— vordenker-archive —

Rudolf Kaehr
(1942-2016)

Title

Self-modifying Forms of Cellular Automata

A sketch on different modi of self-reference for indicational CAs

Archive-Number / Categories

3_43 / K12, K09, K11

Publication Date

2015

Keywords / Topics

Morphograms, Celluilar Automata, Semiotics

Disciplines

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

Abstract

The Calculus of Indication (CI) is another kind of conceptualization and formalization that is not based in principle on 'expressions' and their discrete logic or cellular automata .
Like morphogrammatics is based on patterns (morphé) and not on propositions, indicational calculi are based on the act of distinctions - Draw a distinction! - and not on the utterance of propositions and their propositional logic.
Iterative self-modification is understood as a mainly mono-contextural concept, still lacking general reflectional and interactional features as properties of poly-contextural structurations. Insofar, iterative self-modification of indicational CAs happens intra-contexturally and therefore repeating its immanent structural conditions. Evolution might produce emergent behaviors in time but their structural conditions remain unchanged.
There are just two forms of fixed-points for the CI: a self-loop of length 1 and an iteration of length 2. The reflective CI, indCIR, is structured by a multiplicity of loops with length up to 36 distinctions.
A first step towards a reflectional indicational CA is introduced by the second-order concept of distinctions by the reflective CI covered by the single-reflectional CIR-rules.
A second step in the ladder of the self-reflecting tower is introduced by the double-reflectional distinctions. They are covered by the indicational cellular automata defined by the double-reflectional rules of indRCI. Certainly, their reflectional power are owerhelmingly more complex than the previous ones.
The sub-rule approach is introduced to deal with the whole complexity of the rule-spaces of the calculi, indCIR and indRCI as it was shown also for morphoCAs of higher complexity.
For ruleCIR, the rule-space is 3^10 = 59049.
For ruleRCI, the rule-space is 20^4 = 160000.
With the sub-rule approach all those functions are easily accessible for all kind of reflectional manipulations and computations. That doesen't eclude the chance to list some special classes of automata to deal with other, not sub-rule based, techniques. The case for morphogram based CAs is proposed at: "Self-modifying MorphoCellular Automata"

Citation Information / How to cite

Rudolf Kaehr: "Self-modifying Forms of Cellular Automata", www.vordenker.de (Sommer Edition, 2017) J. Paul (Ed.), http://www.vordenker.de/rk/rk_Self-modifying-Forms-of-CAs_2015.pdf

Categories of the RK-Archive

Self-modifying Forms of Cellular Automata

A sketch on different modi of self-reference for indicational CAs

Dr. phil Rudolf Kaehr

ISSN 2041-4358
(work in progress, v. 0.2, May 2015)

Abstract

The Calculus of Indication (CI) is another kind of conceptualization and formalization that is not based in principle on 'expressions' and their discrete logic or cellular automata .

Like morphogrammatics is based on *patterns* (morphé) and not on propositions, *indicational* calculi are based on the act of *distinctions* - Draw a distinction! - and not on the utterance of propositions and their propositional logic.

Iterative self-modification is understood as a mainly mono-contextural concept, still lacking general reflectional and interactional features as properties of poly-contextural structurations. Insofar, iterative self-modification of indicational CAs happens intra-contexturally and therefore repeating its immanent structural conditions. Evolution might produce emergent behaviors in time but their structural conditions remain unchanged.

There are just two forms of fixed-points for the CI: a *self-loop* of length 1 and an *iteration* of length 2.
The reflective CI, indCIR, is structured by a multiplicity of loops with length up to 36 distinctions.

A first step towards a reflectional indicational CA is introduced by the second-order concept of distinctions by the reflective CI covered by the *single-reflectional* CIR-rules.

A second step in the ladder of the self-reflecting tower is introduced by the double-reflectional distinctions. They are covered by the indicational cellular automata defined by the *double-reflectional* rules of *indRCI*. Certainly, their reflectional power are owerhelmingly more complex than the previous ones.

The *sub - rule* approach is introduced to deal with the whole complexity of the rule-spaces of the calculi, indCIR and indRCI as it was shown also for *morphoCAs* of higher complexity.

For *ruleCIR, the rule-space is* 3^10 = 59049.
For *ruleRCI, the rule-space is* 20^4 = 160000.

With the sub-rule approach all those functions are easily accessible for all kind of reflectional manipulations and computations.

That doesen't eclude the chance to list some special classes of automata to deal with other, not sub-rule based, techniques.

The case for *morphogram* based CAs is proposed at: Self-modifying MorphoCellular Automata.html (pdf, cdf)

Tasks:
Introduction of sub-rules for indCA and sub-rule manipulation for indCAs (CIR, RCI).
Self-modification as modification of the environments by init2init mappings.
Self-modification as modification of the rules and sub-rules by init2rule and rule2rule mappings.
Fixed points of distinctions. Morphic palindromes of distinctional fixed-points.

Self-modification for the Calculus of Indication

First-order calculus of forms

Motivation

The main interest into the *Calculus of Indication* (CI) is motivated by its features of self-referentiality proposed by the re-entry form. As it was observed very early, especially by Elena Esposito (1993), the re-entry construct is not a genuine part of the *Laws of Form* but a necessary albeit arbitrary add-on to the calculus to allow to do some reasonable math-like constructions in its framework.

It is therefore of special interest to study a possible self-referentiality on the base of an implementation of the *deep*-structure of the CI. The 'deep-structure' of the CI takes into account the proto-linguistic property of a '*topology-invariant*' condition of the 'syntax' of the CI (Kibet, Matzka), and doesn't relay on any external add-ons.

The academic reception of the CI was and still is generously neglecting or also denying this simple fact of the add-on character of the re-entry construction.

The result of a modeling of the CI in the framework of *cellular automata* is rather simple: The inherent self-referentiality of the CI has the simplest distinction possible: a *self-loop* of the form "AA" and a *loop* of the form "AB". The architectonics of the CI covers just two basic forms of self-referentiality. One is just *reflexive*, A⇌A, the other *iterativ*, A⇌B.

With that, the range of possible primary self-modifications are structurally very restricted.

More complex structural self-modifications in the '*topology-invariant*' mode of writing of the *Laws of Form* are accessible to implementations with the introduction of more complex types of indicational CAs, like $\text{indCA}^{(3,2)}$ and $indCA^{(4,3)}$.

An explicit distinction of an '*inner*' and an '*outer*' environment of the indicational space, like in $\text{indCA}^{(3,3)}$, enables loops of length 1 to length 36.

For ruleCIR[{11,23, 32, 42,53,61,72,81,92,102 }] the length is 24 and the length for the ruleCIR[{13,23, 33, 42,51,61,72,81,93,103 }] is 36. Hence the indicational domain of the CIR-rules opens up a field of self-referential complexity of at least 36 irreducible features that are enabling complex and intriguing modi of self-modification of systems based on the second-order distinctions of indication, *ruleCIR* and *ruleCIRT*.

This newly available complexity of indicational forms is challenging the trend of established interpretations and applications in many fields of research and practice.

What are the differences in the different approach to self-referentiality and self-modification?

The well established mathematical approach is based on formal languages, atomic signs and its applications in information processing.

Both approaches, the morphogrammatics and the indication based, are not founded, primarily, in signs but in *patterns* (morphe) for morphogrammatic morphoCAs and in *distinctions* for indicational indCAs. Both are not yet well understood and elaborated.

As a result, both approaches are very much relying on established mathematical methods and are therefore forced to balance between authentic realizations of the approaches and playful simulations of the new concepts by well known constructs.

The decision for an application of cellular automata as a tool for the modeling of the CI puts it into the framework of indicational cellular automata, *indCA*, that allows and restricts the attempts to study the *dynamics* of the complex forms of indication without the use of arbitrary constructions like the re-entry into the form as proposed by the *Laws of Form*.

Reflection and conscience in philosphy

Nathan Rotenstreich, Reflection and conscience, *Comment on `The fundamental structures of human reflexion'* by V. A. Lefebvre

"Yet, the reference to the mirror points out the limitations of the optical association or connotation. A mirror does not refer to itself. Another mirror is needed to present a copy of that which is visible in the first one. Moreover, the very introduction of the referential axis, `the self', cannot be con-grous with the optical allusion, because the self connotes a permanent entity or agent, and its `self

understanding' is implied in the very employment of the term `self'. It seems that one should attempt to describe the phenomenon of what goes by the name of reflection without calling upon the various actual or possible associations which can be related to the term.

In the first place, it has to be said that reflection is an act *and as such does not carry in itself any reference to a self as an agent or as a permanent entity (James, 1890) . It is an act even when it recurs as such. The reference to the self as a carrier or an agent can perhaps be based on an additional reflection which relates to the issue of the connection between the act and its performer. As such, it is in a way a construction but not a discernment of that which is inherent in the act as act."*

Slogans from a lost land

“We are led to consider in all seriousness the traditional image of the snake eating its own tail as the guiding image for autonomy as self-law and self-regulation.” Francisco Varela, 1979

http://www.sourceintegralis.org/VarelaUroboros.htm

Background reading

http://transhumanism.memristics.com/CA-Comparatistics/CA-Comparatistics.html

http://www.thinkartlab.com/pkl/media/Diamond%20 Calculus/Diamond%20 Calculus.html

http://memristors.memristics.com/MorphoSR/MorphoSub.html

Requisites

RuleSetCI

indrules

Procedures

Filters

Properties: Commutativity

The aim is to establish the analogon of the indCA commutativity with *IntegerDigits/FromDigits* for a commutativity in morphoCAs with *RuleTableFromkAryInd/ kAryFromRuleTableInd:*

```
RuleTableFromkAryInd[
 kAryFromRuleTableInd[
  RuleTableFromkAryInd
   [init]]                ⟺   RuleTableFromkAryInd
                                   [init]
                                    /. indrules
```

```
RuleTableFromkAryInd[
kAryFromRuleTableInd[RuleTableFromkAryInd[{0,1,1,0}]]]==
RuleTableFromkAryInd[{0,1,1,0}]/.indrules
```

True

```
RuleTableFromkAryInd[{1,1,0,1}]/.indrules
```

{8, 3, 6, 1}

```
kAryFromRuleTableInd[
DeleteDuplicates[Mod[ReLabel /@
Map[Flatten,ruleCI[{8,3,6,1}]/.Rule→List,1],2]]]
```

{1, 0, 0, 1}

```
RuleTableFromkAryInd[{1,0,0,1}]/.indrules
```

{8, 7, 6, 1}

RuleSpace, Groups, Equalities

1234 1634 5678 1278	5234 5634 5674 1678	1274

5234	=	5634
1234	=	1634
5678	=	1278

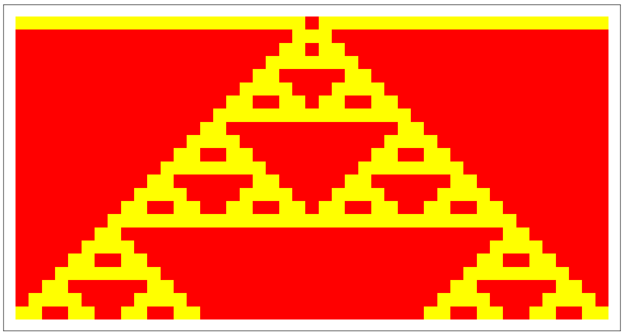
RuleTableCI

R1 ■ ■ ■ – ■ –	R2 ■ ■ □ – ■ –	R3 ■ □ □ – ■ –	R4 □ □ □ – ■ –
R5 ■ ■ ■ – □ –	R6 ■ ■ □ – □ –	R7 ■ □ □ – □ –	R8 □ □ □ – □ –

Self-modification of indCA

RuleTableFromkAryCI ⟷ kAryFromRuleTableCI
↖ ↗
RuleSetCI,
indIrules

```
ArrayPlot[CellularAutomaton[ruleCI[{1, 6, 7, 4}], {{1}, 0}, 22],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green}, ImageSize → {400, 200}]
```



```
Manipulate[
 CellularAutomaton[ruleCI[{1, 6, 7, 4}], {{1}, 0}, {{{n}}}], {n, 1, 16, 1}]
```

```
n
{0, 0, 0}
```

CellularAutomaton::nspecnl :
 Since rule specification ruleCI[{1, 6, 7, 4}] is not a List, it must be an integer or a pure Boolean function. ≫

(Debug) Out[22]=

```
ruleCI  CI1
{{8, 3, 2, 1}}
```

(Debug) Out[23]=

```
n
{{4, 7, 2, 5}, {0, 0, 0}}
```

(Debug) Out[24]=

```
n
ruleCI  CI1
{{8, 3, 2, 1}, {}}
```

```
{4, 7, 2, 5}                     {0, 0, 0}
{8, 7, 6, 5}                   {0, 0, 1, 0, 0}
{4, 7, 2, 5}                 {0, 0, 0, 0, 0, 0, 0}
{8, 7, 6, 5}               {0, 0, 1, 1, 1, 1, 1, 0, 0}
{4, 7, 2, 5}             {0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0}
{8, 7, 6, 5}           {0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0}
{4, 7, 2, 5}         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
{8, 7, 6, 5}       {0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0}
{4, 7, 2, 5}     {0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0}
{8, 7, 6, 5}   {0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0}
{4, 7, 2, 5} {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0}
```

Fixed-Points and palindromes for indCA

```
Table[RuleTableFromkAryInd[Mod[Flatten[
CellularAutomaton[ruleCI[{3,4,5,6}],{0,1,1,1},{{m}}]],2]]/.
indrules,{m,1,22,1}]
```

```
{{4, 7, 6, 5}, {8, 3, 2, 1}, {4, 7, 6, 5}, {8, 3, 2, 1}, {4, 7, 6, 5},
 {8, 3, 2, 1}, {4, 7, 6, 5}, {8, 3, 2, 1}, {4, 7, 6, 5}, {8, 3, 2, 1},
 {4, 7, 6, 5}, {8, 3, 2, 1}, {4, 7, 6, 5}, {8, 3, 2, 1}, {4, 7, 6, 5}, {8, 3, 2, 1},
 {4, 7, 6, 5}, {8, 3, 2, 1}, {4, 7, 6, 5}, {8, 3, 2, 1}, {4, 7, 6, 5}, {8, 3, 2, 1}}
```

```
DeleteDuplicates[Table[RuleTableFromkAryInd[Mod[Flatten[
CellularAutomaton[ruleCI[{5,6,3,4}],{0,1,1,1},{{m}}]],2]]/.
indrules,{m,1,11,1}]]
```

```
{{4, 7, 6, 5}, {8, 3, 2, 1}}
```

```
DeleteDuplicates[Table[RuleTableFromkAryInd[Mod[Flatten[
CellularAutomaton[ruleCI[{5,6,3,4}],{0,1,1,1},{{m}}]],2]]/.
indrules,{m,1,11,1}]/.filterCI]
```

```
{K, B}
```

FormDynInitCI

```
FormDynInitCI[ruleCI[{5, 6, 3, 4}], {0, 1, 1, 1}]
```

```
{K, B}
```

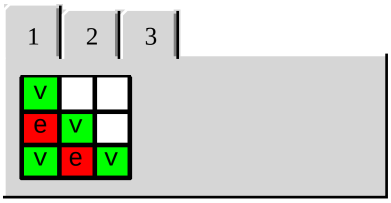
```
FormDynInitCI[ruleCI[{4, 3, 2, 1}], {0, 1, 1, 1}]
```

```
{B}
```

```
FormDynInitCI[ruleCI[{1, 6, 3, 8}], {0, 1, 1, 1}]
```

```
{G, A}
```

GAGA is an asymmetric palindrome

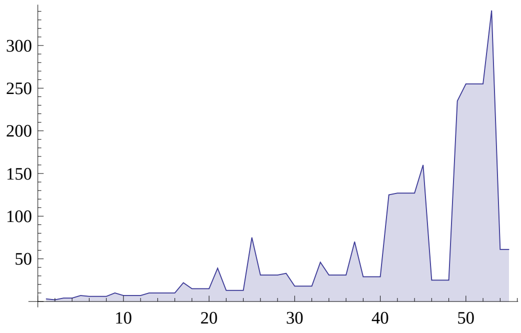


Result

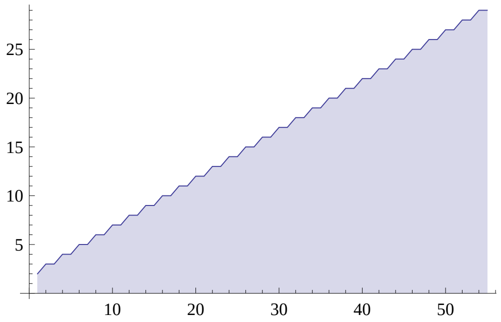
There are just two forms of fixed - points for the Calculus of Indication *indCA*: a self - loop of the form “AA” and an iteration of the form “AB”. Both fixed-points are morphic palindromes.

