— vordenker-archive —

Rudolf Kaehr
(1942-2016)

Title

Self-modifying Morpho Cellular Automata
A sketch on different modi of self-reference in morphoCAs

Archive-Number / Categories

3_42 / 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

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

The case for distinction based CAs is proposed at: "Self-modifying Forms of Cellular Automata"

Citation Information / How to cite

Rudolf Kaehr: "Self-modifying Morpho Cellular Automata", www.vordenker.de (Sommer Edition, 2017) J. Paul (Ed.), http://www.vordenker.de/rk/rk_Self-modifying-morphoCAs_2015.pdf

Categories of the RK-Archive

Self-modifying MorphoCellular Automata

A sketch on different modi of self-reference in morphoCAs

Dr. phil Rudolf Kaehr

ISSN 2041-4358
(work in progress, v. 0.5, June, April 2015)

Abstract

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

The case for distinction based CAs is proposed at: Self-modifying Forms of Cellular Automata.html (pdf,cdf)

Tasks:
Introduction of sub-rules for ECA and sub-rule manipulation for ECA and morphoCAs,
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.
Morphogrammatics of fixed points. Fixed point as morphic palindromes.

Part One: Iterative self-modification

Background and motivations

> "Although the informational feedback loop was initially linked with homeostasis, it quickly led to the more threatening and subversive idea of *reflexivity*. A few years ago I co-taught, with a philosopher and a physicist, a course on reflexivity. As we discussed reflexivity in the writings of Aristotle, Fichte, Kierkegaard, G ̈del, Turing, Borges, and Calvino, aided by the insightful analyses of Roger Penrose and Douglas Hofstader, I was struck not only by the concept's extraordinarily rich history but also by its tendency to mutate, so that virtually any formulation is sure to leave out some relevant instances. Instructed by the experience, I offer the following tentative definition, which I hope will prove adequate for our purposes here.
> *Reflexivity is the movement whereby that which has been used to generate a system is made, through a changed perspective, to become part of the system it generates.*

http://monoskop.org/images/5/50/Hayles_N_Katherine_How_We_Became_Posthuman_Virtual_Bodies_in_Cybernetics_Literature_and_Informatics.pdf

System⟶Part⟶System
Reflexivity is the movement whereby that which has been used to generate a system is made, through a changed perspective, to become part of the system it generates.

And, hence, again, the neglected move, vica versa:
Part⟶System⟶Part
Reflexivity is the movement whereby that which has been part of a system, through a changed perspective, to become the system of the parts it generates.

Gotthard Gunther, CYBERNETIC ONTOLOGY AND TRANSJUNCTIONAL OPERATIONS, 1962

> *"On the other hand, when we speak of individual centers of self-reflection in the world and call them subjects we obviously do not refer to retroverted self-reflection. Such individual centers have, as we know very well, a genuine environment (which the Universe has not!) and what they reflect is this very environment. It stands to reason that these systems of self-reflection with centers of their own could not behave as they do unless they are capable of "drawing a line" between themselves and their environment. We repeat that this is something the Universe as a totality cannot do. It leads to the surprising conclusion that parts of the Universe have a higher reflective power than the whole of it, as has been recognized for a long time. In Hegel´s logic the phenomenon of reflection is subdivided into three parts: He defines them as:*
>
> > *a) retroverted reflection (Reflexion-in-sich)*
> > *b) transverted reflection (Reflexion-in-Anderes)*
> > *c) retroverted reflection of retroversion and transversion (Reflexion-in-sich der Reflexion-in-sich und-Anderes)*
>
> *Section (a) represents the physical system of the external world described by its specific reflective properties. But (b) and (c) signify the additional capacities of reflection which sub-systems of the Universe must possess if they are to be called subjects.*
>
> *This shows that the early philosophic theory of reflection is still ahead of the present logical state of cybernetics. We talk about self-organizing systems and their environments; but Hegel´s distinction between (a), (b) and (c) shows that this is not enough. A self-reflective system which shows genuine traits of subjective behavior must be capable of distinguishing between two types of environment and be able to react accordingly.*
>
> *First it must reflect an "outside" environment which lies beyond its own adiabatic shell and second it must be capable of treating (b) as an environment to (c).*

http://www.thinkartlab.com/pkl/archive/GUNTHER-BOOK/GUNTHER.htm

Alyssa Adams et al

> "By feeding a conventional one-dimensional elementary cellular automaton's own state back into the iteration rule that generates its next state, we have developed a new modeling framework for investigating information control in natural systems.
>
> "By feeding the state of the focal entity back into the rules that govern how the entity evolves, we avoid these complications and still allow for entities to alter and be altered by their environment
>
> "Each new generation of a population leaves an indelible mark on its *environment* and thus affects the selective pressures that shape future generations of that population.
>
> "To model this phenomenon, we have augmented traditional cellular automata with state-dependent feedback.
> Rather than generating automata executions from an initial condition and a static rule, we introduce mappings which generate iteration rules from the cellular automaton itself
>
> "Here, we introduce self-referencing cellular-automaton framework we call PICARD where PICARD Implements CA Rules Differently (PICARD).
>
> "Like a traditional one-dimensional CA, PICARD executions move from one iteration to another by some rule.
> However, whereas traditional CA's require the rule to be *static* and *externally* specified, PICARD infers the iteration rule from the current state of the CA itself.
>
> — Does PICARD Pic H'ard *enough to tackle problems of a theory of living systems*? ----
>
> "A possible criticism of PICARD is that it introduces nonlocal effects to cellular automata. Traditional CA are built from the assumptions that cells update based entirely on local information and information about neighbors. By feeding the state of an entire row back into the CA rule, this locality assumption is

broken.

> "To some extent, PICARD is a simple attempt to add *dynamical feedback* that is missing in traditional evolutionary cellular automata. However, because iterations are generated by rules that are encoded in previous iterations, PICARD feedback is the kind of *self-reference* that is thought to be a characteristic feature of life (Goldenfeld and Woese, 2011; Hofstadter, 1979; Kataoka and Kaneko, 2000a,b; Walker and Davies, 2013)".

http://arxiv.org/abs/1405.4070, May 2014

Further reading

Computability, Self-Reference, and Self-Amendment
George Kampis

http://www.informatics.indiana.edu/rocha/kampis.html

http://www.autopoiesis.com/documents/Varela%201975a.pdf

http://www.informatics.indiana.edu/rocha/ccai.html

https : // mitpress.mit.edu/sites/default/files/titles/alife/0262297140 chap115.pdf

http://www.informatics.indiana.edu/rocha/tilsccai.html

> *"Von Foerster [1977, 1981], introduced the concept of eigen-value as values (functions, operators, algorithms) that could satisfy an indefinite recursive equation describing the organization of sensori-motor interactions, or more generally self-referencial organizations such as cognitive systems. He concluded first, from the possible existence of such values (with no true mathematical description), that they must be discrete even if their primary argument or observable is continuous, since in a recursive loop where they are generated only stable observables will maintain eigen-value representation:*
>
> *"In Other words, Eigenvalues represent* equilibria*, and depending upon the chosen domain of the primary argument [measured observable], these equilibria may be equilibria values ("Fixed Points"), functional equilibria, operational equilibria, structural equilibria, etc."* [Von Foerster, 1977, 1981, page 278].

http://www.informatics.indiana.edu/rocha/tilsccai.html

http://projecteuclid.org/download/pdf_1/euclid.ndjfl/1093882412

Self-Reference and Time According to G. Spencer-Brown
http://sfile.f-static.com/image/users/112431/ftp/my_files/sbrown.pdf?id=11014655

Ouroboros avatars:
A mathematical exploration of Self-reference and Metabolic Closure
https://mitpress.mit.edu/sites/default/files/titles/alife/0262297140chap115.pdf

Cybersemiotics: Why information is not enough!
http://www.infoamerica.org/teoria_articulos/luhmann02.pdf

Alfred Locker
http://www.vordenker.de/locker/metatheor-presupp-autopoiesis.pdf

http://www.revue3emillenaire.com/blog/vers-un-nouveau-paradigme-scientifique-ontologies-et-logiques-selon-gotthard-gunther-par-daniel-verney/

Über die formale Rekonstruktion von Funktionssystemen als mediale Codes der gesellschaftlichen Reproduktion und eine dadurch ermöglichte Erklärung der entsprechenden Strukturgenese mittels transjunktionaler Operationen.

http://www.karafillidis.com/downloads/MedialeCodesExtended.pdf

Self-reference between circularity and proemiality

Re-entry all over

http://homepages.math.uic.edu/~kauffman/SelfRefRecurForm.pdf

Beyond re-entry forms: factum/existence

„Damit zeichnet sich eine Antwort ab auf die Frage,..., inwiefern jemand sich in seinen praktischen Ja/Nein-Stellungnahmen - in seinem 'ich kann -' - zu sich verhält. Die Antwort lautet: nicht indem das Subjekt sich selbst zum *Objekt* wird, sondern indem es sich zu seiner *Existenz* verhält."(Tugendhat 1979, 38)

„Daß ich mich voluntativ-affektiv zu meiner *Existenz* verhalten kann, gründet darin, daß die Proposition, zu der ich mich dabei verhalte, nicht das *Faktum* ist, daß ich existiere, sondern die bevor stehende *Existenz* und das heißt die (praktische) Notwendigkeit, daß ich zu sein habe, und in eins die (praktische) Möglichkeit, zu sein oder nicht zu sein bzw. so und so zu sein oder nicht zu sein." (Tugendhat 1979, p. 189)

"The question that still remained open there - in what sense can the relation of oneself to this propositional content be understood as a practical one - is now answered.
My capacity to relate myself in a voluntative-affective mode to my *existence* rests upon the fact that the proposition to which I thereby relate myself is not the fact that I exist, but my existence as it impends; and this means the (practical) necessity that I have to be, and the together with this the (practical) possibility to be or not to be, or to be or not to be in such and such way." (Ernst Tugendhat, Self-Consciousness and Self-Determinatin, MIT 1986, p. 168)

http://memristors.memristics.com/MorphoReflection/Morphogrammatics%20of%20Reflection.html

There are profound analysis about the difference of the concept of "object" and the concept of "existence". The following study is just taking into account the abstract differentiation of two and not of just one 'domain', i.e. contexture of reflexive behaviors.

It is proposed that the relationsip between the two contextures of reflexivity is of a chiastic, i.e. proemial nature.

Summary

Results which have been achieved might be summarized by a typology of sign-use, algorithms and machines.

bisimular equivalence - co-creative machines	metamorphosis
monomorphic similarity - self-organizing machines	alterability
kenomic equivalence - non-trivial machines	iterability
semiotic equality - trivial machines	iteration

Slogans: "the same is different", "iteration alters", "iteration of the new", "repetition of the identical"

Sketch of mathematical aspects of self-modifying automata

From oscillators to chiasms

A mathematical explication of self-referential forms might itself oscillate between Spencer-Browns f(f)-re-enry form and Dana Scots mapings: D ⟶ [D ⟶ D].

A chiastic deconstruction of the current mathematical models of self-reference leads to the insight of a double movement of poly-contextural thematization and pre-logical inscription as it is developed in diamond category theory.

Such a double thematization and formalization takes into account the double strategy of de-paradoxification: space and time, i.e. '*espacement*' and '*temporalization*' (Raumung /Zeitigung, spacing/temporalization}.

The classic approach to deal with paradoxes is to connect the process of de-paradoxification with time.

Well known oscillatars and atractors appear in this context.

Paradoxes as a logical form that correspond more with a unfolding of the paradox in space are producing contradictions, i.e. antinomies.

From there there is not much to elaborate than paradoxes of paradoxes.
A short formula for a unification of time- and space-related applications could be seen in the diamond model of self-reference: $\mathrm{D} \longrightarrow \begin{pmatrix} D \longrightarrow D \\ \updownarrow \quad x \quad \updownarrow \\ D \longleftarrow D \end{pmatrix}$.

That is the standard formula $f = f(f)$ becomes $F^{(3)} = \begin{pmatrix} F_1 \longrightarrow F_1 \\ \updownarrow \quad x \quad \updownarrow \\ F_2 \longleftarrow F_2 \end{pmatrix}$, i.e. a explication of the chiasm of *f* as operator and *f* as operand: $f = f_1(f_2)$ with f_1= operator and f_2= operand of the operation (function) *f*.

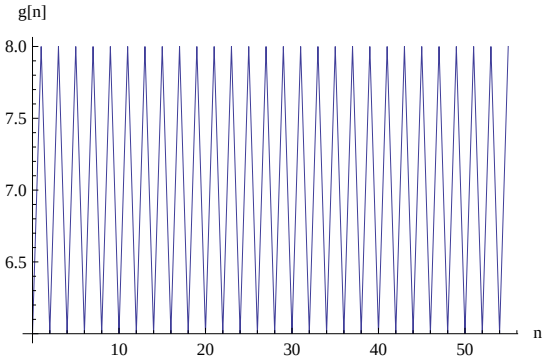
Iterative self-reference as simple combinatorial oscillators

The collaps of operator and operand, term and application, is inscribed as F ≡ F(F) or just F ≡ FF.

Is that enough for a theory of living systems?

> *"Combinatory logic is 'simplified' way of doing the lambda calculus. In the lambda calculus there are variables and constants, applications, and abstractions; combinatory logic makes do with variables and constants, and applications only -- abstractions are done away with."*
>
> *http://softoption.us/content/node/45*

The form of oscillation




```
Table[f[n], {n, 0, 3}] // MatrixForm
```

$$\begin{pmatrix} \mathrm{s[i][i][s[i][i]]} \\ \mathrm{i[s[i][i]][i[s[i][i]]]} \\ \mathrm{s[i][i][s[i][i]]} \\ \mathrm{i[s[i][i]][i[s[i][i]]]} \end{pmatrix}$$

```
Table[f[n], {n, 0, 3}] /. filterSI // MatrixForm
```

$$\begin{pmatrix} \mathrm{A[A]} \\ \mathrm{B[B]} \\ \mathrm{A[A]} \\ \mathrm{B[B]} \end{pmatrix}$$

```
f[n_] := Nest[SKStep, s[i][i][s[i][i]], n]
```

Thus the form of s[i][i][s[i][i]] is the morphic palindrome [ABAB].

The form of the re - entry dynamics of forms

http : // iconicmath.com/logic/lawsofform/

Moulin's mathematical modeling

Following the papers of the French mathematician J-P Moulin, the proposed classical definitions might be of some preliminary use.

Definition 1

- $A \subset \mathrm{N}$, a finite subset of N
- $F_A = \{f : A \to A\}$, a set of functions which map A into itself.

$F_A(A) \subseteq A$ is the set of *images* of A through functions of F_A :

$$F_A(A) = \bigcup_{f \in F_A} f(A)$$

- C : $A \to F_A$ a function .

An *automaton* **a** is defined as an n-uple (S, I, O, F, G) where:

- S is the set of *internal states* of **a**,
- I is the set of *inputs* of **a**,
- O is the set of *outputs* of **a**,
- $F : I \times S \to S$ is the *transition function* of **a**,
- $G : I \times S \to O$ is the *output function* of **a**

Defintion 2

We will say that $\mathbf{a_{sm}} = (A, F_A, C, g)$ is a *self-modifying automaton* (Fig.2) iff:

- $g \in F_A$,
- $S = F_A$,
- I = O = A,
- output function G: $A \times F_A \to A$ is defined by:
  $\forall (e_t, f_t) \in A \times F_A, \quad s_t = G(e_t, f_t) = f_t(e_t)$
- transition function F : $A \times F_A \to F_A$ is defined by:
  $\forall (e_t, f_t) \in A \times F_A, \quad f_{t+1} = F(e_t, f_t) = C[s_t]$
- $e_{t+1} = g(s_t)$

http : // www.self - programming - machines.org/pdf/sfbt3.pdf

Obviously, and again, the crucial point of the definition can be detected at the not so harmless equation: $\boldsymbol{e_{t+1} = g(s_t)}$.

Self - modifying automaton

Let $\mathbf{a_{sm}} = (A, F_A, C, g)$ be a self-modifying automaton characterised by :

$$e_{n+1} = f_n(e_n)$$

$$f_{n+1} = C(e_{n+1})$$

J-P Moulin, Self Programming Machines (I).

"In this paper, we have demonstrated that machines which associate two or more automata which *change their internal organisation* whenever the *external* data vary, converge exclusively at *fixed points*, have the *self programming property* and give a solution to the biological enigma of the fact that appropriate responses occur in the absence of either a programmer or any human intention."

Depiction

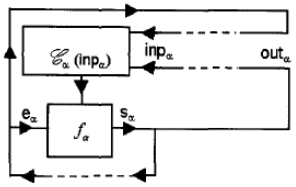


Fig. 1.

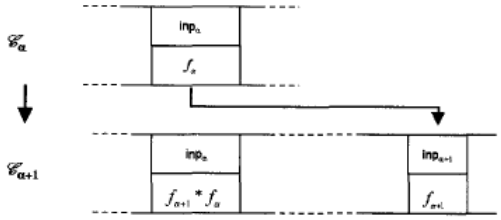


Fig. 2.

http : // www.self - programming - machines.org/pdf/SPM3.pdf

"The internal organisation of these machine is a function (f) of discrete variable, the input is the value of the variable and the output is the value of this function."

Figure 1

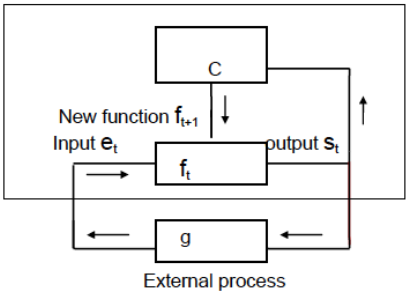
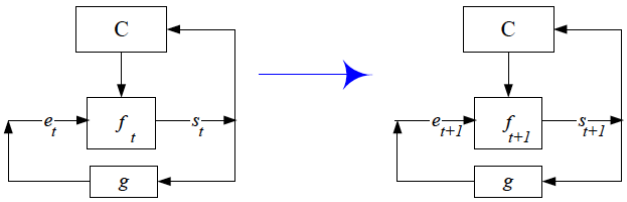


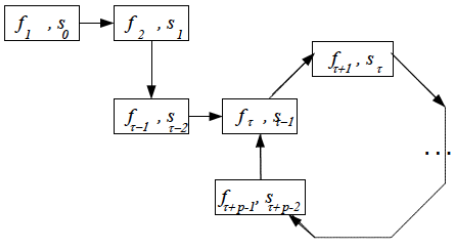
Figure 3



"We will say that self - modifying automaton a_{sm} = (A, FA, C, g) is self - connected (Fig .3) iff g = Id: i.e. :

• $e_{t+1} = s_t$ (4)

"Theorem 1: Any trajectory of a self - modifying automaton a_{sm} converges to a limit cycle."



http : // www.self - programming - machines.org/pdf/sfbt3.pdf

Self-reference within Finite State Machines

JFLAP defines a finite automaton (FA) M as the quintuple
M = (Q, Σ, δ, qs, F)
where
Q is a finite set of *states* $\{q_i \mid i \text{ is a nonnegative integer}\}$
Σ is the finite *input* alphabet
δ is the *transition* function,
$\delta : D \to 2^Q$ where D is a finite subset of $Q \times \Sigma^*$
q_s (is member of Q) is the *initial* state
F (is a subset of Q) is the set of *final* states.

• **Weak self-reference** between *initial* and *final* states of FSM:
$M = (Q, \Sigma, \delta, qs_1, F_1) \Longrightarrow M' = (Q, \Sigma, \delta, qs_2 = F_1, F_2)$

• **Strong self-reference** between the definition of the *transition* function and *final* states of FSM:
$M = (Q, \Sigma, \delta_1, qs_1, F_1) \Longrightarrow M' = (Q, \Sigma, \delta_2 = F_1, qs_2, F_2)$

Further reading

ftp://ftp.cs.wpi.edu/pub/techreports/pdf/93-11.pdf

https : // github.com/machine - intelligence/Botworld

Technicalities of the paper

To follow the constructs for self-reference in *Mathematica* as presented in the demonstrations of Michael Schreiber, the techniques of *enumerations* in morphoCAs had to be adjusted from a single numerical number for the ECA rule number to tuples of numbers for morphoCA rules.

Methods of Pedro P. B. de Oliveira and Micael Schreiber are applied.

Oliveira, *Representing Families of Cellular Automata Rules,*
Schreiber, *Selfish Elementary Cellular Automata* and *Fixed Point Elementary Rules.*

Modi of identification in the process of self-reference

Rules1 based on init1 is producing a result in CA1, this result1 is used in CA2 as a new init2 for the rules of CA2, which are producing a result2 that can be used as new init3 for a CA3.

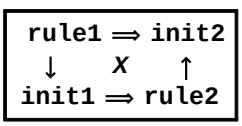
In this wording, the *modus* of self-reference is not specified but is commonly supposed to be a kind of identity.

Secondly, it is supposed that the encounter of the first with the latter happens at the desired locus.

This is just one possibility of at least 4 to refere to it-self.

Modi of identification

• equality,
• equivalence,
• similarity and
• bisimilarity.

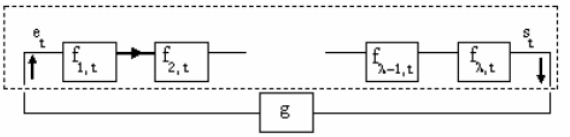


In the case of iterative self-reference, the structural differences of the CAs are collapsing and the difference is distributed into the same CA over the time steps.

This mono-contextural approach is covering all sorts of successive and parallel CA combinations, but still remains restricted to its conceptual contexture. This concept of intra-contextural iterativity is covert e.g. by *"Chains of self-modifying automata"* (J-P Moulin)

Theorem 1
"Any chain of a self-modifying automaton Cas_{sm} has an equivalent self-modifying automaton." (J-P Moulin)

Figure 8



Contextures are not defined by their 'content', say different concepts of CAs, but they might contain structurally different or similar CAs. Therefore a chain that is not defined as an intra-contextural path but as a journey between different contextures is trivially not anymore characterized by the 'mono-space' presumbtion of the Theorem 1.

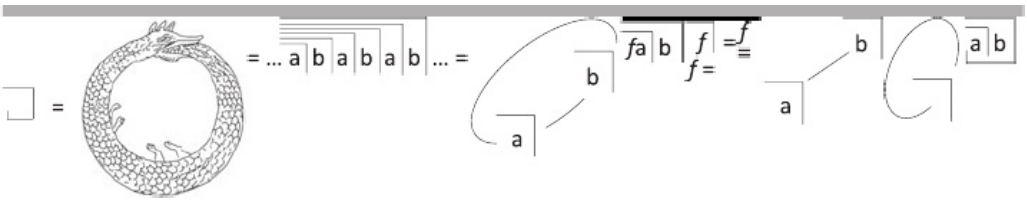
What separates contextures is their index in a polycontextural kenomic grid.

Also different contextures are ruled by their own semiotics, logic and arithmetic and separated discontexturaly from each other by their structural kenomic indexes, they are nevertheless mediated interactively and reflexionaly with each other.

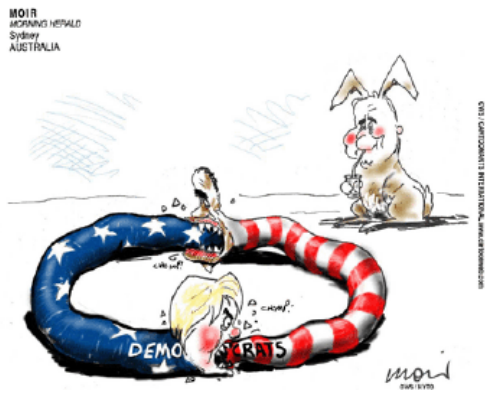
Different contextures might cover different concepts of CAs.

Hence, a mediation of *computational*, *indicational* and *morphic* CAs might cooperate as a polycontextural complexion of discontextural CAs.

Self-reference in sensu abstracto



Hence and therefore, here again :



For instance

http://www.academia.edu/1265199/Snakes_all_the_Way_Down_Varelas_Calculus_for_Self-Reference_and_the_Praxis_of_Paradise

http://www.wolframscience.com/conference/2004/presentations/material/mschreiber-computational.nb

> "The infinite nest of boxes is invariant under the addition of one more surrounding box. Hence this infinite nest of boxes is a fixed point for the recursion. In other words, if X denotes the infinite nest of boxes, then

X = F (X).

This equation is a description of a state of affairs. The form of an infinite nest of boxes is invariant under the operation of adding one more surrounding box. The infinite nest of boxes is one of the simplest eigenforms." (Kauffman)

http://homepages.math.uic.edu/~kauffman/ReflexPublished.pdf

Comments

Self-reference in *sensu abstracto* is self-reference in the mood of Angst.

It is one of the main topics in family oriented societies that if the son leaves the family house he will not find anymore the way back home.

That 's the reason why self-reference in Western society is conceived as a monadic reference of a subject to itself.

There is no detour and no journey in this reference. It happens immediately and causes virulent disturbation, even paradoxes and antinomies in formal systems.

Obviously there is a clash between the identity of the departing son and the identity of the returning son.

Under the conditions of identity metaphysics the conflict is immanent.

Hence, the myth of the re-entry business as it is celebrated, e.g. in German sociology is based profoundly on fear. It is the fundamental fear to miss the entry-point of the re-entry of the recursion or reflection.

Therefore the path without journey has to be as short as possible. The possible journeys have to be reduced to a 1-step self-loop. This step may be repeated or not, its structural poorness remains saved.

This holds for the basic structure of self-referentiality. Obviously, in a secondary and arbitrary way all kind of nice complex networks of references and self-references are trivially possible and commonly in use.

http://www.academia.edu/1265199/Snakes_all_the_Way_Down_Varelas_Calculus_for_Self-Reference_and_the_Praxis_of_Paradise

makingselfiesmakingself

But there is more to discover and to enjoy than such boyishness.

Everything is a selfie. But are selfies selfish? They are *"makingselfiesmakingself"*. "Making Selfies/Marking Self". (Katie Warfield)

Everything is an Actor (Carl Hewitt). Are Actors self-acting actors? No, they have futures. They don't exist, their existence is in futures.

http://letitcrash.com/post/20964174345/carl-hewitt-explains-the-essence-of-the-actor

Beyond crashs:

http://www.academia.edu/8522115/Shooting_myself_in_the_face_Authenticity_and_selfies_TEDxKPU

https://soundcloud.com/ben-manilla-productions/are-selfies-selfish

https://www.youtube.com/watch?v=Jho-2IZ_tnE

Spacing Self-Reference: A graphematic approach

Proof of the fixed point theorem by Barendregt:

Barendregt's proof

$\forall F \exists X FX = X.$
$Y \equiv (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))$ such that
$\forall F F(YF) = YF.$

Define $W \equiv \lambda x.F(xx)$ and $X \equiv WW.$
Then
$X \equiv WW \equiv \lambda x.F(xx)W \equiv F(WW) \equiv FX.$

Graphematics of the Fixed Point scheme for W

$W^1 \equiv \lambda x.\ F(xx),\ \ X \equiv W^2W^3,\ \ \lambda x.\ F(xx)W^4,\ \ F(W^5W^6).$

Thus, the full scheme of the Fixedpoint theorem is now marked by its numbers of occurrence of the term *"W"* in the application.

Graphematic Fixed – Point Scheme

Define $W^1 \equiv \lambda x.\ F(xx)$ and $X \equiv W^2W^3$.
Then
$X \equiv W^2W^3 \equiv \lambda x.\ F(xx)\,W^4 \equiv F\left(W^5W^6\right) \equiv FX.$

There is in fact also a difference of use in "*define* X ≡ WW", W^2W^3, and the logical application "*Then* X ≡ W", W^2W^3. This difference is not specially marked in this discussion.

The fixed-point theorem just says a term as an entity is identical an entity as an action.

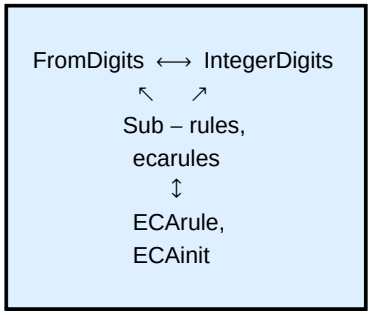
The term as an entity and the term as an application of the term are identical.

Hence the whole construction of the fixed-point theorem is written in a scripture of ultimate identity.

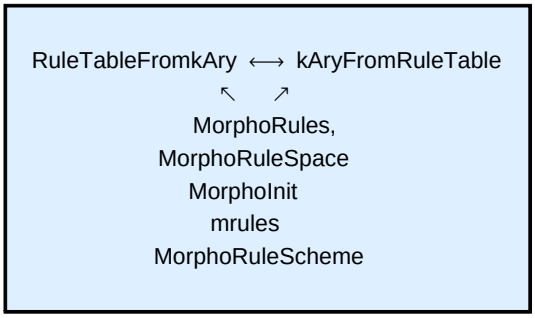
Platonism and constructivism converge to the same abstract ideal concept of identity.

Only with the presumption that all uses of the term "W" are identical and are using the same identical term "W" the proof of the fixed-point theorem is reasonable and working.

Strategies and requisites



To achieve this correspondence between *rule number* and *output values* of the application of the rules for morpho-CAs, two further procedures are introduced: *ReLabel* and the set of *mrules.*



Achievements

- Sub-rule definitions and implementations for ECA,
- Self-reference for ECA and morphoCA

* Sub-rule manipulations for ECA and morphoCA

Two different types of basic self-referentiality for morphic cellular automata are implemented:
• the self-modification of the *initial* conditions, i.e. inits to inits or initis of inits, and
• the self-modification of the *rules* of morphoCAs, i.e. rules to rules or rules of rules.

