On the Iota-Delta Function: Universality in Cellular Automata's Representation

Luan C. de S. M. Ozelim[*]
André L. B. Cavalcante[†]
Lucas P. de F. Borges[‡]
Department of Civil and Environmental Engineering, University of Brasília
Brasília, DF, 70910-900, Brazil
[*]*luanoz@gmail.com*
[†]*abrasil@unb.br. Full address for mailing: Geotecnia - Departamento de Engenharia Civil e Ambiental/FT - UnB, 70910-900 - Brasília/DF/Brazil*
[‡]*lucaspdfborges@gmail.com*

Universality has always played a major role in every branch of science. Since the advent of cellular automata (CAs), this type of model has been widely applicable to the modeling of physical phenomena. On the other hand, the way the evolution rules were described lacked a unified formulation in terms of mathematical functions. In the present paper, a general formulation that is able to describe every elementary CA is derived. The new representation is given in terms of a new function hereby defined: the iota-delta function.

1. Introduction

Be it while studying nature's behavior or analyzing the implementability and efficiency of computational algorithms, scholars of the most diverse areas of human knowledge seek universality. Even though each area has particular concepts of universality, the main concern behind this idea is whether a given system or rule can reproduce/mimic others of the same kind or even different kinds but also be universal. A few definitions from different areas of science are briefly shown below.

In theoretical physics, universality is often related to background independence [1]. An intrinsic robustness is inherent to physical universal systems as they may show a given expected behavior even though other phenomena are concomitantly taking place.

Different from physics, in computer science, universality is discussed in terms of concepts such as Turing completeness and equivalence [2]. In general, universality is a property assigned to a system. For example, there are universal Turing machines, tag systems, and cellular automata (CAs) [2]. Due to the inherent temporal evolutive characteristic of these systems, they are named dynamical systems. In

https://doi.org/10.25088/ComplexSystems.21.4.283

short, the latter consists of a state space S_S and an evolution rule E_R. At each time step t, a system's state is updated by means of its evolution rule.

In mathematics, on the other hand, universality can be assigned to functions. One of the most famous mathematical functions, the Riemann zeta function $\zeta(s)$ is universal by means of the Voronin universality theorem [3]. The latter briefly states that, under certain conditions, any nonvanishing analytic function can be arbitrarily well approximated by $\zeta(s)$.

By establishing a parallel between mathematical and computational systems, the main issue to be addressed by this paper is whether universality can be assigned to evolution rules of dynamical systems. In other words, let such rules be functions whose domain is the state space; thus, the main concern is if there is any general evolution rule that enables every dynamical system to be emulated.

It will be shown that a universal evolution rule does exist and is given in terms of a new function hereby defined: the iota-delta function. At first, it is shown that the iota-delta function can describe the evolution rules of every elementary CA. Finally, since the universal rule 110 [2] is described by means of the Church–Turing thesis [2], every Turing-equivalent dynamical system has its evolution rule expressible in terms of the iota-delta function.

2. Evolution Rules of Elementary Cellular Automata

In the present paper, special attention is given to the evolution rules of CAs because it is interesting to define this class of dynamical systems. First, it is interesting to study elementary CAs. Also, a very important concept has to be introduced: the CA mesh (Figure 1).

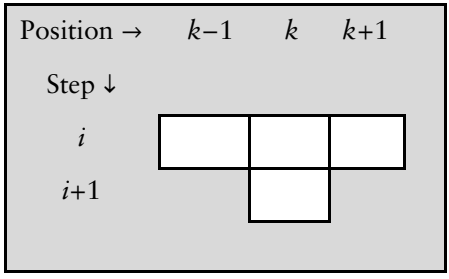

Figure 1. CA mesh.

