

$$\{1^1\}$$

$$\swarrow \quad \downarrow \quad \searrow$$

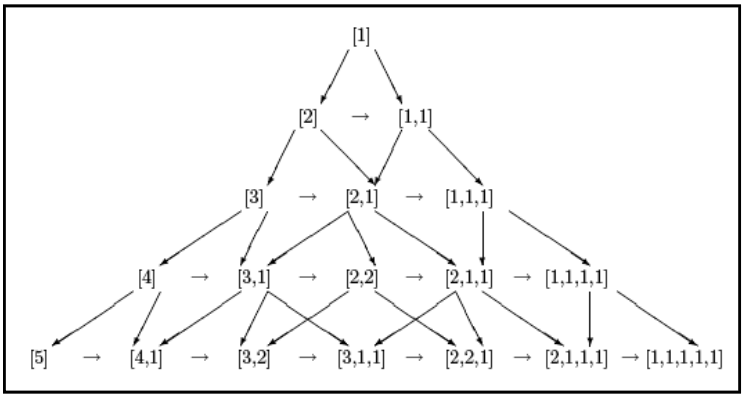
$$\{1^2\}, \{1^1 2^1\}, \{2^2\}$$

$$\swarrow \searrow \swarrow \quad \searrow \quad \swarrow \searrow$$

$$\{1^3\}, \{1^2 2^1\}, \{1^1 2^2\}, \{2^3\}$$

$$\swarrow \searrow \quad \swarrow \searrow \swarrow \searrow \quad \swarrow \searrow$$

$$\{1^4\}, \{1^3 2^1\}, \{1^2 2^2\}, \{1^1 2^3\}, \{2^4\}$$



Hence, for deutero-arithmetics the items $\{1^2\, 2^1\}$ and $\{1^1\, 2^2\}$ are d- equivalent and are represented as [2,1] . This holds for $\{1^2\}$, $\{1^1\, 2^1\}$, $\{2^2\}$ too. The items $\{1^2\}$ and $\{2^2\}$, are deutero-equivalent and represented by $[1^1\, 2^1]$ = [1,1].

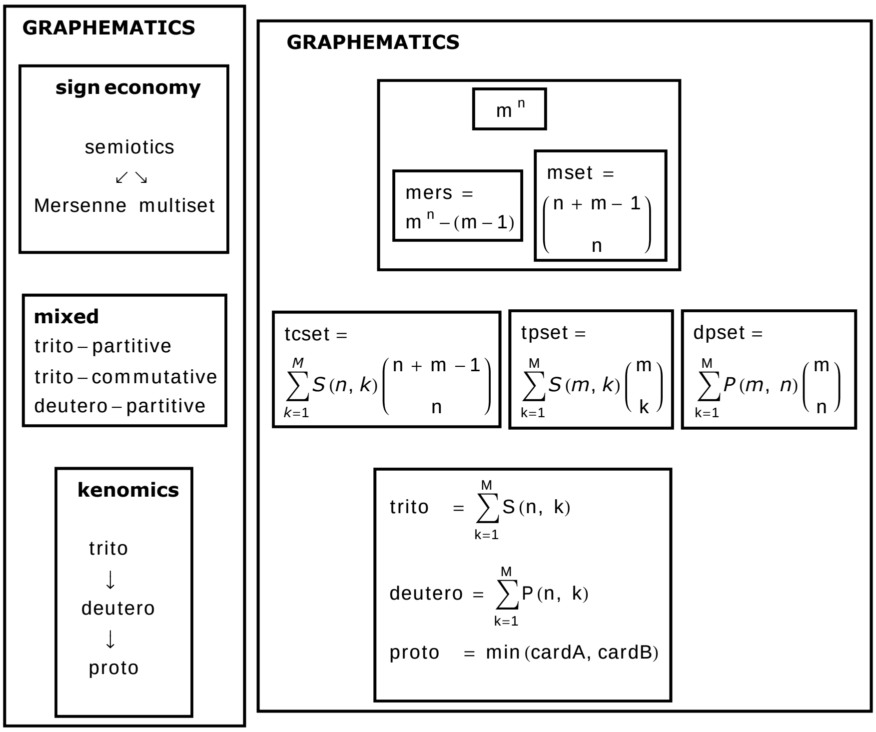
A summary of the systematic situation is given by the following table:

types	order	number	elements Σ	identity V	Ex. = $[a, a, b, c, a]$	combinatorics
posets	□	□	□	□	□	□
pomsets	+	+	−	+	□	□
multisets	−	+	+	+	$[a, b, c]_{3,1,1}$	$\binom{n+m-1}{n}$
sets	−	−	+	+	$[a, a, b, c, a]_{1,1,1}$	m^n
tsets	+	+	−	−	$(1^2\, 2^1\, 3^1\, 1^1)$	$\sum_{k=1}^{m} \text{Sn2}(m, k)$
dsets	−	+	−	−	$\{1^3\, 2^1\, 3^1\}$	$\sum_{k=1}^{M} P(n, k)$
psets	−	+	+	−	$[5:3]$	$\min\{m, n\}$
lists	+	+	+	+	$[a, a, b, c, a]_{1,1,2,3,1}$	m^n

Other methods of complexity reduction for morphoCAs had been presented at:

http://memristors.memristics.com/Decompositions/Decomposition.pdf , (html, cdf)

The deutero-approach to writing systems is well placed in the system of *graphematics*.



http://memristors.memristics.com/Graphematics/Graphematics%20of%20Cellular%20Automata.pdf

http://memristors.memristics.com/Graphematics%20of%20Multisets/Graphematics%20of%20Multisets.pdf

As a consequence of combinatorial sketch to deuterograms it follows that deuteroCAs are placed at a genuine place between multi-sets and tritograms, and are therefore exploring a new domain of an algorithmic poly-verse.

Again,in contrast to Stirling patterns, i.e. morphograms of the *trito*-structure, that are being studied by the new type of CAs, the morphoCAs, the *deutero*-structure is abstracting additionally from the abstraction of the *identity* of the signs also from the *positions* of the elements involved in the morphic patterns. Therefore they are mathematically characterized not by the Stirling numbers of the second kind but by the concept of *integer partitions* (Pascal) and defining its algorithmic domain by deuteroCAs.

Partitions of classical elementary CAs are studied by:
http://www.mathpages.com/home/kmath416/kmath416.htm

Reduction steps

The steps of numerical reduction as part of a reduction of morphoCAs is given by the chain:

$$\boxed{\textbf{symbolic} = m^{n} \Longrightarrow \textbf{trito} = \sum_{k=1}^{M} S(n,\ k) \Longrightarrow \textbf{deutero} = \sum_{k=1}^{M} P(n,\ k)}$$

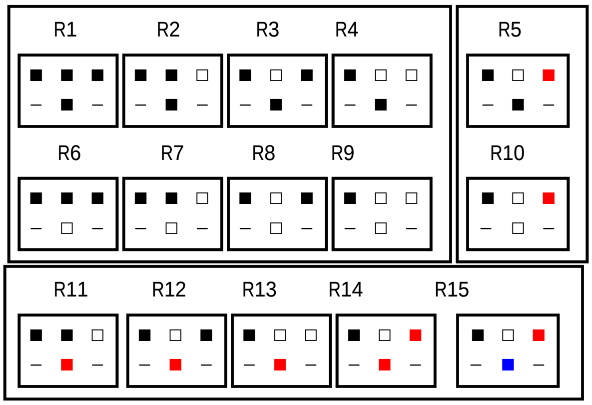
Deutero – arithmetics, Morphogrammatik, 1993, *p*. 66 – 68

```
Trito - contextures
Tcontexture 4;

val it =
 [[1, 1, 1, 1],
   [1, 1, 1, 2], [1, 1, 2, 1], [1, 2, 1, 1], [1, 2, 2, 2],
   [1, 1, 2, 2], [1, 2, 1, 2], [1, 2, 2, 1],
   [1, 1, 2, 3], [1, 2, 1, 3],
   [1, 2, 3, 1], [1, 2, 2, 3], [1, 2, 3, 2], [1, 2, 3, 3],
   [1, 2, 3, 4]] : int list list
```

```
trito-rules

R1,
R6, R2, R3, R9,
R7, R8, R4,
R11, R12, R5, R13, R10, R14,
R15
```

$$\text{rules-morphoCA}^{(4,4)} = \sum_{k=1}^{4} \text{Sn2}(4, k) = 1+6+7+1 = 15$$

System of elementary morphic cellular automata rules



System of elementary deuteroRules

Deutero-contextures

```
Dcontexture 4;
val it =
  [[1, 1, 1, 1],
   [1, 1, 1, 2],
   [1, 1, 2, 2],
   [1, 1, 2, 3],
   [1, 2, 3, 4]] : int list list
```

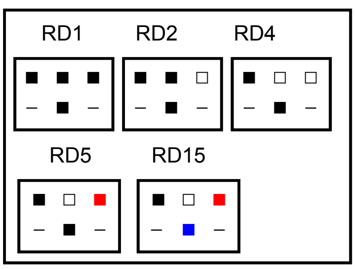
The cardinality of Dcontexture is measured by $\sum_{k=1}^{M} P(n, k)$.

The elements of Dcontexture 4 are defining the rules of the automata deuteroCA$^{(4,4)}$.

DeuteroCA$^{(4,4)}$ rules

```
RD1 = [1, 1, 1, 1], R1
RD7 = [1, 1, 2, 2], R7⋃R8⋃R4
RD6 = [1, 1, 1, 2], R6⋃R2⋃R3⋃R9
RD5 = [1, 1, 2, 3], R5⋃R10⋃R11⋃R12⋃R13⋃R14
RD15 = [1, 2, 3, 4], R15
```

System of the deuteroCA$^{(4,4)}$ rules



Deutero - Arithmetics

Deutero – Number

A deutero – number of cardinality D is a partition part(D) of D with D ∈ N :

part(D) = [p_1, p_2, …, p_{max}]

Deutero – Equality

Two deutero – numbers D and E are deutero – equal if their partitionas are equal :

D = E ⟺ part(D) = part(E)

Rule

[1] ∈ Deutero ⇒ $n_{DTS}(D)$ ∈ Deutero.

Example for $n_{DTS}(D)$

D	$n_{DTS}(D)$	$DTS_1, \ldots, DTS_{nDTS(D)}$
[3, 1]	3	[[4, 1], [3, 1], [3, 1, 1]]
[3, 2, 1]	4	[[4, 2, 1], [3, 3, 1], [3, 2, 2], [3, 2, 1, 1]]

http : // memristors.memristics.com/Interplay/Interplay %20 of %20 Elementary %20 Graphematic %20 Calculi.pdf

http : // www.vordenker.de/ggphilosophy/gg_natural - numbers.pdf

Numeric Deutero – number rules

R0 : ⇒ [1]
R1 .1 : [n] ⇒ [n+1] | [n, 1]
R1 .2 : [1, 1] ⇒ [n+1, 1] | [1, 1, 1]
R1 .3 : [n, 1] ⇒ [n+1, 1] | [n, 2] | [n, 1, 1]

DeuteroEquivalence for deuteroRules =

{
RD1 := R1,
RD2 := R2 $=_D$ R6 $=_D$ R3 $=_D$ R9,
RD4 := R4 $=_D$ R8 $=_D$ R7,
RD5 := R5 $=_D$ R10 $=_D$ R11 $=_D$ R12 $=_D$ R13vR14,
RD15
}

ruleSetDeutero =

{
RD1 = R1,
RD2 = R2 ⋃ R6 ⋃ R3 ⋃ R9,
RD4 = R4 ⋃ R8 ⋃ R7,
RD5 = R5 ⋃ R10 ⋃ R11 ⋃ R12 ⋃ R13 ⋃ R14,
RD15
}

Disjunctivity

RD1 ⋂ RD2 ⋂ RD4 ⋂ RD5 ⋂ RD15 = ∅

Reduction of the trito-rule set

```
trito - rules

R1,
R2, R6, R3, R9,
R4, R7, R8,
R5, R12, R11, R13, R10, R14,
R15
```

$\xrightarrow{\text{d-reduction}}$

```
deutero - rules

RD1,
RD2,
RD4,
RD5,
RD15
```

Representations for deutero - sets

$$\text{card}\left[\mu\right]_{\text{deutero}} = \frac{n!}{(1!)^{e_1}(2!)^{e_2}\ldots(n!)^{e_n}}\binom{m}{k}\frac{k!}{e_1!e_2!\ldots e_n!}$$

Example 1. How many permutations are there of the mset [abccbccbddb]?
Solution. We want to find the number of permutations of the multiset
[A] = [a, b, c, d] 11243442 = {1·a, 4·b, 4·c, 2·d}.

Thus, n = 11, n1 = 1, n2 = 4, n3 = 4, n4 = 2. Then number of permutations is given by

$$\frac{n!}{n_1!n_2!\ldots n_k!} = \frac{11!}{1!4!4!2!} = 330.$$

Thus, the mset[A] = [a, b, c, d] 11243442 has 330 identitive representations. The notation [abccbccbddb] for [A] is therefore a conventional choice and put into mset - normal form notation.

Non-commutativity

A consequence of the loss of the morphic pattern quality, the order of the components of the composed deutero-rules is not anymore commutative in general.

```
Non - commutative constellations

ruleMD[{1, 2}] ≠ rule[{2, 1}],
ruleMD[{4, 2}] ≠ rule[{2, 4}],
ruleMD[{5, 2}] ≠ rule[{2, 5}],

perm[{1, 2, 5}]     ∉ Commutativity,
perm[{1, 2, 5, 15}] ∉ Commutativity,

[{1, 4, 5}]   = [{4, 1, 5}]   ≠ [{5, 1, 4}],
[{15, 1, 2}]  = [{1, 2, 15}]  ≠ [{2, 1, 15}]
[{1, 2, 4, 5, 15}] ≠ [{15, 5, 4 , 2, 1}]
```

Commutativity for ruleM

The rule set for the domain *ruleM* are based on the ordered morphograms of the ‘system of elementary morphic cellular automata rules’. The criterion of the composed rules is not commutativity but the acceptance of the order of the morphic components.

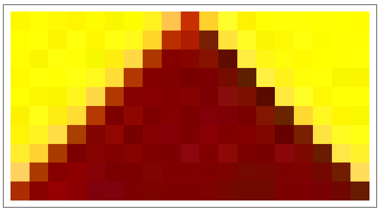
Therefore, a constellation like ruleM[{1,2,7,3,8}] is not accepting the order of the morphogrammatic system and results in a undefined situation.