Initialization

morphoCA requisites

Examples

Examples for *kAryFromRuleTable* and *RuleTableFromkAry*

```
kAryFromRuleTableMorpho[{{1,1,1},{1,1,2},{1,2,1},{1,2,2},{1,2,3}}]
```

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

```
RuleTableFromkAryMorpho[{2,2,1,2,3}]/.mrules
```

```
{6, 7, 3, 9, 14}
```

```
RuleTableFromkAryMorpho0[{0,0,1,0,2}]/.mrules0
```

```
{1, 2, 8, 4, 14}
```

```
kAryFromRuleTableMorpho0[{{1,1,1},{1,1,2},{1,2,1},{1,2,2},{1,2,3}}]
```

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

```
kAryFromRuleTableMorpho[
DeleteDuplicates[Mod[ReLabel /@
Map[Flatten,ruleDM[{6,7,3,9,14}]/.Rule→List,1],4]]]
```

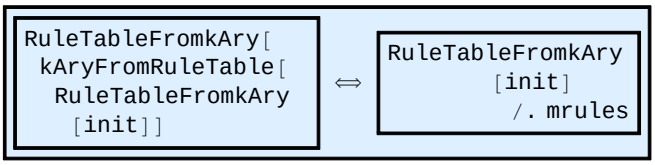
```
{2, 2, 1, 2, 3}
```

```
kAryFromRuleTableMorpho0[
DeleteDuplicates[Mod[ReLabel /@
Map[Flatten,ruleDM[{6,7,3,9,14}]/.Rule→List,1],4]]]
```

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

Properties: Commutativity

The aim is to establish the analogon of the ECA commutativity with *IntegerDigits*/*FromDigits* for a commutativity in morphoCAs with *RuleTableFromkAryMorpho/ kAryFromRuleTableMorpho:*



Direct formulas

```
RuleTableFromkAryMorpho[
kAryFromRuleTableMorpho[RuleTableFromkAryMorpho[{1,2,1,2,3}]]]==
RuleTableFromkAryMorpho[{1,2,1,2,3}]/.mrules
```

```
True
```

```
RuleTableFromkAryMorpho[
kAryFromRuleTableMorpho[RuleTableFromkAryMorpho[{1,2,1,2,3}]]]==
RuleTableFromkAryMorpho[{1,2,1,2,3}]/.mrules
```

```
True
```

```
RuleTableFromkAryMorpho[kAryFromRuleTableMorpho[
RuleTableFromkAryMorpho[{1,2,1,2,3}]]]/.mrules
```

```
{1, 7, 3, 9, 14}
```

morphoCA commutativity

```
kAryFromRuleTable[RuleTableFromkAry[init]]
      ==
kAryFromRuleTable[RuleTableFromkAry[
    kAryFromRuleTable[
RuleTableFromkAry[init]]]] /. mrules

True
```

```
kAryFromRuleTableMorpho[
DeleteDuplicates[Mod[ReLabel /@
Map[Flatten,RuleTableFromkAryMorpho[{1,2,1,2,3}]/.Rule→List,1],4]]]
```

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

Iterations

Commutativity scheme

```
{RuleTableFromkAry[kAryFromRuleTable[
 DeleteDuplicates[Mod[ReLabel /@
 Map[Flatten, ruleDM[{ruleDM}] /. Rule → List, 1], 4]]]] /. mrules}
     ==
 {{ruleDM}}

 True
```

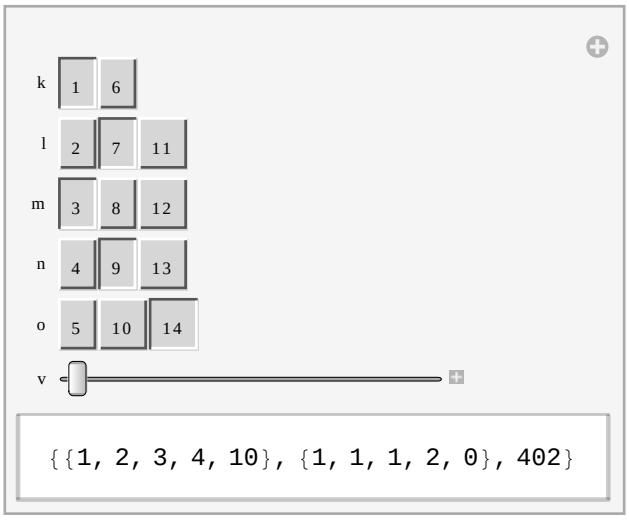
```
RuleTableFromkAryMorpho[kAryFromRuleTableMorpho[
DeleteDuplicates[Mod[ReLabel /@
Map[Flatten,ruleDM[RuleTableFromkAryMorpho[kAryFromRuleTableMorpho[
DeleteDuplicates[Mod[ReLabel /@
Map[Flatten,ruleDM[{1,7,3,9,14}]/.
Rule→List,1],4]]]]/.mrules]/.Rule→List,1],4]]]]/.mrules
```

```
{1, 7, 3, 9, 14}
```

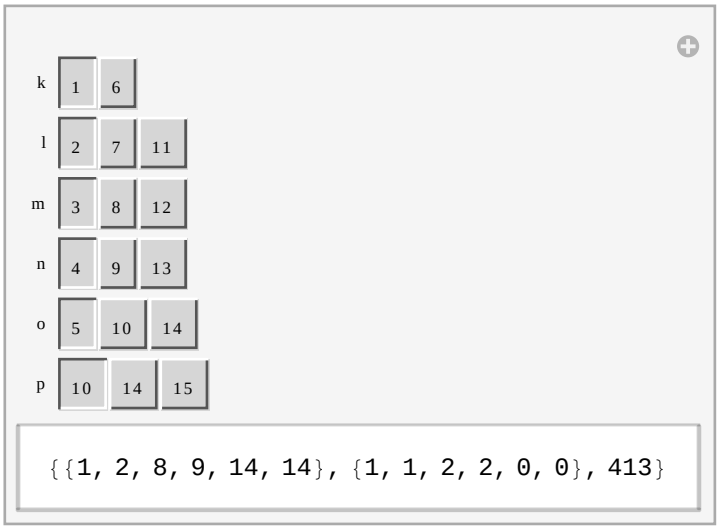
```
kAryFromRuleTableMorpho[
DeleteDuplicates[Mod[ReLabel /@
Map[Flatten,RuleTableFromkAryMorpho[kAryFromRuleTableMorpho[
DeleteDuplicates[Mod[ReLabel /@
Map[Flatten,ruleDM[RuleTableFromkAryMorpho[kAryFromRuleTableMorpho[
DeleteDuplicates[Mod[ReLabel /@
Map[Flatten,ruleDM[{1,7,3,9,14}]/.Rule→List,1],4]]]]/.mrules]/.
Rule→List,1],4]]]]/.Rule→List,1],4]/.mrules]]
```

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

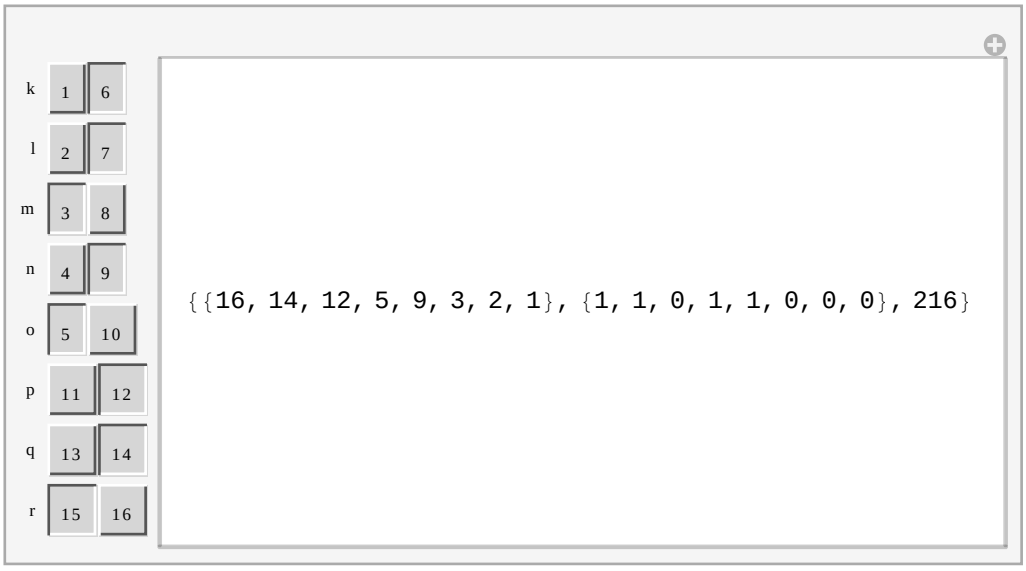
Rule DM resume



Rule DMN resume



ECA resume



First strategy: self-modification of environments

Self-reference of init2init mappings

A weak self-modification of a CA is established with a self-reference of the initial conditions by holding the rules stable.

The mapping is focused at first on just one direction: *init -> rule -> init*.

Production of the new CA on the base of the produced new initial conditions.

The reverse movement, *rule -> init -> rule*, is not yet considered. In a second approach the question shall be answered: What rules had been applied to produce the new init?

Reversible CAs

> "In special cases, a cellular automaton can run backward under the direct rule by using the final *state of the forward run as its initial state. This property is called time reversal invariant, and it is a stronger property than invertibility. Wolfram represented reversible cellular automata in which the new state of a cell is determined not only by the cell itself and its neighbors one step back but by the cell itself two steps back.*"

http://sjsu.rudyrucker.com/~kwanghyung.paek/paper/

Iterative recursion scheme : init2init

$(\text{rule}_i, \text{init}_i) \Longrightarrow (\text{init}_i = \text{init}_{i+1}, \text{init}_{i+1})$

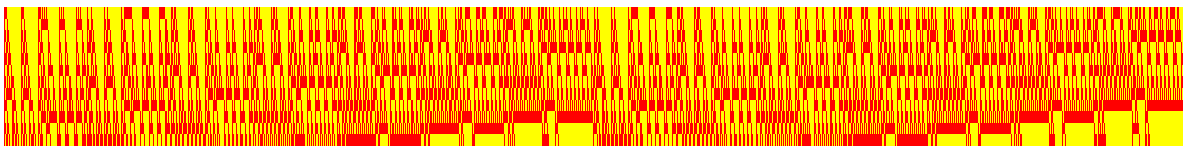
$\text{iter}(CA) = CA \to \text{init}(CA)$

Self-reference of init2init for ECA

```
ca = Flatten[CellularAutomaton[60,
{{0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1}, 0}, 6]];
```

```
ca = Flatten[CellularAutomaton[60, {ca, 0}, 6]];
```

```
ArrayPlot[ ca = CellularAutomaton[60, {ca, 0}, 11],
 ColorRules -> {1 -> Red, 0 -> Yellow},
ImageSize → 400]
```



```
Length[ca = Flatten[CellularAutomaton[60,
{{0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1}, 0}, 6]]]
```

553

```
Total[ca = Flatten[CellularAutomaton[60,
{{0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1}, 0}, 6]]]
```

166

Self-reference of init2init for morphoCA

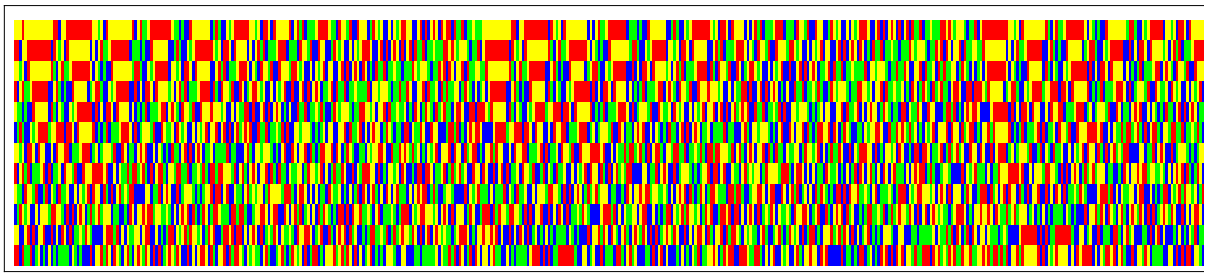
```
Manipulate[ca=CellularAutomaton[
 ruleDM[{6,3,9,11,15}],
 {{1},0}, {{{n}}}],{n,1,216,1}, SaveDefinitions→True]
```

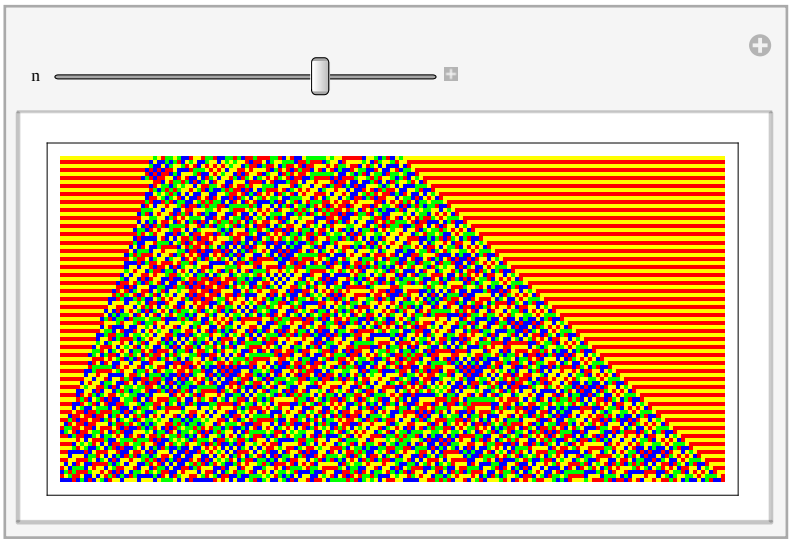
```
Clear[ca]
```

```
ca=Flatten[CellularAutomaton[
 ruleDM[{6,3,9,11,15}],
 {{1},0},11]];
```

```
ca=Flatten[CellularAutomaton[
 ruleDM[{6,3,9,11,15}],
 {ca,0},11]];
```

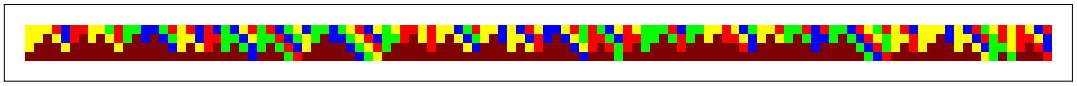
```
ArrayPlot[ca=CellularAutomaton[
 ruleDM[{6,3,9,11,15}],
 {ca,0},11],
ColorRules->{1->Red,0->Yellow,2→ Blue, 3→ Green},
ImageSize→ Medium]
```





```
IterInitRule[ca, ruleDM[{6, 3, 9, 5, 15}], 3]
```

```
IterInitRule[ca, ruleDM[{1, 8, 9, 5, 15}], 3]
```



Second strategy: self-modification of rules

ECA rule approach

Structural self-referentiality of ECA

> *"A selfish automaton uses its initial conditions as its current rule. The result is then used as the new initialization and the new rule, and so on.*
> *Most elementary 8-bit selfish automata reach fixed points quickly, but number 118 evolves 15 generations."*
> (Michael Schreiber)

http://140.177.205.90/SelfishElementaryCellularAutomata/

Examples

```
{{1, 118}, {2, 155}, {3, 115}, {4, 157}, {5, 89}, {6, 28}, {7, 18}, {8, 45},
 {9, 59}, {10, 230}, {11, 107}, {12, 247}, {13, 251}, {14, 255}, {15, 255}}
```

```
{{1, 11}, {2, 114}, {3, 157}, {4, 89}, {5, 28}, {6, 18}, {7, 45},
 {8, 59}, {9, 230}, {10, 107}, {11, 247}, {12, 251}, {13, 255}, {14, 255}}
```

Despite the dynamical complexity of ECAs their *structural* complexity of self-reference is very low.

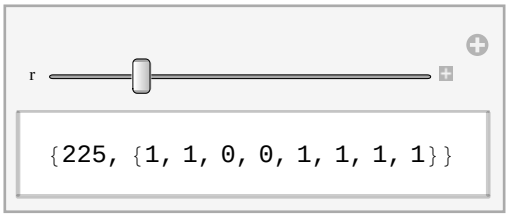
Hence the geno-type, i.e. the deep-structure of the self-referentiality of ECAs is surprisingly simple, and is in contrast to the behavioral complexity of its evolutions.

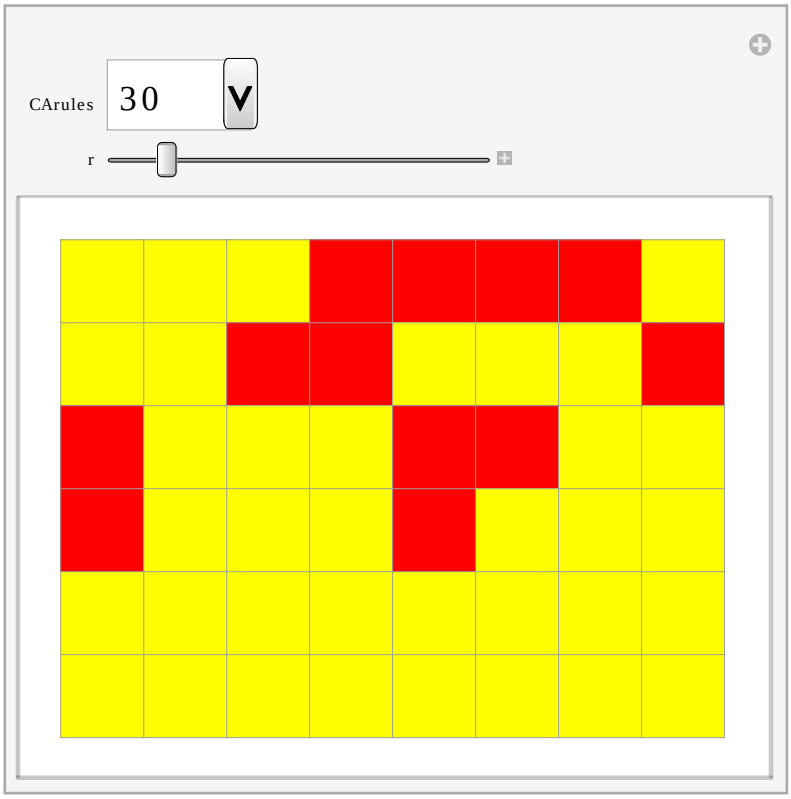
ECA commutativity

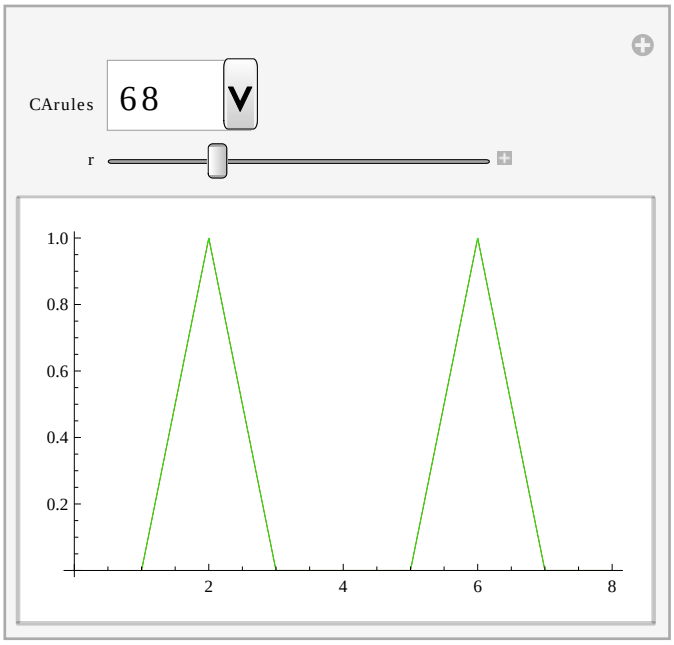
```
FromDigits[IntegerDigits[ECArule, 2], 2]
        ==
FromDigits[IntegerDigits[FromDigits[{ECAinit}, 2]]]
```

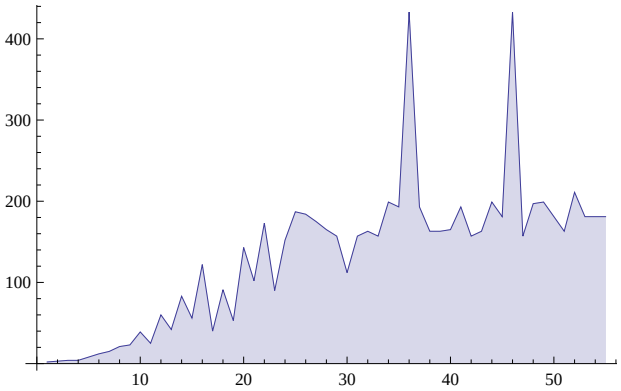
```
Table[MapIndexed[{#2[[1]], FromDigits[#, 2]} &,
  FixedPointList[CellularAutomaton[FromDigits[#, 2], #] &,
   IntegerDigits[k, 2, 8]]], {k, 0, 255}]
```

```
Manipulate[{FromDigits[CellularAutomaton[r, IntegerDigits[r, 2, 8]], 2],
  CellularAutomaton[r, {0, 0, 1, 1, 0, 0, 0, 1}]}, {r, 0, 255, 1}]
```

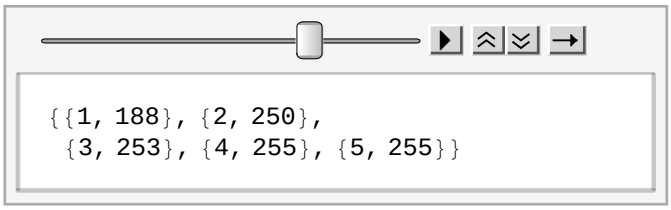






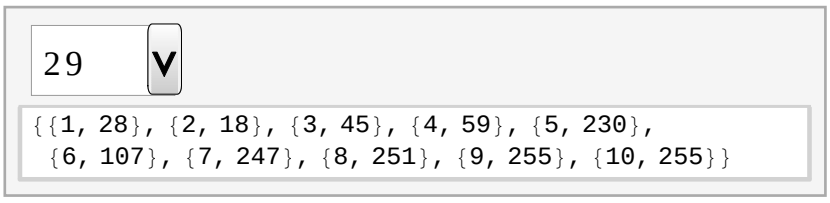



```
ListAnimate[Table[MapIndexed[{#2[[1]], FromDigits[#, 2]} &,
    FixedPointList[CellularAutomaton[FromDigits[#, 2], #] &,
     IntegerDigits[k, 2, 8]]], {k, 0, 255}], ImageSize → 300]
```



```
{{1, 188}, {2, 250},
 {3, 253}, {4, 255}, {5, 255}}
```

```
MenuView[Table[MapIndexed[{#2[[1]], FromDigits[#, 2]} &,
    FixedPointList[CellularAutomaton[FromDigits[#, 2], #] &,
     IntegerDigits[k, 2, 8]]], {k, 0, 255}], ImageSize → 400]
```

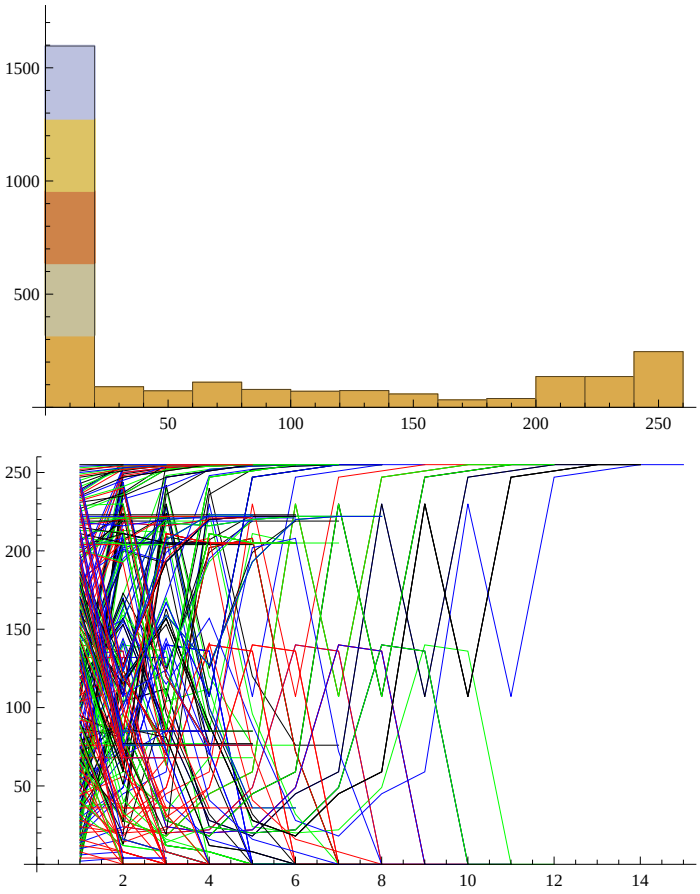



```
{{1, 28}, {2, 18}, {3, 45}, {4, 59}, {5, 230},
 {6, 107}, {7, 247}, {8, 251}, {9, 255}, {10, 255}}
```

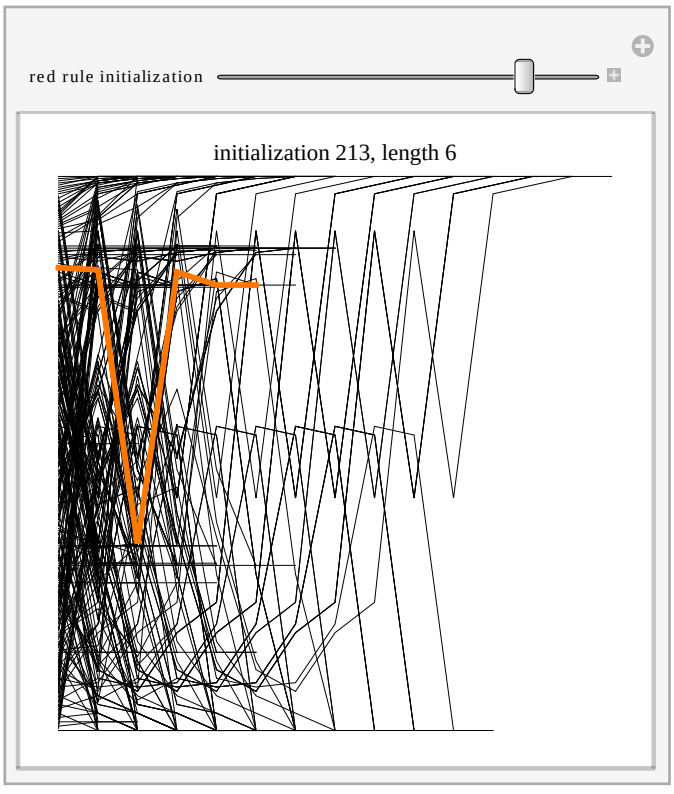
```
Table[MapIndexed[{#2[[1]], FromDigits[#, 2]} &,
  FixedPointList[CellularAutomaton[FromDigits[#, 2], #] &,
   IntegerDigits[k, 2, 8]]], {k, 0, 255}]
```

ECA example

```
{{1, 0}, {2, 0}}, {{1, 1}, {2, 124}, {3, 70}, {4, 202}, {5, 208}, {6, 72}, {7, 0}, {8, 0}},
{{1, 2}, {2, 4}, {3, 4}}, {{1, 3}, {2, 124}, {3, 70}, {4, 202}, {5, 208}, {6, 72}, {7, 0}, {8, 0}},
{{1, 4}, {2, 4}}, {{1, 5}, {2, 117}, {3, 159}, {4, 127}, {5, 193}, {6, 220}, {7, 222}, {8, 222}},
{{1, 6}, {2, 8}, {3, 0}, {4, 0}}, {{1, 7}, {2, 120}, {3, 76}, {4, 76}},
{{1, 8}, {2, 0}, {3, 0}}, {{1, 9}, {2, 96}, {3, 32}, {4, 0}, {5, 0}},
{{1, 10}, {2, 16}, {3, 8}, {4, 0}, {5, 0}}, {{1, 11}, {2, 114}, {3, 157}, {4, 89}, {5, 28},
 {6, 18}, {7, 45}, {8, 59}, {9, 230}, {10, 107}, {11, 247}, {12, 251}, {13, 255}, {14, 255}},
{{1, 12}, {2, 8}, {3, 0}, {4, 0}}, {{1, 13}, {2, 105}, {3, 240}, {4, 120}, {5, 76}, {6, 76}},
{{1, 14}, {2, 24}, {3, 20}, {4, 22}, {5, 49}, {6, 140}, {7, 136}, {8, 0}, {9, 0}},
{{1, 15}, {2, 120}, {3, 76}, {4, 76}}, {{1, 16}, {2, 8}, {3, 0}, {4, 0}}, {{1, 17}, {2, 204}, {3, 204}},
{{1, 18}, {2, 45}, {3, 59}, {4, 230}, {5, 107}, {6, 247}, {7, 251}, {8, 255}, {9, 255}},
{{1, 19}, {2, 236}, {3, 252}, {4, 254}, {5, 255}, {6, 255}}
```



Schreiber's SelfishElementaryCellularAutomata



Structural self-referentiality of morphoCA

Morphogrammatic self-reference and self-modification of morphic cellular automata is a surprisingly new and intriguing endeavor. We know self-referential structures and dynamics from languages, formal and natural, and from other semiotically based constructions.

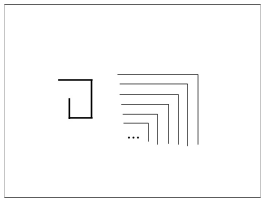
Morphograms are neither propositions, sentences, texts or semiotic constructions nor are they given by identification. Strictly speaking there is no reference of a morphogram to itself because there is no identifiable object to which a reference could refere.

In contrast to self-referential constructions in formal languages that are identified by what they are by definition, morphogrammatic structurations are pretended as simulations of hidden mechanisms.

As a slogan it could help to say, language is understood by the distinction of surface- and deep-structures or by the difference of pheno- and geno-structures. Morphogrammatics is evoked by the geno-structure of the (geno-structure of the pheno-structure) of inscriptions.

There had been two motivations for Varela to choose the *Calculus of Forms* for a theory of living systems:
1. self-referentiality via re-enrtry and
2. a symbolization ‘deeper than truth’.

Theorem 1 : Every recursion has a fixed point.



Bi - directional unfolding

Here, we have allowed the arrows to go end-to-end in two directions. In condensed form we have

$$b = \ldots > > > > > > > > > > > > > > > \ldots$$

The infinite form b has a left–right translational symmetry. If T is the operation of shifting one (arrow) unit to the right, then we can say

$$T(b) = b.$$

This is invariance under a symmetry operation, the same sort of invariance seen every day in looking at frieze patterns and decorated borders.

http : // www.gwu.edu/~rpsol/preconf/wmsci/kaufman2.pdf

Palindromes

The structure of fixed points is circular, the structure of morphic fixed-points is palindromic.

While fixedpoints have a uni-linear circularity, morphic fixed points are bidirectionally circular, hence palindromic.

The form of uni-directional fixedpoint is re-entry.
The morphogrammatics of fixedpoints is a chiastic palindrome.

Therefore, *“snakes all the way down”* is a metaphor for a restricted economy of thinking and living.

All fixedpoint structures are morphogrammatically palindromic. But just a few fixedpoints with a very low complexity are mathematically palindromic.

This is in contrast to Kauffman’s statement of *bi-directional symmetry of fixed-points*. Hence, there is an additional abstraction involved in Kauffman’s argument: the “*invariance* under symmetry operations”.

What is the difference of *“bi-directional unfolding”* of fixedpoints (Kauffman) and morphogrammatics of fixed-points as palindromes?

Fixedpoints might abstractly be conceived as bi-directionally symmetric but morphic palindromes are *bi-symmetrical and* not just symmetrical.

On the Cybernetics of Fixed Points
Louis H. Kauffman

http://www.imprint.co.uk/C&HK/vol8/asc_8_1.pdf

Theorem 2 : Every fixed point has a morpic palindrome.

Simulation

A working result is given in the framework of the self-reference scheme and the procedures: *kAryFromRuleTable* and *RuleTableFromkAry* in connection with *ReLabel* and *mrules.*

http : // www.mathematica - journal.com/data/uploads/2014/07/Oliveira_Representing.nb

The example schows what could be called Eigenvalues for morphoCAs. They are mainly loops of some length and not necessarily simple fixed-points of an evolution.

The difference between information processing *Eigenvalue* systems (von Foerster) and morphic self-referentiality (Gunther) lies in the *structure* of the process. Information is still closely connected to logic and semiotics, morphogrammatic systems as they have been introduced by Gotthard Gunther in his paper *"Cybernetic Ontology and Transjunctional Operations"* (1962) are pre-logical structurations of a higher complexity than logic is able to cover.

It is not necessary to confuse the preliminary *modeling* of morphogrammatic dynamics in a traditional framework with the *intention* to surpass the limits of traditional computation and logic by the project of morphogrammatics itself.

Semi-automated solutions for ruleDM

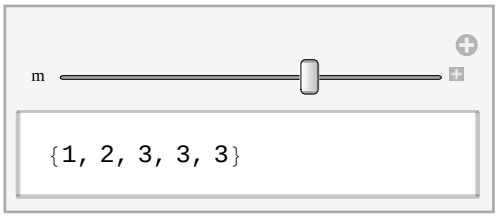
Fixed starting point

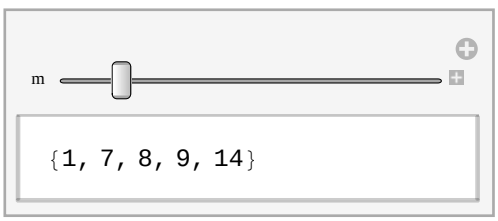
For technical reasons there are two different implementations proposed, one with rules based on lists starting with "0", i.e. the *mrules0*, and one based on lists starting with "1", i.e. the *mrules,* and the implications for the formulations of the *procedures* (*RuleTableFromkAryMorpho* and *kAryFromRuleTableMorpho*).

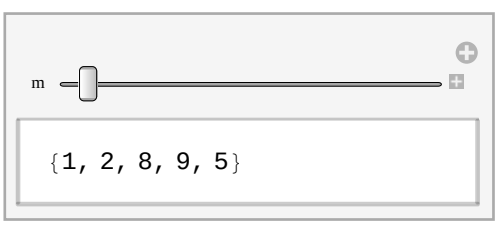
Also the approach works it is still omtitting the morphogram [15].

