A Measure for the Complexity of Elementary Cellular Automata

Thorsten Ewert
Lübeck, Germany
tewert@web.de

A new measure for the complexity of elementary cellular automata (ECAs) is presented. This measure is based on the minimization of Boolean functions with three variables that represent the elementary cellular automaton (ECA) rules. The minimized Boolean functions reduce the number of input bits of the truth table, which is equivalent to the rule table of an ECA. This results in a fractalized number of Boolean variables that are equal to the state variables of a dynamic system. Furthermore, the dynamic nature of complexity in ECAs is considered. Therefore, a new method of defining and deriving the complexity of all 256 ECA rules given in bits is proposed. The results then can be described, classified and grouped. As for other continuous or discrete dynamic systems, the complexity grows with the number and the usage of the state variables. In ECAs, the numbers of the effective state variables range from 0 to 3, resulting in four classes of behavior.

Keywords: complexity; measure; elementary cellular automaton

1. Introduction

Cellular automata (CAs) were first introduced as a mathematical model for biological self-replication phenomena by von Neumann in the early 1950s and published after his death by Burks [1]. Nowadays, CAs are also mathematical models used to simulate complex systems or processes. Using extensive computer simulation, Wolfram classified CAs in his book *A New Kind of Science* [2] into four classes according to the qualitative behavior of their evolution (see [2, Chapter 6, pp. 231–235]). Later on he resorted them into class I (uniform state, low complexity), class II (repetitive or stable state, low complexity), class IV (local complexity, repetitive, intermediate complexity) and class III (global complexity, chaotic, great complexity). He wrote: "For while class 4 is above class 3 in terms of apparent complexity, it is in a sense intermediate between class 2 and 3 in terms of what one might think of as overall activity" (see [2, Chapter 6, p. 242]). So the notation of complexity in dynamic systems is still vague, for continuous dynamic systems as well as for discrete systems like CAs.

Since then, many alternative classification systems have been developed. They all attempt to classify the dynamic behavior and/or the

https://doi.org/10.25088/ComplexSystems.28.2.219

complexity of the 256 rules of the one-dimensional elementary (binary) cellular automaton (CA) rule set with a neighborhood of three bits. Regarding complexity, there are two well-known frameworks for algorithmic complexity. The first is the algorithmic information theory with the Kolmogorov complexity. It uses the uniform model of a Turing machine for computation. The second is the computational complexity theory that quantifies the amount of resources needed to solve an algorithmic problem, that is, time and storage (space) or the amount of communication, the number of inputs and gates used in circuit complexity or the number of processors. Computational complexity theory uses a nonuniform model of computation.

The framework based on algorithmic information theory led to attempts to classify CAs with the quantification of entropies (e.g., Shannon's block entropy) or data compression. But as the Kolmogorov complexity is known to be incomputable and unstable, only methods for approximations can be applied, making the algorithmic complexity K of the algorithmic information theory semi-computable. To approximate the Kolmogorov complexity, a data string of the output of an algorithm is analyzed to find the entropies included or the possibility to compress the output data. The known methods are, for example, the lossless compression method (see [3, 4]), the coding theorem method or the block decomposition method. These methods can be applied without knowing the underlying algorithm of the elementary cellular automaton (ECA) rule itself. That is an advantage, because the exact algorithms for a system of interest are not always known, but there is almost always a possibility to get some output data to analyze. The disadvantage of these methods is that the results (e.g., a compression ratio) only indirectly show that the complexity of the underlying algorithm has a high or a low complexity, whatever the length or the structure of that algorithm may be. Another problem is the instability that leads to slightly different results using different methods of compression and/or definitions of entropy.

But if the underlying algorithm or function of a system of interest is known, another way of analyzing its complexity is possible. For those systems, the methods of the computational complexity theory can be applied. The method described in this paper will give a measure that is based on the complexity of Boolean functions, which is closely related to the definition of circuit size complexity. It will use the equivalence of ECAs and Boolean functions to manipulate the truth table of the rules with Boolean algebra to find their complexity in bits.

An analytical complexity index based on the truth tables of the rule set was first suggested by Chua [5]. There the complexity of a CA rule N is equal to the minimum number of parallel planes that are necessary to separate the colored vertices of his Boolean cube representa-

tion of the lookup table of the rule. This results in four classes (0 to 3) ranging from zero complexity at 0 to the maximum index of 3 with the highest complexity. Chua also managed to transform all rules of the rule set into difference equations with a binary output. The four complexity classes are then reflected in the complexity of the underlying difference equation (e.g., the necessary number of brackets). There are many examples, in which the Chua index matches with the dynamic complexity of the spatiotemporal patterns of the CA. For example, $K = 0$ can be matched with a uniform state (rule 0), $K = 1$ shows repetitive patterns of low complexity (rule 15), $K = 2$ shows universal computation (rule 110) and $K = 3$ shows chaotic patterns of high complexity (rule 150). However, this does not hold for all of the 256 rules, so even rules with a Chua index of $K = 2$ or $K = 3$ can also exhibit simple behavior.

To overcome the limitations of the Chua index, this paper presents a new method to define the complexity of the underlying Boolean functions of the ECA rules. In theoretical computer science, circuit complexity is a branch of computational complexity theory in which Boolean functions are classified according to the size or depth of Boolean circuits that compute them. One speaks of the circuit complexity of a Boolean circuit. Any Boolean circuit with x input nodes realizes some Boolean function F. The circuit size complexity of a function F is then the minimum number of gates in any realization of F. The minimization of the gates is motivated by building a circuit as effectively and cheaply as possible in hardware. In this paper, the circuit complexity itself is not an appropriate measure, but the techniques of Boolean algebra, which normally reduce the size of a circuit, will be used and modified to define a new complexity measure for ECAs.

It is known that every Boolean function can be reduced by applying Boolean algebra or, for example, with the help of a Karnaugh map. Boolean functions and their complexity have been investigated for a long time, at least since Shannon's 1949 paper [6]. But it seems that the results have not yet been fully exploited to classify the complexity of discrete (or continuous) dynamic systems. This paper tries to fill that gap. The method that will be presented—in addition to the work of Chua—not only uses minimization techniques but also takes into account that in a dynamic system such as an ECA, the complexity is not always constant in time.

2. Elementary Cellular Automata

The simplest ECA rule space with the capability of universal computation is one dimensional, binary and follows the nearest-neighbor rules ($r = 1,\ k = 3$) with three coupled bits as variables. Cook and

https://doi.org/10.25088/ComplexSystems.28.2.219

Wolfram proved that rule 110 is Turing complete, that is, a universal computer.

Despite the simplicity of this ECA rule space, an ECA is far from being a trivial model for a dynamical system. A one-dimensional string with a number of n cells (e.g., with periodic boundaries) makes a transition at each time step according to a set of n locally coupled groups of three cells, which act as variables. Therefore we have to deal globally with an n-dimensional spatiotemporal complexity where collective phenomena occur, like in coupled oscillators, lasers or networks. Locally the state space of the lookup table of each rule is three dimensional (three bits as state variables A, B, C) with input intersections for each bit of output (see Section 2.2).

For example, in laser physics the complex dynamic behavior is described by the model of "mode competition." The modes of a laser resonator all experience optical amplification in the same gain medium, for example, a laser crystal, in which they spatially overlap to a significant extent. This leads to the phenomenon of mode competition or gain competition. So different modes experience amplification in the same gain medium, and this leads to cross-saturation effects, where stimulated emission by one mode causes gain saturation not only for itself, but also for the other modes. This also leads to the phenomenon that the power distribution over several modes is unstable. According to ECA rules, there is the one-dimensional binary string as the global medium, which is populated by the locally coupled $k = 3$ bits neighborhoods. These k-blocks are then exposed to competition.

Another related model is the coupled map lattice that has nontrivial properties like space-time mixing. It is also important to distinguish between conservative and dissipative behavior in ECA rules.

2.1 The Wolfram Code

Since there are $2\times2\times2 = 23 = 8$ possible binary states for the three cells neighboring a given cell, there are a total of $28 = 256$ ECA rules, each of which can be indexed with an eight-bit binary number, known as the Wolfram code (see Figure 1).

The elementary algorithm for the transition of a string from time step t to $t+1$ consists of a set of eight groups of three coupled cells (A, B, C) that define the state of one next cell. The three cells can be interpreted as three coupled variables in a dynamic system. Variables that can only have two states are indeed the minimal version of a variable, because with only one state it would be a constant. So these three variables define a three-dimensional state space, which should not be confused with the "physical" dimensionality of one-dimensional ECAs, describing the number of dimensions of the space where

the automaton takes place. To analyze dynamic systems it is normal to use the notation of the state space, or the phase space for continuous systems.

In ECAs, this discrete three-dimensional state space is very compact and therefore consists only of 3 bits $* 8 = 24$ bits for each rule.

The algorithm shown in Figure 1 that is determining the output of the eight groups of cells is often called the "lookup table."

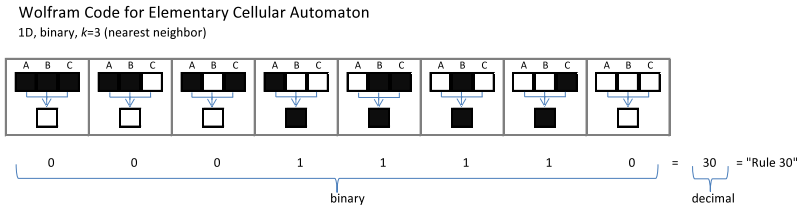


Figure 1. The Wolfram code, indexing the lookup table (or rule table, or truth table).

2.2 The Self-Replicating String

The ECA can also be interpreted as a model of an abstract self-replication. Not as a universal constructor in the sense of von Neumann, but as an automaton that is constituted by groups of three cells and permanently produces those groups of three cells at each time step.

The shortest possible string for the automaton would be a string with three bits (with periodic boundary conditions) as an input string leading to a three-bit output. But then not only one, but all three bits would be completely self-referential. So the minimum string length with the normal one output bit referencing to the first group of three bits and then two bits with overlapping groups is five bits long. For clarity, only one group of three bits is shown in Figure 2. The complete string, of course, would consist of five bits, with five groups of three bits that intersect each other. So each ECA of a string length of n cells with periodic boundary conditions is an automaton that basically takes n groups of three bits and transforms them into new n groups of three bits. After a transformation, the number of the eight different possible groups of three bits can be changed or can stay the same. This fact leads to the important concept of the input entropy or k-block variance coming up next.

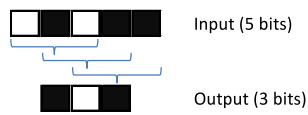


Figure 2. Self-replication of a three-bit group.

https://doi.org/10.25088/ComplexSystems.28.2.219

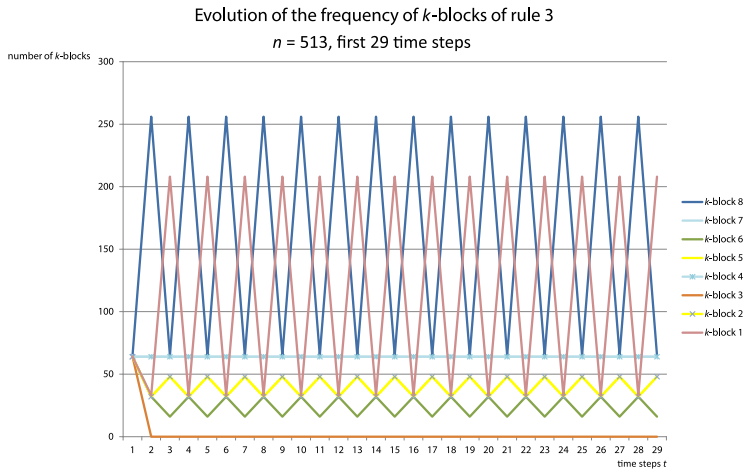
Finally, it should be said that ECAs are very sensitive to the length of the number of cells used for the one-dimensional bit string. Testing with random initial values showed that at least 137 bits are needed to show the typical behavior for each rule.

2.3 From the Input Entropy to the *k*-Block Variance

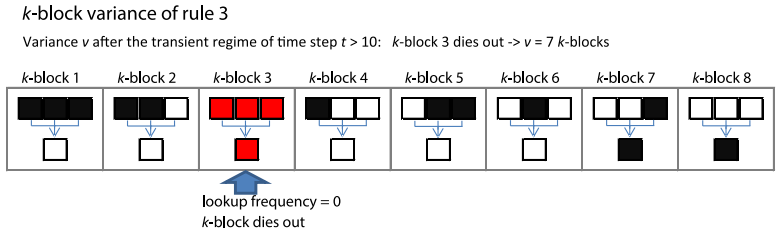
If a string of 137 bit values is used to test the behavior of an ECA rule, the evolution of 137 groups of three bits can be studied. With random initial states for each cell at the beginning ($t = 0$), the number of lookups of the eight k-blocks will be statistically $137/8 \approx 17$ per k-block for the first time step. But as the automaton evolves, the number of lookups per k-block can change and so the frequencies of each k-block must be considered.

The concept of input entropy was introduced by Wuensche in [7] to classify the behavior of ECAs. He traced the number of all eight k-blocks for each time step in a histogram he called the "lookup frequency." Then he defined the Shannon entropy of this frequency distribution as the input entropy. In highly ordered dynamics, he found that after a transient period some k-blocks are never looked at (so their lookup frequency is 0). This means that some k-blocks will die out for certain rules, just like the modes in laser physics.

For the calculation of the complexity in this paper, the information about what k-blocks are actually used in a certain rule is vital. So it is useful to number the k-blocks and to analyze which one dies out and which one survives the "k-block competition." Figure 3 shows the example of rule 3, where one k-block dies out after the transient regime of N time steps. This new use of Wuensche's lookup frequency is called the k-block variance of a rule.



(a)



(b)

Figure 3. (a) Frequency evolution of the k-blocks of rule 3. Length of bit string is 513 bits; sum of k-blocks is 513 at each time step. (b) k-block variance of rule 3.

Regarding the definition of complexity used in this paper, the analysis of which k-blocks are actually looked up for a given time step can shrink the content of information of the k-block group from the initial 24 bits at a level of three bits per k-block. In this case it would be 24 bits minus 3 bits, resulting in 21 bits.

2.4 The Information Loss of a Rule: Conservative and Dissipative Behavior of Elementary Cellular Automata

The possibility of a rule losing information due to the k-blocks that die out after N time steps (see Figure 3(b)) will be essential for the next section. This important phenomenon will be called "information loss" in this paper. Together with the minimization method based on Boolean algebra and presented later on, the complexity of a rule as a minimized Boolean function can be defined.

But regarding dynamical systems, another important analogy can be spotted. In dynamic systems theory, there is a fundamental difference between conservative (Hamiltonian systems) and dissipative systems. Conservative systems do not lose energy, and so the volume of the phase space is preserved. Conservative systems also do not have attractors. In contrast to that, dissipative systems do lose energy, and so the volume of the phase space shrinks. The dynamic of such a system then evolves to an attractor state.

In ECAs, the eight k-blocks are defining the 24-bit state space. Every loss of a k-block is an information loss, and so it shrinks the available state space for a certain rule. With respect to dynamic systems, this could be called "information dissipation." A rule that loses no k-blocks during its evolution will therefore be called "information conservative."

https://doi.org/10.25088/ComplexSystems.28.2.219

3. Defining a Measure for the Complexity of Elementary Cellular Automata as Minimized Boolean Functions

In this section, the explicit method of defining and calculating the complexity $C(r)$ of an ECA rule as a Boolean function is presented.

To calculate this kind of complexity of an ECA rule, three steps are necessary. These steps will be described in detail next. The central idea is the observation that the 24-bit state space of the lookup table can be minimized (or "fractalized") for each rule and each time step.

Because of the evolution of each rule in time, the calculation of the complexity has to be done at the first time step ($t = 1$) and at a time step when the resulting attractor regime is reached ($t \gg 100$). If the complexity does not change over time, it will be called an information-conservative rule and if the value shrinks, it will be called an information-dissipative rule.

A fractal state space (or phase space) is a well-known phenomenon in complex dynamical systems. But the usage in ECAs for describing their complexity is new.

The complexity $C(r)$ of a rule r as a Boolean function is given by the minimum number of input bits I_{bit} in each rule. The upper bound for an ECA rule is therefore given by 24 bits and the lower bounds have to be calculated for every rule:

$$C(r) = \min(i_{\text{bit}}(r)). \tag{1}$$

To do this, the following three steps are carried out.

3.1 Minimization of the Truth Table at $t = 1$

To find the minimum input bits of the truth table as a representation form of a Boolean function, a rule as a function of three variables $F(A, B, C) = r$ is transformed into the canonical forms of the sum of products (SoP, also known as minterm expansion) and the product of sums (PoS, also known as the maxterm expansion). These two canonical forms can then be minimized by Boolean algebra.