A CA mesh is made of cells C, hereby indexed in space by the subscript k and time by the superscript i, as in C_k^i. Following [2, 4], an

https://doi.org/10.25088/ComplexSystems.21.4.283

elementary CA is a dynamical system that consists of an infinite row of cells whose values are either 0 or 1. At each time step, a set of rules is applied to the CA net in order to update the cells' values. While dealing with elementary CAs, the value of a given cell depends on the values of the cell itself and its immediate neighbors on the previous time step. Mathematically, this definition turns to

$$C_k^{i+1} = f\left[C_{k-1}^i, C_k^i, C_{k+1}^i\right]. \tag{1}$$

The binary values 0 and 1 are commonly interpreted as the colors white and black, respectively. In equation (1), no direct information is given about the evolution rule f. The latter is given individually for each of the 0–255 elementary CAs. For instance, rule 90 has the following evolution rule:

$$C_k^{i+1} = \begin{cases} 0, & \text{if } \left[C_{k-1}^i, C_k^i, C_{k+1}^i\right] = [1, 1, 1] \\ 1, & \text{if } \left[C_{k-1}^i, C_k^i, C_{k+1}^i\right] = [1, 1, 0] \\ 0, & \text{if } \left[C_{k-1}^i, C_k^i, C_{k+1}^i\right] = [1, 0, 1] \\ 1, & \text{if } \left[C_{k-1}^i, C_k^i, C_{k+1}^i\right] = [1, 0, 0] \\ 1, & \text{if } \left[C_{k-1}^i, C_k^i, C_{k+1}^i\right] = [0, 1, 1] \\ 0, & \text{if } \left[C_{k-1}^i, C_k^i, C_{k+1}^i\right] = [0, 1, 0] \\ 1, & \text{if } \left[C_{k-1}^i, C_k^i, C_{k+1}^i\right] = [0, 0, 1] \\ 0, & \text{if } \left[C_{k-1}^i, C_k^i, C_{k+1}^i\right] = [0, 0, 0]. \end{cases} \tag{2}$$

One of the main features of CAs is their capability of describing complex behavior by means of simple rules. A quick analysis of equation (2) reveals a conditional definition of the evolution rule. Even though the rule itself is simple, this type of definition does not provide a straightforward representation of the evolution, requiring tedious "if" structures in the programming process of CAs.

In [4], there are different ways of representing rules like the one in equation (2). Also in [4], the possibility of interpreting CA rules as formulas is deeply discussed. Specifically, two main approaches are considered: describing CA rules by means of Boolean expressions and by means of algebraic expressions.

For rule 90, the Boolean correspondent of equation (2) can be given as [4]:

$$C_k^{i+1} = \mathrm{Xor}\left[C_{k-1}^i, C_{k+1}^i\right]. \tag{3}$$

https://doi.org/10.25088/ComplexSystems.21.4.283

On the other hand, an algebraic equivalent of both equations (2) and (3) is easily given as [4]:

$$C_k^{i+1} = \mathrm{mod}\left[C_{k-1}^i + C_{k+1}^i, 2\right], \tag{4}$$

where mod[o, p] denotes the modulus operator, which gives the rest of the division of o by p if o is greater than p or o itself, otherwise. In general, p is called the congruence modulus.

In [4], a list giving the Boolean expression for every elementary CA has been given. However, only a few rules have been algebraically described. In the present paper, equation (4) will be further investigated in order to evaluate the role of the modulus operator in the description of elementary CAs.

3. A Transformation that Relates Elementary Cellular Automata to Modular Arithmetic: The Iota-Delta Function

While observing equation (4), an immediate general rule that would give the representation of other CAs in terms of the modulus operation would be:

$$C_k^{i+1} = \mathrm{mod}\left[\alpha_1 C_{k-1}^i + \alpha_2 C_k^i + \alpha_3 C_{k+1}^i, 2\right], \tag{5}$$

in which α_j are integer coefficients less than the congruence modulus. In this particular case, $\alpha_j = \{0, 1\}$. The application of equation (5) generates eight different rules, which are summarized in Table 1.