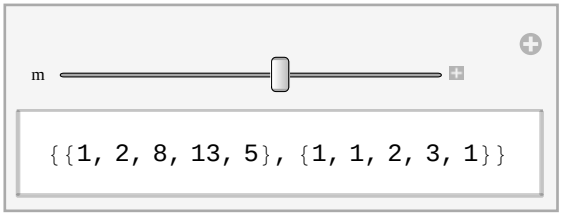
The results seems to be different. Notational difference between mrules and mrules0.

```
Manipulate[ReLabel[Mod[Flatten[CellularAutomaton[
ruleDM[{1,7,8,9,14}],{1,2,2,2,3},{{m}}]],4,2]], {m,1,222,1},
SaveDefinitions→True]
```





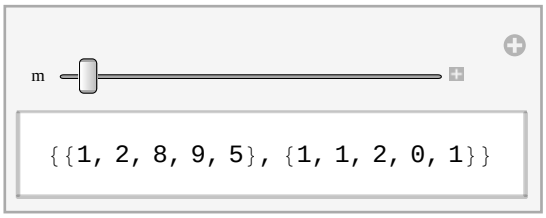





```
Table[{RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[
ruleDM[{1,7,8,9,14}],{1,2,2,2,3},{{m}}]],2]]/.mrules0,
Mod[ReLabel[Flatten[CellularAutomaton[
ruleDM[{1,7,8,9,14}],{0,1,1,1,2},{{m}}]]],3]},{m,1,216,1}]
```

Examples for *mrules0*

```
{{1, 2, 3, 9, 10}, {1, 1, 1, 2, 0}}, {{1, 2, 8, 9, 5}, {1, 1, 2, 0, 1}},
{{1, 7, 8, 4, 5}, {1, 2, 0, 1, 1}}, {{6, 7, 3, 4, 5}, {1, 2, 0, 0, 0}},
{{6, 2, 3, 4, 10}, {1, 2, 2, 2, 0}}, {{1, 2, 3, 9, 10}, {1, 1, 1, 2, 0}},
{{1, 2, 8, 9, 5}, {1, 1, 2, 0, 1}}, {{1, 7, 8, 4, 5}, {1, 2, 0, 1, 1}},
{{6, 7, 3, 4, 5}, {1, 2, 0, 0, 0}}, {{6, 2, 3, 4, 10}, {1, 2, 2, 2, 0}},
{{1, 2, 3, 9, 10}, {1, 1, 1, 2, 0}}, {{1, 2, 8, 9, 5}, {1, 1, 2, 0, 1}},
{{1, 7, 8, 4, 5}, {1, 2, 0, 1, 1}}, {{6, 7, 3, 4, 5}, {1, 2, 0, 0, 0}},
{{6, 2, 3, 4, 10}, {1, 2, 2, 2, 0}}, {{1, 2, 3, 9, 10}, {1, 1, 1, 2, 0}},
```




```
RuleTableFromkAryMorpho[{1,2,3,3,3}]/.mrules
```

```
{1, 7, 12, 13, 14}
```

```
ReLabel[Mod[Flatten[CellularAutomaton[ruleDM[{1,7,12,13,14}],{1,2,2,2,3}]],3]]
```

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

```
Table[{RuleTableFromkAryMorpho[ReLabel[Flatten[
CellularAutomaton[
ruleDM[{1,7,8,9,14}],{1,2,2,2,3},{{m}}]]]]/.mrules,
ReLabel[Mod[Flatten[CellularAutomaton[
ruleDM[{1,7,8,9,14}],{1,2,2,2,3},{{m}}]],3,2]]/.mrules},{m,1,216,1}]
```

Examples for *mrules*

```
{{1, 2, 3, 9, 14}, {1, 1, 1, 2, 3}}, {{1, 2, 8, 13, 5}, {1, 1, 2, 3, 1}},
{{1, 7, 12, 4, 5}, {1, 2, 3, 1, 1}}, {{1, 7, 12, 13, 14}, {1, 2, 3, 3, 3}},
{{1, 7, 8, 9, 14}, {1, 2, 2, 2, 3}}, {{1, 2, 3, 9, 14}, {1, 1, 1, 2, 3}},

{{1, 2, 8, 13, 5}, {1, 1, 2, 3, 1}}, {{1, 7, 12, 4, 5}, {1, 2, 3, 1, 1}},
{{1, 7, 12, 13, 14}, {1, 2, 3, 3, 3}}, {{1, 7, 8, 9, 14}, {1, 2, 2, 2, 3}},
{{1, 2, 3, 9, 14}, {1, 1, 1, 2, 3}}, {{1, 2, 8, 13, 5}, {1, 1, 2, 3, 1}},

{{1, 7, 12, 4, 5}, {1, 2, 3, 1, 1}}, {{1, 7, 12, 13, 14}, {1, 2, 3, 3, 3}},
{{1, 7, 8, 9, 14}, {1, 2, 2, 2, 3}}, {{1, 2, 3, 9, 14}, {1, 1, 1, 2, 3}},
{{1, 2, 8, 13, 5}, {1, 1, 2, 3, 1}}, {{1, 7, 12, 4, 5}, {1, 2, 3, 1, 1}},

{{1, 7, 12, 13, 14}, {1, 2, 3, 3, 3}}, {{1, 7, 8, 9, 14}, {1, 2, 2, 2, 3}},
{{1, 2, 3, 9, 14}, {1, 1, 1, 2, 3}}, {{1, 2, 8, 13, 5}, {1, 1, 2, 3, 1}},
{{1, 7, 12, 4, 5}, {1, 2, 3, 1, 1}}, {{1, 7, 12, 13, 14}, {1, 2, 3, 3, 3}}
```

```
Table[{RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[
ruleDM[{1,7,8,9,14}],{1,2,2,2,3},{{m}}]],2]]/.mrules0,
Mod[ReLabel[Flatten[CellularAutomaton[
ruleDM[{1,7,8,9,14}],{0,1,1,1,2},{{m}}]]],3]},{m,1,216,1}]
```

```
RuleTableFromkAryMorpho0[{0,1,2,2,2}]/.mrules0
```

```
{1, 7, 12, 13, 14}
```

```
Mod[ReLabel[Flatten[CellularAutomaton[
ruleDM[{1,7,12,13,14}],{0,1,2,2,2}]]],3]
```

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

```
RuleTableFromkAryMorpho0[{1,2,0,2,0}]/.mrules0
```

```
{6, 11, 3, 13, 5}
```

```
Mod[ReLabel[Flatten[CellularAutomaton[
ruleDM[{6,11,3,13,5}],{0,1,2,2,2}]]],3]
```

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

```
RuleTableFromkAryMorpho0[{1,2,2,2,0}]/.mrules0
```

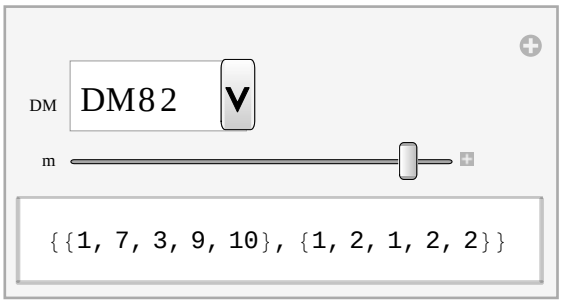
```
{6, 11, 12, 13, 5}
```

Examples0

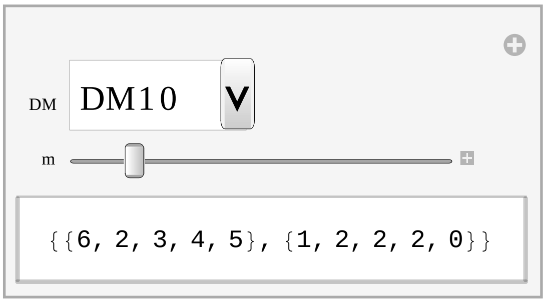
```
{{1, 2, 3, 9, 10}, {1, 1, 1, 2, 0}},
{{1, 2, 8, 9, 5}, {1, 1, 2, 0, 1}},
{{1, 7, 8, 4, 5}, {1, 2, 0, 1, 1}},
{{6, 7, 3, 4, 5}, {1, 2, 0, 0, 0}},
{{6, 2, 3, 4, 10}, {1, 2, 2, 2, 0}},
{{1, 2, 3, 9, 10}, {1, 1, 1, 2, 0}},

{{1, 2, 8, 9, 5}, {1, 1, 2, 0, 1}},
{{1, 7, 8, 4, 5}, {1, 2, 0, 1, 1}},
{{6, 7, 3, 4, 5}, {1, 2, 0, 0, 0}},
{{6, 2, 3, 4, 10}, {1, 2, 2, 2, 0}}, {{1, 2, 3, 9, 10}, {1, 1, 1, 2, 0}}
```

Variable starting points

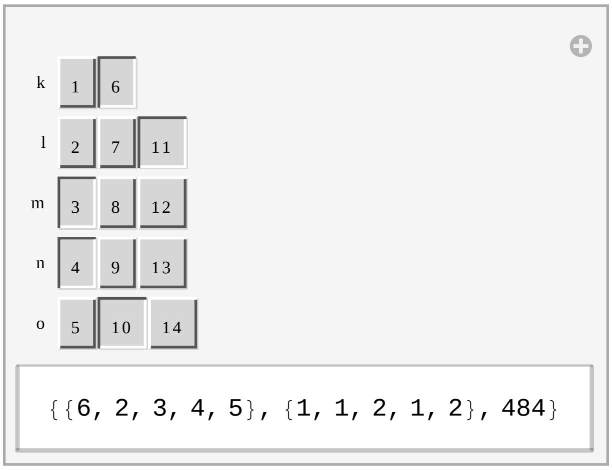
With *mrules*



With *mrules0*



Resume: sub-rule DM, environment and rule number

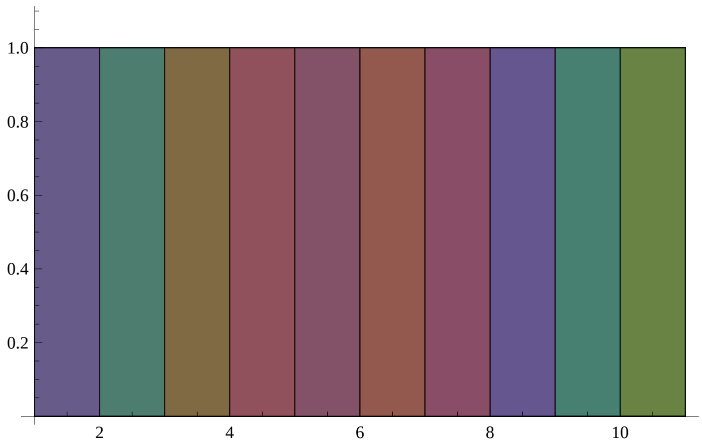



```
Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,14}],{0,1,1,1,2},{{m}}]],2]]/.
mrules0,{m,1,261,1}]
```

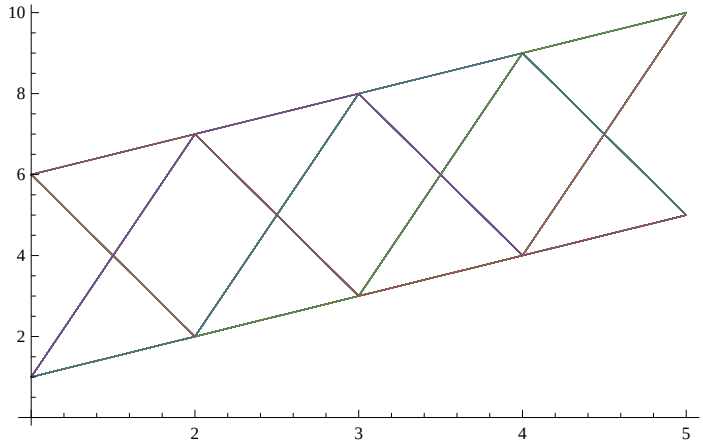
Examples with mrules0

```
{6, 7, 3, 4, 5}, {6, 2, 3, 4, 10}, {1, 2, 3, 9, 10}, {1, 2, 8, 9, 5}, {1, 7, 8, 4, 5},
{6, 7, 3, 4, 5}, {6, 2, 3, 4, 10}, {1, 2, 3, 9, 10}, {1, 2, 8, 9, 5}, {1, 7, 8, 4, 5},
{6, 7, 3, 4, 5}, {6, 2, 3, 4, 10}, {1, 2, 3, 9, 10}, {1, 2, 8, 9, 5}, {1, 7, 8, 4, 5},
{6, 7, 3, 4, 5}, {6, 2, 3, 4, 10}, {1, 2, 3, 9, 10}, {1, 2, 8, 9, 5}, {1, 7, 8, 4, 5},
{6, 7, 3, 4, 5}, {6, 2, 3, 4, 10}, {1, 2, 3, 9, 10}, {1, 2, 8, 9, 5}, {1, 7, 8, 4, 5}, {6, 7, 3, 4, 5}
```

```
Histogram[Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,14}],{0,1,1,1,2},{{m}}]],2]]/.
mrules0,{m,1,261,1}]]
```



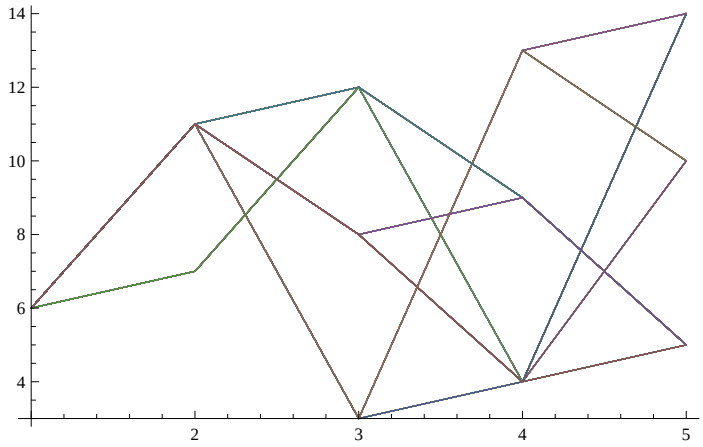
```
ListLinePlot[Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,14}],{0,1,1,1,2},{{m}}]],2]]/.mrules0,{m,1,261,1}]]
```



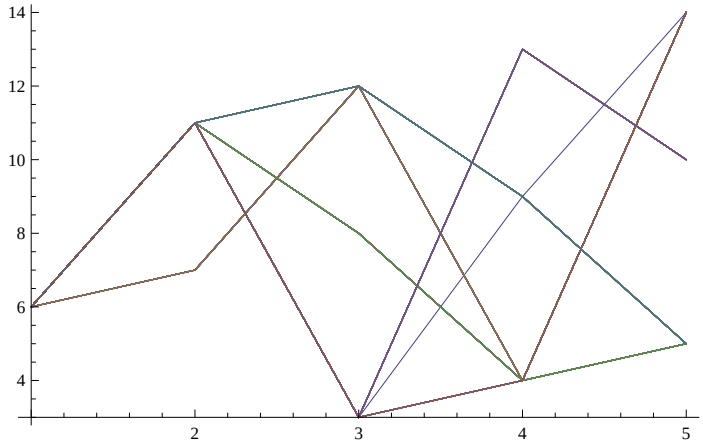
Examples with mrules0

```
{6, 7, 3, 4, 5},
{6, 2, 3, 4, 10},
{1, 2, 3, 9, 10},
{1, 2, 8, 9, 5},
{1, 7, 8, 4, 5}, {6, 7, 3, 4, 5}
```

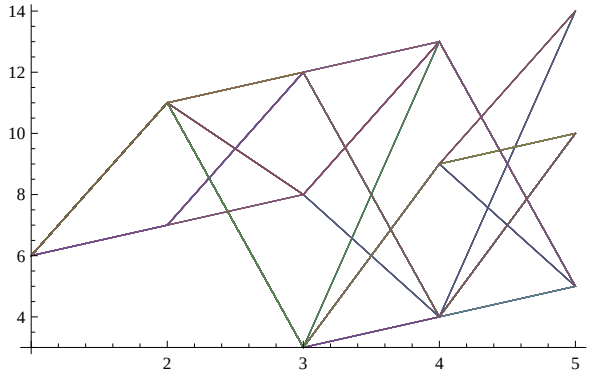
```
ListLinePlot[Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,10}],{0,1,1,1,2},{{m}}]]],3]]/.
mrules0,{m,1,261,1}]]
```



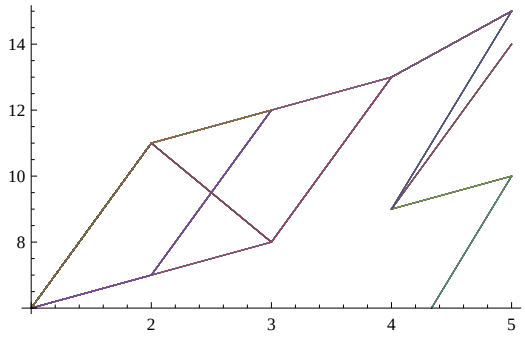
```
ListLinePlot[Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,5}],{0,1,1,1,2},{{m}}]]],3]]/.
mrules0,{m,1,261,1}]]
```




```
ListLinePlot[Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{0,1,1,1,2},{{m}}]]],3]]/.
mrules0,{m,1,261,1}]]
```




```
ListLinePlot[Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{0,1,1,1,2},{{m}}]]],4]]/.
mrules0,{m,1,261,1}]]
```



Interplay between ruleDM and init

The interplay between rules and initial conditions is building chains of different length that are structured by fixed points and loops.

From a more conceptual point of view a modeling of the chiasm between *rules* and *initial* conditions has to distinguish between a *hierarchical* and a *heterarchical* approach. In other words, the self-referential process between *rules* and *inits* can be modeled in a “one-person” or a “multi-person” approach (e.g. Abramsky). The *“beliieve/assume”*-

structure is naturally mapped onto the *rule/initit*-structure or the system/environment-structure.

```
RuleTableFromkAryMorpho[init1] /. mrules  -> [RuleNr1]

ReLabel[Mod[Flatten[CellularAutomaton[[RuleNr1], {init1}]], 3]] -> [init2]

RuleTableFromkAryMorpho[init2] /. mrules  -> [RuleNr2]
```

Hierarchical modeling

Hierarchical modeling of self-reference is modeling the reference in the mode of 'time'.

```
morphoCA : rule0 , init0
   ⇓
 init1 ⟹ rule1
          ⇓
 rule2 ⟸ init2
   ⇓
 init3 ⟹ rule3
```

```
rule0 = {1, 7, 8, 9, 14}, init0 = {1, 2, 2, 2, 3}

init1 = ReLabel[Mod[Flatten[CellularAutomaton[
ruleDM[{1, 7, 8, 9, 14}], {1, 2, 2, 2, 3}]], 3]]     :: init1 = {1, 1, 1, 2, 3}

   init1 = {1, 1, 1, 2, 3} ⟹
   RuleTableFromkAryMorpho[{1, 1, 1, 2, 3}] /. mrules :: rule1 {1, 2, 3, 9, 14}

rule1 {1, 2, 3, 9, 14} ⟹
ReLabel[Mod[Flatten[CellularAutomaton[
ruleDM[{1, 2, 3, 9, 14}], {1, 1, 1, 2, 3}]], 3]]     :: init2 = {1, 1, 1, 2, 1}

   init2 ⟹
   RuleTableFromkAryMorpho[{1, 1, 1, 2, 1}] /. mrules :: rule2 = {1, 2, 3, 9, 5}
```

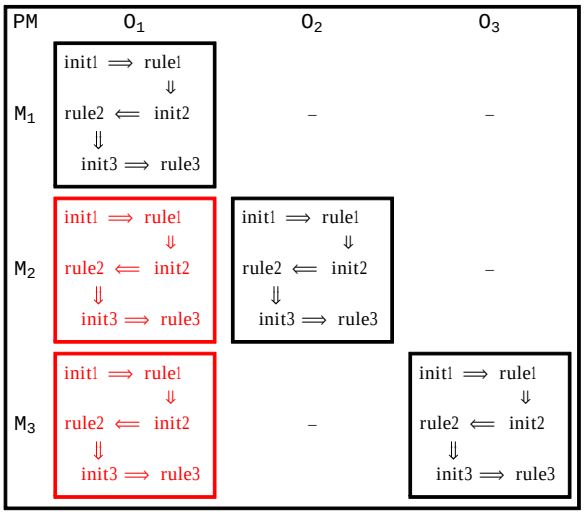
```
rule0 = {1, 7, 8, 9, 14} , init0 = {1, 2, 2, 2, 3}
  ⇓
init1 = {1, 1, 1, 2, 3} ⟹ rule1 {1, 2, 3, 9, 14}
                           ⇓
rule2 = {1, 2, 3, 9, 5} ⟸ init2 = {1, 1, 1, 2, 1}
```

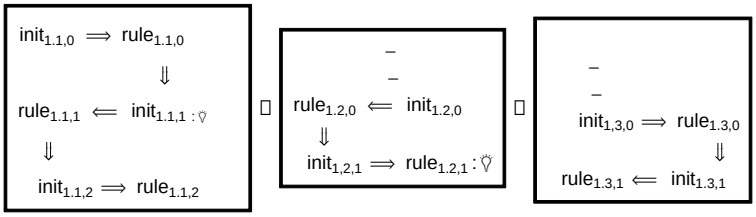
Heterarchical modeling

Heterarchical modeling of self-reference is modeling the reference in the mode of 'space'.

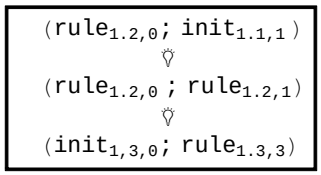
Matrix for replication (□) of $\mathbf{S_{1.1}}$

PM	O_1	O_2	O_3
M_1	$S_{1.1}$	x	x
M_2	$S_{1.2}$	$S_{2.2}$	x
M_3	$S_{1.3}$	x	$S_{3.3}$



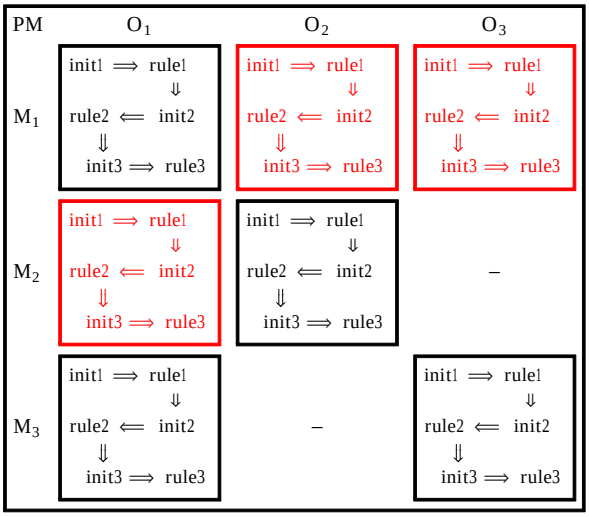


Heterarchical chain for the replication of $S_{1.1}$:



Matrix for transposition of **$S_{1.1}$** and replication of **$S_{1.1}$**

PM	O_1	O_2	O_3
M_1	$S_{1.1}$	$S_{2.1}$	$S_{3.1}$
M_2	$S_{1.2}$	$S_{2.2}$	–
M_3	$S_{1.3}$	–	$S_{3.3}$



Matrix for 'fractalization' of **$S_{2.1}$** and replication **$S_{2.2}$**, **$S_{3.3}$**

PM	O_1	O_2	O_3
M_1	$S_{1.1}$	$S_{2.1}$	–
M_2	$S_{1.2,2.,2}$	$S_{2.2}$	$S_{3.2}$
M_3	–	$S_{2.3}$	$S_{3.3}$

PM	O_1	O_2	O_3
M_1	init1 ⟹ rule1 ⇓ rule2 ⟸ init2 ⇓ init3 ⟹ rule3	init1 ⟹ rule1 ⇓ rule2 ⟸ init2 ⇓ init3 ⟹ rule3	–
M_2	init1 ⟹ rule1 ⇓ rule2 ⟸ init2 ⇓ init3 ⟹ rule3 *i*	init1 ⟹ rule1 ⇓ rule2 ⟸ init2 ⇓ init3 ⟹ rule3	init1 ⟹ rule1 ⇓ rule2 ⟸ init2 ⇓ init3 ⟹ rule3
M_3	–	init1 ⟹ rule1 ⇓ rule2 ⟸ init2 ⇓ init3 ⟹ rule3	init1 ⟹ rule1 ⇓ rule2 ⟸ init2 ⇓ init3 ⟹ rule3

Examples of iterative morphoCA loops and self-loops

DM145 : loop

```
{{1,7,3,4,14},{1,2,1,1,3}}
{{1,7,12,9,10},{1,2,3,2,2}}
{{1,7,3,13,5},{1,2,1,3,1}}
{{1,2,8,4,14},{1,1,2,1,3}}
{{1,7,8,13,10},{1,2,2,3,2}}
{{1,7,3,4,14},{1,2,1,1,3}}
```

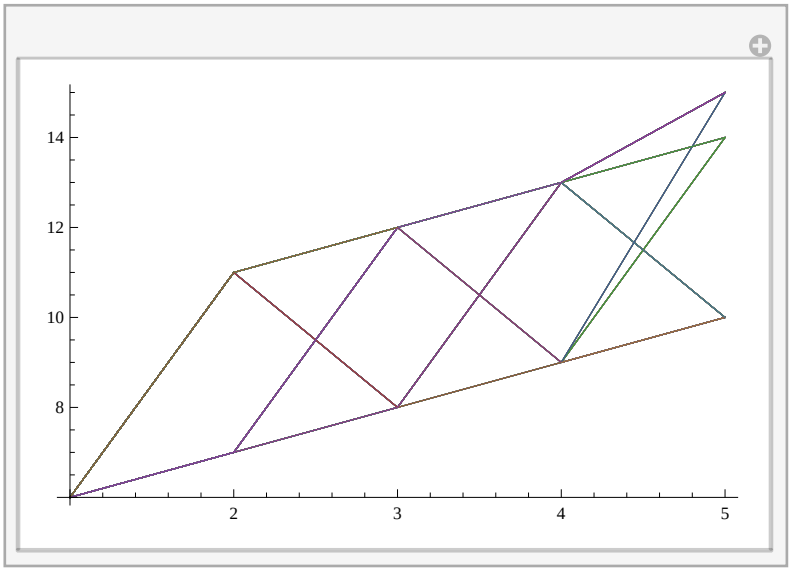
DM146 : self - loop (flag)

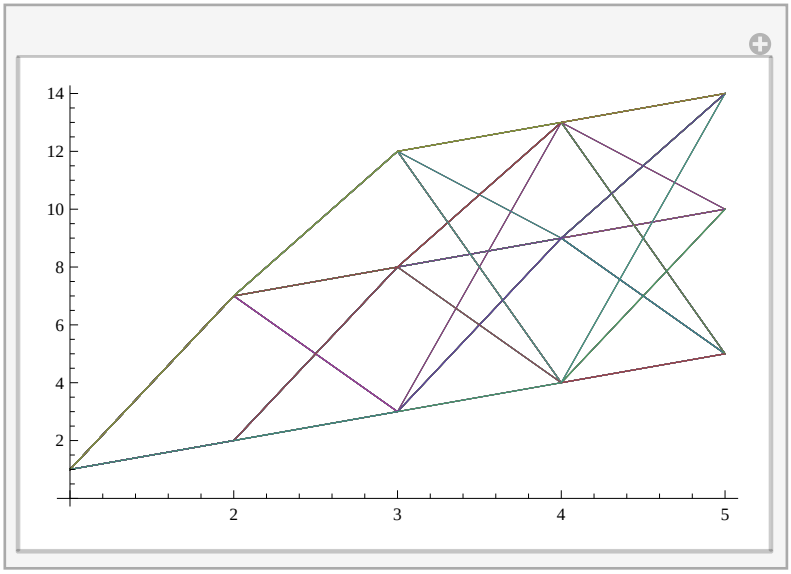
```
{1,2,8,9,10}
{1,2,3,4,10}
{1,2,8,9,10}
{1,2,8,9,10}
```

DM216: self-loop

```
{1,2,3,4,5}
{1,2,3,4,5}
```

ListLinePlots,Tables and Palindromes





MorphoLoop0 and palindromes

```
Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,2},{{m}}]],3]]/.
mrules0,{m,1,161,1}]
```

```
Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,2},{{m}}]],3]]/.
mrules0,{m,1,161,1}]/.filterM
```

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

```
DeleteDuplicates[Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,2},{{m}}]],3]]/.
mrules0,{m,1,161,1}]/.filterM]
```

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

```
ReLabel[
  Reverse[{"A", "B", "C", "D", "E", "F", "G", "H"}]] ==
  ReLabel[{"A", "B", "C", "D", "E", "F", "G", "H"}]
```

```
True
```

```
Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,14}],{0,1,1,1,2},{{m}}]],2]]/.mrules0,{m,1,161,1}]//
MatrixForm
```

```
 6 7 3 4  5    6 2 3 4 10   1 2 3 9 10   1 2 8 9  5    1 7 8 4  5    6 7 3 4  5
 6 2 3 4 10    1 2 3 9 10   1 2 8 9  5    1 7 8 4  5    6 7 3 4  5    6 2 3 4 10
 1 2 3 9 10    1 2 8 9  5   1 7 8 4  5    6 7 3 4  5    6 2 3 4 10    1 2 3 9 10
 1 2 8 9  5    1 7 8 4  5   6 7 3 4  5    6 2 3 4 10    1 2 3 9 10    1 2 8 9  5
 1 7 8 4  5    6 7 3 4  5   6 2 3 4 10    1 2 3 9 10    1 2 8 9  5    1 7 8 4  5
 6 7 3 4  5    6 2 3 4 10   1 2 3 9 10    1 2 8 9  5    1 7 8 4  5    6 7 3 4  5
```

```
Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,2},{{m}}]],2]]/.mrules0,{m,1,161,1}]//
MatrixForm
```

```
6 7 8 4 10   6 7 3 4 10   6 2 8 9  5   6 2 3 4  5
6 7 3 4 10   6 2 8 9  5   6 2 3 4  5   6 2 3 4 10
6 2 8 9  5   6 2 3 4  5   6 2 3 4 10   1 2 3 9 10
6 2 3 4  5   6 2 3 4 10   1 2 3 9 10   1 2 8 4  5
6 2 3 4 10   1 2 3 9 10   1 2 8 4  5   1 7 8 9  5
1 2 3 9 10   1 2 8 4  5   1 7 8 9  5   6 7 8 4 10
1 2 8 4  5   1 7 8 9  5   6 7 8 4 10   6 7 3 4 10
1 7 8 9  5   6 7 8 4 10   6 7 3 4 10   6 2 8 9  5
6 7 8 4 10   6 7 3 4 10   6 2 8 9  5   6 2 3 4  5

 6 2 3 4 10   1 2 3 9 10   1 2 8 4  5   1 7 8 9  5   6 7 8 4 10
 1 2 3 9 10   1 2 8 4  5   1 7 8 9  5   6 7 8 4 10   6 7 3 4 10
 1 2 8 4  5   1 7 8 9  5   6 7 8 4 10   6 7 3 4 10   6 2 8 9  5
 1 7 8 9  5   6 7 8 4 10   6 7 3 4 10   6 2 8 9  5   6 2 3 4  5
 6 7 8 4 10   6 7 3 4 10   6 2 8 9  5   6 2 3 4  5   6 2 3 4 10
 6 7 3 4 10   6 2 8 9  5   6 2 3 4  5   6 2 3 4 10   1 2 3 9 10
 6 2 8 9  5   6 2 3 4  5   6 2 3 4 10   1 2 3 9 10   1 2 8 4  5
 6 2 3 4  5   6 2 3 4 10   1 2 3 9 10   1 2 8 4  5   1 7 8 9  5
 6 2 3 4 10   1 2 3 9 10   1 2 8 4  5   1 7 8 9  5   6 7 8 4 10
```

```
Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,2},{{m}}]],2]]/.
mrules0,{m,1,161,1}]/.filterM
```

```
Print[DeleteDuplicates[Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,2},{{m}}]],2]]/.
mrules0,{m,1,161,1}]/.filterM]]
```

```
{N, O, P, Q, R, S, T, U}
```

