Investigation of Rule 73 as a Case Study of Class 4 Long-Distance Cellular Automata

Lucas Kang

Personal Section

In my sophomore year, I took a post-AP class named Computational Physics at my school. Given that I was simultaneously enrolled in my first computer science class that year, I had very little prior experience to build upon. Nevertheless, I dove headfirst into what turned out to be one of my favorite classes ever.

For my final project, I made a simulator for the behavior of charged particles in Wolfram Mathematica, a programming language I had learned for the class. While doing so, I stumbled upon Stephen Wolfram's book, *A New Kind of Science*. It was full of incredible graphics and information that I couldn't have imagined, and presented to me a whole new world of computer science. *A New Kind of Science,* or NKS, contained a systematic study of computability as well as evolving systems such as Turing machines and cellular automata. I had never before considered what might be at the intersection between mathematics and computer science, and I knew immediately that I wanted to learn more.

That summer, I applied to and was accepted to the Wolfram Science Summer School (WSSS) – WSSS2012 was hosted at Curry College in Milton, Massachusetts. At WSSS2012, I met Stephen Wolfram, members of the Wolfram Science team, and numerous computer science enthusiasts from around the world, all with unique and interesting backgrounds. It was after talking to Dr. Wolfram for the first time that I decided to study long-distance cellular automata, or LDCA, a field of cellular automata that had not been extensively documented before. I began by created a nomenclature for LDCA, and started to study their basic characteristics. The people I met at WSSS2012 provided helpful guidance, as they pointed me in the right direction in a field I had limited knowledge about.

After continuing my project through my junior year, I decided to return to the Wolfram Science Summer School the next summer. I was accepted again, and attended WSSS2013 at Bentley University, this time in Waltham, Massachusetts. During my second summer, I began tackling the challenge of computational universality. After deciding on a systematic approach to studying LDCA, I chose to focus on Rule 73 in LDCA-1-2 due to its Class 4 behavior. By the end of September, I had made significant progress in documenting all of Rule 73's characteristics, and proving its potential universality.

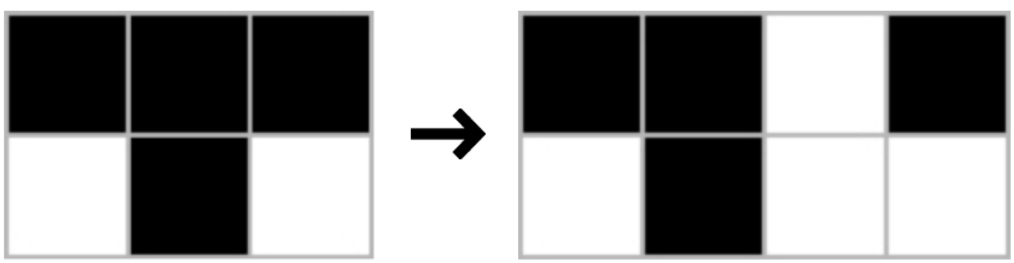
The 16 months that I spent on my research, while demanding, left me with a renewed appreciation for both computer science and mathematics. I came to realize that they were not the isolated, self-contained fields that I had learned about in school, but rather that they are connected, evolving, and constantly building upon each other. I would advise that other high school students with interests in science or mathematics keep this in mind. When conducting research, one should not be afraid to learn new things or venture out of one's comfort zone. Fields that may seem alien may ultimately be helpful to one's project.

Research Section

Cellular automata (CA) have been utilized for decades as discrete models of physical, mathematical, chemical, and biological systems. The most common form of

CA, the elementary cellular automaton (ECA), has been studied intensively in the past due to its simple form and versatility. However, ECA are constrained to evolve according to a neighborhood of adjacent cells, which limits their sampling radius and the environments in which that they can be used.

The purpose of my study was to explore the behavior of one-dimensional CA in configurations other than that of ECA. Namely, “long-distance cellular automata” (LDCA), a construct that had been described in the past but never studied. In my paper, LDCA were described by the notation LDCA-x-y-n, where x and y represent the spacing between the cellular automaton’s target cell and its left and right neighbors, and n denotes the length of the initial tape for LDCA with tapes of finite size. Additionally, basic characteristics of LDCA were explored such as universal behavior, the change in prevalence of complexity with varying neighborhoods, and qualitative evolutionary behavior as a result of configuration.