ListLinePlots

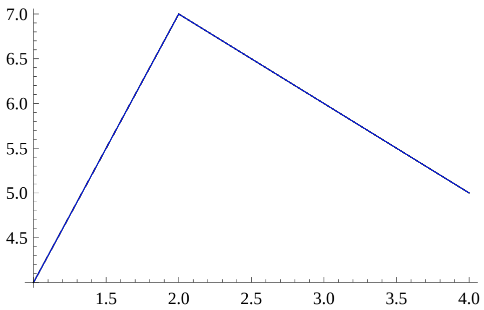
```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleCI[{5, 6, 3, 8}]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```



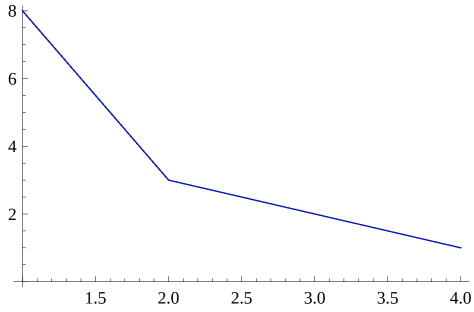
```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleCI[{8, 3, 2, 1}]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```



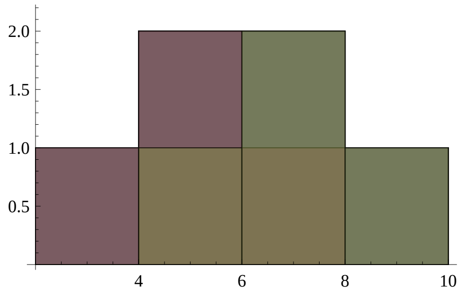
```
ListLinePlot[Table[RuleTableFromkAryInd[Flatten[
CellularAutomaton[ruleCI[{5,6,3,8}],{1,1,0,1},{{m}}]]]
/.indrules,{m,1,55,1}],PlotStyle->{Red,Green,Blue, Black},ImageSize→Medium]
```



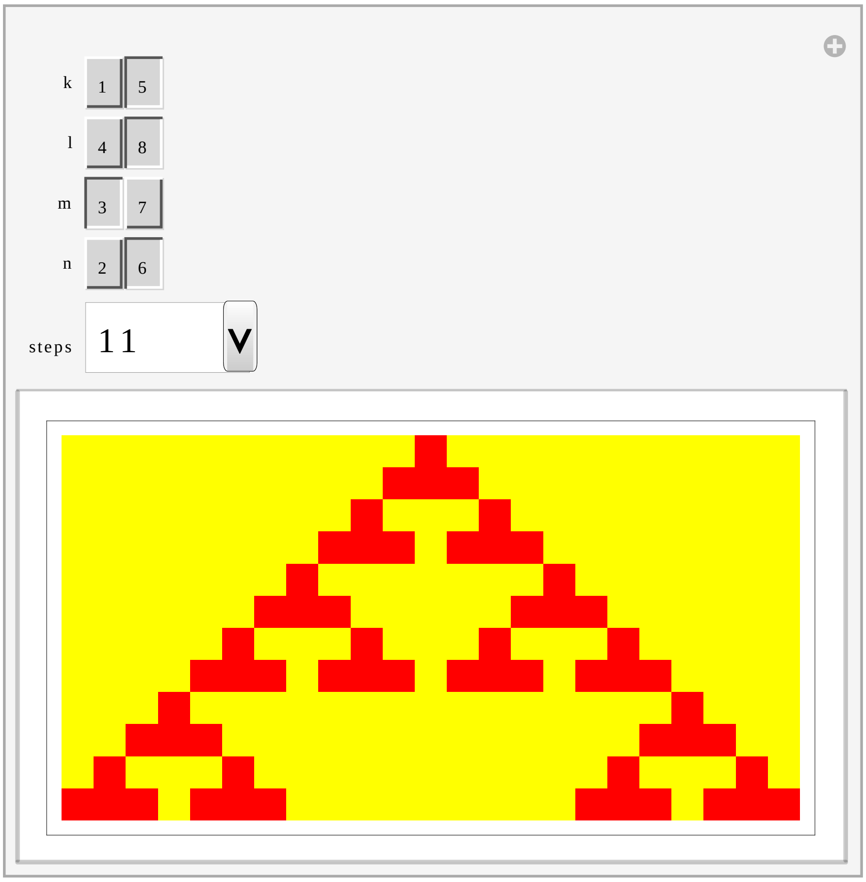
```
ListLinePlot[Table[RuleTableFromkAryInd[Flatten[
CellularAutomaton[ruleCI[{4,3,2,1}],{1,1,0,1},{{m}}]]]
/.indrules,{m,1,55,1}],PlotStyle->{Red,Green,Blue, Black},ImageSize→Medium]
```




```
Histogram[Table[RuleTableFromkAryInd[Flatten[
CellularAutomaton[ruleCI[{1,6,7,4}],{0,1,1,1},{{m}}]]]/.indrules,{m,1,16,1}]]
```



Subrule manipulator for ruleCI



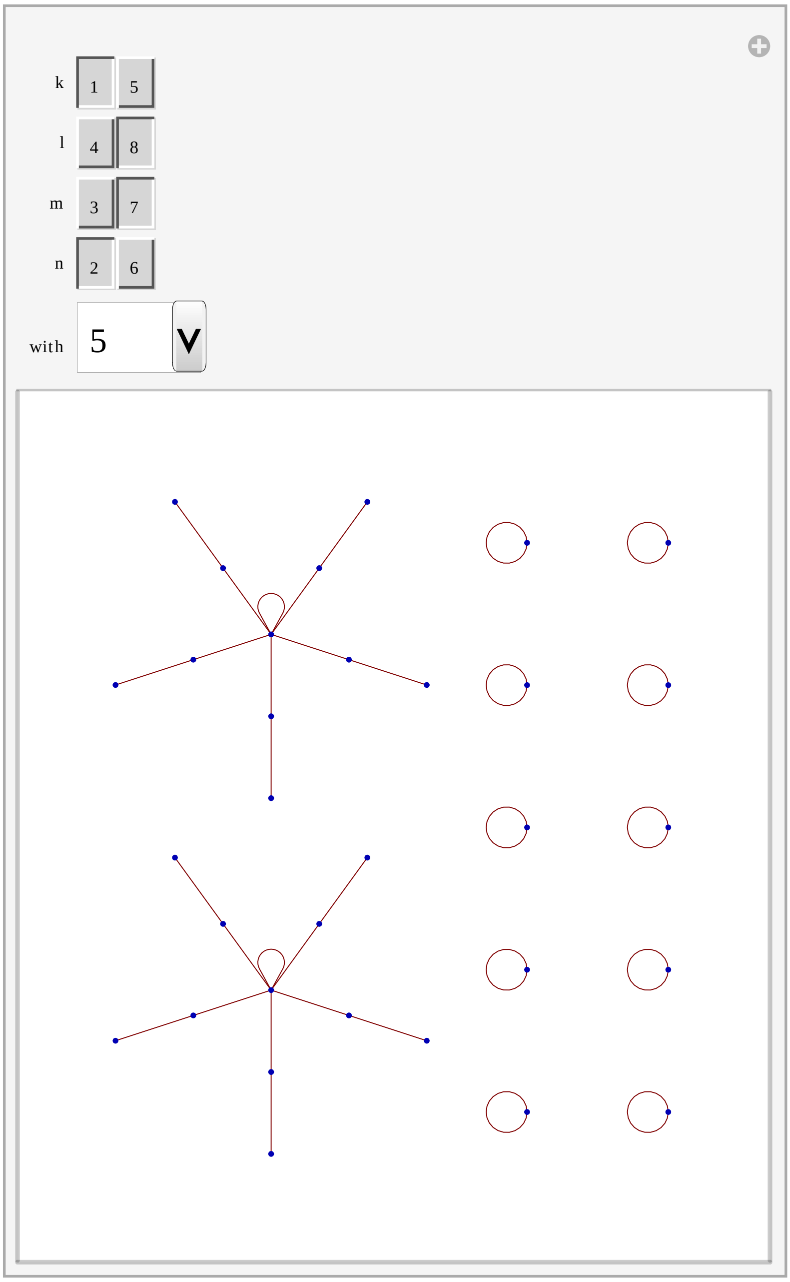
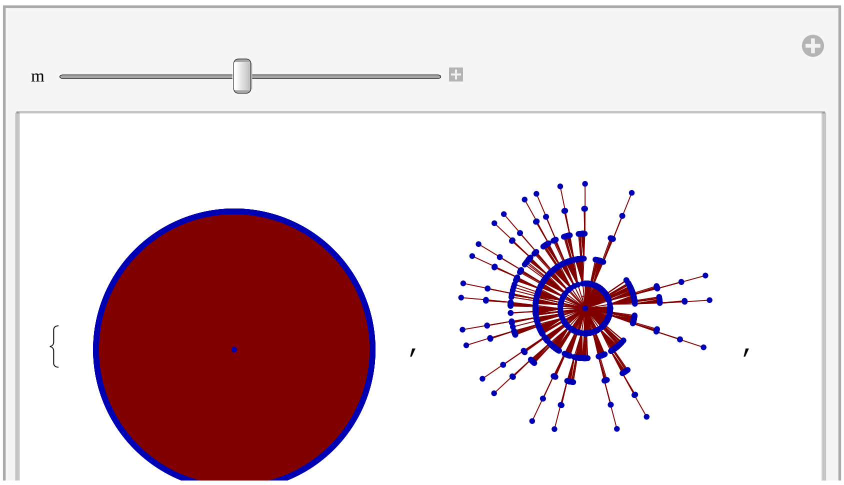
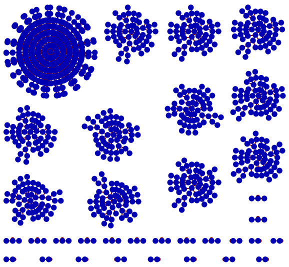
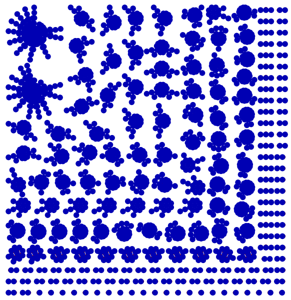
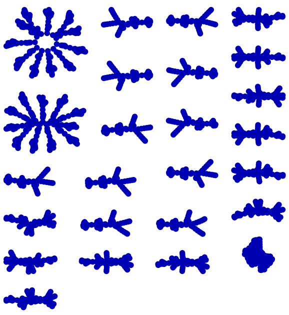
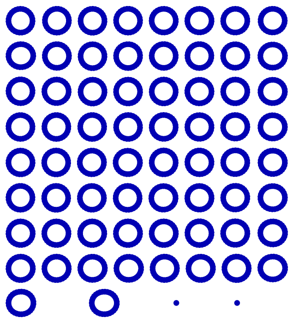
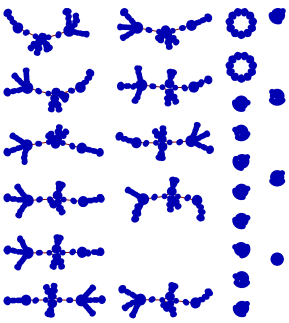
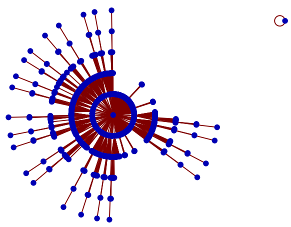


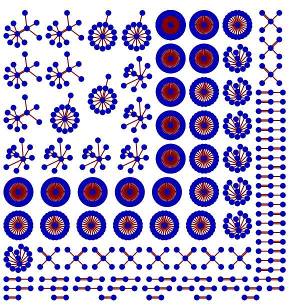
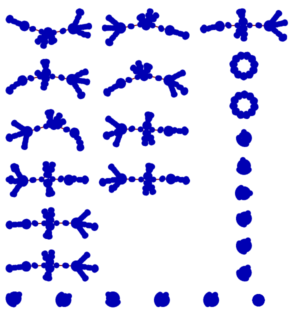
Table of indCA transitions

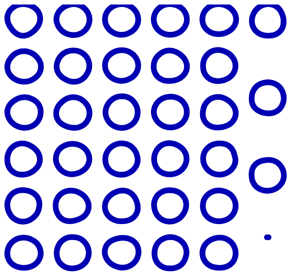
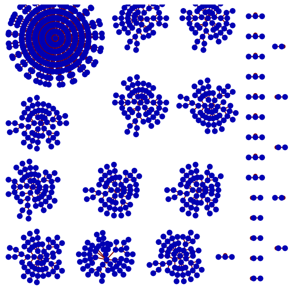
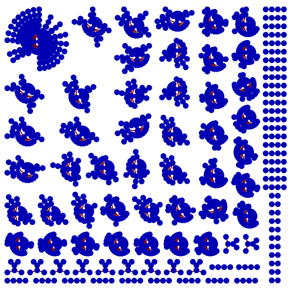
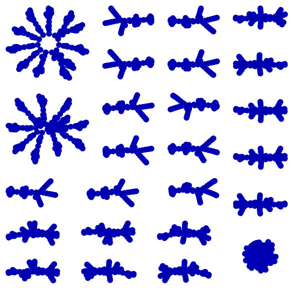
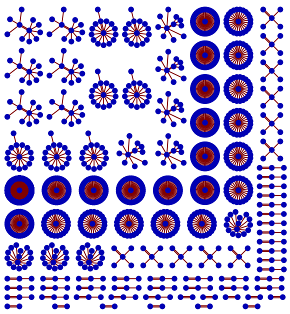
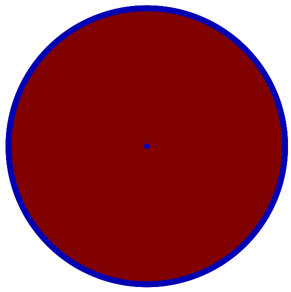


 ,  ,

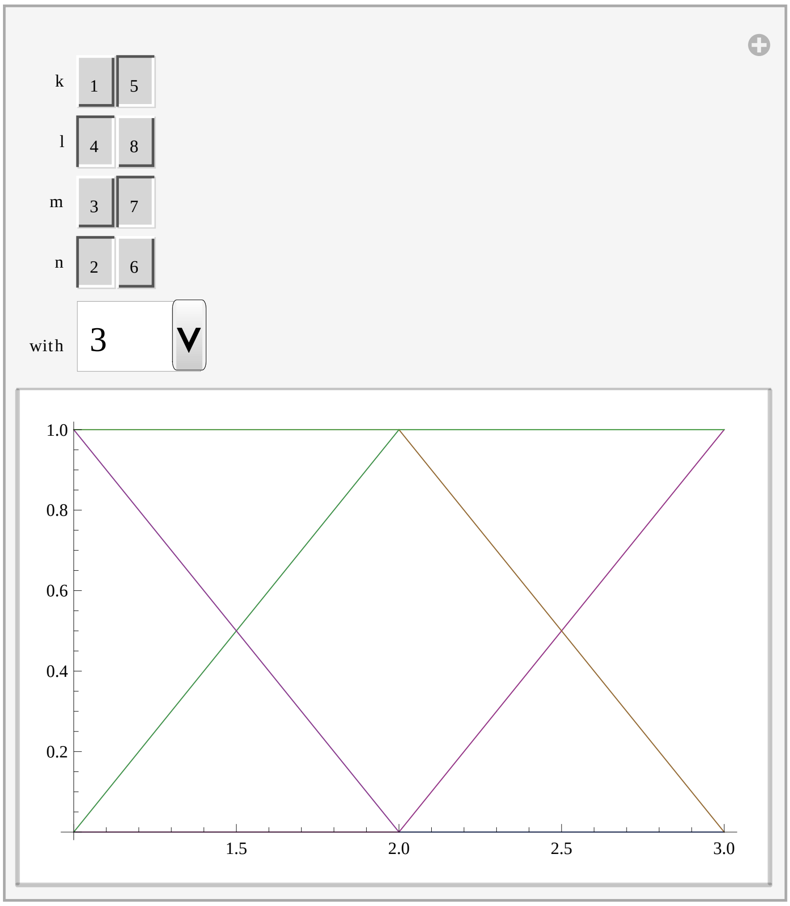
 ,  ,

 ,  ,

 ,  ,


,

,

,

,

,

}

ListLinePlots



Second-order calculus of forms

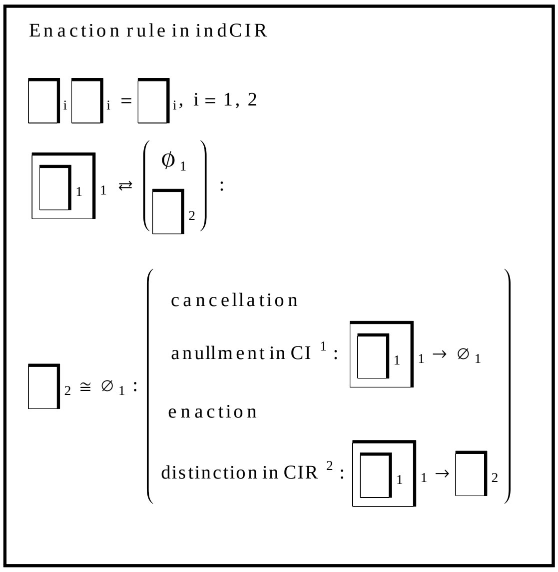
Motivation

"Günther has been alone in pointing out that other possible interpretations of many-valued is as a basis for a 'cybernetic ontology', that is, for systems capable of self-reference, and precisely one additional value, he claims, must be taken as time. I follow here Günther's suggestion that a third value might be taken as time.