Coefficients			Rule Number
α_1	α_2	α_3	↓
0	0	0	0
1	1	0	60
1	0	1	90
0	1	1	102
1	1	1	150
0	0	1	170
0	1	0	204
1	0	0	240

Table 1. Rules described by the application of equation (5).

By inspecting Table 1, not only rule 90 but also seven other cases are simply defined by means of a single rule applied to the whole CA net.

https://doi.org/10.25088/ComplexSystems.21.4.283

It is interesting that the formula in equation (5) will never describe odd rules since in Wolfram's numeration system, the coefficient that multiplies 1 in the binary decomposition of the rule number commands parity. The latter is the result of the combination of three zeros, that is, when $\left[C^i_{k-1}, C^i_k, C^i_{k+1}\right] = [0, 0, 0]$. This way, when applying equation (5) to any combination of three zeros, the null value is obtained, which implies only even rule numbers. In order to overcome this issue, a fourth coefficient α_4 needs to be inserted inside the modulus operator in equation (5), generating

$$C^{i+1}_k = \mathrm{mod}\left[\alpha_1\, C^i_{k-1} + \alpha_2\, C^i_k + \alpha_3\, C^i_{k+1} + \alpha_4,\, 2\right]. \tag{6}$$

The new coefficient, like the other ones, is an integer less than the congruence modulus. This way, the number of rules described by the application of equation (6) is 16, double that obtained by applying equation (5). Table 2 summarizes the rule numbers obtained by the addition of the fourth coefficient.

Coefficients				Rule Number
α_1	α_2	α_3	α_4	↓
0	0	0	0	0
1	0	0	1	15
0	1	0	1	51
1	1	0	0	60
0	0	1	1	85
1	0	1	0	90
0	1	1	0	102
1	1	1	1	105
1	1	1	0	150
0	1	1	1	153
1	0	1	1	165
0	0	1	0	170
1	1	0	1	195
0	1	0	0	204
1	0	0	0	240
0	0	0	1	255

Table 2. Rules described by the application of equation (6).

The application of modular arithmetic perfectly fits the need for simplicity in the description of CAs. On the other hand, the number of automata generated by equation (6) is 1 / 16 of the total number of

https://doi.org/10.25088/ComplexSystems.21.4.283

binary automata—also called simple 1D automata in [4]. The number of automata generated is directly related to how many values the coefficients can assume. While investigating equations (5) and (6), if the congruence modulus is 2, the numbers 0 and 1 are the only possible coefficients. Thus, there has to be a way to generate more combinations of coefficients.

When congruence modulus n is considered, a total of n^4 combinations of four α coefficients is obtained. The possible coefficients are described as:

$$\alpha_j = \{r \,|\, r \leq n-1;\ r \in \mathbb{Z}_+\}; \qquad j = 1, 2, 3, 4. \tag{7}$$

This way, when congruence modulus 3 is taken into account, the possible coefficients are 0, 1, and 2, which ultimately generate a total of 3^4 combinations. However, considering binary automata, a modulus operator with congruence modulus 3 cannot be applied alone as the outcomes of such an operation are not only 0 and 1, but also 2. It is not possible to describe binary automata by a rule of the form

$$C_k^{i+1} = \mathrm{mod}\big[\alpha_1\, C_{k-1}^i + \alpha_2\, C_k^i + \alpha_3\, C_{k+1}^i + \alpha_4,\, 3\big]. \tag{8}$$

Thus, the situation is paradoxically summarized as: in order to describe more CAs by a simple rule that uses the modulus operator, the possible values of the coefficients must be increased. This growth can only be obtained by considering the modulus operator with respect to congruence moduli greater than 2. Notwithstanding, if the modulus operator is considered with respect to integer congruence moduli greater than 2, binary automata cannot be described since the outcomes of the transformation are not only 0 and 1. At this point, a very important concept has to be introduced: filtering operators. In order to preserve the number of possible coefficients obtained by considering the modulus operator with respect to greater integer congruence moduli and yet obtain only 0 and 1 as the outputs of the transformation, mod[*argument*, 2] must be applied to the right-hand side of equation (8). This process is a filtering processes in which the results from equation (8) are filtered in order to obtain binary outputs. This way, equation (8) becomes

$$C_k^{i+1} = \mathrm{mod}\Big[\mathrm{mod}\big[\alpha_1\, C_{k-1}^i + \alpha_2\, C_k^i + \alpha_3\, C_{k+1}^i + \alpha_4,\, 3\big],\, 2\Big]. \tag{9}$$