ECA (LDCA-1-1) LDCA-1-2 (rightmost cell 2 blocks away)

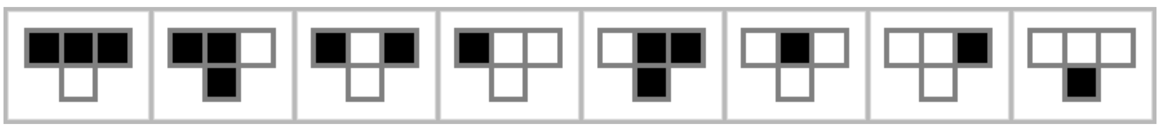
The above figure illustrates the layout of the three sampled cells and one output cell in an ECA (left), and a corresponding image for the configuration of LDCA-1-2-n (right).

The 2-color, 2-state cellular automaton I studied in the LDCA-1-2 configuration is known as “Rule 73” according to Wolfram’s numbering scheme. In the evolution of Rule 73, each cell is in one of two states {0, 1}, and since the rule is being applied to a LDCA-

1-2 configuration, at each discrete time step every cell synchronously updates itself according to the value of itself and its nearest neighbors: $F(C_{i-1}, Ci, C_{i+2})$, where F is the following function:

$F(1, 1, 1) = 0$	$F(1, 1, 0) = 1$	$F(1, 0, 1) = 0$	$F(1, 0, 0) = 0$
$F(0, 1, 1) = 1$	$F(0, 1, 0) = 0$	$F(0, 0, 1) = 0$	$F(0, 0, 0) = 1$

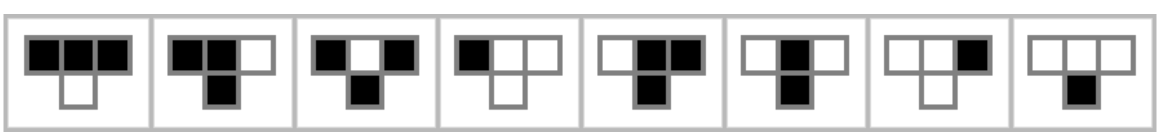
This table depicts the evolutionary substitution "rules" of Rule 73. Rule 73 and Rule 109 are equivalent through both left-right and color equivalence.



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. 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:

$G(1, 1, 1) = 0$	$G(1, 1, 0) = 1$	$G(1, 0, 1) = 1$	$G(1, 0, 0) = 0$
$G(0, 1, 1) = 1$	$G(0, 1, 0) = 1$	$G(0, 0, 1) = 0$	$G(0, 0, 0) = 1$

This table depicts the evolutionary substitution "rules" of Rule 109. Rule 109 and Rule 73 are equivalent through both left-right and color equivalence.



In order to understand how Rule 73 behaved, I decided to document all of its localized structures. I spent a few weeks isolating and categorizing all of Rule 73's gliders, or "particles" as I referred to them, and recorded all possible outcomes between particle pairs. There were several particles in Rule 73 that act as the building blocks for larger constructs, or "compound particles." These I called "fundamental particles," and they were the main focus of my 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.

Particle	Velocity	Mass	Period	Particle	Velocity	Mass	Period
A	0	0	3	F	1	2	2
B	0	5	3	Fbar	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				

