An Alternative Representation of Turing Machines by Means of the Iota-Delta Function

Luan Carlos de Sena Monteiro Ozelim
André Luís Brasil Cavalcante
Department of Civil and Environmental Engineering
University of Brasilia
Campus Universitário Darcy Ribeiro, SG12, Asa Norte
Brasilia, Federal District, 70910-900, Brazil

Todd Rowland
Zeta Cubed LLC
Odenton, Maryland 21113, United States

Jan M. Baetens
KERMIT
Department of Data Analysis and Mathematical Modelling
Ghent University
Coupure links 653, B-9000, Ghent, Belgium

The evolution of universal systems has been of great interest to computer scientists. In particular, the role of Turing machines in the study of computational universality is widely recognized. Even though the patterns emerging from the evolution of this kind of dynamical system have been studied in much detail, the transition functions themselves have received less attention. In the present paper, the iota-delta function is used to encode the transition function of one-head Turing machines. In order to illustrate the methodology, we describe the transition functions of two universal Turing machines in terms of the latter function. By using the iota-delta function in this setting, Turing machines can be represented as a system of transition functions. This new representation allows us to write the transition functions as a linear combination of evolution variables wrapped by the iota-delta function. Thus, the nonlinear part of the evolution is totally described by the iota-delta function.

Keywords: Turing machines; universality; iota-delta function

1. Introduction

The study of Turing machines (TMs) has been of great interest to computer scientists. Since the pioneering work of Turing [1], extensive work has been done on the subject. In fact, one of the key

https://doi.org/10.25088/ComplexSystems.32.4.381

concepts in this area, computational universality, has its basis deeply related to TMs [2, 3].

In short, a Turing machine (TM) consists of four main parts: a tape, a head, a state register and a transition function [2]. The core of the updating process is how the transition function updates the states and heads of this type of machine. In general, transition functions are implemented as tables or replacement rules. When the number of possible states (also represented as colors in [2]) and heads (represented as directions in [2]) is large, defining replacement rules and transition tables becomes impractical due to the large number of input and output combinations.

Boolean functions can represent truth tables; however, when the number of input and output variables increases, functional complexity also increases. This way, Boolean algebra may not always provide the most compact notation [4].

Regarding other representations of Boolean functions (and truth tables), real polynomials are alternatives [5, 6]. Simply using Lagrange interpolating polynomials to describe Boolean functions is also possible [4].

The transition functions of elementary cellular automata (ECAs) can be represented in terms of Boolean functions, as discussed in [2]. It is worth noticing that ECAs are nothing but three-variable Boolean truth tables. Wolfram [2] also discussed algebraic representations for the evolution of a few rules, considering polynomials modulo 2 in such cases.

In a series of recent papers [7, 8], a new function that can represent transition functions of ECAs has been introduced. The so-called iota-delta function can be used to describe transition functions of not only one-dimensional but also of two-dimensional cellular automata (CAs) [9].

The Boolean universality [4] and functional structure of the iota-delta function allow it to represent truth tables in a straightforward and compact way. In addition, since the iota-delta representations are surjective (in the sense that every representation can only be linked to one rule), enumeration schemes can easily be built for the considered computational systems. By representing the transition functions of computational systems in terms of a well-defined mathematical function, standard analysis tools, such as differentiation and integration, can be used. Differentiation, for example, can be used to study stability, as indicated in [10].

Representing the evolution of TMs plays a central role in determining if the iota-delta function can be used to properly represent dynamical systems that are different from CAs.

Finally, since different dynamical systems can be encoded in terms of this new function, a comparison of the evolution of these systems is

simplified by comparing the transition function rather than the evolution patterns.

In the present paper, a generic procedure for obtaining the transition functions of TMs in terms of the iota-delta function is proposed. In particular, the transition functions of two universal TMs are obtained in terms of the iota-delta function.

In Section 2, we introduce the iota-delta function and formally define TMs and CAs. In Section 3, we describe the methodology proposed in the present paper to convert TMs into a sequence of CAs. In Section 4, two TMs are studied. Finally, Section 5 presents the conclusion of the present paper.

