Fractal Replication in Time-manipulated One-dimensional Cellular Automata

Sugata Mitra[*]
Sujai Kumar[†]
Centre for Research in Cognitive Systems, NIIT Ltd.,
Synergy Building, IIT Campus,
New Delhi 110016, India

Properties of elementary one-dimensional cellular automata (CAs) have been studied extensively in the past by varying the number of states each cell can take, the neighborhood of the cell, or the transition rules by which each cell is updated. This paper describes a previously untried variation on a CA system, where each cell is able to anticipate its state one step in the future, and the entire system is allowed to revisit the past over many iterations. Manipulating the time domain in this way allows the CA to exhibit complex fractal replication behavior. Any configuration of active cells can be replicated endlessly while remaining constrained in a self-similar layout.

1. Introduction

A cellular automaton (CA) consists of discrete cells arrayed in a specific geometry. Each cell can be in one of k finite states, and each cell's state is updated on every time step according to a deterministic rule based on the values of the neighboring cells and the value of the cell being updated.

CAs have been studied in depth over the past several decades [1–5]. Although most researchers have examined such automata by manipulating the number of states, update rules, initial conditions, or the structure (changing a square grid to a triangular one, treating it as a torus, etc.) we have chosen to concentrate on some computational experiments in manipulating the time domain of a one-dimensional CA (1-CA). Thus, although our starting point is a 1-CA, our system is not a true CA by any means as we allow the individual sites to peek ahead in the future, and we allow the system to go back to the "past."

The behavior exhibited by this modified system is interesting in that it acts as a fractal replicator—any starting shape (a bitmap of the word "ORDER" in Figure 1) is replicated endlessly, with the different copies

[*]Electronic mail address: SugataM@niit.com.
[†]Electronic mail address: sujai@ylog.org.

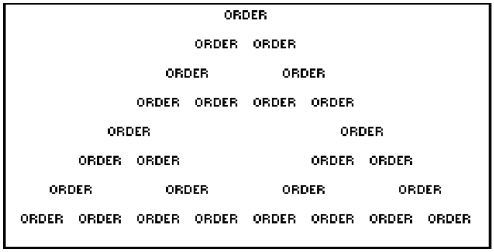


Figure 1. Fractal replication of a simple bitmap.

of this shape always constrained in a self-similar fractal layout. Fractal replicator CAs of this kind, sometimes called "Fredkin's replicators," are described in the context of two-dimensional cellular automata (2-CAs) [6–8], but fractal replication in 1-CAs seems to be a previously undocumented phenomenon.

This paper is organized as follows. Section 2 describes an elementary 1-CA and a set of experiments that progressively demonstrate the increasing complexity of the different systems that result as we manipulate the time domain.

In section 3, we analyze this modified 1-CA using a more standard 2-CA where the second dimension is treated as an analogue of time in the modified 1-CA. This conventional 2-CA also shows the same fractal replicator properties.

Although the emergence of self-similar patterns like Sierpinski's Triangle in 1-CAs is well documented [3, 9], we have not seen any descriptions of 1-CAs that replicate a two-dimensional shape or pattern. In our discussion section, we propose that looking ahead (and retracing steps) in a 1-CA allows the system to exhibit complex regularities (such as fractal replication) that are not possible in an elementary automaton.

2. Manipulation of time domain in a one-dimensional cellular automaton

2.1 Elementary one-dimensional cellular automata

An elementary 1-CA consists of a single row of square cells that are updated on each time step based on their states and on the states of their neighbors. We begin with the following two-state model having a neighborhood of radius 1.

- Number of states: $k = 2$.
- Number of neighbors on each side: $r = 1$.
- Update rule: $\text{Cell}_{t+1} = (\text{Left}_t + \text{Cell}_t + \text{Right}_t) \bmod 2$.

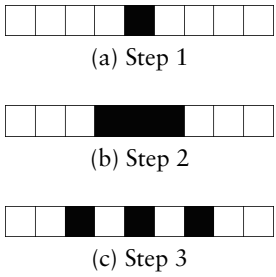


Figure 2. First three steps of an elementary 1-CA with $k = 2$, $r = 1$, and Rule 150.

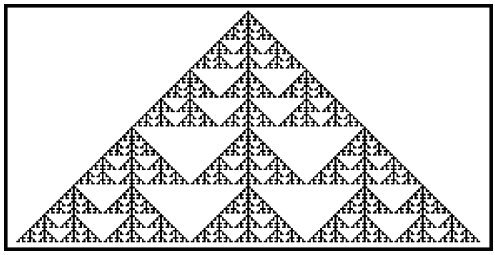


Figure 3. First 128 steps in the evolution of a 1-CA with Rule 150.