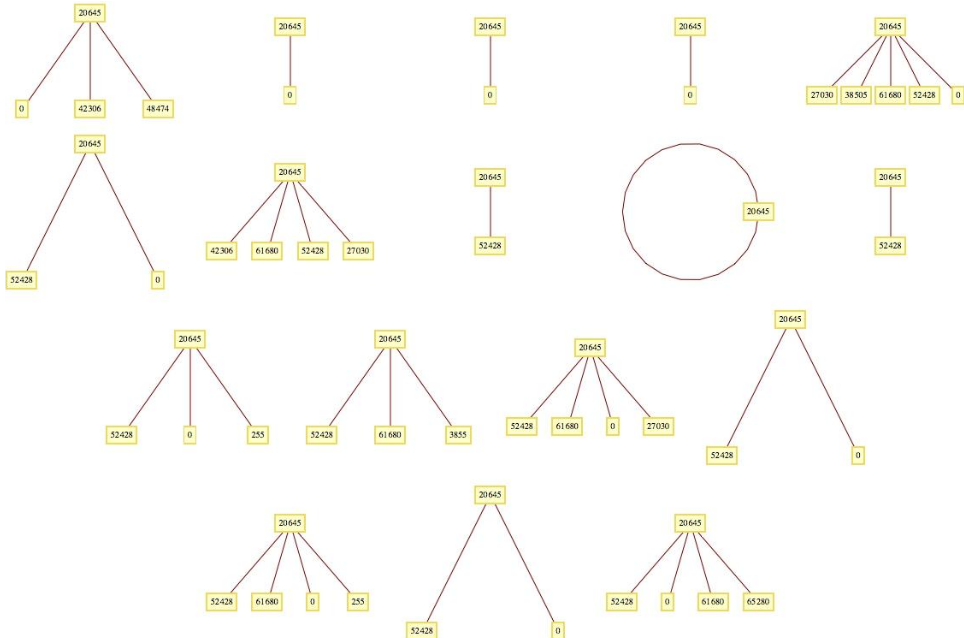
Below is a table of all viable collisions between fundamental particles.

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	D0Fbar	E1F
A_E	B_E	C_F	D1Fbar	E2F
A_F	B_F	C_Fbar	D_G	E0Fbar
A0Fbar	B0Fbar	C_G	D_H	E1Fbar
A1Fbar	B1Fbar	C_H		E2Fbar
A_G	B_G			E0G
A_H	B_H			E1G
				E2G
				E_H

After completely characterizing Rule 73 through an investigation of its state transition diagrams and evolution (of particles/collisions), I decided to attempt to prove that Rule 73 was Turing universal. Universal computational systems are those that are theoretically capable of emulating any other system. 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.

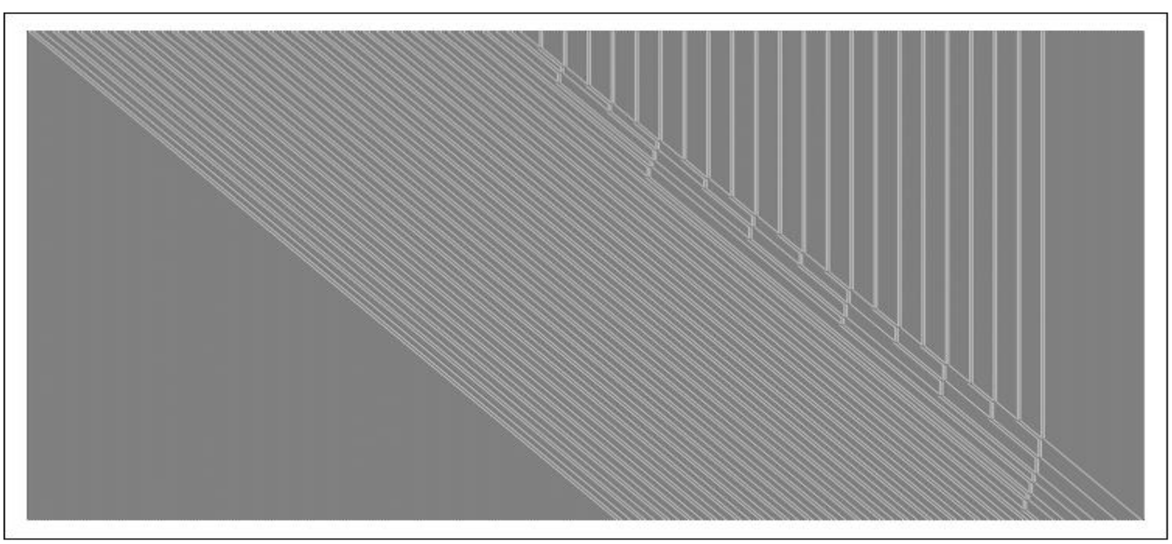
The proving of a computational system's universality is usually done through the emulation of another system, previously known to be universal. As a result of the Church-Turing Thesis, Turing machines have been defined as universal. (Church's "effective calculability", Turing's "computability", Post's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