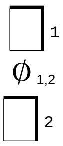
"But I have shown that this third value can be seen at a level deeper than logic, in the calculus of indication, where the form of self-reference is taken as a third value in itself, and in fact confused with time as a necessary component for its contemplation. In the extended calculus, self-reference, time, and reentry are seen as aspects of the same third value arising autonomously in the form of distinction."
F. Varela, Principles of Biological Autonomy, 1979, p.139

http : // www.thinkartlab.com/pkl/media/Diamond Calculus/Diamond Calculus.html

Reflectional laws and mappings for CIR

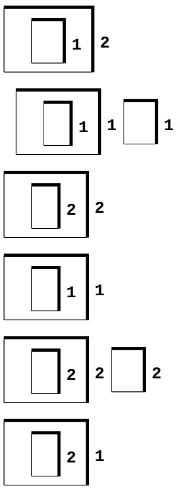


Homomorph mappings



:: cir[{11}] := {1, 1, 1} → 1,
:: cir[{22}] := {0, 0, 0} → 0,
:: cir[{33}] := {2, 2, 2} → 2

Heteromorph mappings



:: cir[{13}] := {1, 1, 1} → 2,
:: cir[{21}] := {0, 0, 0} → 1,
:: cir[{32}] := {2, 2, 2} → 0,
:: cir[{13}] := {1, 1, 1} → 0,
:: cir[{21}] := {0, 0, 0} → 2,
:: cir[{32}] := {2, 2, 2} → 1

Hence, a term like *ruleCIR[{11,22,32,42,52,61,71,83,92,102}]]* denotes a *distinction-structure* composed of the parts of the complex term of ten sub-terms. This structure is embedded into an *environment* denoted as *{2,2,2,2,0,1,2,0,0,0}*. Both together constitute an indicational constellation in a situation. Both indication components, the distinction-structure and the environment, are involved in computational transformations and especially into self-modifications.

http : // memristors.memristics.com/Complementary Calculi/Complementary Calculi.html

Example

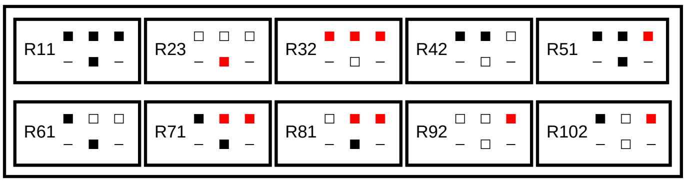
CIR[4] = ruleCIR[{11, 23, 32, 42, 51, 61, 71, 81, 92, 102}]

The complexion CIR[4] has a *diagrammatic* depiction or graphic model with Diagr(CIR[4]).

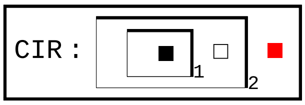
The head of the CIR-patterns is permutative. Thus the depiction shows just a standard form of the permutative pattern.

The head of R71 is the permutation of: [■ ■ ■], i.e. [■ ■ ■], [■ ■ ■].

Diagr (CIR[4]) =



Interpretation



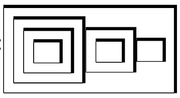
The mark "■" may represent the *'system'* of the indicational distinction $\square_1$ and 'outer' system of $\square_2$,

the mark "□" may represent the *'inner'* environment of the system $\square_2$ and 'outer' of system $\square_1$,

the mark "■" may represent the *'outer'* environment of the system $\square_2$ and system $\square_1$.

Therefore, CIR[4] defines a particular complex constellation of the distinction of system/inner/outer environments which is a minimal condition for reflective systems in general.

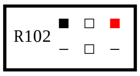
The architectonic distiction of system, inner and outer environment, is certainly not just an iteration of the classical distinction of system and environment like in the case for common CI constructions and their interpretations:



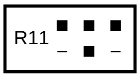
It is a traditional habit to mis-interpret triadic or n-adig structures in the context of polycontexturality just as systems consisting of different connected conceptual entities.

Reflectional interpretation of CIR distinctions

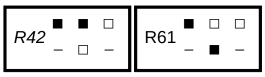
CIR distinctions are reflectional distinctions of the Calculus of Indication, hence CIR.

The pattern  might have a reflectional interpreted as a *trefoil knot*.

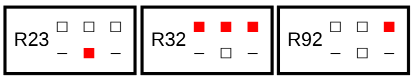
Differentiations of differences



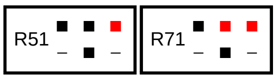
: reflects on the system only,



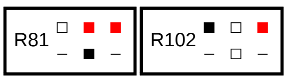
; reflects on the difference of system and inner environment, (symmetric)



: reflects on the difference of inner and outer environment (balanced),



: reflects on the difference of system and outer environment (symmetric),



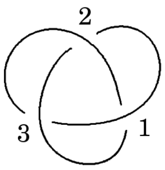
: reflects on the difference of system, inner and outer environment (asymmetric).

Table of reflectional differences

s	i	o	s/o	s/i	i/o	s/i/o
R11	–	–	R51	R42	R23	R81
–	–	–	R71	R61	R32	R102
–	–	–	–	–	R92	–

Trefoil interpretation of CIR distinctions

The knot-interpretation is emphasizing the dynamic aspect of complex structurations.

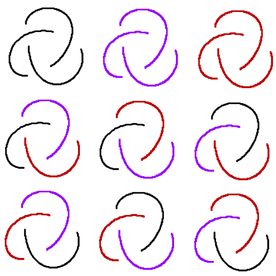


01U2O3U1O2U3

"Figure 14 shows all colorings of the trefoil knot by the three-element kei R_3 with operation table given by

▹	**1**	**2**	**3**
1	1	3	2
2	3	2	1
3	2	1	3

Figure 14. Kei colorings of the trefoil by $\mathbf{R}_3$.



http://www.ams.org/staff/jackson/fea-nelson.pdf

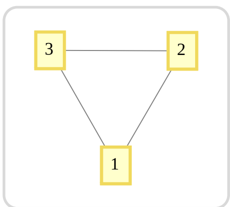
The operation table for Figure 14 has additionally many different interpretations. One is the interpretation of the table as the logical table for a full transjunction in a 3-contextural logic.
With a different coloring the table corresponds the transjunction:

⊗	**1**	**2**	**3**
1	1	3	2
2	3	2	1
3	2	1	3

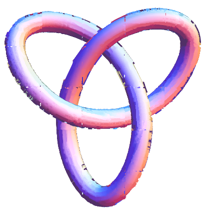
http://homepages.math.uic.edu/~kauffman/KFI.pdf
http://www.maths.manchester.ac.uk/~grant/knotschap4.pdf

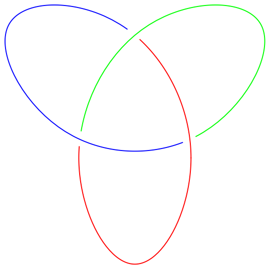
Basic example: Trefoil

```
KnotData["Trefoil"]
```



becomes

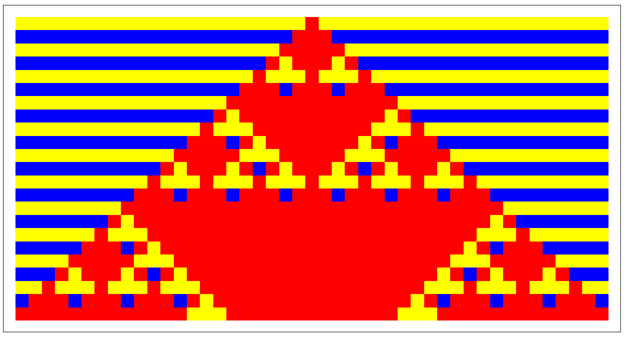




http : // memristors.memristics.com/Negation %20 Cycles/Gunther %20 Negation %20 Cycles.pdf

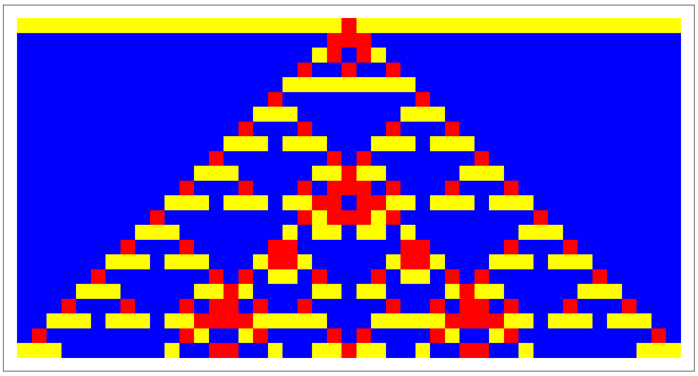
As a consequence, a depiction of the rule *ruleCIR[{11, 23, 32, 42, 51, 61, 71, 81, 92, 102}]* produced by an indicational CA, has an intriguing and ambigous structure. As a product of *ArrayPlot* it is just a planar graphical representation of the evolution of the rule. Concerning its internal structure it has to kept in mind that the diagram has a *knot-like* deep-structure between the the yellow, red and blue aspects of the whole diagram.

```
ruleCIR[{11, 23, 32, 42, 51, 61, 71, 81, 92, 102}]
```



The inter-tangling character of reflection in CIR-based cellular automata might be even more evident with the next example:

```
ruleCIR[{13, 23, 33, 42, 51, 61, 72, 81, 93, 103 }]
```



Requisites

RuleTableCIR

CIR-Rules

Procedures

indCIrules

filterCIR

Commutativity

RuleTableFromkAryCIR[kAryFromRuleTableCIR[RuleTableFromkAryCIR [init]]	⟺	RuleTableFromkAryCIR [init] /. indCIRrules

```
kAryFromRuleTableCIR[
DeleteDuplicates[
Map[Flatten,ruleCIR[{11,22,32,42,52,61,71,83,92,102}]]/.indCIRrules]]
```

{1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0}

```
RuleTableFromkAryCIR[{2,2,2,2,0,1,2,0,0,0}]/.indCIRrules
```

{23, 63, 43, 13, 52, 71, 83, 92, 102, 32}

```
RuleTableFromkAryCIR[
kAryFromRuleTableCIR[RuleTableFromkAryCIR[{2,2,2,2,0,1,2,0,0,0}]]]==
RuleTableFromkAryCIR[{2,2,2,2,0,1,2,0,0,0}]/.indCIRrules
```

True

First strategy: self-modification of environments

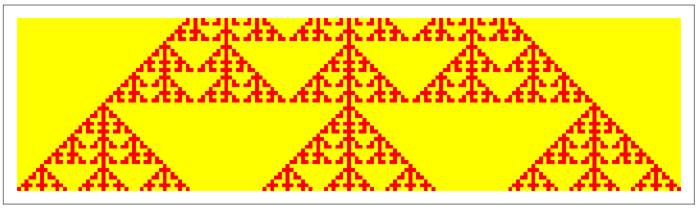
Self-reference of init2init for indCIR

```
Manipulate[ca=CellularAutomaton[
ruleCIR[{11,22,32,42,52,61,71,83,92,102}],
{{1},0}, {{{n}}}],{n,1,216,1}]
```

n

{1, 1, 1}

CellularAutomaton::nspecnl :
Since rule specification ruleCIR[{11, 22, 32, 42, 52, 61, 71, 83, 92, 102}] is not a List, it must be an integer or a pure Boolean function. ≫



Second strategy: self-modification of rules

indCIR rule approach

Fixed-Points and palindromes

RuleTableFromkAryCIR ⟷ kAryFromRuleTableCIR
↖ ↗
RuleSetCIR
indCIRrules

```
Manipulate[
 CellularAutomaton[
  ruleCIR[{11, 22, 32, 42, 52, 61, 71, 83, 92, 102}], {{1}, 0}, {{{n}}}],
 {n, 1, 60, 1}]
```

n

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

CellularAutomaton::nspecnl :
Since rule specification ruleCIR[{11, 22, 32, 42, 52, 61, 71, 83, 92, 102}] is not a List, it must be an integer or a pure Boolean function. ≫

n

```
{{22, 62, 41, 11, 51, 71, 81, 92, 102, 32},
 {1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1}}
```

fixedpoint set CIR4

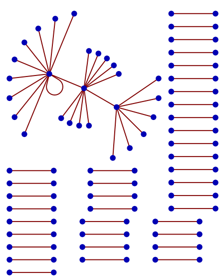
```
{{21, 62, 42, 11, 52, 72, 82, 92, 103, 32},
 {21, 61, 41, 11, 51, 73, 83, 92, 102, 32},
 {22, 61, 41, 11, 51, 71, 81, 92, 103, 31},
 {22, 62, 41, 11, 51, 71, 82, 92, 102, 32},
 {23, 61, 42, 11, 51, 72, 81, 93, 103, 33},
 {21, 62, 42, 12, 52, 72, 82, 91, 102, 32},
 {21, 61, 43, 13, 53, 73, 81, 91, 101, 31},
 {21, 61, 41, 12, 52, 71, 81, 91, 101, 31},
 {21, 61, 42, 11, 51, 72, 81, 91, 101, 31},
 {21, 62, 42, 12, 52, 72, 82, 91, 101, 31},
 {22, 61, 43, 13, 53, 73, 81, 92, 101, 31},
 {22, 62, 41, 12, 52, 71, 82, 92, 102, 32},
 {23, 61, 41, 11, 51, 71, 81, 93, 103, 33},
 {21, 61, 41, 11, 51, 71, 81, 91, 102, 32},
 {22, 61, 41, 11, 51, 71, 81, 92, 101, 31}}
```

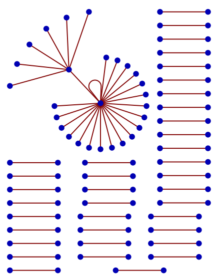
length of fixedpoint set CIR4

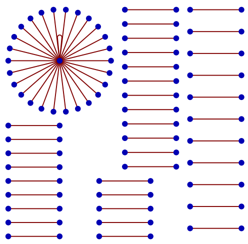
```
15
```

CIR5

```
{{21, 62, 42, 11, 51, 71, 81, 92, 103, 32},
 {21, 61, 41, 11, 51, 71, 81, 91, 102, 31},
 {21, 61, 41, 11, 51, 71, 81, 91, 101, 31}}
```







Palindromes

```
FormDynInitCIR[
ruleCIR[{13,23, 33, 42,51,61,72,81,93,103 }],{1,1,1,0,1,1,1,0,1,1}]
```

```
{{23, 63, 42, 12, 52, 73, 82, 92, 102, 33},
 {23, 61, 43, 13, 53, 73, 83, 93, 103, 31}, {21, 62, 42, 13, 53, 73, 83, 93, 102, 32},
 {21, 61, 43, 11, 53, 73, 83, 91, 103, 31}, {23, 61, 41, 12, 52, 73, 82, 92, 101, 31},
 {21, 61, 42, 11, 53, 73, 83, 91, 102, 31}, {23, 62, 42, 13, 52, 73, 82, 93, 102, 32},
 {23, 63, 43, 13, 51, 73, 81, 93, 103, 33}, {23, 63, 43, 12, 52, 71, 82, 92, 103, 33},
 {23, 63, 41, 13, 51, 71, 81, 93, 101, 33}, {23, 62, 42, 11, 51, 73, 81, 91, 102, 32},
 {23, 63, 41, 12, 51, 71, 81, 92, 101, 33}, {23, 62, 43, 12, 52, 73, 82, 92, 103, 32}}
```

```
DeleteDuplicates[Table[RuleTableFromkAryCIR[Mod[Flatten[
CellularAutomaton[ruleCIR[{22,61,42,11,52,71,83,92,102,33}],
{0,1,0,1,0,1,2,0,0,0},{{m}}]],3]]/.
indCIRrules,{m,1,22,1}]]
```

```
{{21, 61, 42, 11, 52, 72, 82, 92, 102, 32}, {22, 62, 42, 11, 51, 72, 82, 92, 102, 31},
 {21, 62, 41, 12, 52, 71, 82, 92, 101, 31}, {22, 62, 41, 11, 51, 71, 81, 91, 102, 31},
 {21, 61, 42, 11, 51, 71, 81, 92, 102, 31}, {21, 62, 42, 12, 51, 71, 82, 91, 101, 32}}
```

```
DeleteDuplicates[Table[RuleTableFromkAryCIR[Mod[Flatten[
CellularAutomaton[ruleCIR[{22,61,42,11,52,71,83,92,102,33}],
{0,1,0,1,0,1,2,0,0,0},{{m}}]],3]]/.
indCIRrules,{m,1,22,1}]]/.filterCIR
```

```
{A, B, C, D, E, F}
```

```
FormDynInitCIR[
ruleCIR[{22,61,42,11,52,71,83,92,102,33}],
{0,1,0,1,0,1,2,0,0,0}]
```

```
{A, B, C, D, E, F}
```

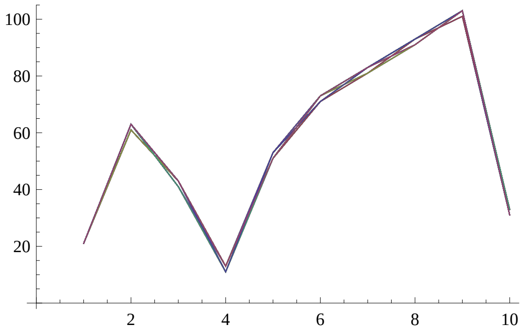
(Debug) Out[42]=

fixedpoint set CIR1

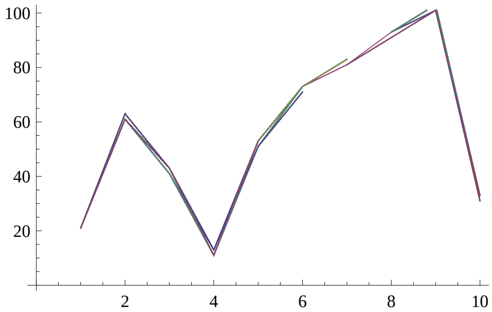
{A, B, C, D, E, F}

ListLinePlots and transitions

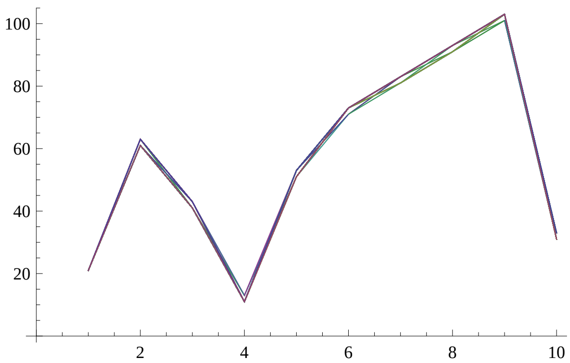
```
ListLinePlot[Table[RuleTableFromkAryCIR[ReLabel[Flatten[
CellularAutomaton[
ruleCIR[{22,61,42,11,53,71,83,92,101,32}],{1,0,2,1,0,1,2,0,0,1},{{m}}]]],5]
/.indCIRrules,{m,1,163,1}]]
```