Figure 4 shows the example of the truth table of rule 234 and its canonical form of SoP and PoS on the left, and the minimized form on the right. This minimization can be done with Boolean algebra or, for example, with the help of a Karnough map. The superfluous bits are then grayed out. The complexity of the input is therefore reduced from 24 bits to seven bits without losing information about the correct output.

In circuit theory, this minimal form is called the circuit complexity, because it leads to a circuit in hardware with the least numbers of inputs and gates. Normally the minimal form of minterms or the minimal form of maxterms is chosen to build a circuit. But in this paper,

the focus is not on building circuits but on the minimal version of the truth table, so min- and maxterms are both used and transformed back into the corresponding minimized truth table shown on the right of Figure 4.

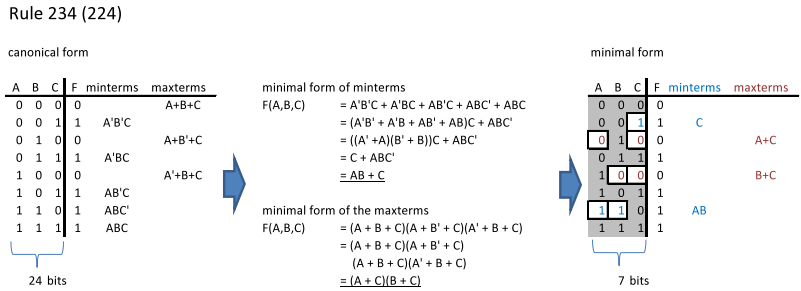


Figure 4. Detecting irrelevant data from the input variables of the truth table (grayed out in right table).

And of course this form can then be transformed into a fractalized ECA rule table as shown in Figure 5.

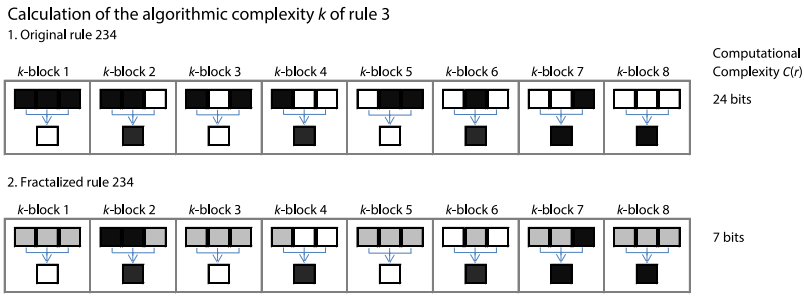


Figure 5. Minimization of rule 234, short term, in the form of a rule table (or lookup table).

3.2 Information Loss: Detecting the Long-Term *k*-Block Variance for Dynamic Minimization

In the second step, the long-term evolution of a rule and its information loss is considered. This information loss is acting like a damping factor in a dissipative dynamic system.

To do this, the initial conditions for all rules should be chosen as a random bit string at the beginning ($t = 0$) with a length of at least 137 bits and periodic boundary conditions. Then in rule 234, all k-blocks except k-block eight die out in less than 100 time steps (Figure 6). So their input and output bits can be crossed out. The

input and output patterns that do not occur anymore are called a "don't-care term" in digital logic. It is an input sequence for which the function output does not matter. In logic design, the don't-care conditions are important to consider in minimizing logic circuit design, using Karnaugh maps and the Quine–McCluskey algorithm. Don't-care optimizations can also be used for size-optimized assembly or machine code. Especially in a Karnough map, these conditions can then be exploited for further minimization.

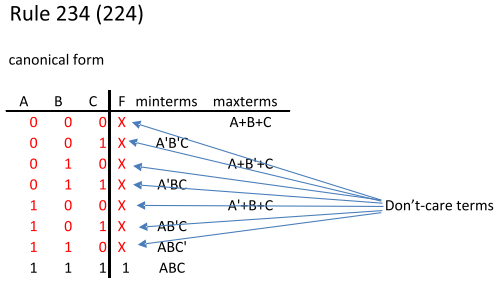


Figure 6. Loss of seven k-blocks at $t \gg 100$ in rule 234 that satisfy the don't-care condition.

3.3 Minimization of the Truth Table at $t \gg 100$ Including Information Loss

The third step reflects the information loss and also uses the don't-care terms for minimization (Figure 7).

So finally, the long-term computational complexity is 0 bits. In this case, the formula for the Boolean function does not even have any input variables any more. The output is always the constant bit 1.

The fractalized long-term rule table of the ECA now looks as shown in Figure 8.

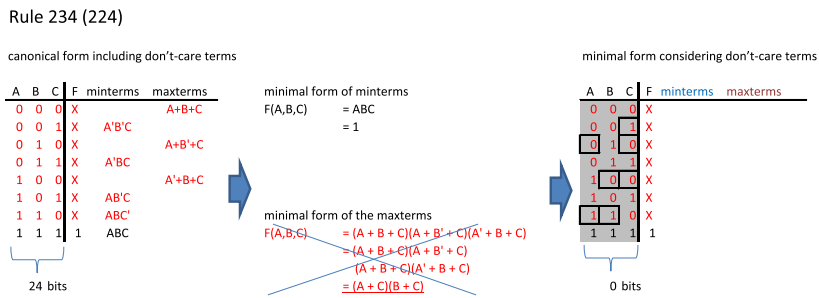


Figure 7. Removing the bits from the don't-care terms for final minimization.

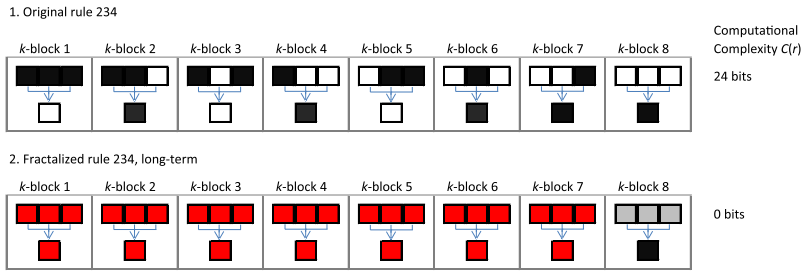


Figure 8. Long-term minimization of rule 234 in the form of a rule table, lookup table.

3.4 An Example of the Complexity of a Certain Rule in Short

As the main focus is on the complexity of the rule tables of ECAs, Figure 9 shows a short version of the complexity of rule 3 (see example of Figure 3(b)) containing only the results in the form of the rule tables.

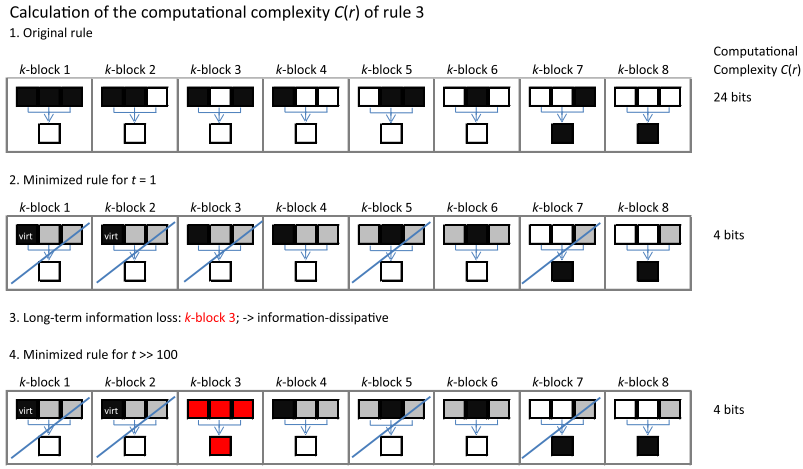


Figure 9. Complexity $C(r)$ of rule 3.

For the complete list of the determination of the complexity of all 88 rules, see Appendix A.

3.5 Proof

To prove the fact that the compressed forms of the rules for the short-term behavior ($t = 1$) and for the long-term behavior ($t \gg 100$) are indeed a lossless representation of the original algorithm, simply apply the reduced lookup tables to a string at $t = 1$ and $t \gg 100$ and

check if the output of the given rule 3 (or any other rule) is exactly the same, using only the reduced amount of bits of information.

4. Clustering of the Rule Space and Its Selected 88 Independent Rules and Their Properties

4.1 The 34 Clusters and the Properties of the Rules

Because of symmetries in the rule space of the ECAs, only 88 fundamentally different rules (out of 256) do exist. These 88 rules are analyzed in detail, and are clustered in 34 groups of attractors. Figure 10 shows the four classes and the groups that can be derived from the complexity of the rules. Figure 10 mainly shows the clusters. This means that the rules in every column evolve to exactly the same minimized truth table. Figure 11 then assigns the properties of each rule.

4.1.1 The Four Classes of Elementary Cellular Automata (First Column)

The classification into four classes is analytically based on the dimensionality of the minimized state space (input of the truth table) of each rule. The calculation of the complexity of the rules shows that the usage of the k-block bits can be separated into rules that trivially use 0 bits of information of the input string (class 0). Then there are rules that only use one variable of the k-blocks (class 1). The third class uses a maximum of two variables (class 2). The fourth and most complex class uses all three variables at least in one k-block (class 3).

The less complex rules of class 3 only use fractals of each k-block, but in sum at least one bit of the variables A, B and C is used. The two most complex rules of class 3, rules 105 and 150, use all 24 bits of the three-dimensional state space and are therefore irreducible regarding their truth table. On the other extreme, the class 0 rules are the simplest. They show no complexity at all. Then class 1 only exhibits trivial repetitive behavior of linear dynamic systems. Class 2 shows periodic behavior as well as low-dimensional chaos (fully developed two-dimensional chaos) in rules 195 and 165, and therefore can be called the "threshold of complexity." Finally, class 3 shows periodic behavior but is also exclusively capable of periodic orbits > 2. The chaos rules ($orbit = \infty$) develop from fractal chaos (with small periodic windows) to the fully developed three-dimensional chaos of rules 105 and 150. In comparison to the two-dimensional chaos, three-dimensional chaos is the high-dimensional chaos of ECAs. The alternation in class 3 between ordered, self-organized patterns and chaos is a typical behavior in spatiotemporal extended high-dimensional complex dynamic systems. See [8, 9].

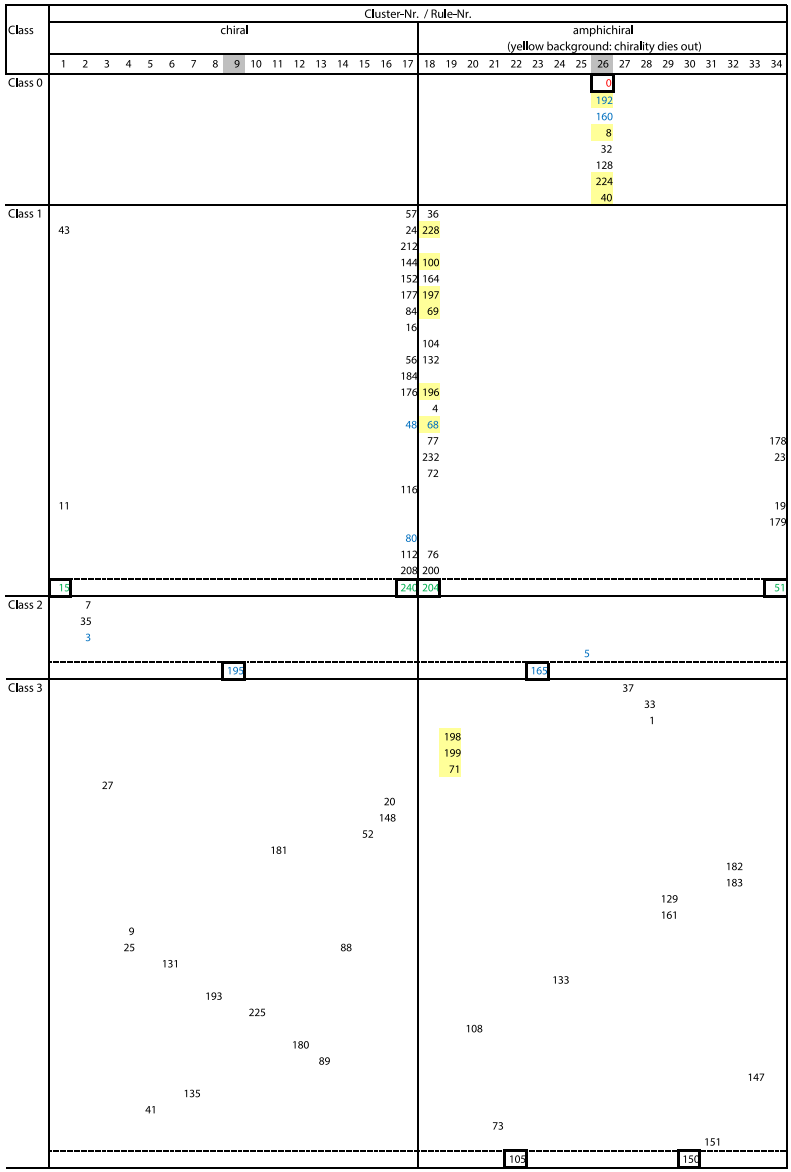


Figure 10. The 34 attractor clusters of the 88 independent rules, long-term behavior.

https://doi.org/10.25088/ComplexSystems.28.2.219

Class	DIM 1 D_{ASS}	DIM 2 D_{ASS2}	Information loss	C(r) (bits) t = 1	C(r) (bits) t >> 100	Periodicity Nr. of Orbits	Rule Number			
Class 0	0	0	diss 7	0	0	0	0			
	0	0	diss 7	4	0	0	192			
	0	0	diss 7	4	0	0	160			
	0	0	diss 7	6	0	0	8			
	0	0	diss 7	6	0	0	32			
	0	0	diss 7	6	0	0	128			
	0	0	diss 7	7	0	0	224			
	0	0	diss 7	11	0	0	40			
Class 1	1	0,33	diss 4	14	2	1 or 2	57	36		
	1	0,33	diss 4	12	2	1 or 2	43	24	228	
	1	0,33	diss 4	12	2	1 or 2	212			
	1	0,33	diss 4	11	2	1 or 2	144	100		
	1	0,33	diss 4	11	2	1 or 2	152	164		
	1	0,33	diss 4	8	2	1 or 2	177	197		
	1	0,33	diss 4	7	2	1 or 2	84	69		
	1	0,33	diss 4	6	2	1 or 2	16			
	1	0,4	diss 3	18	2	1 or 2		104		
	1	0,4	diss 3	11	2	1 or 2	56	132		
	1	0,4	diss 3	8	2	1 or 2	184			
	1	0,4	diss 3	7	2	1 or 2	176	196		
	1	0,4	diss 3	6	2	1 or 2		4		
	1	0,4	diss 3	4	2	1 or 2	48	68		
	1	0,5	diss 2	12	2	1 or 2	77	178		
	1	0,5	diss 2	12	2	1 or 2	232	23		
	1	0,5	diss 2	11	2	1 or 2	72			
	1	0,5	diss 2	8	2	1 or 2	116			
	1	0,5	diss 2	7	2	1 or 2	11	19		
	1	0,5	diss 2	7	2	1 or 2		179		
	1	0,5	diss 2	4	2	1 or 2	80			
	1	0,67	diss 1	7	2	1 or 2	112	76		
	1	0,67	diss 1	7	2	1 or 2	208	200		
	1	1	cons	2	2	1 or 2	15	240	204	51
Class 2	1,33	0,8	diss 2	7	4	1 and 2	7			
	1,33	1	diss 1	7	4	1 and 2	35			
	1,33	1	diss 1	4	4	1 and 2	3			
	1,33	1,33	cons	4	4	1 and 2		5		
	2	2	cons	8	8	∞	195	165		
Class 3	1,25	0,83	diss 2	11	5	1 and 2		37		
	1,50	1,20	diss 1	11	6	1 and 2		33		
	1,50	1,20	diss 1	6	6	1 and 2		1		
	2	1,33	diss 2	14	8	1 and 2		198		
	2	1,33	diss 2	11	8	1 and 2		199		
	2	2	cons	8	8	1 and 2		71		
	2	2	cons	8	8	1 and 2	27			
	2	1,43	diss 2	11	10	1 and 2	20			
	2	1,43	diss 2	11	10	1 and 2	148			
	2,20	1,83	diss 1	11	11	1 and 2	52			
	2,20	1,83	diss 1	11	11	16	181			
	2,20	1,83	diss 1	18	11	∞		182		
	2,20	1,83	diss 1	11	11	∞		183		
	2,20	1,83	diss 1	12	11	∞		129		
	2,20	1,83	diss 1	11	11	∞		161		
	2,20	2,20	cons	11	11	1 and 2	9			
	2,20	2,20	cons	11	11	1 and 2	25	88		
	2,20	2,20	cons	11	11	3	131			
	2,20	2,20	cons	11	11	6		133		
	2,20	2,20	cons	11	11	105	193			
	2,33	2,33	cons	14	14	∞	225			
	2,33	2,33	cons	14	14	1 and 2		108		
	2,33	2,33	cons	14	14	8	180			
	2,33	2,33	cons	14	14	∞	89			
	2,33	2,33	cons	14	14	4 ; (70, 200)		147		
	2,33	2,33	cons	14	14	∞	135			
	2,57	2,57	cons	18	18	8		41		
	2,57	2,57	cons	18	18	>>100 (2000)	73			
	2,57	2,57	cons	18	18	∞		151		
	3	3	cons	24	24	∞	105	150		

Figure 11. Short form of Figure 10, not showing the clusters but the properties of the rules.