Palindrome

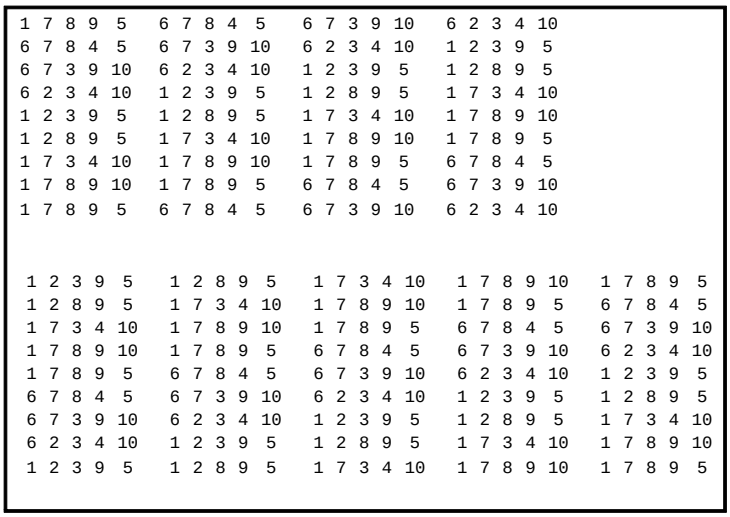
```
ReLabel[
  Reverse[{"N", "O", "P", "Q", "R", "S", "T", "U"}]] ==
  ReLabel[{"N", "O", "P", "Q", "R", "S", "T", "U"}]
```

```
True
```

```
ReLabel[Reverse[DeleteDuplicates[Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,2},{{m}}]],2]]/.
mrules0,{m,1,161,1}]/.filterM]]==
ReLabel[DeleteDuplicates[Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,2},{{m}}]],2]]/.
mrules0,{m,1,161,1}]/.filterM]]
```

```
True
```

```
Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,3},{{m}}]],2]]/.
mrules0,{m,1,161,1}]//
MatrixForm
```

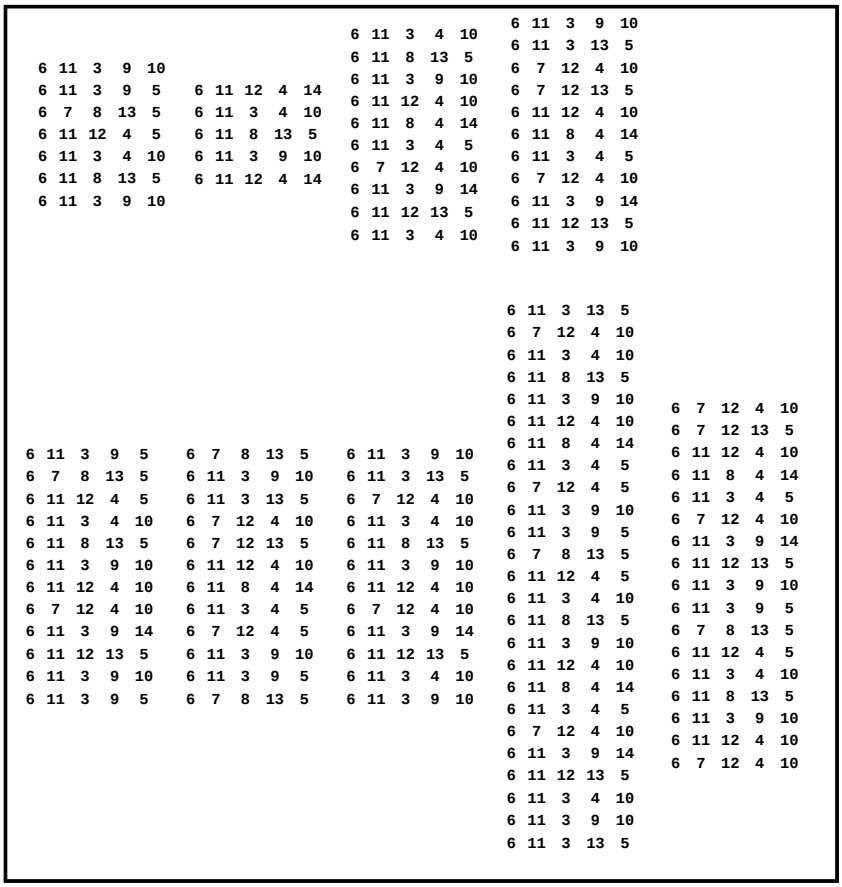


```
DeleteDuplicates[Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{1,7,8,9,15}],{0,1,1,1,3},{{m}}]],2]]/.
mrules0,{m,1,161,1}]/.filterM]
```

{U, k, l, R, m, n, o, p}

MorphoLoop with rules *mrules0*

```
Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{0,1,1,1,2},{{m}}]]],3]]/.
mrules0,{m,1,261,1}]
```



```
Print[Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{0,1,1,1,2},{{m}}]]],3]]/.
mrules0,{m,1,261,1}]/.filterM]
```

```
{V, W, X, Y, Z, a, V, b, Z, a, V, b, Z, a, V, c, d, e, f, C, g, Z, V, h, f, i, c, d, e,
 f, C, g, V, W, X, Y, Z, a, V, c, f, C, g, V, W, X, V, h, f, i, c, d, e, B, V, W, X, V,
 h, f, Z, a, V, c, f, C, g, Z, V, h, f, Z, a, V, c, d, e, B, V, W, X, Y, Z, a, V, c, d,
 e, f, C, g, Z, V, h, f, i, c, d, e, f, C, g, V, W, X, Y, Z, a, V, c, f, C, g, V, W, X,
 V, h, f, i, c, d, e, B, V, W, X, V, h, f, Z, a, V, c, f, C, g, Z, V, h, f, Z, a, V, c,
 d, e, B, V, W, X, Y, Z, a, V, c, d, e, f, C, g, Z, V, h, f, i, c, d, e, f, C, g, V, W,
 X, Y, Z, a, V, c, f, C, g, V, W, X, V, h, f, i, c, d, e, B, V, W, X, V, h, f, Z, a, V,
 c, f, C, g, Z, V, h, f, Z, a, V, c, d, e, B, V, W, X, Y, Z, a, V, c, d, e, f, C, g, Z,
 V, h, f, i, c, d, e, f, C, g, V, W, X, Y, Z, a, V, c, f, C, g, V, W, X, V, h, f, i, c}
```

```
DeleteDuplicates[Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{0,1,1,1,2},{{m}}]]],3]]/.
mrules0,{m,1,261,1}]/.filterM]
```

```
{V, W, X, Y, Z, a, b, c, d, e, f, C, g, h, i, B}
```

```
Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{1,2,2,2,3},{{m}}]]],3]]/.
mrules0,{m,1,261,1}]//MatrixForm
```

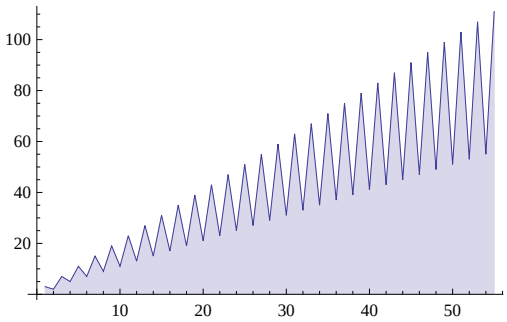
```
Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{0,1,1,1,2},{{m}}]]],3]]/.
mrules0,{m,1,261,1}]//MatrixForm
```

```
Table[RuleTableFromkAryMorpho0[Mod[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{1,1,1,1,3},{{m}}]],2]]/.
mrules0,{m,1,261,1}]//MatrixForm
```

```
Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{1,1,1,1,3},{{m}}]]],2]]/.
mrules0,{m,1,261,1}]//MatrixForm
```

```
Table[RuleTableFromkAryMorpho0[Mod[ReLabel[Flatten[
CellularAutomaton[ruleDM[{6,7,8,9,15}],{1,1,1,1,3},{{m}}]]],3]]/.
mrules0,{m,1,261,1}]//MatrixForm
```

```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleDM[{6, 7, 8, 9, 15}]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```



Structural self-referentiality of sub-rule ECA

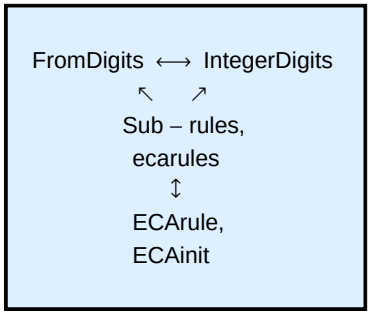
The approach to manipulate the sub-rules of a ECA or more interestingly of the 'micro-analysis' of the morphoRules offers a new method for an implementation of the self-modification of automata rules.

It should be clear that the ECA sub-rules are always *reducible* to total not partitioned ECA rules.

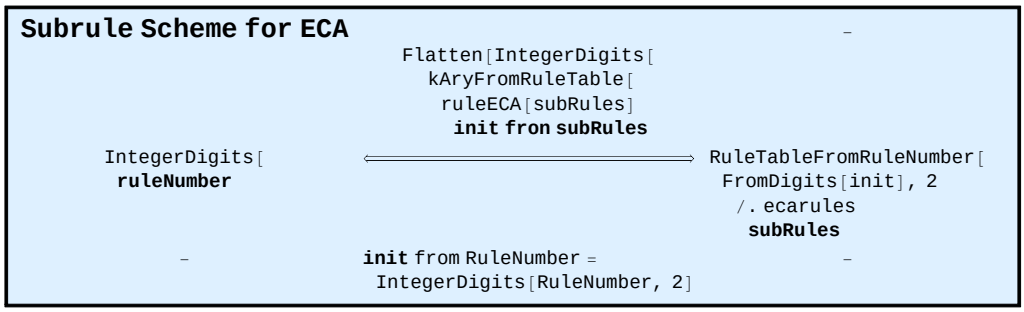
This doesn't hold in the same sense for morphoCA rules because the rules of morphoCAs are structurally irreducible. They are composites of morphograms. Only in an arbitrary and abstract sense they might be represented by a kind of total functions.

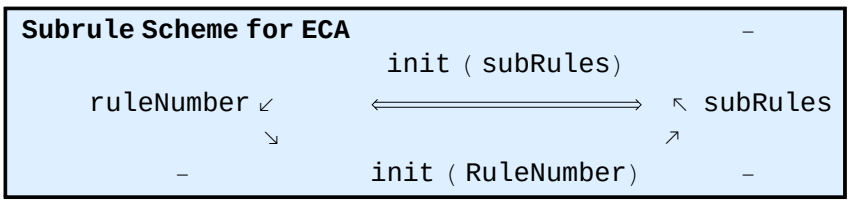
Thus the introduction of sub-rules for ECAs is a first preparation to understand morphoCAs. ECA sub-rules are thus perceived and treated as semiotic forms that are still ruled by the identity of its signs, i.e. states. Therfore all 8 patterns of a 1 DCA have to be considered. They correspond to the 4 classical positions for the morphograms [1] to [4] and [6] to [9] involved in ruleDM based morphoCAs.

Interplay of FromDigits and IntegerDigits for ECA subrules mediated by the ecarules.



Subrule Scheme for ECA





First, an external manipulation of the sub-rules is sketched.

ECA subrules

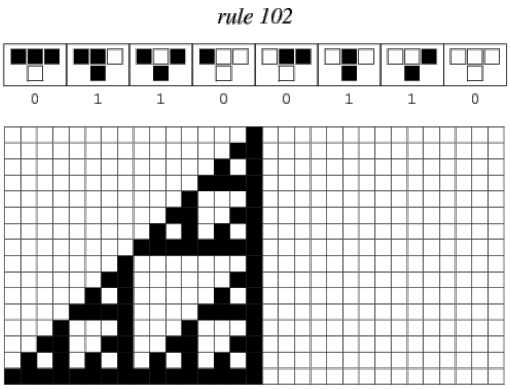
> *"Each ECA rule lookup table is composed of 8 "sub-rules" (although this term is not conventional, we will use it in this description). Here is an example of the 8 sub-rules for rule 110:*

"At each iteration in the environment, the frequency distribution of sub-rules expressed is computed. Take a look at the first few iterations of rule 110 below.

At the first time step, the sub-rule 000->0 is used 6 times, 001->1 is used 1 time, 010->1 is used 1 time, and 100->0 is also used 1 time.
Respectively, the frequency distribution for the first time step is 6/9, 1/9, 1/9, and 1/9."

"In a 1D ECA model, one genotype is selected from the 256 rule space, while the phenotype is the execution of the rule on some initial condition. " (Adams)

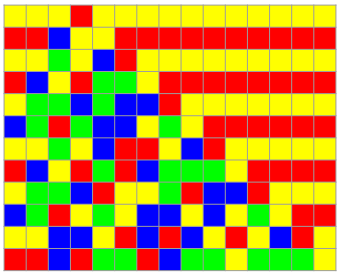
https://www.wolframscience.com/summerschool/2014/alumni/adams.html



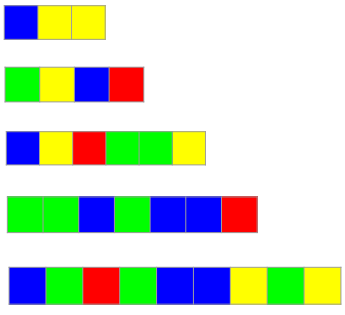
Morpho subrules

Following the above example for ECAs, an application to morphoCAs happens naturally as it has been shown in seminal papers to morphoCA.

```
ArrayPlot[CellularAutomaton[
 ruleDM[{6,3,9,11,15}],
 {{1},0},11],
ColorRules->{1->Red,0->Yellow,2→ Blue, 3→ Green},
ImageSize→Small,Mesh→ True]
```



Frequency distribution of morpho - subsystems



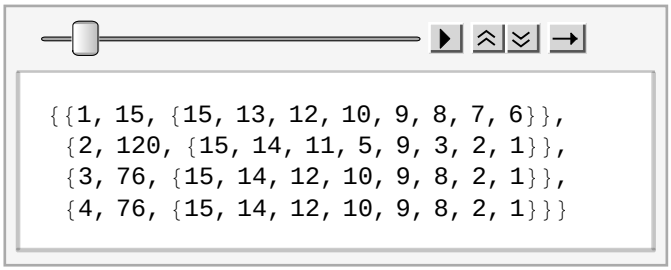
```
CellularAutomaton[
ruleDM[{6,3,9,11,15}],{{1},0},5]
```

```
{{0, 0, 0, 1, 0, 0, 0, 0, 0},   : init,
 {1, 1, 2, 0, 0, 1, 1, 1, 1},   : r6, r6, r11, r3, r9, r6, r6, r6,
 {0, 0, 3, 0, 2, 1, 0, 0, 0},   : r6, r11, r15, r9, r11, r9, r6, r6,
 {1, 2, 0, 1, 3, 3, 0, 1, 1},   : r6, r11, r3, r15, r15, r9, r6, r6,
 {0, 3, 3, 2, 3, 2, 2, 1, 0},   : r11, r15, r15, r15, r9, r11, r15, r9, r6,
 {2, 3, 1, 3, 2, 2, 0, 3, 0}}   : r11, r6, r9, r3, r3, r9, r11, r15, r9,
```

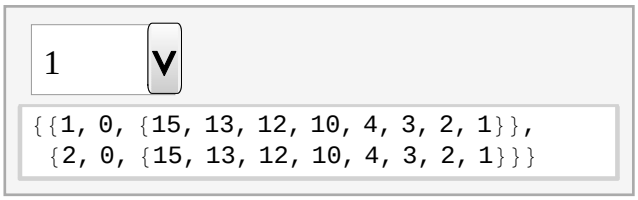
Frequency distribution of subrules for *ruleDM[{6, 3, 9, 11, 15}]*

r6	r3	r9	r11	r15
5	1	1	1	-
3	-	2	2	1
3	1	1	1	2
1	-	1	2	4
1	2	2	2	1

ECA - Sub - Rules with ECA rule numbers



```
{{1, 15, {15, 13, 12, 10, 9, 8, 7, 6}},
 {2, 120, {15, 14, 11, 5, 9, 3, 2, 1}},
 {3, 76, {15, 14, 12, 10, 9, 8, 2, 1}},
 {4, 76, {15, 14, 12, 10, 9, 8, 2, 1}}}
```



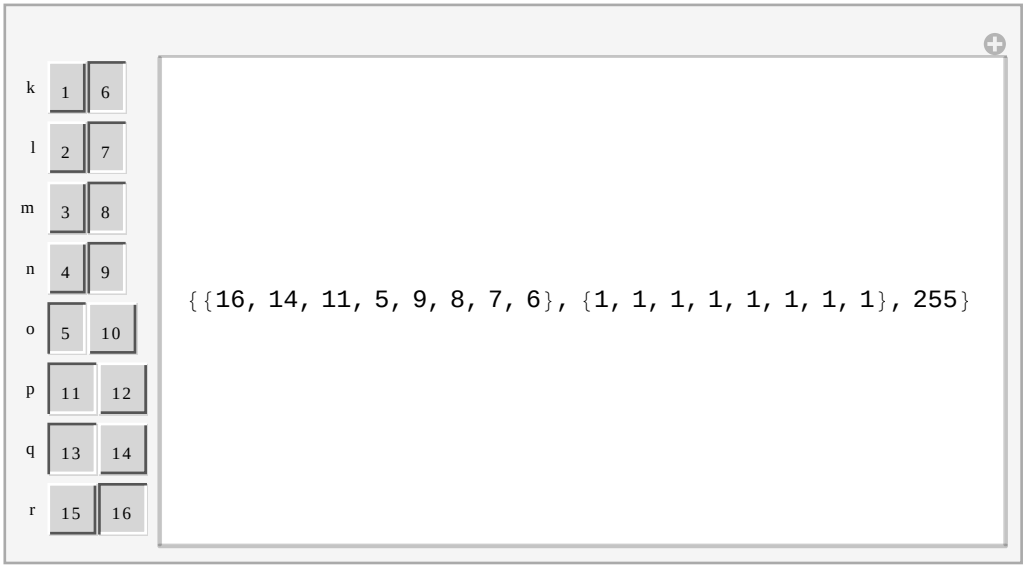

```
{{1, 0, {15, 13, 12, 10, 4, 3, 2, 1}},
 {2, 0, {15, 13, 12, 10, 4, 3, 2, 1}}}
```

Resume : ECA sub - rule, environment, rule number

Mapping of sub-rules to rule number, environment and sub-rule.

It is also desiderable to have the other possible direct mappings too:

Mapping of rule number to environment and sub-rules, and
mapping of environment to rule number and sub-rules to answer direct questions.




```
Table[MapIndexed[{#2[[1]], FromDigits[#, 2],
     RuleTableFromRuleNumber[FromDigits[#, 2]] /. ecarules} &,
   FixedPointList[CellularAutomaton[FromDigits[#, 2], # ] &,
    IntegerDigits[k, 2, 8]]], {k, 0, 255}]
```

Example

```
{{1, 0, {15, 13, 12, 10, 4, 3, 2, 1}}, {2, 0, {15, 13, 12, 10, 4, 3, 2, 1}}},
{{1, 1, {15, 13, 12, 10, 4, 3, 2, 6}}, {2, 124, {15, 14, 11, 5, 9, 8, 2, 1}},
 {3, 70, {15, 14, 12, 10, 4, 8, 7, 1}}, {4, 202, {16, 14, 12, 10, 9, 3, 7, 1}},
 {5, 208, {16, 14, 12, 5, 4, 3, 2, 1}}, {6, 72, {15, 14, 12, 10, 9, 3, 2, 1}},
 {7, 0, {15, 13, 12, 10, 4, 3, 2, 1}}, {8, 0, {15, 13, 12, 10, 4, 3, 2, 1}}},
{{1, 2, {15, 13, 12, 10, 4, 3, 7, 1}}, {2, 4, {15, 13, 12, 10, 4, 8, 2, 1}},
 {3, 4, {15, 13, 12, 10, 4, 8, 2, 1}}},
{1, 3, {15, 13, 12, 10, 4, 3, 7, 6}}, {2, 124, {15, 14, 11, 5, 9, 8, 2, 1}}
```

Non - commutatvity of the ECA sub - rules

```
FromDigits[kAryFromRuleTable[
  ruleECA[{6, 2, 3, 4, 5, 9, 7, 16}]], 2]
```

```
143
```

```
FromDigits[kAryFromRuleTable[
  ruleECA[{2, 6, 3, 4, 5, 9, 7, 16}]], 2]
```

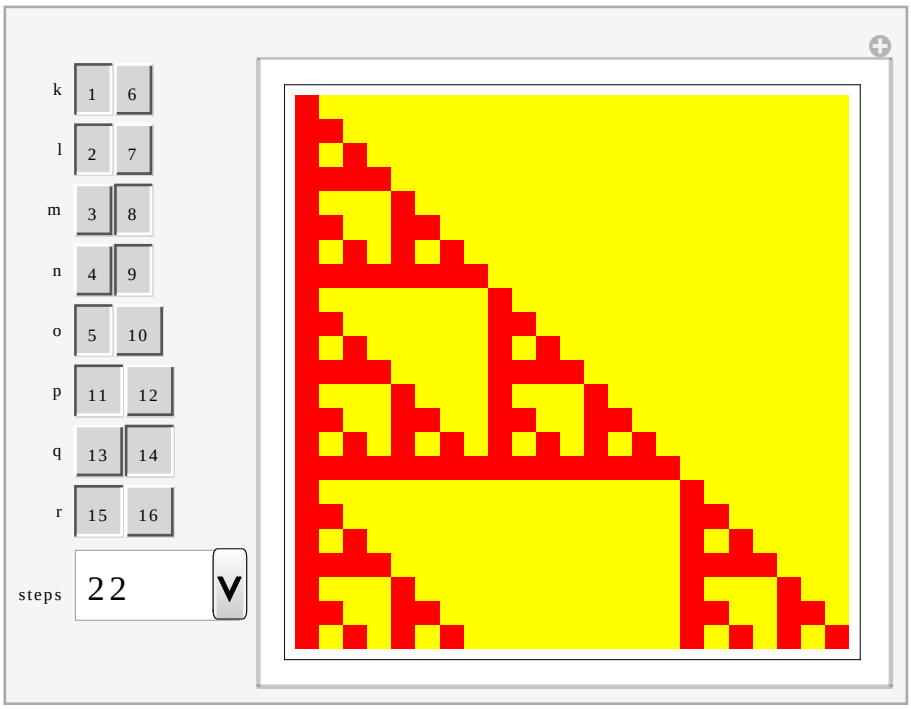
```
79
```

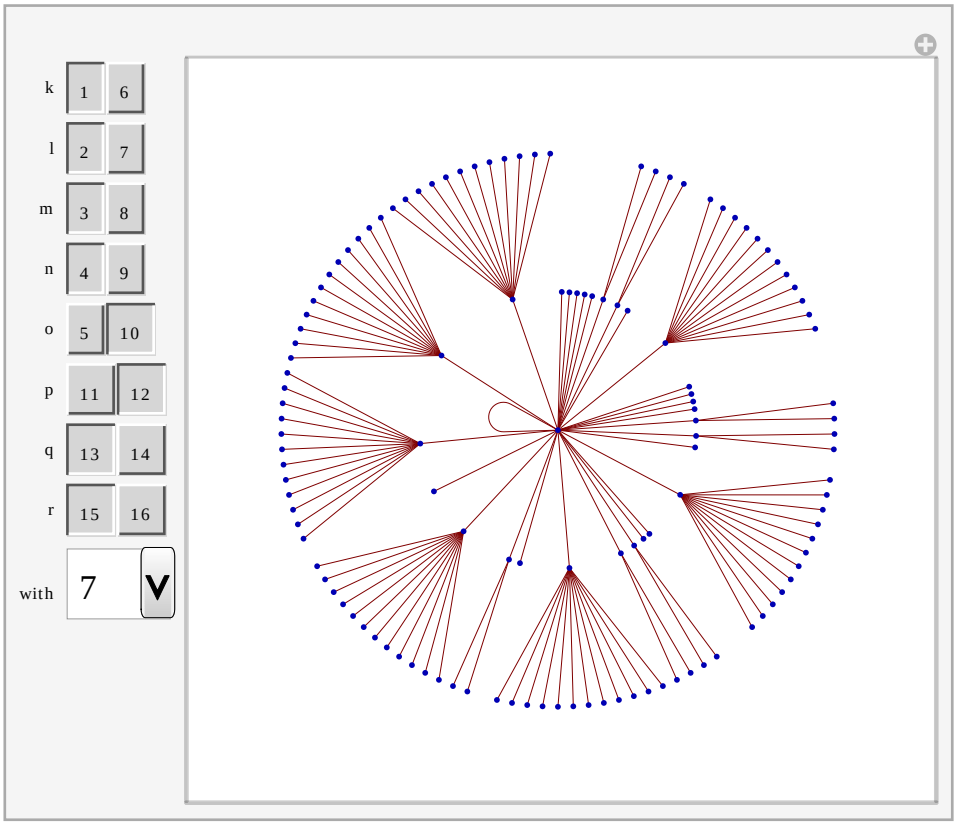
```
FromDigits[kAryFromRuleTable[
   ruleECA[{6, 2, 3, 4, 5, 9, 7, 16}]], 2] ==
 FromDigits[kAryFromRuleTable[
   ruleECA[{2, 6, 3, 4, 5, 9, 7, 16}]], 2]
```

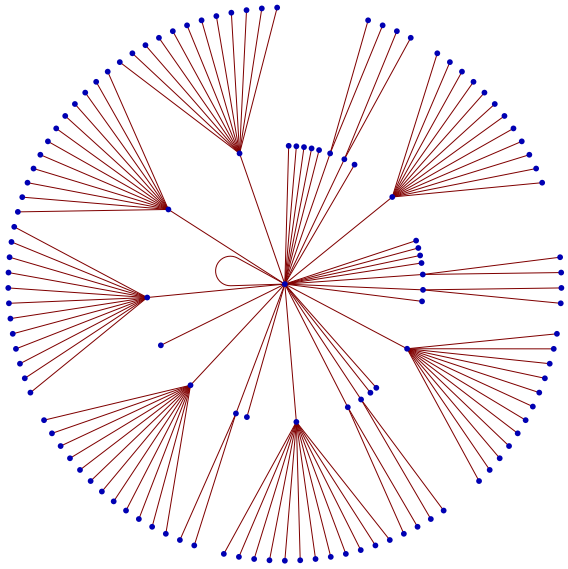
```
False
```

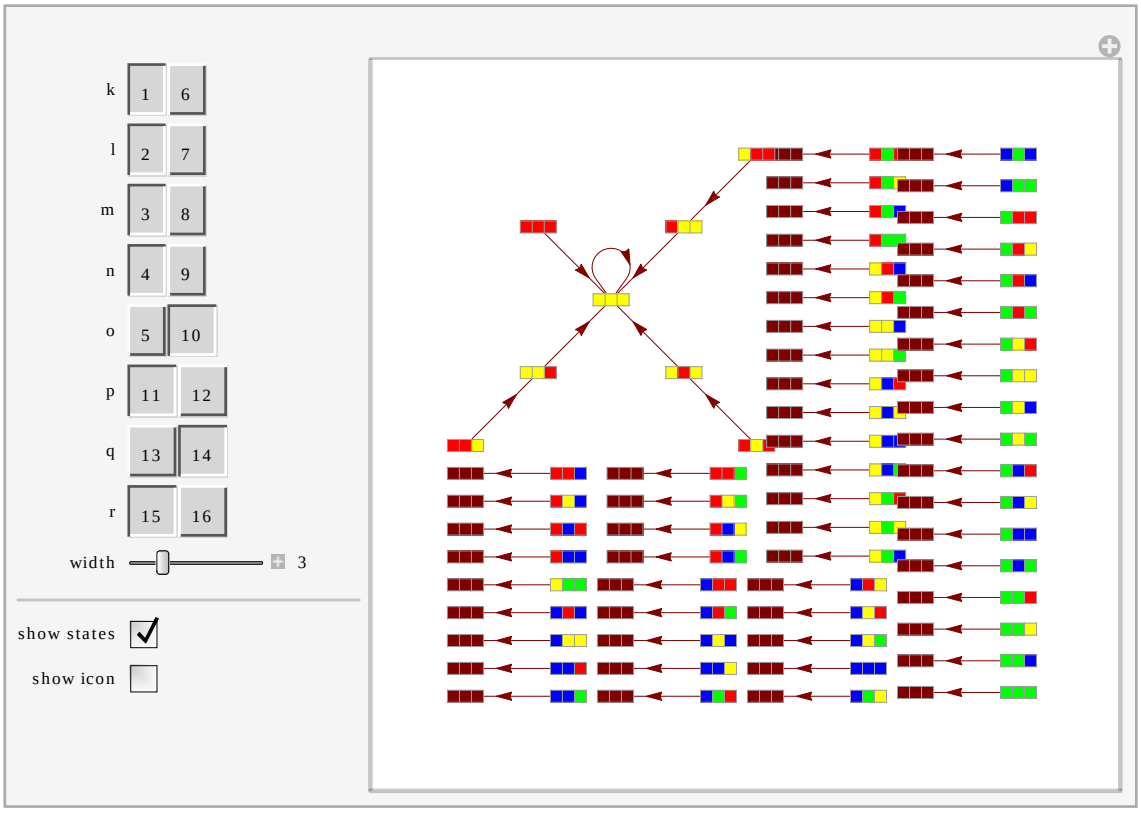
ECA sub-rule manipulators

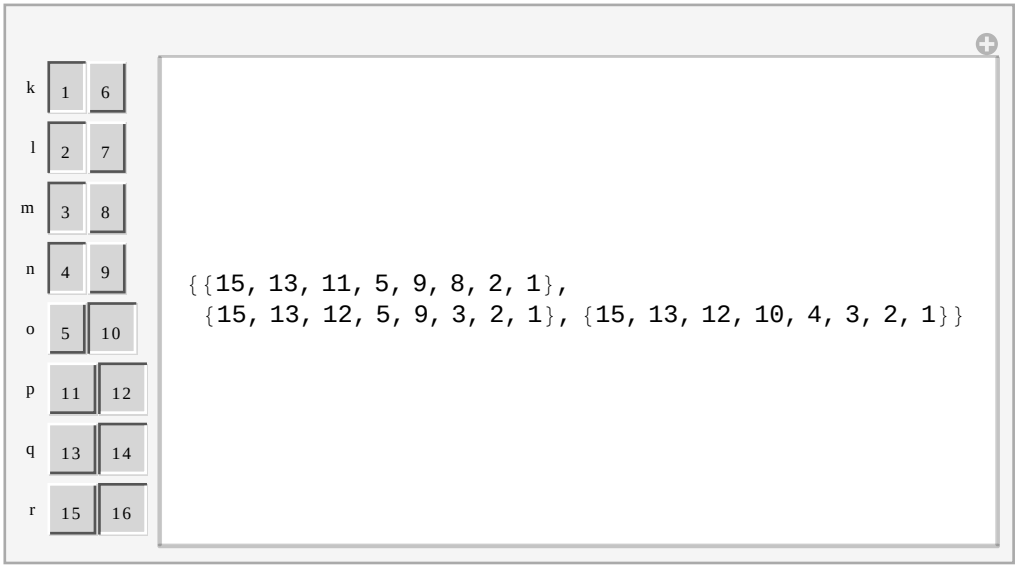
The method of sub-rules for ECAs is an *abstraction* and *parametrization* of the components of the rule schemes that allows a micro-anlysis of the ECAs. The ECA sub-rule manipulator manages all ECA rules of a 1D ECA. The sub-rule manipulators enables a micro-analysis of the behavior of all 2^8 ECA rules.

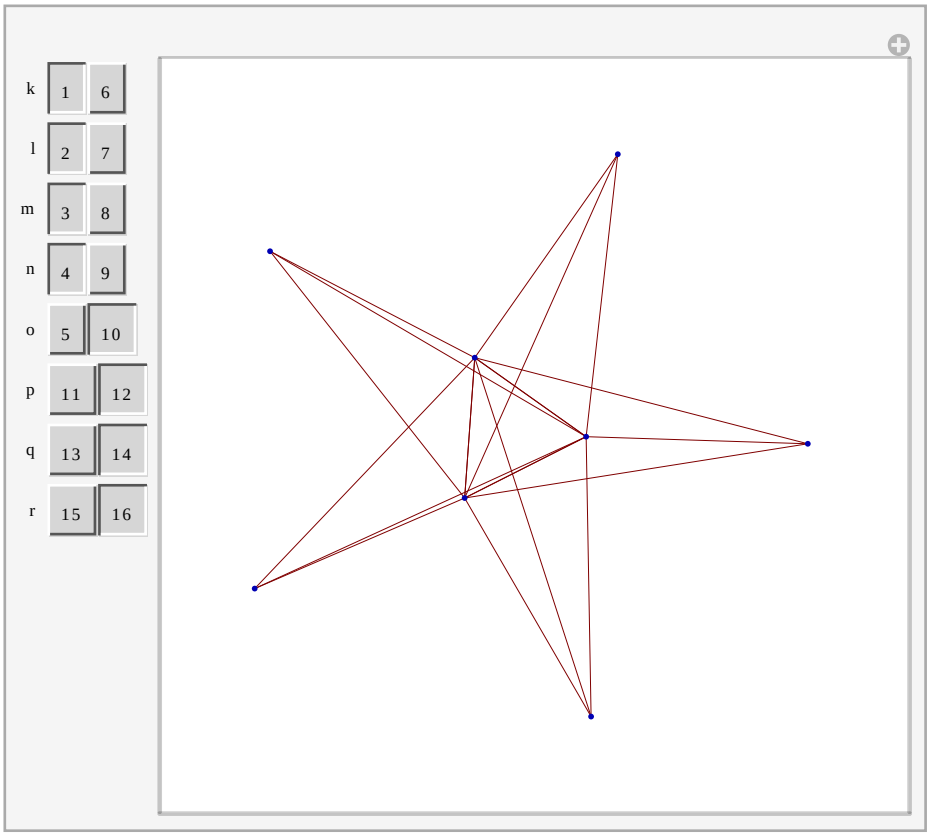

k 1 6
l 2 7
m 3 8
n 4 9
o 5 10
p 11 12
q 13 14
r 15 16
steps 22


k 1 6
l 2 7
m 3 8
n 4 9
o 5 10
p 11 12
q 13 14
r 15 16
with 7




k
1
6
l
2
7
m
3
8
n
4
9
o
5
10
p
11
12
q
13
14
r
15
16
width
3
show states
show icon



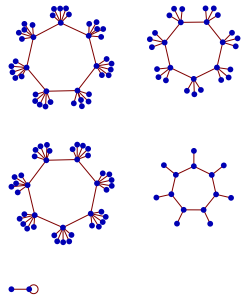


Examples for ECA

ruleECA[{15, 14, 11, 5, 4, 3, 2, 1}]

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

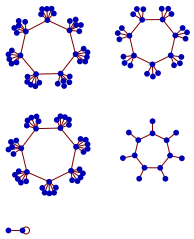
```
GraphPlot[# -> CellularAutomaton[ruleECA[{15, 14, 11, 5, 4, 3, 2, 1}], #] & /@
  Tuples[{0, 1}, 7]]
```