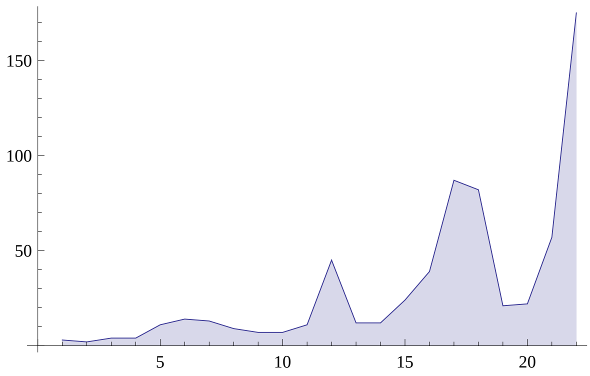

```
ListLinePlot[Table[RuleTableFromkAryCIR[ReLabel[Flatten[
CellularAutomaton[
ruleCIR[{22,61,42,11,53,71,83,92,101,32}],{1,0,2,2,1,1,2,0,1,2},{{m}}]]],5]
/.indCIRrules,{m,1,163,1}]]
```



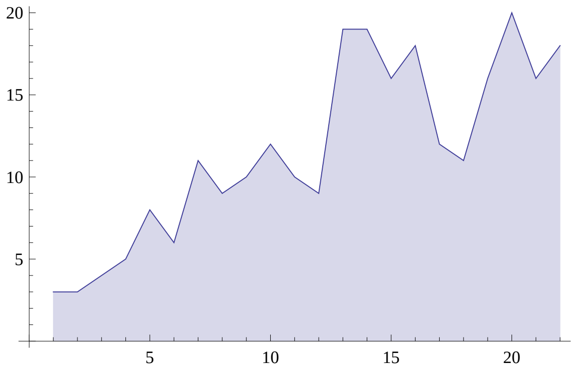

```
ListLinePlot[Table[RuleTableFromkAryCIR[ReLabel[Flatten[
CellularAutomaton[
ruleCIR[{13,23, 33, 42,51,61,72,81,93,103 }],{0,1,0,1,0,1,2,0,0,0},{{m}}]]],5]
/.indCIRrules,{m,1,163,1}]]
```



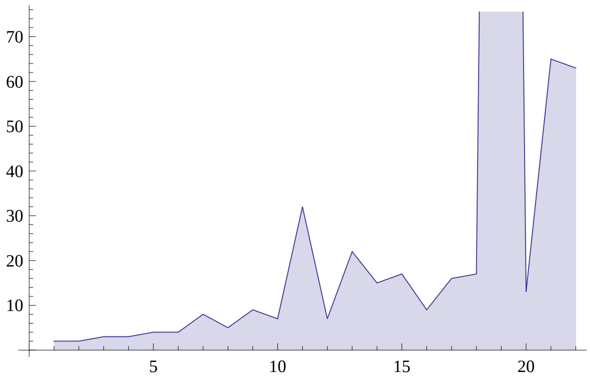
```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleCIR[{13, 23, 33, 42, 51, 61, 72, 81, 93, 103 }]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 22}], Filling → Axis]
```



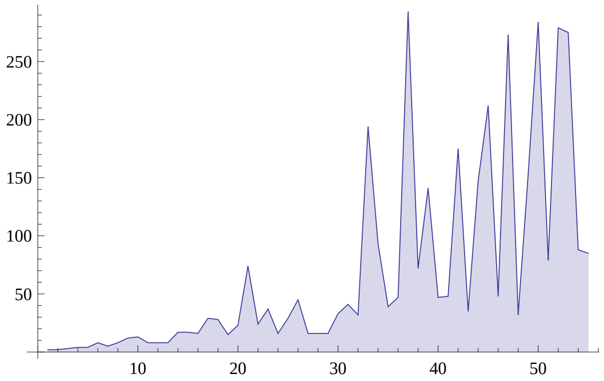

```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleCIR[{13, 23, 33, 43, 52, 61, 71, 83, 92, 102 }]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 22}], Filling → Axis]
```




```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleCIR[{11, 22, 32, 42, 52, 61, 71, 83, 92, 102}]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 22}], Filling → Axis]
```

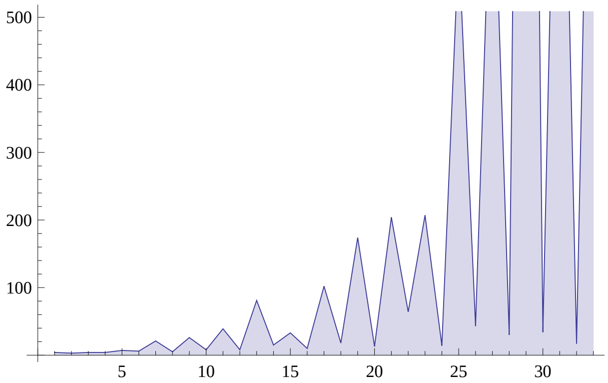


```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleCIR[{11, 23,  32,  42, 51, 61, 71, 81, 92, 102 }]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```

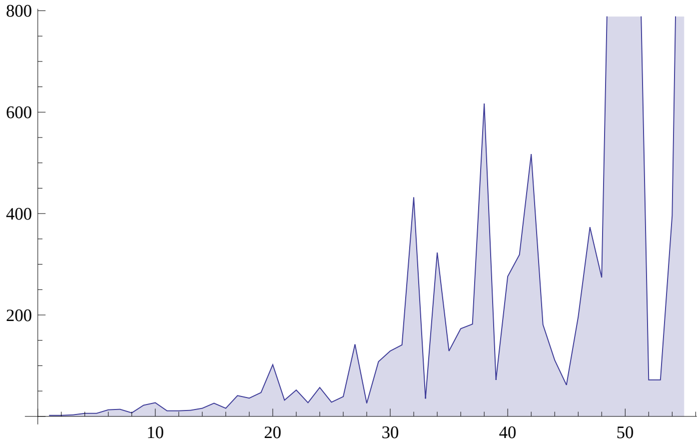



```
ruleCIR[{12, 23,  32,  43, 51, 62, 73, 81, 92, 103 }]
```

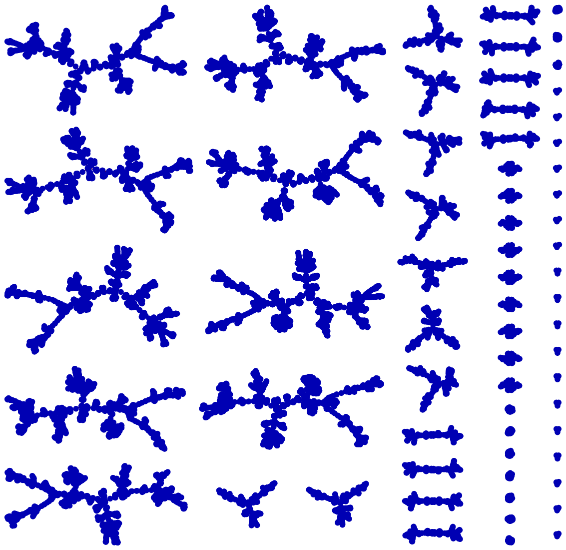
```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleCIR[{12, 23,  32,  43, 51, 62, 73, 81, 92, 103 }]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 33}], Filling → Axis]
```



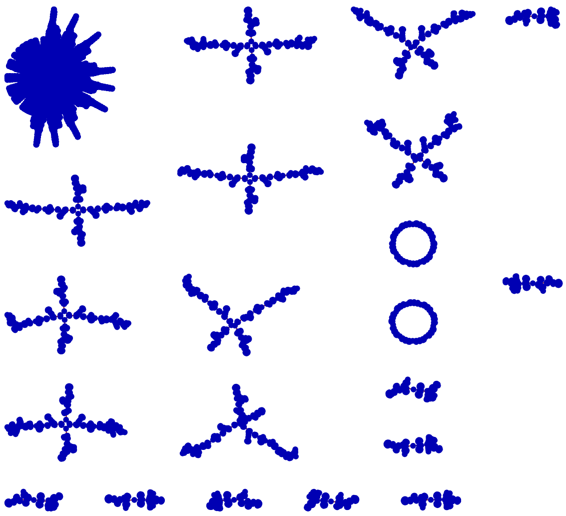

```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleCIR[{11, 23,  32,  42, 53, 61, 72, 81, 92, 102 }]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```



```
GraphPlot[
 # -> CellularAutomaton[ruleCIR[{12, 23, 32, 43, 51, 62, 73, 81, 92, 103 }], #] &/@
  Tuples[{0, 1, 2}, 9]]
```



```
GraphPlot[# -> CellularAutomaton[
      ruleCIR[{13, 23, 33, 42, 51, 61, 72, 81, 93, 103 }], #] &/@
  Tuples[{0, 1, 2}, 9]]
```



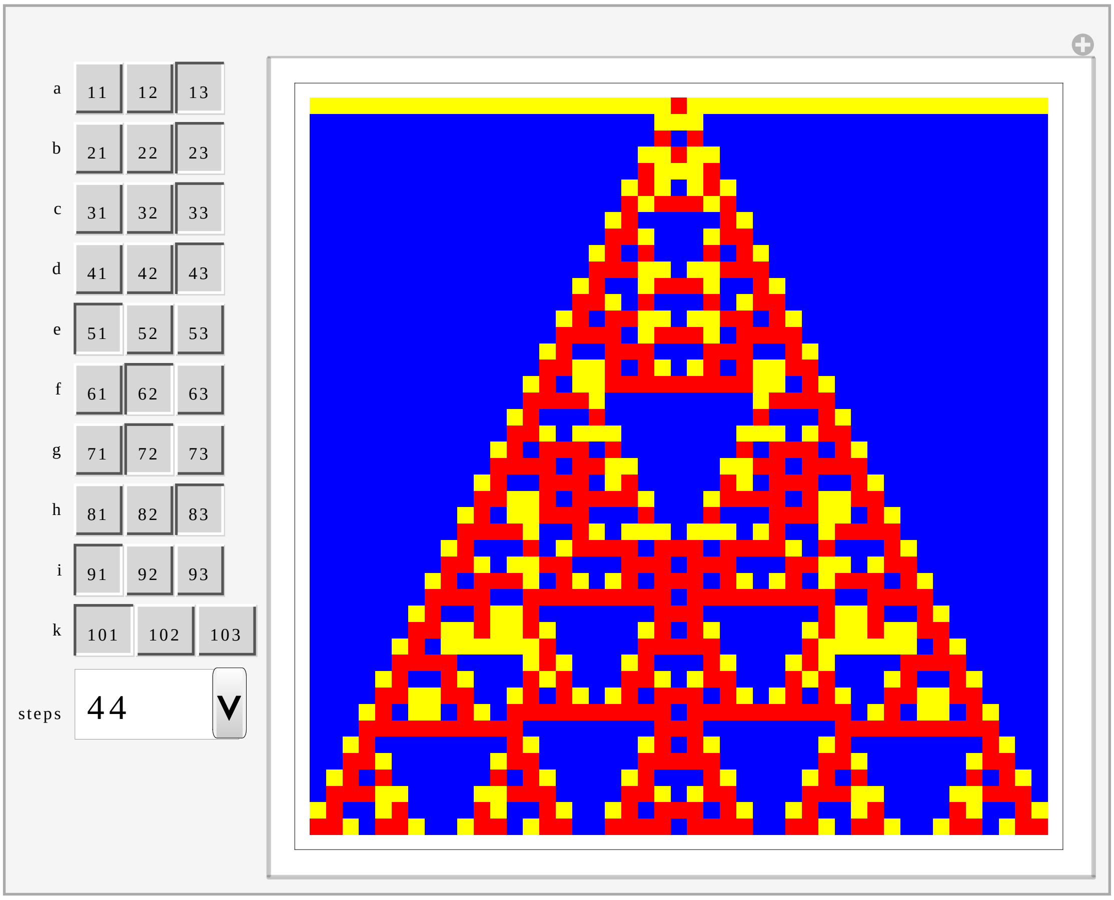
Subrules for indCIR

The sub - rule approach has the convincing practical advantage that the particular constellations can be interac-tively chosen out of the combinatorial set of possibilities. For *ruleCIR* there are 3^10 = 59049 possible constellations.

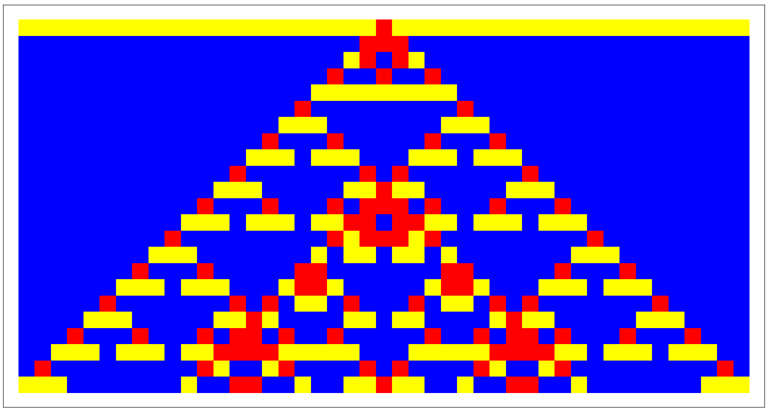
Other approaches are forced to implement the whole set of 59049 instances to cover the possible constellations of the single-reflectional *ruleCIR*.

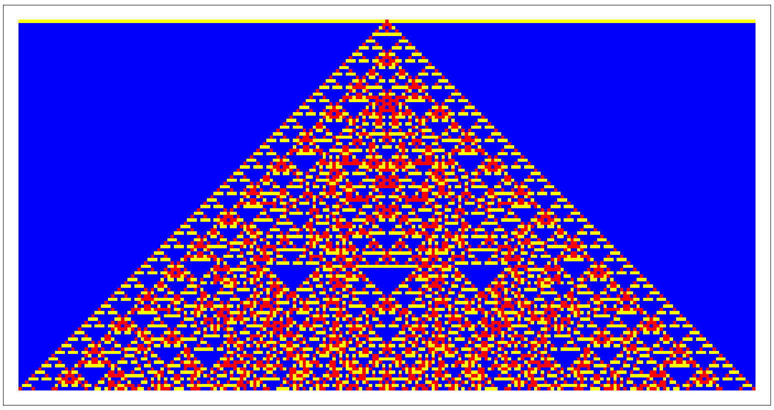
With the *double-reflectional* indcational CAs, the amount of functions is exactly: 20^4 = 160000. With the sub-rule approach all those functions are easily accessible.