Formally:

$$\text{ruleM}[\{a, b, c, d\}] \in \text{morphoCA} = \text{ruleM}[\text{perm}\{a, b, c, d\}] \in \text{morphoCA}$$

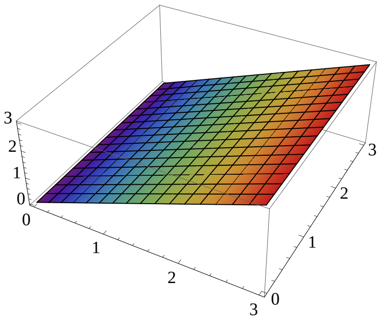
(Debug) In[222]:=
```
ArrayPlot[CellularAutomaton[
  ruleM[{1, 2, 7, 3, 8}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[222]=



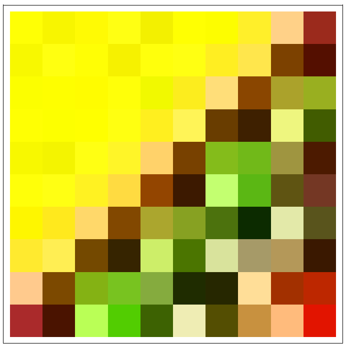
(Debug) In[237]:=
```
ListPlot3D[ruleM[{1, 2, 7, 3, 8}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[237]=



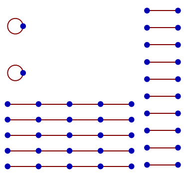
(Debug) In[217]:=
```
ArrayPlot[CellularAutomaton[
  ruleM[{1, 2, 12, 9, 15}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[217]=



(Debug) In[220]:=
```
GraphPlot[
 # -> CellularAutomaton[ruleM[{1, 2, 12, 9, 15}], #] & /@ Tuples[{0, 1}, 5]]
```

(Debug) Out[220]=

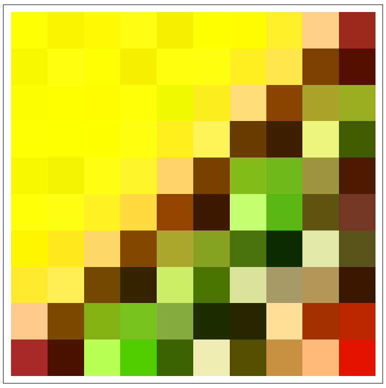


(Debug) In[233]:=
```
Table[Length[NestWhileList[CellularAutomaton[ruleM[{1, 2, 12, 9, 15}]],
   SparseArray[1 -> 1, n], Unequal, All]], {n, 22}]
```

(Debug) Out[233]= $Aborted

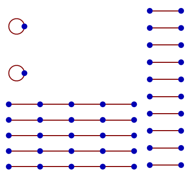
(Debug) In[218]:=
```
ArrayPlot[CellularAutomaton[
  ruleM[{15, 12, 2, 9, 1}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[218]=



(Debug) In[221]:=
```
GraphPlot[
 # -> CellularAutomaton[ruleM[{15, 12, 2, 9, 1}], #] & /@ Tuples[{0, 1}, 5]]
```

(Debug) Out[221]=

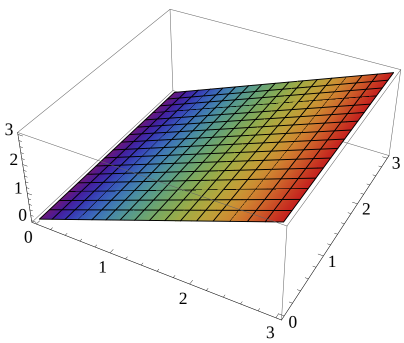


(Debug) In[231]:=
```
 ArrayPlot[Map[Flatten,{
ruleM[{1,2,12,9,15}]
} /. Rule -> List,1],
ColorRules->{1->Red,0->Yellow,
 2→Blue, 3→Green},
ImageSize -> Small, Mesh -> True]
```

(Debug) Out[231]=

(Debug) In[235]:=
```
ListPlot3D[ruleM[{1, 2, 12, 9, 15}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[235]=



(Debug) In[232]:=

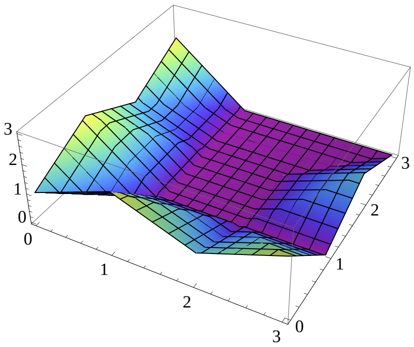
```
 ArrayPlot[Map[Flatten,{
ruleM[{15,12,2,9,1}]
} /. Rule -> List,1],
ColorRules->{1->Red,0->Yellow,
 2→Blue, 3→Green},
ImageSize -> Small, Mesh -> True]
```

(Debug) Out[232]=

(Debug) In[234]:=

```
ListPlot3D[ruleM[{15, 12, 2, 9, 1}], ColorFunction → "Rainbow", Mesh → True]
```

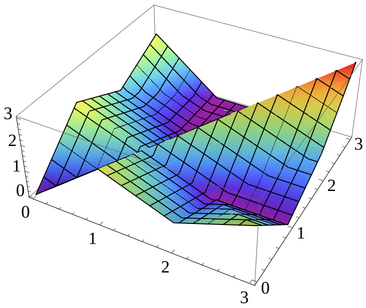
(Debug) Out[234]=



(Debug) In[236]:=

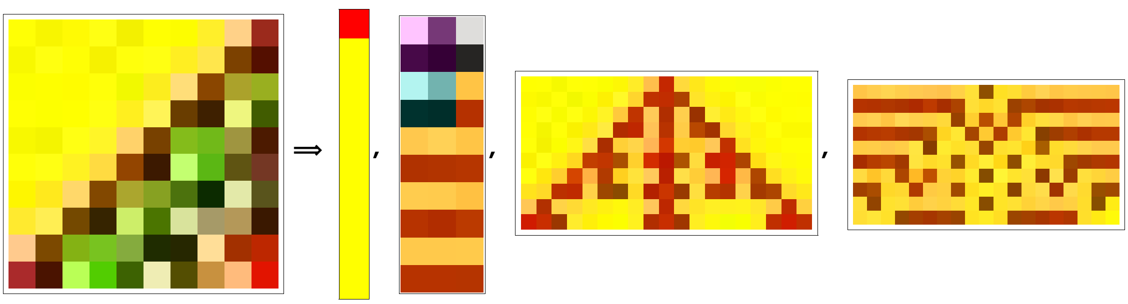
```
ListPlot3D[ruleM[{15, 1, 9, 2, 12}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[236]=



Representations

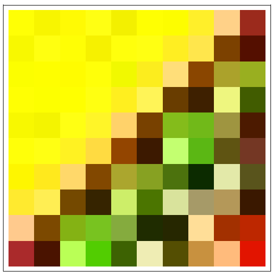
ruleM[{1, 2, 12, 9, 15}] ⟹ permutations of ruleMD4[{1, 2, 4, 5}]



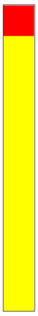
Further reductions

(Debug) In[259]:=
```
ArrayPlot[CellularAutomaton[
  ruleM[{1, 2, 12, 9, 15}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[259]=

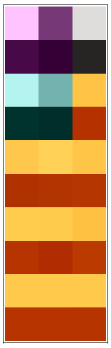


(Debug) In[261]:=
```
ArrayPlot[CellularAutomaton[
  ruleMD4[{1, 2, 4, 5}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```



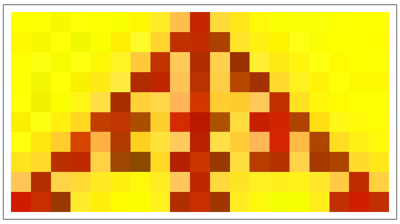
```
ruleMD4[{1, 2, 4, 5}], ruleMD3[{1, 2, 4}]
```

(Debug) In[263]:=
```
ArrayPlot[CellularAutomaton[
  ruleMD4[{5, 2, 4, 1}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```



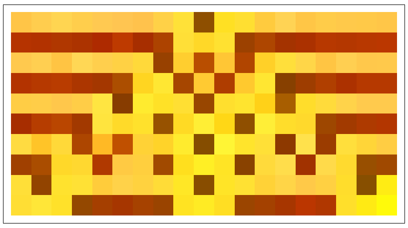
```
ruleMD4[{5, 2, 4, 1}], ruleMD3[{5, 2, 1}]
```

(Debug) In[265]:=
```
ArrayPlot[CellularAutomaton[
  ruleMD4[{1, 4, 2, 5}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```



ruleMD3[{4, 1, 2}], ruleMD3[{1, 4}]

(Debug) In[266]:=
```
ArrayPlot[CellularAutomaton[
  ruleMD4[{2, 1, 4, 5}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

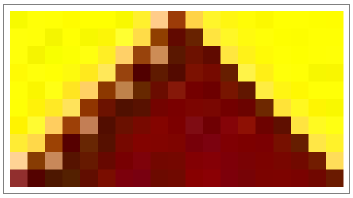


ruleMD3[{2, 4, 1}], ruleMD1[{2}]

Semi - defined representations in domain *ruleCl*

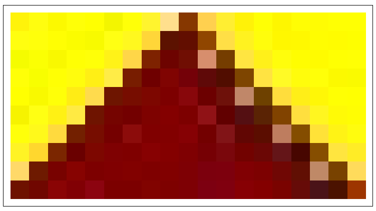
(Debug) In[262]:=
```
ArrayPlot[CellularAutomaton[
  ruleCl[{1, 2, 12, 15}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[262]=



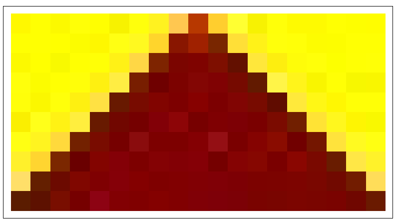
(Debug) In[282]:=
```
ArrayPlot[CellularAutomaton[
  ruleCl[{1, 4, 12, 15}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[282]=



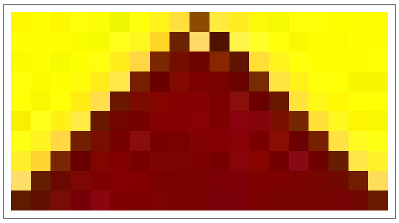
(Debug) In[281]:=
```
ArrayPlot[CellularAutomaton[
  ruleCl[{1, 3, 12, 15}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[281]=



(Debug) In[283]:=
```
ArrayPlot[CellularAutomaton[
  ruleCl[{1, 8, 12, 15}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[283]=



System of deuteroCA$^{(4,4)}$

Definition of the elementary deutero-rule set deuteroCA$^{(4,4)}$

Domain ruleMD1

ruleMD1 = {[R1]#, [R2], [R4]#, [R5]#, [R15]#}

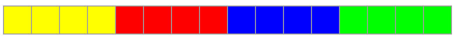
There is just one deuteroCA with a rule length of 1 that is accepted by the *CellularAutomaton* of the deuteroCA construction. It is the deutero-rule ruleMD[{2}].

The other constellations are not accepted by the *CellularAutomaton* of the deuteroCA construction.

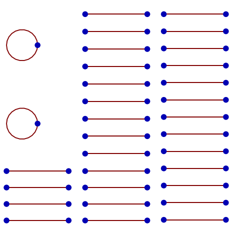
```
ArrayPlot[CellularAutomaton[ruleMD1[{1}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```

$Aborted

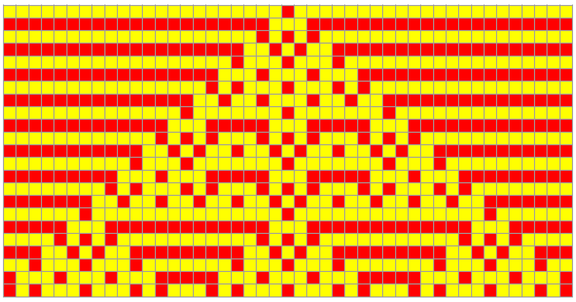
```
ArrayPlot[Map[Flatten,{
ruleMD1[{1}]
} /. Rule -> List,1],
ColorRules->{1->Red,0->Yellow,
 2→Blue, 3→Green},
ImageSize -> Small, Mesh -> True]
```



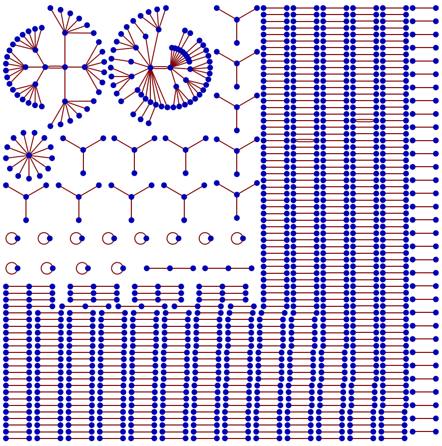
```
GraphPlot[# -> CellularAutomaton[ruleMD1[{1}], #] & /@ Tuples[{0, 1}, 5]]
```



```
ArrayPlot[CellularAutomaton[ruleMD1[{2}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



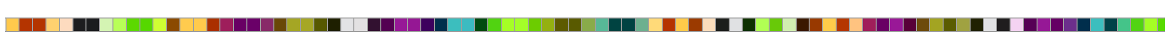
```
GraphPlot[# -> CellularAutomaton[ruleMD1[{2}], #] & /@ Tuples[{0, 1, 2}, 6]]
```



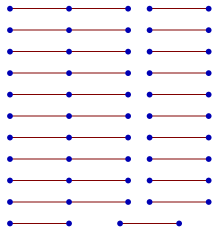
```
ArrayPlot[CellularAutomaton[ruleMD1[{4}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {600, 400}]
```

$Aborted

```
ArrayPlot[Map[Flatten,{
ruleMD1[{4}]
} /. Rule -> List,1],
ColorRules->{1->Red,0->Yellow,
 2→Blue, 3→Green},
ImageSize -> Small, Mesh -> True]
```



```
GraphPlot[# -> CellularAutomaton[ruleMD1[{4}], #] & /@ Tuples[{0, 1}, 5]]
```



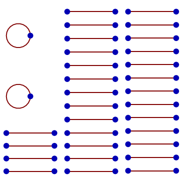
```
ArrayPlot[CellularAutomaton[ruleMD1[{5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {600, 400}]
```