```
FromDigits[kAryFromRuleTable[
  ruleECA[{15, 14, 11, 5, 4, 3, 2, 1}]], 2]
```

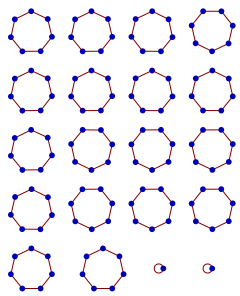
48

```
GraphPlot[# -> CellularAutomaton[48, #] & /@ Tuples[{0, 1}, 7]]
```

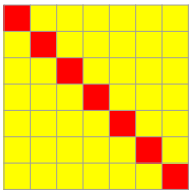


Comparison

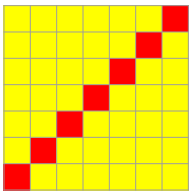
```
GraphPlot[# -> CellularAutomaton[ruleDCl[{1, 7, 3, 9}], #] & /@ Tuples[{0, 1}, 7]]
```



```
ArrayPlot[CellularAutomaton[48, {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
ImageSize → Tiny, Mesh → True]
```



```
ArrayPlot[CellularAutomaton[ruleDCl[{1, 7, 3, 9}], {{1}, 0}, 6],
 ColorRules -> {1 -> Red, 0 -> Yellow, 2 → Blue, 3 → Green},
ImageSize → Tiny, Mesh → True]
```



Structural self-referentiality of sub-rule morphoCA

The approach to manipulate the sub-rules of a ECA or more interestingly of the morphoRules offers a new method for an implementation of the self-modification of the automata rules.

The method is an abstraction and parametrization of the components of the rule schemes.

First, a external manipulation of the sub-rules is sketched.

Hence the rule scheme with its pre-defined range for its parameters

```
ruleDM[{a_,b_,c_,d_,e_}] :=
       Flatten[{dca[{a}],dca[{b}],dca[{c}],dca[{d}],dca[{e}]}]
```

which is at the base for all morphic rules of type DM gets parametrized by:

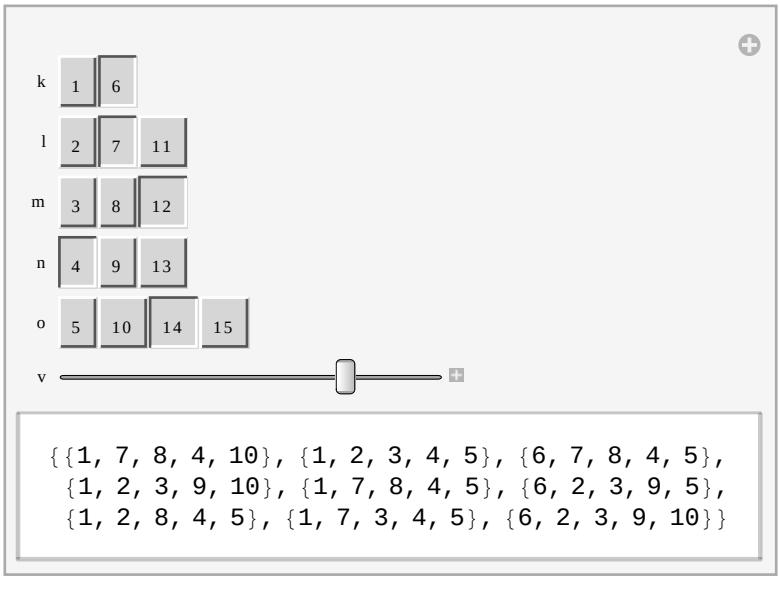
```
Manipulate[
ruleDM[{k,l,m,n,o}],
{k,{1,6}},{l,{2,7,11}},{m,{3,8,12}},{n,{4,9,13}},{o,{5,10,14,15}}]
```

which is delivering all rules of the rule scheme for ruleDM .

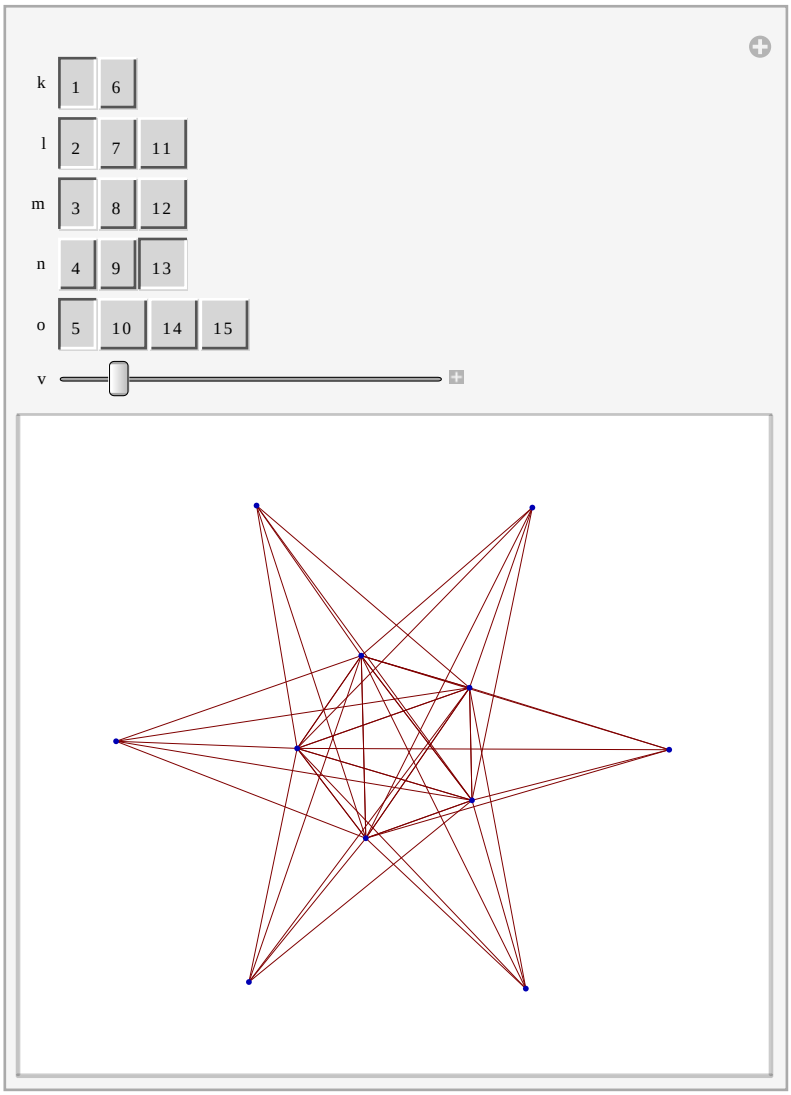
All kind of parametrizations are augmenting the *abstractness* and *flexibility* of an automaton.

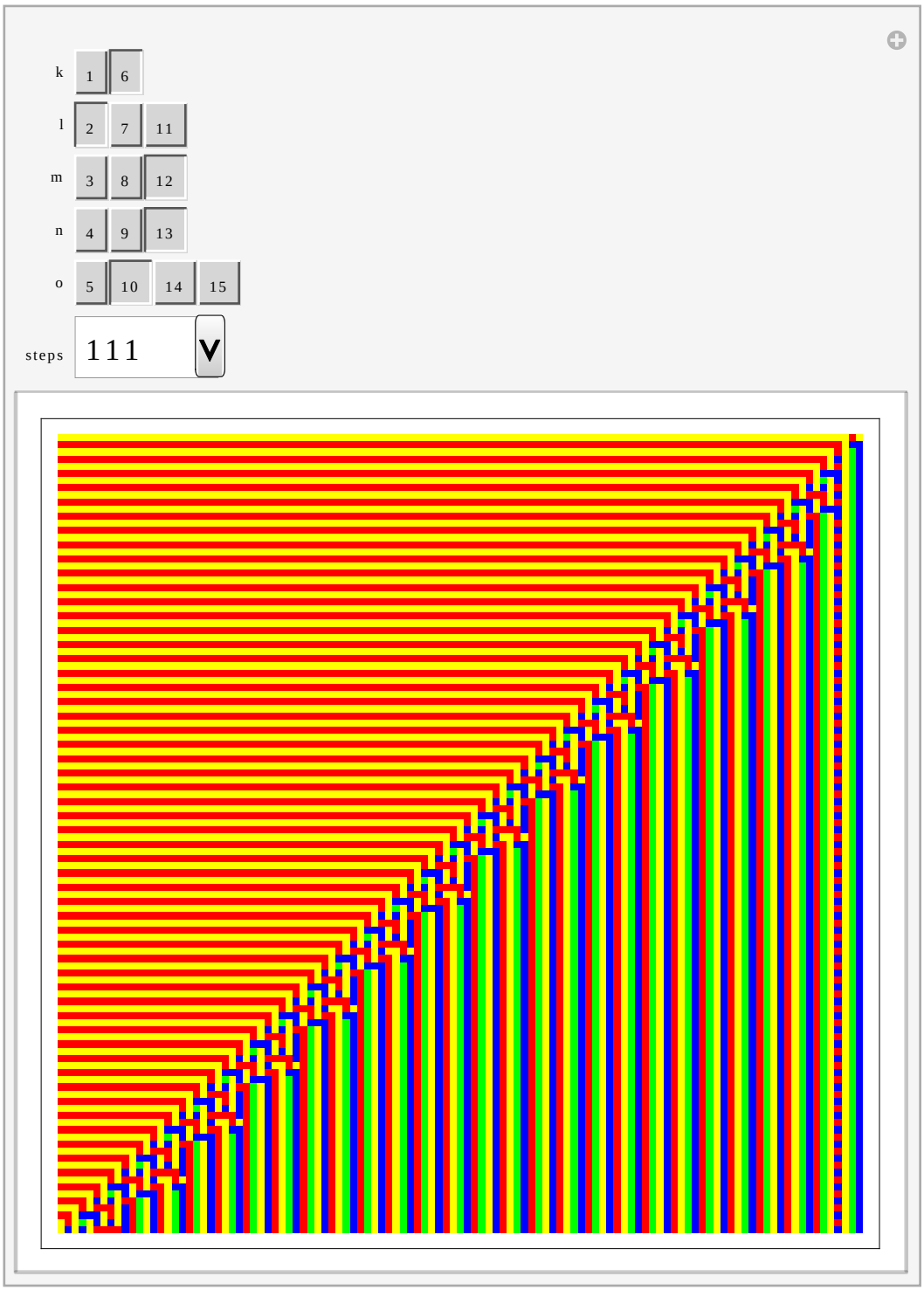
A further abstraction is achieved over the *complexity* of the rule scheme and not just of the sub-rules.
E.g. from *ruleDM[{k,l,m,n,o}] to ruleDMN[{k,l,m,n,o,p}]* or a reduction to *ruleCl[{k,l,m,n}]*.

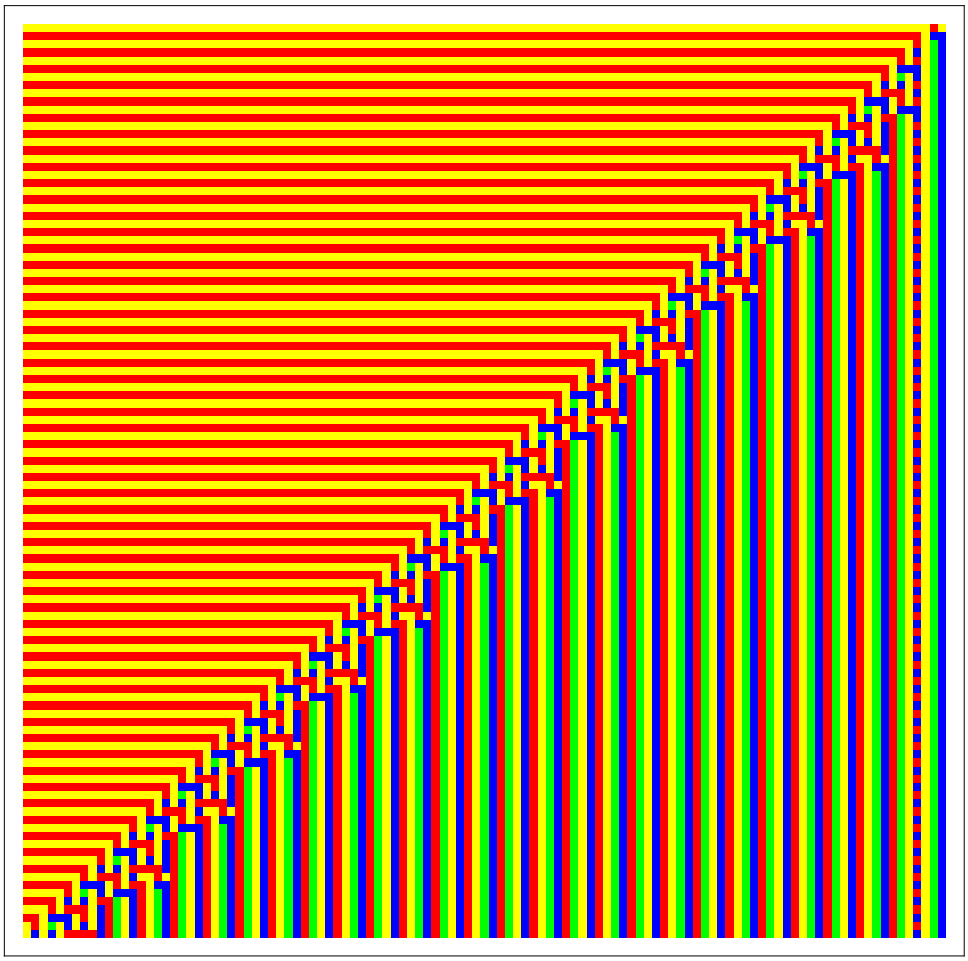
Thus, a parametrization of the head of the parametrized rule schemes has to be implemented as a next step of abstracting from the 'information' based paradigm of automata.

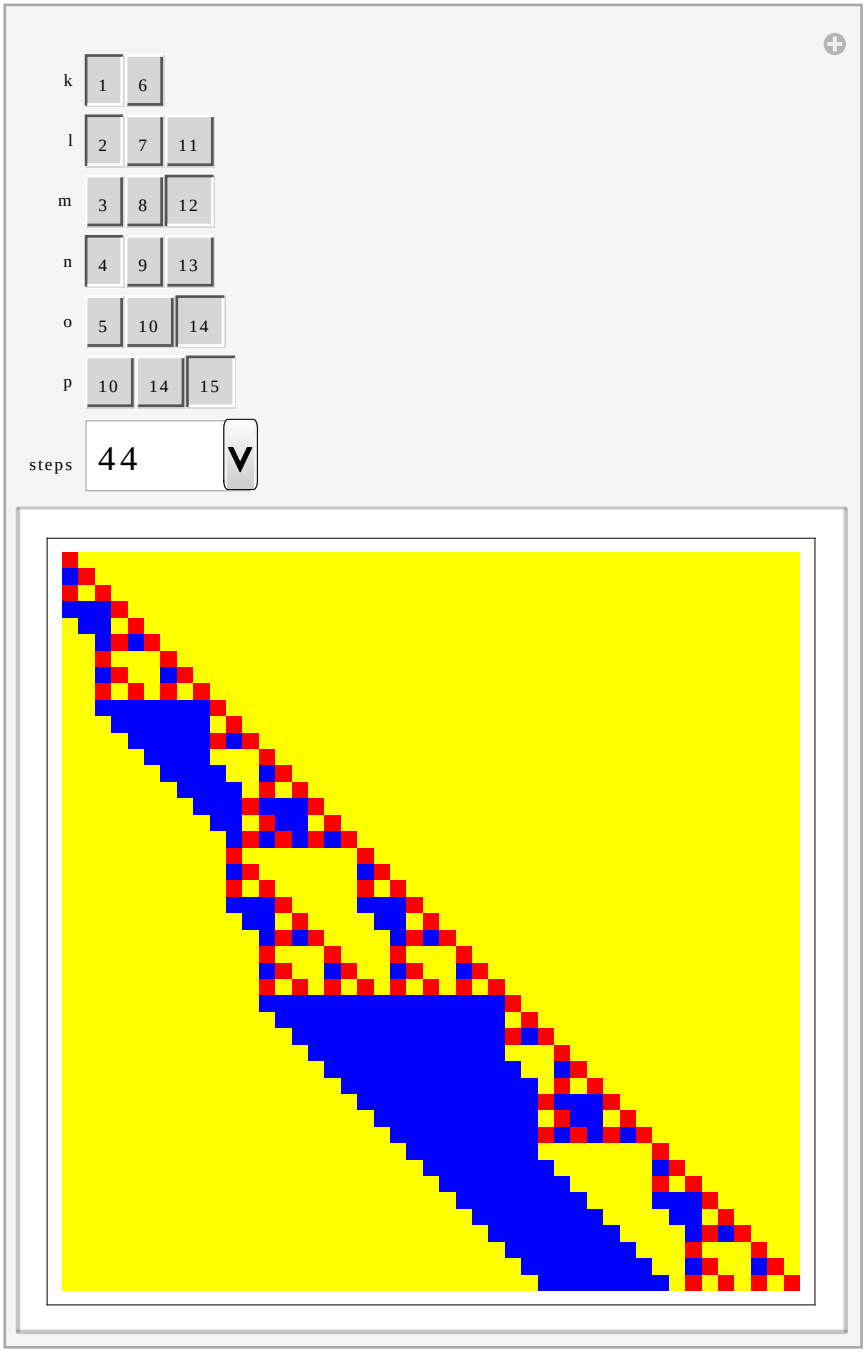


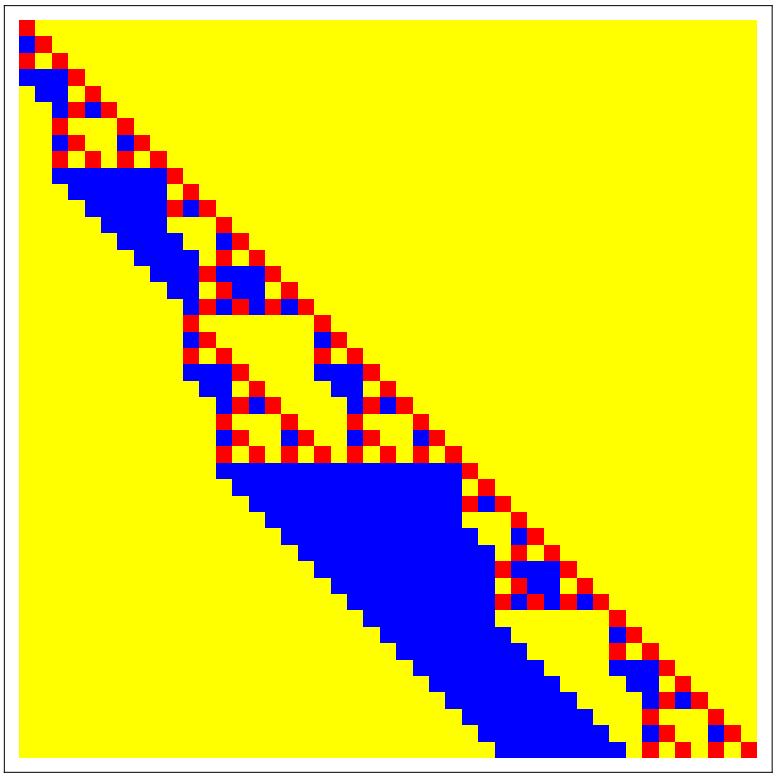