The rule in Figure 2 is also known as Rule 150 according to Wolfram's numbering system [9]. The numbering system describes the value of each cell on the next step, based on the values on the previous step of three cells—the left neighbor, the cell itself, and the right neighbor. There are $2^3 = 8$ possible combinations of three cells with binary values. If we canonically order these eight combinations as $\{1, 1, 1\}$, $\{1, 1, 0\}$, $\{1, 0, 1\}$, $\{1, 0, 0\}$, $\{0, 1, 1\}$, $\{0, 1, 0\}$, $\{0, 0, 1\}$, and $\{0, 0, 0\}$, then the updated cell value for each combination will be 1, 0, 0, 1, 0, 1, 1, and 0 respectively. Rule 150 is the decimal equivalent of 10010110, and there are 256 such rules possible (2^8).

When we begin with a single cell on, and lay out several successive time steps of this 1-CA one below the other, we get the well-documented self-similar nested triangles in Figure 3.

2.2 One-dimensional cellular automaton with "look-ahead"

We now move away from a classic 1-CA where each cell was updated based on the past values of the cell itself and its two neighbors. In our time-manipulated 1-CA, each cell is allowed to look ahead at its own future state. The new value of the cell still depends on three input values, but instead of the neighborhood consisting of the left neighbor, the cell itself, and the right neighbor, the neighborhood now becomes the left neighbor, the cell's own future state, and the right neighbor (Figure 4).

Here is the time-manipulated model.

- Number of states: $k = 2$.
- Number of neighbors on each side: $r = 1$.
- Update rule: $\text{Cell}_{t+1} = (\text{Left}_t + \text{Cell}_{t+1} + \text{Right}_t) \bmod 2$.
- Note: Cell_{t+1} will be 0 for every step in this case.

On starting with a single on cell, we get a perfect Sierpinski Triangle (or Pascal's Triangle) after 128 steps (Figure 5). There is nothing very surprising about this figure as other elementary 1-CA systems such as the well-documented Rule 90 (and Rules 18, 26, 82, 146, 154, 210, and 218) also result in exactly the same figure. Rule 150 with every center cell considered to have value 0 acts just like Rule 90, which takes the sum modulo 2 of the left and right cells, without regard to the middle cell.

2.3 Placement of an object in the future

Figures 3 and 5 show how a 1-CA evolves over time. Each row in these figures is a snapshot in time, with the future and past being visible at

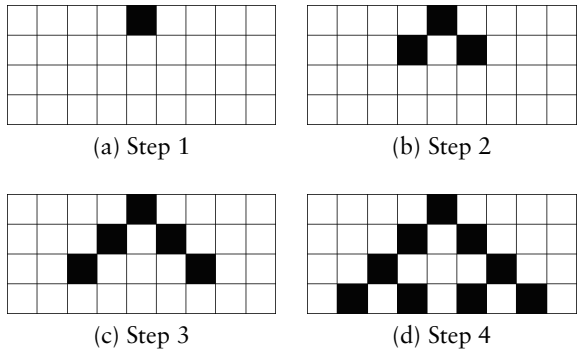


Figure 4. First four steps in the evolution of a time-manipulated 1-CA.

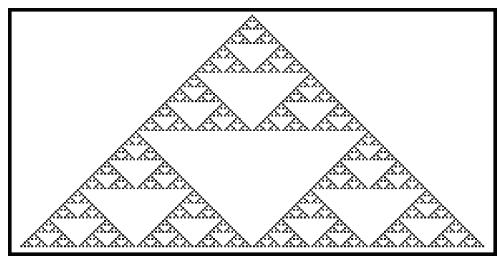
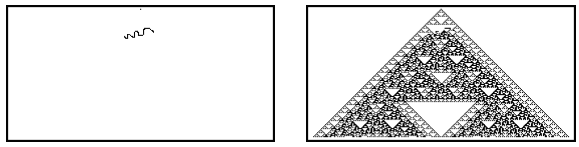


Figure 5. First 128 steps in the evolution of a modified 1-CA where each cell can look ahead.



(a) (b)

Figure 6. (a) Placing a squiggle in the "future" field of the 1-CA (b) results in a chaotic pattern after 128 steps.

the same time. If we now place an "object" (a configuration of active cells) such as a broken line, or a series of objects at different time points in the future (Figure 6(a)), we see the graceful growth of a Sierpinski triangle degenerate into a complex chaotic pattern once it "encounters" the objects in the future.