```
$Aborted
```

```
ArrayPlot[Map[Flatten,{
ruleMD1[{5}]
} /. Rule -> List,1],
ColorRules->{1->Red,0->Yellow,
 2→Blue, 3→Green},
ImageSize -> Small, Mesh -> True]
```



```
GraphPlot[# -> CellularAutomaton[ruleMD1[{1}], #] & /@ Tuples[{0, 1}, 5]]
```



```
ArrayPlot[CellularAutomaton[ruleMD1[{15}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {600, 400}]
```

```
$Aborted
```

```
ArrayPlot[Map[Flatten,{
ruleMD1[{15}]
} /. Rule -> List,1],
ColorRules->{1->Red,0->Yellow,
 2→Blue, 3→Green},
ImageSize -> Small, Mesh -> True]
```

Domain ruleMD2

ruleMD2 = {{1,2}, {1,4}, {1,5}, {1,15}#
{2,4}. (2,5}, {2,15},

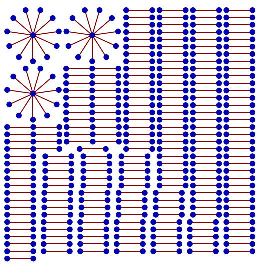
{{4,5}#, {4,15}#,
{5,15}#}

```
ArrayPlot[CellularAutomaton[ruleMD2[{15, 5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```

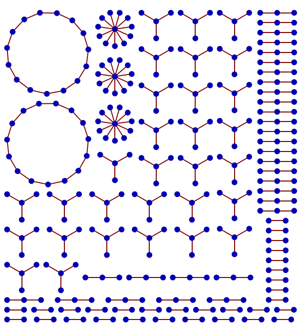
```
$Aborted
```

```
(Debug) In[280]:= ArrayPlot[CellularAutomaton[ruleMD2[{5, 15}], {{1}, 0}, 22],
  ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
  Mesh → True, ImageSize → 400]

(Debug) Out[280]= $Aborted

(Debug) In[279]:= GraphPlot[# -> CellularAutomaton[ruleMD2[{15, 5}], #] & /@ Tuples[{0, 1, 2}, 5]]
```

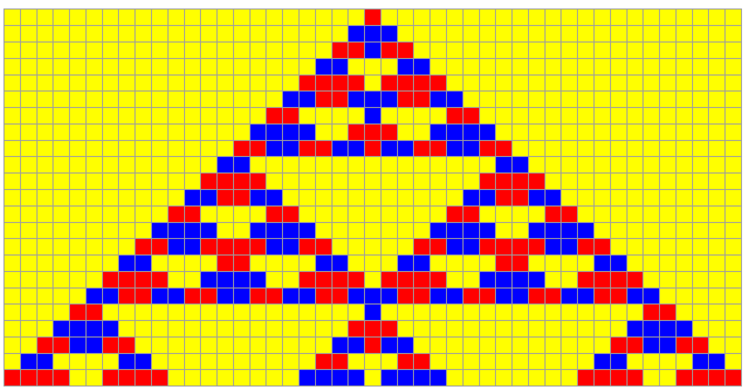
(Debug) Out[279]=



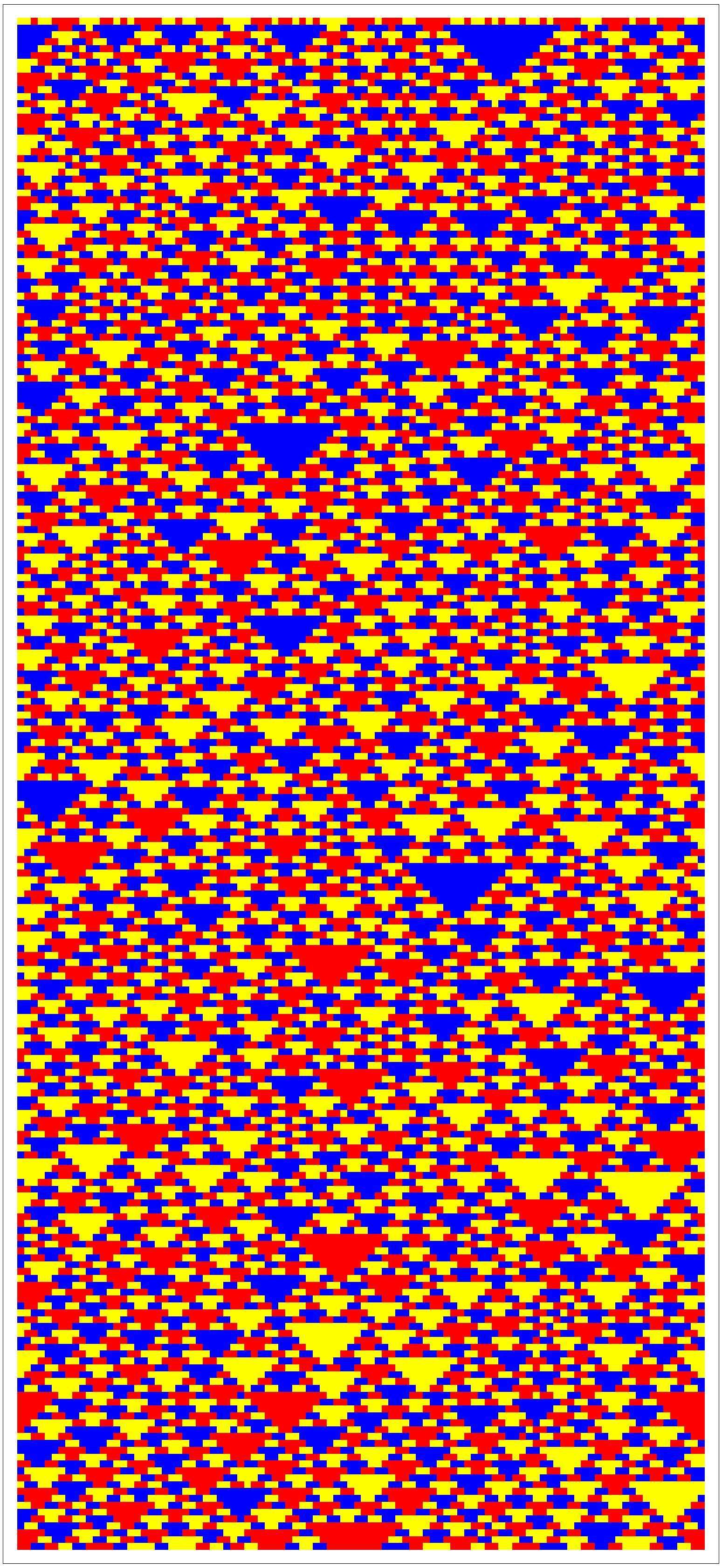
```
GraphPlot[# -> CellularAutomaton[ruleMD2[{5, 15}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



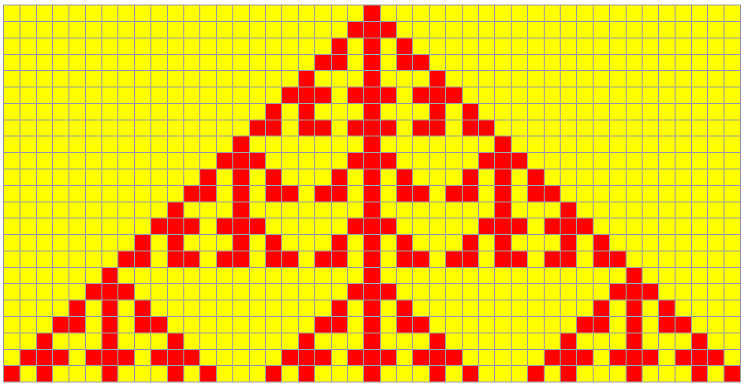
```
ArrayPlot[CellularAutomaton[ruleMD2[{1, 5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



```
ArrayPlot[CellularAutomaton[ruleMD2[{1, 5}], RandomInteger[1, 100], 222],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → False, ImageSize → 400]
```

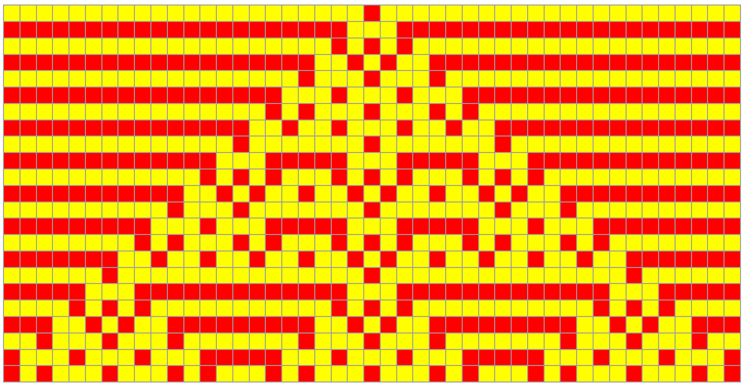


```
ArrayPlot[CellularAutomaton[ruleMD2[{1, 4}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```

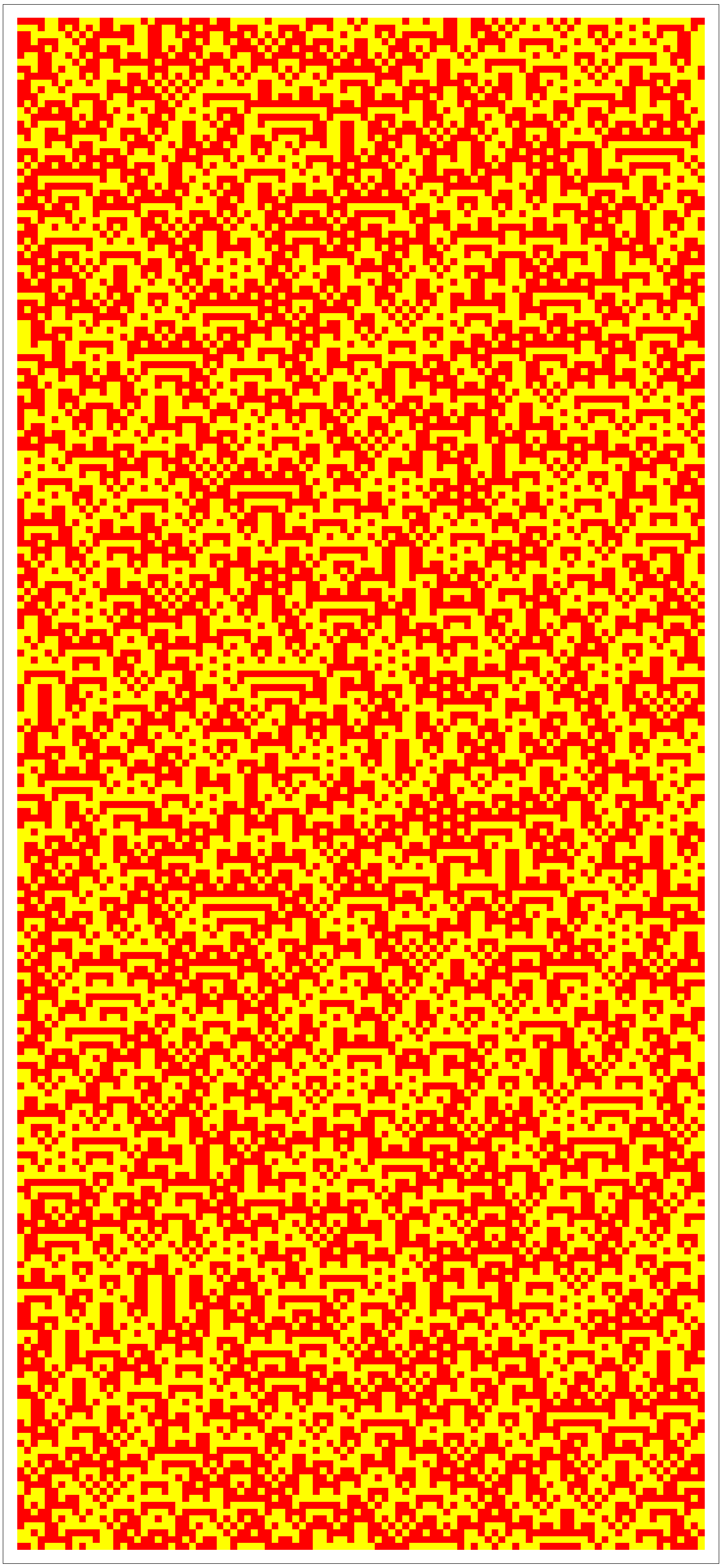


```
ArrayPlot[CellularAutomaton[ruleMD2[{1, 4}], RandomInteger[1, 100], 222],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → False, ImageSize → 400]
```

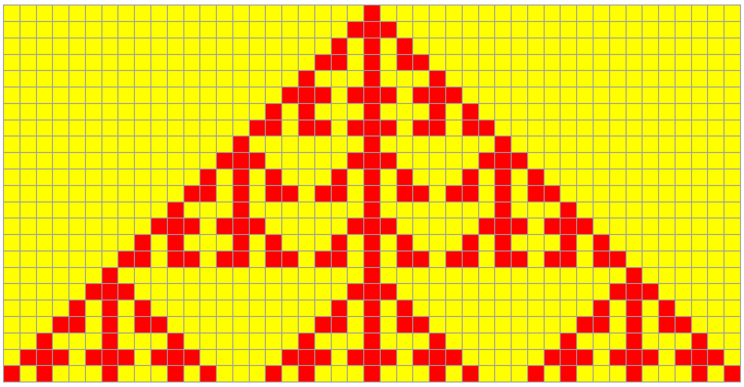
```
ArrayPlot[CellularAutomaton[ruleMD2[{2, 5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



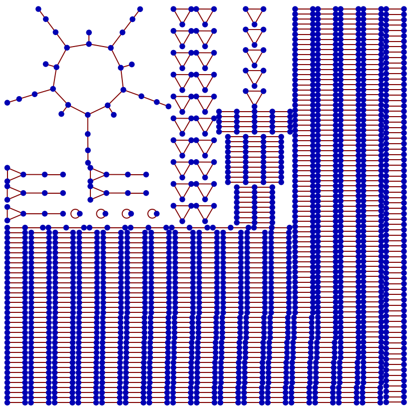
```
ArrayPlot[CellularAutomaton[ruleMD2[{2, 5}], RandomInteger[1, 100], 222],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → False, ImageSize → 400]
```