```
{{1, 7, 8, 4, 10}, {1, 2, 3, 4, 5}, {6, 7, 8, 4, 5},
 {1, 2, 3, 9, 10}, {1, 7, 8, 4, 5}, {6, 2, 3, 9, 5},
 {1, 2, 8, 4, 5}, {1, 7, 3, 4, 5}, {6, 2, 3, 9, 10}}
```


k
1
6
l
2
7
11
m
3
8
12
n
4
9
13
o
5
10
14
15
v


k
1
6
l
2
7
11
m
3
8
12
n
4
9
13
o
5
10
14
15
steps
111




k
1
6
l
2
7
11
m
3
8
12
n
4
9
13
o
5
10
14
p
10
14
15
steps
44



k 1 6

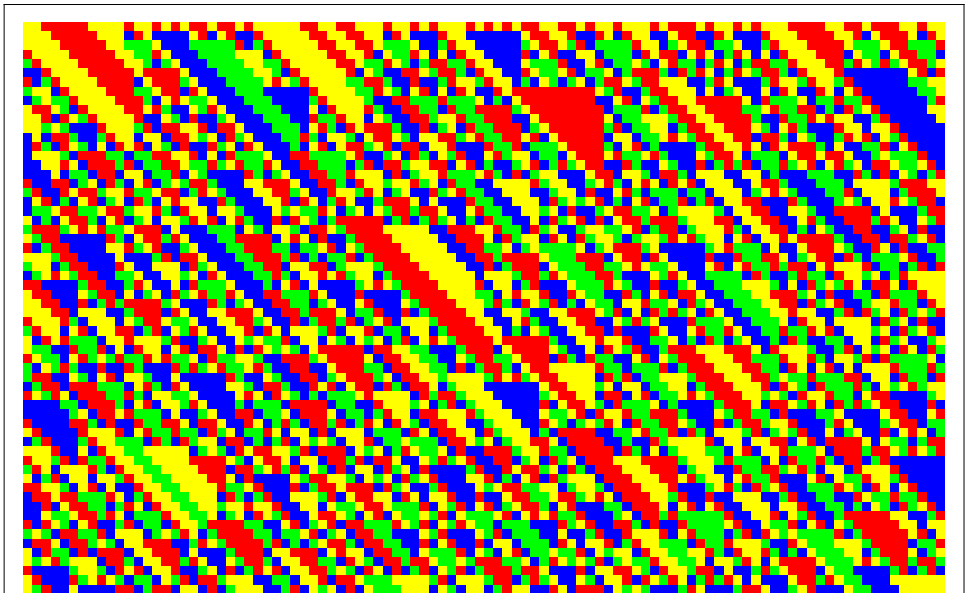
l 2 7 11

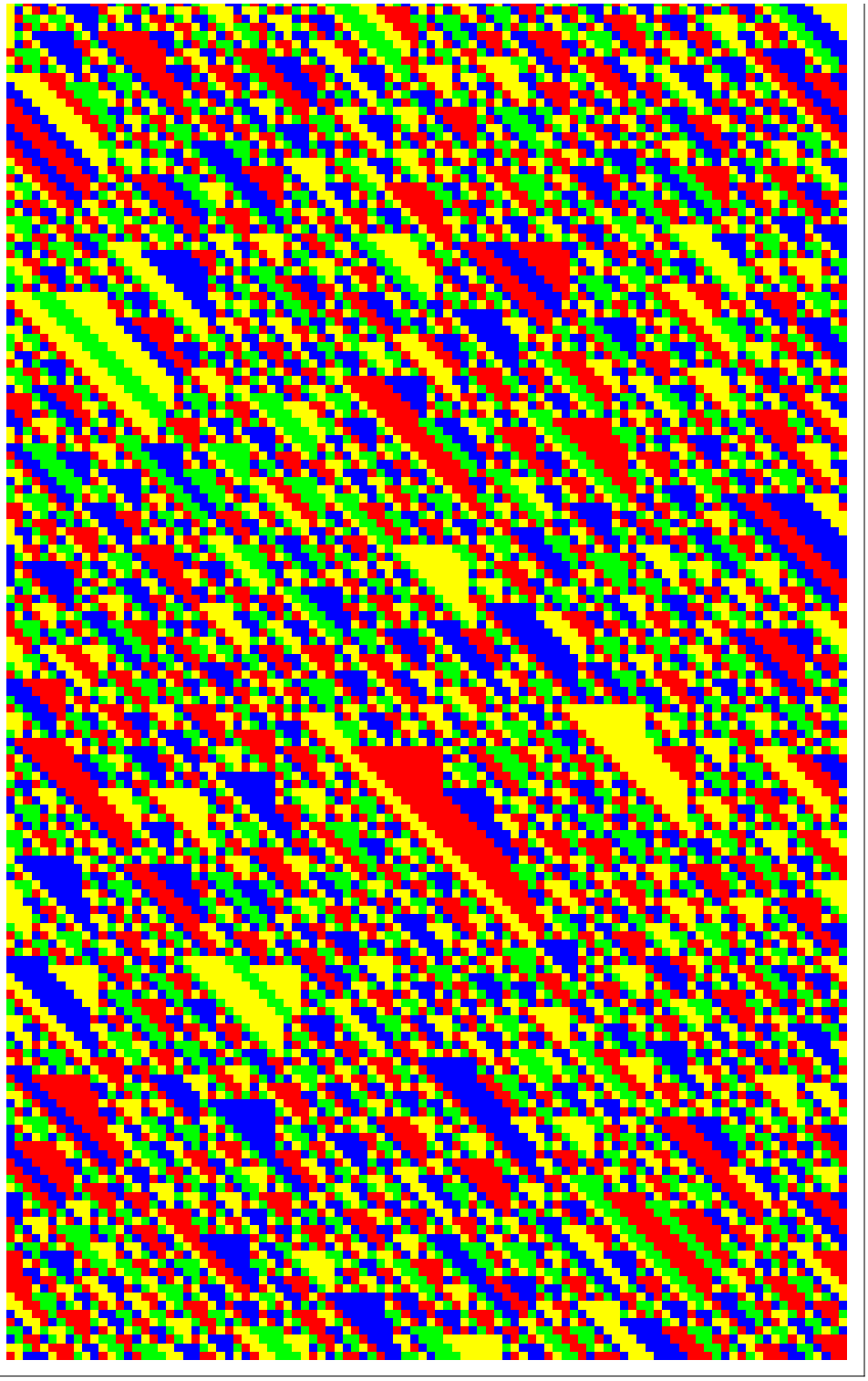
m 3 8 12

n 4 9 13

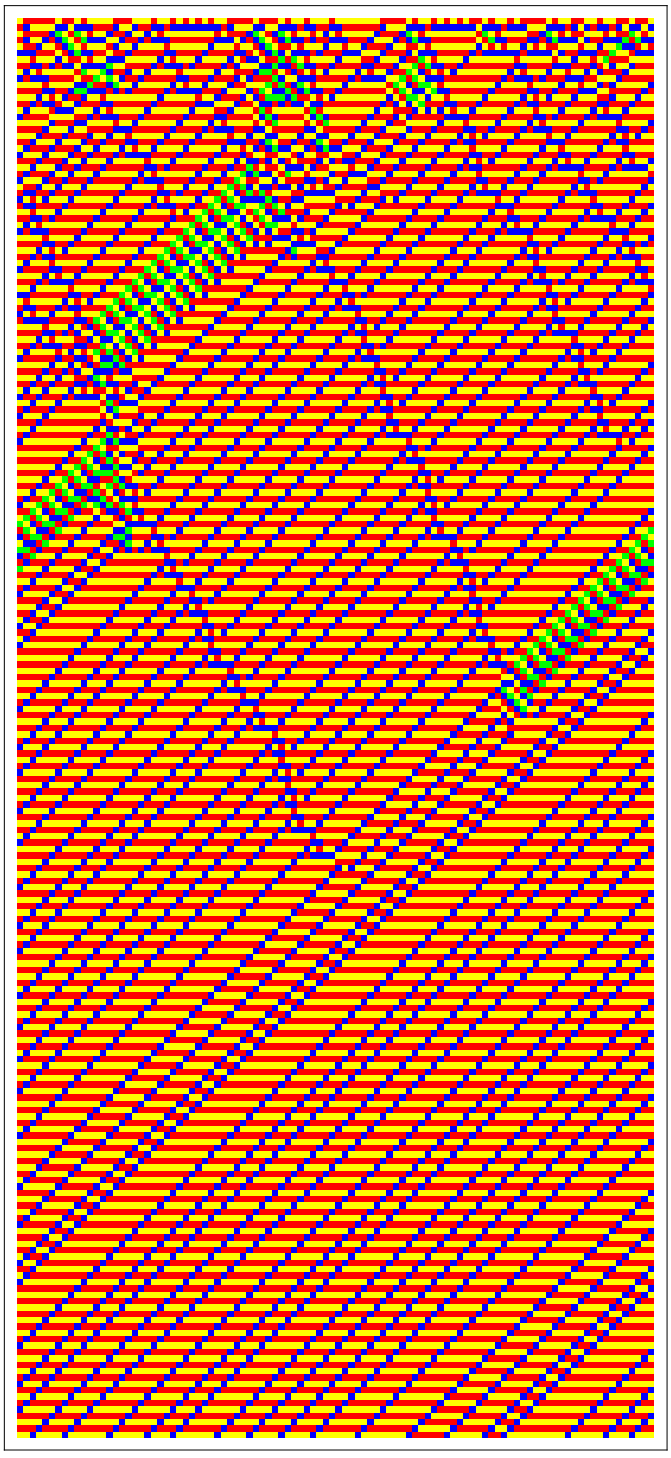
o 5 10 14 15

steps 222

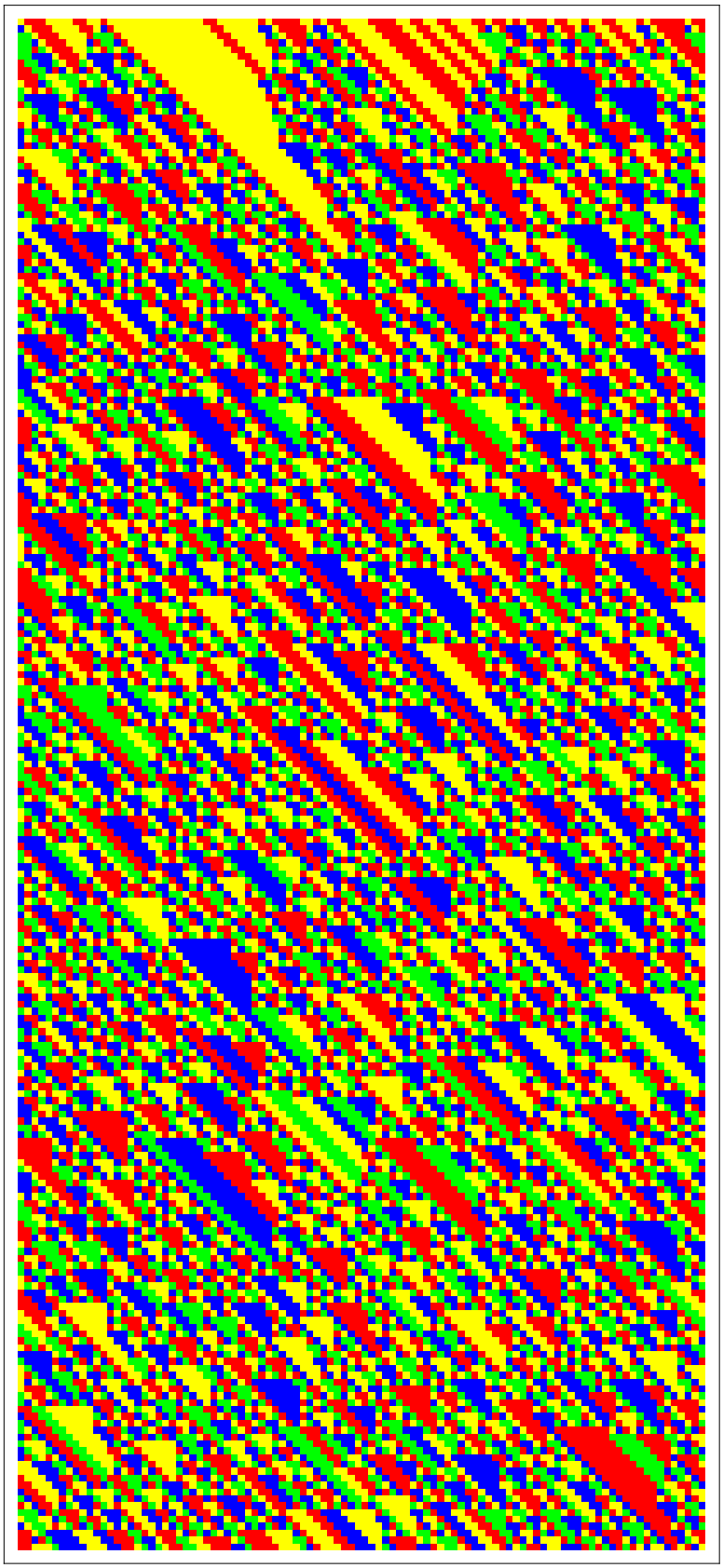


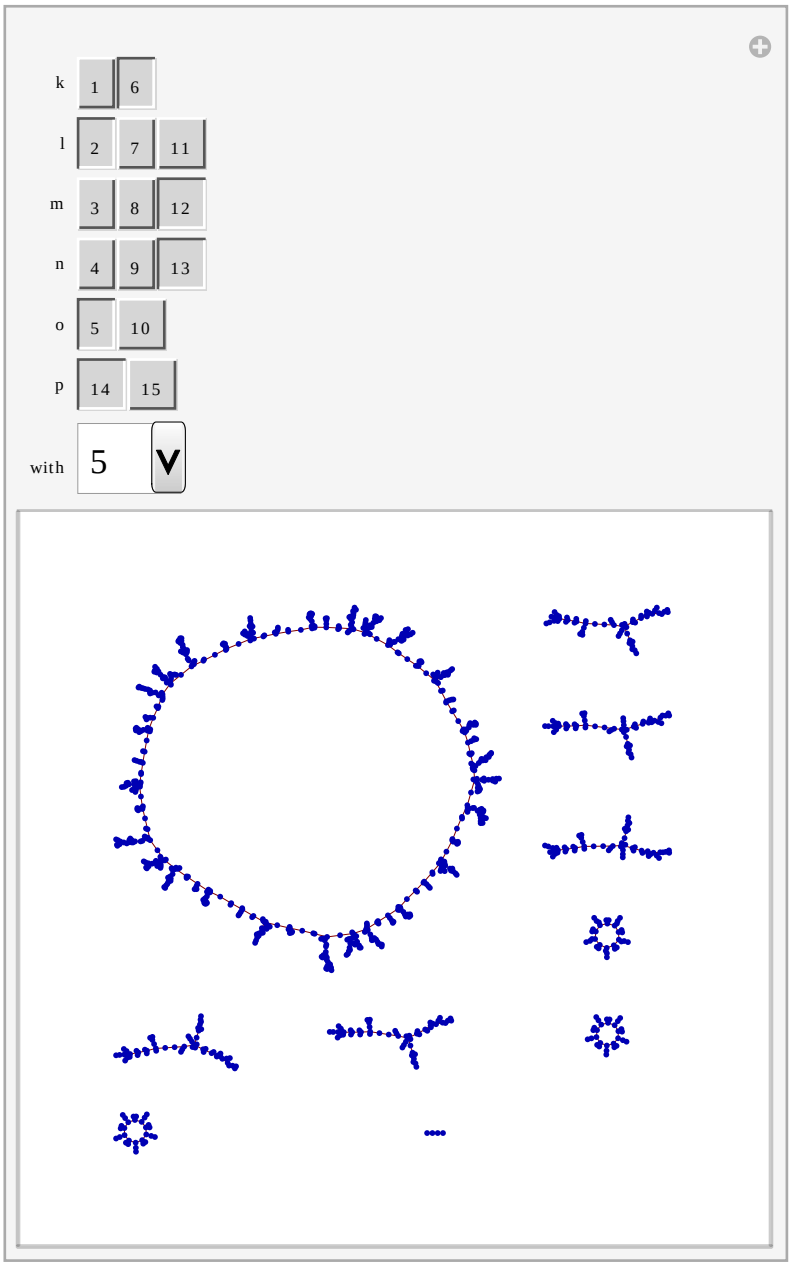


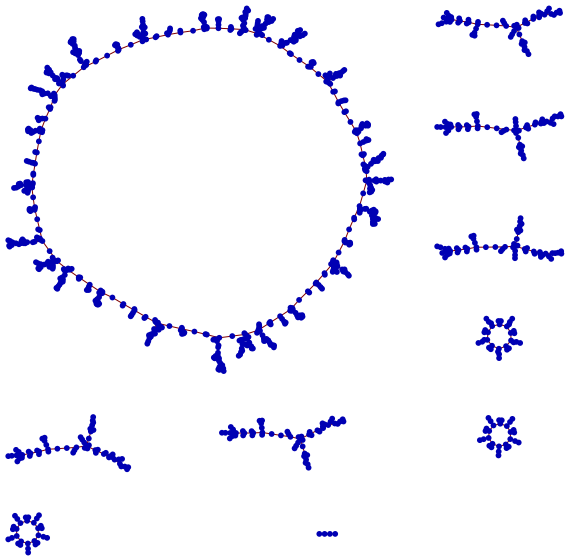
ruleDM[{6, 7, 12, 13, 5}]



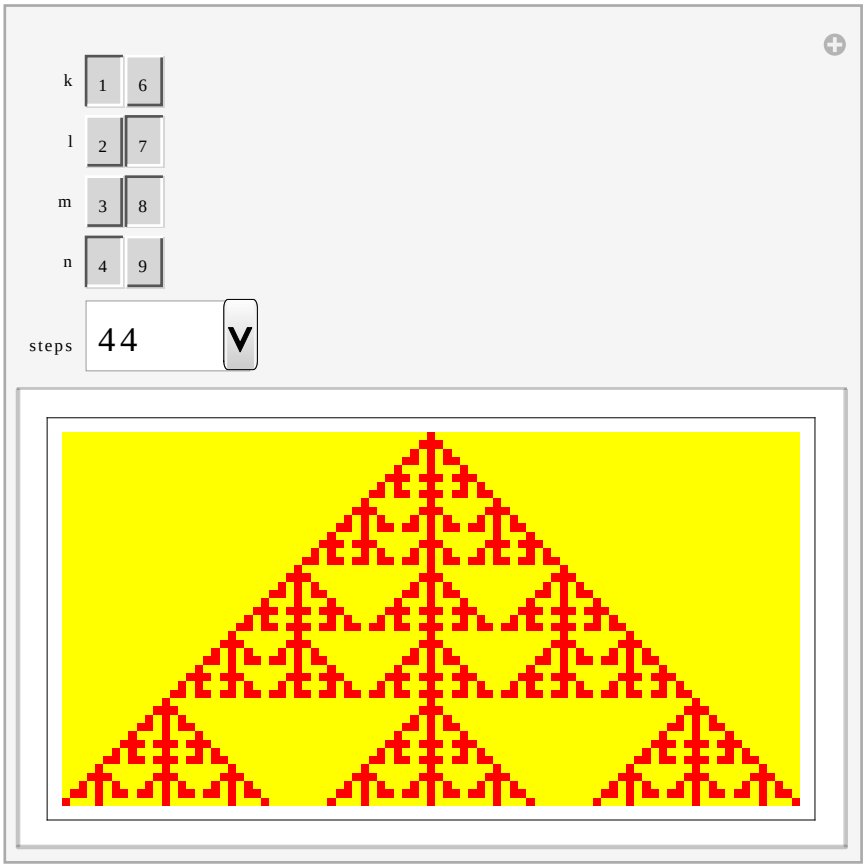
ruleDM[{1, 2, 12, 4, 15}]

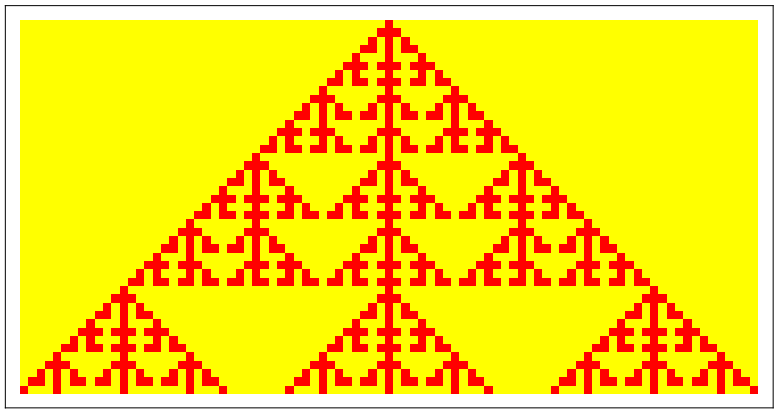



k
1
6
l
2
7
11
m
3
8
12
n
4
9
13
o
5
10
p
14
15
with
5



DCI sub - rule manipulator





ruleDCl DCl9

{{6, 2, 8, 4}, {1, 7, 3, 9}}

Graph of self-modifying morphoCAs of ruleDCl

About the *'stream of consciousness'* (James, Husserl) and a new method to study it.

Towards a deep-structural phenomenology of consciousness as self-modifying morphogrammatically organized streams of changing automata.

The modifications of the rules of the automata happens on the background of their environments (initial conditions).

The temporal structurations of morphogrammatic operations is 'memristive' (retro-grade recursive).

Different environments are supporting different self-modifications of rules.

Different rules are enacting different environments.

It is reasonable to study single self-modifying constellations, small fields of modification or the behavior of the whole system.

Very first results are achieved on the level of simple indicational and morphic CAs.

With the indCA approach it turns out that the self-modifying field of indication is split into two (???) sub-structures, while the morphoCA based analysis has a coherent field of self-modifications interpreted as deep-structural actions of cognitive behavior.

Indicational example indCl with environment = {1, 0, 0, 0}

k 1 5

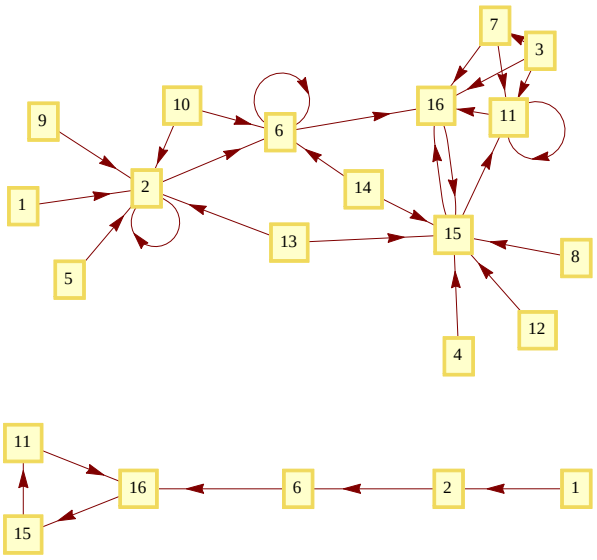
l 2 6

m 3 7

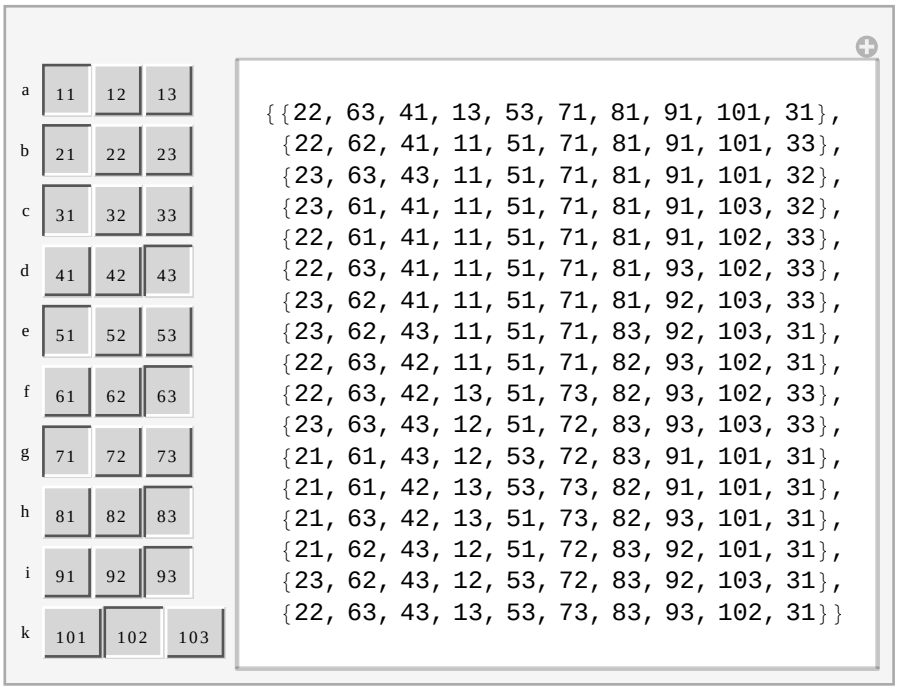
n 4 8

{{4, 7, 2, 5}, {8, 7, 6, 5}}

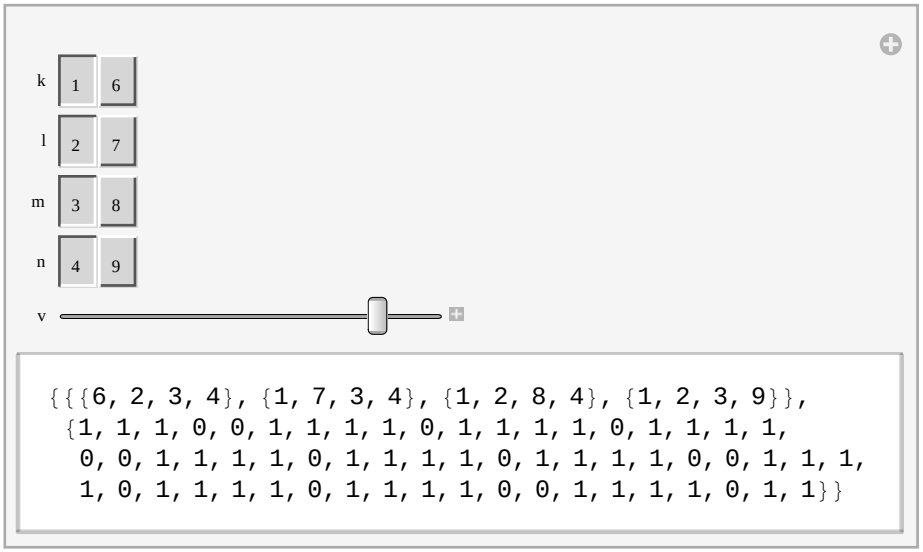
```
[1]:: {1, 2, 3, 4} = {8, 3, 2, 1} = 2,
[2]:: {1, 2, 3, 8} = {8, 3, 6, 1} = 6, {8, 3, 2, 1} = 2,
[3]:: {1, 2, 7, 4} = 7, {4, 7, 2, 5} = 11, {8, 7, 6, 5} = 16,
[4]:: {1, 2, 7, 8} = {4, 7, 6, 5} = 15,

[5]:: {1, 6, 3, 4} = {8, 3, 2, 1} = 2,
[6]:: {1, 6, 3, 8} = {8, 3, 6, 1} = 6, {8, 7, 6, 5} = 16,
[7]:: {1, 6, 7, 4} = {4, 7, 2, 5} = 11, {8, 7, 6, 5} = 16,
[8]:: {1, 6, 7, 8} = {4, 7, 6, 5} = 15,

[9]:: {5, 2, 3, 4} = {8, 3, 2, 1} = 2,
[10]:: {5, 2, 3, 8} = {8, 3, 6, 1} = 6, {8, 3, 2, 1} = 2,
[11]:: {5, 2, 7, 4} = {4, 7, 2, 5} = 11, {8, 7, 6, 5} = 16,
[12]:: {5, 2, 7, 8} = {4, 7, 6, 5} = 15,

[13]:: {5, 6, 3, 4} = {8, 3, 2, 1} = 2, {4, 7, 6, 5} = 15,
[14]:: {5, 6, 3, 8} = {8, 3, 6, 1} = 6, {4, 7, 6, 5} = 15,
[15]:: {5, 6, 7, 4} = {4, 7, 2, 5} = 11, {8, 7, 6, 5} = 16,
[16]:: {5, 6, 7, 8} = {4, 7, 6, 5} = 15
```



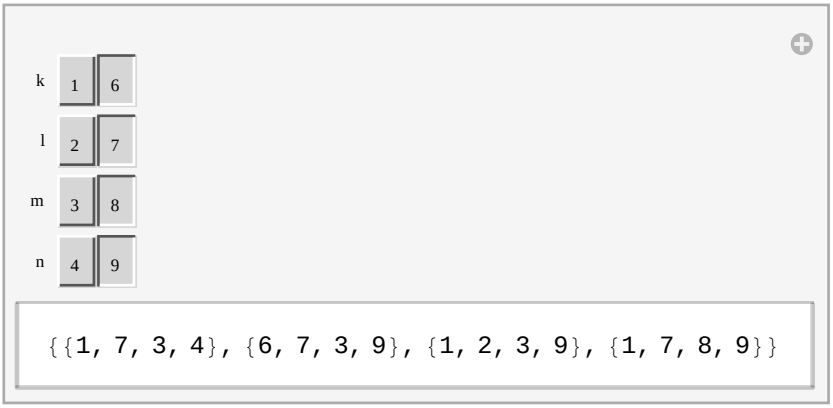
Indicational example indCIR with environment = {1,0,0,0,1,1,1,2,2,2}



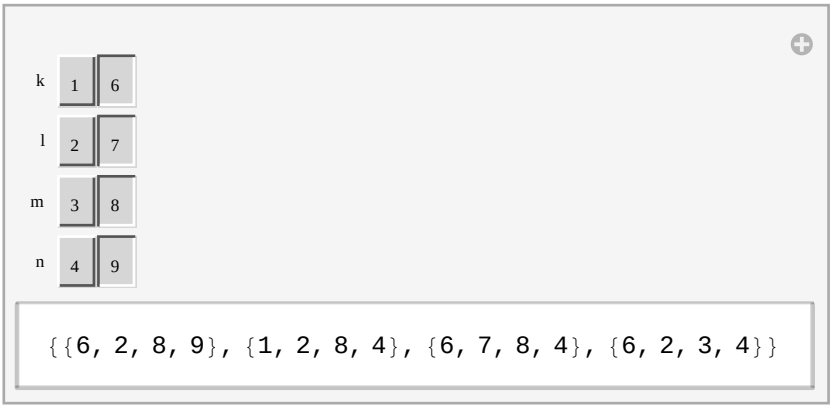
DCl-Environment = {{v}}



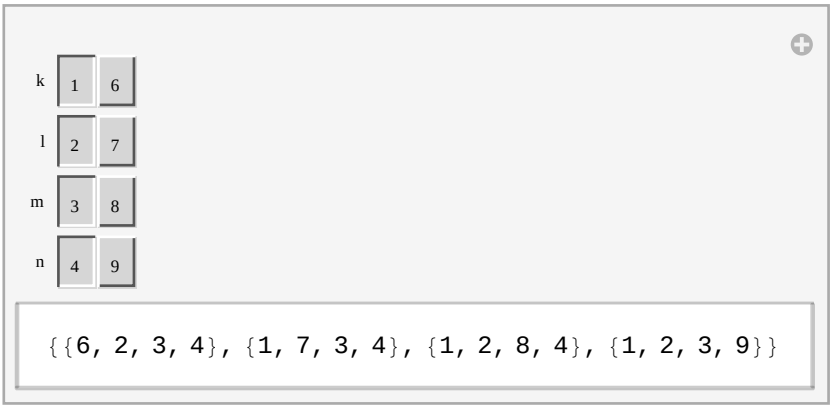
Environment = {0, 1, 1, 1}



Environment = {0, 0, 0, 1}



Environment = {0, 0, 0, 1}

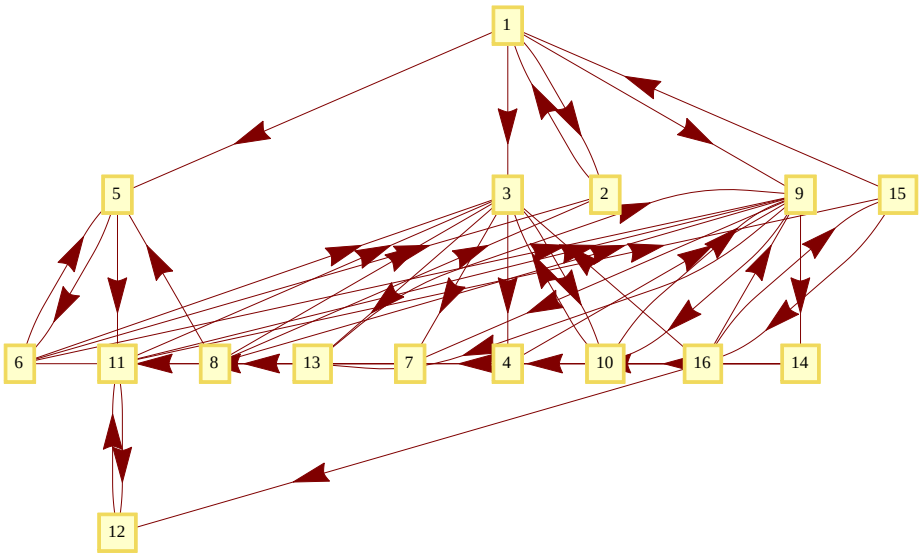


Environment = {1, 0, 0, 0}

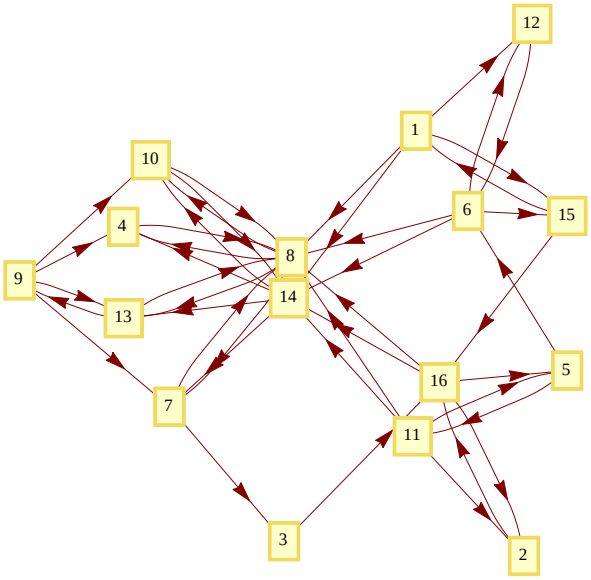
k	1	6
l	2	7
m	3	8
n	4	9

```
{{1, 2, 3, 9}, {1, 2, 8, 4}, {1, 7, 3, 4}, {6, 2, 3, 4}}
```

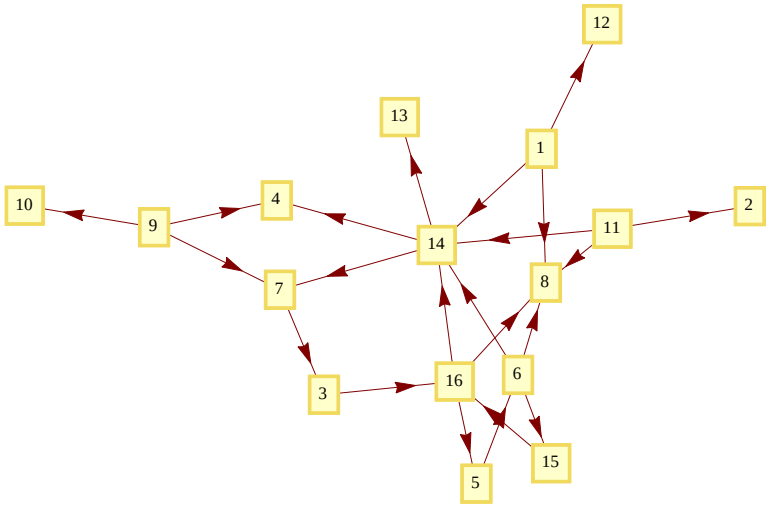
```
[1] = [1, 2, 3, 4],
[1]:: {{1, 7, 3, 4} = 5, {1, 2, 8, 4} = 3, {1, 2, 3, 9} = 2, {6, 2, 3, 4} = 9}
[2]:: {{1, 2, 3, 4} = 1}
[3]:: {{6, 7, 3, 4} = 13, {1, 7, 8, 4} = 7, {1, 2, 8, 9} = 4, {6, 2, 3, 9} = 10}
[4]:: {{6, 2, 3, 4} = 9}
[5]:: {{1, 7, 3, 9} = 6, {6, 2, 8, 4} = 11}
[6]:: {{1, 2, 3, 9} = 2, {1, 2, 8, 4} = 3, {1, 7, 3, 4} = 5, {6, 2, 3, 4} = 9}
[7]:: {{1, 7, 3, 9} = 6, {6, 2, 8, 4} = 11}
[8]:: {{1, 2, 3, 9} = 2, {1, 2, 8, 4} = 3, {1, 7, 3, 4} = 5, {6, 2, 3, 4} = 9}

[9]:: {{1, 7, 8, 4} = 7, {1, 2, 8, 9} = 14, {6, 2, 3, 9} = 10, {6, 7, 3, 4} = 13}
[10]:: {{1, 2, 8, 4} = 3, {6, 2, 3, 4} = 9}
[11]:: {{6, 7, 8, 4} = 15, {1, 2, 8, 4} = 3, {6, 2, 8, 9} = 12, {6, 2, 3, 4} = 9}
[12]:: {{6, 2, 8, 4} = 11}
[13]:: {{1, 7, 8, 9} = 8, {6, 2, 3, 4} = 9}
[14]:: {{1, 2, 8, 9} = 4, {1, 7, 8, 4} = 7, {6, 7, 3, 4} = 13, {6, 2, 3, 9} = 10}
[15]:: {{6, 7, 8, 9} = 16, {1, 2, 3, 4} = 1}
[16]:: {{6, 2, 8, 9} = 12, {1, 2, 8, 4} = 3, {6, 7, 8, 4} = 15, {6, 2, 3, 4} = 9}
```

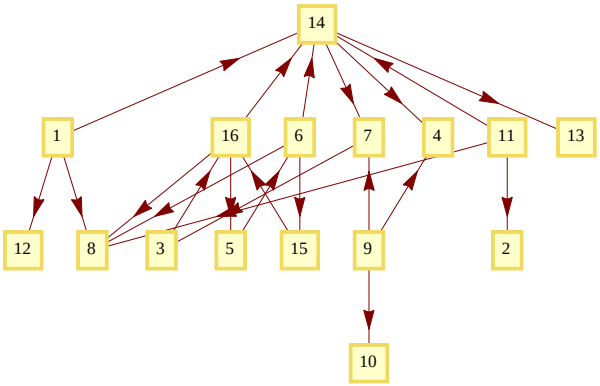


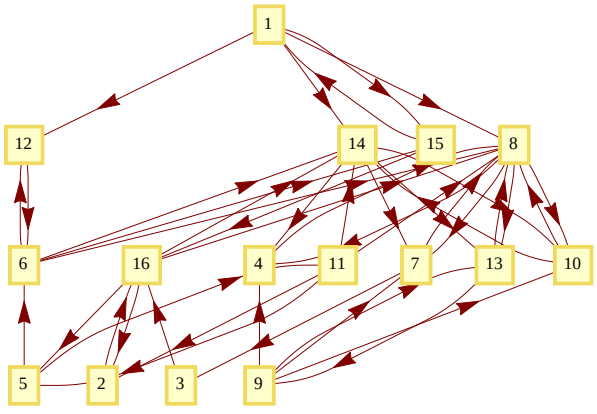
```
[1] :: {{6, 2, 8, 9} = 12, {6, 7, 3, 9} = 14, {6, 7, 8, 4} = 15, {1, 7, 8, 9} = 8}
[2] :: {{6, 7, 8, 9} = 16}
[3] :: {{6, 7, 8, 9} = 16}
[4] :: {{1, 7, 8, 9} = 8}
[5] :: {{6, 2, 8, 4} = 11, {1, 7, 3, 9} = 6}
[6] :: {{6, 7, 8, 4} = 15, {6, 7, 3, 9} = 14, {6, 2, 8, 9} = 12, {1, 7, 8, 9} = 8}
[7] :: {{1, 2, 8, 4} = 3, {1, 7, 8, 9} = 8}
[8] :: {{1, 7, 8, 4} = 7, {6, 7, 3, 4} = 13, {6, 2, 3, 9} = 10, {1, 2, 8, 9} = 4}

