Complex Behavior in Long-Distance Cellular Automata

Lucas Kang
Thomas Jefferson High School for Science and Technology

The purpose of this study was to systematically explore the behavior of one-dimensional long-distance cellular automata (LDCAs). Basic characteristics of LDCAs are explored, such as universal behavior, the prevalence of complexity with varying neighborhoods, and qualitative behavior as a function of configuration. It was found that rule 73 could potentially be Turing universal through the emulation of a cyclic tag system, and that a connection between the mathematics of binary trees and Eulerian numbers might provide insight into unsolved problems.

1. Introduction

Cellular automata (CAs) have been used for decades as mathematical idealizations of physical systems in which space and time are discrete and in studies of self-organizing physical, chemical, and biological phenomena [1–5]. A cellular automaton (CA) consists of an array of cells in some number of dimensions that generates a new state, following a simple evolutionary rule in each successive step [1, 2]. In the world of computational science, several classifications of CAs have risen to prominence, ranging from one- to three-dimensional automata to those of even higher dimension. (A survey of CA classification is available in [6].) However, Wolfram's classification of elementary CAs (ECAs) has become one of the most widely known forms of evolving computational systems and has arguably revolutionized the exploration of CAs [1]. The purpose of this paper is to study the complexity of one-dimensional CAs in configurations other than that of Wolfram's ECA. A new class of long-distance CAs (LDCAs) was explored whose properties are not governed by ECAs. LDCAs, which are briefly described in Wolfram's notes, are considered a generalization of one-dimensional elementary rules that encompasses all possible neighborhood configurations [1]. It was demonstrated that certain LDCAs possess interesting qualities and are candidates for computational universality, which implies that they may be used to solve problems in other fields of research [1, 7].

ECAs are defined as the simplest class of one-dimensional CAs and have two possible states for each cell (0 or 1). These states are placed

Complex Systems, 23 © 2014 Complex Systems Publications, Inc.
https://doi.org/10.25088/ComplexSystems.23.2.93

in an array, so that every cell dictates the condition of a particular part of the system. This array, which can be finite or infinite in length, when taken from a one-dimensional CA or ECA can be thought of as resembling a tape of a Turing machine, so it will be referred to as such through the extent of this paper [1, 2]. The evolution of an array is the collective evolution of all individual cells, which is dependent only on the values of each cell's neighborhood. The neighborhood is defined as containing the cell itself and all immediately adjacent sites; in one-dimensional CAs, the neighborhood consists of three adjacent cells. As a result, the evolution of any ECA can be described through a table of evolutionary steps, specifying the state a given cell will have in the next generation, based on the values of the cell to its left, the cell to its right, and the target cell itself. For every ECA, there must exist $2^3 = 8$ such steps in a table, and because there are two possible outcomes (0 or 1) of every step, there exists a total of $2^8 = 256$ total tables to describe ECAs. Each of these tables is said to be a "rule" of evolution and so there exist 256 defined ECAs, ranging from rule 1 to rule 256 [1].

Research on ECAs often involves specific forms or cases of one-dimensional CAs. The case of time offsets in CAs (especially ECAs) evolution has been explored, notably by Alonso-Sanz [8–11] and Letourneau [12, 13]. But although related, spatial offsets have been relatively unstudied in the past.

LDCAs, a class of spatially offset ECA rules, use spaced sample cells, unlike ECAs, and are described by the notation LDCA-x-y-n, where x and y represent the amount of spacing between the cell and its left and right neighbors, and n denotes the length of the initial tape (for tapes of finite size). The values for x and y must always be greater than 0, and since the tape of a finite LDCA is cyclic, $x + y$ is unrestricted. In this experiment, LDCA-1-2 was studied, so as to form a basis for a systematic exploration with larger x and y values (Figure 1). While many compendiums of classifications of CAs exist, there exists no formal experimentation on the properties and applications of LDCAs, specifically LDCA-1-2 [6, 14, 15].

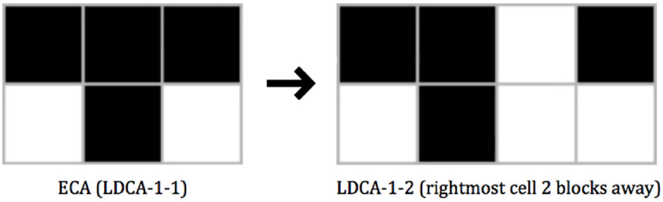


Figure 1. The layout of the three sampled cells and one output cell in an ECA (left), and a corresponding image for the configuration of LDCA-1-2-n (right).

https://doi.org/10.25088/ComplexSystems.23.2.93

It was observed that LDCAs, due to their separated sample cells, provide functions that ECAs cannot and theoretically can be used to model chaotic systems in which every evolutionary step possesses an inherent random input. This is the case because as an ECA rule is applied in configurations LDCA-x-1 or LDCA-1-y on a tape of random initial conditions, for $1 < x$ or $y < n - 4$, its behavior can be thought of as that of a two-cell CA, while the third, more remote cell provides a seemingly unrelated and potentially random source of input. This form of evolution, in which an outside effect or random input actively modifies a system, is observed in maps of electrical activity where negligible but existent charges affect components, biological systems such as angiogenesis where capillary growth is predictable but nondeterministic, and chaos theory's butterfly effect, in which a relatively small initial change can affect a larger outcome. Additionally, when the CA's configuration approaches LDCA-x-y, for $1 < x$ or $y < n - 4$, the result can be unrecognizable in comparison to standard ECAs.

LDCAs also have the distinct property of possessing repetitive state transition diagrams. When applying an ordered list of all 256 ECA rules to an LDCA-x-y-n such that $x + y = 0 \pmod n$ or $x + y = x$ or y $(\operatorname{mod} n)$, then the rules will generate a series of repeated state transition diagrams. If the LDCA samples only the target cell and one other cell, so that $x + y = 0 \pmod n$, then the table of rules will create two distinct lists of diagrams with four rules each that will be alternated (Figure 2). If the LDCA samples one cell from anywhere on the tape and one from the same cell as the center cell, so that $x + y = x$ or y $(\operatorname{mod} n)$, then there are two possible outcomes. If $x = n$ and therefore $x + y = y \pmod n$, then the repeated lists of state transition diagrams will be four rules in length (Figure 3). If $y = n$ and so $x + y = x$ $(\operatorname{mod} n)$, then the lists will be eight rules in length (Figure 4). And if the LDCA samples all three cells from the same cell so that $x + y = 0$ $(\operatorname{mod} n)$ and $-x + y = 0 \pmod n$, then each repeated list of diagrams will have two rules each (Figure 5).