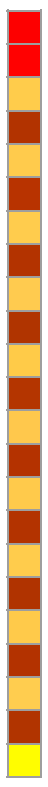
```
ArrayPlot[CellularAutomaton[ruleMD2[{1, 4}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



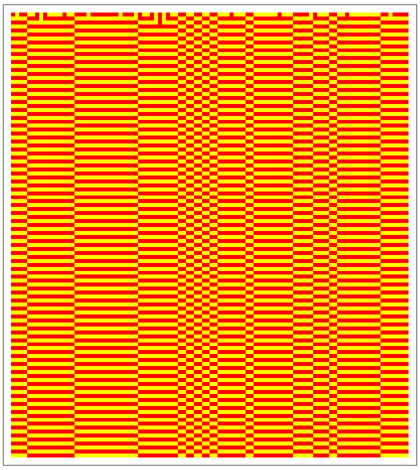
```
GraphPlot[# -> CellularAutomaton[ruleMD2[{1, 4}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```



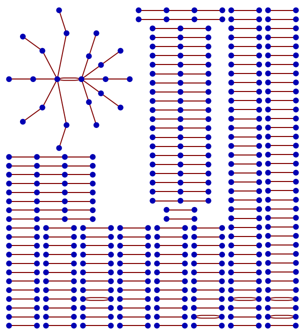
```
ArrayPlot[CellularAutomaton[ruleMD2[{4, 2}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {200, 200}]
```



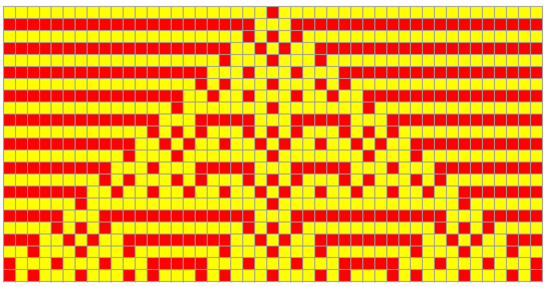
```
ArrayPlot[CellularAutomaton[ruleMD2[{4, 2}], RandomInteger[1, 100], 111],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → False, ImageSize → {200, 200}]
```



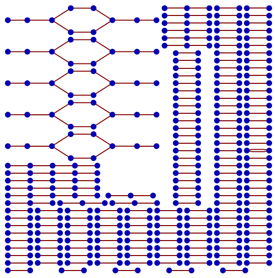
```
GraphPlot[# -> CellularAutomaton[ruleMD2[{4, 2}], #] & /@Tuples[{0, 1, 2}, 5]]
```



```
ArrayPlot[CellularAutomaton[ruleMD2[{2, 4}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 200]
```

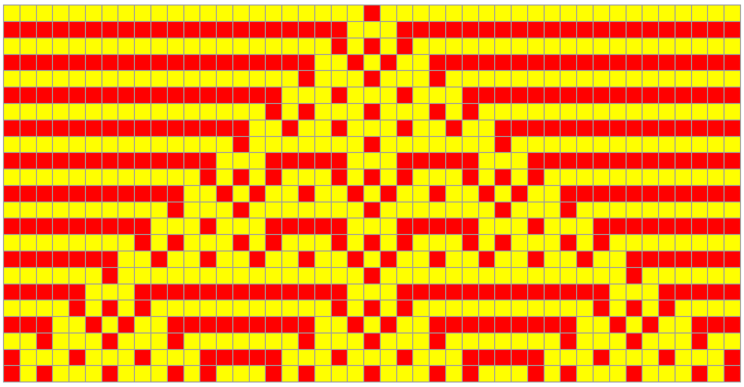


```
GraphPlot[# -> CellularAutomaton[ruleMD2[{2, 4}], #] & /@ Tuples[{0, 1, 2}, 5]]
```

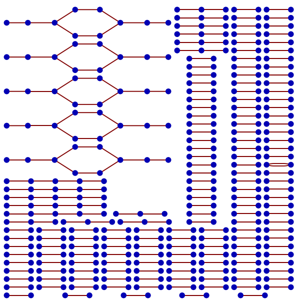


Equivalence and difference in ruleMD2

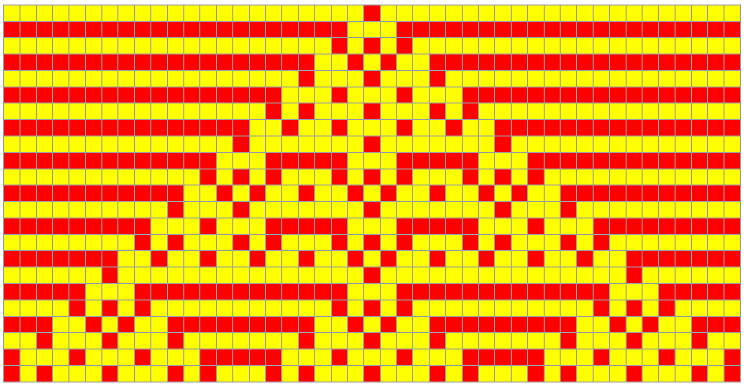
```
ArrayPlot[CellularAutomaton[ruleMD2[{2, 4}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



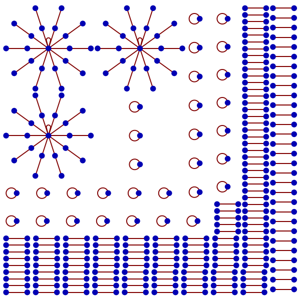
```
GraphPlot[# -> CellularAutomaton[ruleMD2[{2, 4}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



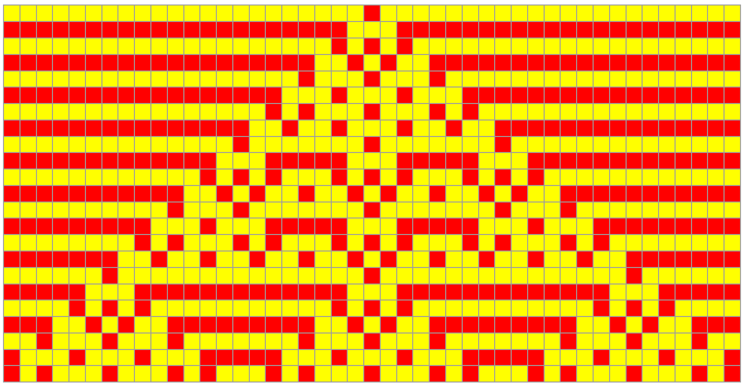
```
ArrayPlot[CellularAutomaton[ruleMD2[{2, 1}], {{1}, 0}, 22] ColorRules ->
  {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}, Mesh → True, ImageSize → 400]
```



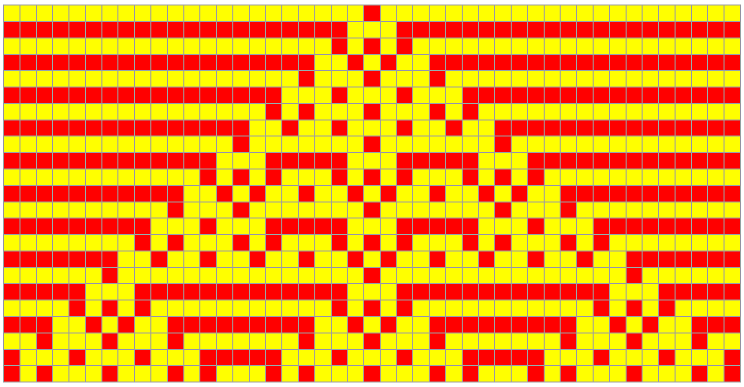
```
GraphPlot[# -> CellularAutomaton[ruleMD2[{1, 2}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



```
ArrayPlot[CellularAutomaton[ruleMD2[{2, 15}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



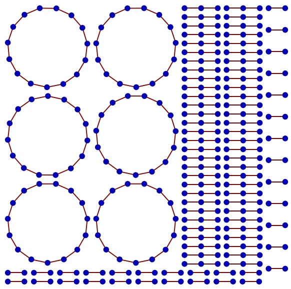
```
ArrayPlot[CellularAutomaton[ruleMD2[{15, 2}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



```
ArrayPlot[CellularAutomaton[ruleMD2[{4, 5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {600, 400}]
```

$Aborted

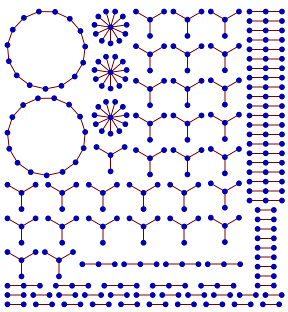
```
GraphPlot[# -> CellularAutomaton[ruleMD2[{4, 5}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



```
ArrayPlot[CellularAutomaton[ruleMD2[{5, 4}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {600, 400}]
```

$Aborted

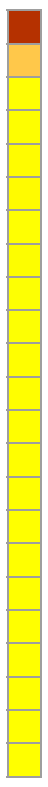
```
GraphPlot[# -> CellularAutomaton[ruleMD2[{5, 4}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



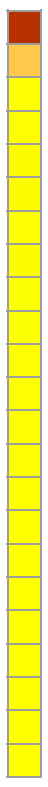
Domain ruleMD3

ruleMD3 = {{1,2,4}, {1,2,5}, {1,2,15}, {1,4,5},{1,4,15},
{2,4,5}, {2,4,15}, {4,5,15}#}

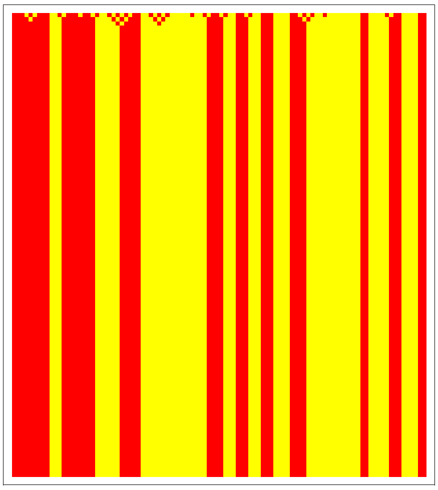
```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 2, 4}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {200, 200}]
```



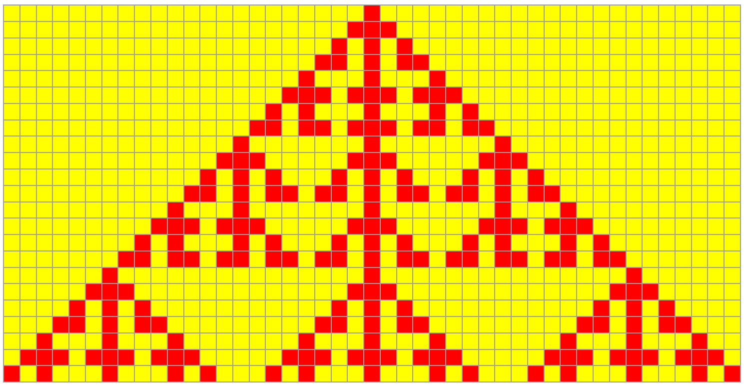
```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 2, 15}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {200, 200}]
```



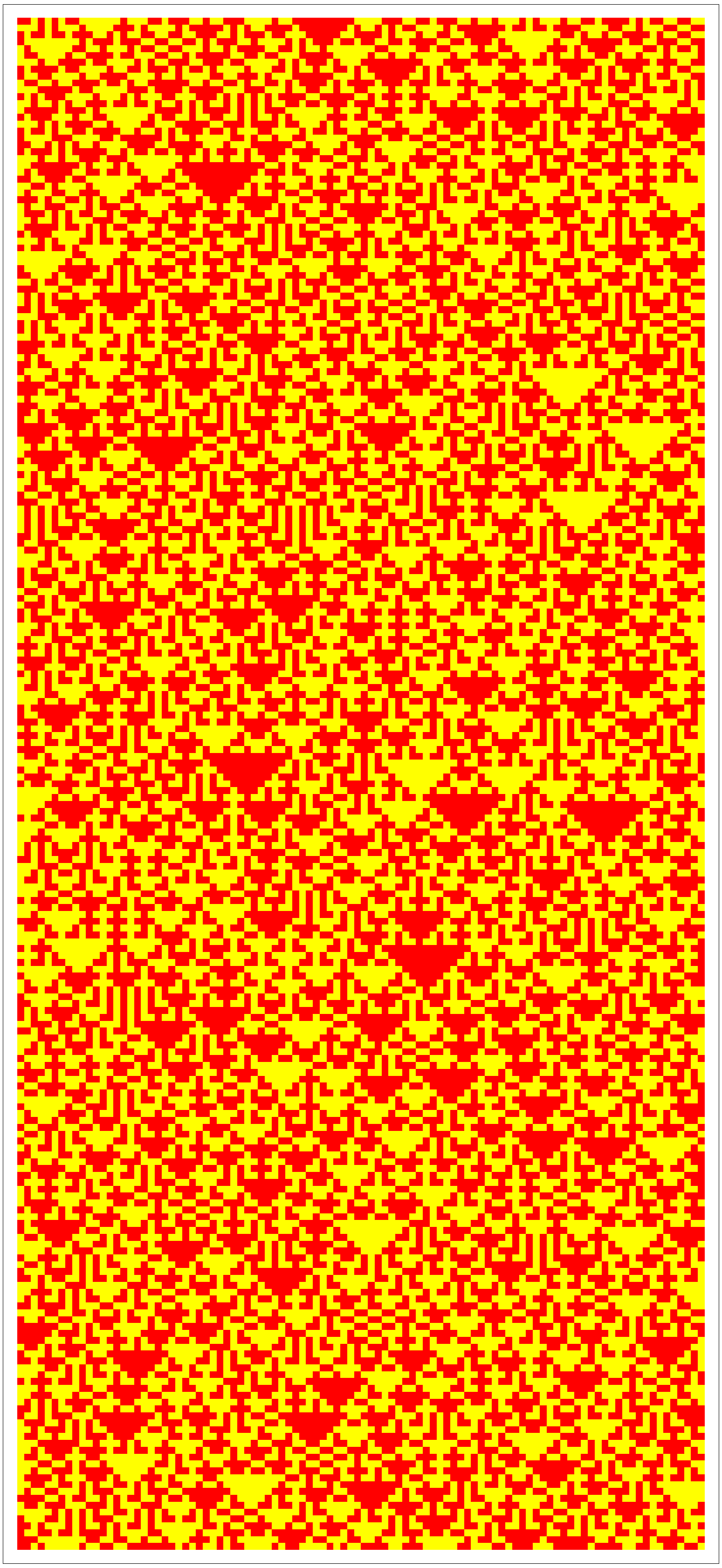
```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 2, 15}], RandomInteger[1, 100], 111],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → False, ImageSize → {200, 200}]
```



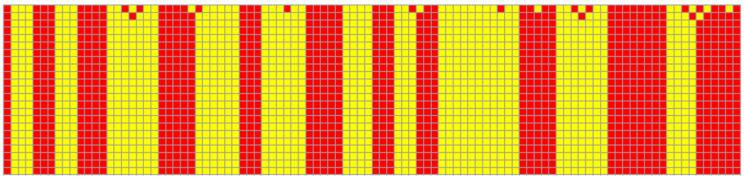
```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 4, 15}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



