IJIRR

International Journal of Information Research and Review
Vol. 03, Issue, 09, pp. 2764-2783, September, 2016

Research Article

THEORY OF MATHEMATICAL STRINGS: THE FIRST STEPS

***Kornyushkin, A.**

Moscow Institute of Physics and Technology, Moscow

ARTICLE INFO

Article History:

Received 24th June 2016
Received in revised form
20th July 2016
Accepted 18th August 2016
Published online 30th September 2016

Keywords:

Cellular automation,
Lattice Z^N, String, Computer sciece .

ABSTRACT

The numerous internal symmetries are found in N-dimensional integer lattices (Z^N). The relation of these symmetries with the new mathematical category – the so-called the Masks (or Neighborhoods) – is shown. A set of definitions for the *Correct* Masks and *Perfect* Masks is presented; an identity between the *Correct* and *Perfect* Masks is hypothesized. The relationship between the *Perfection* of the Mask and the new category named “Mathematical String” is shown. The *Correctness* of the several Masks in Z^N (N=1,2) is proven and a simple method to find the *Correctness* for all other N is outlined. The hypothesis of high population density of *Perfect* Masks in integer lattices Z^N with large N is stated.

INTRODUCTION

This work declares a discovery of a new mathematical object: Mathematical Strings. This is a fairly simple object. A specific example is shown below (see Fig. 19 and also Fig. 10C); the origin of each line in Fig. 19 will be explained later on in the paper. At the same time, this article has its own specifics, and it requires an extended preamble. First, it is hard to find any section in mathematics that matches the scope of this work. It is just an article about a new algebra, a new symmetry, with the word symmetry to be understood in a broad sense. There is certain type of symmetry and, accordingly, mathematical category called *group*. There is a concrete example: "tetrahedral group of rotations". We introduce another type of symmetry and call it *Mathematical String*. Correspondingly, there is a concrete example: Mathematical String of the Mask (19,11). This paper is written about such an object. Due to the fact that the paper is about a completely new object, its understanding does not require any prior knowledge. So what does it take to understand it? (See Fig. 1). To understand Mathematical Strings, we need only to turn on a reversible Cellular Automaton of the second kind on the integer lattice Z^N.

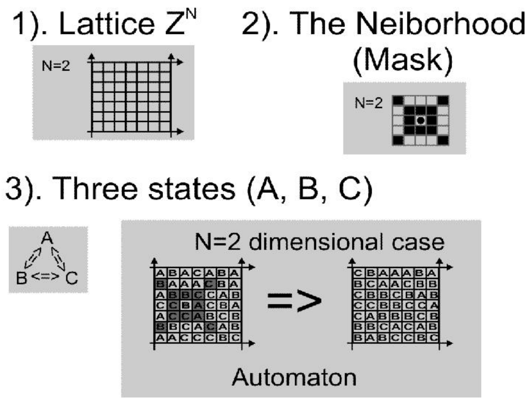


Fig. 1. What is needed for TMS?

****Corresponding author: Kornyushkin, A.,***
Moscow Institute of Physics and Technology, Moscow

We use the Automaton in its most primitive form, a particular type of the Automaton depends only on the corresponding Neighborhood; and this Automaton produces Mathematical Strings. Now, the word Neighborhood comes up. This term, which is extensively used in Automata Theory, is also crucial for Mathematical Strings. Yet, we decided to replace it with Mask. First the new term is shorter; secondly, it has more appropriate meaning for a purpose of our paper.Lets us take a *Simple* Automaton in two dimensions in zero moment of time. The Mask (as a physical object) in this case can be imagined as a sheet of black engineering paper with a few holes cut in some of its "squares". Impose it on our lattice Z^2 filled with letters A, B, and C and then find at least one letter C in the holes. Dependent on the answer, derive the meaning of the Automaton in the given point in a consecutive moment of time using certain transfer rules.

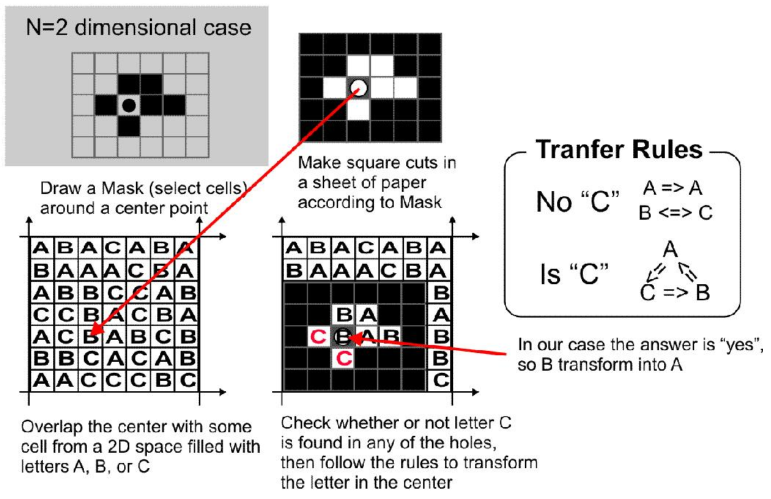

N=2 dimensional case
Draw a Mask (select cells) around a center point
Make square cuts in a sheet of paper according to Mask
Tranfer Rules
No "C" A => A B <=> C
Is "C" A C => B
Overlap the center with some cell from a 2D space filled with letters A, B, or C
Check whether or not letter C is found in any of the holes, then follow the rules to transform the letter in the center
In our case the answer is "yes", so B transform into A

Fig. 2. Explanation of Mask's principle of operation. Transfer Rules for our *Simple* Automaton

In the end of our narrative, we report on the main and totally unexpected result. All the Masks and in all (apparently) dimensions N are divided into two very distinct categories. There are Masks that form the String (we call them *Correct* or *Perfect* Masks), and the Masks that do not form the String (*Incorrect* or *Imperfect* Masks). See Fig. 3.

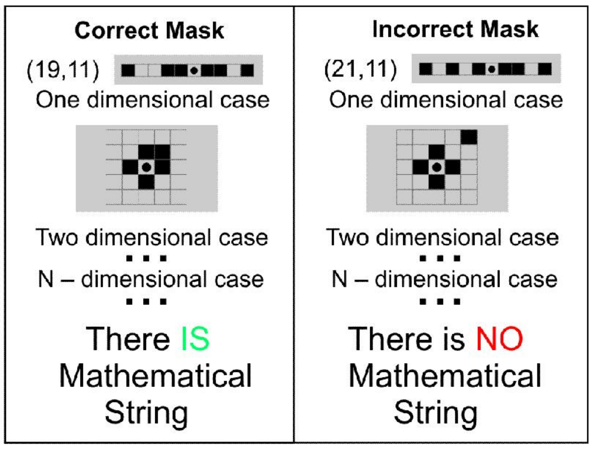


Fig. 3. *Correct* and *Incorrect* Masks

Mathematical String, in its turn, also has two very characteristic features. First, the String resembles the electron in quantum physics, the latter is both the wave and particle. Accordingly, the Mathematical String exists simultaneously in two representations: as a so-called *Simple* option and a *Table* one. It is a pure luck that this *Table* option exists. The existence of this option allows us to rigorously define and fix the String position. In fact, the very existence of *Tables* transforms a set of empirical, yet, peculiar facts into a rigorous mathematical science. Secondly, the aid of a computer is crucial to describe such a mathematical category as String. It means that Strings as a mathematical category could not been discovered till the last decade of the 20-th century, simply because powerful computers did not exist prior to that time. Below, we mention the average time it takes for a modern personal computer to perform tasks related to finding the Strings. If the Mask is *Incorrect*, then less than 1 millisecond (on average) is enough to find about this fact and stop further investigation. If the Mask is *Correct*, then it takes approximately a second to fully establish all *Transition Tables* for the *Table* Cellular Automaton. Then the personal computer (PC) can work for years, but the *Transition Tables* do not change. In order to prove the *Correctness* of these Tables, even for a simple Mask in two dimensions, it may take hours or even days of PC operation.

Do we have a confidence that the content of the Tables does not change during the extended computer operation? Strictly speaking, even after spending years checking the content of the Tables using the most powerful computer one, after all, cannot declare a direct proof. So it still remains the algebraic problem. Nevertheless, we are able to answer "yes". Once we prove that the *Table* Cellular Automaton found by our computer is self-consistent, closed, and possess symmetry of a certain kind, we can safely assert this. We hope that this article can be of interest for physicists, particularly, for quantum physicists. More specifically, it can be interested for the physicists engaged in quantization of various objects or engaged in developing the String theory (in the physics realm not in mathematical one). After all, what is the meaning of "quantization"? There is a transition from a continuous space to some kind of "lattice", at the base of which lays some sort of quantum. Perhaps, all the interested people would be surprised to learn that this very "lattice" has an infinite number of new and quite unexpected internal symmetries.

Definition of Neighborhood (Mask)

Let us define Mask *M* in *N*-dimensional space as the set of *n* integer, nonzero, and the different vectors of dimension *N*:

$$M = \{\bar{a}_1; \bar{a}_2; \dots \bar{a}_n\}; \quad \bar{a}_j = \{i_1, i_2, \dots, i_N\}; \quad j = 1, \dots, n; \quad i_k \in Z\,. \tag{1}$$

The Mask is *normal* if among other vectors it comprises all the vectors (we can call it *unit* vectors) wherein one element i_k ($k = 1, \dots, N$) is +1 or -1 while all others elements are 0. The Mask is called *primitive* if it contains ONLY *unit* vectors (in other terms, it is called *von Neumann neighborhood*). Thus, *primitive* Masks are a subclass of *normal* Masks. The Mask that for any vector $\{ i_1, i_2, \dots, i_N\}$ contains vector $\{ -i_1, -i_2, \dots, -i_N\}$ is called *central-symmetric*. In one-dimensional integer space, any Mask is easy to describe with two numbers (*m, k*) where the binary representation of *m* gives negative *i,* and the binary representation of *k* (after mirroring) gives positive *i*. For instance, (19,11) stands for Mask {-5;-2;-1;1;2;4}. This is due to the fact that (19,11) = ($2^4+2^1+2^0$, $2^0+2^1+2^3$) = (10011; 1101); after inversing ("mirroring") the sequence of digits in the second component ($1101^{\text{Mir}} \rightarrow 1011$), we obtain {-5;-2;-1;1;3;4}. Similarly, expression (21,11) =($2^4+2^2+2^0$, $2^0+2^1+2^3$) = (10101; 1101^{mir}) =(10101; 1011) denotes Mask {-5;-3;-1;1;2;4}. In the future, we will consider only *normal* Masks. From this point, any Mask = *normal* Mask. Examples of different Masks are presented in Fig. 4.

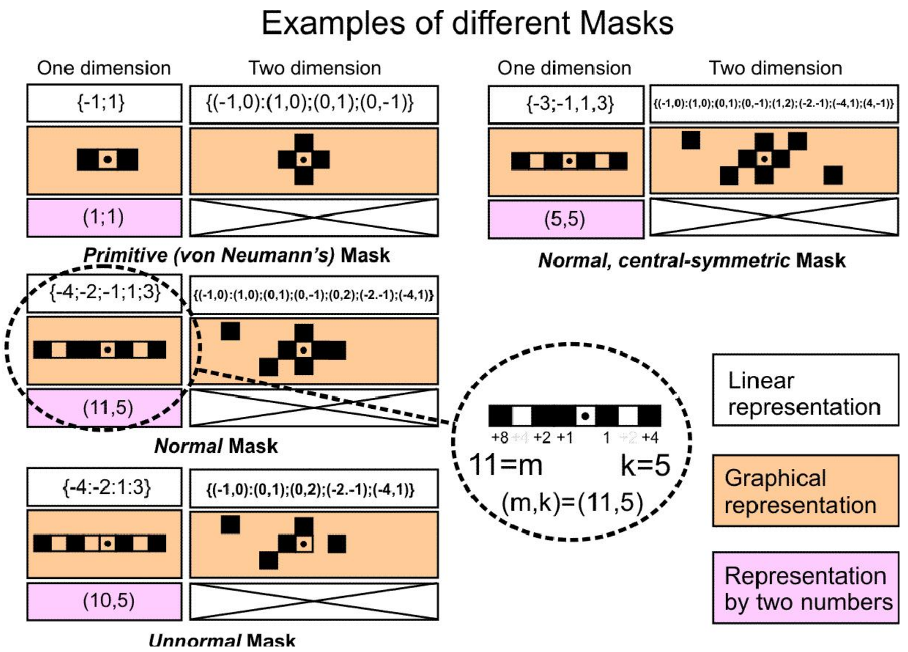


Fig. 4. ***Primitive (von Neumann's), normal,*** **and** ***central-symmetric*** **Masks**

Definition of the *Correct* Mask Theorem 1

Consider lattice Z^N. Some or none of lattice dimensions can be closed (like in a torus) whereas all the other stay infinite, this property has no impact on our future analysis. Take any Mask in *N*-dimensional integer lattice and add *Zero* vector to it (all the elements of *Zero* vector equal 0). Using this Mask with Zero vector we define a Cellular Automaton (*CA*; further we refer to this Automaton as *Table CA*) with the following properties:

1). CA cells can be in one of 6 states. At every moment of time, the state is one of the letters from following set {-x;-y;-z;+x;+y;+z}

2). The state of each cell in the next moment of time is defined by Transition Tables R_{-x}, R_{-y}, R_{-z}, R_{+x}, R_{+y}, R_{+z}, that can be generally written as

$$R_{-x} = \begin{pmatrix} a_{1,0}, a_{1,1}, .., a_{1,n} \\ a_{2,0}, a_{2,1}, .., a_{2,n} \\ ... \\ a_{c,0}, a_{c,1}, .., a_{c,n} \end{pmatrix}; \quad a_{i,j} \in \{-x;-y;-z;+x;+y;+z\}. \tag{2}$$

The number of columns in the Tables is *n* (the number points in the Mask) plus 1. We take *c* as the number of rows in the Tables . Both *n* and *c* will be constantly used in the text below. (All the variables that are used in this paper are listed in Appendix 1).