4.1.2 The Fractal Dimension of the State Space (Second and Third Columns)

Besides the classification in four classes, the fractal dimension D of the algorithmic state space ("ASS") is defined as:

$$D_{\mathrm{ASS}} = C(r) \,/\, n_{\mathrm{KBC}}, \tag{2}$$

with $C(r)$ as the complexity of the minimized Boolean function in bits divided by n_{KBC} as the number of remaining k-blocks of the compressed lookup table. k-blocks that die out are not counted:

$$D_{\mathrm{ASS2}} = C(r) \,/\, (n_{\mathrm{KBC}} + n_{\mathrm{KBD}}). \tag{3}$$

In addition to equation (2), n_{KBD} is the number of k-blocks that die out. So the dimensionality index is reduced and takes into account that the die outs make the output simpler.

4.1.3 The Information Loss of a Rule (Fourth Column)

There are 26 information-conservative (cons.) and 62 information-dissipative (diss.) rules, as shown in Figure 11. The number behind the dissipative marker shows the number of k-blocks that die out.

4.1.4 The Complexity $C(r)$ in Bits (Fifth and Sixth Columns)

This column of Figure 11 gives the complexity $C(r)$ of the lookup table in bits (as defined in equation (3)), first for the short-term behavior ($t = 1$) and then for the long-term behavior ($t \gg 100$).

4.1.5 The Periodicity of the Rules (Seventh Column)

Regarding the periodicity of the stable and unstable orbits, Figure 11 shows:

Stable periodic orbit 0:	8 rules
Stable periodic orbit 1:	34 rules
Stable periodic orbit 2:	8 rules
Stable periodic orbits 1 and 2:	18 rules
Stable periodic orbits > 2 and $< \infty$:	8 rules
Unstable periodic orbits $= \infty$:	12 rules (10 rules high-dim., 2 rules low-dim. chaos)

As in other complex dynamical systems, the unstable periodic orbits (also known as UPOs) are forming the skeleton of the chaotic rules.

4.1.6 Nested Elementary Cellular Automata

Note that the ECA as an example for a CA with the neighborhood $k = 3$ contains the simpler CAs with $k = 0$, $k = 1$ and $k = 2$. In Figure 10, these 14 nested rules are colored red for $k = 0$, green for $k = 1$ and blue for $k = 2$. In detail, these 14 rules consist of nine blue rules of exclusively $k = 2$, plus the nested four green rules plus the

https://doi.org/10.25088/ComplexSystems.28.2.219

nested one red rule, which gives the sum of 14 rules of $k = 2$. Analogously, $k = 1$ (green plus red) are five rules and $k = 0$ is only one rule. As 14 rules are an uncommon number for $k = 2$, it should be mentioned that the embedding consists of three rule sets of 16 rules each (48 rules) and analogously to the reduction of the 256 rules of $k = 3$, the removing of symmetric and superfluous rules leads to only 14 embedded rules.

The remaining 74 black rules all belong exclusively to the $k = 3$ neighborhood and cannot be found in simpler CAs. It is interesting to see that most of them are forced back into class 2, class 1 or even class 0 behaviors via attractors. Only the color-marked 18 out of the 88 rules of class 3 also exhibit complex class 3 behavior; that means periodic orbits > 2, or high-dimensional chaos.

4.2 Comparisons of the Classes of Figure 10 with Other Indexes and Methods

4.2.1 Comparison with the Chua Index

In [5] Chua gives a complexity index k for each rule.

The original index classification of Chua only consists of three classes because classes 0 and 1 were summarized. With respect to the long-term behaviors, it seems to be better to define an additional index of 0, because there are zero separation planes for the Boolean cube for rules 0 and 255 according to the definition of Chua. So the modification starts with four classes and reassigns the rules that can be simplified in their long-term behavior. For example, rule 8 equals rule 0 after a sufficient amount of time steps and therefore is migrated from class 1 to class 0. Rule 24 equals rule 240 migrated to class 1 and rule 27 equals rule 25 from class 2.

The resulting structure of the classification shown in Figure 12 matches the suggested classification of this paper very well. As the index classification is an integer, the fractal separation of the classes is a little vague. So class 2 could also mean: greater than class 1 and smaller than class 3, like in the classification of Figure 10.

4.2.2 Comparison with the Lossless Compression Method

In comparison with the results of the lossless compression (NC) of the output strings in [3], there is a relatively good match in terms of classification. For example, the compression rate of rule 105 is 1.0 (i.e., full incompressibility) and therefore an indicator for high complexity. The rate of rule 128 is 0.14 and represents a group of eight rules with the highest compression rate (lowest complexity). Rule 4 has a rate of 0.26 and is slightly more complex, and rule 25 with a rate of 0.64 shows an intermediate complexity. For a complete comparison, see Figure 13.

	Modification of the Index Classification		
Complexity Index k	Original Index Classification by Chua Short-term Behavior	Modified Class	Modified Chua Classification Long-term Behavior
Index 1	0	Class 0	0 8 32 128 224 192 160 40
Index 1	8 32 128 224 192 160 1 16 4 3 5 48 68 80 7 11 69 84 19 35 112 179 76 208 196 176 200 15 23 43 51 77 240 212 178 204 232	Class 1	1 16 4 3 5 48 68 80 7 11 69 84 19 35 112 179 76 208 196 176 200 15 23 43 51 77 240 212 178 204 232 24 36 72 144 132 100 56 104 152 57 164 116 177 197 228 184
Index 2	40 20 9 24 33 36 72 144 132 199 25 37 52 100 56 88 104 148 152 57 108 198 164 131 133 73 180 181 41 193 147 225 89 182 183 129 161 151 135 195 165	Class 2 Class >1 and < 3	20 9 33 199 25 37 52 88 148 108 198 131 133 73 180 181 41 193 147 225 89 182 183 129 161 151 135 195 165 27 71
Index 3	27 71 116 177 197 228 184 105 150	Class 3	105 150

Figure 12. Original and modified Chua index classification.

Part 1

			green: 46	green: 34	green: 36
rule	equiv. Rule	norm. C(r)	NC	nor. BDM	nor. Bl. En.
150	150	1.000	0.98	0.367	0.821
105	105	1000	1.0	0.397	0.821
22	151	0.750	0.88	0.426	0.678
73	73	0.750	1.0	0.514	0.857
41	41	0.750	0.91	0.250	0.571
30	135	0.583	1.0	1.000	1.000
54	147	0.583	0.80	0.264	0.714
45	89	0.583	1.0	0.941	0.964
154	180	0.583	0.76	0.205	0.607
108	108	0.583	0.34	0.025	0.060
106	225	0.583	0.60	0.250	0.392
110	193	0.458	1.0	0.558	0.821
94	133	0.458	0.74	0.264	0.607
62	131	0.458	0.97	0.382	0.678
25	25	0.458	0.64	0.205	0.500
74	88	0.458	0.35	0.111	0.221
9	9	0.458	0.70	0.176	0.329
122	161	0.458	0.81	0.294	0.607
126	129	0.458	0.84	0.397	0.678
18	183	0.458	0.84	0.338	0.642
146	182	0.458	0.84	0.367	0.678
26	181	0.458	0.76	0.235	0.607
38	52	0.458	0.35	0.104	0.225
134	148	0.417	0.36	0.110	0.271
6	20	0.417	0.36	0.102	0.242
27	27	0.333	0.62	0.145	0.357
29	71	0.333	0.44	0.032	0.203
28	199	0.333	0.64	0.138	0.571
156	198	0.333	0.64	0.138	0.571
1	1	0.250	0.40	0.039	0.253
33	33	0.250	0.42	0.039	0.253
37	37	0.208	0.49	0.076	0.392
60	195	0.333	0.78	0.147	0.678
90	165	0.333	0.79	0.220	0.678
5	5	0.167	0.45	0.036	0.239
3	3	0.167	0.62	0.161	0.357
35	35	0.167	0.63	0.161	0.392
7	7	0.167	0.56	0.122	0.267
15	15	0.083	0.66	0.117	0.257
170	240	0.083	0.33	0.110	0.264
204	204	0.083	0.33	0.020	0.060
51	51	0.083	0.45	0.025	0.060
138	208	0.083	0.33	0.104	0.239
200	200	0.083	0.28	0.023	0.060

Part 2

			green: 46	green: 34	green: 36
rule	equiv. Rule	norm. C(r.)	NC	nor. BDM	nor. Bl. En.
42	112	0.083	0.33	0.107	0.253
76	76	0.083	0.33	0.023	0.060
10	80	0.083	0.33	0.100	0.225
50	179	0.083	0.70	0.220	0.535
19	19	0.083	0.40	0.027	0.032
11	11	0.083	0.66	0.138	0.310
46	116	0.083	0.34	0.105	0.264
72	72	0.083	0.24	0.025	0.060
232	232	0.083	0.28	0.023	0.046
23	23	0.083	0.40	0.027	0.046
178	178	0.083	0.70	0.220	0.535
77	77	0.083	0.81	0.235	0.535
12	68	0.083	0.33	0.026	0.046
34	48	0.083	0.33	0.110	0.214
4	4	0.083	0.26	0.025	0.032
140	196	0.083	0.33	0.026	0.046
162	176	0.083	0.33	0.114	0.246
184	184	0.083	0.34	0.117	0.325
56	56	0.083	0.34	0.116	0.325
132	132	0.083	0.29	0.025	0.046
104	104	0.083	0.21	0.025	0.050
2	16	0.083	0.33	0.098	0.164
14	84	0.083	0.34	0.111	0.289
13	69	0.083	0.71	0.161	0.500
58	177	0.083	0.55	0.191	0.535
78	197	0.083	0.62	0.132	0.535
152	152	0.083	0.34	0.097	0.267
164	164	0.083	0.26	0.025	0.032
130	144	0.083	0.33	0.102	0.189
44	100	0.083	0.33	0.026	0.060
142	212	0.083	0.34	0.116	0.296
43	43	0.083	0.67	0.142	0.332
24	24	0.083	0.34	0.102	0.271
172	228	0.083	0.33	0.027	0.060
57	57	0.083	0.87	0.352	0.571
36	36	0.083	0.22	0.023	0.032
40	40	0.000	0.15	0.023	0.0054
168	224	0.000	0.15	0.023	0.0054
128	128	0.000	0.14	0.023	0.0036
32	32	0.000	0.14	0.023	0.0036
8	8	0.000	0.14	0.023	0.0036
160	160	0.000	0.14	0.023	0.0036
136	192	0.000	0.14	0.023	0.0036
0	0	0.000	0.14	0.023	0.0036

Figure 13. Comparison norm. $C(r)$ with lossless compression method (NC), BDM and block entropy; green background indicating a good match and red background showing no match.

There are also differences between the complexities of certain rules. But that could also be an artifact of the method or the compression

algorithm, because the compression of the output data makes no direct reference to the underlying code.

4.2.3 Comparison with the Block Decomposition Method and the Block Entropy

The block decomposition method (BDM) also shows a relatively good match. For example, rule 110 exhibits an intermediate complexity, rule 2 a low complexity and rule 128 a very low complexity. But in comparison to the lossless compression method, the overall matching is a little bit inferior. The block entropy has the least agreement, but still has many matches, like rule 94 having intermediate complexity, rule 51 having low complexity and rule 128 having one of the lowest complexities. Figure 13 shows the complete comparison.

5. Conclusion

In this paper a new method is presented to calculate the complexity of cellular automata rules. Its analytical defined complexity $C(r)$ for each rule can directly be given by the number of relevant bits of the underlying input algorithm.

As a result, the former assumption that chaotic rules are the most complex rules can be confirmed for fully developed two- and three-dimensional chaos.

In spite of their simplicity, the 88 independent elementary cellular automata (ECAs) exhibit an amazing variety of complex behavior. It incorporates high- and low-dimensional chaos as well as information-dissipative and information-conservative behavior.

Another important result is the universal use for at least the class of cellular automata (CAs). The model can easily be applied to larger neighborhoods or/and more states, totalistic CAs or CAs in two or three space dimensions. So in the future, the steps to almost infinite complexity can be defined analytically.

Because of the fact that there are many tools to calculate the minimized forms of Boolean functions, the analysis of the complexity of Boolean functions with more than three variables can be computed automatically. But as the number of variables grows, the number of rules will become so big that this will soon be a challenging task for any supercomputer. Another challenge will be the transformation of the given definition of complexity from Boolean systems into discrete systems with continuous variables (modeled by difference equations) and continuous systems (modeled by differential equations or partial differential equations).

The primary goal of this paper was to make the term of complexity more precise for dynamic systems. The meaning of the word complex-

ity is often described as "interwoven" or "hard to describe." In the context of dynamic systems, the meaning of complexity was not (or still is not) clear at all. The given method to calculate an analytical form of complexity now can answer the questions concerning what makes a complex dynamic system complex and why a complex system is in fact harder to describe than a simple one. The big advantage in comparison to the complexity given in algorithmic information theory is a better understanding of where the complexity originally comes from. It is the algorithmic use of the three input variables as state variables in their possible combinations. The fewer bits used, the lower the complexity.

So the author hopes that this work makes a valuable contribution to the foundations of complexity science, namely the notation of complexity for dynamic systems itself.

Acknowledgments

The author wants to thank Hector Zenil, Andrew Wuensche and Leon Chua for many useful suggestions, corrections and inspirations.

References

[1] J. von Neumann, *Theory of Self-Reproducing Automata* (A. W. Burks, ed.), Urbana, IL: University of Illinois Press, 1966.

[2] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[3] H. Zenil and E. Villarreal-Zapata, "Asymptotic Behavior and Ratios of Complexity in Cellular Automata," *International Journal of Bifurcation and Chaos*, **23**(9), 2013 1350159. doi:10.1142/S0218127413501599.

[4] H. Zenil, "Compression-Based Investigation of the Dynamical Properties of Cellular Automata and Other Systems," *Complex Systems*, **19**(1), 2010 pp. 1–28. complex-systems.com/pdf/19-1-1.pdf.

[5] L. O. Chua, S. Yoon and R. Dogaru, "A Nonlinear Dynamics Perspective of Wolfram's New Kind of Science. Part I: Threshold of Complexity," *International Journal of Bifurcation and Chaos*, **12**(12), 2002 pp. 2655–2766. doi:10.1142/S0218127402006333.

[6] C. E. Shannon, "The Synthesis of Two-Terminal Switching Circuits," *The Bell System Technical Journal*, **28**(1), 1949 pp. 59–98. doi:10.1002/j.1538-7305.1949.tb03624.x.

[7] A. Wuensche, "Classifying Cellular Automata Automatically: Finding Gliders, Filtering, and Relating Space-Time Patterns, Attractor Basins,

and the Z Parameter," *Complexity*, **4**(3), 1999 pp. 47–66. doi:10.1002/(SICI)1099-0526(199901/02)4:3%3C47::AID-CPLX9%3E3.0.CO;2-V.

[8] R.-H. Du, S.-X. Qu and Y.-C. Lai, "Transition to High-Dimensional Chaos in Nonsmooth Dynamical Systems," *Physical Review E*, **98**(5), 2018 052212. doi:10.1103/PhysRevE.98.052212.

[9] T. Huang, X. Cong, H. Zhang, S. Ma and G. Pan, "Pattern Self-Organization and Pattern Transition on the Route to Chaos in a Spatiotemporal Discrete Predator-Prey System," *Advances in Difference Equations*, **2018**(1), 2018 175. doi:10.1186/s13662-018-1598-7.

Appendix

A. Computational Complexity $C(r)$ of All 88 Rules

The following figures show the complete list of all 88 qualitatively different rules of ECA. The form of presenting the evolution of the complexity from the original to the minimized rule (after many time steps) is equal to the form shown in Figure 9 as an example, and the rules are sorted by the 34 attractor clusters shown in Figure 10 in the main article. Therefore, the list starts with cluster 1 and rule 43 and ends with cluster 34 and rule 51.