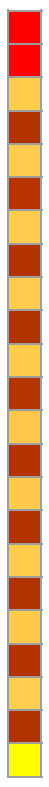
```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 4, 5}], RandomInteger[1, 100], 222],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → False, ImageSize → 400]
```



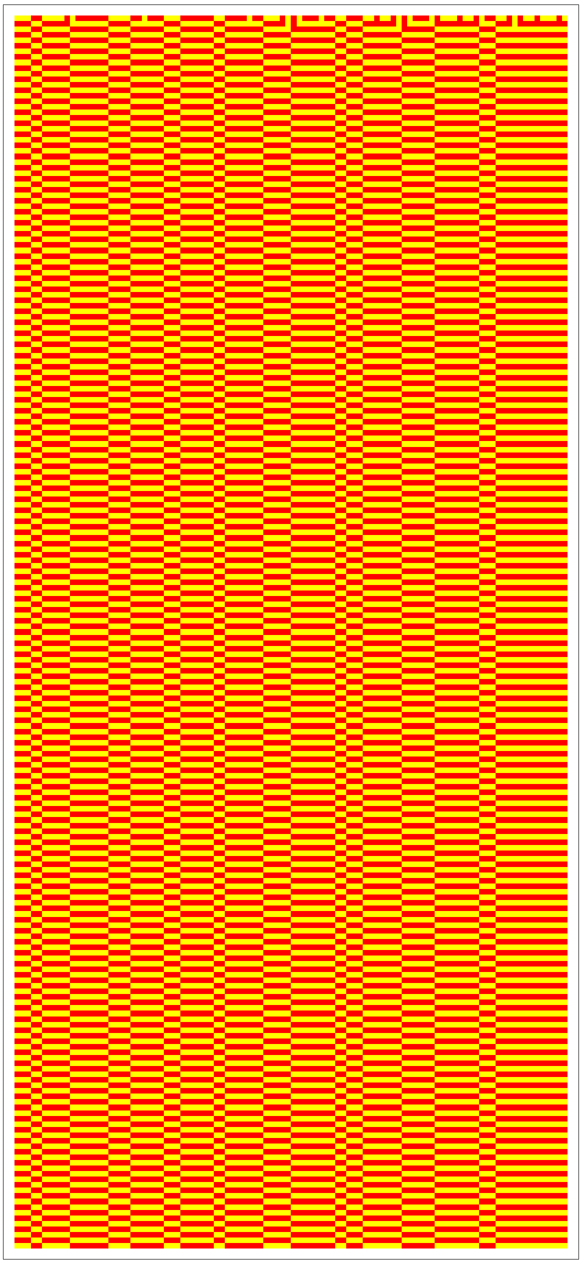
```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 2, 5}], RandomInteger[1, 100], 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



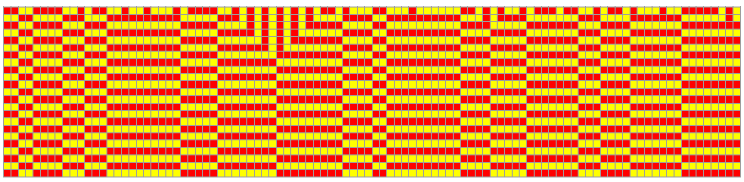
```
ArrayPlot[CellularAutomaton[ruleMD3[{4, 2, 15}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {400, 200}]
```



```
ArrayPlot[CellularAutomaton[ruleMD3[{4, 2, 15}], RandomInteger[1, 100], 222],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → False, ImageSize → 300]
```



```
ArrayPlot[CellularAutomaton[ruleMD3[{4, 2, 5}], RandomInteger[1, 100], 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



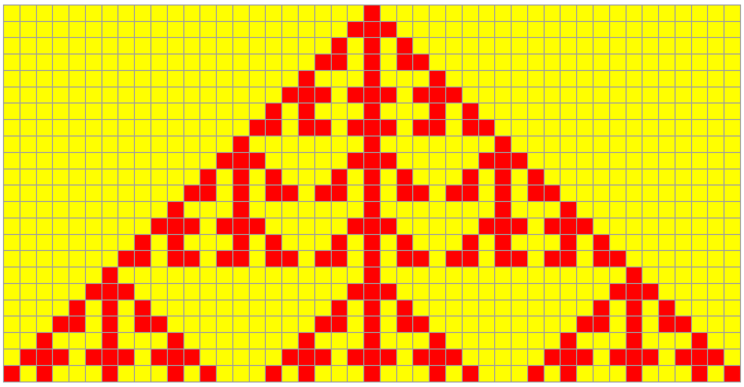
```
ArrayPlot[CellularAutomaton[ruleMD3[{4, 2, 5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {400, 200}]
```

$Aborted

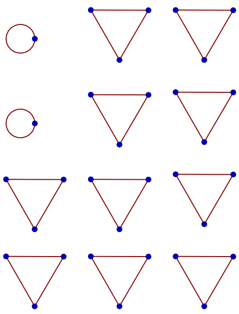
```
ArrayPlot[CellularAutomaton[ruleMD3[{4, 5, 15}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {400, 200}]
```

$Aborted

```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 4, 5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```

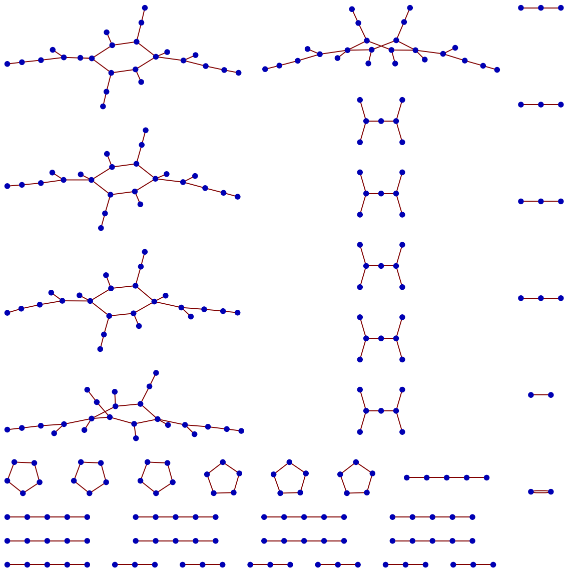


```
GraphPlot[# -> CellularAutomaton[ruleMD3[{1, 4, 5}], #] & /@ Tuples[{0, 1}, 5]]
```

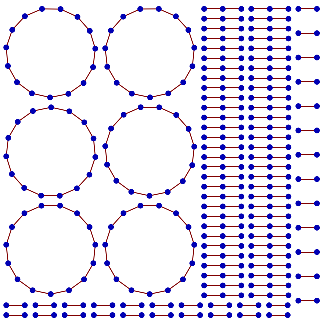


Aborted Cases

```
GraphPlot[# -> CellularAutomaton[ruleMD3[{2, 4, 5}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



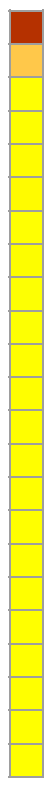
```
GraphPlot[# -> CellularAutomaton[ruleMD3[{4, 5, 15}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



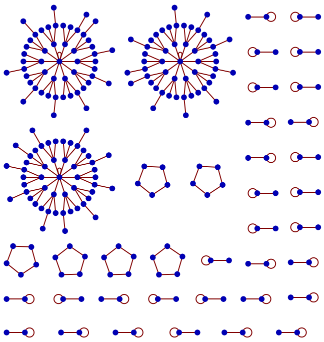
Domain ruleMD4

```
ruleMD4 = {{1, 2, 4, 5}, {1, 2, 4, 15}, { 2, 4, 5, 15}}
```

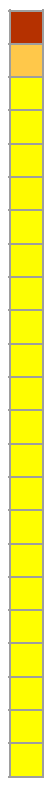
```
ArrayPlot[CellularAutomaton[ruleMD4[{1, 2, 4, 5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {200, 200}]
```



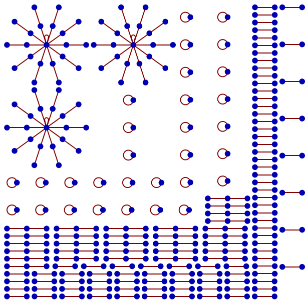
```
GraphPlot[# -> CellularAutomaton[ruleMD4[{1, 2, 4, 5}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



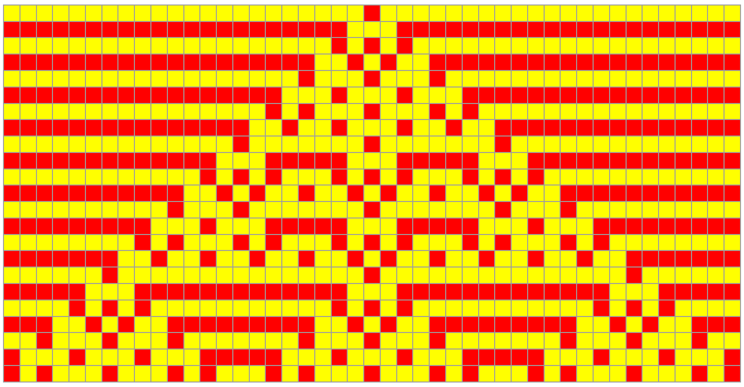
```
ArrayPlot[CellularAutomaton[ruleMD4[{1, 2, 4, 15}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {200, 200}]
```



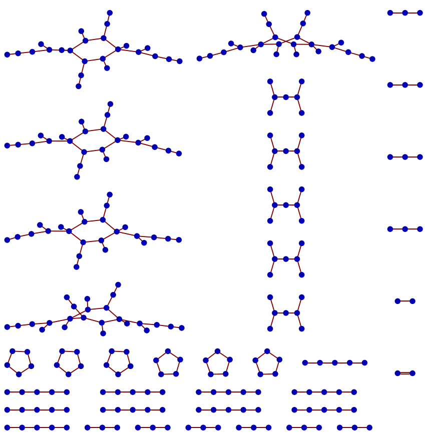
```
GraphPlot[
 # -> CellularAutomaton[ruleMD4[{1, 2, 4, 15}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



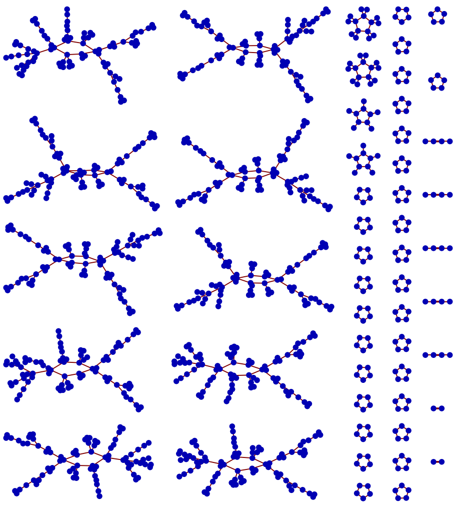
```
ArrayPlot[CellularAutomaton[ruleMD4[{2, 4, 5, 15}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



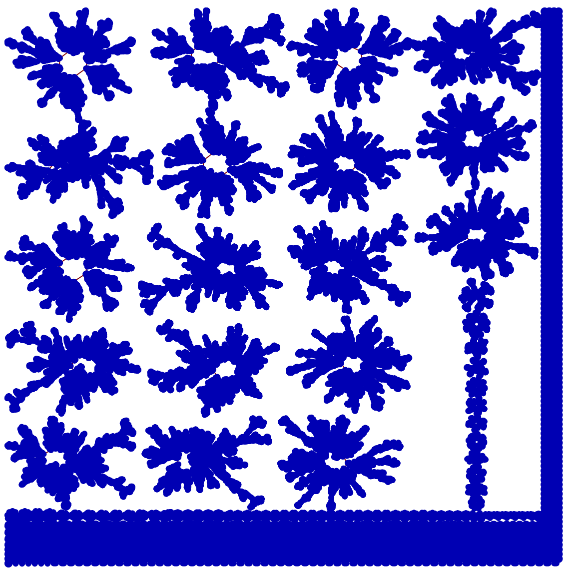
```
GraphPlot[
 # -> CellularAutomaton[ruleMD4[{2, 4, 5, 15}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



```
GraphPlot[
 # -> CellularAutomaton[ruleMD4[{2, 4, 5, 15}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```

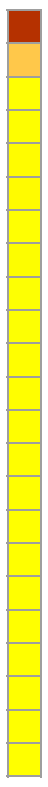


```
GraphPlot[
 # -> CellularAutomaton[ruleMD4[{2, 4, 5, 15}], #] & /@ Tuples[{0, 1, 2, 3}, 9]]
```

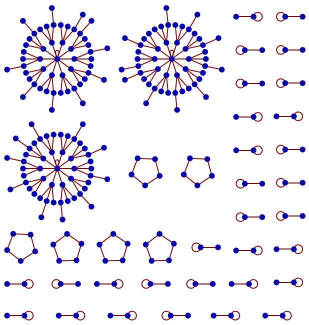


Domain ruleMD5

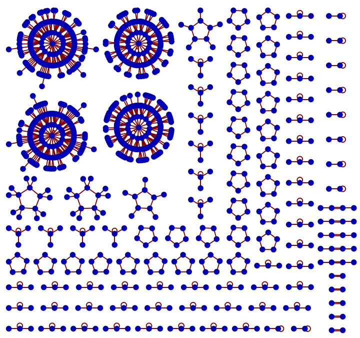
```
ArrayPlot[CellularAutomaton[ruleMD5[{1, 2, 4, 5, 15}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {200, 200}]
```



```
GraphPlot[
 # -> CellularAutomaton[ruleMD5[{1, 2, 4, 5, 15}], #] & /@ Tuples[{0, 1, 2}, 5]]
```