The number of combinations of the possible coefficients in equation (9) are 3^4 and the outputs of the latter equation are only 0 and 1. Pay close attention to the fact that each automaton is not generated by a single combination. Due to the cyclic property of the modulus operator, more than one combination generates the same automaton. This can be verified in Table 3, which shows all the combinations of

https://doi.org/10.25088/ComplexSystems.21.4.283

possible coefficients in equation (9) and the correspondent rules generated.

By inspecting Table 3, it can be seen that the 81 combinations generated only 53 different automata.

Coefficients				RN	Coefficients				RN	Coefficients				RN
α_1	α_2	α_3	α_4	↓	α_1	α_2	α_3	α_4	↓	α_1	α_2	α_3	α_4	↓
0	0	0	0	0	2	1	2	0	36	1	1	1	1	129
0	0	0	2	0	1	1	2	1	41	2	2	2	1	129
0	0	1	2	0	2	2	1	1	41	1	2	2	2	134
0	0	2	0	0	1	2	0	0	48	2	1	1	0	134
0	1	0	2	0	2	1	0	2	48	0	1	1	2	136
0	2	0	0	0	0	1	0	1	51	0	2	2	0	136
1	0	0	2	0	0	2	0	1	51	1	2	1	0	146
2	0	0	0	0	1	1	0	0	60	2	1	2	2	146
1	1	0	1	3	2	2	0	2	60	1	1	2	0	148
2	2	0	1	3	1	1	2	2	66	2	2	1	2	148
1	0	1	1	5	2	2	1	0	66	0	1	2	1	153
2	0	2	1	5	0	1	2	0	68	0	2	1	1	153
1	0	2	2	10	0	2	1	2	68	1	0	1	2	160
2	0	1	0	10	1	2	1	1	73	2	0	2	0	160
1	2	0	2	12	2	1	2	1	73	1	0	2	1	165
2	1	0	0	12	1	0	2	0	80	2	0	1	1	165
1	0	0	1	15	2	0	1	2	80	0	0	1	0	170
2	0	0	1	15	0	0	1	1	85	0	0	2	2	170
0	1	1	1	17	0	0	2	1	85	1	1	0	2	192
0	2	2	1	17	1	0	1	0	90	2	2	0	0	192
1	1	1	0	22	2	0	2	2	90	1	2	0	1	195
2	2	2	2	22	1	2	2	1	97	2	1	0	1	195
1	2	2	0	24	2	1	1	1	97	0	1	0	0	204
2	1	1	2	24	0	1	1	0	102	0	2	0	2	204
0	1	2	2	34	0	2	2	2	102	1	0	0	0	240
0	2	1	0	34	1	1	1	2	104	2	0	0	2	240
1	2	1	2	36	2	2	2	0	104	0	0	0	1	255

Table 3. Rules described by the application of equation (9).

The filtering process consists of a repetitive composition of the modulus operator in order to obtain more possible values of the coefficients, and yet get as the output of such a transformation the values necessary to define the automata being studied. This way, for example, in order to get ternary CAs, the last composition needs to be with

https://doi.org/10.25088/ComplexSystems.21.4.283

respect to congruence modulus 3, instead of 2 as for the binary case. Special care has to be taken while applying the filtering process. The modulus operation composition has to be taken with respect to prime numbers. The final filter—which determines the possible outputs of the transformation—must be situated regarding the prime numbers greater than such a number. By doing this, the chance of getting a multiple of the modulus of congruence is diminished.