At first glance, the dark region created on encountering objects in the future (Figure 6(b)) might look like it can be characterized as class 4 behavior according to Wolfram's schema [9]—with patterns that border on the edge of chaos but occasionally show nested regularity (such as the repeated white triangles of different sizes). However, subsequent iterations of this CA (as described next) demonstrate that the behavior is very regular and not at all chaotic as it initially seems.

2.4 Iterations of a modified one-dimensional cellular automaton

After the modified 1-CA has finished a specific number of time steps (128 for the example in Figure 6(b)), we allow it to revisit the past and go back to step 1. It is difficult to come up with a physical analogue for this process but the computational and algorithmic specification is straightforward.

We take all the states (past and present) together as a two-dimensional field and update each row again according to our look-ahead rule, starting from the first row. The process of running the CA system through one complete set of time steps is defined as one iteration.

Many iterations of the CA result in a very interesting phenomenon—the system now acts as a fractal replicator. Any objects or cell configurations placed in the system will replicate infinitely—but the layout of the copies will be constrained to the fractal Sierpinski Triangle (Figure 7).

On the 16th iteration in Figure 7, we see many copies of the original active cell and the squiggly strand in a Sierpinski Triangle arrangement with some overlapping. On the 17th iteration, each solitary active cell and each squiggle causes darker, more chaotic regions to be formed similar to the formation in the first iteration. By the 32nd iteration, the field has sharpened again to show clear, nonoverlapping copies of the complete configurations of the single active cell and the squiggle, arranged fractally, but fewer in number than in the 16th iteration. Although

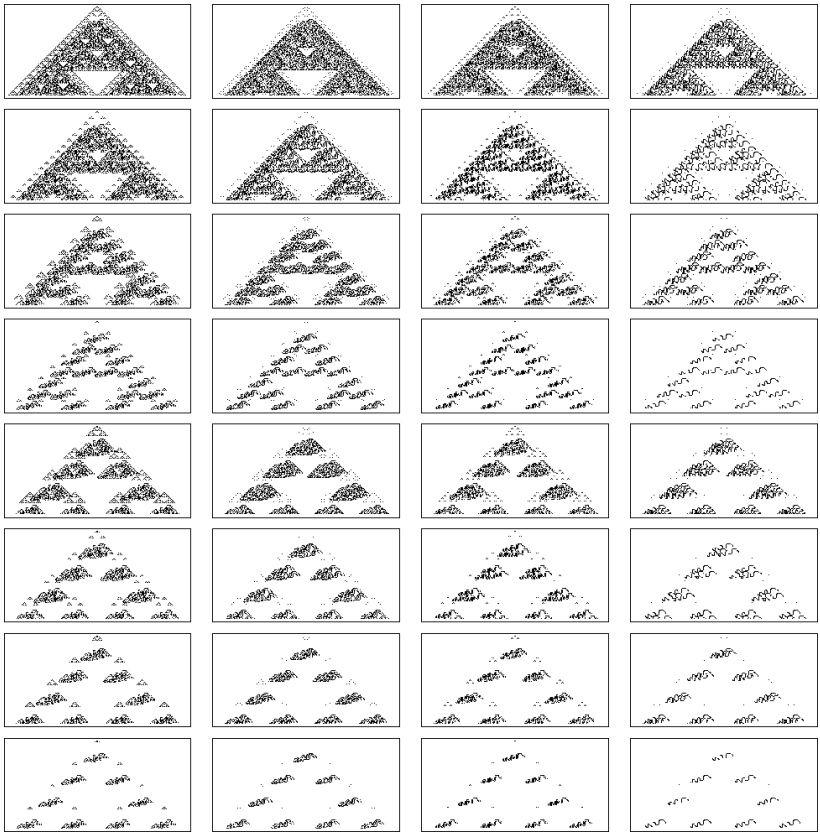


Figure 7. First 32 iterations (from left to right, top to bottom) of the future field in Figure 6.

replication seems to start by the 16th iteration, perfect nonoverlapping replication takes place only on the 32nd iteration.

If we iterate this system some more (Figure 8), we see the same types of behavior repeating at different scales. Moving from the 32nd to the 33rd iteration is similar to moving from the 16th to the 17th iteration. Similarly, by the time we get to the 64th iteration, we see only three perfect nonoverlapping copies of the original configuration. This reduction in the number of copies over iterations reaches the original configuration by the 128th iteration.

If there is no object in the future field of the CA, the iterations make no difference at all, and the Sierpinski triangle in Figure 5 remains unchanged.