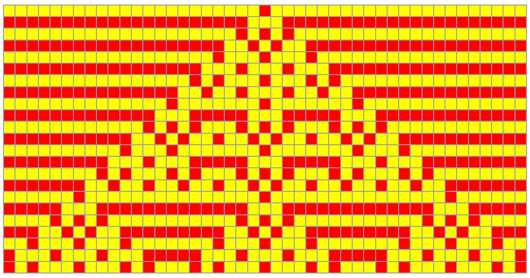


```
GraphPlot[
 # -> CellularAutomaton[ruleMD5[{1, 2, 4, 5, 15}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```

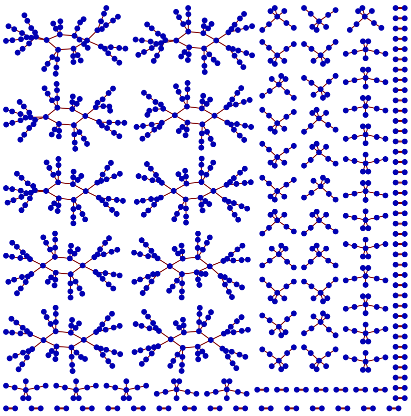


Same visualization but different transition graphs

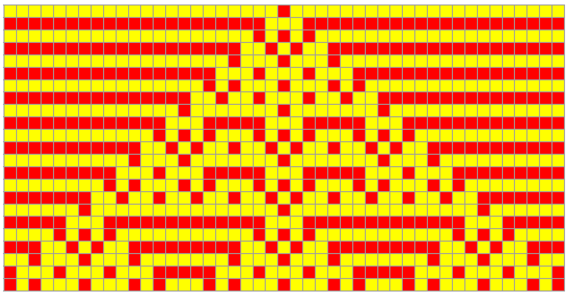
```
ArrayPlot[CellularAutomaton[ruleMD5[{15, 2, 5, 4, 1}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {200}]
```



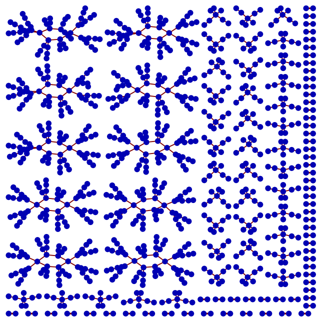
```
GraphPlot[
 # -> CellularAutomaton[ruleMD5[{15, 2, 5, 4, 1}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```



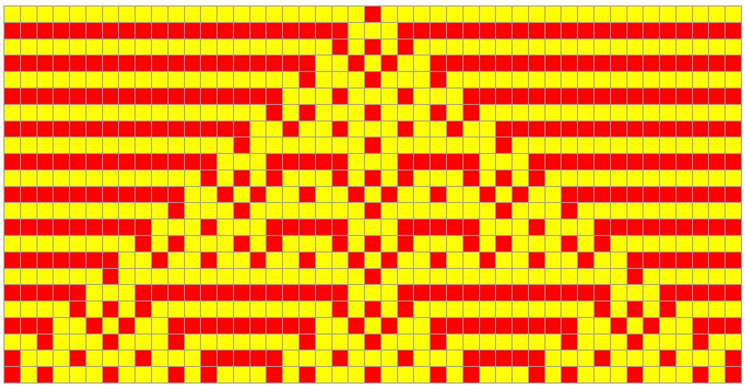
```
ArrayPlot[CellularAutomaton[ruleMD5[{15, 2, 1, 4, 5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → {200}]
```



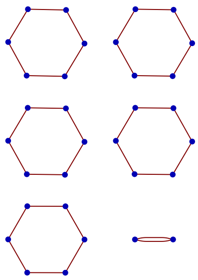
```
GraphPlot[
 # -> CellularAutomaton[ruleMD5[{15, 2, 1, 4, 5}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```



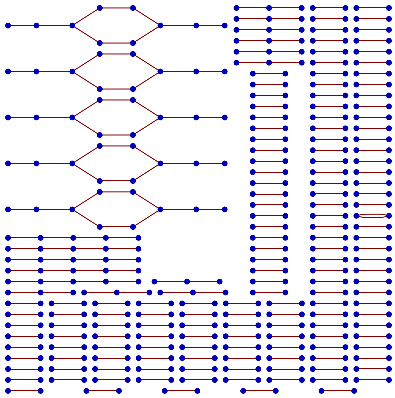
```
ArrayPlot[CellularAutomaton[ruleMD1[{2}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



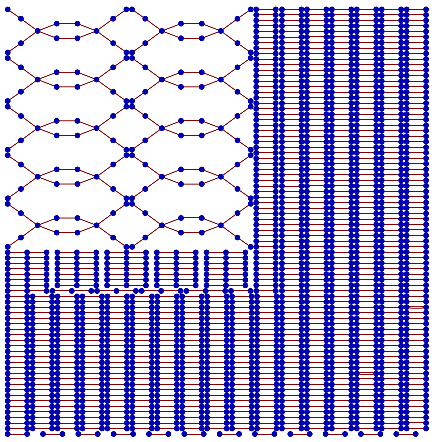
```
GraphPlot[# -> CellularAutomaton[ruleMD1[{2}], #] & /@ Tuples[{0, 1}, 5]]
```



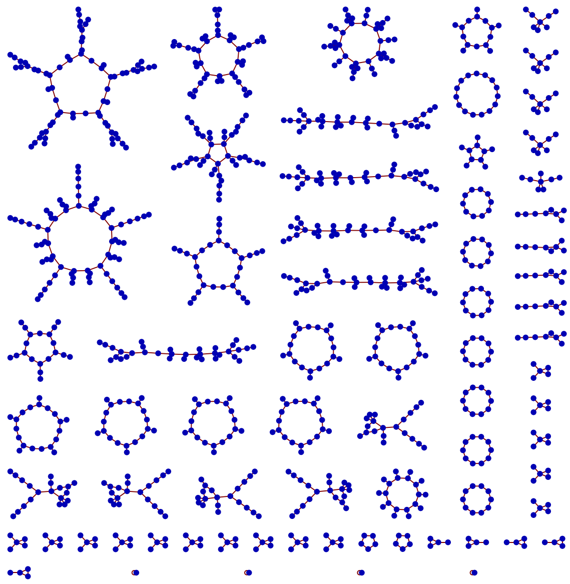
```
GraphPlot[# -> CellularAutomaton[ruleMD1[{2}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



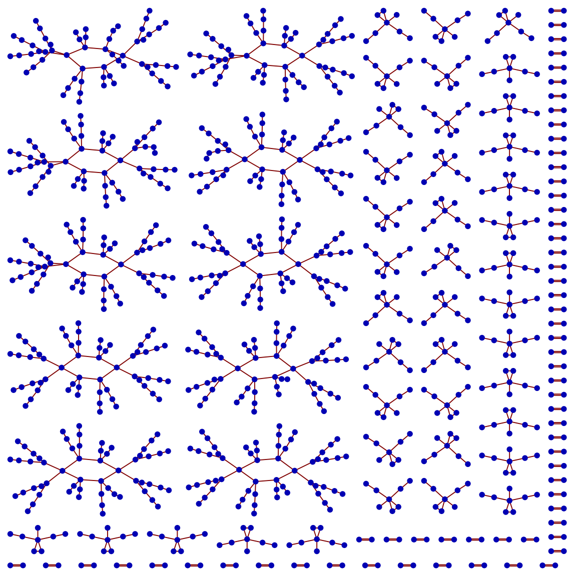
```
GraphPlot[# -> CellularAutomaton[ruleMD1[{2}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```



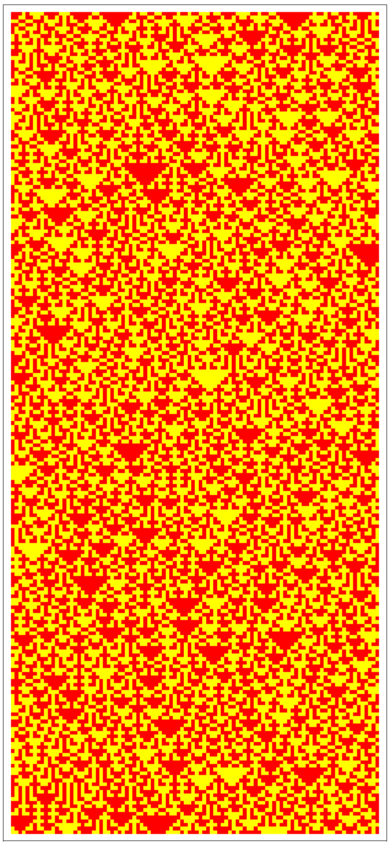
```
GraphPlot[
 # -> CellularAutomaton[ruleMD4[{1, 4, 2, 5}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```



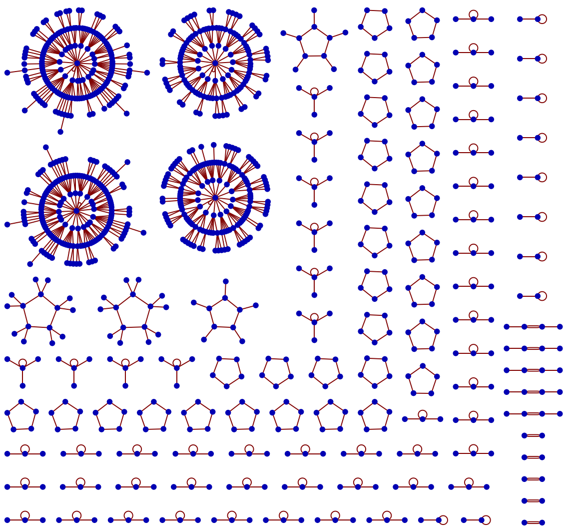
```
GraphPlot[
 # -> CellularAutomaton[ruleMD3[{2, 4, 15}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```



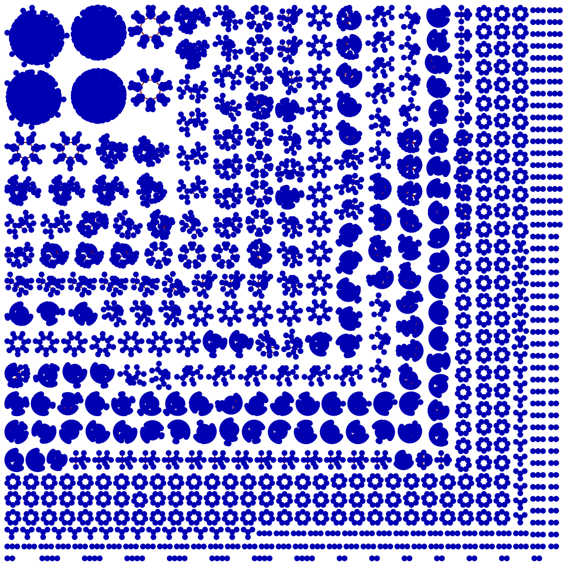
```
ArrayPlot[CellularAutomaton[ruleMD5[{1, 4, 2, 5, 15}],
   RandomInteger[1, 100], 222],
  ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue,  3 → Green},
  Mesh → False, ImageSize → 200]
```



```
GraphPlot[
 # -> CellularAutomaton[ruleMD5[{1, 2, 4, 5, 15}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```



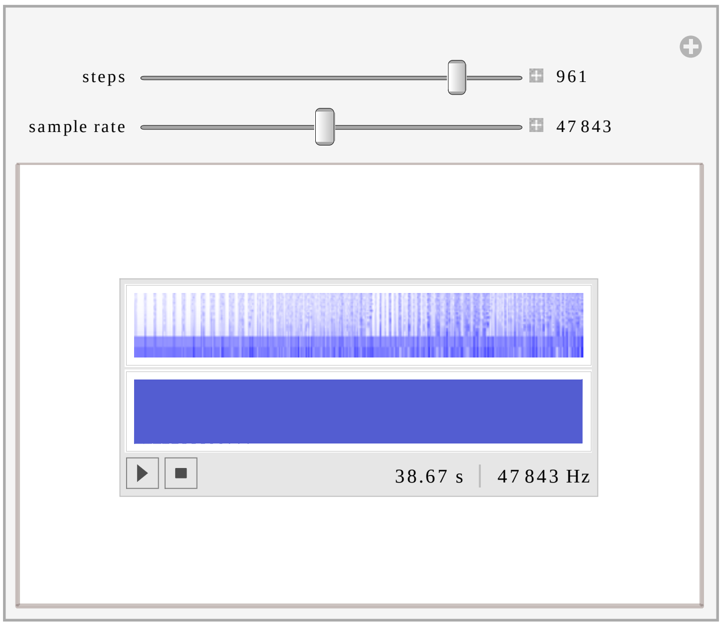
```
GraphPlot[
 # -> CellularAutomaton[ruleMD5[{1, 2, 4, 5, 15}], #] & /@Tuples[{0, 1, 2, 3}, 7]]
```



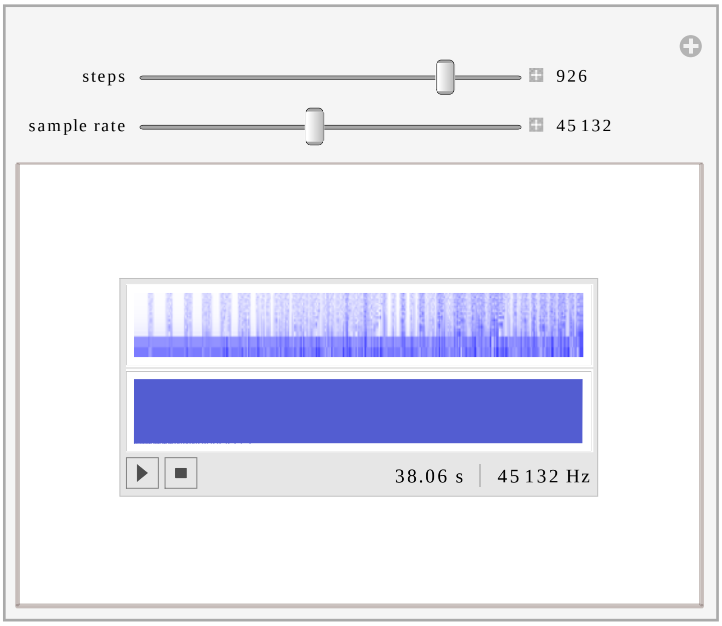
Sound representations