Sub-rules for indCIR

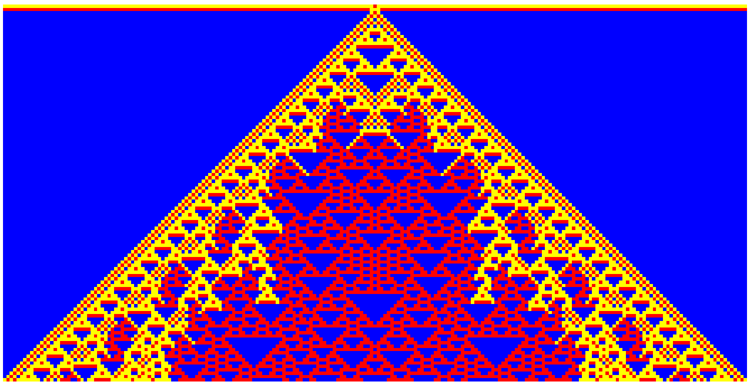


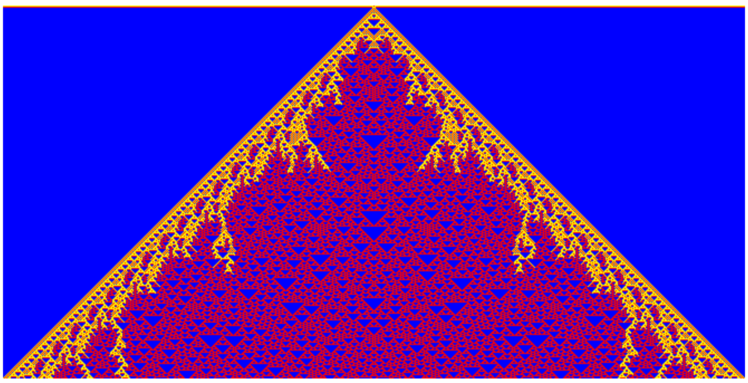

```
ruleCIR[{13, 23, 33, 42, 51, 61, 72, 81, 93, 103 }]
```

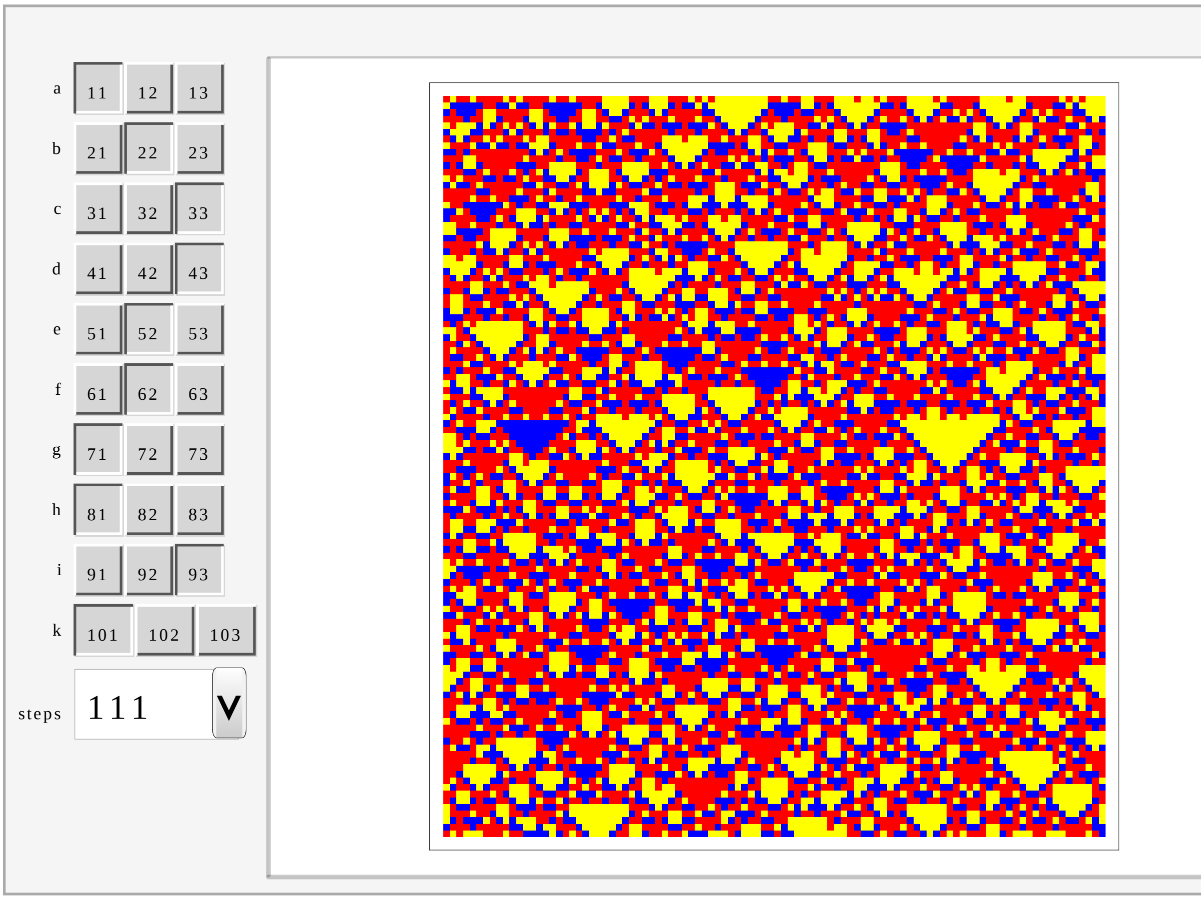




ruleCIR[{**13, 21, 33, 43, 51, 62, 72, 82, 92, 101** }]

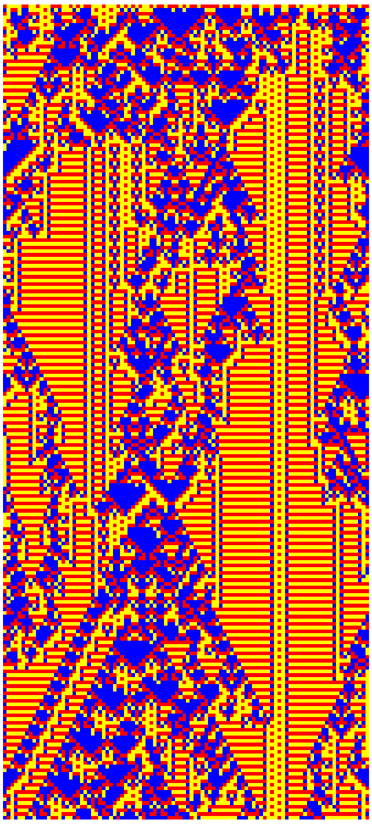




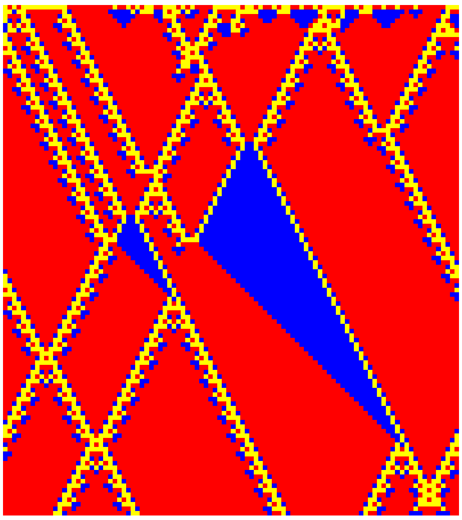


Random

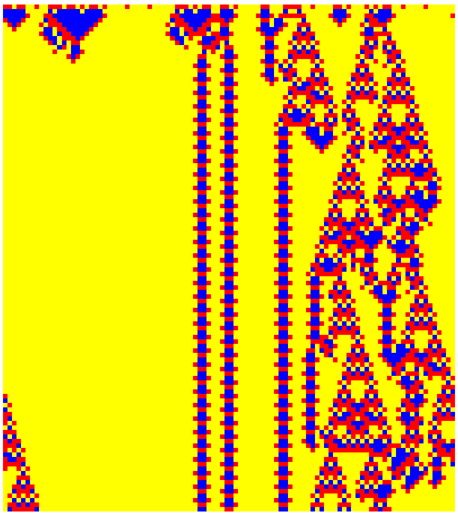
```
ruleCIR[{12, 21, 33, 43, 53, 62, 71, 83, 91, 102}]
```



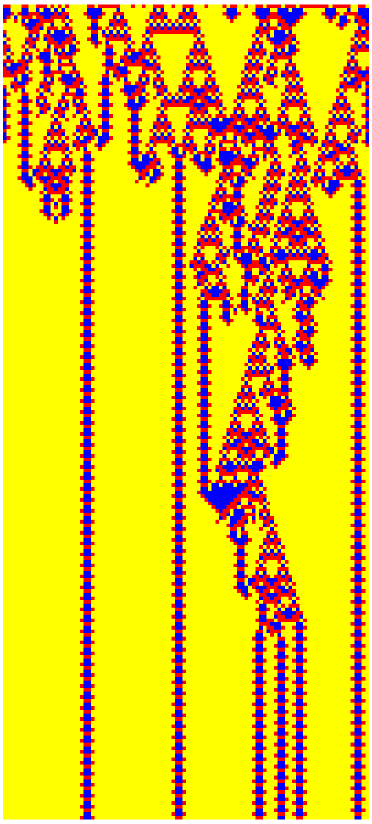
ruleCIR[{**11, 21, 33, 43, 51, 62, 71, 83, 91, 102**}]



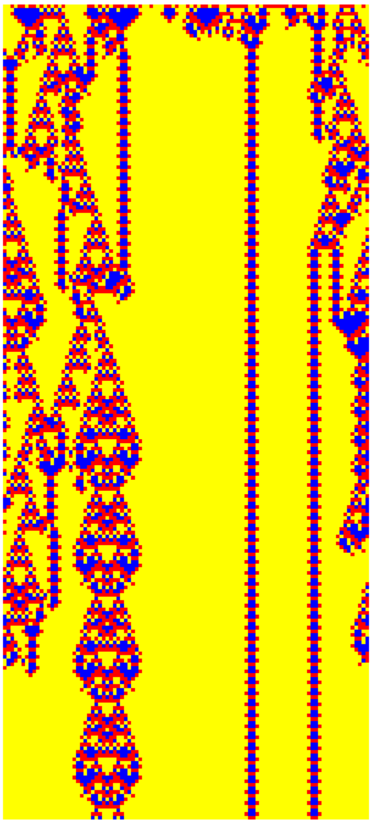
ruleCIR[{**12, 22, 33, 43, 51, 62, 71, 83, 91, 102**}]



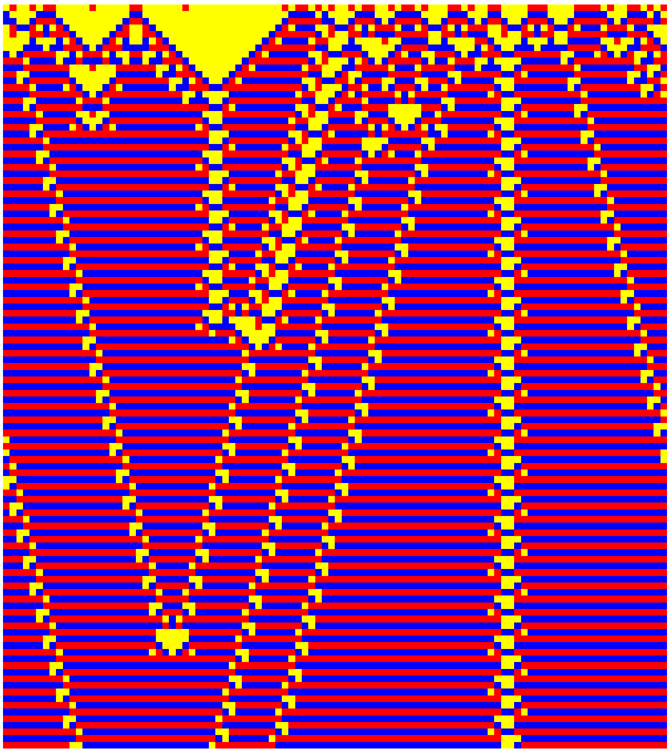
ruleCIR[{**11, 22, 33, 43, 51, 62, 71, 83, 91, 102**}]



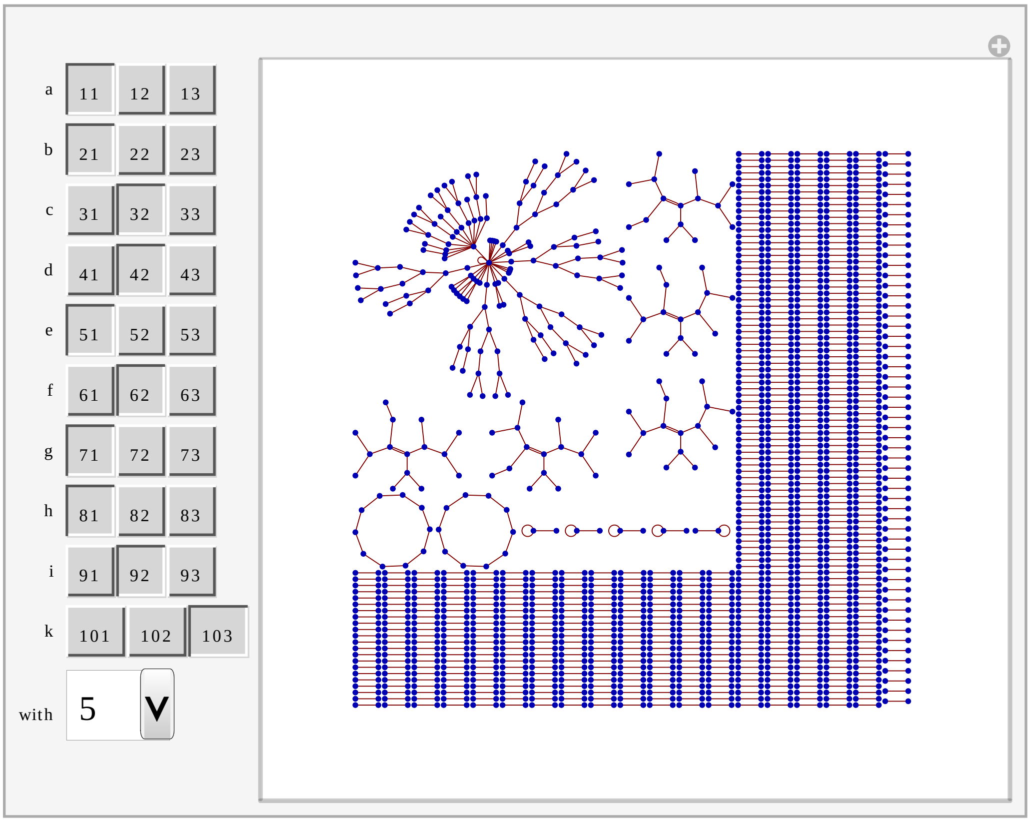
ruleCIR[{**11, 22, 33, 43, 51, 62, 71, 83, 91, 102**}]

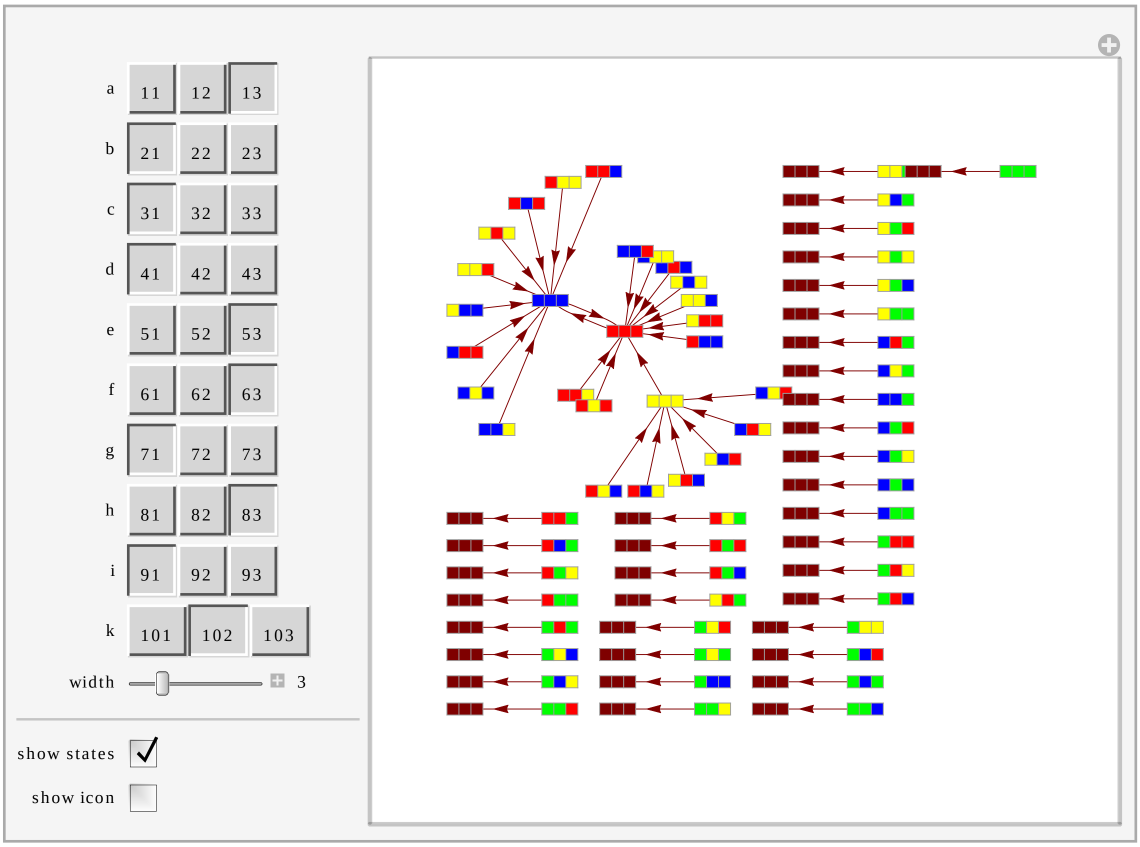


ruleCIR[{**13, 22, 31, 43, 52, 62, 72, 81, 93, 101**}]