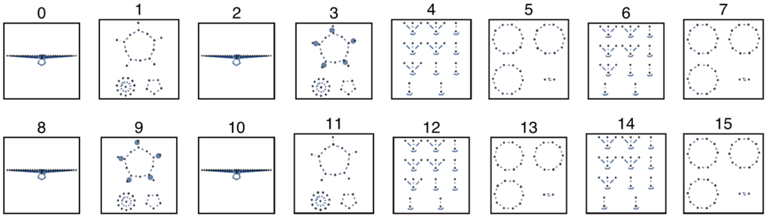


Figure 2. ECA rules in LDCA-1-4-5. Since $x + y = 0 \pmod n$, the state transition diagrams repeat in groups of four [16].

https://doi.org/10.25088/ComplexSystems.23.2.93

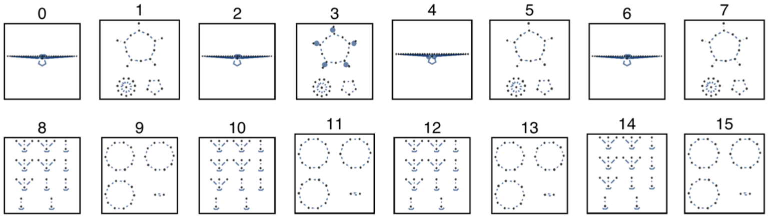


Figure 3. ECA rules in LDCA-5-1-5. Since $x + y = y \pmod{n}$, the state transition diagrams repeat in groups of four [16].

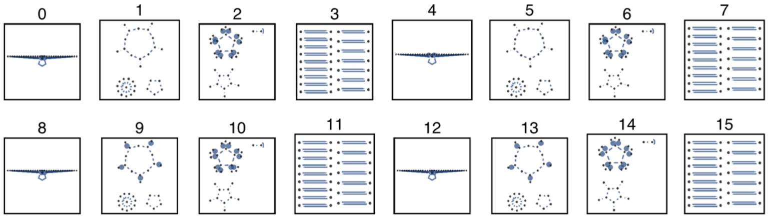


Figure 4. ECA rules in LDCA-1-5-5. Since $x + y = x \pmod{n}$, the state transition diagrams repeat in groups of eight [16].

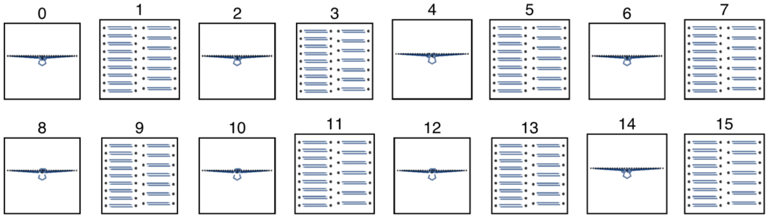


Figure 5. ECA rules in LDCA-5-5-5. Since $x + y = 0 \pmod{n}$ and $-x + y = 0 \pmod{n}$, the state transition diagrams repeat in groups of two [16].

2. Rules 73 and 109

The two-color, two-state CA that was studied in the LDCA-1-2 configuration is known as rule 73 according to Wolfram's numbering scheme [1]. Rule 73 is one of two purely class 4 CAs in LDCA-1-2 and was suspected of computational universality.

In the evolution of rule 73 (Figure 6), each cell is in one of two states {0, 1} and, because the rule is being applied to an LDCA-1-2 configuration, every cell synchronously updates itself according to the value of itself and its nearest neighbors, according to the function $F(C_{i-1}, C_i, C_{i+2})$ [17]:

$$\begin{array}{llll} F(1,1,1)=0 & F(1,1,0)=1 & F(1,0,1)=0 & F(1,0,0)=0 \\ F(0,1,1)=1 & F(0,1,0)=0 & F(0,0,1)=0 & F(0,0,0)=1. \end{array}$$

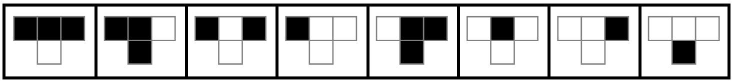
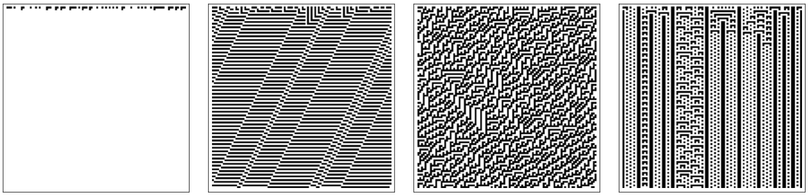


Figure 6. This table depicts the evolutionary substitution rules of rule 73. Rules 73 and 109 are equivalent through both left-right and color equivalence.

Figure 7 shows examples of the four classes of CAs in LDCA-1-2. It is notable that rules 8, 21, 45, and 73 behave differently than their ECA counterparts. Rule 73 is presented as the example of a class 4 CA in LDCA-1-2.



Rule 8 Rule 21 Rule 45 Rule 73

Figure 7. These four images exemplify class 1, 2, 3, and 4 behavior in LDCA-1-2 through rules 8, 21, 45, and 73, respectively.

Interestingly, rule 73 is equivalent to rule 109 through both left-right and color equivalence, which means that the two rules are identical after either one's color is inverted and evolution is mirrored horizontally (Figure 8). This entails that studying either rule implies the exploration of the other. Rule 109 evolves according to the function $G(C_{i-1}, C_i, C_{i+2})$, where G is the following function [18]:

$$\begin{array}{llll} G(1,1,1)=0 & G(1,1,0)=1 & G(1,0,1)=1 & G(1,0,0)=0 \\ G(0,1,1)=1 & G(0,1,0)=1 & G(0,0,1)=0 & G(0,0,0)=1. \end{array}$$

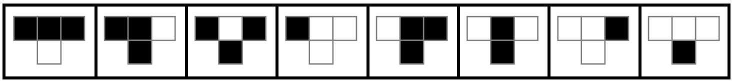


Figure 8. This table depicts the evolutionary substitution rules of rule 109. Rules 109 and 73 are equivalent through both left-right and color equivalence.

3. Computational Universality

Universality was a major factor in the choosing of rule 73; it was suggested that class 4 CAs might be capable of universal computation in 1984 [7], and rule 73 is one of two purely class 4 automata in LDCA-1-2, with the other being rule 109. (Both rules 73 and 109 are defined as class 2 rules in the ECA rule space, but when evolved as LDCA-1-2, they exhibit purely class 4 behavior. They have also been shown to possess class 4 behavior in other contexts [14].) Besides exploring the properties and characteristics of rule 73, this study will also attempt to demonstrate the universality of rule 73.