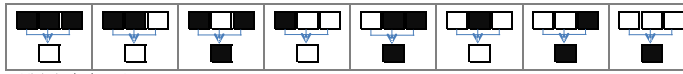
The rules with similar or equal long-term behaviors are grouped together. Furthermore, the first 39 rules visually show the asymmetry of the chiral rules and the last 49 rules the symmetry of the amphichiral ones, in their use of the fractal parts of the k-blocks.

In every attractor cluster the complexity rises monotonically, but the clusters themselves are not sorted by complexity. They represent the symmetric properties of a symmetry group of four sets of 88 rules that intersect each other and build the 256 ECA rules. The most complex rules therefore can be found at the end of clusters 22 and 30 (rule 105 and 150).

Appendix A is available at
wpmedia.wolfram.com/uploads/sites/13/2019/06/28-2-4-Appendix.pdf
and is included in full at
wpmedia.wolfram.com/uploads/sites/13/2019/06/28-2-4.pdf

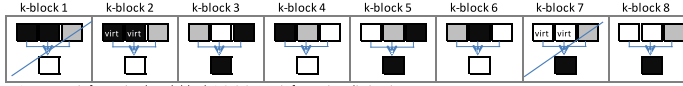
1. Rule 43

Original rule — Computational Complexity C(r)



24 bits

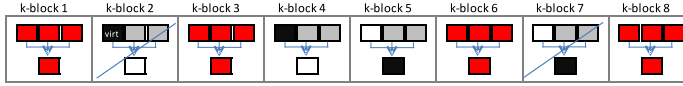
Minimized rule, t = 1



12 bits

Long-term information loss: k-block 1,3,6,8; -> information-dissipative

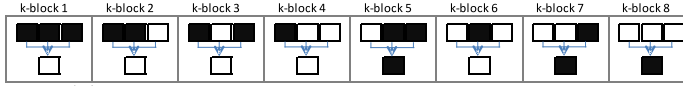
Minimized rule, t >> 100; including information loss



2 bits

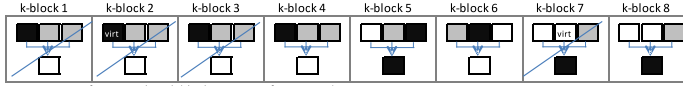
2. Rule 11

Original rule — Computational Complexity C(r)



24 bits

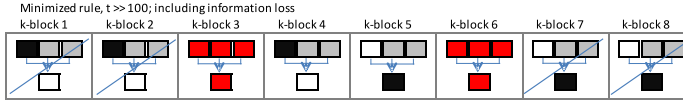
Minimized rule, t = 1



7 bits

Long-term information loss: k-block 3,6; -> information-dissipative

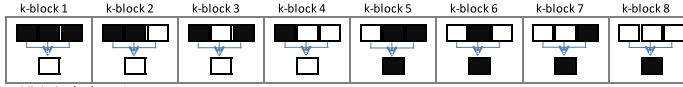
Minimized rule, t >> 100; including information loss



2 bits

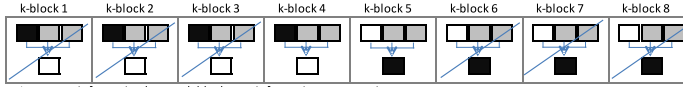
3. Rule 15

Original rule — Computational Complexity C(r)



24 bits

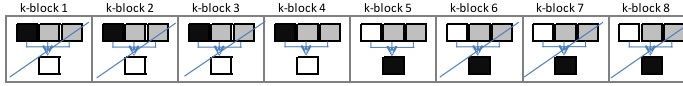
Minimized rule, t = 1



2 bits

Long-term information loss: no k-block; -> information-conservative

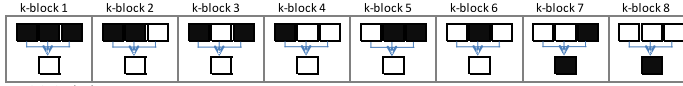
Minimized rule, t >> 100; including information loss



2 bits

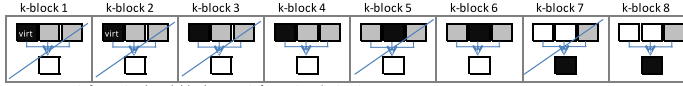
4. Rule 3

Original rule — Computational Complexity C(r)



24 bits

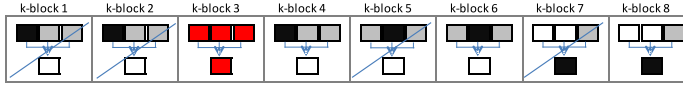
Minimized rule, t = 1



4 bits

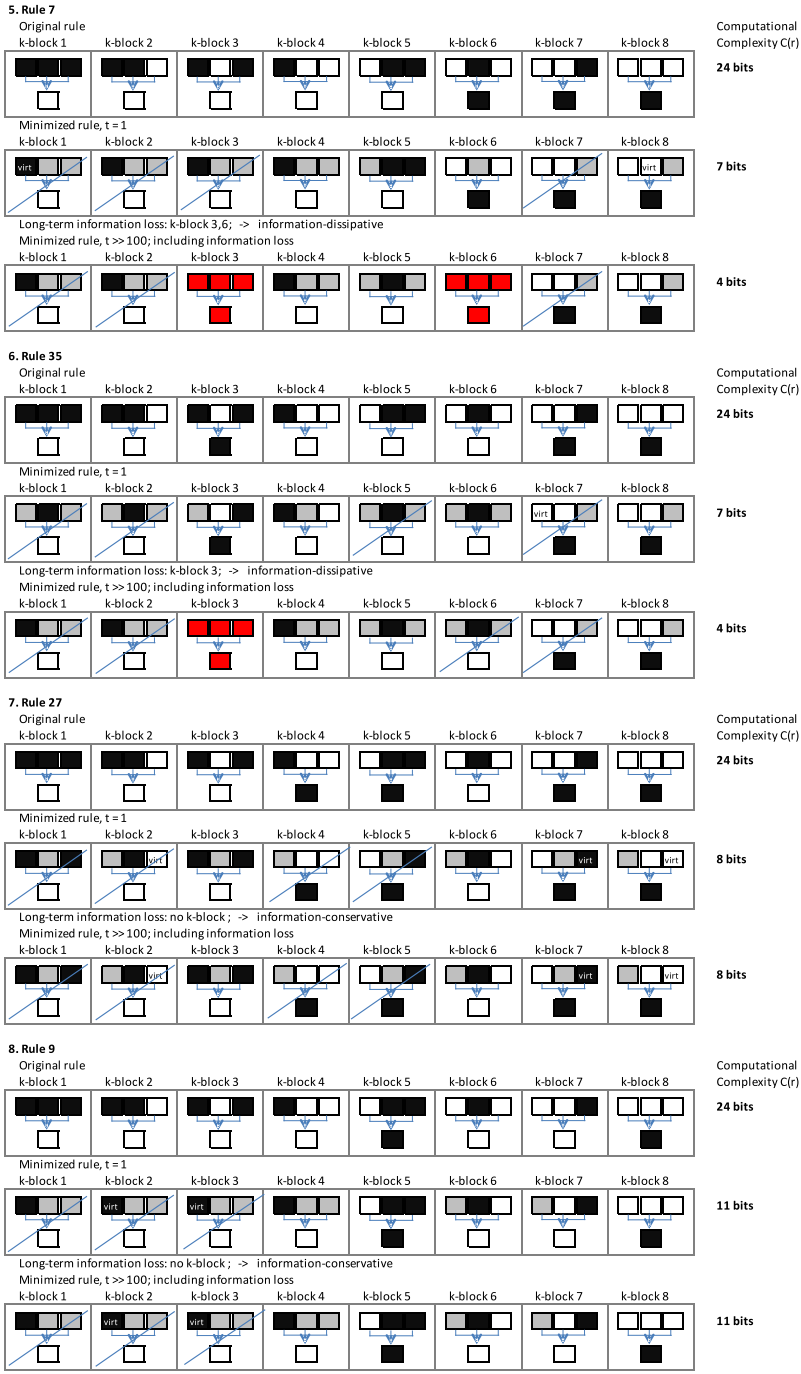
Long-term information loss: k-block 3; -> information-dissipative

Minimized rule, t >> 100; including information loss



4 bits

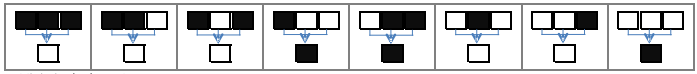
https://doi.org/10.25088/ComplexSystems.28.2.219


5. Rule 7
Original rule
k-block 1
k-block 2
k-block 3
k-block 4
k-block 5
k-block 6
k-block 7
k-block 8
Computational Complexity C(r)
24 bits
Minimized rule, t = 1
virt
7 bits
Long-term information loss: k-block 3,6; -> information-dissipative
Minimized rule, t >> 100; including information loss
4 bits
6. Rule 35
Original rule
24 bits
Minimized rule, t = 1
7 bits
Long-term information loss: k-block 3; -> information-dissipative
Minimized rule, t >> 100; including information loss
4 bits
7. Rule 27
Original rule
24 bits
Minimized rule, t = 1
8 bits
Long-term information loss: no k-block ; -> information-conservative
Minimized rule, t >> 100; including information loss
8 bits
8. Rule 9
Original rule
24 bits
Minimized rule, t = 1
11 bits
Long-term information loss: no k-block ; -> information-conservative
Minimized rule, t >> 100; including information loss
11 bits

9. Rule 25

Original rule — Computational Complexity C(r)

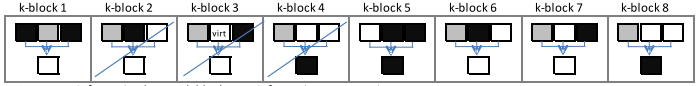
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

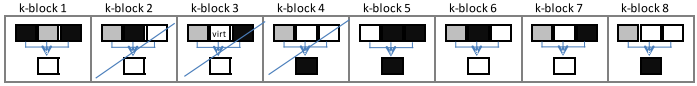


11 bits

Long-term information loss: no k-block ; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

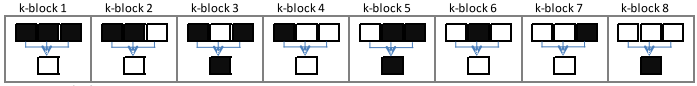


11 bits

10. Rule 41

Original rule — Computational Complexity C(r)

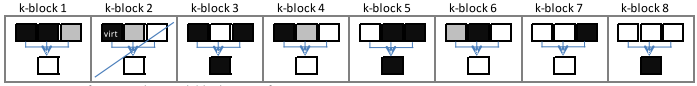
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

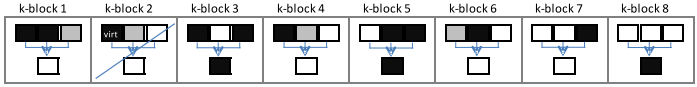


18 bits

Long-term information loss: no k-block ; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

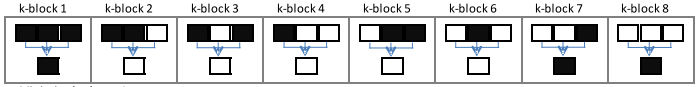


18 bits

11. Rule 131

Original rule — Computational Complexity C(r)

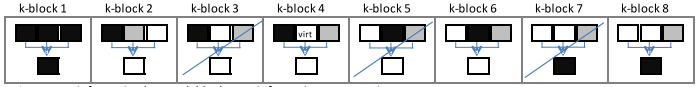
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

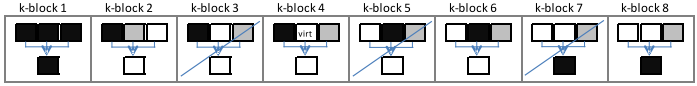


11 bits

Long-term information loss: no k-block ; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

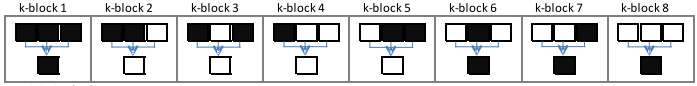


11 bits

12. Rule 135

Original rule — Computational Complexity C(r)

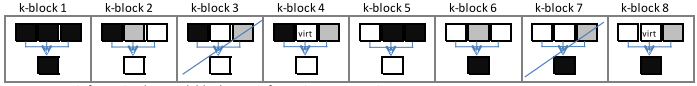
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

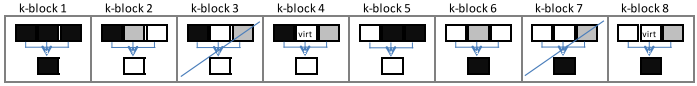


14 bits

Long-term information loss: no k-block ; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



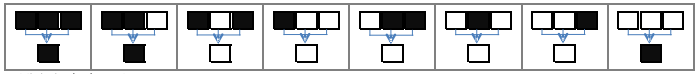
14 bits

https://doi.org/10.25088/ComplexSystems.28.2.219

13. Rule 193

Original rule — Computational Complexity C(r)

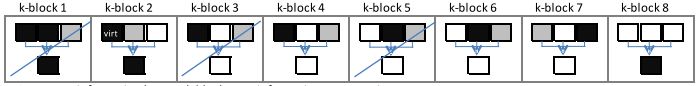
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

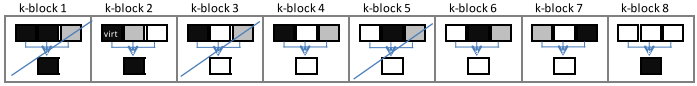


11 bits

Long-term information loss: no k-block ; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

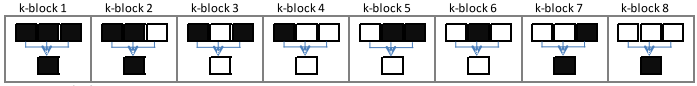


11 bits

14. Rule 195

Original rule — Computational Complexity C(r)

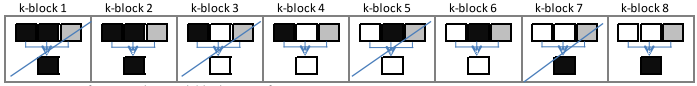
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

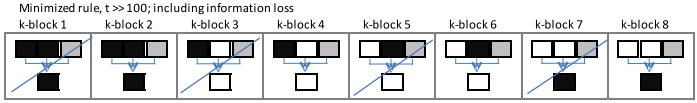


8 bits

Long-term information loss: no k-block ; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

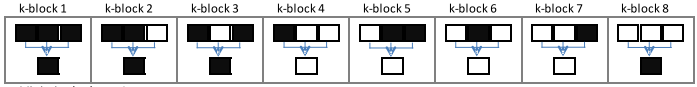


8 bits

15. Rule 225

Original rule — Computational Complexity C(r)

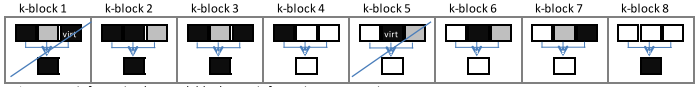
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

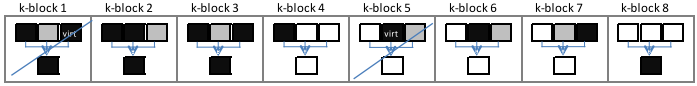


14 bits

Long-term information loss: no k-block ; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

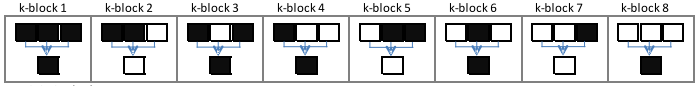


14 bits

16. Rule 181

Original rule — Computational Complexity C(r)

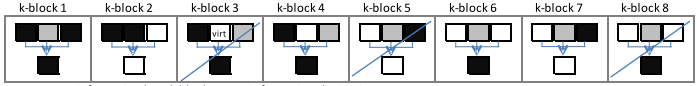
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

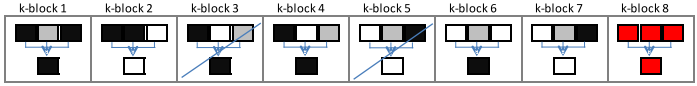


11 bits

Long-term information loss: k-block 1 ; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

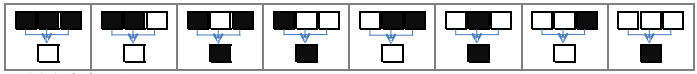


11 bits

17. Rule 180

Original rule — Computational Complexity C(r)