Example of *Transitional Table* for a Mask (1,1); *n=2*, *c=9* (the rule how to find *c* for a particular Mask will be explained further on) is shown below:

$$R_{-x} = \begin{pmatrix} -y,+y,-y \\ -y,+y,-z \\ -z,+y,-y \\ -y,+y,+y \\ +y,+y,-y \\ -z,-z,-z \\ -z,-z,+y \\ +y,-z,-z \\ +y,-z,+y \end{pmatrix}. \tag{3}$$

For a given Mask in Z^N, *Transition Tables* can be established for other indices (that is -y,-z, +x,+y,+z). We remind here that there are *n+1* columns in *Transition Tables R* due to inclusion of *Zero* vector. No two rows in a set of *Transition Tables* can be the same. Then a set of standard routines is executed. First, we arbitrarily enumerate our Mask points plus *Zero* vector (the center) with numbers *j* . (Fig. 5 shows *Zero* "point" with a dot). Secondly, we superimpose (overlay) each cell in Z^N with zero point of our Mask. Note: it is due to that "overlay" operation, the name "*Mask*" seems more appropriate than the "*Neighborhood*". Thirdly, we write out row $\{a_0, ..., a_j, ..., a_n\}$ containing the sequence of states that are taken from the cells on the lattice Z^N where the enumerated vectors comprising our Mask end up (see Fig. 5 for illustration). Fourthly, we check *Transition Tables* R_{-x}, R_{-y}, R_{-z}, R_{+x}, R_{+y}, R_{+z} on a subject of whether or not such a row exists in any of the *Transition Tables.* If we find such a Table, we put its index as the state of *CA* in the next moment of time (*t+1*). For example, if row$\{a_0, ..., a_j, ..., a_n\}$ appears in table R_{-x} , then the next state is *-x* ; if some other row appears in table R_{-y} , then the next state is *-y*, *etc*. Because no rows of the *Transition Tables* can be the same, there is only one state of lattice Z^N in each time point.

Fig. 5 shows the transformation of arbitrary selected element from integer lattice Z^2. It illustrates a single step of operation of a *Table CA* for a *primitive* Mask in two dimensions on a closed lattice (i.e., the lattice without "boundary") in this particular case, a surface of 6 x 6 torus.

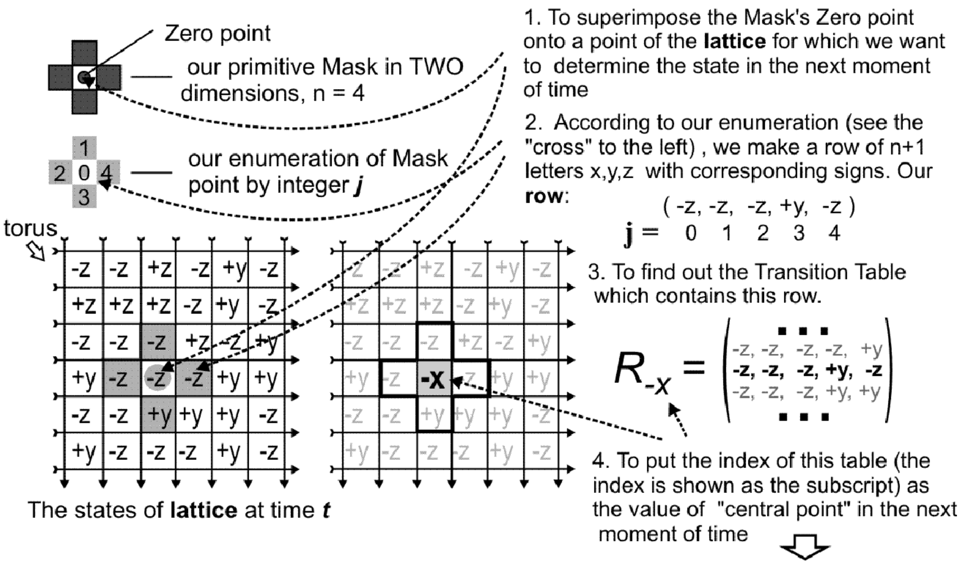

Zero point
our primitive Mask in TWO dimensions, n = 4
our enumeration of Mask point by integer j
torus
The states of lattice at time t
1. To superimpose the Mask's Zero point onto a point of the lattice for which we want to determine the state in the next moment of time
2. According to our enumeration (see the "cross" to the left) , we make a row of n+1 letters x,y,z with corresponding signs. Our row:
(-z, -z, -z, +y, -z)
j = 0 1 2 3 4
3. To find out the Transition Table which contains this row.
R-x =
-z, -z, -z, -z, +y
-z, -z, -z, +y, -z
-z, -z, -z, +y, +y
4. To put the index of this table (the index is shown as the subscript) as the value of "central point" in the next moment of time

Fig. 5. The illustration of one step of the Table CA "run" for a primitive Mask in two dimensions. For clarity, the Automaton run is shown for a single cell

In accordance to *CA* definition, the same operation should be performed at once for all cells of *CA*. Let us compare, for example, the run of our Cellular Automaton with that of well-known *CA* named *Rule 30* by Stephen Wolfram[1] (see Fig. 7). *Rule 30* has 2 states of the cells ({0;1}) while our *CA* has 6 ({-x;-y;-z;+x;+y;+z}).

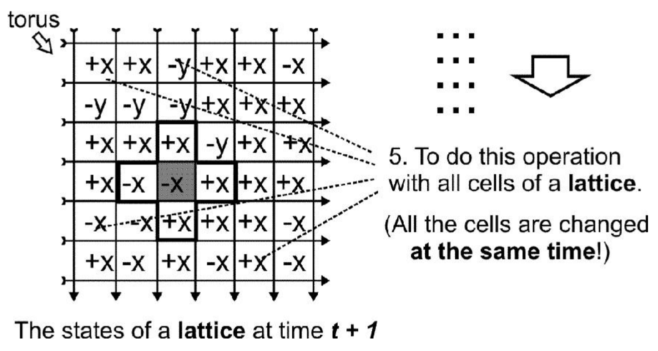

torus
5. To do this operation with all cells of a **lattice**.
(All the cells are changed **at the same time**!)
The states of a **lattice** at time ***t + 1***

Fig. 6. The operation shown in Fig. 5 are performed for all the cells at once

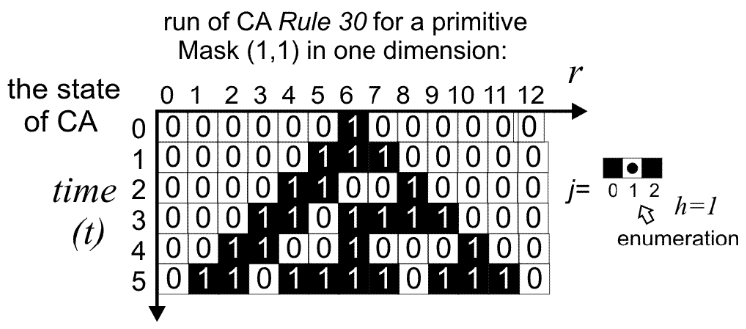

run of CA *Rule 30* for a primitive Mask (1,1) in one dimension:
the state of CA
r
time (t)
j=
0 1 2
h=1
enumeration

Fig. 7. The plot "*Automaton state vs Time*" for Wolfram's Automaton "30"

Rule 30 works or runs (if we consider the "works" through our terms) for a *primitive (von Neumann's)* Mask (1,1) in one dimension. The binary decomposition of 30 is abbreviation for the transition rules (see [Wolfram, 1983] and Table 1).

Table 1. Transition rules for *Rule 30* Automaton

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	0	0	1	1	1	1	0

From those rules, we can determine the *Transition Tables:*

$$R_0 = \begin{pmatrix} 0,0,0 \\ 1,0,1 \\ 1,1,0 \\ 1,1,1 \end{pmatrix}; \quad R_1 = \begin{pmatrix} 0,0,1 \\ 0,1,0 \\ 0,1,1 \\ 1,0,0 \end{pmatrix}. \qquad (4)$$

3) The obtained Transition Tables are interdependent and can be obtained from table R_{-x} after following substitutions:

$$R_{-x} : e = \{-x,-y,-z,+x,+y,+z\}; \quad R_{-y} : e_1 = \{-z,-x,-y,+z,+x,+y\}; \quad R_{-z} : e_2 = \{-y,-z,-x,+y,+z,+x\};$$

$$R_{+x} : e_3 = \{+x,+z,+y,-x,-z,-y\}; \quad R_{+y} : e_4 = \{+y,+x,+z,-y,-x,-z\}; \quad R_{+z} : e_5 = \{+z,+y,+x,-z,-y,-x\}.$$

These substitutions form a group; its multiplication table (***u*** x ***v***) is shown below

u \ *v*	e	e_1	e_2	e_3	e_4	e_5
e	e	e_1	e_2	e_3	e_4	e_5
e_1	e_1	e_2	e	e_4	e_5	e_3
e_2	e_2	e	e_1	e_5	e_3	e_4
e_3	e_3	e_5	e_4	e	e_2	e_1
e_4	e_4	e_3	e_5	e_1	e	e_2
e_5	e_5	e_4	e_3	e_2	e_1	e

Fig. 8.

4) Our Automaton is reversible. The inverting Transition Tables are calculated from the same Table R_{-x} using following substitutions:

$$R^{-1}_{-x}:\{-x,-z,-y,+z,+y,+x\};\quad R^{-1}_{-y}:\{-y,-x,-z,+x,+z,+y\};\quad R^{-1}_{-z}:\{-z,-y,-x,+y,+x,+z\};$$
$$R^{-1}_{+x}:\{+z,+x,+y,-x,-y,-z\};\quad R^{-1}_{+y}:\{+y,+z,+x,-z,-x,-y\};\quad R^{-1}_{+z}:\{+x,+y,+z,-y,-z,-x\}. \tag{6}$$

5) When getting started from any initial state of the integer lattice (Z^N) filled with only -x and + x (further we refer to this the state as the Beginning Point or BP), our CA will last forever. In other words, all the time as CA runs, only those states appear which are present in tables R.

6*) At least one row containing only letters "-x" and "+ x" must be found in tables R_{-y} or R_{+y} or R_{-z} or R_{+z}. (In other words, the "unionized" table R_{-y} U R_{+y} U R_{-z} U R_{+z} should contain this row).*

7) All points (vectors) in the Mask are important. In other words, no point of the Mask can be safely eliminated without changing the Automaton.

Definition 1. (Definition of a *Correc*t Mask). If, for a given Mask, it is possible to find *CA* meeting all seven conditions, then we call this Mask *Correct*. If it is not possible (or until proven otherwise), we call such a Mask *Incorrect*.

Let us explain why the very existence of *Correct* Mask*s* is a "miracle" of its kind. It is far from being obvious that at least one Mask is *Correct* in some N-dimensional space. From point 5 it follows that the Automaton must be able to make at least the first step, therefore, the *Tables* should contain all the 2^{n+1} states formed from the letters of the same type "*x"*. At some initial conditions after the first step (see point 6) there must appear a new letter (not "*+x" or "–x"*) in the states. Now, we are curious about the way the rows containing these letters are transformed. Assuming that we know the answer, it will immediately be transferred into six *Transition Tables* by performing substitutions, see expressions (5). Note that it will also be transferred into six *Inverting Tables* and, in the end, into the same *Transition Tables*. Each step causes new transfers. It may very soon turn out that two identical rows appear in two different tables, and this is prohibited. Yet, a situation is not that bad. As a matter of fact, there is a huge number of the *Correct* Masks and apparently in all dimensions.