2. Preliminaries

2.1 Elementary Cellular Automata

A comprehensive study of ECAs can be found in [2]. In short, an elementary cellular automaton (ECA) consists of a one-dimensional array of cells, each colored black or white (1 or 0, respectively, in a binary description). At discrete time steps, a transition function assigns the new color (or value) of a given cell by considering the colors (or values) of that cell and its immediate left and right neighbors at the last time step [2].

Mathematically, let $g(.)$ be the transition function of an ECA. Then, the color C at position k and time step $i+1$ is given by:

$$C_k^{i+1} = g\left(C_{k-1}^i, C_k^i, C_{k+1}^i\right). \tag{1}$$

2.2 Turing Machines

Turing defined the class of abstract machines that now bear his name. A TM is a finite-state machine associated with a special kind of environment—its tape—in which it can store (and later recover) sequences of symbols [1, 11]. A TM consists of a tape, a head, a state register and a transition function [2]. The tape can be interpreted as a one-dimensional grid. The head can be thought of as two discrete variables, namely, head existence H and head state HS. The head indicates how the colors C of the cells are updated.

Mathematically, let $h(.)$ be the set of transition functions of a TM. Then, the color, head existence and head state at tape position k and time step $i+1$ can be symbolically expressed as:

$$\left(C_k^{i+1}, H_k^{i+1}, HS_k^{i+1}\right) = h\left(H_k^i, C_k^i, HS_k^i\right). \tag{2}$$

The universal TMs considered in the present paper are the 2-state, 3-color and 2-state, 5-color ones presented in Figure 1 and discussed in [2]. The evolution of such TMs can be visualized in Figure 2.

https://doi.org/10.25088/ComplexSystems.32.4.381

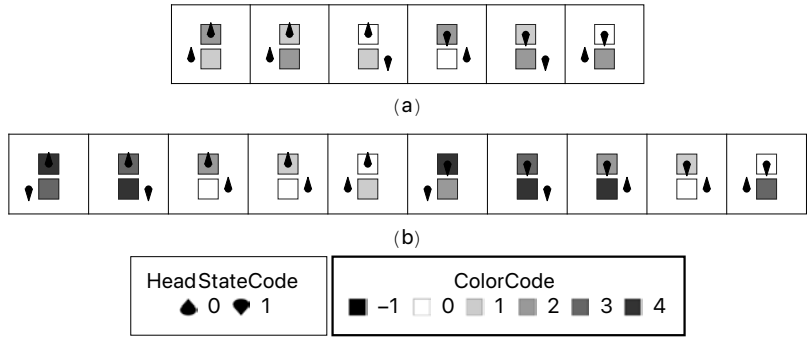

Figure 1. Transition functions of universal Wolfram (a) 2,3 and (b) 2,5 TMs.

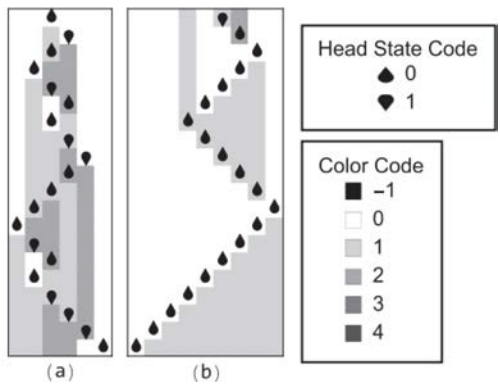

Figure 2. Evolution of universal TMs for 10 time steps: (a) Wolfram 2,3 TM with initial configuration of an up head state at a blank color cell, and (b) 2,5 TM with an initial configuration of a down head state at the third position of a colored tape with colors {1, 0, 1, 2}.The color and head state codes are presented at the right of the figure.

The next subsections present the fundamentals of the iota-delta function, as well as the general idea behind its usage to represent CA transition functions.