In order to represent every binary automaton, since the combinations of the coefficients do not uniquely define each rule, a compact notation has to be introduced to better represent the filtering process. Let the iota-delta function be defined as follows:

$$\begin{aligned} &\iota\delta_n^m[x] = \\ &\quad \mathrm{mod}\big[\mathrm{mod}\big[\ldots \mathrm{mod}[\mathrm{mod}[x, p_m], p_{m-1}], \ldots, p_j\big], n\big], \\ &\qquad m \geq j;\ m, n \in \mathbb{Z}_+; x \in \mathbb{C};\ j = \pi[n] + 1, \end{aligned} \tag{10}$$

in which m and n are parameters of the iota-delta function, p_m is the m^{th} prime number, and $\pi[n]$ stands for the prime counting function that gives the number of primes less than or equal to n. Note that it is considered that $p_1 = 2$. The value of n determines how many states the automata generated have. Thus, for a binary automaton, $n = 2$; for ternary ones, $n = 3$; for quaternary ones, $n = 4$; and so on. Also, the iota-delta function is taken to be non-negative and $\max[\iota\delta_n^m[x]] \to n$ when $x \in \mathbb{R}$. A *Mathematica* code that readily implements equation (10) is:

```
iotadelta[m_,n_,x_] := Mod[Fold[Mod,x,Table[Prime[m-j],
    {j,0,m-1-PrimePi[n]}]],n]
```

Based on equation (7), the number of combinations allowed by means of the iota-delta function is p_m^4 and the possible coefficients are $\alpha_j = \{r \,|\, r \leq p_m - 1;\ r \in \mathbb{Z}_+\};\ j = 1, 2, 3, 4$.

By means of the iota-delta function, the filtering process is better represented. For example, equation (9) can be written in a compact way as:

$$C_k^{i+1} = \iota\delta_2^2\big[\alpha_1 C_{k-1}^i + \alpha_2 C_k^i + \alpha_3 C_{k+1}^i + \alpha_4\big]. \tag{11}$$

In order to represent every binary CA in the simplest way possible, it must be determined which is the smallest value of m such that for $n = 2$, every binary rule is expressed. By means of experimentation, when $m = 5$, that is, $\mathrm{mod}[\mathrm{mod}[\mathrm{mod}[\mathrm{mod}[\mathrm{mod}[x, 11], 7], 5], 3], 2]$, every binary CA is described. By means of the iota-delta function, every binary CA is represented by a single algebraic rule applied to the whole cellular net. Table 4 gives the first combination, that is, the smaller m, which generates each of Wolfram's rules 0–255.