Theorem 1. All Masks in one and two dimensions which are not colored by black in Fig. 9 (at least with n<9) are *Correct*.

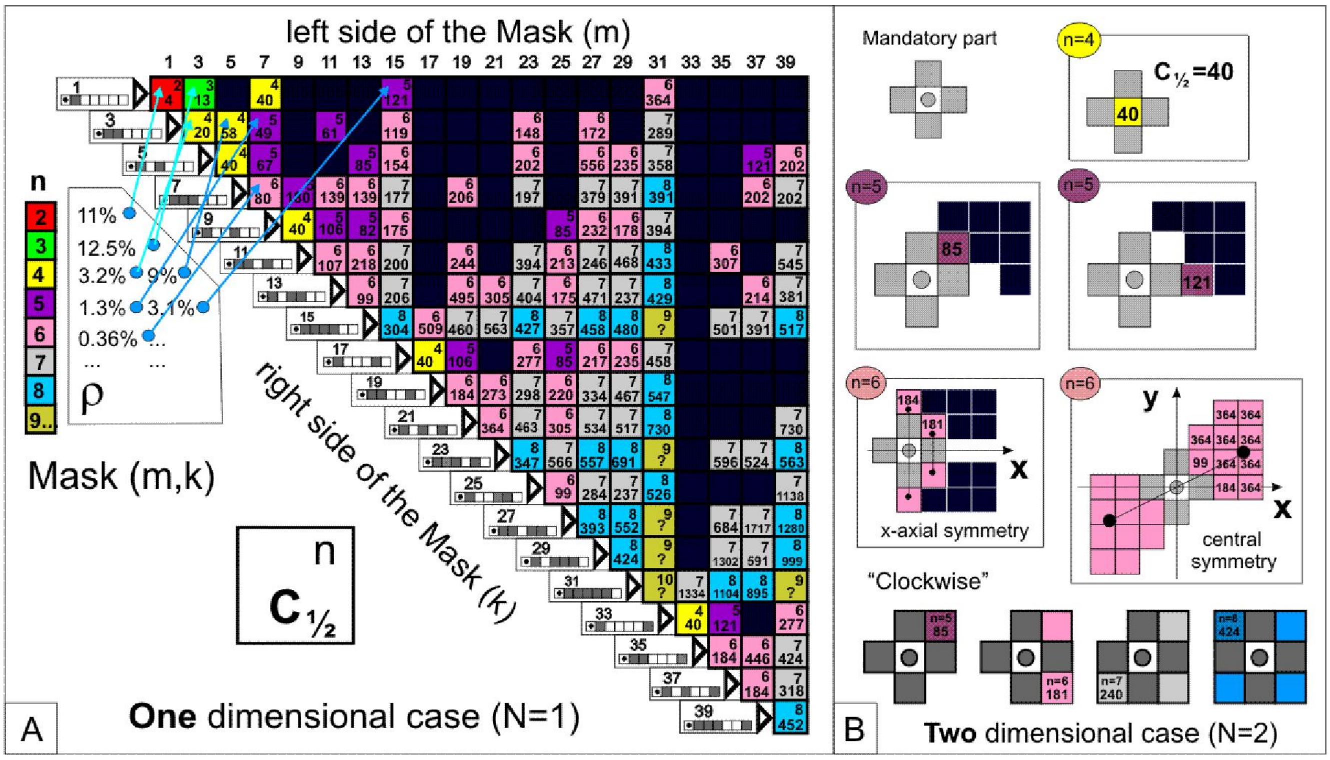


Fig. 9. In the entire figure, black color means that the Mask is *Incorrect*. Panel *A* shows the *Correct* Masks *(m, k)* in one dimension – those are the cells with the numbers inside, colored differently from black. Because the number of rows in Tables *R* is always odd, we introduce important parameter: $c_{1/2}$=(c-1)/2, which value is indicated in the cell. Numbers in cells: on the top – the number of points in a Mask; on the bottom – value $c_{1/2}$ for the same *Correct* Mask; a question mark means that $c_{1/2}$ was not calculated, yet most likely, the Mask is *Correct.* Values of ρ represent the "density" of our *Table CA* for seven Masks (see explanations in the text). Bordered area to the left shows ρ calculated for the cells indicated by arrows

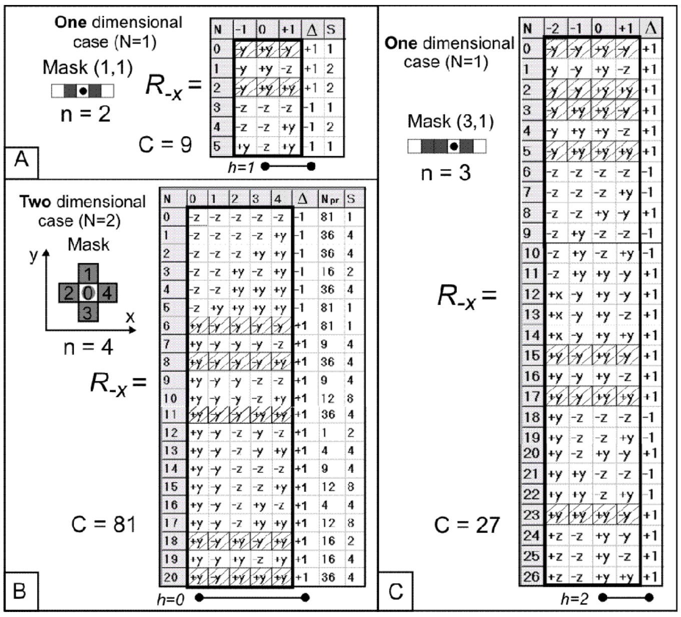
Panel *B* shows $c_{1/2}$ for the first *Correct* Masks in two dimensions. We remind that in this case *n* equals the number of cells in the Mask minus one whereas value $c_{1/2}$ will be explained in the next chapter where we will deal with the *Transition Tables*. Further details are discussed in the text.

Panel *B* in Fig. 9 also shows: the upper row – «a mandatory part» of all *normal* Masks in two dimensions and a *primitive* Mask ("cross" with $n=4$, $c_{1/2}=40$); the second row – examples of Masks for $n=5$; the third row – examples of Masks for $n=6$; and the bottom row there is "the transition from *von Neumann neighborhood* to *Moore neighborhood*" – the clockwise "attachment" of the cells to the *primitive* Mask that produces the *Correct* Masks with *n=5, 6, 7,* and *8*. It is "obvious" that all *primitive* Masks in all dimensions are *Correct*, and values *c* for these Masks equal 9^{N-1} for any N, where N is the dimension of the space. The second row (see Fig. 9B) shows two examples of *Correct* Masks for $n=5$ composed by the addition of points *(x=1, y=1)* or *(x=2, y=0)*. The third row of panel *B* in Fig. 9 also shows *Correct* Masks for $n=6$: one way to built the *Correct* Mask for $n=6$ is using the x-axis symmetry and another one, using central symmetry ($c_{1/2}$ for $n=6$ are shown in the corresponding cells). In the bottom panel, we add the cells, one by one, to the mandatory part of the two-dimensional Mask starting from the top-right "cell" and then going clockwise. We obtain four *Correct* Masks: ($n=5$, $c_{1/2}=85$); ($n=6$, $c_{1/2}=181$); ($n=7$, $c_{1/2}=240$); and ($n=8$, $c_{1/2}=424$). The black cells in Fig. 9B indicate some "additions" that yield the *Incorrect* Mask.

Parameter ρ in Fig 9 is the "density" of corresponding CA, e.g., 11% for Mask *(1,1)*; 12.5 % for Mask (3,1) etc. We define here the "density" as the ratio of the number of rows in six matrices *R* to a number of all possible states $6c/6^{n+1}=c/6^n$ expressed in percent. It is hard to imagine how accurately our CA must walk! In all states, except the first two, more than 90% of the rows are "forbidden". Yet, those CA in its time propagation never set a foot on the "forbidden" ground. That is why it can be considered as a "miracle" of its kind.

Proof. Note, the proof for Theorem 1 is obtained using personal computer. The software program with the source code (common to dimensions N=1 and N=2) can be found on the site http://k3e.hop.ru/proofT1.zip.

In the beginning, we present the *Transition Tables* for *primitive* Masks in one (Fig. 10A) and two (Fig. 10B) dimensions as well as for Mask (3,1) in one dimension (Fig. 10C).



One dimensional case (N=1)
Mask (1,1)
n = 2
R_{-x} =
C = 9
h=1
A
Two dimensional case (N=2)
Mask
y
x
1
2 0 4
3
n = 4
R_{-x} =
C = 81
h=0
B
One dimensional case (N=1)
Mask (3,1)
n = 3
R_{-x} =
C = 27
h=2
C

N	-1	0	+1	Δ	S
0	-y	+y	-y	+1	1
1	-y	+y	-z	+1	2
2	-y	+y	+y	+1	2
3	-z	-z	-z	-1	1
4	-z	-z	+y	-1	2
5	+y	-z	+y	-1	1

N	0	1	2	3	4	Δ	N pr	S
0	-z	-z	-z	-z	-z	-1	81	1
1	-z	-z	-z	-z	+y	-1	36	4
2	-z	-z	-z	+y	+y	-1	36	4
3	-z	-z	+y	-z	+y	-1	16	2
4	-z	-z	+y	+y	+y	-1	36	4
5	-z	+y	+y	+y	+y	-1	81	1
6	+y	-y	-y	-y	-y	+1	81	1
7	+y	-y	-y	-y	-z	+1	9	4
8	+y	-y	-y	-y	+y	+1	36	4
9	+y	-y	-y	-z	-z	+1	9	4
10	+y	-y	-y	-z	+y	+1	12	8
11	+y	-y	-y	+y	+y	+1	36	4
12	+y	-y	-z	-y	-z	+1	1	2
13	+y	-y	-z	-y	+y	+1	4	4
14	+y	-y	-z	-z	-z	+1	9	4
15	+y	-y	-z	-z	+y	+1	12	8
16	+y	-y	-z	+y	-z	+1	4	4
17	+y	-y	-z	+y	+y	+1	12	8
18	+y	-y	+y	-y	+y	+1	16	2
19	+y	-y	+y	-z	+y	+1	16	4
20	+y	-y	+y	+y	+y	+1	36	4

N	-2	-1	0	+1	Δ
0	-y	-y	+y	-y	+1
1	-y	-y	+y	-z	+1
2	-y	-y	+y	+y	+1
3	-y	+y	+y	-y	+1
4	-y	+y	+y	-z	+1
5	-y	+y	+y	+y	+1
6	-z	-z	-z	-z	-1
7	-z	-z	-z	+y	-1
8	-z	-z	+y	-y	+1
9	-z	+y	-z	-z	-1
10	-z	+y	-z	+y	-1
11	-z	+y	+y	-y	+1
12	+x	-y	+y	-y	+1
13	+x	-y	+y	-z	+1
14	+x	-y	+y	+y	+1
15	+y	-y	+y	-y	+1
16	+y	-y	+y	-z	+1
17	+y	-y	+y	+y	+1
18	+y	-z	-z	-z	-1
19	+y	-z	-z	+y	-1
20	+y	-z	+y	-y	+1
21	+y	+y	-z	-z	-1
22	+y	+y	-z	+y	-1
23	+y	+y	+y	-y	+1
24	+z	-z	+y	-y	+1
25	+z	-z	+y	-z	+1
26	+z	-z	+y	+y	+1

Fig. 10. Tables R_{-x} for *Correct* Masks. Panel *A* for Mask (1,1); panel *B* for *primitive* Mask in two dimensions; panel *C* for Mask (3,1) in one dimension. The explanation is in the text

Let us explain what is depicted in Fig. 10. In each of three panels, we move from the left to the right.