```
Manipulate[
 Pane[
  Quiet@ListPlay[
    Flatten[CellularAutomaton[ruleMD3[{1, 4, 15}], {{1}, 0}, st]],
    SampleRate → sr], {325, 200}, Alignment → Center],
 {{st, 5, "steps"}, 0, 1111, 1, Appearance → "Labeled"},
 {{sr, 99 000, "sample rate"}, 200, 99 000, 1, Appearance → "Labeled"}]
```

(Debug) Out[11]=

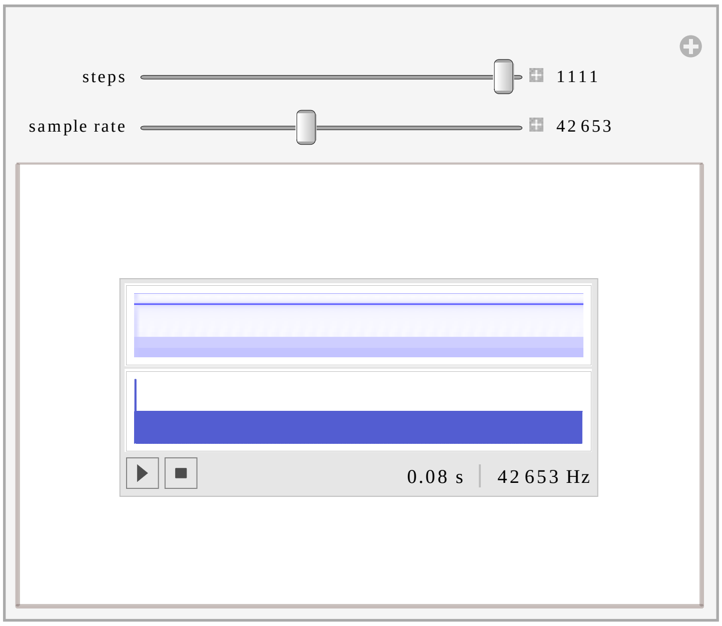


```
Manipulate[
 Pane[
  Quiet@ListPlay[
    Flatten[CellularAutomaton[ruleMD3[{4, 1, 2}], {{1}, 0}, st]], SampleRate → sr],
  {325, 200}, Alignment → Center],
 {{st, 5, "steps"}, 0, 1111, 1, Appearance → "Labeled"},
 {{sr, 99 000, "sample rate"}, 200, 99 000, 1, Appearance → "Labeled"}]
```




```
Manipulate[
 Pane[
  Quiet@ListPlay[
    Flatten[CellularAutomaton[ruleMD4[{5, 4, 2, 1}], {{1}, 0}, st]],
    SampleRate → sr], {325, 200}, Alignment → Center],
 {{st, 5, "steps"}, 0, 1111, 1, Appearance → "Labeled"},
 {{sr, 99 000, "sample rate"}, 200, 99 000, 1, Appearance → "Labeled"}]
```

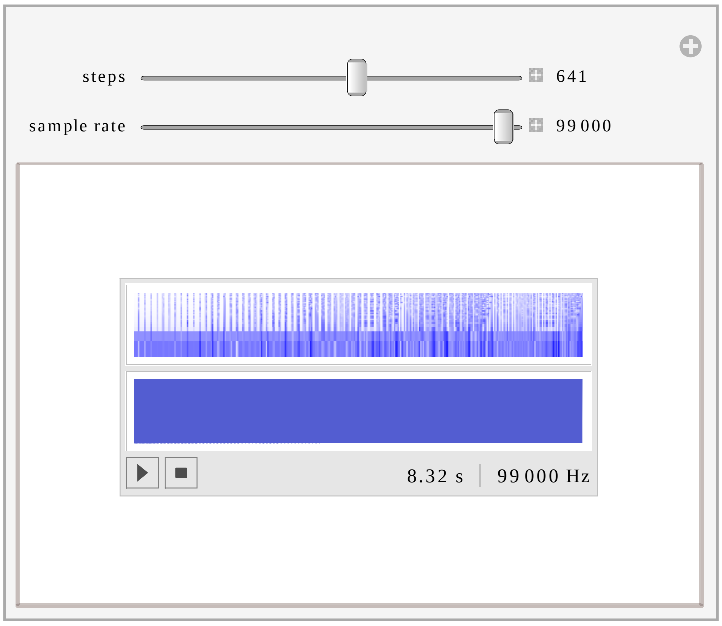
(Debug) Out[16]=



(Debug) In[17]:=

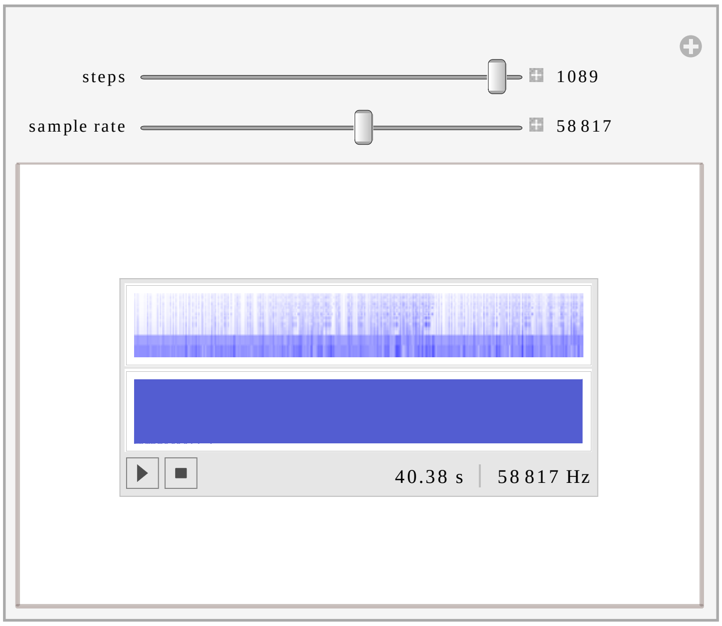
```
Manipulate[
 Pane[
  Quiet@ListPlay[
    Flatten[CellularAutomaton[ruleMD3[{1, 4, 15}], {{1}, 0}, st]],
    SampleRate → sr], {325, 200}, Alignment → Center],
 {{st, 5, "steps"}, 0, 1111, 1, Appearance → "Labeled"},
 {{sr, 99 000, "sample rate"}, 200, 99 000, 1, Appearance → "Labeled"}]
```

(Debug) Out[17]=



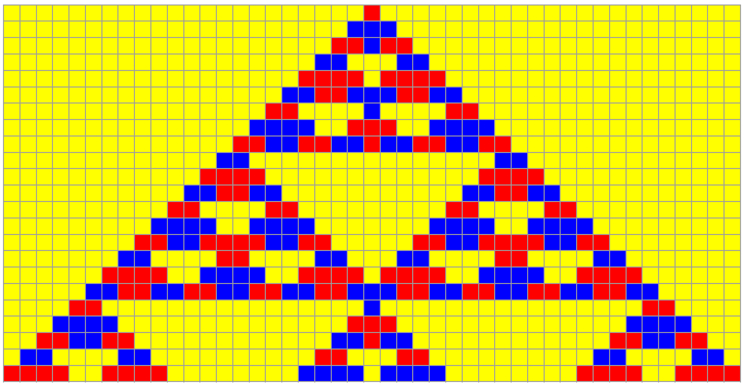

```
Manipulate[
 Pane[
  Quiet@ListPlay[
    Flatten[CellularAutomaton[ruleMD2[{1, 5}], {{1}, 0}, st]], SampleRate → sr],
  {325, 200}, Alignment → Center],
 {{st, 5, "steps"}, 0, 1111, 1, Appearance → "Labeled"},
 {{sr, 99 000, "sample rate"}, 200, 99 000, 1, Appearance → "Labeled"}]
```

(Debug) Out[13]=

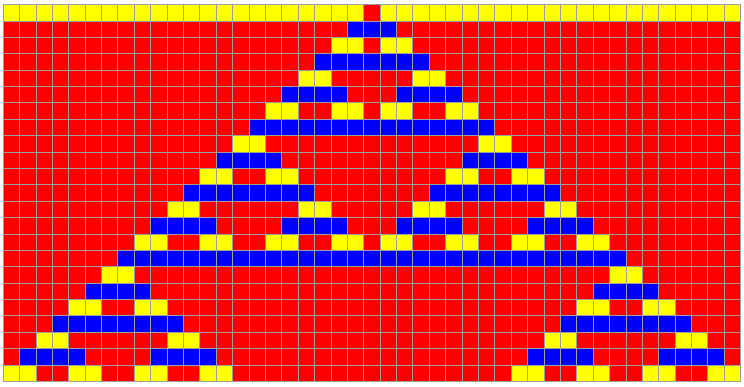


Comparison of deutero- and trito-Rules

```
ArrayPlot[CellularAutomaton[ruleMD2[{1, 5}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



```
ArrayPlot[CellularAutomaton[ruleMN[{1, 5, 12, 11, 10, 13}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 400]
```



Non-commutativity of rule components

In contrast to the morpho-rules of the type *ruleM*, *ruleMN*, etc. deutero-rules are not generally commutative.

For the domain of *ruleM*, the constituents are strictly disjunctive and are defining patterns and not partitions, therefore they are supporting commutativity.

Disjunctivity of the components of for *ruleMD* holds too, but the functions are defining *partitions* and not patterns, hence the commutativity of the domain *ruleMD* is not granted in general.

Non-commutative constellations

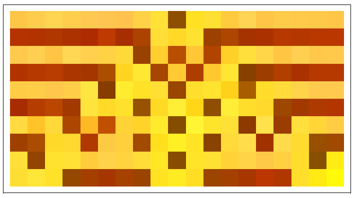
Non – commutative constellations

(1, 2) ≠ (2, 1),
(4, 2) ≠ (2, 4),
(5, 2) ≠ (2, 5).
perm[{1, 2, 5}] ∉ Commutativity,
perm[{1, 2, 5, 15}] ∉ Commutativity,
[{1, 4, 5}] = [{4, 1, 5}] ≠ [{5, 1, 4}],
[{15, 1, 2}] = [{1, 2, 15}] ≠ [{2, 1, 15}]
[{1, 2, 4, 5, 15}] ≠ [{15, 5, 4, 2, 1}]

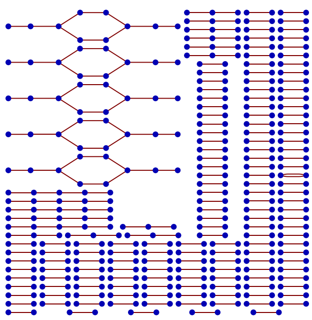
Examples for non-commutativity

(Debug) In[20]:=
```
ArrayPlot[CellularAutomaton[
  ruleMD2[{2, 4}],
  {{1}, 0}, 9],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[20]=

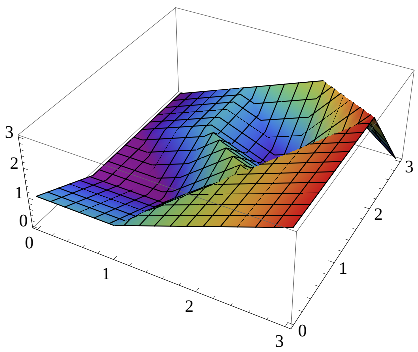


```
GraphPlot[# -> CellularAutomaton[ruleMD2[{2, 4}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



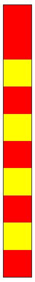
(Debug) In[238]:=
```
ListPlot3D[ruleMD2[{2, 4}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[238]=

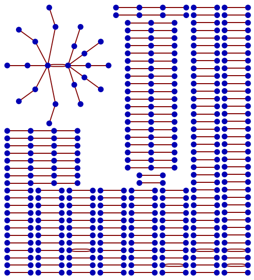


(Debug) In[21]:=
```
ArrayPlot[CellularAutomaton[
   ruleMD2[{4, 2}],
   {{1}, 0}, 9],
  ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}]
```

(Debug) Out[21]=

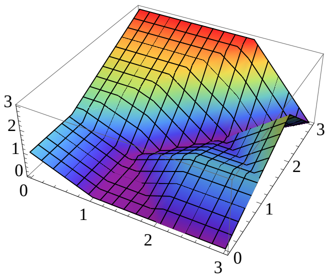


```
GraphPlot[# -> CellularAutomaton[ruleMD2[{4, 2}], #] & /@ Tuples[{0, 1, 2}, 5]]
```

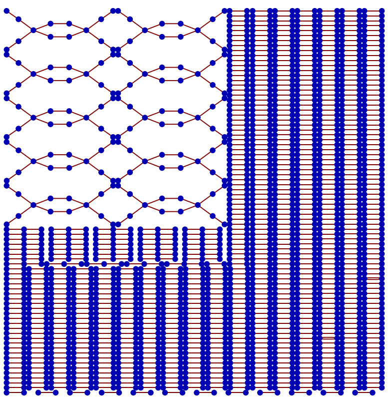


(Debug) In[239]:=
```
ListPlot3D[ruleMD2[{4, 2}], ColorFunction → "Rainbow", Mesh → True]
```

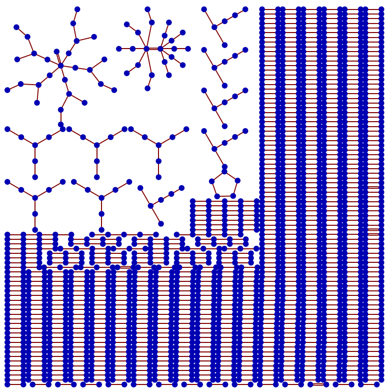
(Debug) Out[239]=



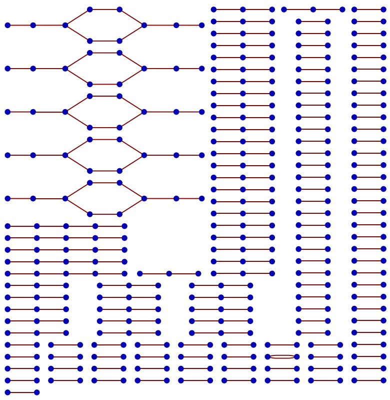

```
GraphPlot[# -> CellularAutomaton[ruleMD2[{2, 4}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```