Universal computational systems are those that are theoretically capable of emulating any other system [1, 7, 19]. This means that a singular system would be capable of behaving as any other mathematically definable system, which has significant implications in computational science. Such systems usually require an encoding and decoding process, in order to translate information and behavior [19]. For example, Boolean logic systems, or computer programs, are universal, but only after the system being emulated has been coded in binary, and the result of the program has been translated back into the language of the original system.

Church's effective calculability, Turing's computability, Post's canonical systems, Kleene's general recursive functions, and Smullyan's elementary formal systems have all resulted in the exact same computational capability. This phenomenon has led to the generally accepted thesis that these systems are capable of carrying out any specifiable procedure whatsoever [19, 20]. The proving of a computational system's universality is usually done through the emulation of another system previously known to be universal. As a result of the Church–Turing thesis, Turing machines have been defined as universal. Then in 2004, Cook proved that cyclic tag systems could successfully emulate universal Turing machines and were therefore universal [1, 19]. While several CAs have been shown or suspected to be universal, the most commonly known example is that of the ECA rule 110, which was shown to emulate a universal cyclic tag system [19].

4. Methodology: Visualization

The shift from ECAs to LDCA-1-2 brings several changes to rule 73, including the way that it is visualized (Figure 9). When the evolution of rule 73 in LDCA-1-2 is plotted normally, gliders, or particles as they will be referred to in this paper, tend to blend into the background or become difficult to distinguish as they collide.

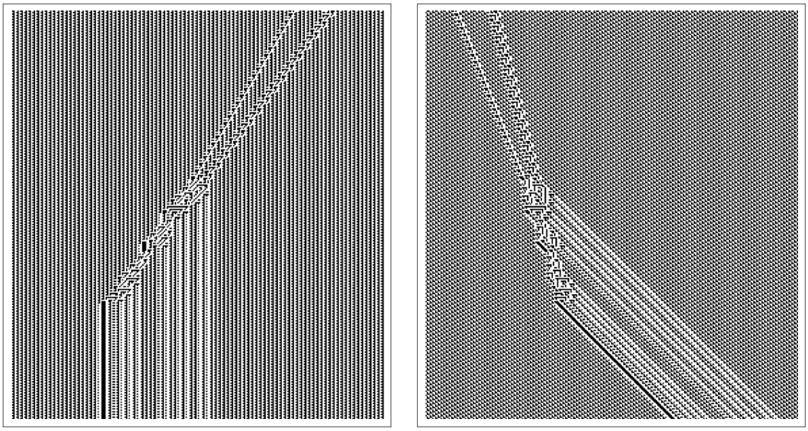


Figure 9. Here it can be seen that LDCA 73 possesses several stationary particles (vertical lines in the picture on the left). When the plot of rule 73's evolution is skewed, however, collisions between gliders and the resulting particles become more apparent.

While it is standard to plot the tape of a CA so that cell i of a previous state is directly above cell i in the next state, the evolution of LDCA-1-2 was skewed so that the position of cell i in time step x corresponds to that of cell $i-1$ in $x+1$. This was done through a rotation function:

```
Table[Nest[RotateRight, #[[i]], i - 1], {i, 1, Length[#]}] &
CellularAutomaton[…]
```

When rule 73 was evolved in LDCA-1-y for y greater than 2, it was graphed with an $i \to i-y+1$ skew. This aided in the visualization of rule 73 in various LDCA configurations, but became less meaningful as rule 73 approached chaotic behavior in LDCA-1-5 and beyond (Figure 10).

In this study, it was assumed that all calculations and plots regarding rule 73 in LDCA-1-2 are developed with an $i \to i-1$ skew.

https://doi.org/10.25088/ComplexSystems.23.2.93

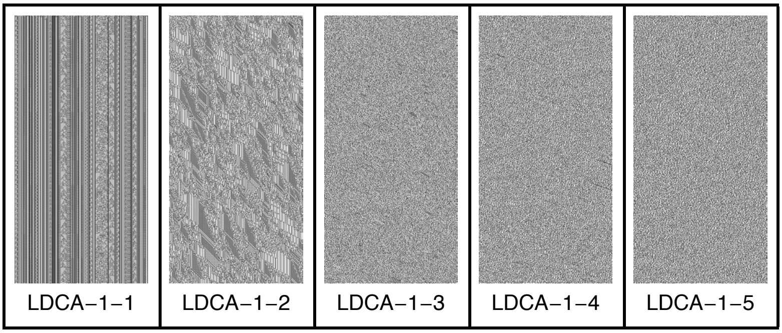


Figure 10. As the y value in LDCA-1-y was increased, rule 73 approached purely chaotic, or class 3, behavior. Despite applying an increasing skew, it was difficult to identify localized structures in the evolution.

5. Methodology: Block Emulation

Block emulation is a form of emulation that can be used to find emulations of CAs in different rule spaces. By substituting certain blocks in a certain CA, a simpler or more complicated automaton can be created, which may emulate other rules or a different computational system. The main idea is to encode one cell of rule A into n blocks of cells of rule B. By replacing corresponding blocks according to a set of rules, one can transform a cellular automaton of one rule into a replica of another rule [1]. With this concept, it can be shown, for example, that rule 22 is able to emulate rule 90. However, this type of emulation is not possible in all cases. Some rules with blocks up to a certain block size, rule 30 for example, are not able to emulate any fundamental rules at all through block emulation.

In an effort to see what rule 73 in LDCA-1-2 emulated, block emulation was utilized in the 3/2 rule space. To begin, rule 73 was converted to a rule in the 3/2 rule space, and all evolutionary rules in rule 73 that return a black cell were found:

$$\begin{array}{llll} F(1,1,1)=0 & F(1,1,0)=1 & F(1,0,1)=0 & F(1,0,0)=0 \\ F(0,1,1)=1 & F(0,1,0)=0 & F(0,0,1)=0 & F(0,0,0)=1. \end{array}$$

Then, all corresponding configurations of cells in the 3/2 rule space were found. In 3/2, the cells are formatted as a, b, c, d, with the CA sampling cells a, b, and d. Since cell c remains unsampled, it can be either 1 or 0 in the translation from ECA to 3/2. Therefore:

$$F_{3/2}(1, 1, 0, 0) = 1 \quad F_{3/2}(1, 1, 1, 0) = 1$$
$$F_{3/2}(0, 1, 0, 1) = 1 \quad F_{3/2}(0, 1, 1, 1) = 1$$
$$F_{3/2}(0, 0, 0, 0) = 1 \quad F_{3/2}(0, 0, 1, 0) = 1.$$

Next, all of the possible inputs were converted into base 10:

$$(1, 1, 0, 0) \to 12 \quad (1, 1, 1, 0) \to 14$$
$$(0, 1, 0, 1) \to 5 \quad (0, 1, 1, 1) \to 7$$
$$(0, 0, 0, 0) \to 0 \quad (0, 0, 1, 0) \to 2.$$

The sum was taken of 2 to the power of each of the results to get rule 20645:

$$2^{12} + 2^{14} + 2^5 + 2^7 + 2^0 + 2^2 = 20\,645.$$

Finally, a comparison between rule 20645 in 3/2 and rule 73 in ECAs confirmed that they were identical rules (Figure 11).