Some rows in Fig. 10 containing only letter "*y*" and marked with inclined lines (shading) indicate that these rows are important to perform the "mathematical induction" step in our proof, which is conducted further on in the text. Let us consider letter "*y*" (note that in case of letter "*x*" and "*z*" everything is considered similarly). To prove the *Correctness* of our Mask, first, we need to make sure that our table contains all the 2^{n+1} rows consisting of one and the same letter "x". The R_{-x} table has a row containing only "-z". Therefore, table R_{-z} has a row containing only "*-y*" (see Eq.5, point 3), and table R_{+x} has a row containing only "*+y*". The remaining $(2^{n+1}-2)/2$ variants are distributed over tables R_{-x} and R_{+z}. Moreover, the variants in R_{-x} are complementary to variants R_{+z} (note that all rows from R_{-x} are multiplied by -1 in R_{+z}). Let us check the number of shaded rows. Consider panel *A* (Fig. 10): on one hand, we have $(2^{2+1}-2)/2=3$ rows containing only letter "y" . On the other hand, if in the same panel *A*, we sum up the numbers in the column S for rows containing only "y" (S stands for symmetry, explained below) then 1 + 2 = 3. Panel *B* (Fig. 10):

similarly, using the above expression we can find $(2^{3+1}-2)/2 = 7$ rows or , on another hand, summing up number in columns S, we get the same number 7 (S =1 everywhere because of no symmetry for the Mask). Finally, panel *C*: ($2^{4+1}-2$)/ 2 = 15 or summing up the rows with account of their symmetry we get 1 + 4 + 4 + 2 + 4 = 15 lines. Column "Δ" is just a sign of a letter in "zero" point (column) of table *R* (the relationship is shown by the connected dots; Fig. 10). More detailed information about that column will be given in the next chapter.Column "N_{pr}" (see Fig. 10C) shows the number of instances the given configuration appeared in our proof (see below). The values in this column are given in multiples of *c* (i.e., this entire column should be multiplied by 81). Then for Figs.10A and 10B it turns out that this entire column is filled only with value 1. We could have hypothesized that all the values in column N_{pr} were multiples of *c* all the time. But this is not so. This hypothesis turns invalid already for the next *Correct* Mask (5,3).

Column "S" serves to reduce the list of entries. Each row can be found S times in that list, taking into account the axial symmetry (Fig. 10A) or square symmetry (Fig. 10B). Let us return to the proof. If table R_{-x} is already known for some Mask, then the following action on the proof is clear. Yet, we are not certain about the outcome of such a process. We made the first step in the mathematical induction, i.e., we showed that all 2^{n+1} rows containing only letter "x" are present in the *Transition Tables*. Now, let us move on. Given: assume that at time point *t* in every point of *CA* the content of the Mask is permitted, i.e. all the rows in the Masks belong to tables *R*. It is necessary to prove that at the next time point *(t +1)* our CA will have the same property. The solution can be found through the proof by exhaustion performed on a personal computer. Let us unite all *Transitions Tables* (R_{-x}, R_{-y},R_{-z}, R_{+x}, R_{+y},R_{+z}) into one *Table* and named it $R^{(6)}$. It has *6c* rows numbered *0, 1, ... k, ... 6c-1* and the numbering of its rows has the following construction. First, the rows of table R_{-x} are placed in a descending order in accordance with the selected seniority -x >-y >-z >+x >+y >+z. Other *Transition Tables* can be found by using corresponding substitutions, i.e., using the rule (5) from point 3, and numbering in these Tables does not change relatively to that in R_{-x}. After that, the Tables are concatenated (here it means the second one is attached to the bottom of the first one, etc) in similar order: R_{-x} ... R_{-y} ... R_{-z} ... R_{+x} ...R_{+y} ...R_{+z} . To determine the specific *Table* for the row with number *k*, one has to determine the floor function *[k/c]*, where *c* is the number of rows in a single Table of our Mask. Let us introduce a new term. Let us name M_{kj} as the Mask filled with the row numbered k_j from $R^{(6)}$; $k_j = 0, ..., 6c-1$ (see an example of such a filling in Fig. 11). We call set $\{k_0, k_1, ...k_n\}$ from *n+1* (*n* –number of the points in the Mask) filled Masks "illegal" if after the placement of set M_{k0} into a point with number 0 of our Masks, M_{k1} – into a point with number of 1 our Mask, etc., some point of our lattice will be filled with two different letters, that is the letters not equal to each other. Otherwise, we call our set "legal".

Now consider, for example, the primitive Mask in two dimensions (see panel *B* in Fig. 10). In Fig. 10B a natural sequence of row numbering in the considered *Transition Table* $R^{(6)}$ is truncated (or skipped) in many places due to symmetries (e.g., instead of writing in three rows, one after another, we have placed a single row and indicated the corresponding symmetry). Yet, if we write out all the rows of combined *Transition Table* $R^{(6)}$ as they are, then it will turn out, that for instance, rows number $k_0=220; k_1=165; k_2=221; k_3=358; k_4=177$ form a "legal" set (see Fig. 11).

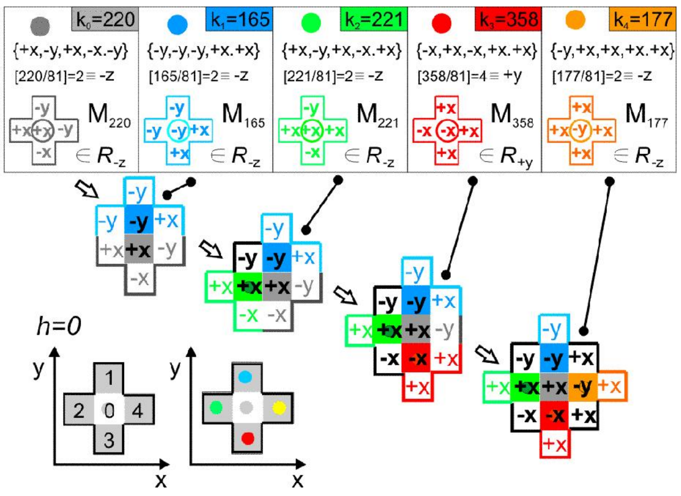
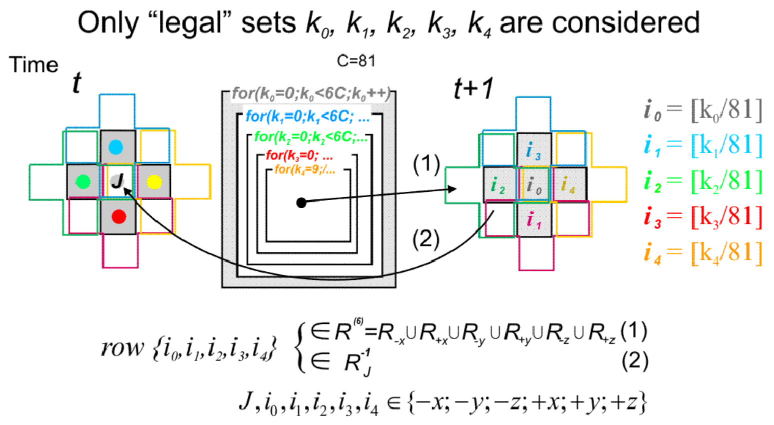


Fig. 11. "Legal" set *{220, 165, 221, 358, 177}* in case of a *primitive* Mask in two dimensions.

It is clear that only a limited number of all sets are "legal". For example, it is easy to see, that if we have any "legal" combination $\{k_0, k_1, ..., k_n\}$ and *h=0* (see Fig. 10 to determine *h*) then any set $\{m, k_1, ..., k_n\}$ where *m* it is not equal k_0, will be "illegal". Further we find "legal" sets $\{k_0, k_1, ..., k_n\}$ using the *n+1* nested loops (each loop considers *6c* variants for k_i) thus going through all possible combinations of k_i (Fig. 12). For each "legal" combination, we determine a row in the Mask in consecutive point in time – $\{[k_0/c], [k_1/c],...,[k_n/c]\}$ – and make two checks: i) whether or not the row under consideration can be found in $R^{(6)}$ and ii) whether or not the inverse transformation works. After that, we add "one" to column N_{pr} corresponding to that row and move on to a next step in a cycle...



$$row\ \{i_0,i_1,i_2,i_3,i_4\}\ \begin{cases}\in R^{(6)}=R_{-x}\cup R_{+x}\cup R_y\cup R_{+y}\cup R_z\cup R_{+z} & (1)\\ \in R^{-1}_J & (2)\end{cases}$$

$$J,i_0,i_1,i_2,i_3,i_4\in\{-x;-y;-z;+x;+y;+z\}$$

Fig. 12. Five nested loops in the proof of *Correctness* of a primitive Mask in two dimensions. In each loops, it is checked: i) whether the row *{i0, i1, i2, i3, i4}* belongs $R^{(6)}$; and ii) whether this row belongs to R^{-1}_J, where *J* – the value of the Mask k_h *(h=0)* in a Zero point in time *t*. In case of Fig. 11, row *{-z,-z,-z, y,-z}* belongs to R_{-x} and to R^{-1}_{+x}.

We reach the end, and declare that the Theorem is proved. So, we have got a key observation. If we have the *Transition Tables* the proof of *Correctness* is always (at least for the Masks shown in Fig. 9, site k3e.hop.ru/proofT1.zip) leads to a success. This fact can be considered as the greatest mystery in mathematics.

Definition of a *Simple* reversible Automaton for the Mask

Let us return to the question how one obtains the *Transition Table*. To answer this, let us briefly describe the results presented in Ref [http://arxiv.org/abs/1308.0136]. Let us introduce the concept of *Simple* reversible Automaton on a Mask. It works on the same lattice (Z^N), yet closed on all sides (because we require a limited number of states), and has 3 states A, B, C. Here we describe an ordinary second-order Automaton found by E. Fredkin that was studied by many authors, see for instance Refs (Margolus, Norman, 1984; Vichniac, 1984; Wolfram, Stephen, 1984). Each consecutive state depends on whether the Mask superimposed with a given point contains state C or not. In Case 1 (when the Mask contains state C) the following transitions are made A => C, B => A, C => B. In opposite Case 2, (no state C is contained in the Mask) the following transitions are made: A => A, B => C, C => B. It is well known that Fredkin's Automaton is reversible (see, Ref [Toffol and Margolus, 1987]), thus to change the direction of time, it is enough to replace all B with C and all C with B. Let us fill the cells only with the states taken from A and C *(t=0)* and consider the of the Automaton runs from the Beginning (Start) Point to the Mirror Point, and then back to the Beginning Point (Fig. 13).

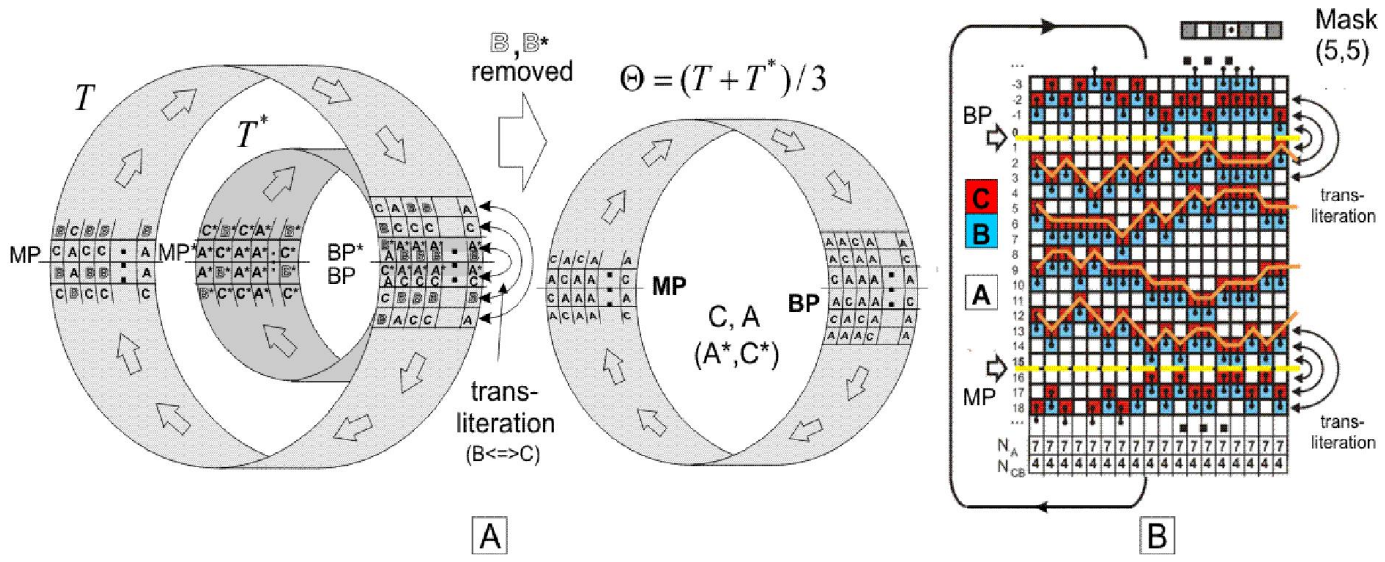


Fig. 13. Panel *A* – illustration of *Simple* CA run. One needs to take a notion that each grey "band" shown in this figure is actually a torus (it is rather difficult to depict the considered operations on a torus). Any run that starts from the Beginning Point (i.e., the state filled only "A" and "C" cells) follows the circle: the Beginning Point => the Mirror Point (MP also comprises cells "A" and "C") => Beginning Point. We have two runs: Ordinary and Additional (in the initial conditions of the latter, the replacement A⇔C is made). Ordinary and Additional runs may have different Half Periods: *T* and *T. Yet, for some Masks (the *Correct* ones), it becomes clear that after removing all cells B from both pictures, the pictures will look the same after making replacement all A⇔C. Panel *B* shows the picture "Automaton state *vs* time" for any of our runs (Ordinary or Additional). The straight yellow lines indicate the Beginning and Mirror Points. Orange lines connecting the centers of cells C show distinct C/B "bands" which do not cross each other. Further explanations see in the text.**

An additional commentary to Fig. 13. A *normality* of the Mask is necessary for two reasons. The first reason is that the Beginning point (BP) comprising cells A and C should have the Mirror point (MP) which is also composed of cells A and C (see Fig. 13A). The second reason is that the run of Automaton on the plot "*Automaton state vs Time"* follows the distinct and non-crossing bands (see Fig. 13B).