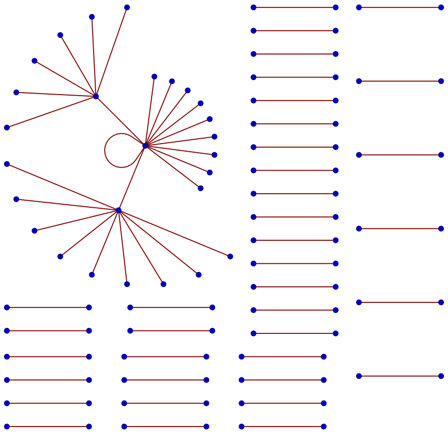


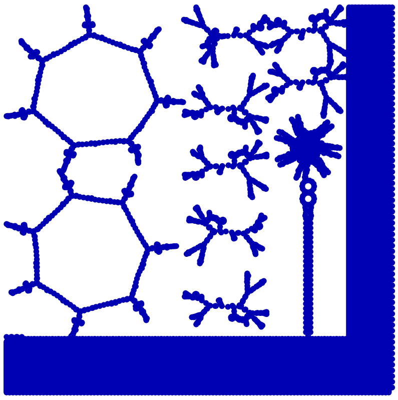
Subrules for CIR-transitions

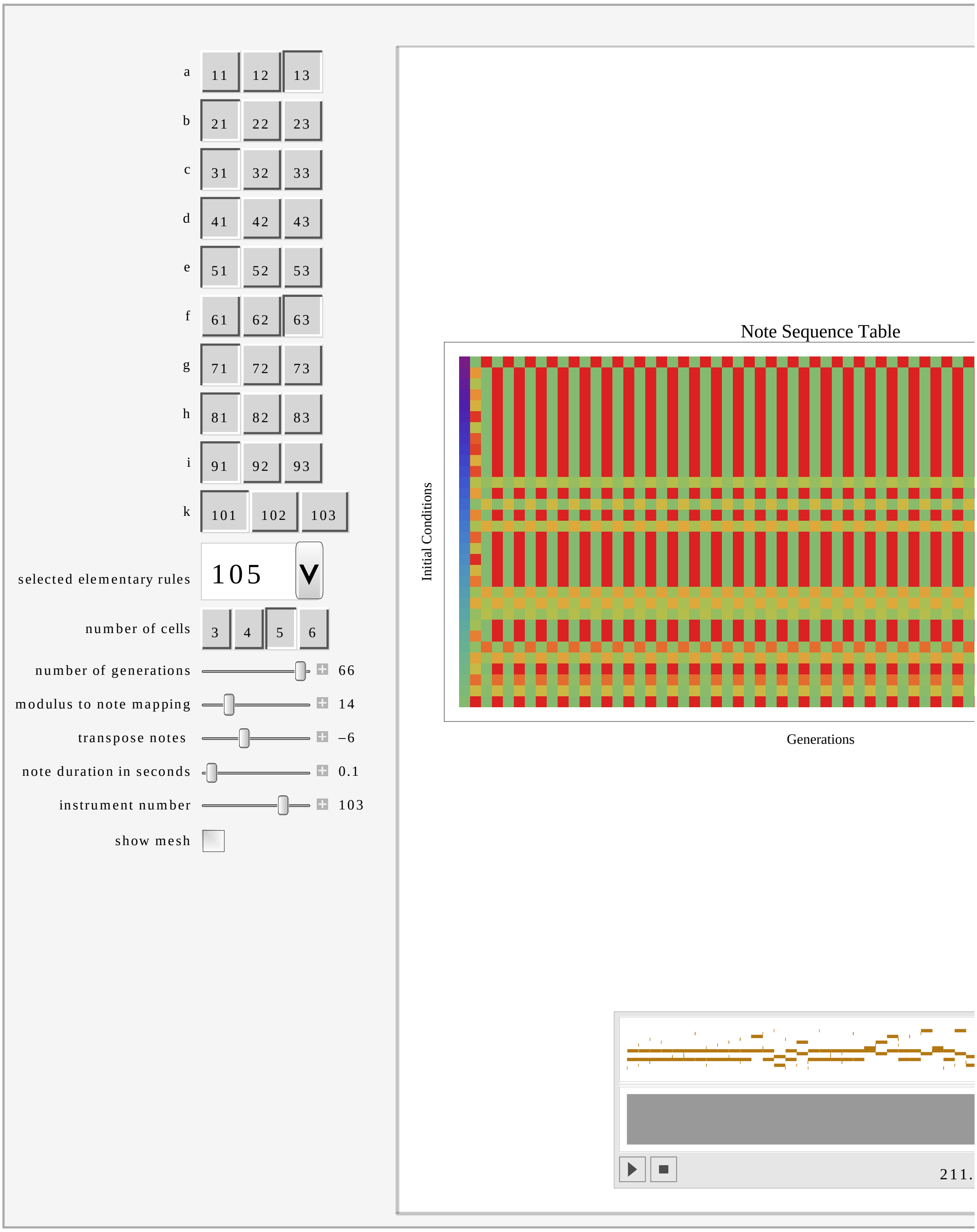




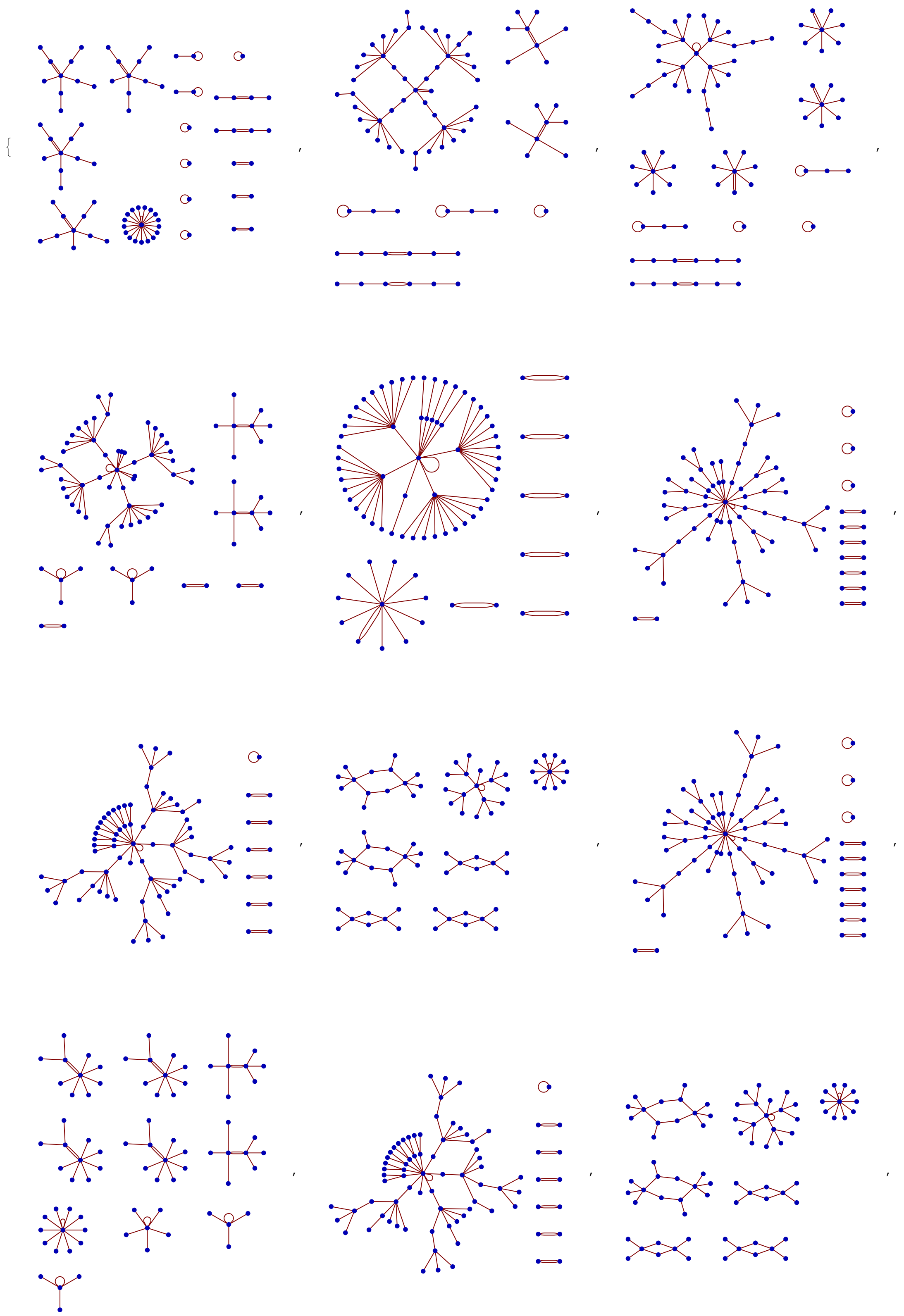

```
ruleCIR[{13, 23, 33, 42, 51, 61, 72, 81, 93, 103}]
```

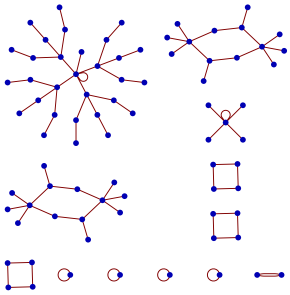
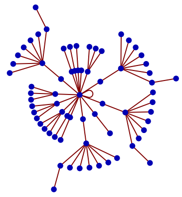
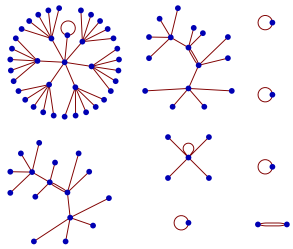


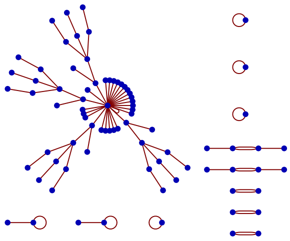
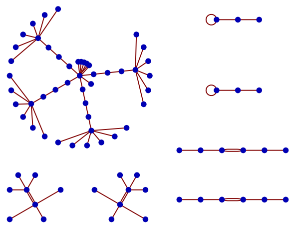
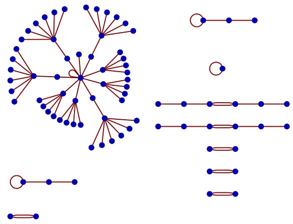


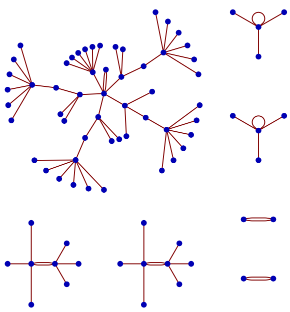
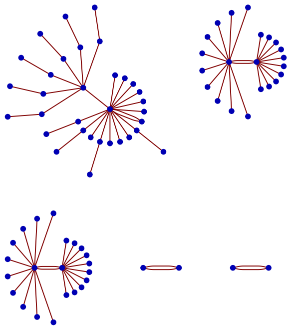
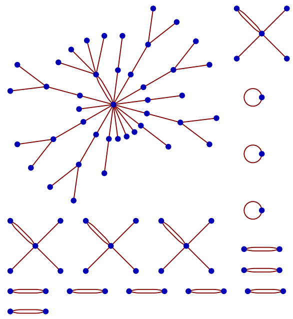


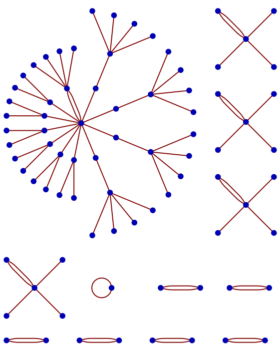
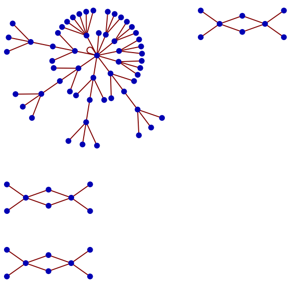
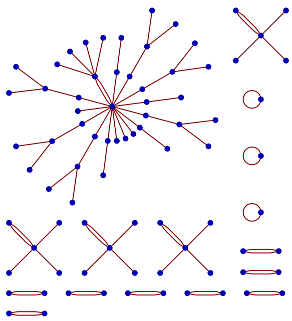
Transition table for some ruleCIR