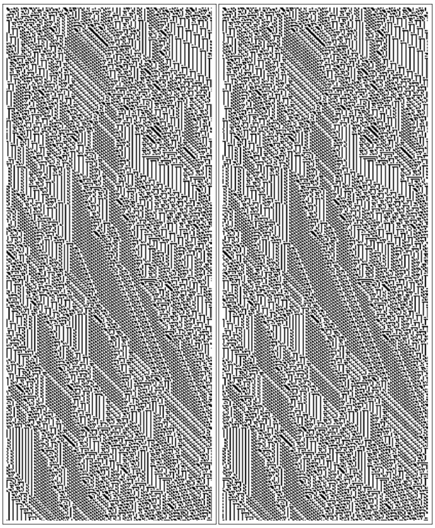


Figure 11. A comparison between rule 73 in LDCA-1-2 (left) and rule 20645 in the 3/2 rule space (right) with equivalent initial conditions. These two rules are identical (hence the identical evolutions) and can be derived from one another by converting from an ECA rule space to the 3/2 rule space and vice versa. As always, the LDCA-1-2 is plotted with an $i \to i - 1$ skew.

6. Methodology: Neighbor-Dependent Substitution System

Substitution systems form the backbone of most computational systems, but cannot in general emulate CAs. However, when substitution systems have rules that depend not only on the color of a single element, but also on the color of at least one of its neighbors, they display more complicated behavior [1]. In order to utilize this behavior, we must be able to identify the current state of a CA with an array or string of values and construct a substitution system for its behavior that is closed under evolution (so that as substitutions occur, no results arise that cannot be interpreted). Neighbor-dependent substitution systems are known to emulate certain CAs and could be helpful in proving the universality of rule 73 [1] (Figure 12).

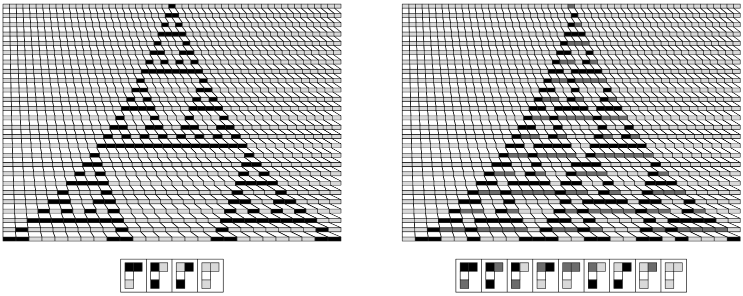


Figure 12. Neighbor-dependent substitution systems that emulate rules 90 (left) and 30 (right). The systems shown are examples of neighbor-dependent substitution systems with highly uniform rules that always yield one cell and correspond directly to known CAs [16].

7. Results: Characteristics of Rule 73

After a brief exploration of rule 73, several interesting phenomena were found, two of which follow. First, for rule 73 being evolved in an LDCA-1-y configuration, as y increases in value from 1, complex behavior arises at $y = 2$ and is extinguished after $y = 3$, giving rise to pseudo-chaotic behavior (Figure 13). An analysis of rule 73's behavior for $y > 3$ revealed that the Gaussian distribution of all possible states of rule 73 was not normally distributed. This implies that the chaotic nature of rule 73 at high values of y is not random, but rather is complex.

Increasing the value of y also results in a secondary effect (Figure 14). When viewing the state transition diagrams, one notices a repetition of structure, not unlike that mentioned in the introduction.

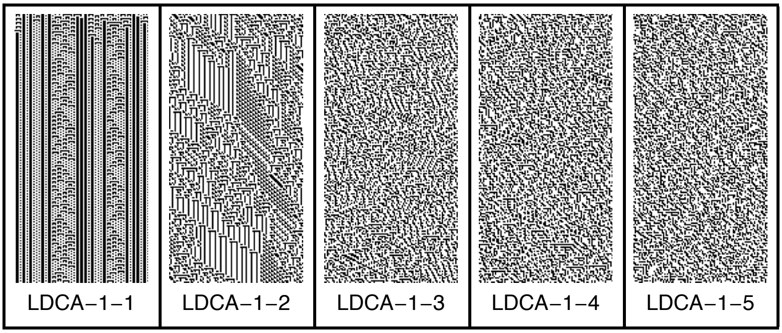


Figure 13. While evolving rule 73 in an LDCA-1-y configuration, as y increases in value from 1, complex behavior arises and is extinguished, giving rise to chaotic behavior that resembles white noise. The Gaussian distribution of all possible states is not normally distributed in these chaotic states, which implies that the chaotic nature of rule 73 at high values of y is not random, but is complex and difficult to perceive.

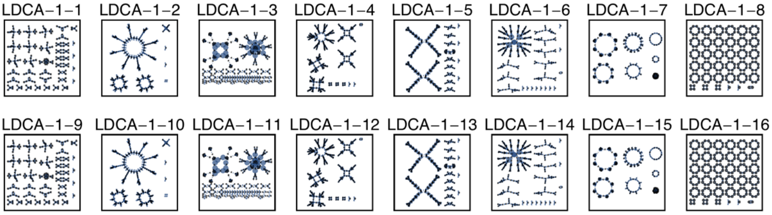


Figure 14. As rule 73 is applied in different configurations, the effect of having a high y value becomes apparent through state transition diagrams. Given rule 73 applied as LDCA-1-y-n, the state transition diagrams are repeated with a period of n as the y value increases, since the tape is cyclic [16].

8. Results: Particles

The gliders or particles that exist in rule 73 move with four different velocities (0, 1/4, 2/5, and 1) over a constant background (Figure 15). A single *block* of background is represented by 101100, 110010, or 001011 and has an evolutionary period of 3 and a spatial period of 6. In the naming convention for rule 73 in LDCA-1-2, a single block of background is represented with a “ - ”, half of a background block is denoted with a “ ’ ”, and a single particle is a letter (its name).