We will remind, in brief, how it has been proven. Imagine a chess figure which makes only a one-unit move forward or backward along any of axes of the *N*-dimensional integer lattice (at each step, the figure can move along some arbitrary axis); also note that it is *permitted* for this figure to make a move on the field where the figure was in the past. It is obvious that the figure can eventually visit all the points of any closed integer lattice and return to its starting position. So, using the course of this figure we transform any *N* dimensions into one as shown in Fig. 13B). After that, to prove that our bands do not cross each other bands, we use the property of a *normality,* see Ref [http://arxiv.org/abs/1308.0136]. Further, there is the main observation that, for any *Correct* Mask, CA of arbitrary size, and any initial conditions selected in the diagram "*Automaton state vs Time* " where all B cells are removed, pictures for the "Ordinary run" and "Additional run" (wherein the replacement of the initial conditions A⇔ C is made), literally coincide after the replacement all of cells A⇔ C. Consider the issue again and in detail. Let us name the diagram "*Automaton state vs time*" for the normal run of Automaton as the Ordinary World; and a similar diagram for the "additional run", as Parallel World. First, let us introduce the terms describing the Ordinary World. The state of *Simple* CA in point *r* as a function of time *t* is denoted as

$$f_{ABC}(t,r) \in \{\, A; B; C\} \tag{7}$$

Then, let us introduce function *b* which indicates whether a given cell is the B cell at a given time, or not

$$b(t,r) = \begin{cases} 1, & if\ f_{ABC}(t,r) = B \\ 0, & if\ f_{ABC}(t,r) = C \vee A \end{cases} \tag{8}$$

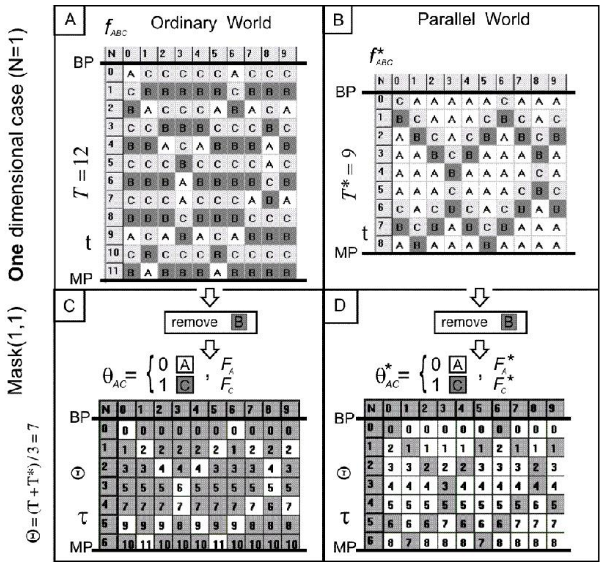
Transformation functions (with B removed) are defined as follows:

$$\theta_{AC}(t - \sum_{0 \le i \le t}^{i} b(i,r), r) = \begin{cases} 0, & if\ f_{ABC}(t,r) = A \\ 1, & if\ f_{ABC}(t,r) = C \end{cases} \tag{9}$$

$$F_A(t - \sum_{0 \le i \le t}^{i} b(i,r), r) = \begin{cases} -1, & if\ f_{ABC}(t,r) = C \\ t, & if\ f_{ABC}(t,r) = A \end{cases}$$

$$F_C(t - \sum_{0 \le i \le t}^{i} b(i,r), r) = \begin{cases} -1, & if\ f_{ABC}(t,r) = A \\ t, & if\ f_{ABC}(t,r) = C \end{cases} \tag{10}$$

See for example Fig. 14…


One dimensional case (N=1)
Mask(1,1)
Θ=(T+T*)/3=7
A
f_{ABC}
Ordinary World
BP
$T=12$
t
MP
B
Parallel World
f^*_{ABC}
BP
$T^*=9$
t
MP
C
remove B
$\theta_{AC} = \begin{cases} 0 \ \boxed{A} \\ 1 \ \boxed{C} \end{cases}, \ F_A, F_C$
BP
Θ
τ
MP
D
remove B
$\theta^*_{AC} = \begin{cases} 0 \ \boxed{A} \\ 1 \ \boxed{C} \end{cases}, \ F^*_A, F^*_C$
BP
Θ
τ
MP

N	0	1	2	3	4	5	6	7	8	9
0	A	C	C	C	C	C	A	C	C	C
1	C	B	B	B	B	B	C	B	B	B
2	B	A	C	C	C	A	B	A	C	A
3	C	C	B	B	B	C	C	C	B	C
4	B	B	A	C	A	B	B	B	A	B
5	C	C	C	B	C	C	C	C	A	C
6	B	B	B	A	B	B	B	B	C	B
7	C	C	C	A	C	C	C	A	B	A
8	B	B	B	C	B	B	B	C	C	C
9	A	C	A	B	A	C	A	B	B	B
10	C	B	C	C	C	B	C	C	C	C
11	B	A	B	B	B	A	B	B	B	B

N	0	1	2	3	4	5	6	7	8	9
0	C	A	A	A	A	A	C	A	A	A
1	B	C	A	A	A	C	B	C	A	C
2	A	B	C	A	C	B	A	B	C	B
3	A	A	B	C	B	A	A	A	B	A
4	A	A	A	B	A	A	A	A	C	A
5	A	A	A	C	A	A	A	C	B	C
6	C	A	C	B	C	A	C	B	A	B
7	B	C	B	A	B	C	B	A	A	A
8	A	B	A	A	A	B	A	A	A	A

N	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	1	2	2	2	2	2	1	2	2	2
2	3	3	4	4	4	3	3	3	4	3
3	5	5	5	6	5	5	5	5	5	5
4	7	7	7	7	7	7	7	7	6	7
5	9	9	9	8	9	9	9	8	8	8
6	10	11	10	10	10	11	10	10	10	10

N	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	2	1	1	1	1	1	2	1	1	1
2	3	3	2	2	2	3	3	3	2	3
3	4	4	4	3	4	4	4	4	4	4
4	5	5	5	5	5	5	5	5	6	5
5	6	6	6	7	6	6	6	7	7	7
6	8	7	8	8	8	7	8	8	8	8

Fig. 14. Illustration to finding the T*ransition Tables R*. Panels *A* and *B*: the diagrams *"State of Automaton vs Time"* for two Automaton runs (for Ordinary and Parallel worlds). Panels *C* and *D* – the corresponding transformations of cells with letters "A" and "C" after removal of all cells with letters "B"

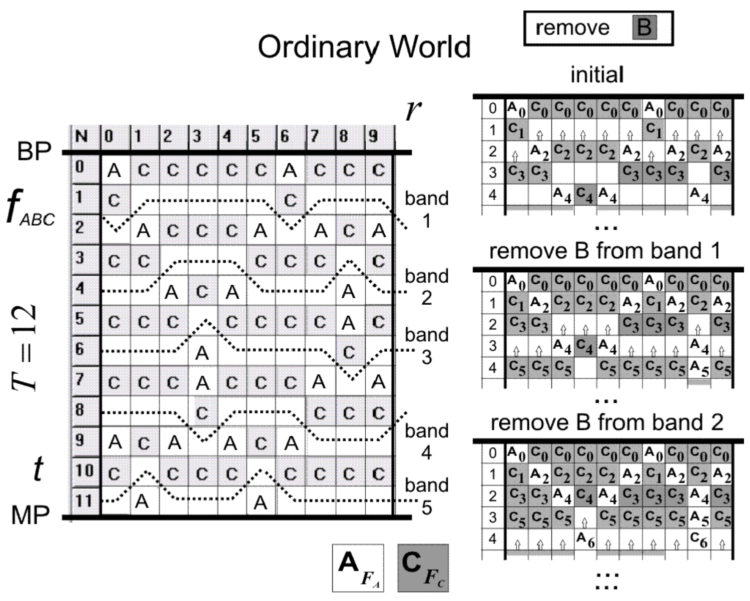


Fig. 15. Illustration of the transformation step shown in Fig. 15 – i.e., the transition from panel *A* to panel *C*. First, we remove letters B from the cells (empty the cells of letter B). After the removal, letters "A_t" and "C_t", where *t* is the row number (time coordinate) in the original *(t, r)* space , move up to fill the vacant spaces. After that, we apply Eq. 10 to "A_t" and "C_t", thus converting the "colors" into numbers, and obtain Fig. 14C

Let us take the *primitive* Mask in one dimension on the closed row (a ring) of ten points, two points of 10 being colored with two different ("opposite") colors. Yet, we repeat one more time, that all our methods can be applied without any changes to any *Correct* Mask in the *N*-dimensional space, for Automaton of any size, and for any initial conditions.

Let us introduce four new functions as follows:

$$F_{AC}(\tau,r)=\begin{cases} F_A(\tau,r), & if\ \theta_{AC}(\tau,r)=0 \\ F_C(\tau,r), & if\ \theta_{AC}(\tau,r)=1 \end{cases} \qquad F_{CA}(\tau,r)=\begin{cases} F_A(\tau,r), & if\ \theta_{AC}(\tau,r)=1 \\ F_C(\tau,r), & if\ \theta_{AC}(\tau,r)=0 \end{cases} \tag{11}$$

Note that the construction of F_{AC} *(τ, r)* was shown merely as an intermediate supplementary function to demonstrate the construction of similar function $F_{AC}^{(3)}$ with the only difference in *mod 3* (we emphasize that *mod 3* is an important manipulation, it makes all other things happen):

$$F_{AC}^{(3)}(\tau,r)=\begin{cases} F_A(\tau,r)\bmod 3, & if\ \theta_{AC}(\tau,r)=0 \\ 3+F_C(\tau,r)\bmod 3, & if\ \theta_{AC}(\tau,r)=1 \end{cases} \qquad F_{CA}^{(3)}(\tau,r)=\begin{cases} 3+F_A(\tau,r)\bmod 3, & if\ \theta_{AC}(\tau,r)=0 \\ F_C(\tau,r)\bmod 3, & if\ \theta_{AC}(\tau,r)=1 \end{cases} \tag{12}$$

It should be understood that these operations create completely different tables $F_{AC}^{(3)}$ and $F_{CA}^{(3)}$. We introduce new notations: 0 ⇔ *-x*, 1 ⇔ *-y*, 2 ⇔ *-z*, 3 ⇔ *+x*, 4 ⇔ *+y*, 5 ⇔ *+z*. We can see from Fig. 16 that these notations definitely give us the *Transition Tables*. The first *Table* is for one direction in time, and the second, for the opposite one. (If we have taken F_{CA} function instead of F_{AC} one, it would lead to replacement $R ⇔ R^{-1}$).

Now it is clear how to obtain *Transition Tables*. First, we set random initial conditions from A and C states defined on "small" integer lattices that are closed in all directions and form the torus and turn on a *Simple* Automaton on the Mask. Then, for every three moves of a *Simple* Automaton on the time scale, we can find L (total cell number) multiplied by 2 states in each of our new tables $F^{(3)}_{AC}$. Finally, we stop after we reach the point of half-cycle and analyze the result. Then we change the initial conditions and repeat the trial. At each point in table $F^{(3)}_{AC}$, we can determine 12(!) new rows R_{-x} taking into account the inverse motion. If the Mask is *Correct*, then R_{-x} will be entirely filled in a short time. After that, it is necessary to perform standard check for *Correctness* (see Chapter 3).

Definition of a *Perfect* Mask Strings

So, let us repeat again. Let us take a look on two runs (we call it as Standard and Additional) of i) a *Simple* Automaton on the Mask (f_{ABC}; as a function of time *t*) and ii) *Table* Automaton on the same Mask ($F^{(3)}_{AC}$; as a function of time *τ*) induced by *Simple* Automaton. The cells we have obtained are the same and numbered with integers *r*.