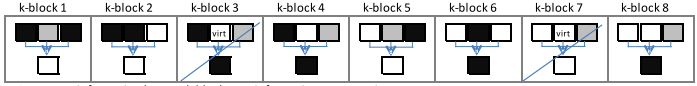
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

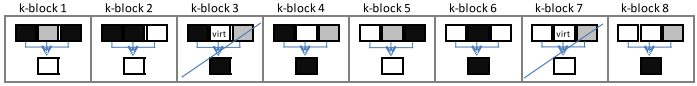


14 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

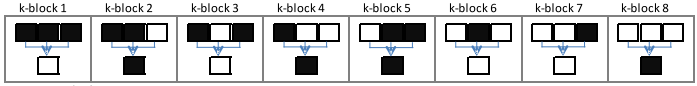


14 bits

18. Rule 89

Original rule — Computational Complexity C(r)

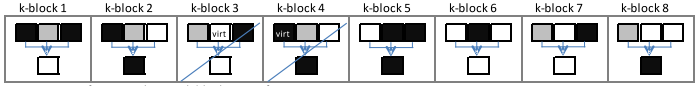
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

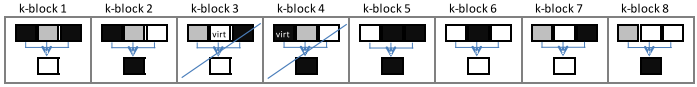


14 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

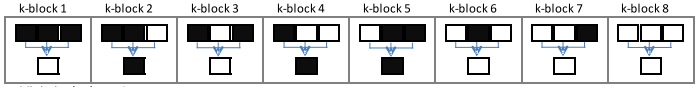


14 bits

19. Rule 88

Original rule — Computational Complexity C(r)

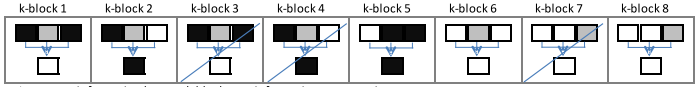
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

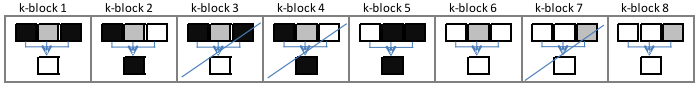


11 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

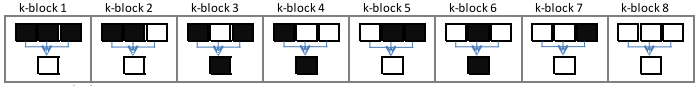


11 bits

20. Rule 52

Original rule — Computational Complexity C(r)

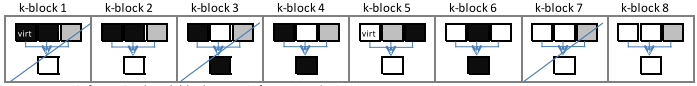
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

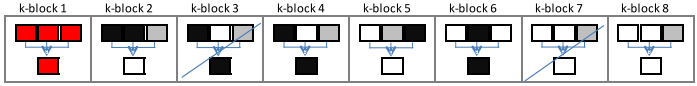


11 bits

Long-term information loss: k-block 1; -> information-dissipative

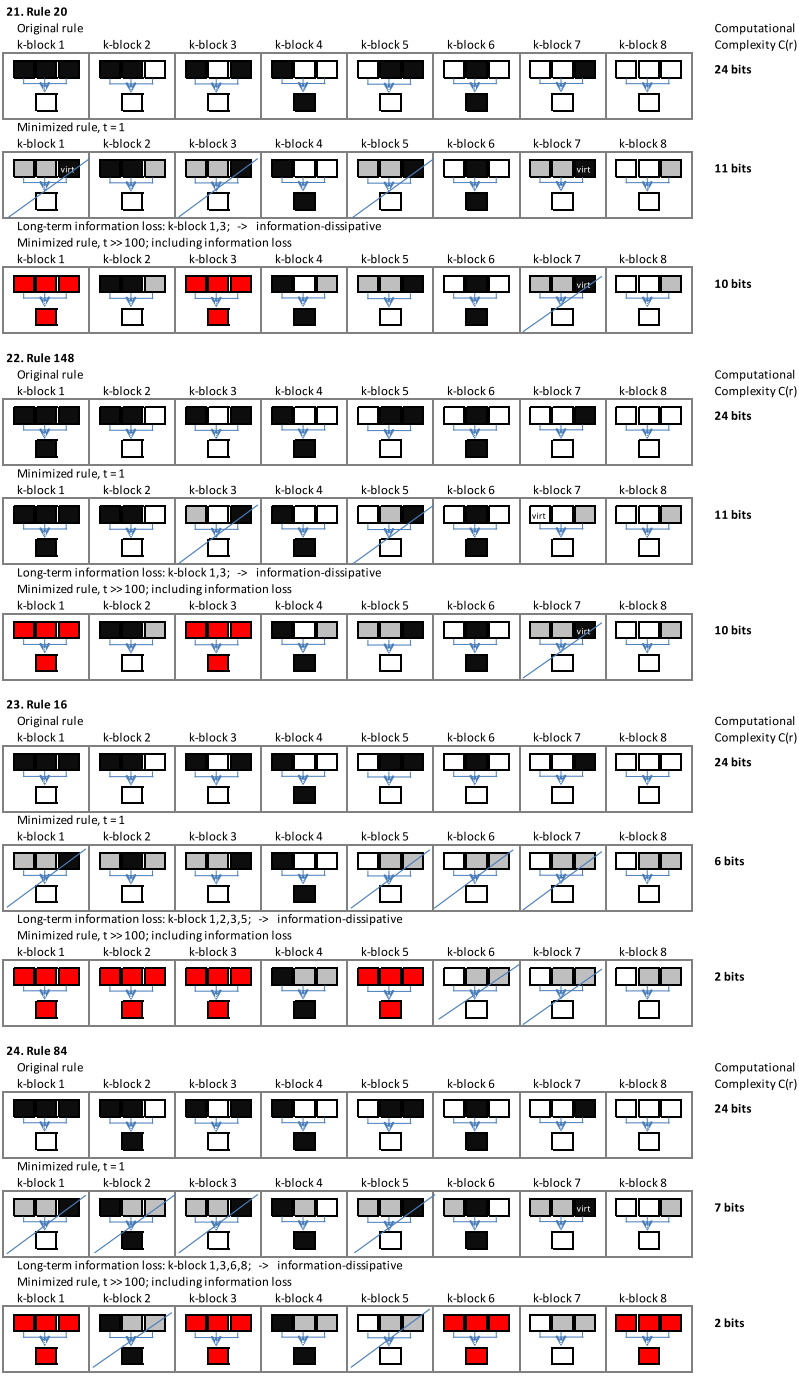
Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



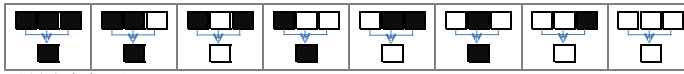
11 bits

https://doi.org/10.25088/ComplexSystems.28.2.219


21. Rule 20
Original rule
k-block 1
k-block 2
k-block 3
k-block 4
k-block 5
k-block 6
k-block 7
k-block 8
Computational Complexity C(r)
24 bits
Minimized rule, t = 1
11 bits
Long-term information loss: k-block 1,3; -> information-dissipative
Minimized rule, t >> 100; including information loss
10 bits
22. Rule 148
Original rule
24 bits
Minimized rule, t = 1
11 bits
Long-term information loss: k-block 1,3; -> information-dissipative
Minimized rule, t >> 100; including information loss
10 bits
23. Rule 16
Original rule
24 bits
Minimized rule, t = 1
6 bits
Long-term information loss: k-block 1,2,3,5; -> information-dissipative
Minimized rule, t >> 100; including information loss
2 bits
24. Rule 84
Original rule
24 bits
Minimized rule, t = 1
7 bits
Long-term information loss: k-block 1,3,6,8; -> information-dissipative
Minimized rule, t >> 100; including information loss
2 bits

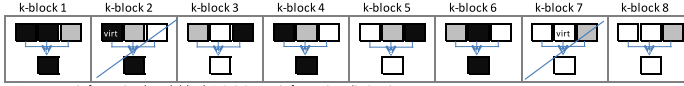
25. Rule 212

Original rule — Computational Complexity C(r)



24 bits

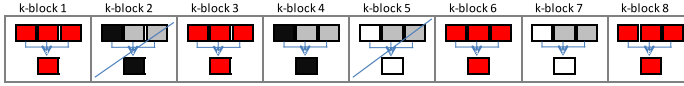
Minimized rule, t = 1



12 bits

Long-term information loss: k-block 1,3,6,8; -> information-dissipative

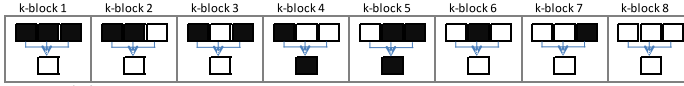
Minimized rule, t >> 100; including information loss



2 bits

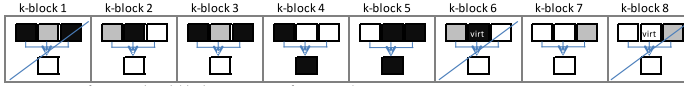
26. Rule 24

Original rule — Computational Complexity C(r)



24 bits

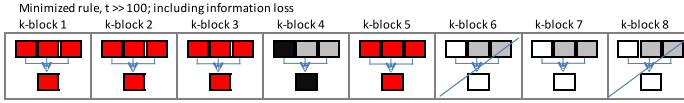
Minimized rule, t = 1



12 bits

Long-term information loss: k-block 1,2,3,5; -> information-dissipative

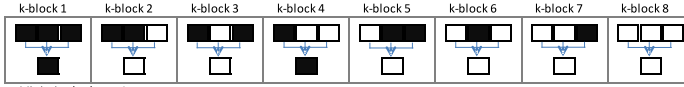
Minimized rule, t >> 100; including information loss



2 bits

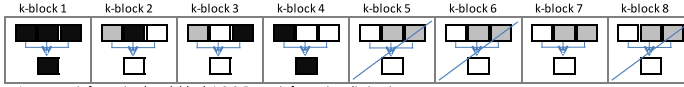
27. Rule 144

Original rule — Computational Complexity C(r)



24 bits

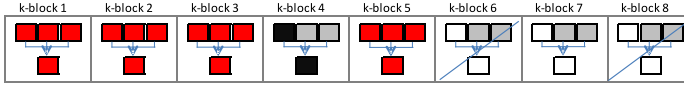
Minimized rule, t = 1



11 bits

Long-term information loss: k-block 1,2,3,5; -> information-dissipative

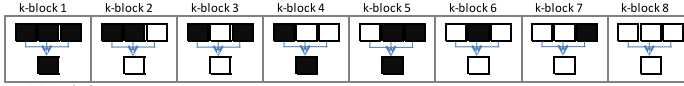
Minimized rule, t >> 100; including information loss



2 bits

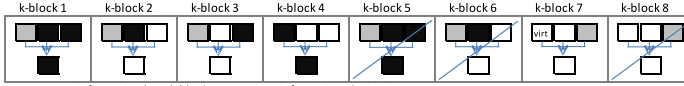
28. Rule 152

Original rule — Computational Complexity C(r)



24 bits

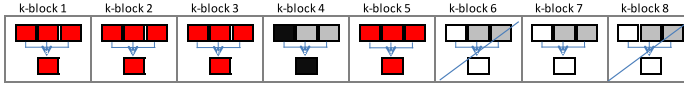
Minimized rule, t = 1



11 bits

Long-term information loss: k-block 1,2,3,5; -> information-dissipative

Minimized rule, t >> 100; including information loss



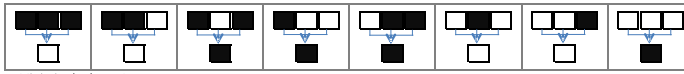
2 bits

https://doi.org/10.25088/ComplexSystems.28.2.219

29. Rule 57

Original rule — Computational Complexity C(r)

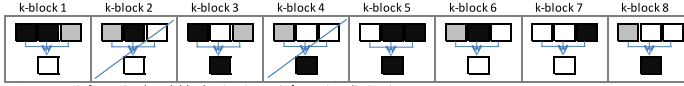
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

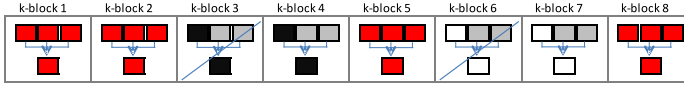


14 bits

Long-term information loss: k-block 1,2,5,8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

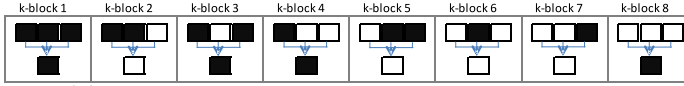


2 bits

30. Rule 177

Original rule — Computational Complexity C(r)

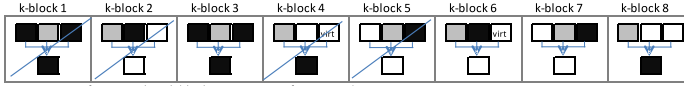
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

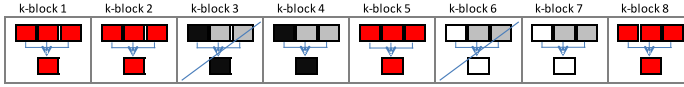


8 bits

Long-term information loss: k-block 1,2,5,8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

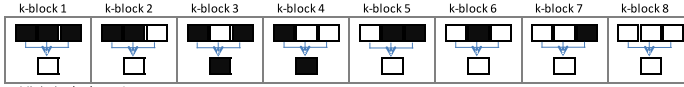


2 bits

31. Rule 48

Original rule — Computational Complexity C(r)

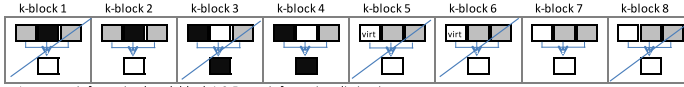
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

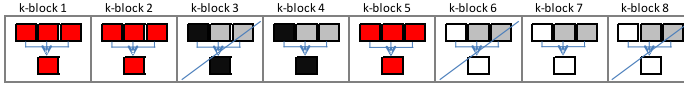


4 bits

Long-term information loss: k-block 1,2,5; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

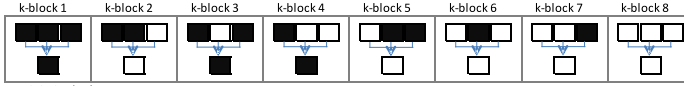


2 bits

32. Rule 176

Original rule — Computational Complexity C(r)

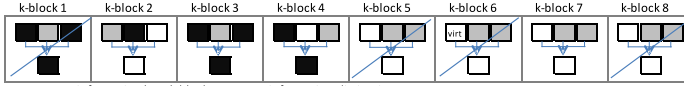
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

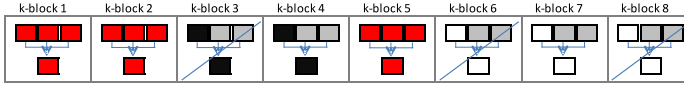


7 bits

Long-term information loss: k-block 1,2,5; -> information-dissipative

Minimized rule, t >> 100; including information loss

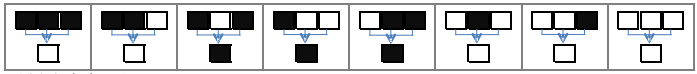
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



2 bits

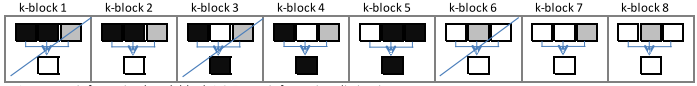
33. Rule 56

Original rule — Computational Complexity C(r)



24 bits

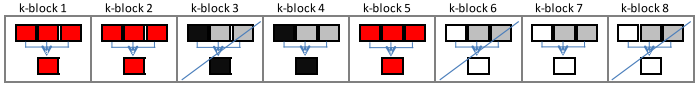
Minimized rule, t = 1



11 bits

Long-term information loss: k-block 1,2,5; -> information-dissipative

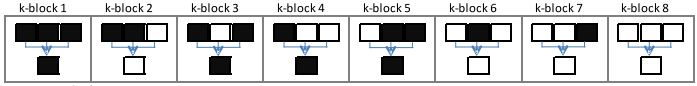
Minimized rule, t >> 100; including information loss



2 bits

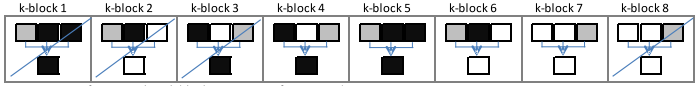
34. Rule 184

Original rule — Computational Complexity C(r)