https://doi.org/10.25088/ComplexSystems.21.4.283

RN	m	n	Coefficients				RN	m	n	Coefficients			
↓	↓	↓	α_1	α_2	α_3	α_4	↓	↓	↓	α_1	α_2	α_3	α_4
0	2	2	0	0	0	0	128	3	2	2	2	2	3
1	3	2	2	2	2	1	129	2	2	1	1	1	1
2	3	2	2	2	3	3	130	5	2	1	1	6	9
3	2	2	1	1	0	1	131	3	2	1	1	3	1
4	3	2	2	3	2	3	132	5	2	1	2	7	2
5	2	2	1	0	1	1	133	3	2	1	3	1	1
6	5	2	1	2	6	2	134	2	2	1	2	2	2
7	3	2	1	3	3	1	135	3	2	1	2	2	4
8	3	2	2	3	3	0	136	2	2	0	1	1	2
9	5	2	1	2	2	8	137	3	2	2	4	4	1
10	2	2	1	0	2	2	138	3	2	2	3	1	0
11	3	2	1	3	2	4	139	5	2	2	3	4	8
12	2	2	1	2	0	2	140	3	2	2	1	3	0
13	3	2	1	2	3	4	141	5	2	2	4	3	8
14	3	2	1	2	2	2	142	3	2	2	3	3	3
15	2	2	1	0	0	1	143	4	2	3	2	2	4
16	3	2	2	3	3	2	144	5	2	1	2	2	7
17	2	2	0	1	1	1	145	3	2	2	4	4	4
18	5	2	1	2	1	7	146	2	2	1	2	1	0
19	3	2	2	4	2	4	147	3	2	2	1	2	4
20	5	2	1	1	2	7	148	2	2	1	1	2	0
21	3	2	2	2	4	4	149	3	2	2	2	1	4
22	2	2	1	1	1	0	150	3	2	1	1	1	3
23	3	2	2	2	2	4	151	5	2	2	2	2	6
24	2	2	1	2	2	0	152	3	2	2	1	1	2
25	3	2	2	1	1	4	153	2	2	0	1	2	1
26	3	2	1	2	4	0	154	3	2	2	3	4	2
27	5	2	2	7	8	4	155	4	2	2	3	4	4
28	3	2	1	4	2	0	156	3	2	2	4	3	2
29	5	2	2	8	7	4	157	4	2	2	4	3	4
30	3	2	1	3	3	3	158	5	2	2	9	9	10
31	4	2	3	5	5	1	159	5	2	4	2	2	4
32	3	2	2	3	2	0	160	2	2	1	0	1	2
33	5	2	1	2	3	8	161	3	2	1	3	1	4
34	2	2	0	1	2	2	162	3	2	2	3	4	0
35	3	2	2	4	3	1	163	5	2	2	4	5	1

Table 4. (*continues*).

https://doi.org/10.25088/ComplexSystems.21.4.283

RN	m	n	Coefficients				RN	m	n	Coefficients			
↓	↓	↓	α_1	α_2	α_3	α_4	↓	↓	↓	α_1	α_2	α_3	α_4
36	2	2	1	2	1	2	164	3	2	1	2	1	2
37	3	2	1	2	1	4	165	2	2	1	0	2	1
38	3	2	2	1	4	0	166	3	2	2	3	1	3
39	5	2	2	7	3	1	167	4	2	3	2	4	4
40	5	2	1	1	2	9	168	3	2	2	2	4	3
41	2	2	1	1	2	1	169	3	2	2	2	1	1
42	3	2	2	2	1	3	170	2	2	0	0	1	0
43	3	2	2	2	3	1	171	4	2	2	2	3	1
44	3	2	1	4	3	2	172	5	2	2	5	3	7
45	3	2	1	3	2	1	173	4	2	3	5	4	6
46	5	2	2	8	4	0	174	4	2	2	5	3	3
47	4	2	3	5	2	6	175	3	2	2	0	3	1
48	2	2	1	2	0	0	176	3	2	1	2	3	0
49	3	2	2	1	3	4	177	5	2	2	3	4	6
50	3	2	2	1	2	2	178	3	2	2	3	2	2
51	2	2	0	1	0	1	179	4	2	2	3	2	4
52	3	2	1	4	3	0	180	3	2	1	3	2	3
53	5	2	2	4	6	4	181	4	2	3	5	4	1
54	3	2	2	4	2	2	182	5	2	2	6	2	2
55	4	2	2	4	2	4	183	5	2	2	4	2	4
56	3	2	1	4	2	3	184	5	2	2	4	3	10
57	3	2	2	4	3	4	185	4	2	2	4	3	6
58	5	2	2	4	5	10	186	4	2	2	5	4	2
59	4	2	2	4	5	6	187	3	2	0	2	3	1
60	2	2	1	1	0	0	188	4	2	3	3	2	3
61	4	2	3	3	2	1	189	3	2	2	2	3	4
62	4	2	3	3	5	3	190	5	2	2	2	5	10
63	3	2	2	2	0	4	191	4	2	2	2	5	6
64	3	2	2	2	3	0	192	2	2	1	1	0	2
65	5	2	1	1	3	6	193	3	2	1	1	3	4
66	2	2	1	1	2	2	194	3	2	1	1	2	2
67	3	2	1	1	2	4	195	2	2	1	2	0	1
68	2	2	0	1	2	0	196	3	2	2	4	3	0
69	3	2	2	3	4	1	197	5	2	2	3	6	1
70	3	2	2	4	1	0	198	3	2	2	1	3	3
71	5	2	2	3	7	1	199	4	2	3	4	2	4