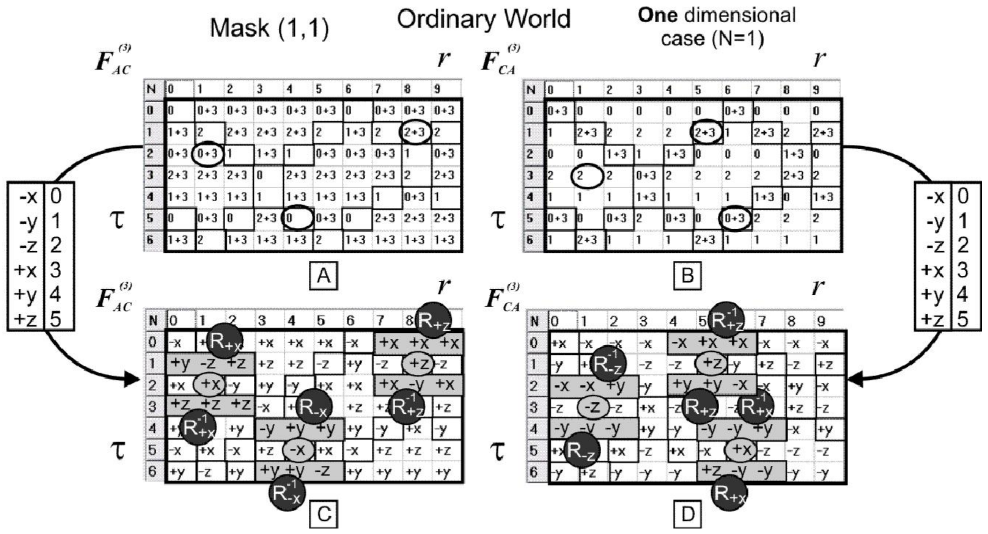


Fig. 16. Continuation of illustration shown in Fig. 14. In this figure, panels (*A*, *C*) and (*B*, *D*) show the two pairs of practically identical Tables, the only difference between the top Table and bottom one is that in the bottom one, the numbers in the cells are replaced with letters (see straightforward conversion rules depicted on the left and right sides). Our task at this point is to find NEW rows in the table R_{-x}. In each drawing (see Figs. 16*C* and *D* we selected three arbitrary points – the results of the $F^{(3}_{AC}$ and $F^{(3)}_{CA}$ functions where the input parameter is the same cell *(τ, r)* in the corresponding top panels (are shown in ovals). For three points we can determine 6 new rows in different tables *R* (the corresponding rows are allocated in a gray three-cell rectangle). In the table R_{+x} – a row *{+y,-z,+z}*; in the table R^{-1}_{+x} – a row *{+z,+z,+z}*; in the table R_{-x} – a row *{- y,+y,- y}* etc. All these rows from the inverted substitution (see Eq.5-6) are then transferred to the table R_{-x}. Further these rows are compared to the ones that are already present in R_{-x}, and if the new rows aren't there, we add those rows to the Table.

In the case of *Simple* Automaton, our run (the *Ordinary World*) starts with state filled with cells containing letters "A" and "C" at time *t=0*; Fig. 13A, 14. In the additional run (Parallel World), the replacements A⇔C take place. In the case of *Table* Automaton, our run (the *Ordinary World*) starts with state filled with *–x* (cells with letters "A" in *Simple* Automaton; *t=0*) and *+x* (cells with letters "C" in *Simple* Automaton; *t=0*). In the additional run (*Parallel World*) of the *Table* Automaton the replacements *-x* ⇔ *x* take place.

Let us introduce function Δ(τ,*r*):

$$\Delta(\tau,r)=\begin{cases}-1 & if\ F^{(3)}_{AC}(\tau,r)=-x\vee -y\vee -z\\ +1 & if\ F^{(3)}_{AC}(\tau,r)=+x\vee +y\vee +z\end{cases} \tag{13}$$

and write out two formulas (14, 15) connecting our two Automatons.

In any point *r*, parameters *t* and *τ* are connected by the relation:

$$t=\frac{3\tau}{2}+\frac{1}{2}\sum_{0\le u<\tau}^{u}\Delta(u,r) \tag{14}$$

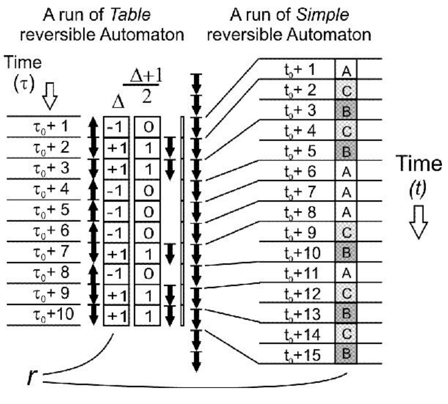


Fig. 17. Illustration of Eq. 14

So

$$f_{ABC}(t,r)=\begin{cases}A, & if\ \Delta(\tau,r)=-1\\ C, & if\ \Delta(\tau,r)=+1\end{cases} \tag{15}$$

Let us turn to the Parallel World (the functions that correspond to that World are shown with the asterisk), and let us introduce an simple lemma...

Lemma 1. Let us designate substitution $e_3=\{+x,+z,+y,-x,-y,-z\}$ as S. Let us designate the state of Lattice Z^N at time τ as L_τ with made substitution S as $S(L_\tau)$. So, for any *Correct* Mask if L_τ transfer to $L_{\tau+1}$ then $S(L_\tau)$ transfer to $S(L_{\tau+1})$.

Proof. We take our *Transition Table* $R^{(6)}$ and show that for any row k from $R^{(6)}$ the following is correct : if k belongs to R_u and $S(k)$ belongs to R_v, then $u=S(v)$.

Let's take $u=-x$. The row k belongs to R_{-x}. The substitution for R_{-x} equals e. From Fig. 8 we have: $S*e=e_3*e=e_3$ so $v=+x$ and $-x=S(+x)$. Suppose that $u=-y$. The row k belongs to R_{-y}. From Fig. 8 $S*e_1= e_3*e_1=e_5$ so $v=+z$ and $-y=S(+z)$. The remaining 4 letters are checked in the same way. So we declare that the lemma is proven. Fig. 18A illustrates formula $u=S(v)$ for two random pairs of rows in *Transition Tables* for Mask (1,1).

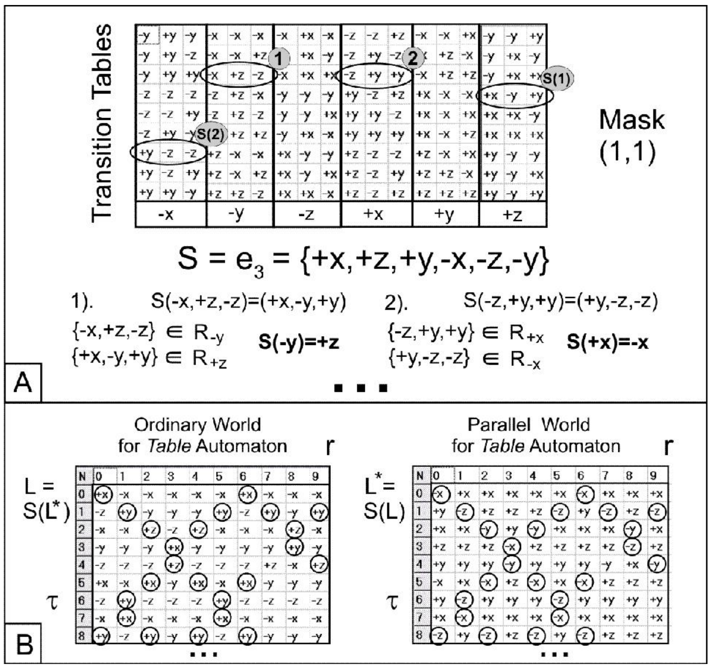


Fig. 18*A* – illustration of the formula *u=S (v)* for three rows (*{-x, + z, -z}, {-z, + y, + y}, {-y, -y, -y}*) of the *Transition Tables* for Mask (1,1) . Panel *B* illustrates the operation of Eq. 16 and Consequence 1 for our case depicted in Fig. 14

So, we have

Consequence 1. So, for any *Correct* Mask (the Ordinary World) = S(the Parallel World) and (the Parallel World) = S(the Ordinary World) for *Table* Automatons.

Proof. It is obvious that $L_{\tau=0}=S(L^*_{\tau=0})$. Using the induction we get that for all elements of lattice L the following is carried out: $L_\tau=S(L^*_\tau)$ for any τ. See Fig. 18B.

Consequence 2. So, for any τ and r one can write

$$\Delta(\tau,r)=-\Delta^*(\tau,r) \tag{16}$$

We will enter the designation

$$\sum_{0\le u<\tau}^{u} \Delta(u,r) = -\sum_{0\le u<\tau}^{u} \Delta^*(u,r) = 2d_{1/2} \qquad (17)$$

Our final observation is that: if all this is correct for the *Ordinary World* then, one can write that for the *Parallel Word* and for all τ и *r* the following is correct

$$if\ f_{ABC}\left(\frac{3\tau}{2}+d_{1/2},r\right)=A \ \Rightarrow\ f^*_{ABC}\left(\frac{3\tau}{2}-d_{1/2},r\right)=C$$
$$if\ f_{ABC}\left(\frac{3\tau}{2}+d_{1/2},r\right)=C \ \Rightarrow\ f^*_{ABC}\left(\frac{3\tau}{2}-d_{1/2},r\right)=A \qquad (18)$$

Let us talk about the Strings. What do we mean by Strings?

Our Strings are constructed using two halves: one is taken from the Ordinary World and another – from the Parallel One on the plot "Automaton state vs Time" at some given point of time τ. Below we show our String using a simple example (Fig. 19).