There are several particles in rule 73 that act as the building blocks for larger constructs and compound particles. These are called fundamental particles, and are the main focus of this exploration. All fundamental particles are organized and labeled by velocity and phase shift (or mass), with the mass ranging in value from 0 to +6 and representing the number of cells that the background is shifted to the right by the presence of the particle (Figure 16).

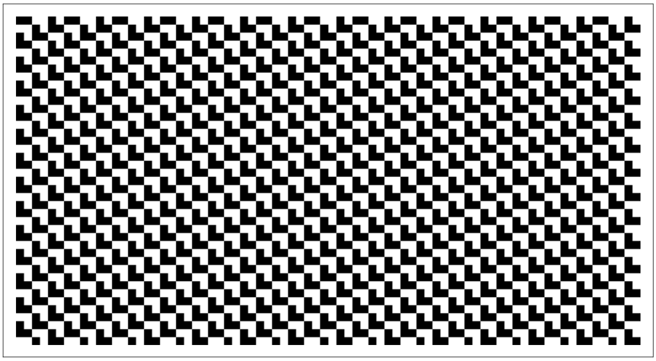


Figure 15. An image of the background of rule 73 in LDCA-1-2. The background resembles a surface tessellated with L-shaped units; a single row of background is represented by 101100, 110010, or 001011 and has period 3 (vertically) and spatial period 6 (horizontally). This image is scaled and partitioned, to better show the structures of cells in the background.

Particle	Velocity	Mass	Period	Particle	Velocity	Mass	Period
A	0	0	3	F	1	2	2
B	0	5	3	$\overline{\text{F}}$	1	3	4
C	1 / 4	1	8	G	1	0	2
D	2 / 5	4	5	H	1	3	2
E	2 / 5	2	15				

Figure 16. Table of all fundamental particles in rule 73 in the LDCA-1-2 configuration. All fundamental particles are organized and labeled by velocity, phase shift (or mass), and period. The mass ranges in value from 0 to +6 and represents the number of cells that the background is shifted to the right by the presence of the particle.

The structures in Figures 17 through 19 represent the nine fundamental particles and are listed in order of increasing velocity. Particles A, B, and C have velocities of 0 and 1 / 4. Particles D and E are the only fundamental particles with velocity 2 / 5. Particles F, $\overline{\text{F}}$, G, and H travel with velocity 1, or at the speed of light.

https://doi.org/10.25088/ComplexSystems.23.2.93

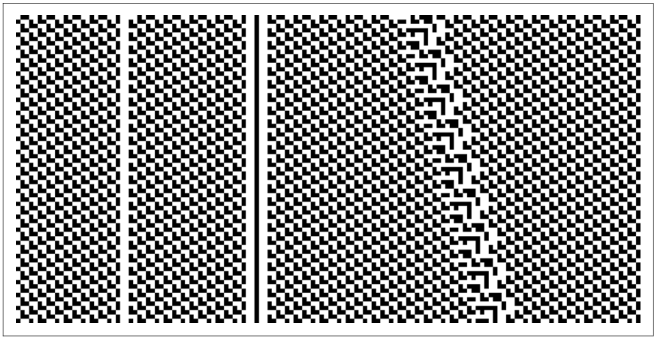


Figure 17. Particles A, B, and C; velocities 0, 0, and 1 / 4.

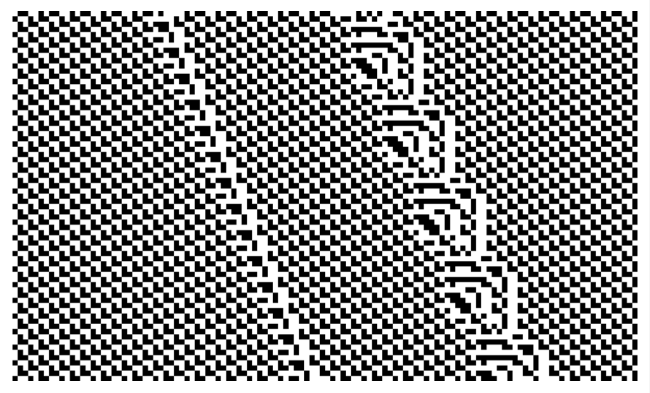


Figure 18. Particles D and E; velocities 2 / 5, 2 / 5.

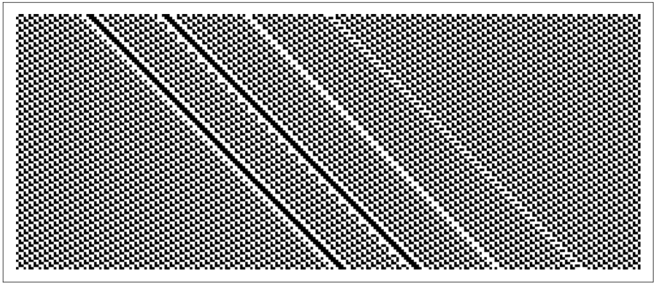


Figure 19. Particles F, $\overline{\mathrm{F}}$, G, and H; velocities of 1 for all.

However, merely placing two fundamental particles next to each other on a tape cannot create certain compound particles. For example, in the cases of G'G and G'H, the second particle in the pair must be shifted vertically by one evolutionary step (Figures 20 and 21).

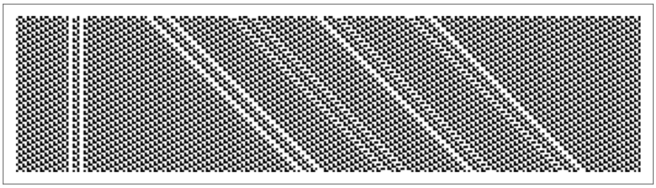


Figure 20. Particles B'B, G'G, H'H, G'H, and H'G. These are unique compound particles that must be formed by vertically shifting one of the two particles before placing them adjacent.

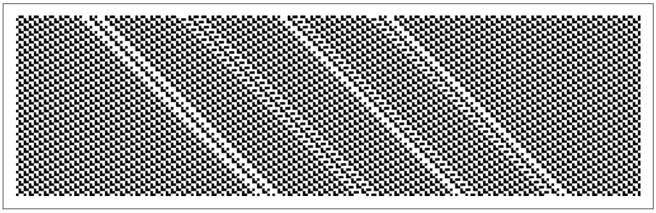


Figure 21. Particles GG, HH, GH, and HG. These are documented due to their prevalence in collisions and systems made from rule 73.

9. Results: Collisions

This study found all viable collisions between pairs of fundamental particles A, B, C, D, E, F, $\overline{F}$, G, and H. These do not include repeated or compound particles such as AA, ... FF, GG, G'G, HH, H'H, GH, HG, G'H, or H'G.