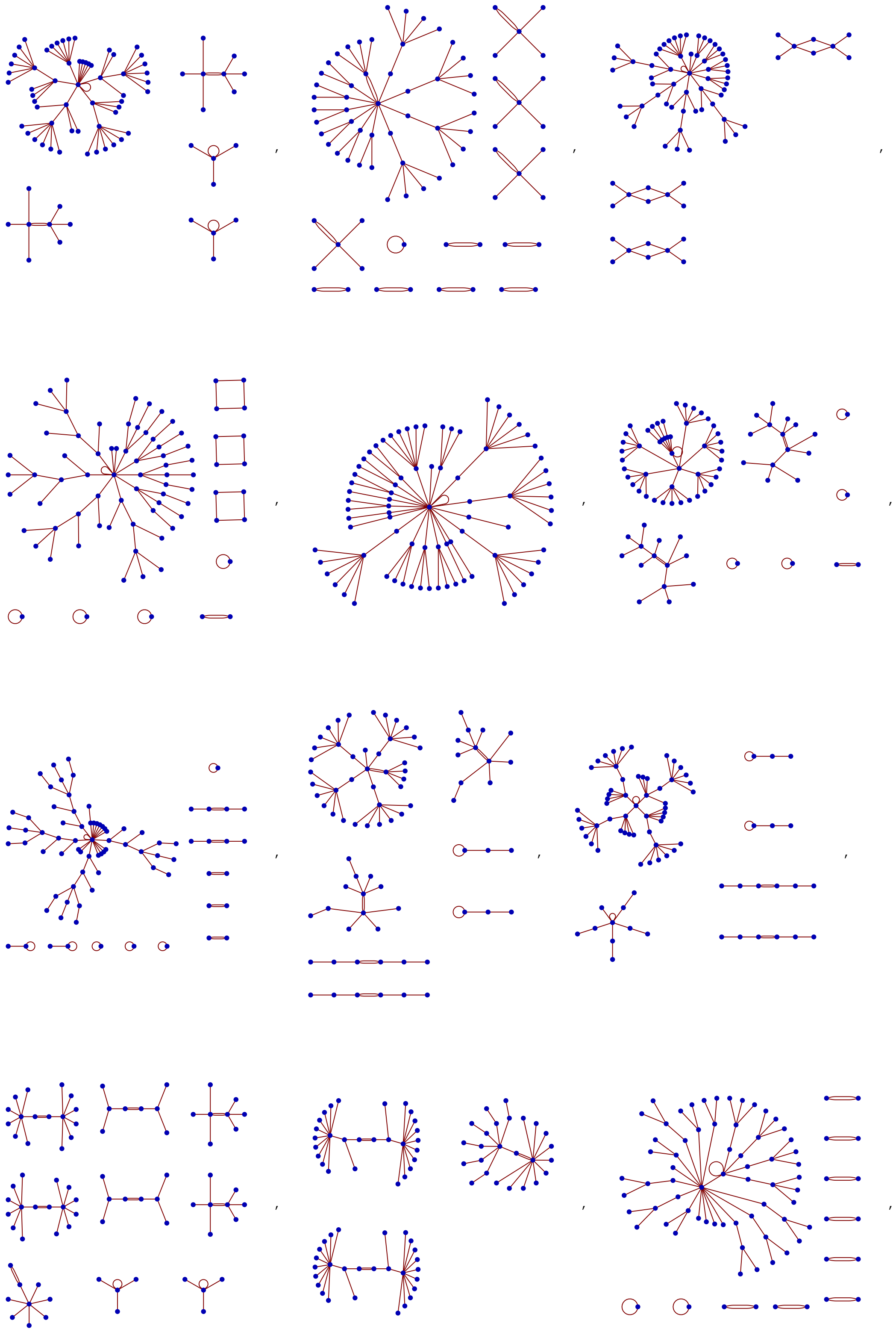


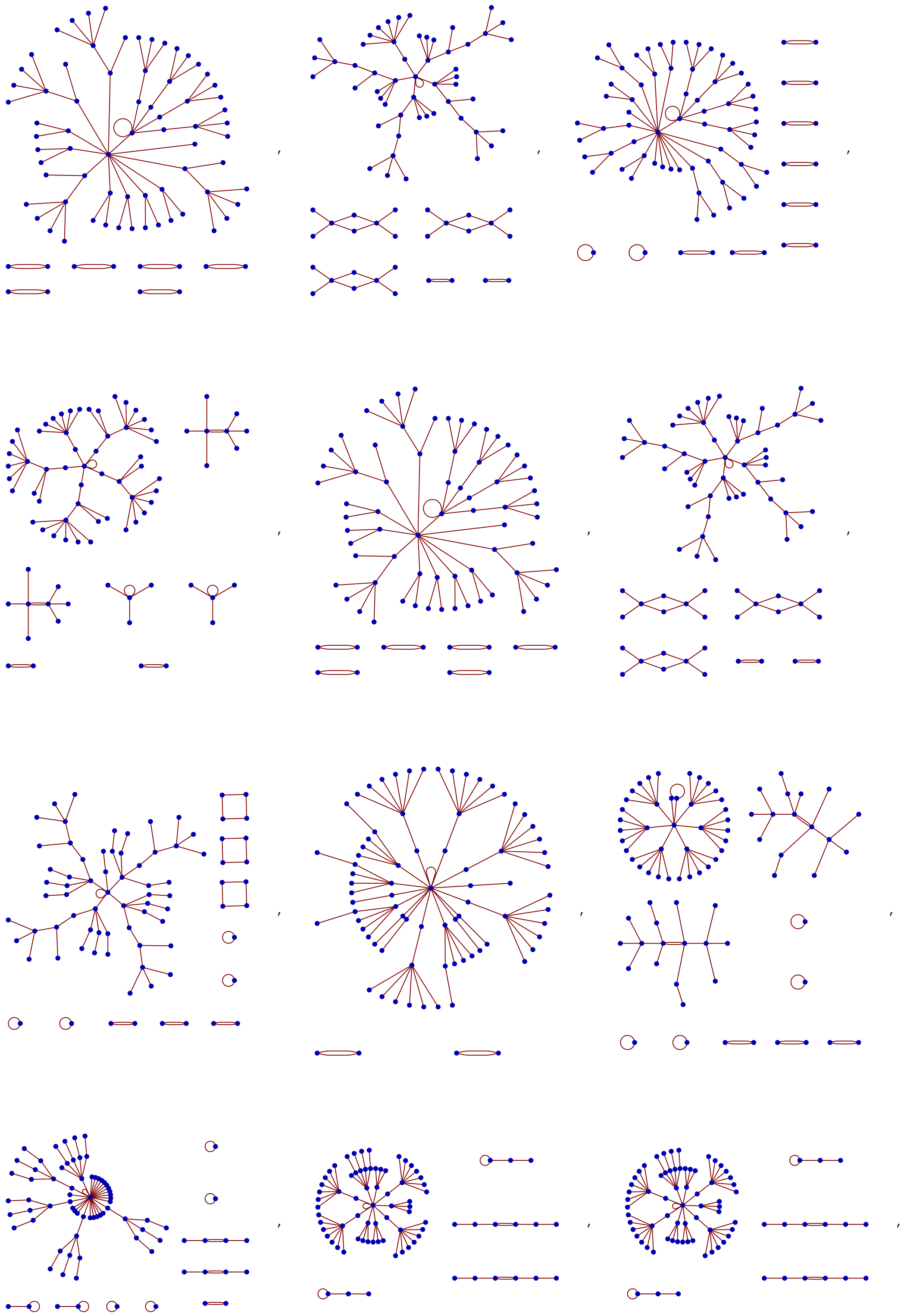

,

,

,

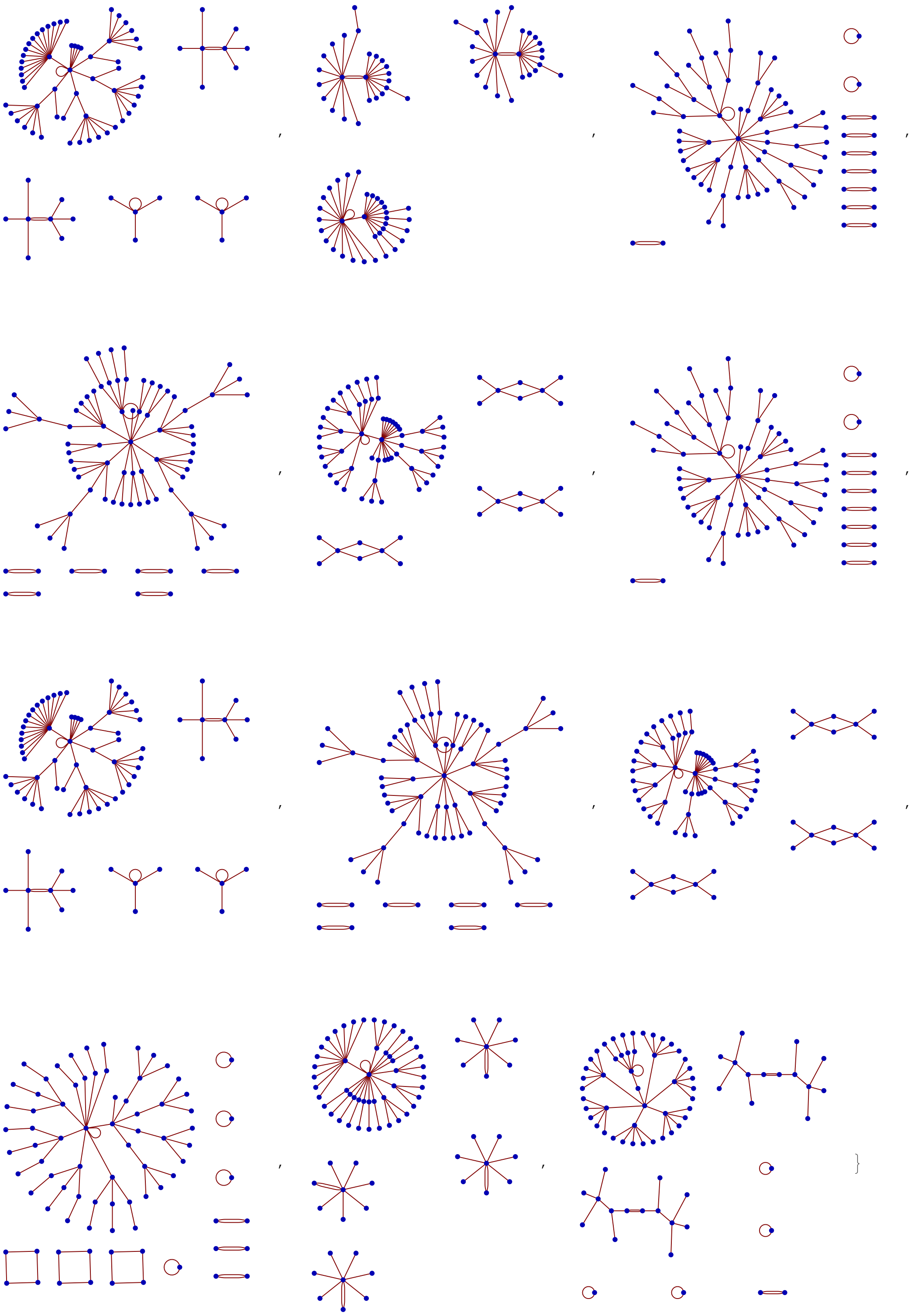

,

,

,


,

,

,


,

,

,







Third-order calculus of forms

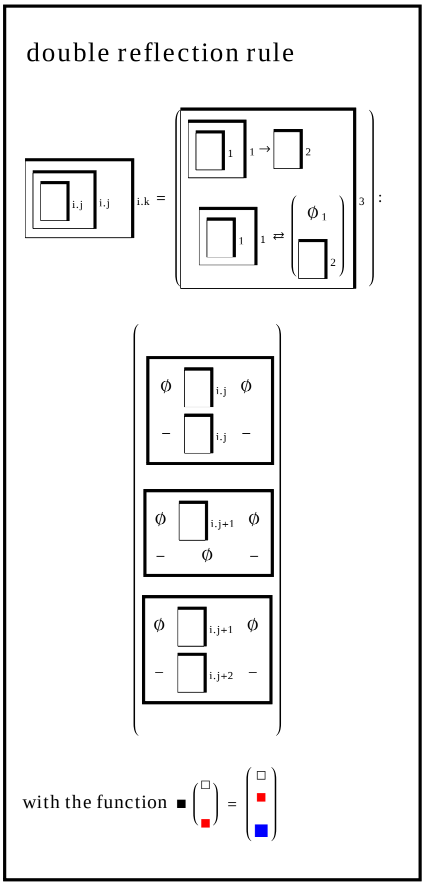
Following Georg Friedrich Hegel (1770 - 1831), a theory of reflection/reflexion is not achieved until the *third* level of reflexions is established. There might be a level of reflexion that is deconstructing the very idea of reflexion in itself.

Nina Ort, *“Reflexionslogische Semiotik”. Semiotics in the Logic of Reflection: A Non- classical extended Semiotics based on Gotthard Günther and Charles S. Peirce and its Application to Literary Theory*

http : // trans.revues.org / 276

“Reflection may one day be as common as recursion” - Brian Smith, Reflection and semantics in Lisp

http://nl.ijs.si/~damjan/cr.html



Sub-rule manipulation for ruleRCI

How to deal with cellular automata defined by a rule set of 20^4 = 160000 single rules? Instead of listing all rules it is certainly more reasonable and feasible to abstract from the *rule scheme* of the sub-rules and to implement a choice-table that allows to pick up any rule out of the whole rule set. There is no need to restrict to a special sub-set of rules. A study of the field of possible rules might lead in a further step to a selection of specifically interesting rules.

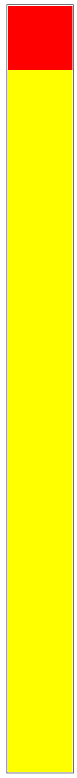
RCI requisites

Sub-rule manipulator for ruleRCI

(Debug) Out[57]=

a	4440	4441	4442	4443
b	1110	1111	1112	1113
c	2220	2221	2222	2223
d	3330	3331	3332	3333
e	1000	1001	1002	1003
f	2000	2001	2002	2003
g	3000	3001	3002	3003
h	1100	1101	1102	1103
i	1120	1121	1122	1123
j	1130	1131	1132	1133
k	1220	1221	1222	1223
l	1330	1331	1332	1333
m	2200	2201	2202	2203
n	3300	3301	3302	3303
o	2230	2231	2232	2233
p	2330	2331	2332	2333
q	1300	1301	1302	1303
r	3200	3201	3202	3203
s	3210	3211	3212	3213
t	2300	2301	2302	2303

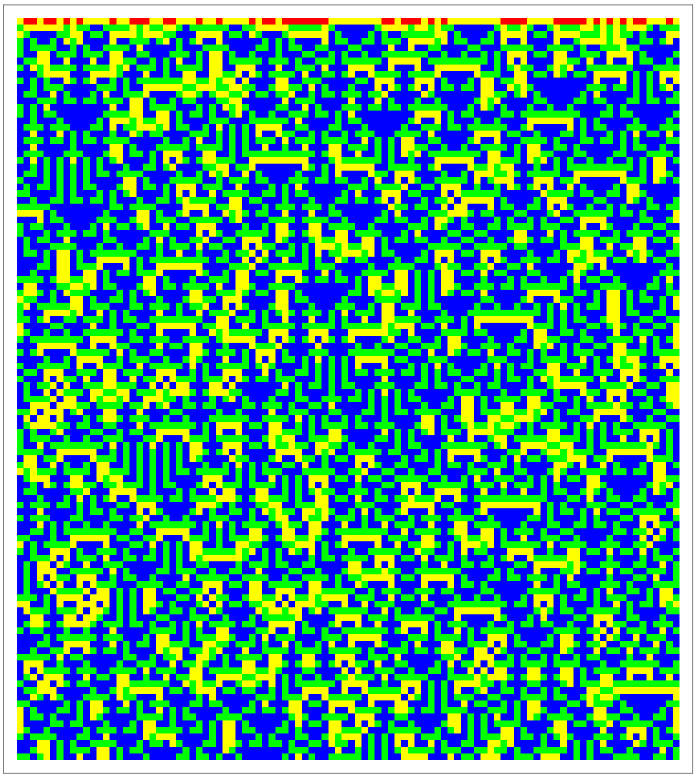
steps 11

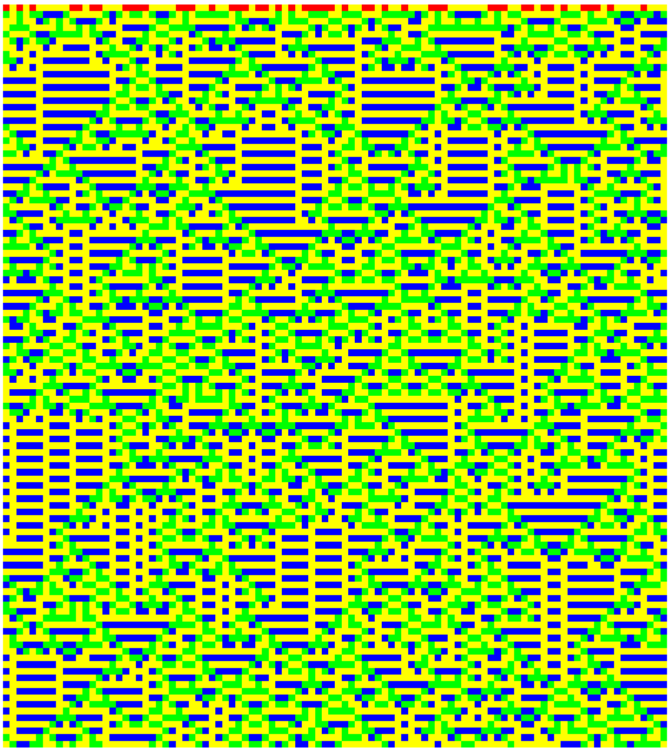


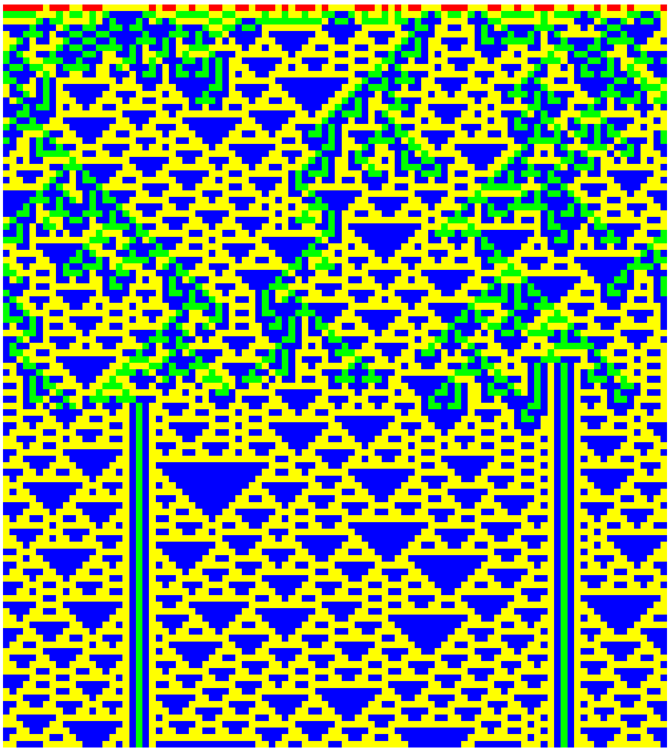
Sub-rule manipulator for random ruleRCI

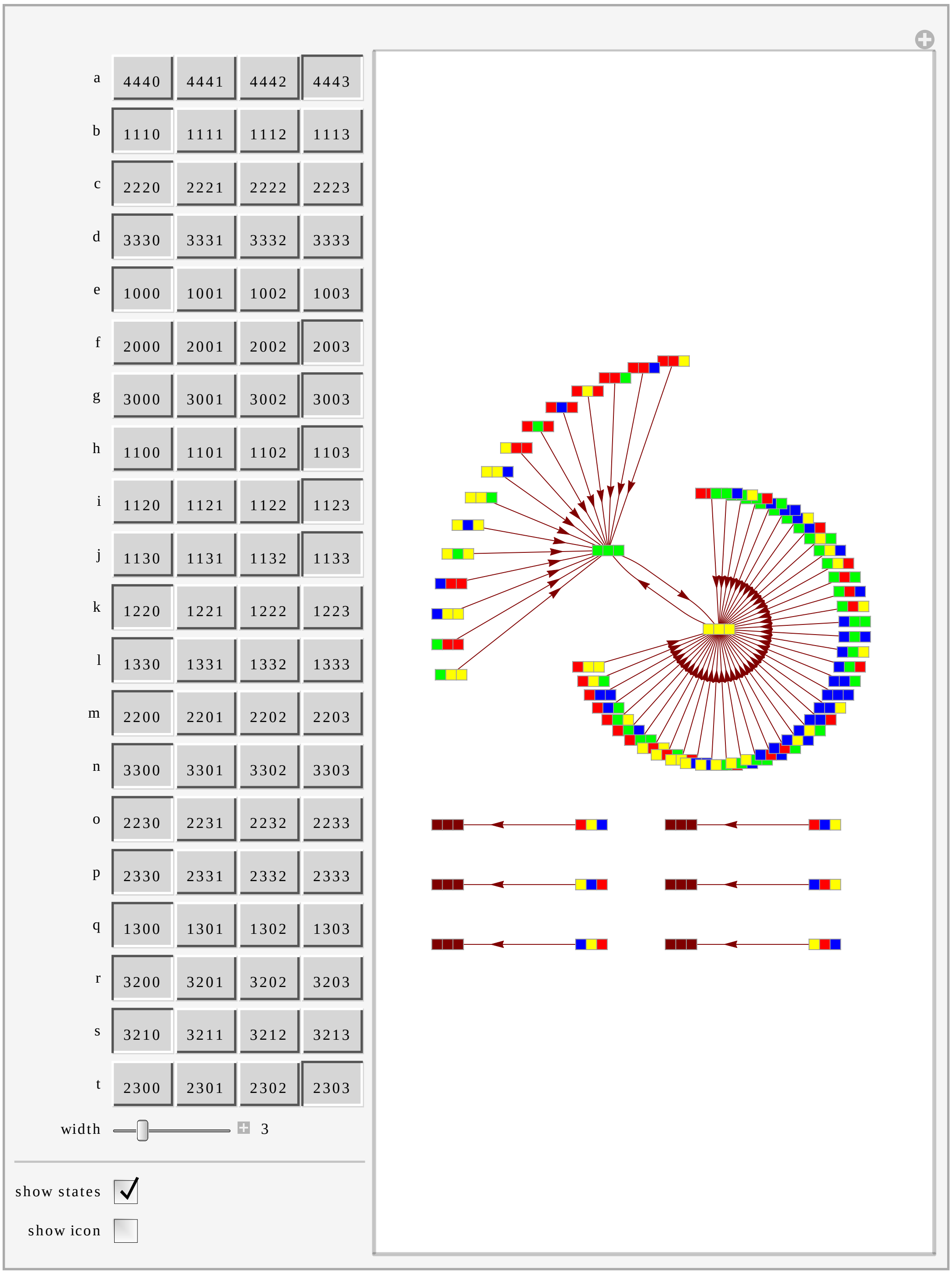
a	4440	4441	4442	4443
b	1110	1111	1112	1113
c	2220	2221	2222	2223
d	3330	3331	3332	3333
e	1000	1001	1002	1003
f	2000	2001	2002	2003
g	3000	3001	3002	3003
h	1100	1101	1102	1103
i	1120	1121	1122	1123
j	1130	1131	1132	1133
k	1220	1221	1222	1223
l	1330	1331	1332	1333
m	2200	2201	2202	2203
n	3300	3301	3302	3303
o	2230	2231	2232	2233
p	2330	2331	2332	2333
q	1300	1301	1302	1303
r	3200	3201	3202	3203
s	3210	3211	3212	3213
t	2300	2301	2302	2303