24 bits

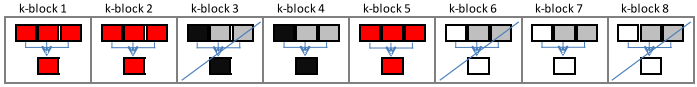
Minimized rule, t = 1



8 bits

Long-term information loss: k-block 1,2,5; -> information-dissipative

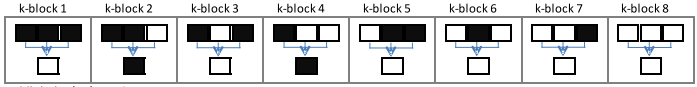
Minimized rule, t >> 100; including information loss



2 bits

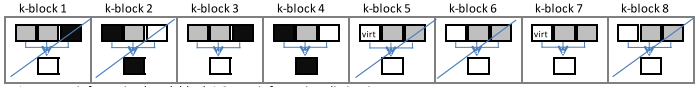
35. Rule 80

Original rule — Computational Complexity C(r)



24 bits

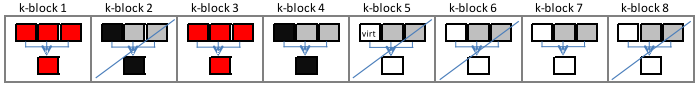
Minimized rule, t = 1



4 bits

Long-term information loss: k-block 1,3; -> information-dissipative

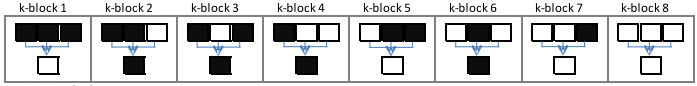
Minimized rule, t >> 100; including information loss



2 bits

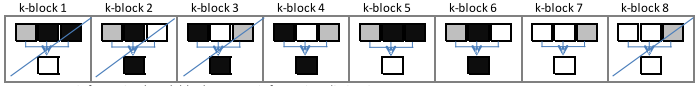
36. Rule 116

Original rule — Computational Complexity C(r)



24 bits

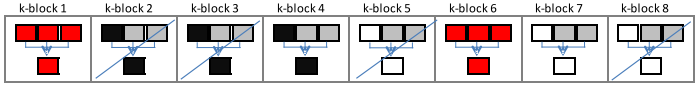
Minimized rule, t = 1



8 bits

Long-term information loss: k-block 1,6; -> information-dissipative

Minimized rule, t >> 100; including information loss

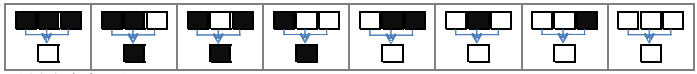


2 bits

37. Rule 112

Original rule — Computational Complexity C(r)

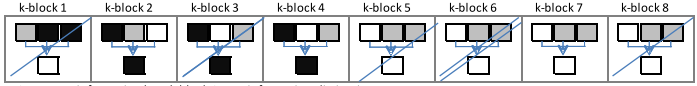
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

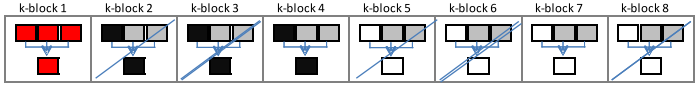


7 bits

Long-term information loss: k-block 1; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

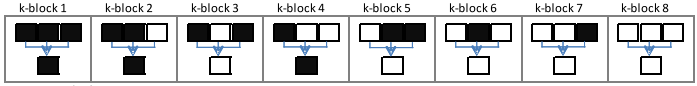


2 bits

38. Rule 208

Original rule — Computational Complexity C(r)

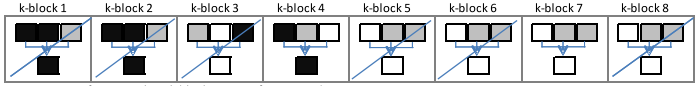
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

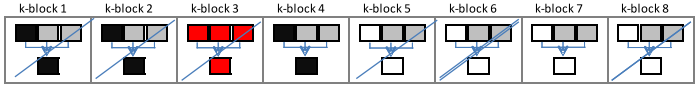


7 bits

Long-term information loss: k-block 3; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

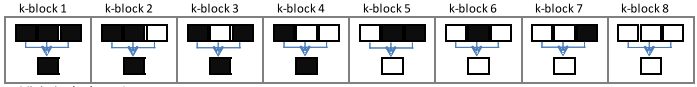


2 bits

39. Rule 240

Original rule — Computational Complexity C(r)

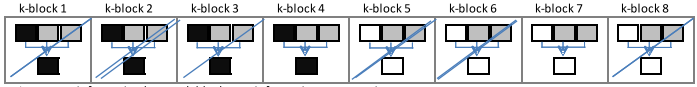
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

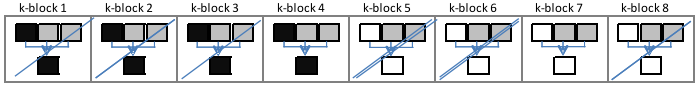


2 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

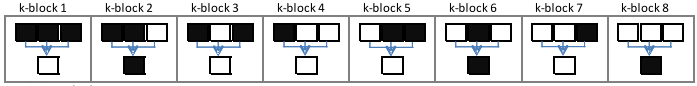


2 bits

40. Rule 69

Original rule — Computational Complexity C(r)

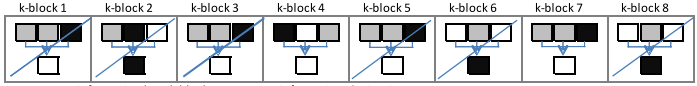
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

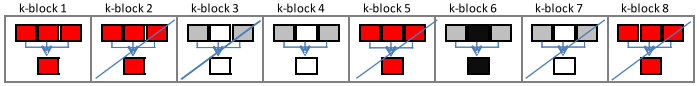


7 bits

Long-term information loss: k-block 1,2,5,8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

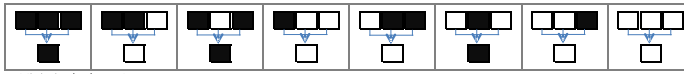


2 bits

41. Rule 164

Original rule — Computational Complexity C(r)

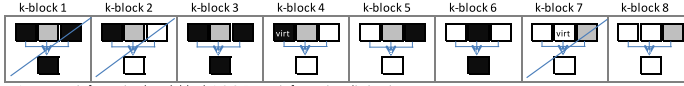
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

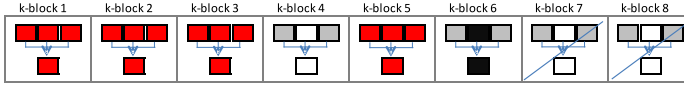


11 bits

Long-term information loss: k-block 1,2,3,5; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

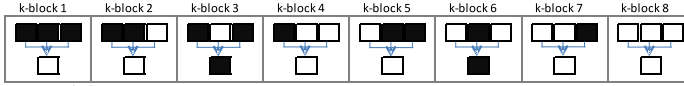


2 bits

42. Rule 36

Original rule — Computational Complexity C(r)

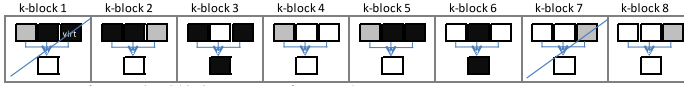
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

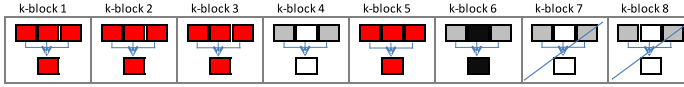


14 bits

Long-term information loss: k-block 1,2,3,5; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

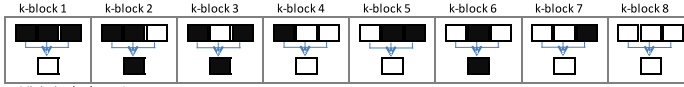


2 bits

43. Rule 100

Original rule — Computational Complexity C(r)

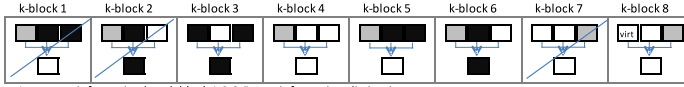
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

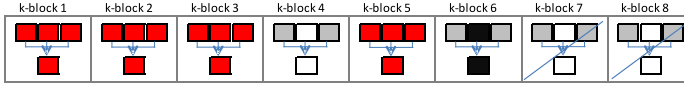


11 bits

Long-term information loss: k-block 1,2,3,5; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

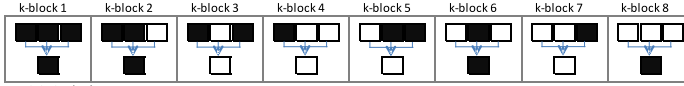


2 bits

44. Rule 197

Original rule — Computational Complexity C(r)

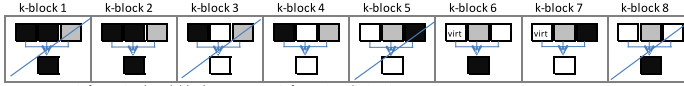
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

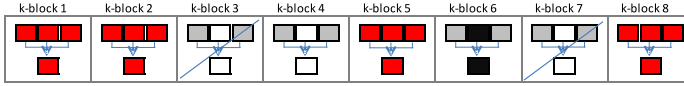


8 bits

Long-term information loss: k-block 1,2,5,8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



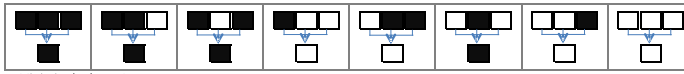
2 bits

https://doi.org/10.25088/ComplexSystems.28.2.219

45. Rule 228

Original rule — Computational Complexity C(r)

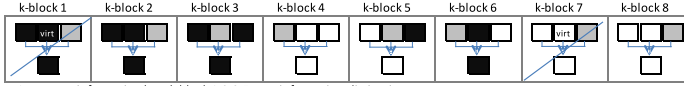
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

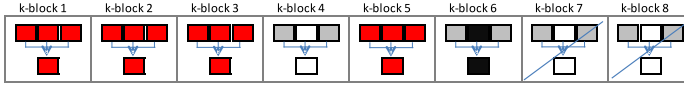


12 bits

Long-term information loss: k-block 1,2,3,5; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

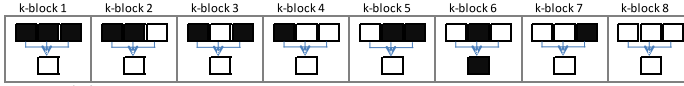


2 bits

46. Rule 4

Original rule — Computational Complexity C(r)

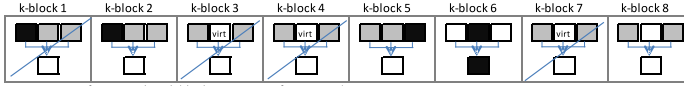
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

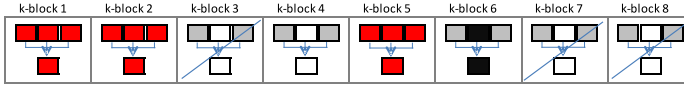


6 bits

Long-term information loss: k-block 1,2,5; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

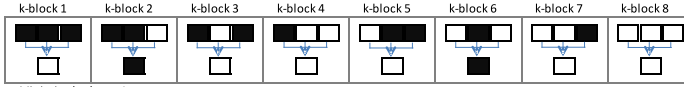


2 bits

47. Rule 68

Original rule — Computational Complexity C(r)

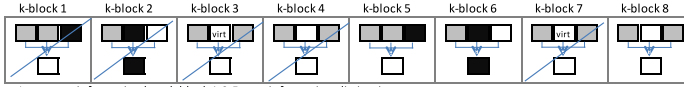
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

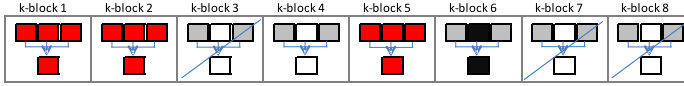


4 bits

Long-term information loss: k-block 1,2,5; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

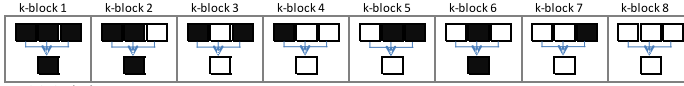


2 bits

48. Rule 196

Original rule — Computational Complexity C(r)

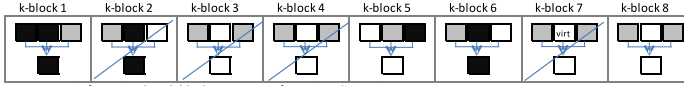
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

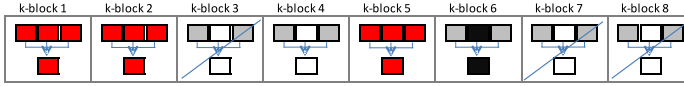


7 bits

Long-term information loss: k-block 1,2,5; -> information-dissipative

Minimized rule, t >> 100; including information loss

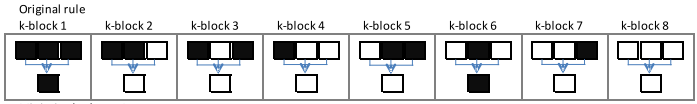
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



2 bits

49. Rule 132

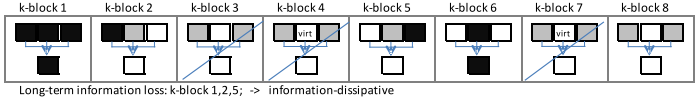
Original rule



Computational Complexity C(r)

24 bits

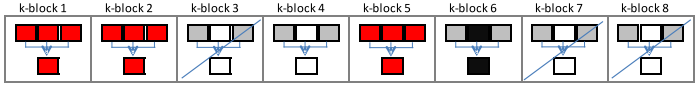
Minimized rule, t = 1



11 bits

Long-term information loss: k-block 1,2,5; -> information-dissipative

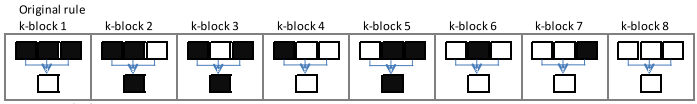
Minimized rule, t >> 100; including information loss



2 bits

50. Rule 104

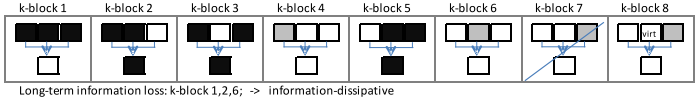
Original rule



Computational Complexity C(r)

24 bits

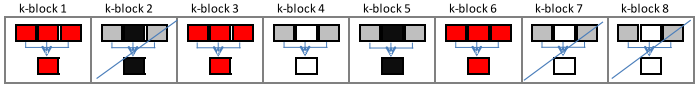
Minimized rule, t = 1



18 bits

Long-term information loss: k-block 1,2,6; -> information-dissipative

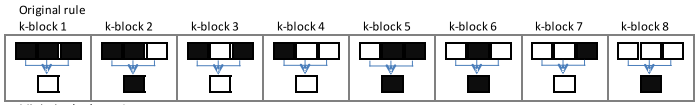
Minimized rule, t >> 100; including information loss



2 bits

51. Rule 77

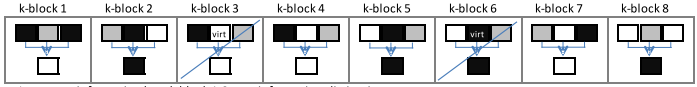
Original rule



Computational Complexity C(r)

24 bits

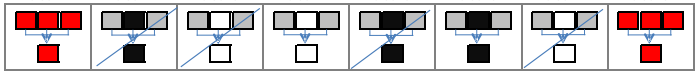
Minimized rule, t = 1



12 bits

Long-term information loss: k-block 1,8; -> information-dissipative

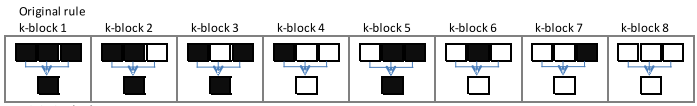
Minimized rule, t >> 100; including information loss



2 bits

52. Rule 232

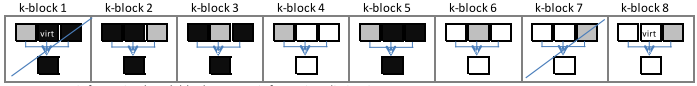
Original rule



Computational Complexity C(r)