Some collisions were found to have varying resulting particles, due to their reactants interacting at different distances away from each other. Any collision between two fundamental particles in which the spacing between the particles does not affect the result is denoted by A_B for any particles A and B. The "_" in between particles' names implies that the spacing in between the particles does not affect the outcome of the collision. However, in the collisions that have integers between the particles' names, the integer represents the number of spaces between the two particles in that specific collision. A space is a full spatial period of the background and consists of six consecutive cells. The integer must first be evaluated in a modular function that is specific to the collision. Figure 22 shows a table of all viable collisions between fundamental particles.

Particle A	Particle B	Particle C	Particle D	Particle E
A_C	B_C	C_D	D_F	E0F
A_D	B_D	C_E	D0$\overline{F}$	E1F
A_E	B_E	C_F	D1$\overline{F}$	E2F
A_F	B_F	C_$\overline{F}$	D_G	E0$\overline{F}$
A0$\overline{F}$	B0$\overline{F}$	C_G	D_H	E1$\overline{F}$
A1$\overline{F}$	B1$\overline{F}$	C_H		E2$\overline{F}$
A_G	B_G			E0G
A_H	B_H			E1G
				E2G
				E_H

Figure 22. Table of all possible collisions between fundamentals.

10. Discussion: Block Emulation

Using rule 20645 in the 3 / 2 rule space, several rules that rule 73 can emulate through block emulation were identified, with blocks ranging in size from 0 to 16 cells [21]. To search the emulated rules for signs of universality, the rule numbers for ECA rules 110 and 193 were identified in the 3 / 2 rule space (Figure 23). Using the procedure in Section 5, it was found that rule 110 is equivalent to rule 232903/2, and rule 193 is equivalent to rule 614453/2.

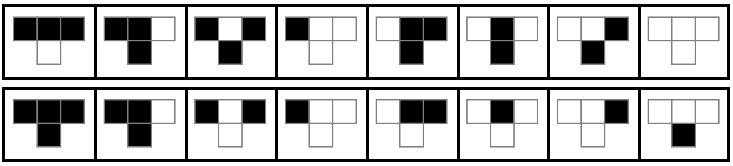


Figure 23. Rules 110 (top) and 193 (bottom) are equivalent and universal. They are rules 232903/2 and 614453/2, respectively.

This form of block emulation was continued until the block size was 30 cells, but none of the emulated rules were 23290 or 61445, so no desired result was generated. Additionally, as the emulation approached a block size of 40 cells, the estimated runtime jumped incredibly high, and the block emulation became limited by the amount of physical memory available. This prevented consistency in results with any block larger than 30 cells and greatly handicapped the feasibility of proving the universality of rule 73 through this method. Figure 24 shows all of the rules that rule 20645 emulates up to a block size of 16 cells.

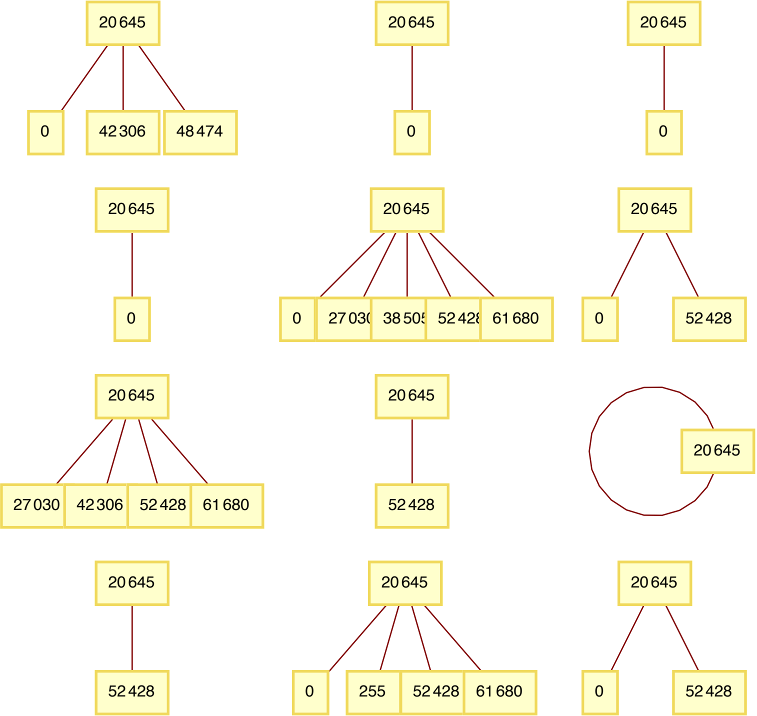


Figure 24. A diagram for all of the rules in the 3 / 2 rule space that rule 20645 emulates up to a block size of 16 cells. Each tree of rules represents a different block size, ranging from 0 to 4 cells on the first row, 5 to 9 cells on the second row, 10 to 13 cells on the third row, and 14 to 16 cells on the last row [21].

11. Discussion: Neighbor-Dependent Substitution System

Despite the constraints of the previous method, universality can be proven in a variety of other ways. For example, the neighbor-dependent substitution system [1] was considered. A particle detector that was created for rule 73 in LDCA-1-2 returns a string of particle names and background that resembles A----B--------C-GG-----H-G at

https://doi.org/10.25088/ComplexSystems.23.2.93

any evolutionary step. Following suit, a neighbor-dependent substitution system could be constructed that takes a similar string as an input and substitutes pairs of colliding particles with their outputs. The program should continue to collide particles until there are no legitimate pairs of particles left and then return a list of past states at every collision.

In the neighbor-dependent substitution system, the spacing between any consecutive particles in a collision was replaced with an integer signifying the number of background rows and evaluated with modular functions, so that large numbers of background cells would not cause the program to malfunction. For every spacing-dependent collision, the substitution system would classify the result of the collision using the integer between the particles and replace the pair with a corresponding result sandwiched between two additional spacing values, to account for background that was lost as the string was modulated. Then after every substitution, any adjacent numbers of background cells in the string were added so that they behaved as a singular spacing. The modular function that was applied to the spacing was either mod 2 or mod 3, depending on the pair of particles involved.

Unfortunately, this neighbor-dependent substitution system proves inconclusive. The set of rules that dictated the substitutions in the system should have stayed restricted to certain usable collisions, since some collisions resulted in a particle generator that created an infinite number of products. However, the rules were unable to stay restricted, as particles that had been excluded from the list eventually resulted from collisions and were added to the rule set. New collisions had to be added to the substitution system, and ultimately, the neighbor-dependent substitution system failed to simplify any string of particles.