2.3 The Iota-Delta Function

The iota-delta function is defined as follows [7, 8]:

$$\iota\delta_n^m[x] = \mathrm{mod}[\mathrm{mod}[\ldots\mathrm{mod}[\mathrm{mod}[x, p_m], p_{m-1}], \ldots, p_j], n], \tag{3}$$

where $m \geq j$; $m, n \in \mathbb{Z}_+$; $x \in \mathbb{C}$; $j = \pi[n] + 1$ and $\mathrm{mod}[o, p]$ denotes the modulus operator, which gives the remainder of the division of o by p if o is greater than p or o itself. Otherwise, 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, which gives the number of primes less than or equal to n. Note that $p_1 = 2$. Essentially, n determines how many different outputs the function may have. Thus, for binary outputs, $n = 2$; for ternary ones, $n = 3$; for quaternary ones, $n = 4$ and so on. On the other hand, m gives a notion of functional complexity. As m increases, more mod functions need to be nested. Also, the iota-delta function is taken to be non-negative and $\max[\iota\delta_n^m[x]] \to n$ when $x \in \mathbb{R}$ [7, 8]. The iota-delta function is periodic with period p_m.

2.4 The Iota-Delta Function and Elementary Cellular Automata

It has been shown [7] that every ECA can be represented by the following transition function:

$$C_k^{i+1} = \iota\delta_2^m\left[\alpha_1 C_{k-1}^i + \alpha_2 C_k^i + \alpha_3 C_{k+1}^i + \alpha_4\right], \tag{4}$$

where the coefficients are $\alpha_j = \{r \mid r \le p_m - 1;\ r \in \mathrm{Z}_+\};\ j = 1, 2, 3, 4$. Also, in equation (4), the indices k and i relate to the position of the cell in the bidimensional (space and time, respectively) state space of the CA. In the case of ECAs, the minimum value of m that enables encoding every rule is $m = 5$, that is:

$$\iota\delta_2^5[x] = \mathrm{mod}[\mathrm{mod}[\mathrm{mod}[\mathrm{mod}[\mathrm{mod}[x, 11], 7], 5], 3], 2]. \tag{5}$$

In this way, every ECA is characterized by a set of tuples $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ [7]. For example, rule 110 can be described by [7]:

$$C_k^{i+1} = \iota\delta_2^5\left[2C_{k-1}^i + 4C_k^i + 4C_{k+1}^i\right], \tag{6}$$

so $\alpha_1 = 2$, $\alpha_2 = 4$, $\alpha_3 = 4$ and $\alpha_4 = 0$. A possible iota-delta representation of rule 30 is given by:

$$C_k^{i+1} = \iota\delta_2^5\left[C_{k-1}^i + 4C_k^i + 4C_{k+1}^i\right]. \tag{7}$$

Clearly, the iota-delta function is a powerful tool to represent the transition functions of ECAs. As can be seen from equation (6), the nonlinearity of the transition function is totally concentrated in the iota-delta function, while the dependency on the neighbor cells is linear.

We refer the reader to [9] to obtain the transition functions of two-dimensional CAs in terms of the iota-delta function.

2.5 Usage

The philosophy behind the usage of the iota-delta function for the representation of transition functions is to identify the variables that are important for determining the value of a given cell [9]. For example, in the case of ECAs, the value of a given cell depends on the value of its immediate left and right neighbors and the value of the cell itself in

the current step. So, three variables determine the value of the cell of interest in the next step. This way, the iota-delta function representation given by equation (4) fits the need. It is worth noting that in addition to the coefficients multiplying each variable, a fourth coefficient is needed to represent odd rules.

The main advantage of the iota-delta function comes from the fact that its representation isolates the transition function's nonlinearity in the function itself, while the dependency on the neighborhood cells becomes linear. Besides that, by representing the transition functions in terms of a mathematical function, regular operations such as differentiation and integration can be easily performed. This allows us to analyze the transition functions using the robust tools mathematical analysis has to offer.

This separation of linearity and nonlinearity may be fundamental to link computational systems to real-world physical phenomena. In [12], the iota-delta representation of ECAs made it possible to link such dynamical systems and partial differential equations describing advective-dispersive phenomena.

The next section discusses how to interpret TMs as a sequence of CAs.

3. Turing Machines as a Sequence of Cellular Automata

As stated in the Section 2, the representation of the transition function of a CA in terms of the iota-delta function requires knowledge of the variables that govern the updating process.

In a TM of the type presented in Figure 1, it is clear that there are three entities of interest, namely, the color of the cell, the state of the head and finally its position. This suggests that three mutually dependent entities should be taken into account by different transition functions. This will be outlined in the remainder of this section.