[9] :: {{6, 2, 3, 9} = 10, {6, 7, 3, 4} = 13, {1, 7, 8, 4} = 7, {1, 2, 8, 9} = 4}
[10] :: {{6, 7, 3, 9} = 14, {1, 7, 8, 9} = 8}
[11] :: {{1, 2, 3, 9} = 2, {6, 7, 3, 9} = 14, {1, 7, 3, 4} = 5, {1, 7, 8, 9} = 8}
[12] :: {{1, 7, 3, 9} = 6}
[13] :: {{6, 2, 3, 4} = 9, {1, 7, 8, 9} = 8}
[14] :: {{6, 7, 3, 4} = 13, {6, 2, 3, 9} = 10, {1, 2, 8, 9} = 4, {1, 7, 8, 4} = 7}
[15] :: {{1, 2, 3, 4} = 1, {6, 7, 8, 9} = 16}
[16] :: {{1, 7, 3, 4} = 5, {6, 7, 3, 9} = 14, {1, 2, 3, 9} = 2, {1, 7, 8, 9} = 8}
```



Reduced complete graph


12
13
1
10
9
4
14
11
2
7
8
3
16
6
15
5


14
1
16
6
7
4
11
13
12
8
3
5
15
9
2
10

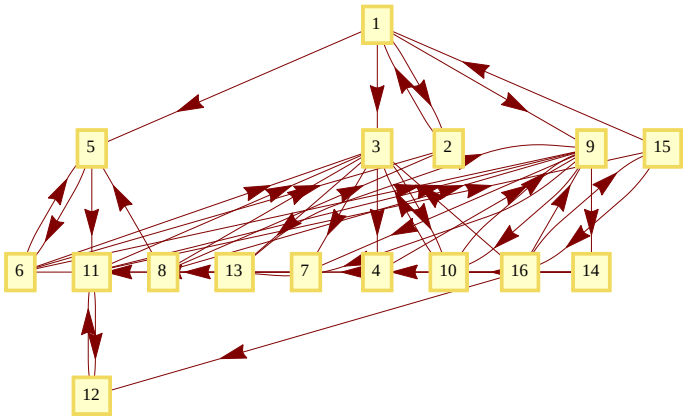

1
12
14
15
8
6
16
4
11
7
13
10
5
2
3
9

k 1 6
l 2 7
m 3 8
n 4 9

{{1, 2, 3, 9}, {1, 2, 8, 4}, {1, 7, 3, 4}, {6, 2, 3, 4}}

[1]:: {{1, 7, 3, 4} = 5, {1, 2, 8, 4} = 3, {1, 2, 3, 9} = 2, {6, 2, 3, 4} = 9}
[2]:: {{1, 2, 3, 4} = 1}
[3]:: {{6, 7, 3, 4} = 13, {1, 7, 8, 4} = 7, {1, 2, 8, 9} = 4, {6, 2, 3, 9} = 10}
[4]:: {{6, 2, 3, 4} = 9}
[5]:: {{1, 7, 3, 9} = 6, {6, 2, 8, 4} = 11}
[6]:: {{1, 2, 3, 9} = 2, {1, 2, 8, 4} = 3, {1, 7, 3, 4} = 5, {6, 2, 3, 4} = 9}
[7]:: {{1, 7, 3, 9} = 6, {6, 2, 8, 4} = 11}
[8]:: {{1, 2, 3, 9} = 2, {1, 2, 8, 4} = 3, {1, 7, 3, 4} = 5, {6, 2, 3, 4} = 9}

[9]:: {{1, 7, 8, 4} = 7, {1, 2, 8, 9} = 14, {6, 2, 3, 9} = 10, {6, 7, 3, 4} = 13}
[10]:: {{1, 2, 8, 4} = 3, {6, 2, 3, 4} = 9}
[11]:: {{6, 7, 8, 4} = 15, {1, 2, 8, 4} = 3, {6, 2, 8, 9} = 12, {6, 2, 3, 4} = 9}
[12]:: {{6, 2, 8, 4} = 11}
[13]:: {{1, 7, 8, 9} = 8, {6, 2, 3, 4} = 9}
[14]:: {{1, 2, 8, 9} = 4, {1, 7, 8, 4} = 7, {6, 7, 3, 4} = 13, {6, 2, 3, 9} = 10}
[15]:: {{6, 7, 8, 9} = 16, {1, 2, 3, 4} = 1}
[16]:: {{6, 2, 8, 9} = 12, {1, 2, 8, 4} = 3, {6, 7, 8, 4} = 15, {6, 2, 3, 4} = 9}



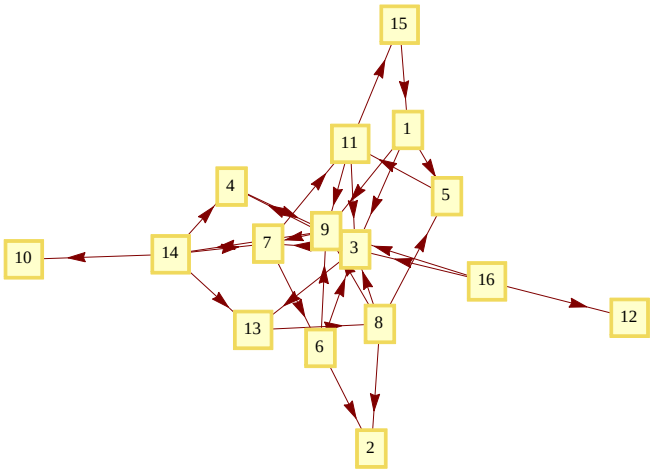
k 1 6

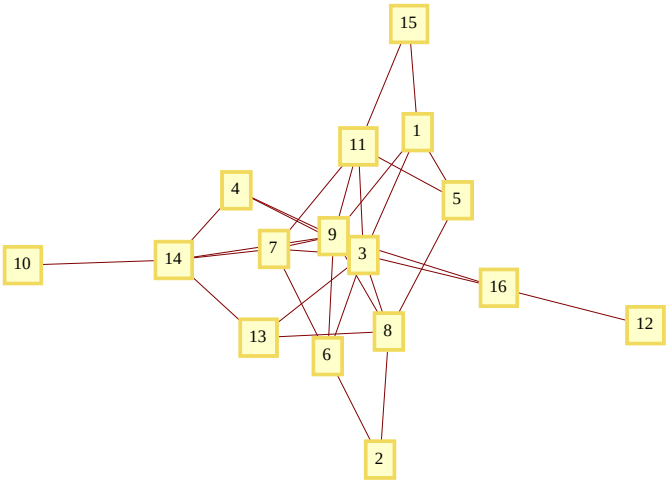
l 2 7

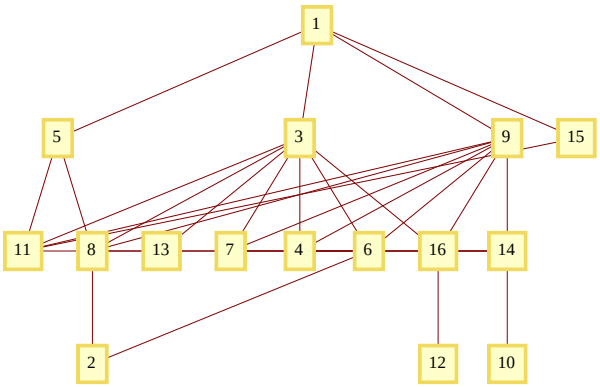
m 3 8

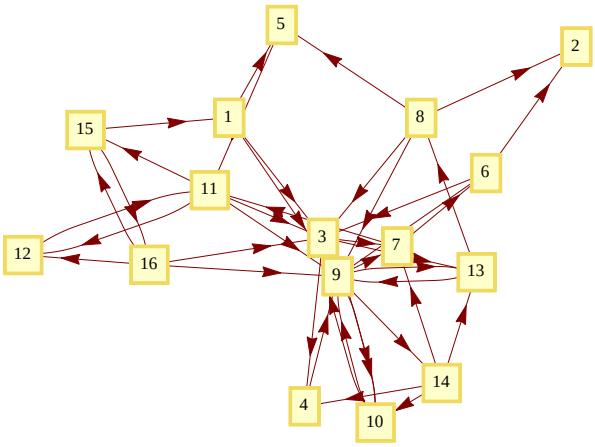
n 4 9

```
{{1, 7, 8, 4}, {1, 2, 8, 9}, {6, 2, 3, 9}, {6, 7, 3, 4}}
```









k 1 6

l 2 7

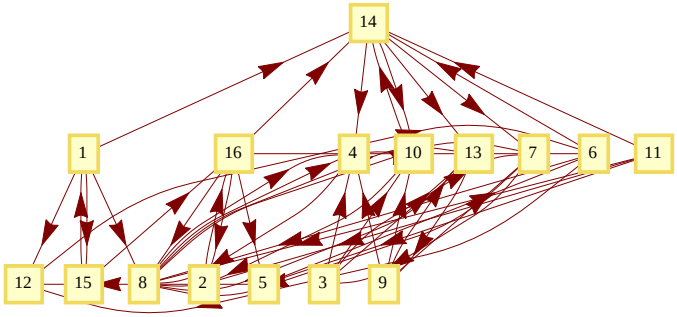
m 3 8

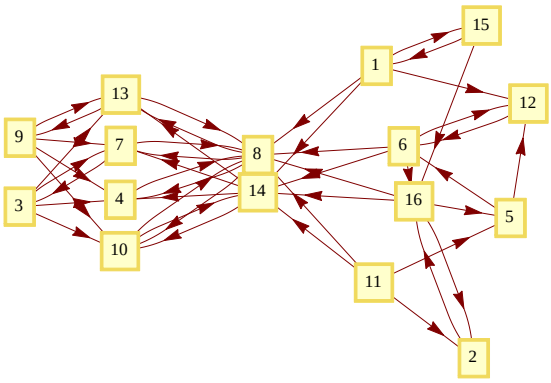
n 4 9

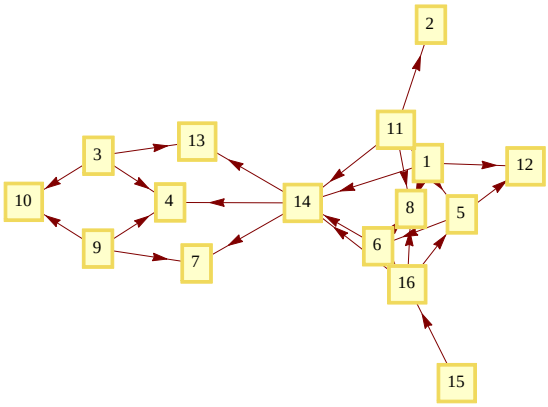
```
{{6, 7, 8, 4}, {6, 7, 3, 9}, {6, 2, 8, 9}, {1, 7, 8, 9}}
```

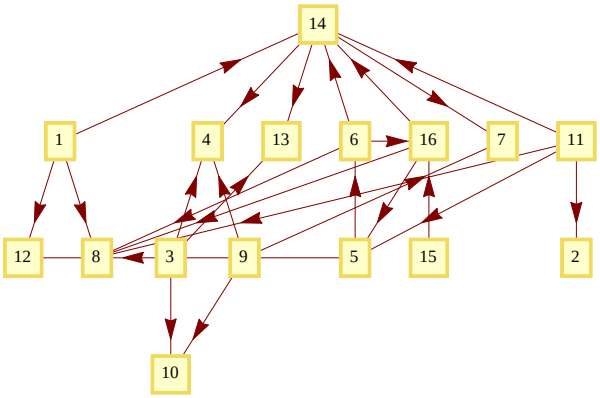
[1]:: {{6, 2, 8, 9} = 12, {6, 7, 3, 9} = 14, {6, 7, 8, 4} = 15, {1, 7, 8, 9} = 8}
[2]:: {{6, 7, 8, 9} = 16}
[3]:: {{1, 2, 8, 9} = 4, {6, 2, 3, 9} = 10, {6, 7, 3, 4} = 13, {1, 7, 8, 4} = 7}
[4]:: {{1, 7, 8, 9} = 8}
[5]:: {{6, 2, 8, 4} = 12, {1, 7, 3, 9} = 6}
[6]:: {{6, 7, 8, 4} = 16, {6, 7, 3, 9} = 14, {6, 2, 8, 9} = 12, {1, 7, 8, 9} = 8}
[7]:: {{1, 2, 8, 4} = 3, {1, 7, 8, 9} = 8}
[8]:: {{1, 7, 8, 4} = 7, {6, 7, 3, 4} = 13, {6, 2, 3, 9} = 10, {1, 2, 8, 9} = 4}

[9]:: {{6, 2, 3, 9} = 10, {6, 7, 3, 4} = 13, {1, 7, 8, 4} = 7, {1, 2, 8, 9} = 4}
[10]:: {{6, 7, 3, 9} = 14, {1, 7, 8, 9} = 8}
[11]:: {{1, 2, 3, 9} = 2, {6, 7, 3, 9} = 14, {1, 7, 3, 4} = 5, {1, 7, 8, 9} = 8}
[12]::{{1, 7, 3, 9} = 6}
[13]:: {{6, 2, 3, 4} = 9, {1, 7, 8, 9} = 8}
[14]:: {{6, 7, 3, 4} = 13, {6, 2, 3, 9} = 10, {1, 2, 8, 9} = 4, {1, 7, 8, 4} = 7}
[15]:: {{1, 2, 3, 4} = 1, {6, 7, 8, 9} = 16}
[16]:: {{1, 7, 3, 4} = 5, {6, 7, 3, 9} = 14, {1, 2, 3, 9} = 2, {1, 7, 8, 9} = 8}









Graph of self-modifying morphoCAs of ruleDM

Self - modification of single DM rules

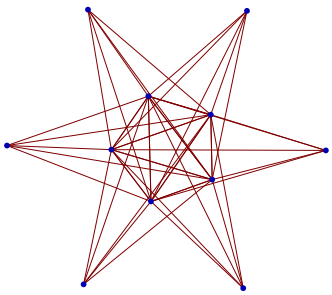
Encountered a single constellation of morphograms, for ruleDM[{a,b,c,d,e}], what is the possible field of self-modification in respect to its environment?

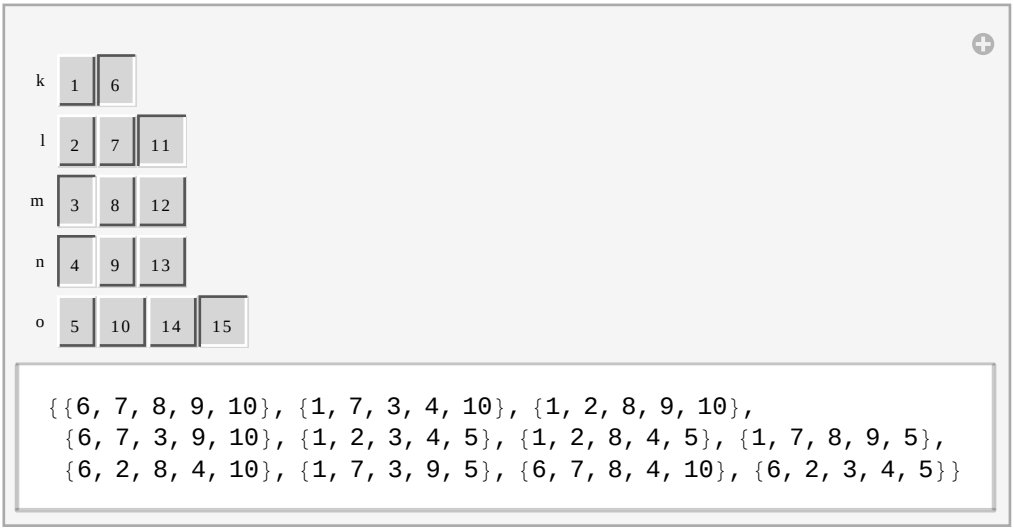
With an environment init={1,0,0,0,2}, there are for the ruleDM[{6,11,3,4,15}] exactly 11 self-modifications of the rule possible.

The new field is presented with:

```
{{6, 7, 8, 9, 10}, {1, 7, 3, 4, 10}, {1, 2, 8, 9, 10},
 {6, 7, 3, 9, 10}, {1, 2, 3, 4, 5}, {1, 2, 8, 4, 5}, {1, 7, 8, 9, 5},
 {6, 2, 8, 4, 10}, {1, 7, 3, 9, 5}, {6, 7, 8, 4, 10}, {6, 2, 3, 4, 5}}
```

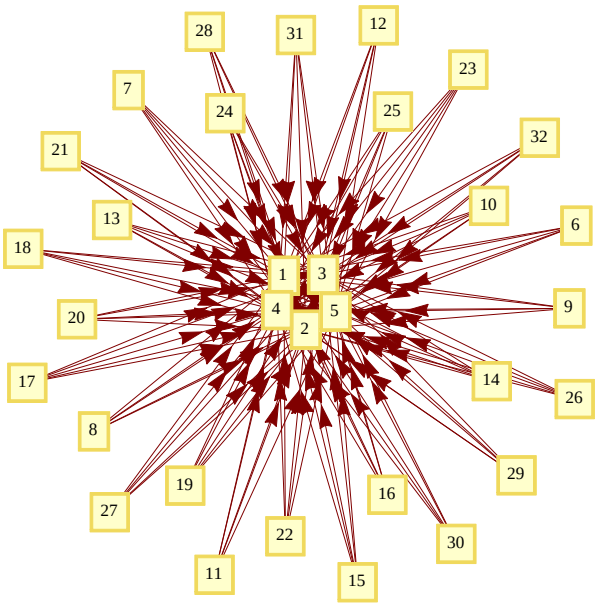
The structural graph of ruleDM[{6,11,3,4,15}] is given by *GraphPlot*

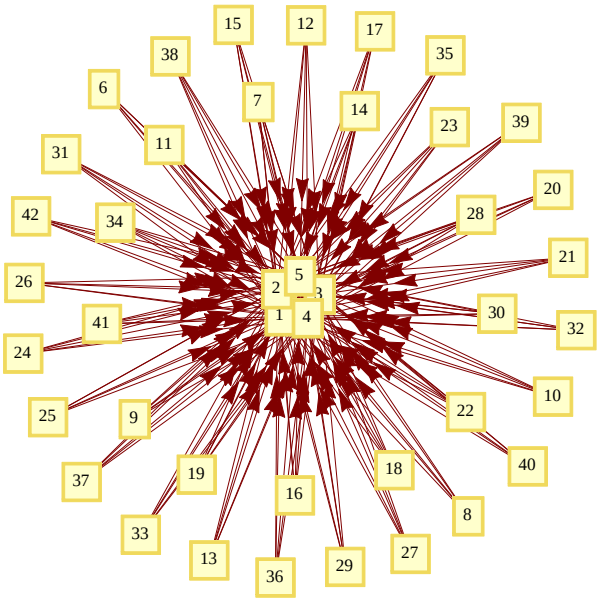


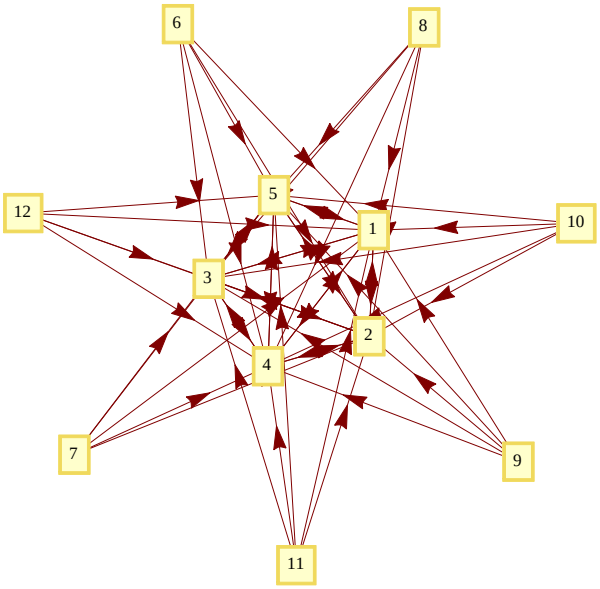



```
{{6, 7, 8, 9, 10}, {1, 7, 3, 4, 10}, {1, 2, 8, 9, 10},
 {6, 7, 3, 9, 10}, {1, 2, 3, 4, 5}, {1, 2, 8, 4, 5}, {1, 7, 8, 9, 5},
 {6, 2, 8, 4, 10}, {1, 7, 3, 9, 5}, {6, 7, 8, 4, 10}, {6, 2, 3, 4, 5}}
```

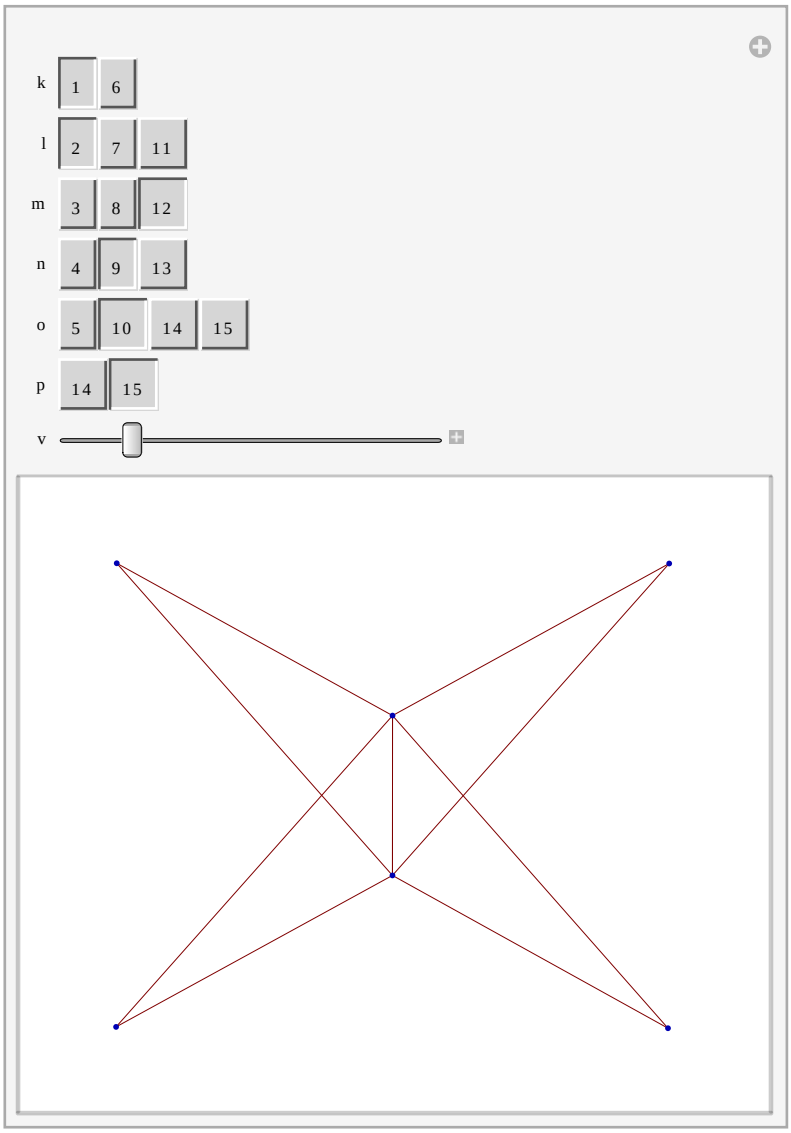
Full GraphPlot for [1, 2, 3, 4, 5/10/14/15]



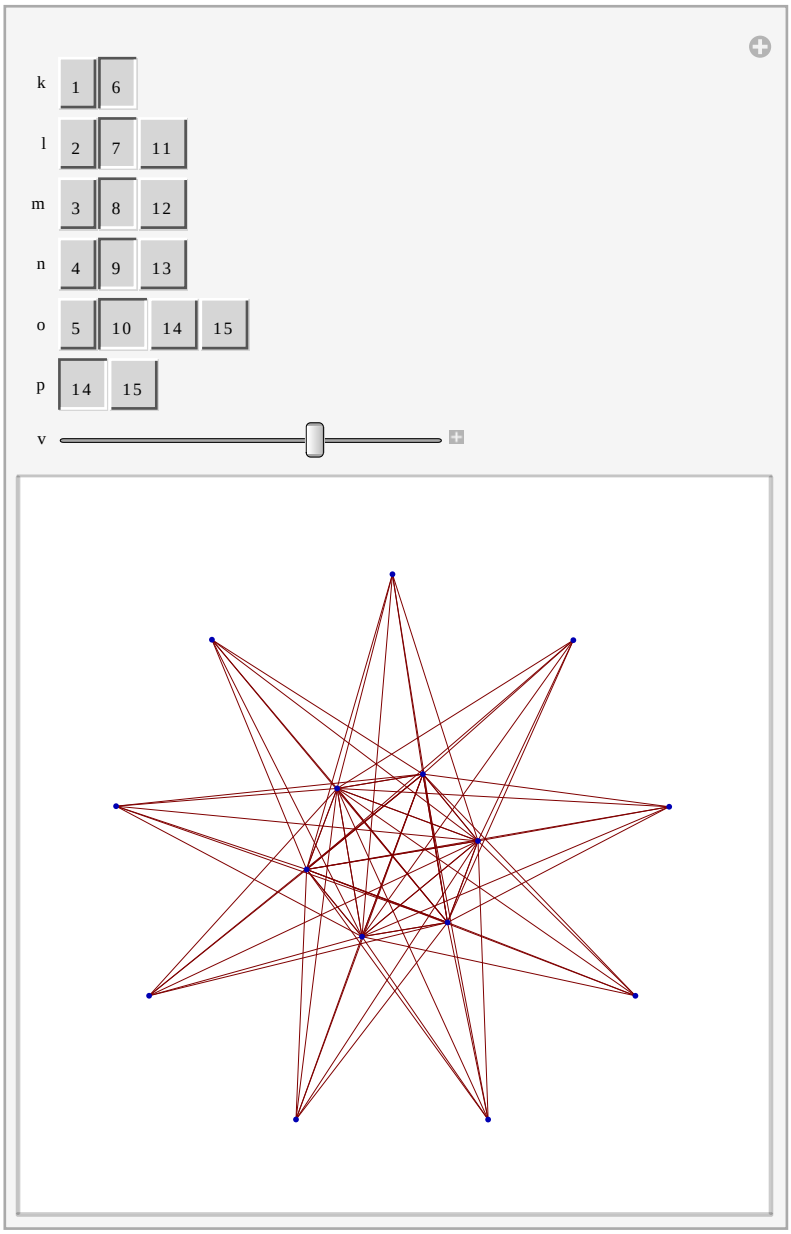

15
12
17
38
35
6
7
14
23
39
11
31
20
42
34
28
21
26
5
2
30
41
1
4
32
24
10
25
9
22
40
37
19
18
16
8
33
13
27
36
29


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

init = {1, 0, 1, 1, 2, 2}



init = {1, 0, 0, 1, 0, 2}


k
1
6
l
2
7
11
m
3
8
12
n
4
9
13
o
5
10
14
15
p
14
15
v

Generalizations

ruleDMN

k 1 6

l 2 7 11

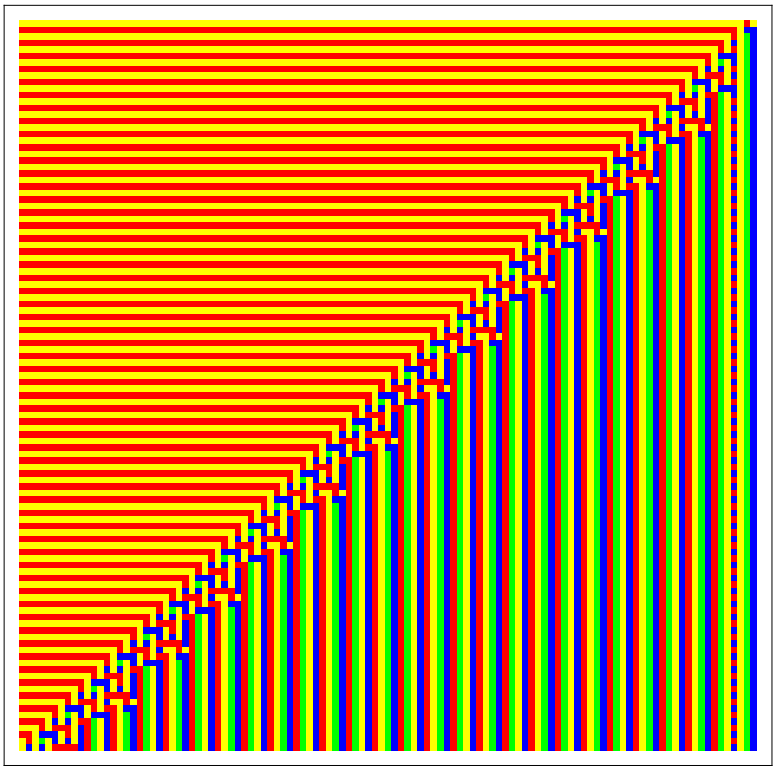
m 3 8 12

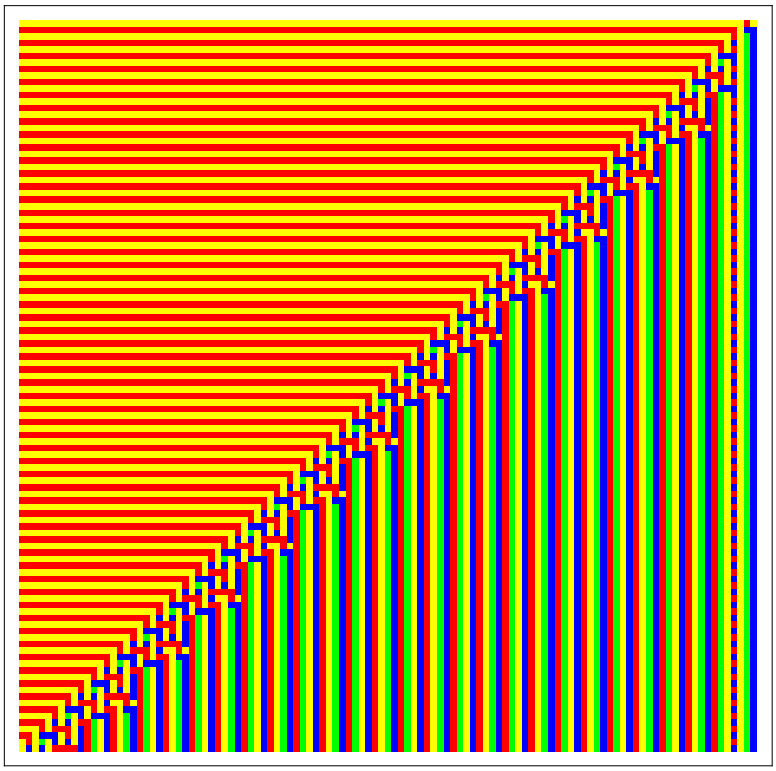
n 4 9 13

o 5 10

p 14 15

steps 111





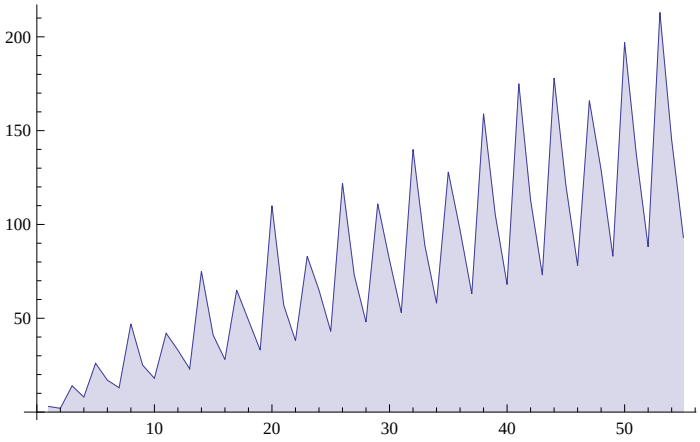
m

{{1, 2, 2, 2, 3, 3}, 489}

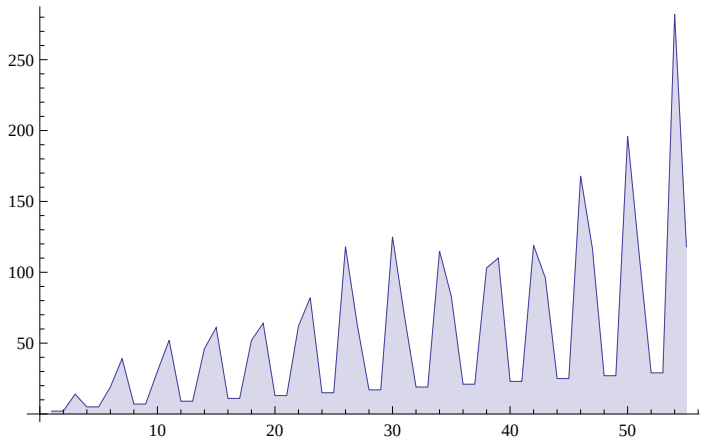
```
RuleTableFromkAryMorphoMN[kAryFromRuleTableMorpho[
RuleTableFromkAryMorphoMN[{2,2,2,3,3,1}]]]/.mrules
```

{6, 7, 8, 13, 14, 5}

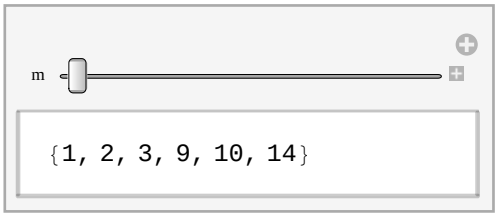
```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleDMN[{6, 7, 8, 13, 14, 5}]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```

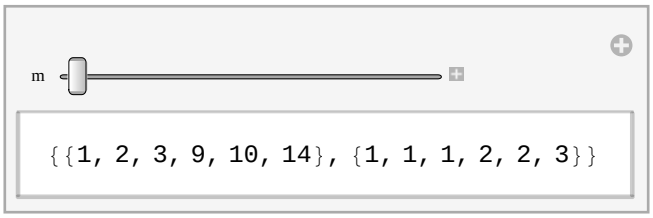


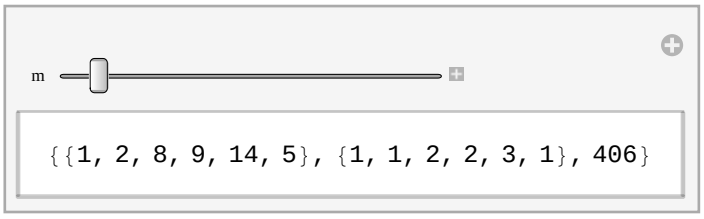
```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
       ruleDMN[{1, 7, 8, 13, 14, 15}]],
     SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```