Among other methods, I attempted to use block emulation to show that Rule 73 could behave like Rule 110; the emulation of Rule 110 would be enough to prove the universality of Rule 73. First I converted Rule 72 in LDCA-1-2 into Rule 20645 in the 3/2 rule space. Then, I began to use increasingly larger block sizes to find out which cellular automata rules that Rule 20645 was capable of emulating. Here we can see 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 on the last row



But my block emulation was eventually limited by the processing power of my laptop, and so I moved onto searching for universal behavior with particle collisions. I attempted to construct a universal neighbor-dependent substitution system.

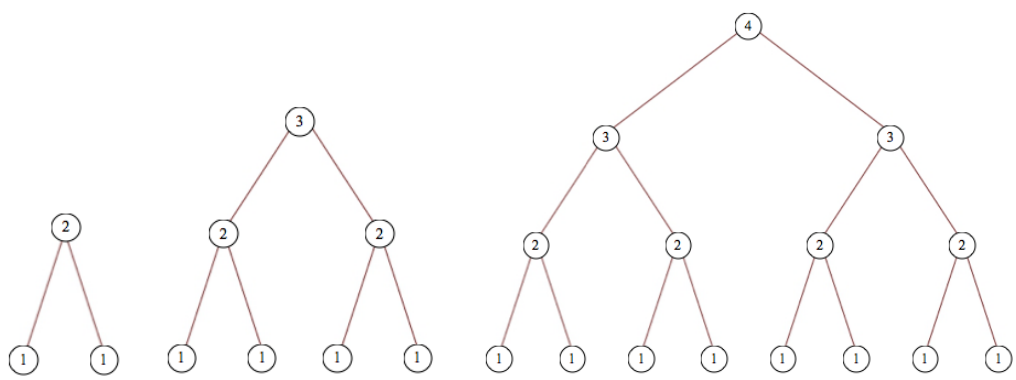
Using collision G’G_B’B which emits particles B’B and G, and collision G_B’B which emits particle G’G, I was able to construct a system that consists of two different substitution rules {{AB → B_1C}, {CB → A}}, and plot the behavior of said system in which alternating rows of G’G and G are colliding with B’B.



I observed that there are 3 points in total where all the particles that are colliding with B’B are of particle type 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 converted 3 G’G particles, the second converts 5, and the third converts 7. These “G’G conversion” collisions count consecutive odd numbers as the system progresses, in the pattern 3, 5, 7, … 2n+1. Additionally, counting the number of 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 * …

However, I then noticed that this string of numbers could be thought of as an inorder traversal of a series of binary trees. And, when counting the number of collisions in which B’B interacts with k number of particles, I 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 1 particle, in between G’G conversion points, follows the pattern 2, 4, 8, … 2^n.



I also found that when counting the *total* number of interactions of any type in between the G’G conversion points, one is left with a progression of Eulerian numbers, which follows the series, 4, 11, 26, 57, 120, 247, 502, … Interestingly though, the interactions between G’G conversion points don’t define the standard Eulerian numbers, but instead correspond to the values of the second column (k = 2) of the standard Euler Triangle. While this didn’t prove universality in Rule 73, the ability of Rule 73 in LDCA-1-2 to associate the behaviors of binary trees with Eulerian numbers could provide valuable insight to unsolved problems, and lead to future mathematical exploration.

In future research, it is suggested that the compound particles of Rule 73 be studied in more detail, so that a functioning neighbor-dependent substitution system might be generated. With more complex particles, the behavior of their collisions may be diverse enough that a defined set of rules can be used to evolve or simplify a string of particles. This would verify that a neighbor-dependent substitution system could emulate other universal systems.

And, 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 computational 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.

Focusing mainly on purely Class 4 behavior in LDCA-1-2, my exploration found that Rule 73 could potentially be Turing universal through the emulation of a cyclic tag system, and began to explore the applications of LDCA. My project also paved the way for future exploration of LDCA with larger neighborhoods, and illustrated a connection between the mathematics of binary trees and Eulerian numbers that may provide insight into unsolved problems.