3.1 The Color Rule

Figure 1 reveals that the color of a given cell at the next step $i+1$ depends on its color in the present step i. Also, by definition, only the cells in which a head is present are updated; thus, the color at the next step also depends on the existence of heads. Finally, besides these two variables, the color at the next step depends on the state of the head.

This way, there are three variables that control the evolution of the colors in TMs like the one shown in Figure 1, namely, the j^{th} color of the cell in the present state, denoted by $C_k^i \in \{0, 1, \ldots, j-1\}$; the

existence of a head, $H_k^i \in \{0, 1\}$; and finally, the q^{th} state of the latter, $HS_k^i \in \{0, 1, \ldots, q-1\}$.

It is straightforward to suppose that the transition function of the color is given by:

$$C_k^{i+1} = \left(1 - H_k^i\right)C_k^i + H_k^i \iota\delta_{n_1}^{m_1}\left[\alpha_{1,1}C_k^i + \alpha_{1,2}HS_k^i + \alpha_{1,3}\right] \tag{8}$$

where the double subscripts for the α values are just an indexation trick. These indices will be used to build a matrix of coefficients later on and will indicate the position of the α as (row, column) inside the matrix.

In short, equation (8) states that the colors of all k tape positions are updated at time step $i+1$ according to a rule of the type

$$\iota\delta_{n_1}^{m_1}\left[\alpha_{1,1}C_k^i + \alpha_{1,2}HS_k^i + \alpha_{1,3}\right]$$

when the head is present. Otherwise, the color remains the same.

3.2 The Head Rule

There are two possibilities, either a head exists or it does not. This way, it is important to investigate how a given head is displaced at each time step.

In order to better understand the update process, a parameter μ_k^i is defined:

$$\mu_k^i = H_k^i\left(1 - 2\iota\delta_{n_2}^{m_2}\left[\alpha_{2,1}C_k^i + \alpha_{2,2}HS_k^i + \alpha_{2,3}\right]\right). \tag{9}$$

This parameter indicates, based on the color C_k^i, head state HS_k^i and head existence H_k^i, how the head should move. Note that we must have $n_2 = 2$, since the heads may move at most one cell to the left or to the right at each subsequent time step. Thus, if $\mu_k^i = 0$, the head does not move. On the other hand, when $\mu_k^i = 1$, the head moves one cell to the right and when $\mu_k^i = -1$, the head moves one cell to the left.

The allocation parameter can have the following values:

$$\mu_k^i = \begin{cases} 0, & \text{if} \quad H_k^i = 0, \\ 1, & \text{if} \quad H_k^i = 1 \quad \text{and} \quad \iota\delta_2^{m_2}\left[\alpha_{2,1}C_k^i + \alpha_{2,2}HS_k^i + \alpha_{2,3}\right] = 0, \\ -1, & \text{if} \quad H_k^i = 1 \quad \text{and} \quad \iota\delta_2^{m_2}\left[\alpha_{2,1}C_k^i + \alpha_{2,2}HS_k^i + \alpha_{2,3}\right] = 1. \end{cases} \tag{10}$$

The allocation parameter is then used to update the existence of heads as follows:

$$H_k^{i+1} = H^i_{k-(\mu^i_{k-1}+\mu^i_k+\mu^i_{k+1})} \tag{11}$$

where:

$$\overline{\mu}^i_k = \mu^i_{k-1} + \mu^i_k + \mu^i_{k+1}. \tag{12}$$

3.3 The Head State Rule

It can be seen from Figure 1 that the head state at position k depends on the color of a given cell at position $k - \overline{\mu}^i_k$ and, if a head is present in the latter, the head state of this cell (position $k - \overline{\mu}^i_k$ as well). Thus, a suitable transition function is:

$$HS_k^{i+1} = H^i_{k-\overline{\mu}^i_k} \iota\delta^{m_3}_{n_3}\left[\alpha_{3,1} C^i_{k-\overline{\mu}^i_k} + \alpha_{3,2} HS^i_{k-\overline{\mu}^i_k} + \alpha_{3,3}\right]. \tag{13}$$

Equations (8), (9) and (13) can be expressed in a compact way by means of the matrix of coefficients of the involved iota-delta functions as:

$$\begin{pmatrix} m_1 & n_1 & \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} \\ m_2 & n_2 & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ m_3 & n_3 & \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} \end{pmatrix}_{\iota\delta}. \tag{14}$$

3.4 Implementing the Iota-Delta Representation of Single-Head Turing Machines

The implementation of the equations presented is quite straightforward. We coded them in Wolfram Language, but they can be easily written in any other programming language.