24 bits

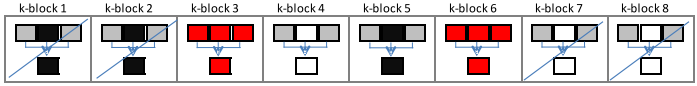
Minimized rule, t = 1



12 bits

Long-term information loss: k-block 3,6; -> information-dissipative

Minimized rule, t >> 100; including information loss



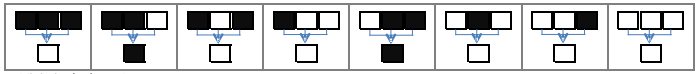
2 bits

https://doi.org/10.25088/ComplexSystems.28.2.219

53. Rule 72

Original rule — Computational Complexity C(r)

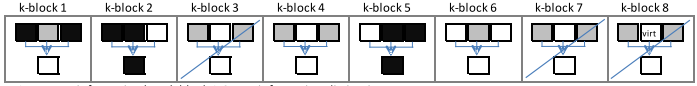
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

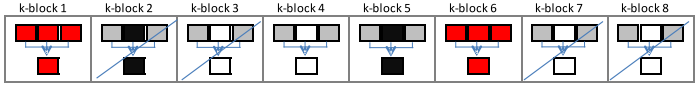


11 bits

Long-term information loss: k-block 1,6; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

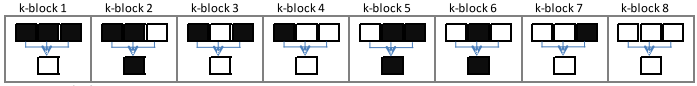


2 bits

54. Rule 76

Original rule — Computational Complexity C(r)

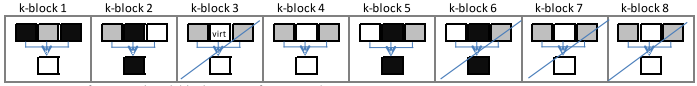
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

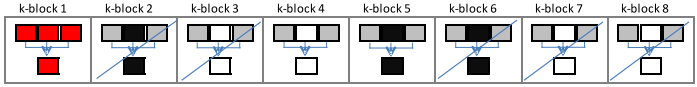


7 bits

Long-term information loss: k-block 1; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

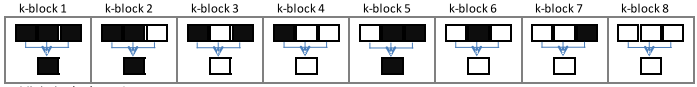


2 bits

55. Rule 200

Original rule — Computational Complexity C(r)

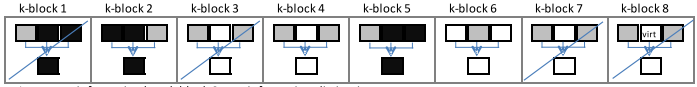
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

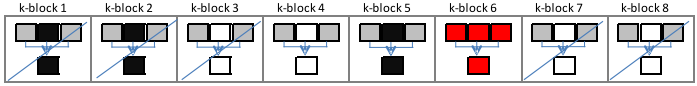


7 bits

Long-term information loss: k-block 6; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

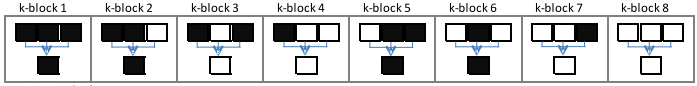


2 bits

56. Rule 204

Original rule — Computational Complexity C(r)

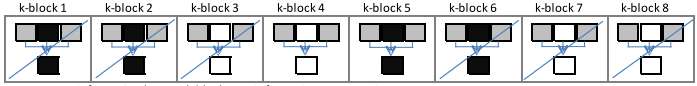
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

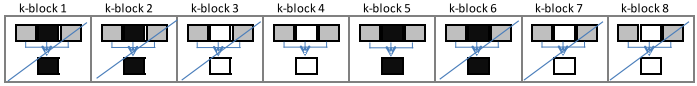


2 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

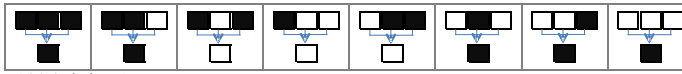


2 bits

57. Rule 199

Original rule — Computational Complexity C(r)

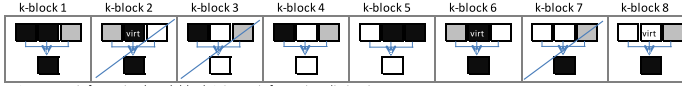
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

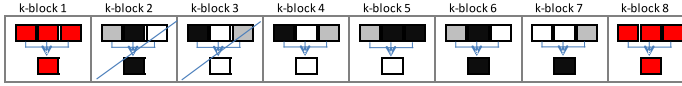


11 bits

Long-term information loss: k-block 1,8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

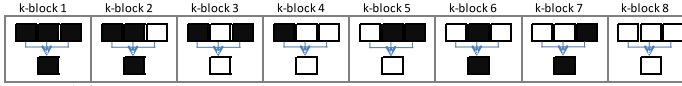


8 bits

58. Rule 198

Original rule — Computational Complexity C(r)

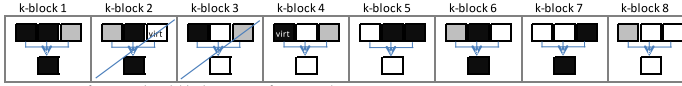
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

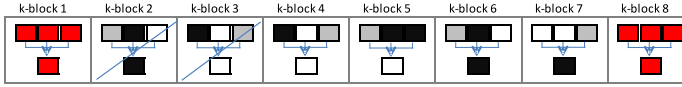


14 bits

Long-term information loss: k-block 1,8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

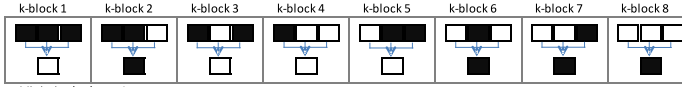


8 bits

59. Rule 71

Original rule — Computational Complexity C(r)

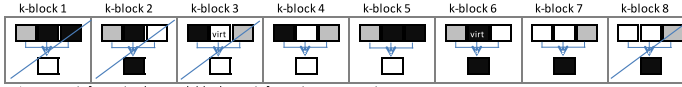
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

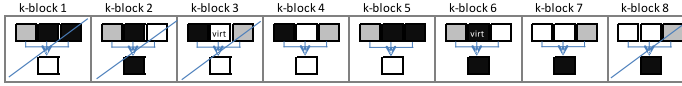


8 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

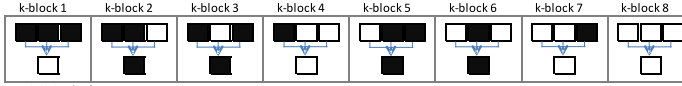


8 bits

60. Rule 108

Original rule — Computational Complexity C(r)

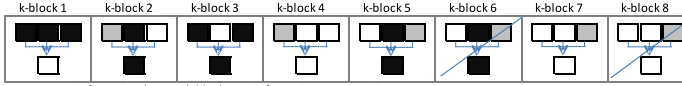
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

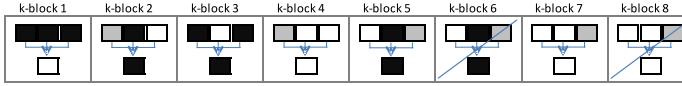


14 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



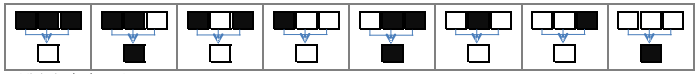
14 bits

https://doi.org/10.25088/ComplexSystems.28.2.219

61. Rule 73

Original rule — Computational Complexity C(r)

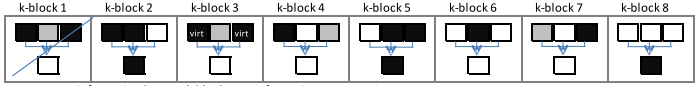
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

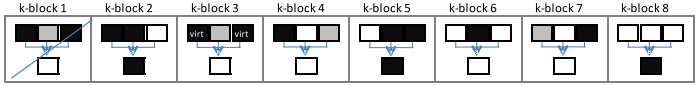


18 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

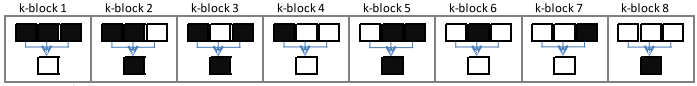


18 bits

62. Rule 105

Original rule — Computational Complexity C(r)

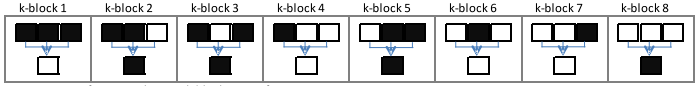
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

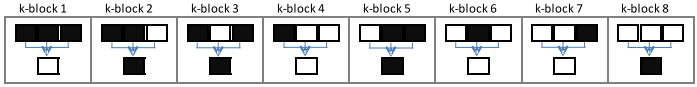


24 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

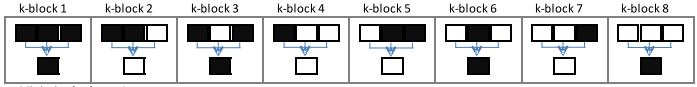


24 bits

63. Rule 165

Original rule — Computational Complexity C(r)

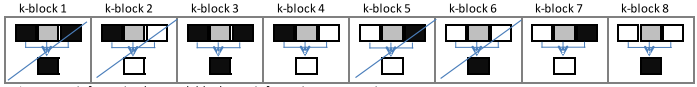
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

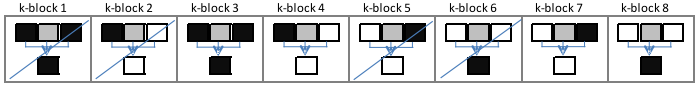


8 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

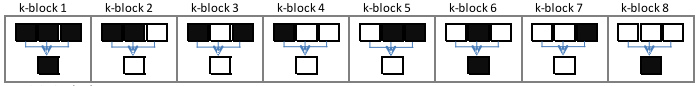


8 bits

64. Rule 133

Original rule — Computational Complexity C(r)

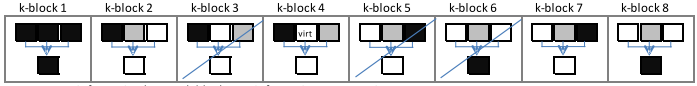
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

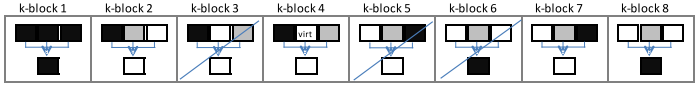


11 bits

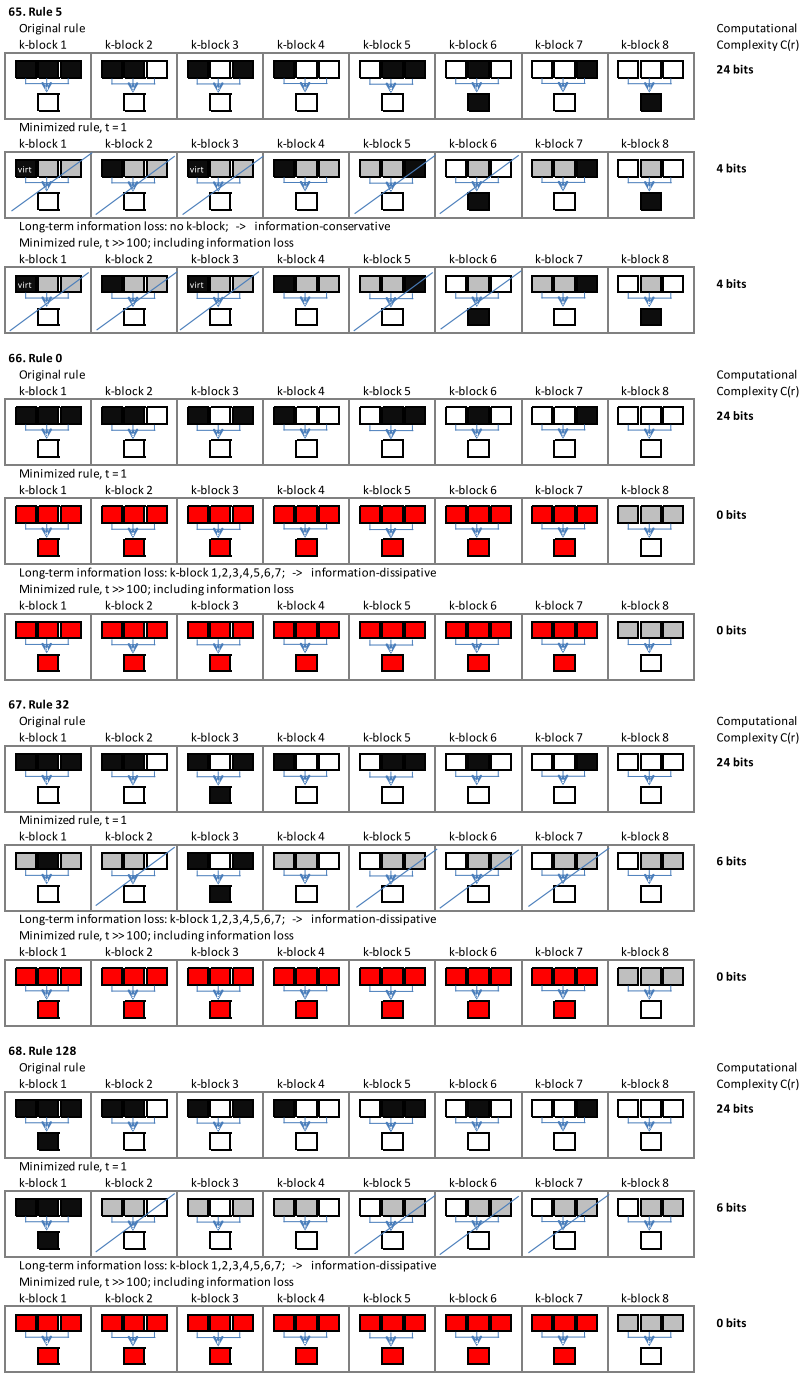
Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

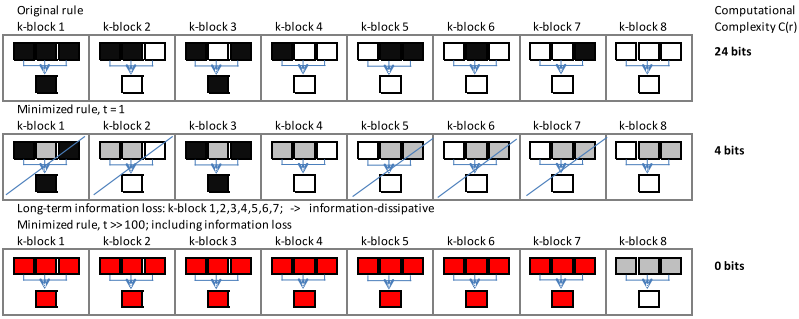
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



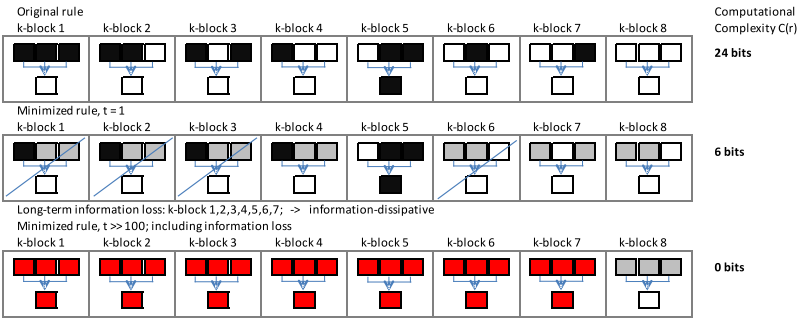
11 bits


65. Rule 5
Original rule
k-block 1
k-block 2
k-block 3
k-block 4
k-block 5
k-block 6
k-block 7
k-block 8
Computational Complexity C(r)
24 bits
Minimized rule, t = 1
virt
4 bits
Long-term information loss: no k-block; -> information-conservative
Minimized rule, t >> 100; including information loss
4 bits
66. Rule 0
Original rule
24 bits
Minimized rule, t = 1
0 bits
Long-term information loss: k-block 1,2,3,4,5,6,7; -> information-dissipative
Minimized rule, t >> 100; including information loss
0 bits
67. Rule 32
Original rule
24 bits
Minimized rule, t = 1
6 bits
Long-term information loss: k-block 1,2,3,4,5,6,7; -> information-dissipative
Minimized rule, t >> 100; including information loss
0 bits
68. Rule 128
Original rule
24 bits
Minimized rule, t = 1
6 bits
Long-term information loss: k-block 1,2,3,4,5,6,7; -> information-dissipative
Minimized rule, t >> 100; including information loss
0 bits