Table 4. (*continues*).

https://doi.org/10.25088/ComplexSystems.21.4.283

RN	m	n	Coefficients				RN	m	n	Coefficients			
↓	↓	↓	α_1	α_2	α_3	α_4	↓	↓	↓	α_1	α_2	α_3	α_4
72	5	2	1	2	1	9	200	3	2	2	4	2	3
73	2	2	1	2	1	1	201	3	2	2	1	2	1
74	3	2	1	3	4	2	202	5	2	2	3	5	7
75	3	2	1	2	3	1	203	4	2	3	4	5	6
76	3	2	2	1	2	3	204	2	2	0	1	0	0
77	3	2	2	3	2	1	205	4	2	2	3	2	1
78	5	2	2	4	8	0	206	4	2	2	3	5	3
79	4	2	3	2	5	6	207	3	2	2	3	0	1
80	2	2	1	0	2	0	208	3	2	1	3	2	0
81	3	2	2	3	1	4	209	5	2	2	4	3	6
82	3	2	1	3	4	0	210	3	2	1	2	3	3
83	5	2	2	4	6	6	211	4	2	3	4	5	1
84	3	2	2	2	1	2	212	3	2	2	2	3	2
85	2	2	0	0	1	1	213	4	2	2	2	3	4
86	3	2	2	2	4	2	214	5	2	2	2	6	2
87	4	2	2	2	4	4	215	5	2	2	2	4	4
88	3	2	1	2	4	3	216	5	2	2	3	4	10
89	3	2	2	3	4	4	217	4	2	2	3	4	6
90	2	2	1	0	1	0	218	4	2	3	2	3	3
91	4	2	3	2	3	1	219	3	2	2	3	2	4
92	5	2	2	5	4	10	220	4	2	2	4	5	2
93	4	2	2	5	4	6	221	3	2	0	2	3	4
94	4	2	3	5	3	3	222	5	2	2	5	2	10
95	3	2	2	0	2	4	223	4	2	2	5	2	6
96	5	2	1	2	2	9	224	3	2	1	3	3	2
97	2	2	1	2	2	1	225	3	2	1	2	2	1
98	3	2	2	4	1	3	226	5	2	2	4	8	9
99	3	2	2	1	3	1	227	4	2	3	4	2	6
100	3	2	2	1	4	3	228	5	2	2	3	7	3
101	3	2	2	3	1	1	229	4	2	3	2	4	6
102	2	2	0	1	1	0	230	4	2	2	3	3	3
103	4	2	2	3	3	1	231	3	2	2	3	3	1
104	2	2	1	1	1	2	232	3	2	2	2	2	0
105	3	2	1	1	1	4	233	5	2	2	2	5	8
106	3	2	2	2	4	0	234	4	2	2	2	4	0
107	5	2	2	2	9	8	235	5	2	2	2	7	8

Table 4. (*continues*).