In Figure 3, the dotted arrows in the flow chart represent the last iteration. Dashed arrows represent the current iteration and full paths represent the next iteration.

Figure 3 shows that the value of the allocation parameter μ^i_k at time step i depends on the values of the other parameters at this time step. This is why the order presented in equation (15) becomes important, as the subsequent equations depend on previous ones.

By drawing a parallel to the standard evolution of TMs, the well-known visualization of each evolution step can be achieved by superposing the rows of the matrices C, H and HS, as Figure 3 indicates. The code used to implement the update process in Figure 3 is available upon request.

To illustrate how the evolution is carried out, Figure 4 presents the updating process at an intermediate set of stages.

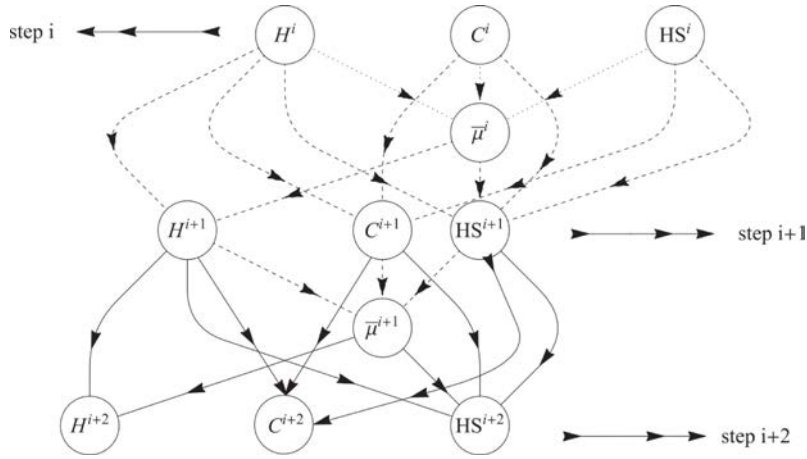

Figure 3. Update process of the iota-delta representation of TMs.

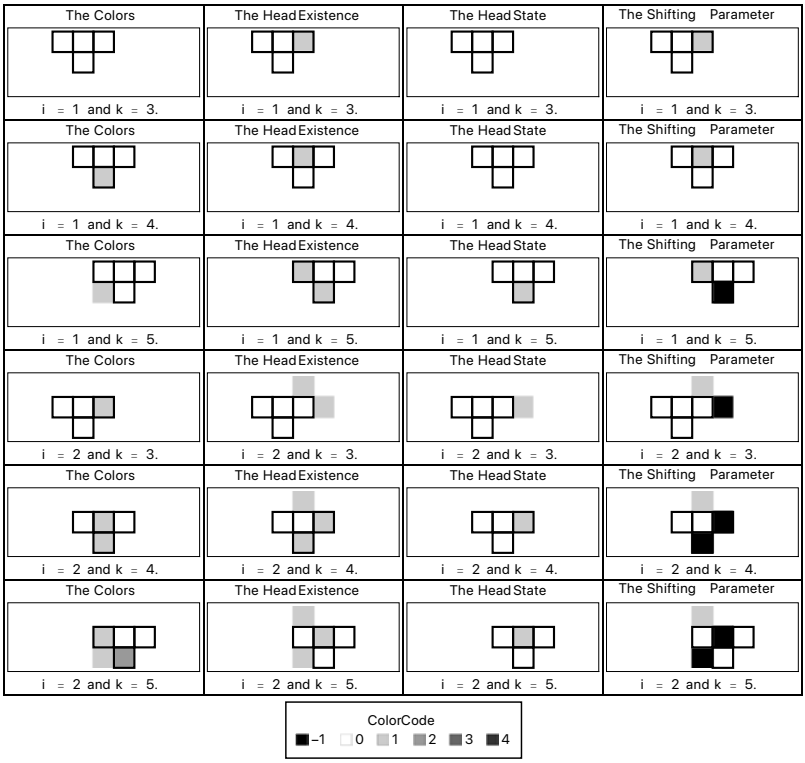

Figure 4. Evolution of universal Wolfram 2,3 TM for two time steps between tape positions 3 and 5. The initial condition is an up head state at a blank cell at position $k = 4$.