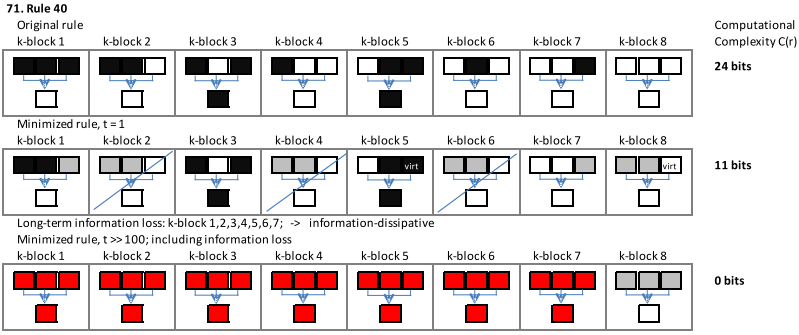
69. Rule 160



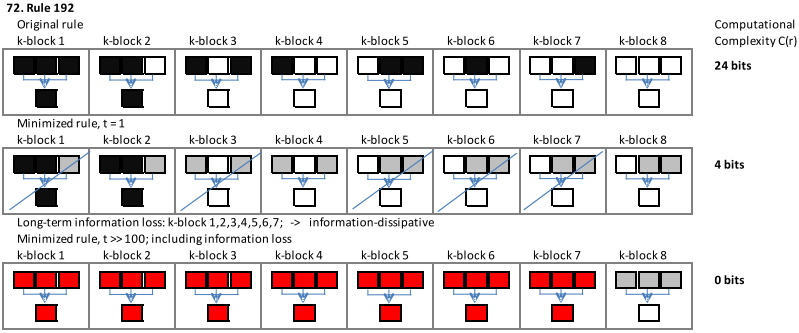
70. Rule 8



71. Rule 40

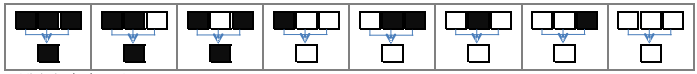


72. Rule 192



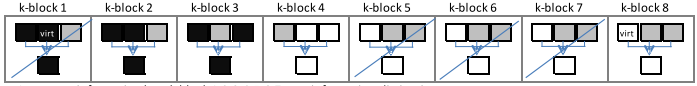
73. Rule 224

Original rule — Computational Complexity C(r)



24 bits

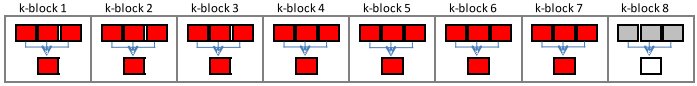
Minimized rule, t = 1



7 bits

Long-term information loss: k-block 1,2,3,4,5,6,7; -> information-dissipative

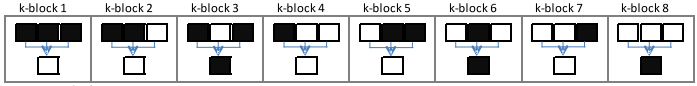
Minimized rule, t >> 100; including information loss



0 bits

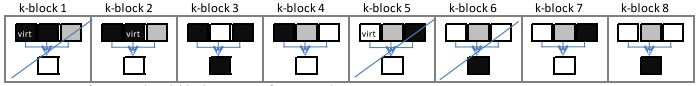
74. Rule 37

Original rule — Computational Complexity C(r)



24 bits

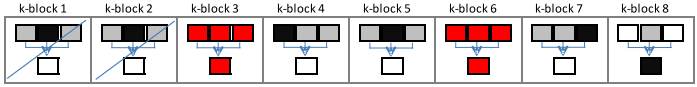
Minimized rule, t = 1



11 bits

Long-term information loss: k-block 3,6; -> information-dissipative

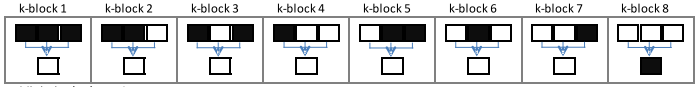
Minimized rule, t >> 100; including information loss



5 bits

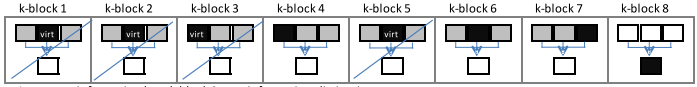
75. Rule 1

Original rule — Computational Complexity C(r)



24 bits

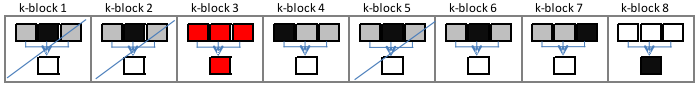
Minimized rule, t = 1



6 bits

Long-term information loss: k-block 3; -> information-dissipative

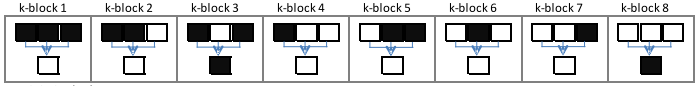
Minimized rule, t >> 100; including information loss



6 bits

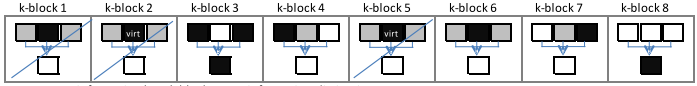
76. Rule 33

Original rule — Computational Complexity C(r)



24 bits

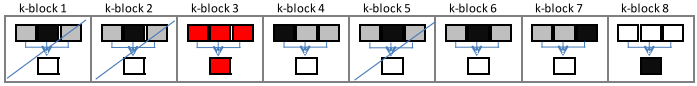
Minimized rule, t = 1



11 bits

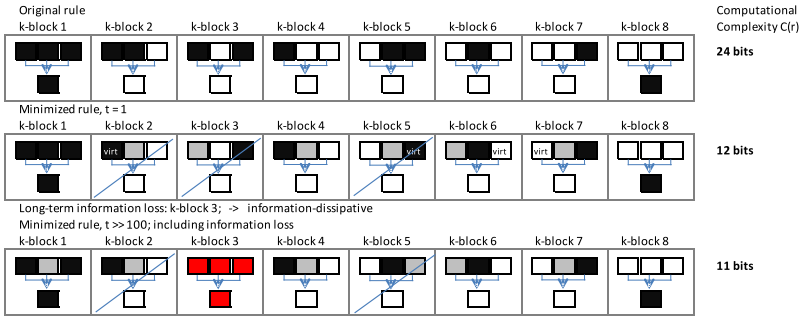
Long-term information loss: k-block 3; -> information-dissipative

Minimized rule, t >> 100; including information loss

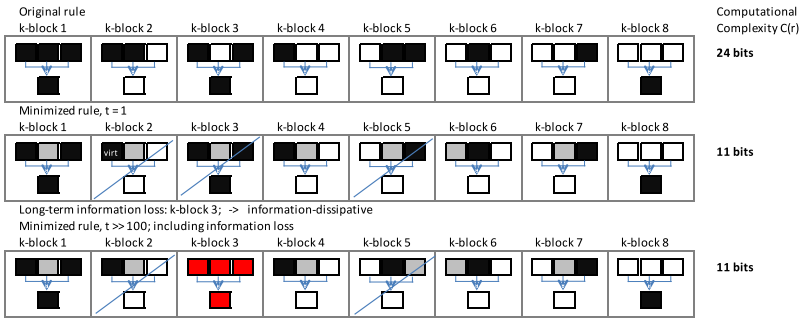


6 bits

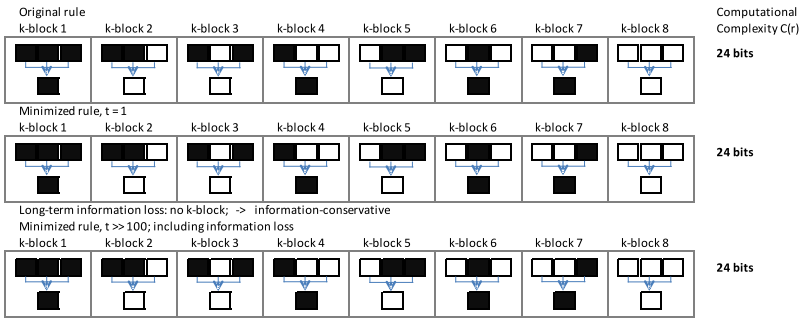
77. Rule 129



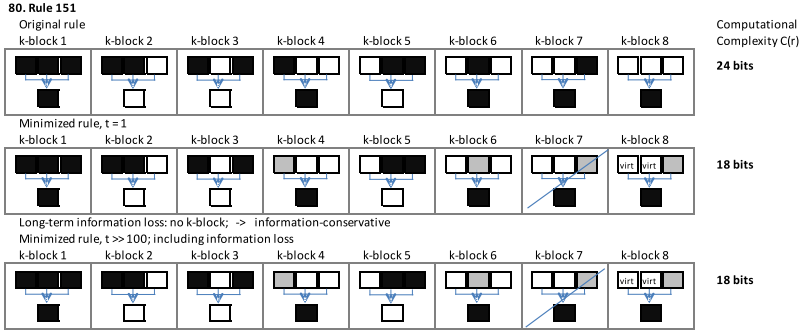
78. Rule 161



79. Rule 150



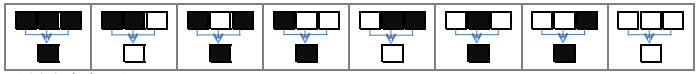
80. Rule 151



81. Rule 182

Original rule — Computational Complexity C(r)

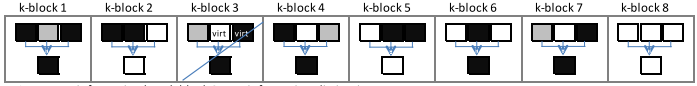
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

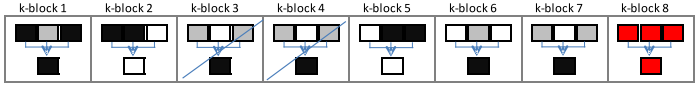


18 bits

Long-term information loss: k-block 8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

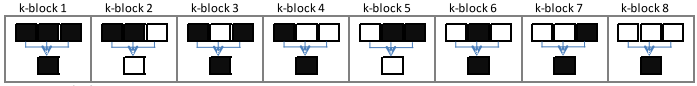


11 bits

82. Rule 183

Original rule — Computational Complexity C(r)

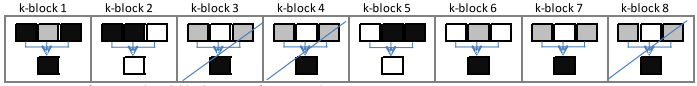
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

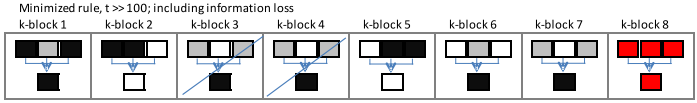


11 bits

Long-term information loss: k-block 8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

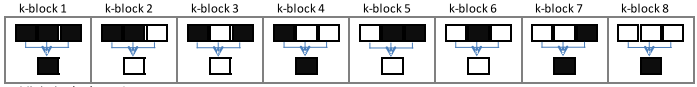


11 bits

83. Rule 147

Original rule — Computational Complexity C(r)

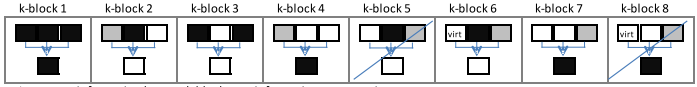
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

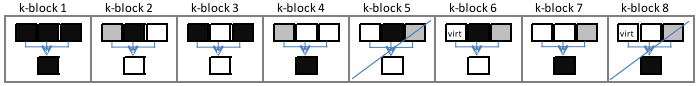


14 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

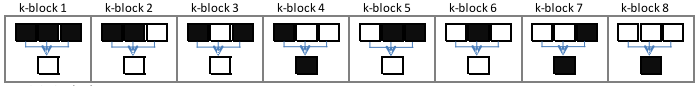


14 bits

84. Rule 19

Original rule — Computational Complexity C(r)

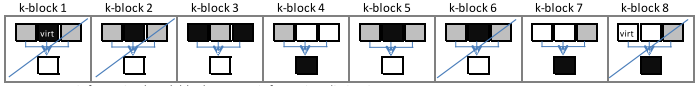
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

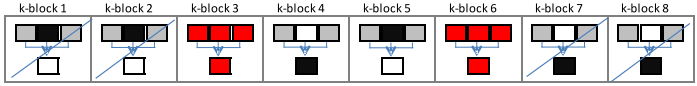


7 bits

Long-term information loss: k-block 3,6; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



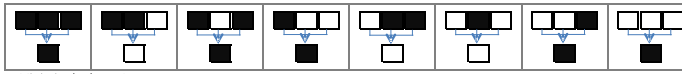
2 bits

https://doi.org/10.25088/ComplexSystems.28.2.219

85. Rule 179

Original rule — Computational Complexity C(r)

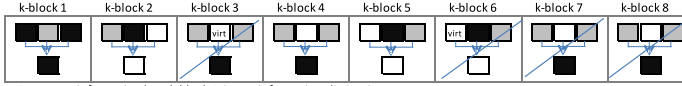
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

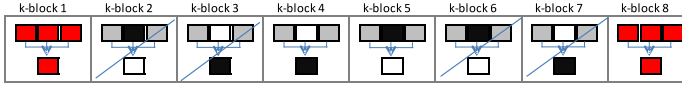


7 bits

Long-term information loss: k-block 1,8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

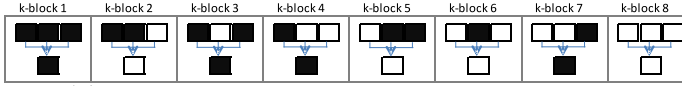


2 bits

86. Rule 178

Original rule — Computational Complexity C(r)

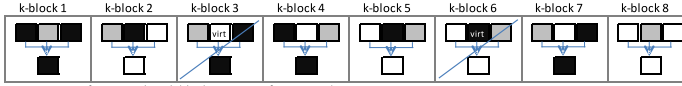
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

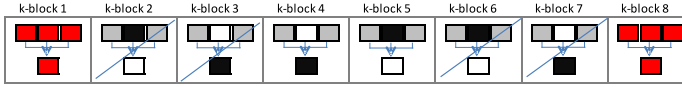


12 bits

Long-term information loss: k-block 1,8; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

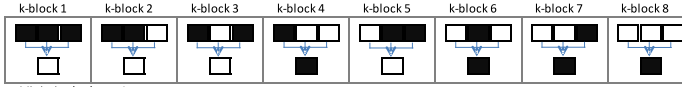


2 bits

87. Rule 23

Original rule — Computational Complexity C(r)

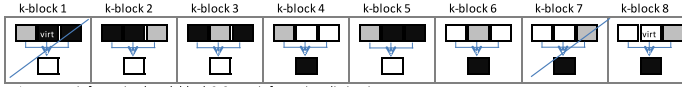
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

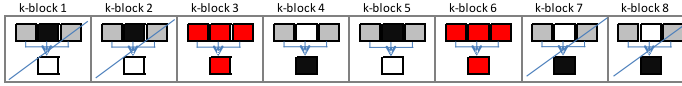


12 bits

Long-term information loss: k-block 3,6; -> information-dissipative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

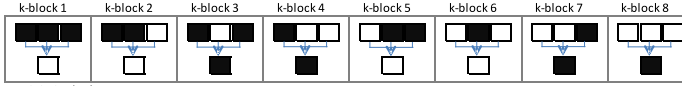


2 bits

88. Rule 51

Original rule — Computational Complexity C(r)

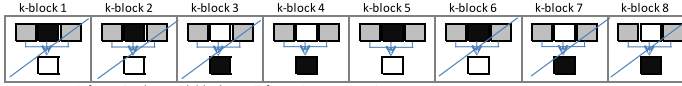
k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



24 bits

Minimized rule, t = 1

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8

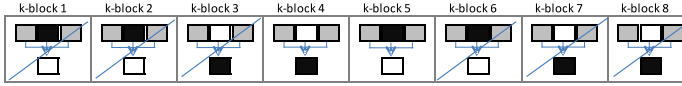


2 bits

Long-term information loss: no k-block; -> information-conservative

Minimized rule, t >> 100; including information loss

k-block 1 | k-block 2 | k-block 3 | k-block 4 | k-block 5 | k-block 6 | k-block 7 | k-block 8



2 bits