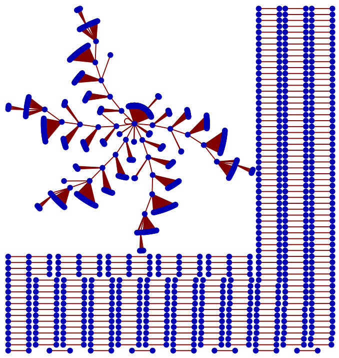
steps 111

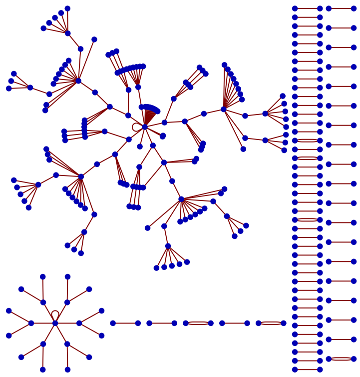


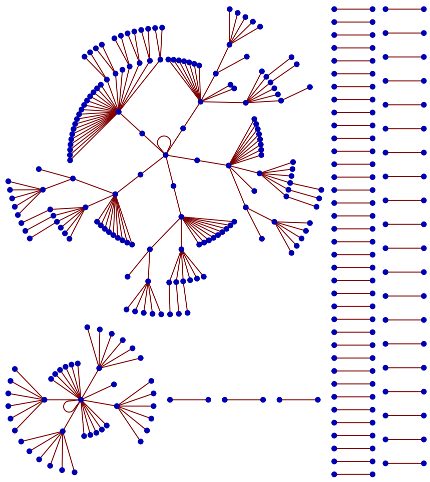


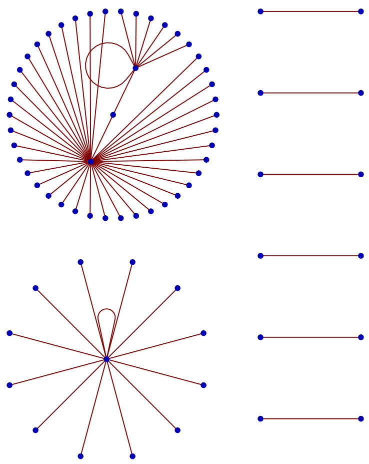


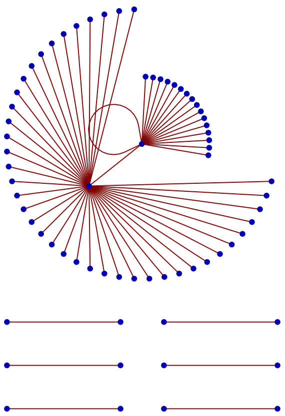

a 4440 4441 4442 4443
b 1110 1111 1112 1113
c 2220 2221 2222 2223
d 3330 3331 3332 3333
e 1000 1001 1002 1003
f 2000 2001 2002 2003
g 3000 3001 3002 3003
h 1100 1101 1102 1103
i 1120 1121 1122 1123
j 1130 1131 1132 1133
k 1220 1221 1222 1223
l 1330 1331 1332 1333
m 2200 2201 2202 2203
n 3300 3301 3302 3303
o 2230 2231 2232 2233
p 2330 2331 2332 2333
q 1300 1301 1302 1303
r 3200 3201 3202 3203
s 3210 3211 3212 3213
t 2300 2301 2302 2303
width 3
show states
show icon

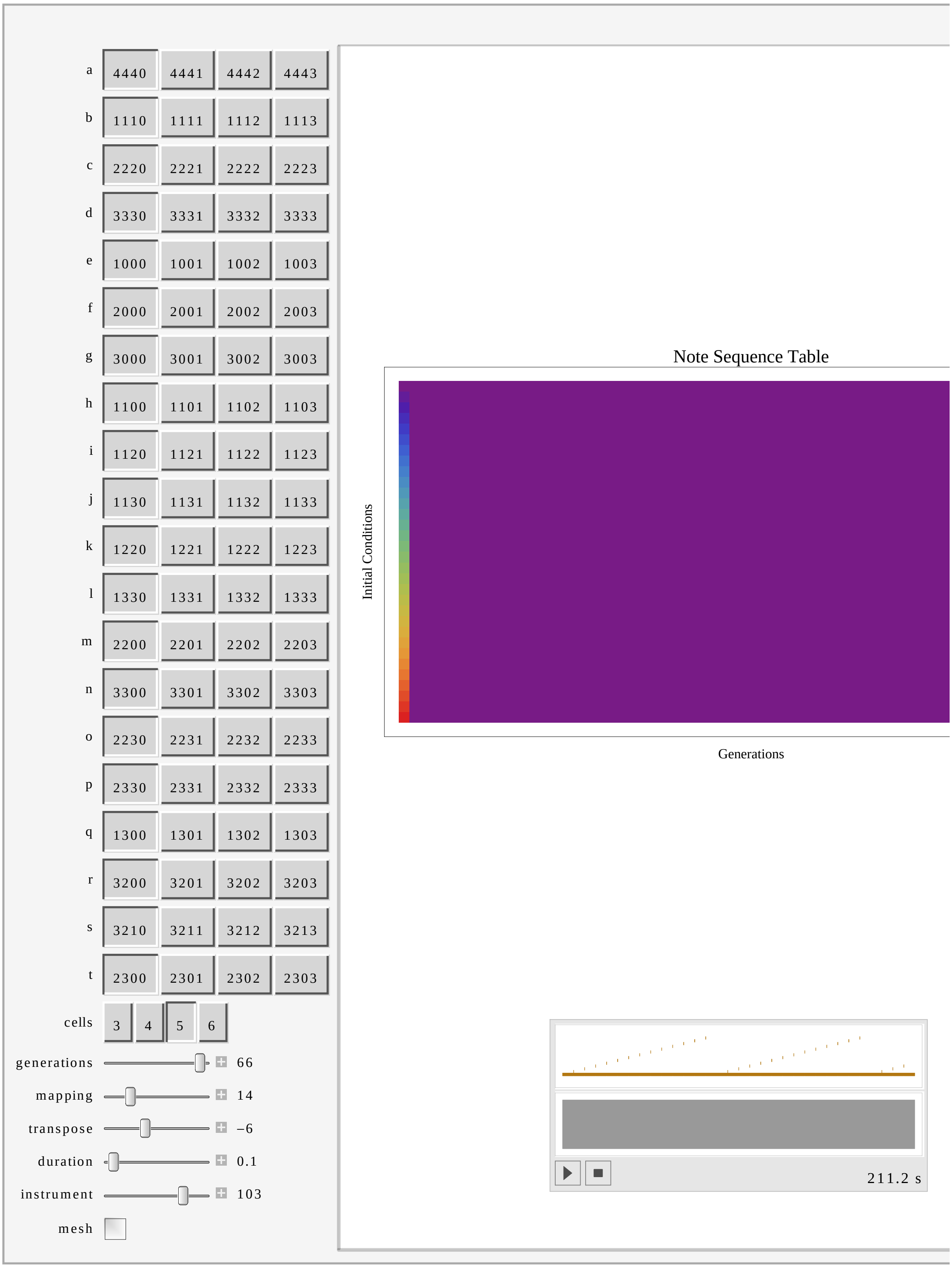












RCI-Interactions: Commutativity, Loops and Plots

Commutativity

RuleTableFromkAryRCI ⟷ kAryFromRuleTableRCI
↖ ↗
RuleSetRCI
indRCIrules

```
RuleTableFromkAryRCI[
kAryFromRuleTableRCI[
RuleTableFromkAryRCI[{0,1,0,1,0,1,2,0,1,2,0,1,0,1,3,1,2,3,1,1}]]]==
RuleTableFromkAryRCI[{0,1,0,1,0,1,2,0,1,2,0,1,0,1,3,1,2,3,1,1}]/.indRCIrules
```

```
True
```

```
kAryFromRuleTableRCI[
Mod[ReLabel /@
Map[Flatten,ruleRCI[
{4443,1110, 2222, 3330,1001,
2000, 3000,1102,1123,1133,
1220,1332,2203,3300,2330,
1301,2232,3200,3211,2302}]/.Rule→List,1],4]]
```

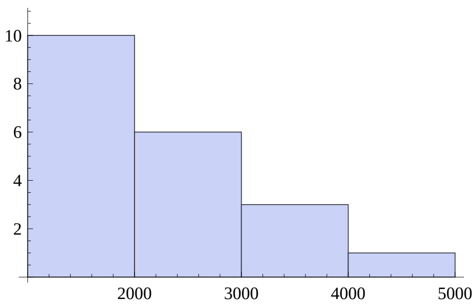
```
{2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 2, 1, 1, 3, 3, 3, 3, 3, 3,
 2, 2, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 1, 2, 2, 3, 3, 3, 2, 3, 2, 1,
 1, 3, 1, 1, 2, 2, 3, 2, 3, 1, 1, 3, 2, 3, 2, 1, 1, 1, 2, 1, 3, 2, 3}
```

```
RuleTableFromkAryRCI[{0,1,0,1,3,1,2,0,0,0,0,1,0,1,0,1,2,0,0,1}]/.indRCIrules
```

```
{4440, 1001, 1100, 1111, 1123, 1221, 2202, 2000, 1200,
 2220, 1130, 1331, 3300, 2331, 3000, 1301, 2302, 1230, 2230, 3331}
```

```
RuleTableFromkAryRCI[Flatten[
CellularAutomaton[
    ruleRCI[{4440, 1001, 1100, 1111, 1120, 1221, 2202, 2000, 1020, 2220, 1130,
      1331, 3300, 2331, 3000, 1301, 2302, 1230, 2230, 3331}], {0, 1, 0, 1,
     0, 1, 2, 0, 0, 0, 0, 1, 0, 1, 0, 1, 2, 0, 0, 1}, {{3}}]]]] /. indRCIrules
```

```
{4441, 1001, 1100, 1110, 1120, 1221, 2200, 2000, 1201,
 2221, 1131, 1330, 3301, 2330, 3000, 1301, 2301, 1231, 2231, 3330}
```



Loops

m

```
{4441, 1001, 1101, 1112, 1122, 1222, 2201, 2002, 1202, 2222,
 1131, 1332, 3302, 2331, 3001, 1301, 2302, 1232, 2231, 3332}
```

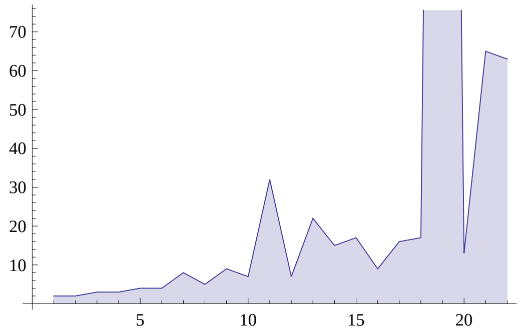
m

```
{{4440, 1001, 1100, 1110, 1121, 1220, 2200, 2000, 1200, 2220,
  1130, 1331, 3300, 2331, 3000, 1300, 2301, 1231, 2230, 3331},
 {1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 2, 2, 1, 2}}
```

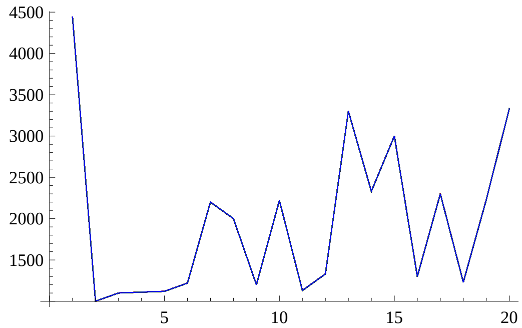
```
{23, {{4443, 1001, 1103, 1111, 1123, 1221, 2201, 2001, 1201,
    2221, 1131, 1331, 3303, 2331, 3000, 1303, 2303, 1233, 2230, 3333},
   {4441, 1001, 1100, 1111, 1120, 1220, 2201, 2001, 1201, 2221,
    1131, 1330, 3300, 2331, 3001, 1300, 2301, 1230, 2230, 3330},
   {4443, 1003, 1103, 1111, 1121, 1221, 2203, 2001, 1201, 2221,
    1133, 1331, 3301, 2333, 3003, 1303, 2301, 1231, 2233, 3331},
   {4441, 1001, 1101, 1110, 1121, 1220, 2200, 2000, 1201, 2220,
    1130, 1330, 3300, 2331, 3001, 1301, 2300, 1230, 2230, 3331},
   {4441, 1001, 1103, 1113, 1121, 1221, 2203, 2001, 1201, 2221,
    1133, 1333, 3301, 2333, 3001, 1303, 2301, 1233, 2231, 3333},
   {4440, 1000, 1101, 1111, 1120, 1220, 2200, 2000, 1201, 2220,
    1131, 1331, 3301, 2330, 3001, 1300, 2301, 1230, 2231, 3330},
   {4443, 1001, 1103, 1113, 1121, 1223, 2203, 2001, 1201, 2223,
    1133, 1331, 3303, 2333, 3001, 1303, 2301, 1233, 2231, 3331},
   {4440, 1001, 1101, 1111, 1121, 1221, 2201, 2000, 1200, 2221,
    1131, 1331, 3301, 2331, 3001, 1300, 2301, 1230, 2230, 3330},
   {4441, 1003, 1101, 1111, 1121, 1221, 2203, 2001, 1201, 2223,
    1131, 1331, 3301, 2331, 3003, 1303, 2301, 1231, 2233, 3333},
   {4441, 1000, 1100, 1111, 1121, 1220, 2200, 2000, 1200, 2220,
    1130, 1331, 3301, 2330, 3001, 1301, 2300, 1230, 2231, 3331},
   {4443, 1001, 1101, 1113, 1123, 1221, 2203, 2003, 1203, 2223,
    1131, 1333, 3303, 2333, 3003, 1303, 2301, 1231, 2233, 3331},
   {4440, 1000, 1100, 1111, 1121, 1221, 2201, 2001, 1201, 2221,
    1131, 1331, 3301, 2331, 3001, 1301, 2300, 1230, 2230, 3331},
   {4441, 1003, 1101, 1113, 1121, 1221, 2201, 2001, 1201, 2221,
    1131, 1331, 3301, 2331, 3001, 1303, 2301, 1233, 2231, 3331},
   {4440, 1000, 1101, 1110, 1120, 1221, 2201, 2001, 1201, 2221,
    1131, 1331, 3301, 2331, 3000, 1300, 2301, 1230, 2230, 3331},
   {4441, 1001, 1101, 1111, 1121, 1223, 2201, 2001, 1201, 2221,
    1131, 1331, 3301, 2333, 3001, 1301, 2301, 1231, 2231, 3331},
   {4441, 1001, 1101, 1111, 1120, 1220, 2200, 2001, 1201, 2221,
    1131, 1331, 3300, 2330, 3000, 1301, 2301, 1231, 2231, 3331},
   {4441, 1001, 1101, 1113, 1121, 1223, 2201, 2003, 1201, 2221,
    1131, 1333, 3301, 2333, 3001, 1303, 2301, 1231, 2231, 3331},
   {4441, 1001, 1100, 1110, 1121, 1220, 2201, 2000, 1200, 2221,
    1130, 1330, 3301, 2330, 3001, 1300, 2300, 1231, 2231, 3331},
   {4441, 1003, 1101, 1111, 1121, 1223, 2201, 2001, 1201, 2221,
    1131, 1331, 3301, 2333, 3001, 1301, 2301, 1233, 2231, 3331},
   {4440, 1000, 1100, 1111, 1120, 1220, 2200, 2001, 1201, 2221,
    1131, 1331, 3300, 2330, 3000, 1301, 2300, 1230, 2230, 3331},
   {4441, 1003, 1101, 1111, 1121, 1223, 2201, 2003, 1201, 2221,
    1131, 1333, 3301, 2333, 3001, 1301, 2301, 1233, 2231, 3331},
   {4440, 1000, 1100, 1111, 1120, 1220, 2201, 2000, 1200, 2221,
    1130, 1330, 3301, 2330, 3000, 1301, 2300, 1230, 2230, 3331},
   {4441, 1003, 1101, 1111, 1121, 1221, 2201, 2001, 1201, 2221, 1131,
    1331, 3301, 2331, 3001, 1301, 2301, 1233, 2231, 3331}}}
```

Plots

```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleRCI[{4440, 1001, 1100, 1111, 1120, 1221, 2202, 2000, 1201,
2222, 1130, 1331, 3300, 2331, 3003, 1301, 2302, 1233, 2231, 3331}]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 22}], Filling → Axis]
```




```
ListLinePlot[Table[RuleTableFromkAryRCI[Mod[Flatten[
CellularAutomaton[ruleRCI[{4440,1001,1100,1111,1120,1221,2202,2000,1201,
2222,1130,1331,3300,2331,3003,1301,2302,1233,2231,3331}],
{0,1,0,1,0,1,2,0,0,0,0,1,0,1,0,1,2,0,0,1},{{m}}]],4]]
/.indRCIrules,{m,1,111,1}],PlotStyle->{Red,Green,Blue, Black},ImageSize→Medium]
```




```
Histogram[Table[RuleTableFromkAryRCI[Mod[Flatten[
CellularAutomaton[ruleRCI[{4440,1001,1100,1111,1120,1221,2202,2000,1201,
2222,1130,1331,3300,2331,3003,1301,2302,1233,2231,3331}],
{0,1,0,1,0,1,2,0,0,0,0,1,0,1,0,1,2,0,0,1},{{m}}]],4]]
/.indRCIrules,{m,1,111,1}]]
```