https://doi.org/10.25088/ComplexSystems.32.4.381

4. Transition Functions of Turing Machines in Terms of the Iota-Delta Function

By means of the representation in equation (14), the transition functions of single-headed TMs with n_1 colors and n_3 head states are expressible in terms of iota-delta functions. To illustrate our methodology, the transition functions of two universal TMs are constructed in terms of the iota-delta function. For simplicity, up head states are taken as 0, while down head states are taken as 1, and the color range is from 0 to $n_1 - 1$.

In general, finding the iota-delta representation of a given TM comes down to solving the system of equations given by equations (8), (9) and (13). These three equations lead to a nonlinear system whose solution is not unique; therefore, more than one set of coefficients provides a solution. The simplest approach to finding a solution is: given the inputs and outputs, use search algorithms to find the coefficients that solve the equations. The naive approach, which was considered in the present paper, was to perform a brute force search. Depending on how intricate the TM evolution is, other alternatives such as Bayesian optimization with probabilistic reparameterization can be useful tools to find the discrete-valued coefficients [13]. Continuous-valued counterparts are the focus of future research efforts.

4.1 The 2,3 Wolfram Universal Turing Machine

As shown in Figure 1, the 2,3 Wolfram universal TM has one head, two head states and three colors, so the approach proposed in Section 3 can be used. By means of equations (8), (11) and (13), the update process of the TM can be represented as:

$$\begin{cases} C_k^{i+1} &= \left(1 - H_k^i\right)C_k^i + H_k^i \iota\delta_3^{10}\left[25C_k^i + 19HS_k^i + 6\right], \\ H_k^{i+1} &= H_{k-\overline{\mu}_k^i}^i, \\ HS_k^{i+1} &= H_{k-\overline{\mu}_k^i}^i \iota\delta_2^2\left[2C_{k-\overline{\mu}_k^i}^i + HS_{k-\overline{\mu}_k^i}^i + 1\right], \end{cases} \tag{15}$$

where $\mu_k^{i+1} = H_k^{i+1}\left(1 - 2\iota\delta_2^3\left[2C_k^{i+1} + 4HS_k^{i+1} + 2\right]\right)$. Alternatively, equation (15) can be written as:

$$\begin{pmatrix} 10 & 3 & 25 & 19 & 6 \\ 3 & 2 & 2 & 4 & 2 \\ 2 & 2 & 2 & 1 & 1 \end{pmatrix}_{\iota\delta}. \tag{16}$$

4.2 The 2,5 Wolfram Universal Turing Machine

Figure 1 reveals that the 2,5 Wolfram universal TM has one head, two head states and five colors. Again, by means of equations (8), (9),

(11) and (13), the update process of this TM can be represented as:

$$\begin{cases} C_k^{i+1} &= \left(1 - H_k^i\right)C_k^i + H_k^i \iota\delta_5^{45}\left[119C_k^i + 58HS_k^i + 72\right], \\ H_k^{i+1} &= H_{k-\overrightarrow{\mu}_k^i}^i, \\ HS_k^{i+1} &= H_{k-\overrightarrow{\mu}_k^i}^i \iota\delta_2^3\left[3C_{k-\overrightarrow{\mu}_k^i}^i + 2\right], \end{cases} \tag{17}$$

where $\mu_k^{i+1} = H_k^{i+1}\left(1 - 2\iota\delta_2^3\left[2C_k^{i+1} + 1\right]\right)$. The more compact formulation of equation (17) reads:

$$\begin{pmatrix} 45 & 5 & 119 & 58 & 72 \\ 3 & 2 & 2 & 0 & 1 \\ 3 & 2 & 3 & 0 & 2 \end{pmatrix}_{\iota\delta}. \tag{18}$$

Equation (17) reveals that the position of the head in the next step $i+1$ does not depend on its state in the previous step, and similarly for the head state.