```
Manipulate[RuleTableFromkAryMorphoMN[ReLabel[Mod[Flatten[
CellularAutomaton[
ruleDMN[{1,7,8,9,14,5}],{1,2,2,2,3,3},{{m}}]],4]]/.mrules,{m,1,222,1},
SaveDefinitions→True]
```








```
Table[{RuleTableFromkAryMorphoMN[ReLabel[Mod[Flatten[
CellularAutomaton[
ruleDMN[{1,7,8,9,14,5}],{1,2,2,2,3,3},{{m}}]],3]]/.mrules,
Mod[ReLabel[Flatten[CellularAutomaton[
ruleDMN[{1,7,8,9,14,5}],{1,2,2,2,3,3},{{m}}]]],4]},{m,1,216,1}]
```

Table example

```
{{1, 2, 3, 9, 10, 14}, {1, 1, 1, 2, 2, 3}}, {{1, 2, 8, 9, 14, 5}, {1, 1, 2, 2, 3, 1}},
{{1, 7, 8, 13, 5, 5}, {1, 2, 2, 3, 1, 1}}, {{1, 2, 8, 13, 14, 14}, {1, 1, 2, 3, 3, 3}},
{{1, 7, 12, 13, 14, 5}, {1, 2, 3, 3, 3, 1}}, {{1, 7, 8, 9, 14, 14}, {1, 2, 2, 2, 3, 3}},
{{1, 2, 3, 9, 10, 14}, {1, 1, 1, 2, 2, 3}}, {{1, 2, 8, 9, 14, 5}, {1, 1, 2, 2, 3, 1}},
{{1, 7, 8, 13, 5, 5}, {1, 2, 2, 3, 1, 1}}, {{1, 2, 8, 13, 14, 14}, {1, 1, 2, 3, 3, 3}},
{{1, 7, 12, 13, 14, 5}, {1, 2, 3, 3, 3, 1}}, {{1, 7, 8, 9, 14, 14}, {1, 2, 2, 2, 3, 3}}
```

```
Table[RuleTableFromkAryMorphoMN[ReLabel[Mod[Flatten[
CellularAutomaton[ruleMN[{6,7,8,9,15,5}],{1,2,2,2,3,3},{{m}}]],3]]]
/.mrules,{m,1,111,1}]//MatrixForm
```

```
Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6,7,8,9,15,5}],{1,2,2,2,3,3},{{m}}]]]]
/.mrules,{m,1,111,1}]//MatrixForm
```

```
Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6,7,8,9,15,5}],{1,2,2,2,3,3},{{m}}]]]]
/.mrules,{m,1,111,1}]
```

```
FormDynInit[ruleMN[{1, 7, 8, 9, 11, 5}], {0, 1, 1, 1, 1, 2}]
```

```
{G, P, O, N, M, L, K}
```

```
FormDynInit[
 ruleMN[{1, 7, 8, 9, 11, 5}], {0, 1, 1, 1, 1, 2}]

 {"G", "P", "O", "N", "M", "L", "K"}
```

```
DeleteDuplicates[Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6,7,8,9,15,5}],{1,2,2,2,3,3},{{m}}]]]]
/.mrules,{m,1,111,1}]/.filterMN]
```

```
FormDynInit[ruleMN[{6, 7, 8, 9, 15, 5}], {1, 2, 2, 2, 3, 3}]
```

```
{R, n, o, p, q}
```

```
 FormDynInit[
 ruleMN[{6, 7, 8, 9, 15, 5}], {1, 2, 2, 2, 3, 3}] =

                {R, n, o, p, q}
```

```
FormDynInit[ruleMN[{6, 7, 8, 9, 5, 15}], {1, 1, 1, 1, 2, 3}]
```

```
{r, s, K, P, O, N, M, L}
```

Examples

```
{1, 7, 3, 13, 14, 15},
{1, 7, 12, 4, 14, 10},
{1, 7, 8, 4, 10, 10},
{1, 7, 3, 4, 10, 5},

{1, 2, 8, 4, 5, 10},
{1, 7, 8, 4, 10, 10},
{1, 7, 3, 4, 10, 5},
{1, 2, 8, 4, 5, 10},

{1, 7, 8, 4, 10, 10},
{1, 7, 3, 4, 10, 5},
{1, 2, 8, 4, 5, 10},
{1, 7, 8, 4, 10, 10},

{1, 7, 3, 4, 10, 5},
{1, 2, 8, 4, 5, 10},
{1, 7, 8, 4, 10, 10},
{1, 7, 3, 4, 10, 5},
```

```
1  7  3  13  14  15
1  7  12  9  15  10
1  7  8  13  10  5
1  7  3  4  14  15
```

```
Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6, 11, 12, 9,5, 15}],{1,2,2,2,3,3},{{m}}]]]]
/.mrules,{m,1,222,1}]//MatrixForm
```

```
Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6, 11, 12, 9,5, 15}],{1,2,2,2,3,3},{{m}}]]]]
/.mrules,{m,1,222,1}]//TableForm
```

```
Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6, 11, 12, 9,5, 15}],{1,2,2,2,3,3},{{m}}]]]]
/.mrules,{m,1,163,1}]
```

```
Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6, 11, 12, 9,5, 15}],{1,2,2,2,3,3},{{m}}]]]]
/.mrules,{m,1,163,1}]//MatrixForm
```

```
Print[Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6, 11, 12, 9,5, 15}],{1,2,2,2,3,3},{{m}}]]]]
/.mrules,{m,1,163,1}]/.filterMN]
```

```
FormDynInit (ruleMN[{6, 11, 12, 9, 5, 15}], {1, 2, 2, 2, 3, 3}) =

 {{Q, R, S, T, U, V},
  {W, X, Y, Z, a, b, c, d, e, B, f, g, h, i, j, k, l, m}^(m)}
```

```
DeleteDuplicates[Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6, 11, 12, 9,5, 15}],{1,2,2,2,3,3},{{m}}]]]]
/.mrules,{m,1,163,1}]/.filterMN]
```

```
{Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, B, f, g, h, i, j, k, l, m}
```

Palindrome of FormDyn(ruleMN[{6, 11, 12, 9,5, 15}]

```
ReLabel[
  Reverse[{"Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "a", "b",
    "c", "d", "e", "B", "f", "g", "h", "i", "j", "k", "l", "m"}]] ==
 ReLabel[{"Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "a", "b",
   "c", "d", "e", "B", "f", "g", "h", "i", "j", "k", "l", "m"}]
```

True

```
ReLabel[
  Reverse[{"Q", "R", "S", "T", "U", "V"}]] ==
 ReLabel[{"Q", "R", "S", "T", "U", "V"}]
```

True

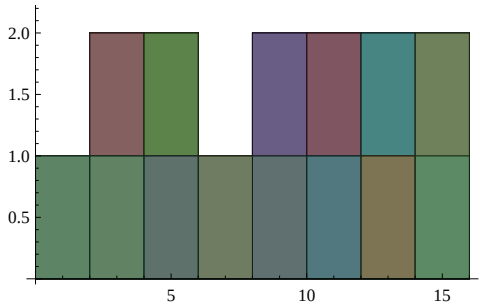
```
ReLabel[
  Reverse[{"X", "Y", "Z", "a", "b", "c",
    "d", "e", "B", "f", "g", "h", "i", "j", "k", "l", "m"}]] ==
 ReLabel[{"X", "Y", "Z", "a", "b", "c", "d", "e", "B",
   "f", "g", "h", "i", "j", "k", "l", "m"}]
```

True

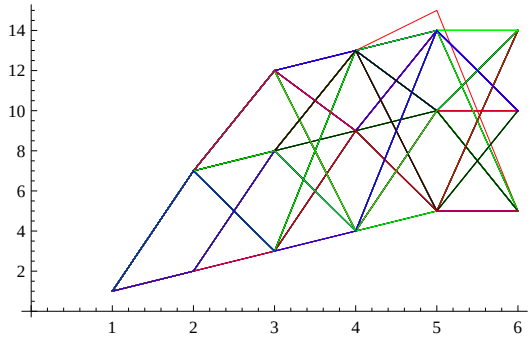
Example for ruleMN

```
{1, 7, 12, 13, 5, 14},
{1, 7, 3, 13, 14, 15},
{1, 7, 12, 9, 15, 10},
{1, 7, 8, 13, 10, 5},
{1, 7, 3, 4, 14, 15},
{1, 2, 8, 4, 10, 14},
{1, 7, 8, 9, 5, 14},
{1, 7, 12, 4, 10, 10},
{1, 7, 12, 4, 14, 5},
{1, 7, 12, 13, 14, 10},
{1, 7, 3, 13, 10, 5},
{1, 2, 8, 13, 5, 14},
{1, 7, 3, 13, 14, 14},
{1, 2, 8, 4, 14, 10},
{1, 7, 8, 13, 5, 10},
{1, 7, 12, 9, 5, 5},
{1, 7, 8, 4, 10, 14},
{1, 7, 3, 4, 14, 10},
{1, 2, 8, 13, 10, 5},
{1, 7, 12, 13, 10, 14},
{1, 7, 3, 9, 10, 14},
{1, 2, 3, 9, 14, 10},
{1, 7, 12, 4, 5, 14},
{1, 7, 12, 9, 14, 14},
{1, 7, 8, 9, 5, 14},
{1, 7, 12, 4, 10, 10},
{1, 7, 12, 4, 14, 5},
{1, 7, 12, 13, 14, 10},
{1, 7, 3, 13, 10, 5},
{1, 2, 8, 13, 5, 14},
{1, 7, 3, 13, 14, 14},
{1, 2, 8, 4, 14, 10},
{1, 7, 8, 13, 5, 10}
```

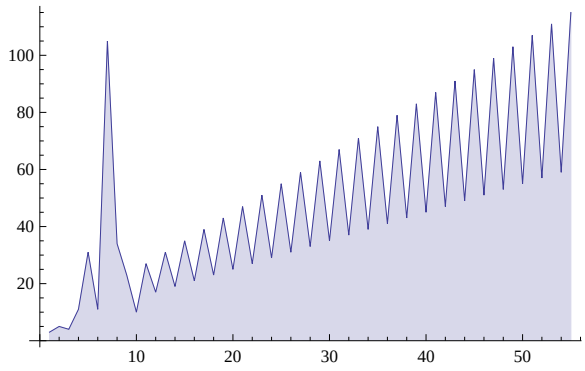
```
Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6, 11, 12, 9, 5, 15}],{2,3,3,2,2,3},{{m}}]]]]
/.mrules,{m,1,261,1}]//TableForm
```




```
ListLinePlot[Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6, 11, 12, 9,5, 15}],{2,3,3,2,2,3},{{m}}]]]]
/.mrules,{m,1,111,1}],
PlotStyle->{Red,Green,Blue, Black},ImageSize→Medium]
```



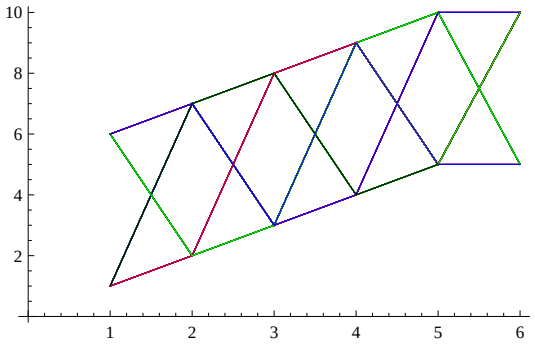

```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleMN[{6, 11, 12, 9, 5, 15}]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```



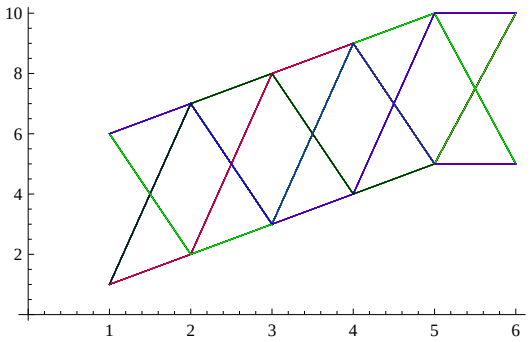
Behavioral commutativity and non-commutativity of MN sub-rules

Commutativity

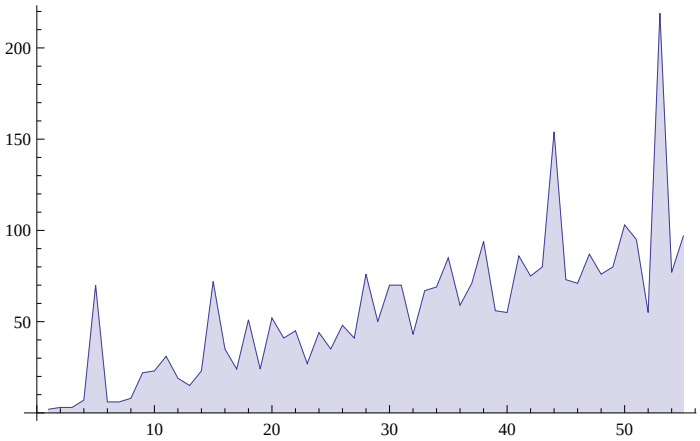
```
ListLinePlot[Table[RuleTableFromkAryMorphoMN[Mod[Flatten[
CellularAutomaton[ruleMN[{1, 2, 8, 4, 14, 15}],ReLabel[{2,3,3,2,2,3}],{{m}}]],4]]
/.mrules,{m,1,111,1}],PlotStyle->{Red,Green,Blue, Black},ImageSize→Medium]
```



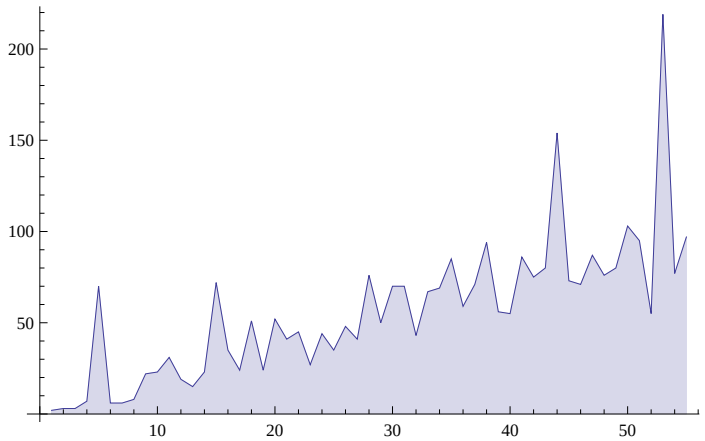

```
ListLinePlot[Table[RuleTableFromkAryMorphoMN[Mod[Flatten[
CellularAutomaton[ruleMN[{8, 2, 1, 4, 14, 15}],ReLabel[{2,3,3,2,2,3}],{{m}}]],4]]
/.mrules,{m,1,111,1}],PlotStyle->{Red,Green,Blue, Black},ImageSize→Medium]
```




```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
      ruleMN[{1, 11, 12, 13, 10, 15}]],
    SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```

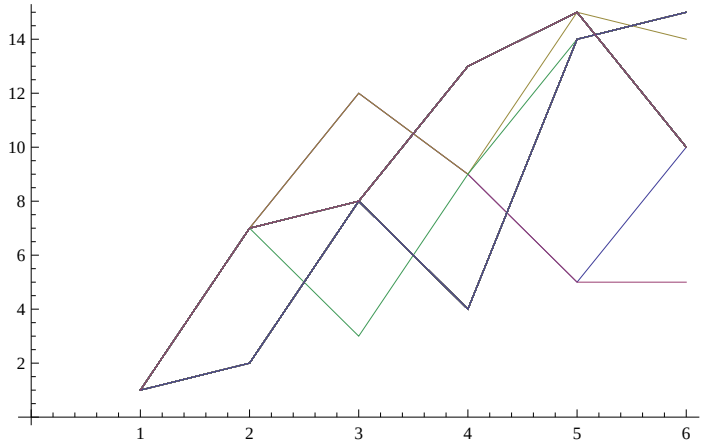


```
ListLinePlot[Table[Length[NestWhileList[m = CellularAutomaton[
       ruleMN[{11, 1, 12, 13, 10, 15}]],
     SparseArray[1 -> 1, n], Unequal, All]], {n, 55}], Filling → Axis]
```

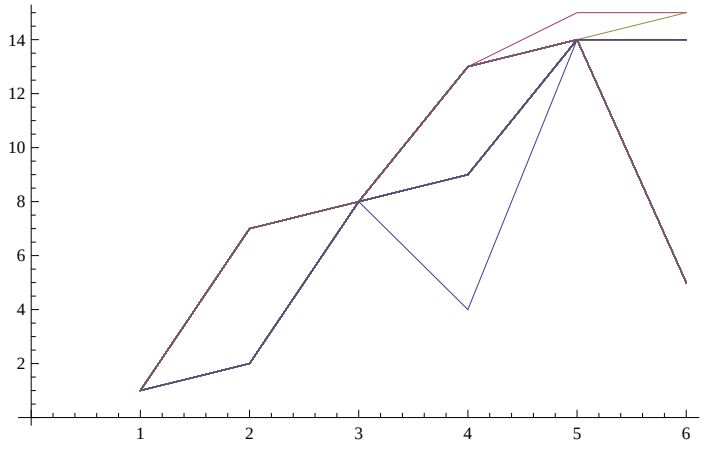


Non - commutativity

```
ListLinePlot[Table[RuleTableFromkAryMorphoMN[Mod[ReLabel[Flatten[
CellularAutomaton[ruleMN[{1, 11, 12, 13, 10, 15}],{1,2,2,2,3,3},{{m}}]]],5]]
/.mrules,{m,1,163,1}]]
```




```
ListLinePlot[Table[RuleTableFromkAryMorphoMN[Mod[ReLabel[Flatten[
CellularAutomaton[ruleMN[{11, 1, 12, 13, 15, 10}], {1, 2, 2, 2, 3, 3}, {{m}}]]],
     5]] /. mrules, {m, 1, 163, 1}]]
```



Form of Dynamics : FormDyn

```
Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{1, 2, 8, 4,14, 15}],{2,3,3,2,2,3},{{m}}]]]]
/.mrules,{m,1,111,1}]/.filterMN
```

```
{K, L, M, N, O, P, K, L, M, N, O, P, K, L, M, N, O, P, K, L, M, N, O,
 P, K, L, M, N, O, P, K, L, M, N, O, P, K, L, M, N, O, P, K, L, M, N, O, P,
 N, O, P, K, L, M, N, O, P, K, L, M, N, O, P, K, L, M, N, O, P, K, L, M, N, O, P,
 K, L, M, N, O, P, K, L, M, N, O, P, K, L, M, N, O, P, K, L, M, N, O, P, K, L, M}
```

```
FormDynInit[ruleMN[{1, 2, 8, 4, 14, 15}], {2, 3, 3, 2, 2, 3}]
```

```
{K, L, M, N, O, P}
```

```
FormDynInit[
   ruleMN[{1, 2, 8, 4, 14, 15}], {2, 3, 3, 2, 2, 3}] =

                {K, L, M, N, O, P}
```

```
Table[RuleTableFromkAryMorphoMN[ReLabel[Flatten[
CellularAutomaton[ruleMN[{1, 2, 8, 4,14, 15}],{2,3,3,2,2,3},{{m}}]]]]
/.mrules,{m,1,111,1}]//TableForm
```

```
1   2   8   9   5    10
1   7   8   4   10   5
1   2   8   4   10   10
1   7   3   9   10   5
1   7   3   4   10   10
1   7   8   4   5    10
1   2   8   9   5    10
```

Palindrome of FormDyn[ruleMN[{1, 2, 8, 4, 14, 15}]]

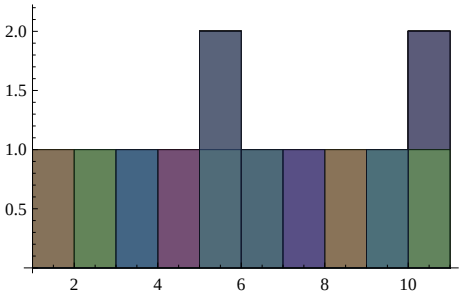
```
ReLabel[
  Reverse[{"K", "L", "M", "N", "O", "P"}]] ==
 ReLabel[{"K", "L", "M", "N", "O", "P"}]
```

```
True
```

```
Table[RuleTableFromkAryMorphoMN[Mod[Flatten[
CellularAutomaton[ruleMN[{1, 2, 8, 4, 14, 15}],ReLabel[{2,3,3,2,2,3}],{{m}}]],4]]
/.mrules,{m,1,111,1}]//TableForm
```

```
1   2   8   9   5    10
6   2   3   9   5    10
6   7   3   9   5    5
1   7   3   9   10   5
1   7   3   4   10   10
```

```
Histogram[Table[RuleTableFromkAryMorphoMN[Mod[Flatten[
CellularAutomaton[ruleMN[{1, 2, 8, 4,14, 15}],ReLabel[{2,3,3,2,2,3}],{{m}}]],4]]
/.mrules,{m,1,111,1}]]
```




```
Table[RuleTableFromkAryMorphoMN[Mod[ReLabel[Flatten[
CellularAutomaton[ruleMN[{1, 11, 12, 13, 10, 15}],{1,2,2,2,3,3},{{m}}]]],5]]
/.mrules,{m,1,163,1}]/.filterMN
```

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

```
DeleteDuplicates[Table[RuleTableFromkAryMorphoMN[Mod[ReLabel[Flatten[
CellularAutomaton[ruleMN[{1, 11, 12, 13, 10, 15}],{1,2,2,2,3,3},{{m}}]]],5]]
/.mrules,{m,1,163,1}]/.filterMN]
```

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

$$\text{FormDyn}\left[\text{ruleMN}[\{1,\ 11,\ 12,\ 13,\ 10,\ 15\}] = \left[(A, B, C, D), (E, F)^{(n)}\right]\right]$$

```
Table[RuleTableFromkAryMorphoMN[Mod[ReLabel[Flatten[
CellularAutomaton[ruleMN[{6, 11, 12, 13, 10, 15}],{1,2,2,2,3,3},{{m}}]]],5]]
/.mrules,{m,1,163,1}]/.filterMN
```

```
FormDynInit[ruleMN[{6, 11, 12, 13, 10, 15}],{1,2,2,2,3,3}]
```

```
{G, H, I}
```

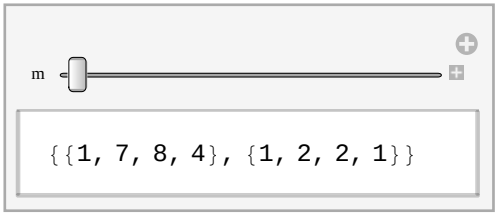
$$\text{FormDynInit}\left[\text{ruleMN}[\{6,\ 11,\ 12,\ 13,\ 10,\ 15\}],\ \{1,\ 2,\ 2,\ 2,\ 3,\ 3\}\right] = \left[(\text{"G"}, \text{"H"}), (I)^{(n)}\right]$$

Loops and palindromes for ruleDCl

“This possibility of a simultaneity of semiotic asymmetry and morphic symmetry is the new challenge to the theory of computation.”

http://memristors.memristics.com/Godels%20Palindromes/Godels%20Palindromes.html

```
Manipulate[{RuleTableFromkAryMorphoDCl[Mod[Flatten[
CellularAutomaton[
ruleDCl[{1,7,8,9}],{1,2,2,2},{{m}}]],3]]/.mrules,
Mod[ReLabel[Flatten[CellularAutomaton[
ruleDCl[{1,7,8,9}],{1,2,2,2},{{m}}]]],3]},{m,1,216,1},
 SaveDefinitions→True]
```




```
Table[RuleTableFromkAryMorphoDCl[ReLabel[Flatten[
CellularAutomaton[ruleDCl[{6,7,8,9}],{2,2,2,2},{{m}}]]]]/.
mrules,{m,1,16,1}]//MatrixForm
```

```
DeleteDuplicates[Table[RuleTableFromkAryMorphoDCl[ReLabel[Flatten[
CellularAutomaton[ruleDCl[{6,7,8,4}],{2,2,2,2},{{m}}]]]]/.
mrules,{m,1,16,1}]]//MatrixForm
```

(1 2 3 4)

Iteration: (1 2 3 4) ⟹ (1 2 3 4)

```
Table[RuleTableFromkAryMorphoDCl[Flatten[
CellularAutomaton[ruleDCl[{1,7,8,9}],{1,2,2,2},{{m}}]]]/.
mrules,{m,1,16,1}]//MatrixForm
```

Loops and morphic palindromes for ruleDCl[{1,7,8,9}]

1 7 8 4A	6 7 3 4B	6 2 3 9C	1 2 8 9D
6 7 3 4B	6 2 3 9C	1 2 8 9D	1 7 8 4A
6 2 3 9C	1 2 8 9D	1 7 8 4A	6 7 3 4B
1 2 8 9D	1 7 8 4A	6 7 3 4B	6 2 3 9C
1 7 8 4A	6 7 3 4B	6 2 3 9C	1 2 8 9D

ruleDCl forms: {ABCDA, BCDAB, CDABC, DABCD} :: {1,2,3,4,1}

```
ReLabel[{A,B,C,D,A}] == ReLabel[{B,C,D,A,B}] ==
ReLabel[{C,D,A,B,C}] == ReLabel[{D,A,B,C,D}]
```

True

```
ReLabel[Reverse[{A, B, C, D, A}]] == ReLabel[{B, C, D, A, B}]
```

True

Each component of the forms of ruleDCl is iterative. It repeats itself in an end - to - begin loop. But morphogrammatically, each component is morphogramatically palindromic too. It reads in both directions and keeps its morphic equivalence stable.

This leads to a morphogrammatic concept of fixed points where the loop runs at once in both directions of the interaction.

The form of fixed-points: FormDyn

A further conceptualization of fixed-points as morphograms leads to an explicit characterization of the form of the fixed-point palindromes.

Palindromic	iterative loops
$\{A, B, C, D, A\}$ ↕ ↕ $\{A, B, C, D, A\}$	$\{A, B, C, D, A\} \Longrightarrow \{A, B, C, D, A\}$

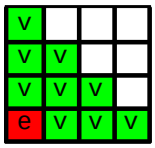
Chain of palindromic and/or iterative loops

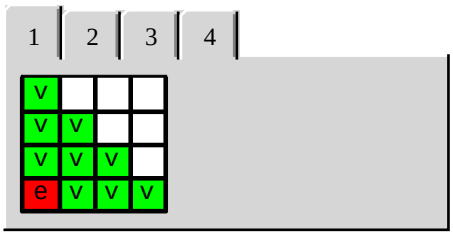
ABCDA,
↕
BCDAB,
↕
CDABC,
↕
DABCD

ENstructure of ABCDA

A more adequate modeling of the morphogrammatic is obtained with the introduction of the e/v-structure replacing the ReLabel construct.

```
- ENstructureEN["A","B","C","D","A"];
val it = [[],[N],[N,N],[N,N,N],[E,N,N,N]] : EN list list
```



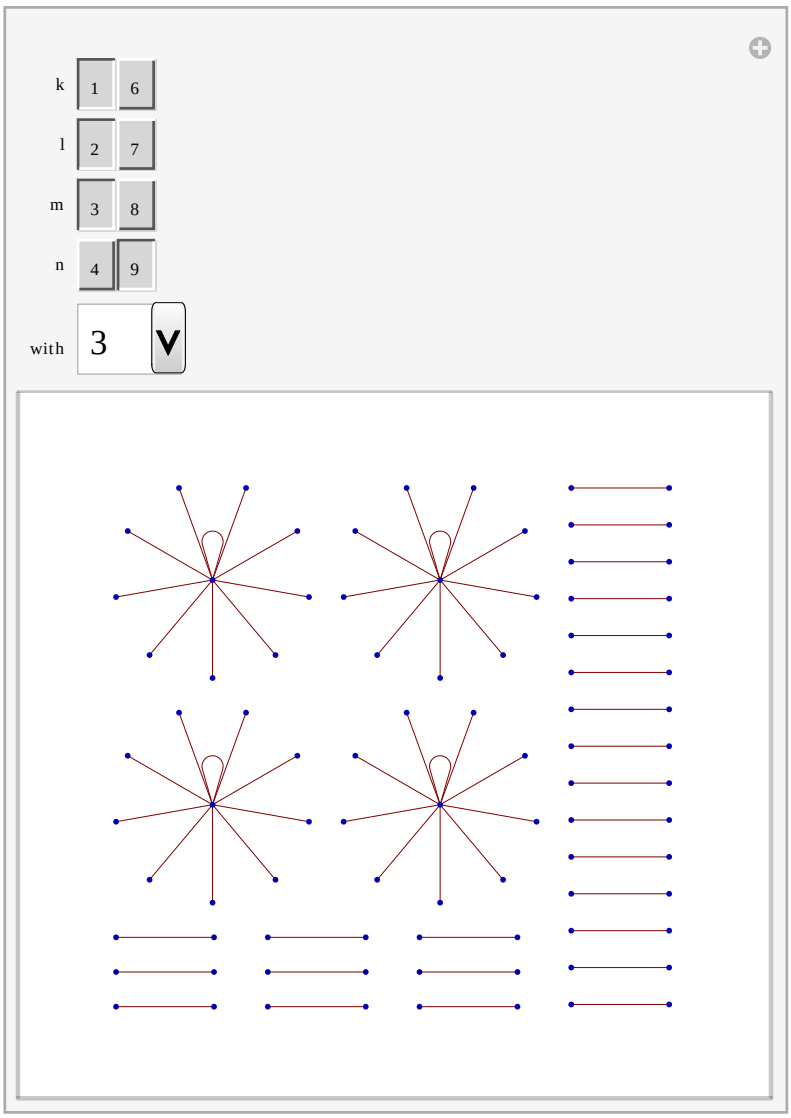


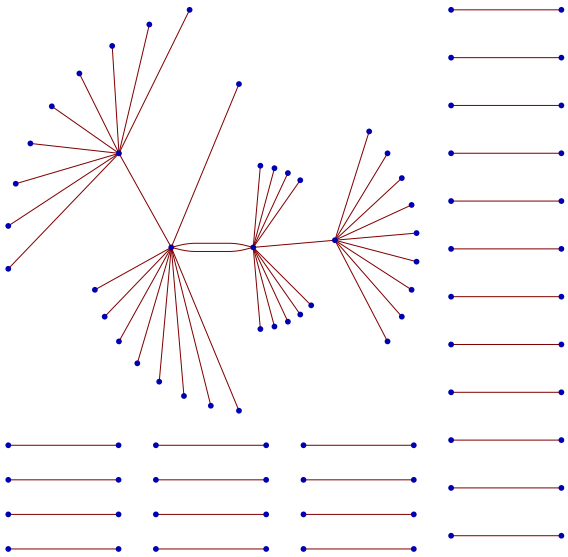
Composed palindromes

FormDyn[ruleMN[{1, 11, 12, 13, 10, 15}] =

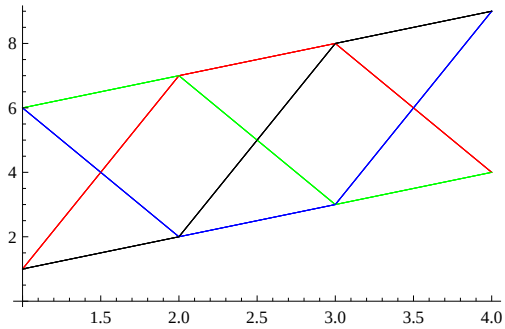
$\left[(A, B, C, D), (E, F)^{(n)}\right]$

Transition graphs and plots for DCl





```
ListLinePlot[Table[RuleTableFromkAryMorphoDCl[Flatten[
CellularAutomaton[ruleDCl[{1,7,8,9}],{1,2,2,2},{{m}}]]]/.mrules,{m,1,16,1}],
PlotStyle->{Red,Green,Blue, Black},ImageSize→Medium]
```




```
Histogram[Table[RuleTableFromkAryMorphoDCl[Flatten[
CellularAutomaton[ruleDCl[{1,7,8,9}],{1,2,2,2},{{m}}]]]/.mrules,{m,1,16,1}]]
```