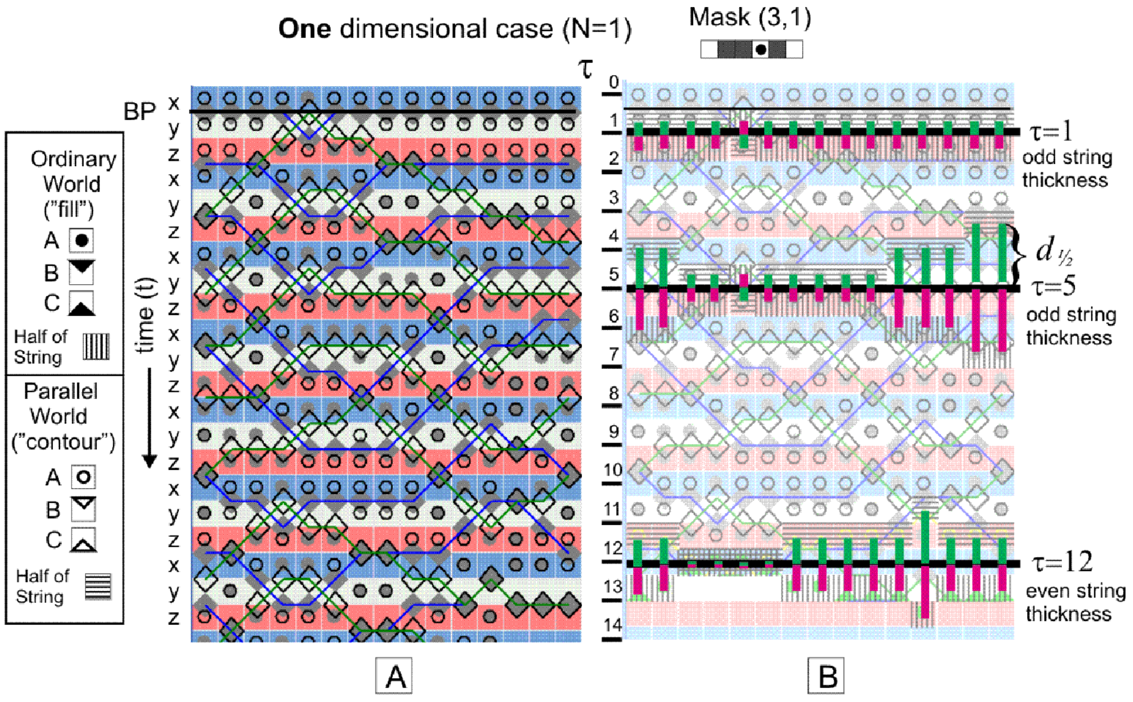


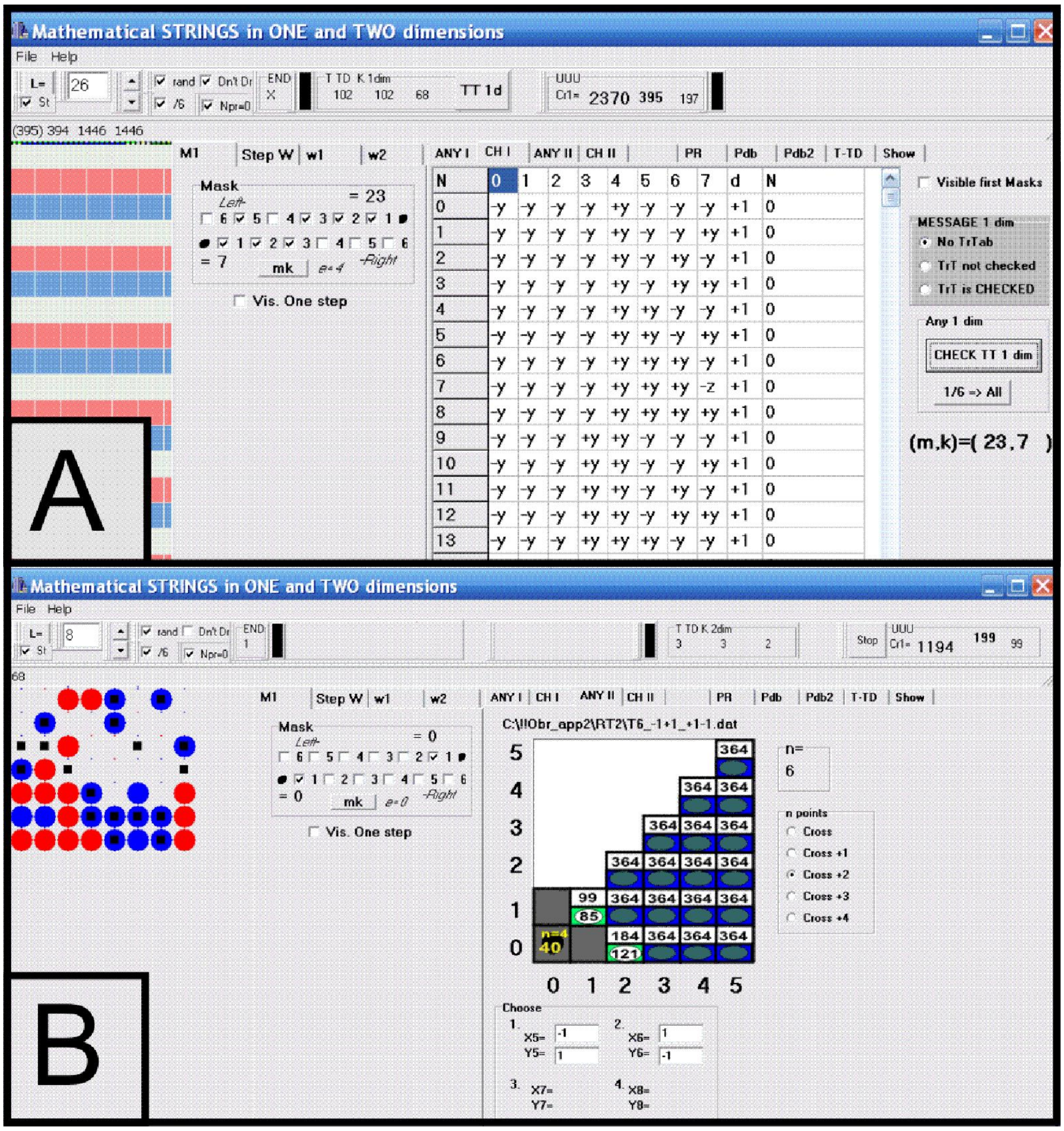
Fig. 19. Panel *A* shows the two superimposed runs for Mask (3, 1). It seems to be chaotic at the first glance. Panel *B*: vertical bars designate halves of Strings $d_{1/2}$ (see Eq. 17) from the *Ordinary World* (pink bars) and *Parallel* one (green bars). It is not easy to recognize the String in the picture taken as a whole, but it is there. The situation is very unusual. Both Automatons work completely independently, yet the two runs are connected through some sort of "wormhole"

Let us show in the same drawing each of two one-dimensional runs of *Simple* Automaton for Masks (3,1). The length of row (closed into a ring) is 15 cells. Initial conditions are as follows: one point of "opposite" color (in *Ordinary World*: all cells C, one cell is A; in *Parallel World*: all cells A, one cell is C). Moreover, we use line of different colors to depict C/B «bands» for both runs: the green line indicate the C/B band in the *Ordinary World* while the blue line shows the C/B band in the *Parallel World*. Horizontal strips shown in panel *A* of Fig. 19: x (light-blue) designate $t \bmod 3 = 0$, band y (white) $t \bmod 3 = 1$, and z (pink) $t \bmod 3 = 2$, where t is time (see above).

So, there goes a reference run presenting a single step of *Table* CA. Over the course of this step, one and a half steps of *Simple CA* occur. The dashed shaded horizontal and vertical cells – representing a "half" of the Strings from the *Ordinary World* and the *Parallel* one – move down one cell (if there is no cell B beneath these cells) or two cells (if there is cell B beneath these cell, then one needs to jump over it). At every step and in every cell, the following event happens: either *Ordinary World* can go ahead of the reference (and then the *Parallel World* stays behind by the same amount) or lag (and then Parallel World is ahead). Parity of the String width ($d=2*d_{1/2}$; the distance between the differently shaded cells) in each time point of τ coincides with the parity of the number τ itself (see Fig. 19B). At the Half Period point, Strings are stretched into a line with width $\lambda = T - T^*$ from Ref

[http://arxiv.org/abs/1308.0136]. (Note that if we had taken *normal* but *Incorrect* Mask (5,1) instead of Mask (3,1), we would have obtained no *Strings* at all. Our beautiful picture from Fig. 19 would have turned into a mess at once).
Coming back to that chess figure from Section 4 which carries out converting ***the N dimensions of integer lattice into a single one***, the author suggests that whatever the method of transformation is chosen, the String will keep the same ***appearance***: i.e., the distance (taken at modulus) between the adjacent cells (in one measurement) for each half of the String will never be more than one. In this case we will say that our String is continuous.

Definition 2. (Definition of a *Perfect* Mask). Assume that i) Eqs. 15, 18 are satisfied for some *Correct* Mask and for every run of its *Simple* Automaton and ii) the String keeps its continuity whatever the method of transformation is chosen, then the Mask is *Perfect* (otherwise – *Imperfect*). In Section 4, we did not use the notion of Parallel World anywhere. In principle, the *Correctness* of Masks could exist without existence of Eqs. 15, 18 and continuity of the Strings. But, it is obvious that these two things are connected directly. If the Mask is *Correct*, that means there Eqs.15,18 exists and the Strings are continuous; if the Mask is *Incorrect* then Eqs. 15, 18 do not exist and thus the "right" Strings do not exist either. Naturally, the fact that the Mask is *Perfect* leads to a considerable reduction in the number of possible "ways" (rows of the states, parameter ρ, Fig. 9) and as a consequence, to a significant reduction (in a statistical sense) in the values of both half periods *T* and *T** (see Ref http://arxiv.org/abs/1109.4552.).



N	0	1	2	3	4	5	6	7	d	N
0	-y	-y	-y	-y	+y	-y	-y	-y	+1	0
1	-y	-y	-y	-y	+y	-y	-y	+y	+1	0
2	-y	-y	-y	-y	+y	-y	+y	-y	+1	0
3	-y	-y	-y	-y	+y	-y	+y	+y	+1	0
4	-y	-y	-y	-y	+y	+y	-y	-y	+1	0
5	-y	-y	-y	-y	+y	+y	-y	+y	+1	0
6	-y	-y	-y	-y	+y	+y	+y	-y	+1	0
7	-y	-y	-y	-y	+y	+y	+y	-z	+1	0
8	-y	-y	-y	-y	+y	+y	+y	+y	+1	0
9	-y	-y	-y	+y	+y	-y	-y	-y	+1	0
10	-y	-y	-y	+y	+y	-y	-y	+y	+1	0
11	-y	-y	-y	+y	+y	-y	+y	-y	+1	0
12	-y	-y	-y	+y	+y	-y	+y	+y	+1	0
13	-y	-y	-y	+y	+y	+y	-y	-y	+1	0

	0	1	2	3	4	5
5						364
4					364	364
3				364	364	364
2			364	364	364	364
1		99 85	364	364	364	364
0	n=4 40		184 121	364	364	364

Fig. 20. Two screenshots of the program proofT1.exe. Determination of the *Correctness* of the Mask (23,7) in one dimension (panel *A*). The definition of the *Transition Table* for one of the *Correct* Masks from Fig. 9B, two-dimensional case (panel *B*)

Immediately, there arises the Main problem (more correctly, a lot of problems). We denote it as the "*Problem of the Number 3*". We do not know, yet, how broad is the problem. It is quite possible that it can be a very interesting problem in a broader mathematical sense. Assume we know that the Mask under consideration is *Correct.* More accurately, assume that we have found the *Transition Tables* with the aid of personal computer and proved by exhaustion that the given Mask is *Correct*. Now, we need to prove that the same very Mask is *Perfect*. Here arises a fundamental question of "why the proof of Theorem 1 always works". In some sense, it can be the case that we only have found a new and "better looking" wrap for the *Correct* Masks. How can we solve this problem? Well, it is necessary to investigate algebraically our Strings in *N* dimensions. Obviously, it is quite a challenging task. How many "problems" do we have? In other words, how large is the number of all *Correct* Masks that we can identify? We introduce, informally, certain number *K* which is the approximate number of Masks in all dimensions, for which the *Jaguar* – the most powerful supercomputer on this day (February 2016) – can prove the *Correctness* of all the Masks in one day or maximum a week. (This definition, obviously, is necessary to correct somehow. It is "well known", for example, that in one dimension, all Masks (-2^k+1, 2^k+1) are *Correct* and *c* (= 81) for those Masks is not very large. We, of course, do not consider such Masks!) We do not know how large is *K*. It can be equal to 10,000 or may be equal to a factorial of 100. No doubts that the Jaguar is a powerful computer. On the other hand, the number of inspections grows very rapidly with the increase of *n* and *c*. However, the inspections use the algebraic operations/steps of a similar type. So perhaps it is better to do the calculations using a specialized computer. Verifying the *Correctness* of a particular Mask can soon become a boring task (in that case that all the Masks, one after another, turn to be *Correct*), so we will focus only on a calculation of *c* for Masks but in larger dimensions and for larger *n*. Yet, to do this "truncated" task, one also requires an aid of a powerful computer. In view of that, let us introduce another number K_{cor}. This is the approximate number of Masks for which the most powerful supercomputer can correctly determine the value *c* after a week of operation under condition that the considered Masks are *Correct* at 95% confidence (estimated from other considerations). It is clear that $K_{cor} >> K$. As of today, the general program has been written to prove the *Correctness* of any Mask in one and two dimensions. Site http://k3e.hop.ru/proofT1.zip, shows the proofs of the *Correctness* for all of Masks from Fig. 9 with $n<9$. The breadth of the proof at this point is limited by a performance of author's personal computer. In brief, *Mathematical Strings* definitely exist and in huge numbers. The author still does not know the answer to the question: whether or not *all central-symmetric(!)* Masks are *Correct* Masks in all dimensions. The author employed the personal computer for weeks searching for abnormalities. The dimensions 1, 2, 3 have been checked. The criterion of *Incorrectness* was getting 2 in the expression $\theta_{AC}+\theta_{AC}*$ (see Fig. 14). Yet, not a single *Incorrect* Mask among *central-symmetric* ones has been found. To make the situation more intriguing, we note that there exist special *Incorrect* but *"self-healing"* Masks-Strings (Kornyushkin, 2013.) We cannot exclude a situation that starting with some *n* and *N* our proof stops working. Yet, the author tends to believe that a probability of such an outcome is extremely low.We suggest that all the material presented in the above to be named as the foundation of Theory of *Mathematical Strings*. Note: our theory should not be confused it with a well-known theory of strings in physics.

Conclusion

Let us briefly summarize our findings.

The Number 3 (in other words, a group from Fig. 8, or configuration 2x3) generates the unimaginable number of different symmetries in the *N*-dimensional integer lattice Z^N. Here we remind that the number of symmetries is greater than the number of *central-symmetric* Masks in the *N*-dimensional integer lattice Z^N. Computer calculations indicate (yet, indirectly) that in Z^N with large *N* "almost any" Mask is *Correct*, and each *Correct* Mask represents a new "symmetry" in this Z^N. The author developed the approach to finding the *Correct* Masks for low-dimensional Z^N. Utilizing the same approach (yet exploiting capabilities of a very powerful computer) it is feasible to find the *Correct* Masks for integer lattices of higher dimensions. It seems certain that these symmetries are directly related to the physical nature of space.We do this to suggest that professionals in the field of theory of numbers should take over the theoretical physicists concerning the question of constructing a consistent theory of the Universe. Yet, while physicists argue and debate about various peculiarities of string theory, one can contemplate how to use the theory of Mathematical Strings to build a consistent model describing fundamental properties of space.