4.3 Possible Physical Interpretation

The physical interpretation of the iota-delta representation of ECAs was given in [12]. In that paper, the authors compared the iota-delta transition function to a finite difference scheme of the one-dimensional advective-dispersive equation. For the system of equations summarized in equation (14), a possible physical interpretation would be to link the system of iota-delta transition functions to a system of finite difference approximations of a system of partial differential equations. A possible candidate would be the system of equations that model shallow water flow [14], but further exploration of this approach is out of the scope of the present paper.

5. Conclusion

Computational universality has been of great interest to computer scientists. Ways of achieving universality are often related to Turing machines (TMs). In particular, the transition functions of the latter are of special interest in such studies.

In the present paper, a general framework for describing the transition functions of single-head TMs was proposed. In short, the methodology is based on understanding a Turing machine (TM) as a sequence of mutually dependent cellular automata (CAs). In order to provide a solid mathematical representation of the transition functions, the iota-delta function was employed.

By using the iota-delta function, questions concerning universality of TMs will find a better mathematical framework to be addressed.

https://doi.org/10.25088/ComplexSystems.32.4.381

Acknowledgments

This study was financed in part by the Coordination for the Improvement of Higher Education Personnel (CAPES)—Finance Code 001. The authors also acknowledge the support of the National Council for Scientific and Technological Development (CNPq Grant 305484/2020-6) and of the Research Support Foundation of the Federal District (FAP-DF 00193.00000920/2021-12).

References

[1] A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, **2**(42), 1936–1937 pp. 230–265. doi:10.2307/2268808.

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

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

[4] L. Ozelim and A. Cavalcante, "The Iota-Delta Function as an Alternative to Boolean Formalism," *International Journal of Foundations of Computer Science*, **29**(3), 2018 pp. 415–423. doi:10.1142/S0129054118500120.

[5] D. A. Mix Barrington, R. Beigel and S. Rudich, "Representing Boolean Functions as Polynomials Modulo Composite Numbers," *Computational Complexity*, **4**(4), 1994 pp. 367–382. doi:10.1007/BF01263424.

[6] N. Nisan and M. Szegedy, "On the Degree of Boolean Functions as Real Polynomials," *Computational Complexity*, **4**(4), 1994 pp. 301–313. doi:10.1007/BF01263419.

[7] L. Ozelim, A. Cavalcante and L. Borges, "On the Iota-Delta Function: Universality in Cellular Automata's Representation," *Complex Systems*, **21**(4), 2013 pp. 283–296. doi:10.25088/ComplexSystems.21.4.283.

[8] L. Ozelim, A. Cavalcante and L. Borges, "Continuum versus Discrete: A Physically Interpretable General Rule for Cellular Automata by Means of Modular Arithmetic," *Complex Systems*, **22**(1), 2013 pp. 75–99. doi:10.25088/ComplexSystems.22.1.75.

[9] L. Ozelim and A. Cavalcante, "On the Iota-Delta Function: Mathematical Representation of Two-Dimensional Cellular Automata," *Complex Systems*, **22**(4), 2013 pp. 405–422. doi:10.25088/ComplexSystems.22.4.405.

[10] J. M. Baetens and B. De Baets, "Phenomenological Study of Irregular Cellular Automata Based on Lyapunov Exponents and Jacobians," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **20**(3), 2010 033112. doi:10.1063/1.3460362.

[11] M. L. Minsky, *Computation: Finite and Infinite Machines*, Englewood Cliffs, NJ: Prentice-Hall, 1967.

[12] L. Ozelim, A. Cavalcante and J. Baetens, "On the Iota-Delta Function: A Link between Cellular Automata and Partial Differential Equations for Modeling Advection–Dispersion from a Constant Source," *The Journal of Supercomput*ing, **73**(2), 2017 pp. 700–712. doi:10.1007/s11227-016-1795-7.

[13] S. Daulton, X. Wan, D. Eriksson, M. Balandat, M. A. Osborne and E. Bakshy, "Bayesian Optimization over Discrete and Mixed Spaces via Probabilistic Reparameterization," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, (NeurIPS 2022)*, New Orleans, (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, eds.), 2022. proceedings.neurips.cc/paper_files/paper/2022/file/531230cfac80c65017ad0f85d3031edc-Paper-Conference.pdf.

[14] C. B. Vreugdenhil, *Numerical Methods for Shallow-Water Flow*, Boston: Kluwer Academic Publishers, 1994.