12. Discussion: Collision System

Finally, collision G'G_B'B (which returns particles B'B and G) and collision G_B'B (which returns particle G'G) were considered. Using these collisions, a system was constructed that consists of two different substitution rules $\{\{AB \to B_1C\}, \{CB \to A\}\}$, in which alternating rows of G'G and G are colliding with B'B (Figure 25).

In every collision, the G or G'G output is released with a three-cell shift to the right, while the B'B output is shifted six cells to the left. This means that for every pair of G_B'B and G'G_B'B collisions, B has an overall shift of six cells to the right, and the resulting G particle is shifted six cells to the right.

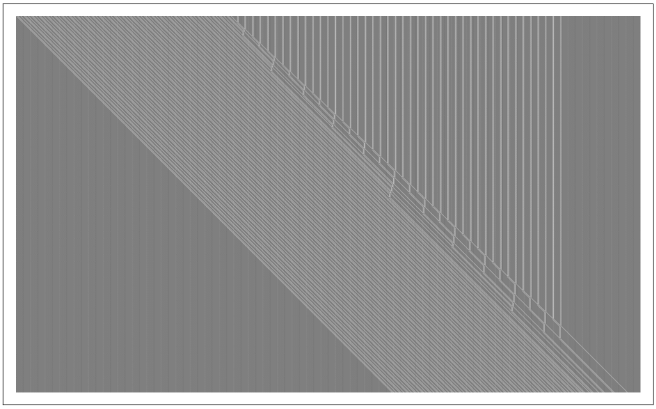


Figure 25. A chain of collisions between alternating rows of G’G and G with B’B [3]. The system consists of two different substitution rules {{AB → B_1C}, {CB → A}}.

As is observable, there are three points in total (on Figure 26) where all the particles that are colliding with B’B are particle G’G (except for the G that annihilates the B’B at the end), and they are all converted at once into particle G. At those points, the first B’B converts three G’G particles, the second converts five, and the third converts seven. These G’G conversion collisions count consecutive odd numbers as the system progresses, in the pattern 3, 5, 7, $\ldots 2n+1$. Additionally, counting the collisions between the G’G conversion points reveals interesting results, as can be seen in this chronological list of collisions, where each number represents how many collisions the B’B particle endures at that relative location (“*” is a G’G conversion point):

1 2 1 * 1 2 1 3 1 2 1 * 1 2 3 2 1 4 1 2 3 2 1 * 1 2 3 4 3 2 1 5 1
2 3 4 3 2 1 * ....

This string of numbers can be thought of as an inorder traversal of a series of binary trees [22]. When counting the number of collisions in which B’B interacts with k number of particles, it is found that the sums are of the form 2^{n-k+1}. For example, the total number of collisions in which B’B interacts with one particle, in between G’G conversion points, follows the pattern 2, 4, 8, $\ldots 2^n$.

It was also found that counting the total number of interactions of any type in between the G’G conversion points results in a progression of Eulerian numbers, which follows the series 4, 11, 26, 57, 120, 247, 502, ... [23, 24]. Interestingly though, the interactions between G’G conversion points do not define the standard Eulerian numbers of series A008292 [25, 26], but instead correspond to the values of the second column ($k = 2$) of the standard Euler triangle (which define series A000295) [24]. Thus, through the G, G'G, and B'B collision system, sequence A000295 is inherently related to the mathematical properties of complete binary trees. This ability of rule 73 in LDCA-1-2 to associate the behaviors of binary trees with Eulerian numbers (in series A000295 of [24]) can in turn provide valuable insight to unsolved problems such as those in [27] and lead to future mathematical exploration.

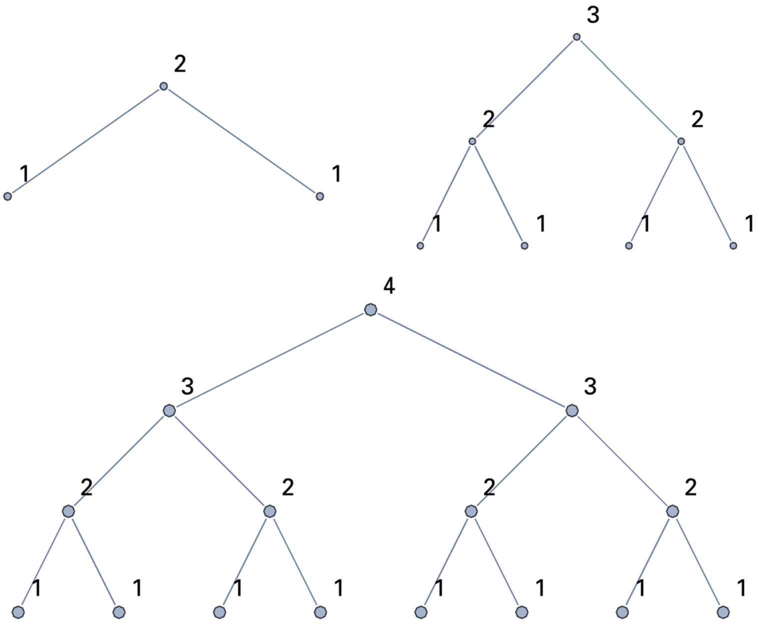


Figure 26. Complete binary that dictates the progression of collisions in the G'G and G with B'B collision system [22].

13. Future Research

In future research, it is suggested that the connections between binary trees and Eulerian numbers be studied through the use of rule 73, not exclusively through mathematical expression (as it has been studied in the past). With an LDCA approach, we may be able to better understand, both technically and conceptually, the mathematical nature of both constructs and gain an insight into problems similar to those presented by Baril and Pallo [27]. By utilizing the properties of rule 73 in LDCA-1-2, it is possible that further mathematical exploration could prove fruitful.

The compound particles of rule 73 should be studied in more detail, so that a functioning neighbor-dependent substitution system might be generated. With more complex gliders, the behavior of collisions may be diverse enough that a defined set of rules can be used to evolve and simplify a string of particles, which would mean that a neighbor-dependent substitution system could become a viable option to emulate other universal systems.

The block emulation of rule 73 as rule 20645 in the 3 / 2 rule space will be more feasible to study in the future. While the scope of this study was limited by physical constraints, it is likely that future attempts at proving the universality of rule 73 could make more progress with the block emulation of rule 20645 with further code optimization and improved hardware and succeed in emulating rules with block sizes much larger than six cells [21].