Acknowledgements

The author thanks Drs. Eugene Moskovets and Maxim Olshanii for reading and correcting this manuscript.

REFERENCES

Kornyushkin. A. 2013. The X-problem of number 3. http://arxiv.org/abs/1308.0136

Kornyushkin. A. 2011. *About a Discrete Cellular Soliton (computer simulation).* http://arxiv.org/abs/1109.4552.

Kornyushkin. A. 2013. *About New Mathematics: Automaton of pure number three.* (О новой математике: Автомате Чистой Тройки). LAP Lambert Academic Publishing - ISBN: 978-3-659-33017-9.

Margolus, Norman 1984. "*Physics-like models of computation*", Physica D: Nonlinear Phenomena 10: 81–95, Reprinted in Wolfram, Stephen, 1986. *Theory and Applications of Cellular Automaton*, Advanced Series on Complex Systems 1, World Scientific, pp. 232–246; A. Adamatzky (Ed.) (2002) *Collision-Based Computing*, Springer-Verlag, , pp. 83–104.

McIntosh, H. V. 2009. "*One Dimensional Cellular Automaton*", Luniver Press, pp. 205–246.

Toffoli T., Margolus N. 1987. *Cellular Automaton Machines: A New Environment for Modeling,* MIT Press Series, Section 14.2, "*Second-order technique"*, pp. 147–149; Wolfram, Stephen (2002), *A New Kind of Science*, Wolfram Media, pp. 437.;

Vichniac, G. (1984) "*Simulating physics with cellular Automaton*", Physica D: Nonlinear Phenomena 10: 96–115.

Wolfram, S. 1983. «Statistical mechanics of cellular Automaton». *Rev. Mod. Phys.* 55 (3): 601–644. DOI:10.1103/RevModPhys.55.601
Wolfram, Stephen. 1984. "*Cellular Automaton as models of complexity*", Nature 311 (5985): 419–424.

Appendix 1.

Designation	Variable type	Key value found in the article
N	Natural	The dimension of the Lattice Z
N	Natural	Number of points in the Mask minus 1; the number of columns in the *Transition Tables* minus 1 (Zero point included in the Mask by definition)
c	Natural, odd	The number of rows in a single *Transitions Table*. Since c values are always odd, we use another designation $c_{1/2} = (c-1)/2$
h	Natural+"0"	The column number in the *Transition Table* responsible for the Zero point of the Mask
ρ	Real, (%)	$\rho = 100*c/6^n$; "density" of the corresponding *Table* CA
T, T^*	Natural	Half periods of the return of the *Simple* CA: for usual run – T; for additional run – T^*
Θ	Natural	$\Theta = (T+T^*)/3$; half period of the return of the *Table* CA
λ	Integer	$\lambda = T-T^*$; fixed width of a String at the moment of time corresponding to a half period
d	Integer	String width
Δ	(-1/+1)	The decrement/increment of the String width determined for every point (cell) of the Lattice and in any time τ
m	Natural, odd	The left side of the one-dimensional Masks (m, k)
k	Natural, odd	The right side of the one-dimensional Masks (m, k)
K	Natural, big	Approximate number of Masks for which the most modern supercomputer can prove its *Correctness* after a week of operation
K_{cor}	Natural, very big	Approximate number of Masks for which the most modern supercomputer can correctly determine the value c after a week of operation, under condition that the considered Masks are *Correct* with 95% confidence (estimated from other considerations)

Appendix 2.

Example of a real code for program "proofT1.exe". The central place in the program is the check whether the Mask is Correct. Programming language is C++. Comments have been removed to save the space.

```
GRAN=2*sm00[8]+1; K162=Cr1; K117=Cr1/6;
mK162=Cr1; if(CheckBox68->Checked) mK162=mK162/6;

for(lll1=0; lll1<K162; lll1++) VES[lll1]=0;
//__________________________ for(lll1=0; lll1<K162; lll1++)
for(lll1=0; lll1<mK162; lll1++)
{
 for(klll9=0; klll9< GRAN; klll9++)
 ssll[klll9][0]=-1;
 for(klll9=0; klll9<9; klll9++)
 ssll[sm00[klll9]][0]=rtrt3[lll1][klll9];
//__________________________ for(lll2=0; lll2<K162; lll2++)
 for(lll2=0; lll2<K162; lll2++)
 {
  for(klll9=0; klll9< GRAN; klll9++)
  ssll[klll9][1]=ssll[klll9][0];
 for(klll9=0; klll9<9; klll9++)
  {
   smPR=sm00[1]+sm00[klll9];
   if(ssll[smPR][1]!=-1)
   {
   if(ssll[smPR][1]!=rtrt3[lll2][klll9]) goto nnhh2;
   }
   else
   ssll[smPR][1]=rtrt3[lll2][klll9];
 }
//_________________________________  for(lll3=0; lll3<K162; lll3++)
  for(lll3=0; lll3<K162; lll3++)

  {
strcpy(ccdd,"("); ltoa(mK162,strin,10); strcat(ccdd,strin); strcat(ccdd,") "); ltoa(lll1,strin,10); strcat(ccdd,strin); strcat(ccdd,"  ");
ltoa(lll2,strin,10); strcat(ccdd,strin); strcat(ccdd,"  "); ltoa(lll2,strin,10); strcat(ccdd,strin); strcat(ccdd,"  ");
 StatusBar1 -> SimpleText = ccdd;

   for(klll9=0; klll9< GRAN; klll9++)  ssll[klll9][2]=ssll[klll9][1];
   for(klll9=0; klll9<9; klll9++)
   {
   smPR=sm00[2]+sm00[klll9];
    if(ssll[smPR][2]!=-1)
    {
    if(ssll[smPR][2]!=rtrt3[lll3][klll9]) goto nnhh3;
    }
    else
    ssll[smPR][2]=rtrt3[lll3][klll9];
   }
//_________________________________  for(lll4=0; lll4<K162; lll4++)
   for(lll4=0; lll4<K162; lll4++)
   {

  for(klll9=0; klll9< GRAN; klll9++)  ssll[klll9][3]=ssll[klll9][2];
 for(klll9=0; klll9<9; klll9++)
 {
   smPR=sm00[3]+sm00[klll9];
  if(ssll[smPR][3]!=-1)
  {
  if(ssll[smPR][3]!=rtrt3[lll4][klll9]) goto nnhh4;
  }
  else
  ssll[smPR][3]=rtrt3[lll4][klll9];
 }
//_________________________________  for(lll5=0; lll5<K162; lll5++)
   for(lll5=0; lll5<K162; lll5++)
   {

   for(klll9=0; klll9< GRAN; klll9++) ssll[klll9][4]=ssll[klll9][3];
 for(klll9=0; klll9<9; klll9++)
 {
  smPR=sm00[4]+sm00[klll9];
  if(ssll[smPR][4]!=-1)
  {
  if(ssll[smPR][4]!=rtrt3[lll5][klll9]) goto nnhh5;
  }
  else
  ssll[smPR][4]=rtrt3[lll5][klll9];
 }
//_________________________________  for(lll6=0; lll6<K162; lll6++)
   for(lll6=0; lll6<K162; lll6++)
   {

   for(klll9=0; klll9< GRAN; klll9++)  ssll[klll9][5]=ssll[klll9][4];
 for(klll9=0; klll9<9; klll9++)
 {
  smPR=sm00[5]+sm00[klll9];
  if(ssll[smPR][5]!=-1)
  {
  if(ssll[smPR][5]!=rtrt3[lll6][klll9]) goto nnhh6;
  }
  else
  ssll[smPR][5]=rtrt3[lll6][klll9];
 }
//_________________________________  for(lll7=0; lll7<K162; lll7++)
    for(lll7=0; lll7<K162; lll7++)
    {
    for(klll9=0; klll9< GRAN; klll9++) ssll[klll9][6]=ssll[klll9][5];
  for(klll9=0; klll9<9; klll9++)
  {
   smPR=sm00[6]+sm00[klll9];
   if(ssll[smPR][6]!=-1)
   {
   if(ssll[smPR][6]!=rtrt3[lll7][klll9]) goto nnhh7;
   }
   else
   ssll[smPR][6]=rtrt3[lll7][klll9];
  }
//_________________________________  for(lll8=0; lll8<K162; lll8++)
     for(lll8=0; lll8<K162; lll8++)
     {

     for(klll9=0; klll9< GRAN; klll9++) ssll[klll9][7]=ssll[klll9][6];

   for(klll9=0; klll9<9; klll9++)
   {
    smPR=sm00[7]+sm00[klll9];
    if(ssll[smPR][7]!=-1)
    {
    if(ssll[smPR][7]!=rtrt3[lll8][klll9]) goto nnhh8;
    }
    else
    ssll[smPR][7]=rtrt3[lll8][klll9];
   }
//___________________________________ for(lll9=0; lll9<K162; lll9++)
     for(lll9=0; lll9<K162; lll9++)
     {
     for(klll9=0; klll9< GRAN; klll9++) ssll[klll9][8]=ssll[klll9][7];
   for(klll9=0; klll9<9; klll9++)
   {
    smPR=sm00[8]+sm00[klll9];

    if(ssll[smPR][8]!=-1)
    {
    if(ssll[smPR][8]!=rtrt3[lll9][klll9]) goto nnhh9;
    }
    else
    ssll[smPR][8]=rtrt3[lll9][klll9];
   }
//___________________________________ VSTA

rrNADx[0]=(long)lll1/K117; rrNADx[1]=(long)lll2/K117; rrNADx[2]=(long)lll3/K117; rrNADx[3]=(long)lll4/K117;
rrNADx[4]=(long)lll5/K117; rrNADx[5]=(long)lll6/K117; rrNADx[6]=(long)lll7/K117; rrNADx[7]=(long)lll8/K117;
rrNADx[8]=(long)lll9/K117;

EstNet=0;
 for(lll=0; lll<K162; lll++)
 {
  if
  (
  (rrNADx[0]==rtrt3[lll][0])&& (rrNADx[1]==rtrt3[lll][1])&& (rrNADx[2]==rtrt3[lll][2])&& (rrNADx[3]==rtrt3[lll][3])&&
  (rrNADx[4]==rtrt3[lll][4])&& (rrNADx[5]==rtrt3[lll][5])&& (rrNADx[6]==rtrt3[lll][6])&& (rrNADx[7]==rtrt3[lll][7])&&
  (rrNADx[8]==rtrt3[lll][8])
  )
  {
   NMtest=lll;
   VES[lll]=VES[lll]+1;
   EstNet=1;
   break;
  }
 }

if(!EstNet) { Application->MessageBox("Error !!!!!!!!!!!!!!","Warning",MB_OK);
goto nnhh;
}

PrNprOb=0;
 for(lll=0; lll<K162; lll++)
 {
  if
  (
  (rrNADx[0]==rtrt3OB[lll][0])&&  (rrNADx[1]==rtrt3OB[lll][1])&&  (rrNADx[2]==rtrt3OB[lll][2])&& (rrNADx[3]==rtrt3OB[lll][3])&&
  (rrNADx[4]==rtrt3OB[lll][4])&&  (rrNADx[5]==rtrt3OB[lll][5])&&  (rrNADx[6]==rtrt3OB[lll][6])&&  (rrNADx[7]==rtrt3OB[lll][7])&&
  (rrNADx[8]==rtrt3OB[lll][8])
  )
  {
   if(lll/K117==ssll[SS0][8])
   {PrNprOb=1;
   }
   break;
  }
 }

if(!PrNprOb)
{Application->MessageBox("N0 R-1 !!!!!!!!!!!!!!!!!!!","Warning",MB_OK);
 goto nnhh;
       }
       //--------------------------- VSTA

       nnhh9:
          }
       //--------------------------------  for(lll9=0; lll9<K162; lll9++)
       nnhh8:
          }
       //--------------------------------  for(lll8=0; lll8<K162; lll8++)
       nnhh7:
          }
       //--------------------------------  for(lll7=0; lll7<K162; lll7++)
       nnhh6:
         }
       //--------------------------------  for(lll6=0; lll6<K162; lll6++)
       nnhh5:
         }
       //--------------------------- for(lll5=0; lll5<K162; lll5++)
       nnhh4:
         }
       //--------------------------  for(lll4=0; lll4<K162; lll4++)
```

```
nnhh3:
  }
//---------------------------  for(lll3=0; lll3<1V62; lll3++)
nnhh2:
 }
//-------------------------- for(lll2=0; lll2<K162; lll2++)
}
//-------------------------- for(lll1=0; lll1<K162; lll1++)
nnhh:
```

****