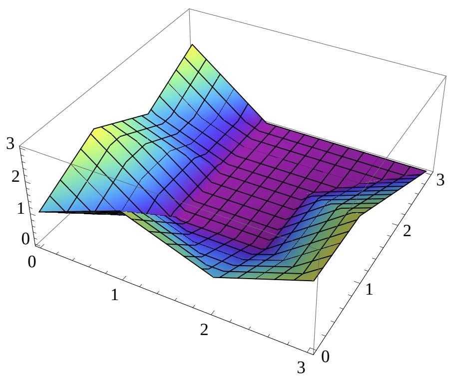
```
GraphPlot[# -> CellularAutomaton[ruleMD2[{4, 2}], #] & /@ Tuples[{0, 1, 2, 3}, 5]]
```



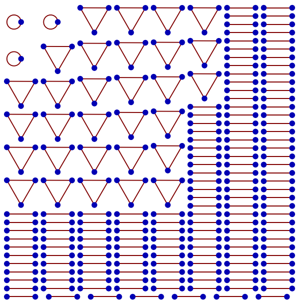
```
GraphPlot[
 # -> CellularAutomaton[ruleMD4[{15, 2, 4, 1}], #] & /@ Tuples[{0, 1, 2}, 5]]
```



(Debug) In[240]:=
```
ListPlot3D[ruleMD4[{15, 2, 4, 1}], ColorFunction → "Rainbow", Mesh → True]
```

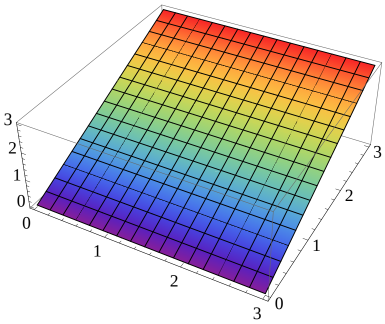


```
GraphPlot[
 # -> CellularAutomaton[ruleMD4[{1, 4, 2, 15}], #] & /@ Tuples[{0, 1, 2}, 5]]
```

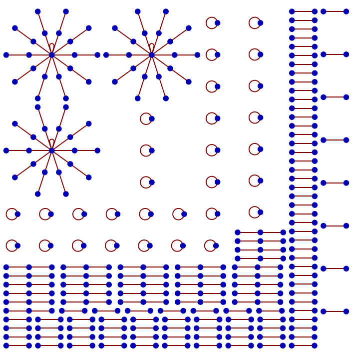


(Debug) In[241]:= `ListPlot3D[ruleMD4[{1, 4, 2, 15}], ColorFunction → "Rainbow", Mesh → True]`

(Debug) Out[241]=

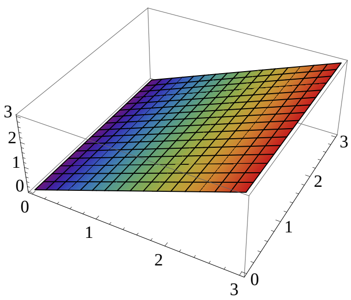



```
GraphPlot[
 # -> CellularAutomaton[ruleMD4[{1, 2, 15, 4}], #] & /@ Tuples[{0, 1, 2}, 5]]
```

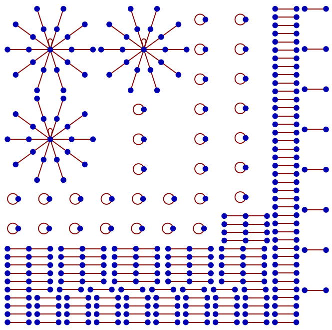


(Debug) In[242]:= `ListPlot3D[ruleMD4[{1, 2, 15, 4}], ColorFunction → "Rainbow", Mesh → True]`

(Debug) Out[242]=

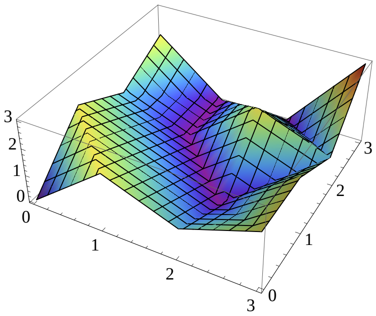


```
GraphPlot[
 # -> CellularAutomaton[ruleMD4[{1, 15, 2, 4}], #] & /@ Tuples[{0, 1, 2}, 5]]
```

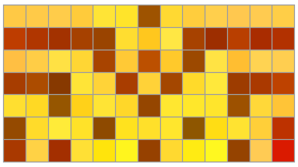


(Debug) In[243]:=
```
ListPlot3D[ruleMD4[{1, 15, 2, 4}], ColorFunction → "Rainbow", Mesh → True]
```

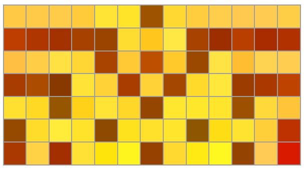
(Debug) Out[243]=



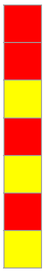

```
ArrayPlot[CellularAutomaton[ruleMD1[{2}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



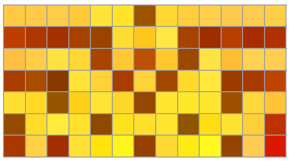
```
ArrayPlot[CellularAutomaton[ruleMD2[{2, 4}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 200]
```



```
ArrayPlot[CellularAutomaton[ruleMD2[{4, 2}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 200]
```

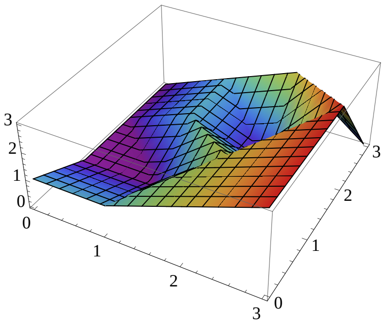


```
ArrayPlot[CellularAutomaton[ruleMD2[{2, 5}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 200]
```

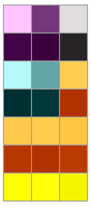


(Debug) In[245]:= `ListPlot3D[ruleMD2[{2, 5}], ColorFunction → "Rainbow", Mesh → True]`

(Debug) Out[245]=

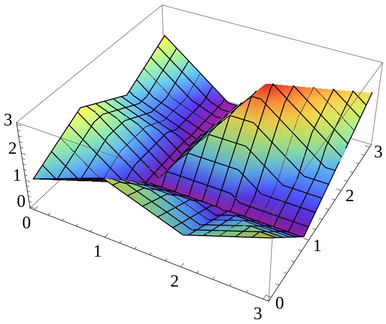



```
ArrayPlot[CellularAutomaton[ruleMD2[{5, 2}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

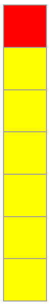


(Debug) In[244]:= `ListPlot3D[ruleMD2[{5, 2}], ColorFunction → "Rainbow", Mesh → True]`

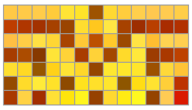
(Debug) Out[244]=



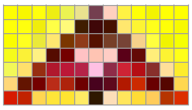
```
ArrayPlot[CellularAutomaton[ruleMD2[{1, 2}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



```
ArrayPlot[CellularAutomaton[ruleMD2[{2, 1}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

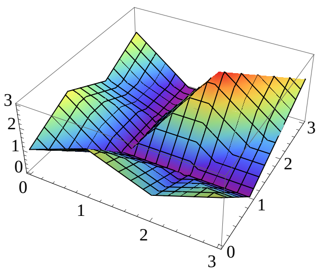


```
ArrayPlot[CellularAutomaton[ruleMD3[{5, 1, 2}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

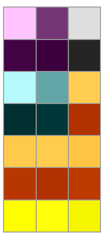


(Debug) In[248]:=
```
ListPlot3D[ruleMD3[{5, 1, 2}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[248]=

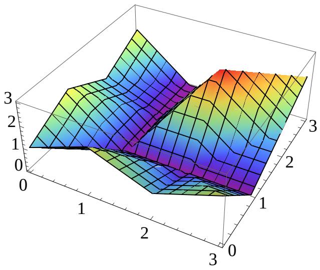


```
ArrayPlot[CellularAutomaton[ruleMD3[{5, 2, 1}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

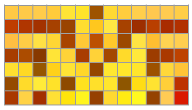


(Debug) In[249]:=
```
ListPlot3D[ruleMD3[{5, 2, 1}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[249]=

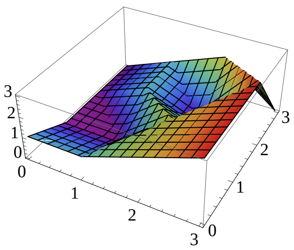



```
ArrayPlot[CellularAutomaton[ruleMD3[{2, 1, 5}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

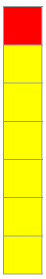


(Debug) In[250]:=
```
ListPlot3D[ruleMD3[{2, 1, 5}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[250]=

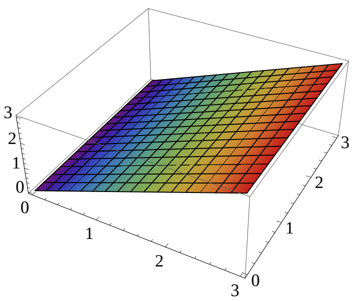



```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 2, 5}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

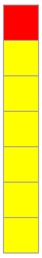


(Debug) In[251]:=
```
ListPlot3D[ruleMD3[{1, 2, 5}], ColorFunction → "Rainbow", Mesh → True]
```

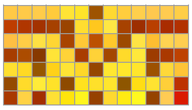
(Debug) Out[251]=



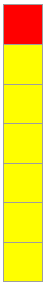
```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 2, 15}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



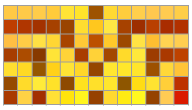
```
ArrayPlot[CellularAutomaton[ruleMD3[{2, 1, 15}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



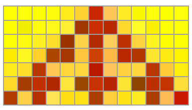
```
ArrayPlot[CellularAutomaton[ruleMD3[{15, 1, 2}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



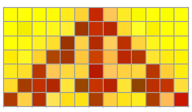
```
ArrayPlot[CellularAutomaton[ruleMD3[{15, 2, 1}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



```
ArrayPlot[CellularAutomaton[ruleMD3[{1, 4, 5}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

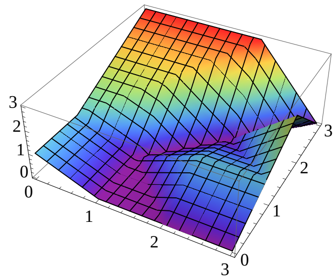


```
ArrayPlot[CellularAutomaton[ruleMD3[{4, 1, 5}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

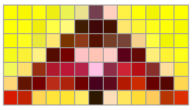


(Debug) In[254]:=
```
ListPlot3D[ruleMD3[{4, 1, 5}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[254]=

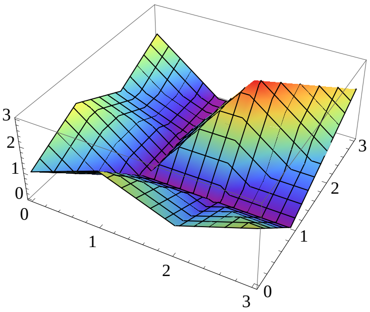


```
ArrayPlot[CellularAutomaton[ruleMD3[{5, 1, 4}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

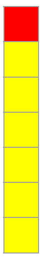


(Debug) In[253]:=
```
ListPlot3D[ruleMD3[{5, 1, 4}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[253]=

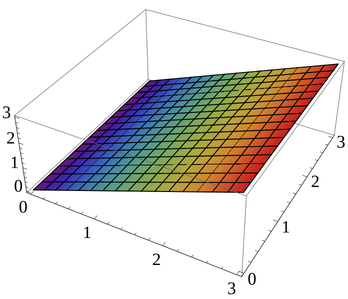


```
ArrayPlot[CellularAutomaton[ruleMD5[{1, 2, 4, 5, 15}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

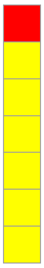


(Debug) In[255]:=
```
ListPlot3D[ruleMD5[{1, 2, 4, 5, 15}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[255]=

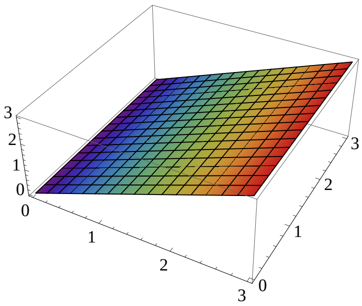


```
ArrayPlot[CellularAutomaton[ruleMD5[{1, 2, 5, 4, 15}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



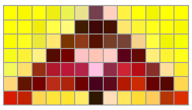
(Debug) In[256]:=
```
ListPlot3D[ruleMD5[{1, 2, 5, 4, 15}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[256]=



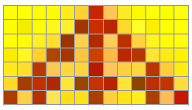
(Debug) In[18]:=
```
ArrayPlot[CellularAutomaton[ruleMD5[{1, 5, 4, 2, 15}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

(Debug) Out[18]=

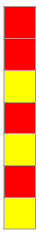


(Debug) In[19]:=
```
ArrayPlot[CellularAutomaton[ruleMD5[{1, 15, 4, 2, 5}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```

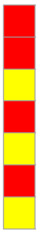
(Debug) Out[19]=



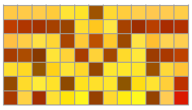
```
ArrayPlot[CellularAutomaton[ruleMD5[{4, 2, 5, 1, 15}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



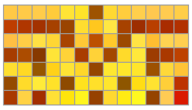
```
ArrayPlot[CellularAutomaton[ruleMD5[{4, 2, 1, 5, 15}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



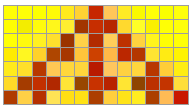
```
ArrayPlot[CellularAutomaton[ruleMD5[{2, 4, 1, 5, 15}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



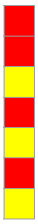
```
ArrayPlot[CellularAutomaton[ruleMD5[{2, 4, 5, 1, 15}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



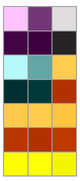
```
ArrayPlot[CellularAutomaton[ruleMD5[{15, 4, 5, 1, 2}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



```
ArrayPlot[CellularAutomaton[ruleMD5[{15, 4, 5, 2, 1}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



```
ArrayPlot[CellularAutomaton[ruleMD5[{15, 5, 4, 2, 1}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
 Mesh → True, ImageSize → 100]
```



(Debug) In[252]:=
```
ListPlot3D[ruleMD5[{15, 5, 4, 2, 1}], ColorFunction → "Rainbow", Mesh → True]
```

(Debug) Out[252]=