https://doi.org/10.25088/ComplexSystems.21.4.283

RN	m	n	Coefficients				RN	m	n	Coefficients			
↓	↓	↓	α_1	α_2	α_3	α_4	↓	↓	↓	α_1	α_2	α_3	α_4
108	3	2	2	4	2	0	236	4	2	2	4	2	0
109	5	2	2	9	2	8	237	5	2	2	5	9	1
110	4	2	2	4	4	0	238	3	2	0	2	2	2
111	5	2	4	2	9	6	239	4	2	2	5	5	1
112	3	2	1	2	2	3	240	2	2	1	0	0	0
113	3	2	2	3	3	4	241	4	2	3	2	2	1
114	5	2	2	3	5	10	242	4	2	3	2	5	3
115	4	2	2	3	5	6	243	3	2	2	3	0	4
116	5	2	2	4	8	2	244	4	2	3	5	2	3
117	4	2	2	5	3	6	245	3	2	2	0	3	4
118	4	2	2	4	4	2	246	5	2	2	4	4	2
119	3	2	0	2	2	4	247	4	2	2	5	5	6
120	3	2	1	3	3	0	248	4	2	3	5	5	5
121	5	2	2	6	9	4	249	5	2	2	4	7	6
122	4	2	3	5	3	5	250	3	2	2	0	2	2
123	5	2	2	4	9	6	251	4	2	2	5	2	4
124	4	2	3	3	5	5	252	3	2	2	2	0	2
125	5	2	2	2	6	4	253	4	2	2	2	5	4
126	3	2	2	2	2	2	254	4	2	2	2	2	2
127	4	2	2	2	2	4	255	2	2	0	0	0	1

Table 4. Rules 0–255 and their coefficients.

Every elementary CA can be represented in terms of the iota-delta function. This leads to the understanding that this function is CA universal in the sense that it can be used in the definition of every elementary CA.

By defining the evolution rules as functions whose domain is the CA mesh by means of the Church–Turing thesis, since the iota-delta function can be used to represent rule 110 (which has been proved to be universal), it is possible to say that this new function is also Turing universal. Finally, still based on the Church–Turing thesis, every dynamical system that is Turing equivalent to rule 110 is also representable by means of the iota-delta function.

A direct physical application of the iota-delta function has been given in [5].

https://doi.org/10.25088/ComplexSystems.21.4.283

4. Conclusions

In the present paper, a general transformation that can be applied to the whole cellular net is developed. By means of such transformation, every binary, that is, 0–255 cellular automata (CAs) are described. In addition, in order to provide a compact version of the transformation developed, a new function has been introduced: the iota-delta function. This new function is closely related to prime numbers and to the prime number theorem by means of the prime counting function, which reinforces the importance of this kind of number in science.

By drawing a parallel between the mathematical and computational notions of universality, it is possible to say that the iota-delta function is universal in the sense that it can be used to represent every dynamical system that is Turing equivalent to rule 110.

The simplicity of the iota-delta function and the way it has been used to describe elementary CAs requires further investigation. One interesting topic that would demand attention is how to prove the universality of rule 110 by means of this new function.

Acknowledgments

The authors would like to deeply thank Dr. Todd Rowland from Wolfram Research, Inc., for his contributions to the development of the ideas hereby presented. The authors would also like to thank the Brazilian National Research Council (CNPq), the Coordination for the Improvement of Higher Level Personnel (CAPES), and the University of Brasília (UnB) for funding this research.

References

[1] L. Smolin, "The Case for Background Independence." http://arxiv.org/abs/hep-th/0507235.

[2] M. Cook, "Universality in Elementary Cellular Automata," *Complex Systems*, **15**(1), 2004 pp. 1–40. http://www.complex-systems.com/pdf/15-1-1.pdf.

[3] S. M. Voronin, "Theorem on the Universality of the Riemann Zeta Function," *Izvestiya Akademii Nauk SSSR, Seriya Matematicheskaya*, **39**, 1975 pp. 475–486. Reprinted in *Mathematics of the USSR Izvestiya*, **9**(3), 1975 pp. 443–445. doi:10.1070/IM1975v009n03ABEH001485.

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

https://doi.org/10.25088/ComplexSystems.21.4.283

[5] L. C. de S. M. Ozelim, A. L. B. Cavalcante, and L. P. de F. Borges, "Continuum versus Discrete: A Physically Interpretable General Rule for Cellular Automata by Means of Modular Arithmetic." http://arxiv.org/abs/1206.2556.

https://doi.org/10.25088/ComplexSystems.21.4.283