Besides making progress on methods already used in this study, any more research that is done on rule 73 in LDCA-1-2 should have an emphasis on the emulation of a cyclic tag system. In a cyclic tag system, a standard tag system is applied to an initial condition in sequential order and in a cyclic fashion that loops back to the front of the tape [19] (Figure 27). The discovery of the G, G'G, and B'B collision system is a large step forward in the process of emulating a cyclic tag system with rule 73, since it contains types of collisions that were used in Cook's proof of universality for rule 110 [19].

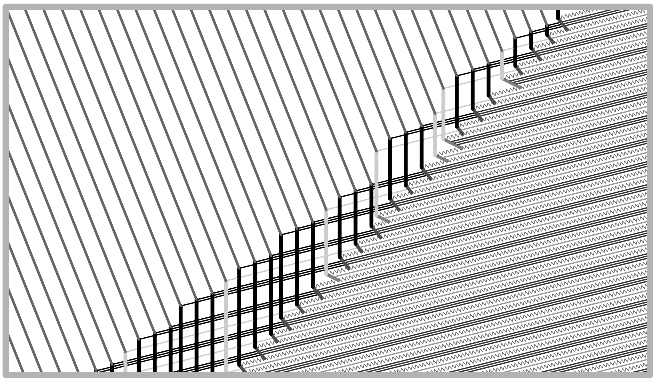


Figure 27. An example of a CA (rule 110) emulating a universal cyclic tag system [19]. Some of the collisions represented to the left bear resemblance to the G'G, G, and B'B collisions that were found in this study.

Finally, it is advised that more research be conducted on the properties of LDCAs. The effects of changing rule 73's configuration from LDCA-1-1 (ECA) to LDCA-1-2 have been significant and have turned rule 73 into a potentially universal CA. In the future, additional research should be conducted on the effects of different LDCA configurations on universality. Hopefully, the exploration of LDCAs will bring to light new possibilities and help further current knowledge of evolving computation systems.

Further exploration of rule 73 in LDCA-1-2, uncovering of its applications to problems in other fields of research, and showing of its candidacy for computational universality will assist in the expansion and improvement of modern computation and mathematical modeling.

References

[1] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002 pp. 23–60, 78–81, 95, 112, 187–192, 644–675, 865–866, and 927.

[2] S. Wolfram, "Statistical Mechanics of Cellular Automata," *Reviews of Modern Physics*, **55**(3), 1983 pp. 601–644. doi:10.1103/RevModPhys.55.601.

[3] S. Maerivoet and B. de Moor, "Cellular Automata Models of Road Traffic," *Physics Reports*, **419**(1), 2005 pp. 1–64. doi:10.1016/j.physrep.2005.08.005.

[4] B. Chopard and M. Droz, *Cellular Automata Modeling of Physical Systems*, Cambridge: Cambridge University Press, 1998 pp. 1–18.

[5] G. Ermentrout and L. Edelstein-Keshet, "Cellular Automata Approaches to Biological Modeling," *Journal of Theoretical Biology*, **160**(1), 1993 pp. 1–37.

[6] G. Martinez, "A Note on Elementary Cellular Automata Classification," *Journal of Cellular Automata*, **8**(3–4), 2013 pp. 233–259.

[7] S. Wolfram, "Universality and Complexity in Cellular Automata," *Physica D: Nonlinear Phenomena*, **10**(1–2), 1984 pp. 1–35. doi:10.1016/0167-2789(84)90245-8.

[8] R. Alonso-Sanz, *Cellular Automata with Memory*, Philadelphia: Old City Publishing, 2008 pp. 1–60.

[9] R. Alonso-Sanz and L. Bull, "Elementary Cellular Automata with Minimal Memory and Random Number Generation," *Complex Systems*, **18**(2), 2009 pp. 195–213. http://www.complex-systems.com/pdf/18-2-2.pdf.

[10] R. Alonso-Sanz and M. Martín, "Elementary Cellular Automata with Memory," *Complex Systems*, **14**(2), 2003 pp. 99–126. http://www.complex-systems.com/pdf/14-2-1.pdf.

https://doi.org/10.25088/ComplexSystems.23.2.93

[11] R. Alonso-Sanz and M. Martín, "Two-Dimensional Cellular Automata with Memory: Patterns Starting with a Single Site Seed," *International Journal of Modern Physics C,* **13**(1), 2002 pp. 49–65. doi:10.1142/S0129183102002973.

[12] P. Letourneau, "Statistical Mechanics of Cellular Automata with Memory," Master of Science thesis, Theoretical Physics, The University of Calgary, 2006, pp. 1–31.

[13] P. Letourneau. "Elementary Cellular Automata with Memory." NKS2006 Wolfram Science Conference. (Feb 21, 2014) http://www.wolframscience.com/conference/2006/presentations/materials /letourneau.pdf.

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

[15] S. Wolfram, "Cellular Automata," *Los Alamos Science*, **9**, 1983 pp. 2–21.

[16] S. Wolfram, *Theory and Applications of Cellular Automata,* Singapore: World Scientific, 1986 pp. 485–557.

[17] "Rule 73" from Wolfram|Alpha—A Wolfram Web Resource. http://www.wolframalpha.com/input/?i=Rule+73.

[18] "Rule 109" from Wolfram|Alpha—A Wolfram Web Resource. http://www.wolframalpha.com/input/?i=Rule+109.

[19] 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.

[20] R. Penrose, "Algorithms and Turing Machines," in *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*, Oxford: Oxford University Press, 1989 pp. 30–73.

[21] J. Riedel, "Quantifying Emulation for Computation Universality in Cellular Automata," project from Wolfram Science Summer School, 2013. http://www.wolframscience.com/summerschool/2013/alumni/riedel.html.

[22] A. A. Puntambekar, *Data Structures*, Pune, India: Technical Publications, 2010 pp. 254–273.

[23] M. Abramowitz and I. A. Stegun, eds., "Bernoulli and Euler Polynomials and the Euler–Maclaurin Formula," in *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*, 9th printing, New York: Dover, 1972 pp..804–806.

[24] N. J. A. Sloane. "Eulerian Numbers." The On-Line Encyclopedia of Integer Sequences. (Feb 22, 2014) http://oeis.org/A000295.

[25] J. H. Conway and R. K. Guy, *The Book of Numbers*, New York: Copernicus, 1996 pp..110–111.

https://doi.org/10.25088/ComplexSystems.23.2.93

[26] N. J. A. Sloane. “Triangle of Eulerian Numbers.” The On-Line Encyclopedia of Integer Sequences. (Feb 22, 2014) http://oeis.org/A008292.

[27] J. L. Baril and J. M. Pallo, “The Pruning-Grafting Lattice of Binary Trees,” *Theoretical Computer Science*, **409**(3), 2008 pp. 382–393. doi:10.1016/j.tcs.2008.08.031.