Unlike proper CAs that assume infinite grids and infinite time steps, an iteration by its very nature restricts this CA to a certain number of time steps after which the system has to iterate. We chose 128 time steps

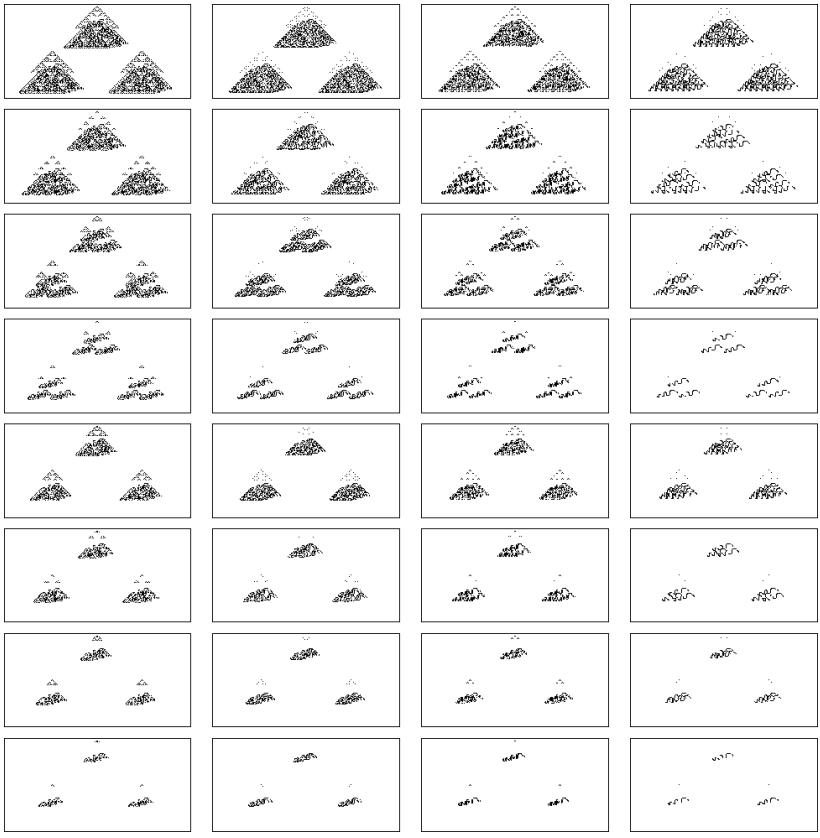


Figure 8. Iterations 33 to 64 (from left to right, top to bottom) of the future field in Figure 6.

because it is easy to visualize smaller graphs, but iterations with more time steps also exhibit the same behavior.

The number of iterations taken for the CA to create perfect nonoverlapped copies of a shape depends on the size of the shape. The simplest shape (a single black cell) takes one step to replicate, whereas larger shapes take more time.

The fractal replication behavior of our modified 1-CA remains the same even without the initial active cell. The only difference is that no copies of the initial cell are produced.

The complete configuration of the field in Figure 6(a) is 30 cells wide and 30 cells high, and it took 32 iterations for nonoverlapping replication to occur.

As the log–log plot in Figure 9 shows, an $N \times N$ square configuration of cells takes $2^{\lceil \mathrm{Log}_2 N \rceil}$ iterations to replicate.

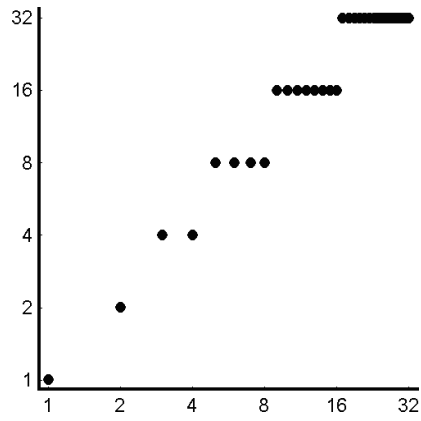


Figure 9. Number of iterations needed for a square pixel configuration of side N to replicate.

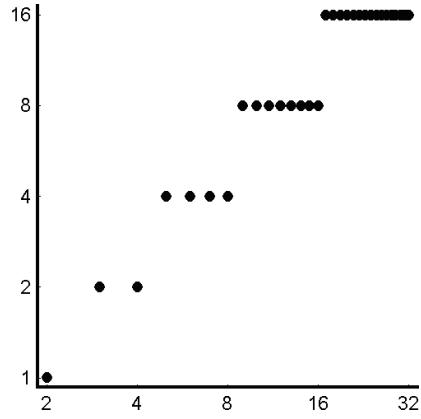


Figure 10. Number of iterations needed for a rectangular pixel configuration $1 \times N$ in size.

Rectangular configurations with dimensions M and N ($M < N$) will replicate in fewer iterations than $N \times N$ square configurations, but the formula remains a logarithmic step function as in Figure 9. As an extreme case, the number of iterations needed for rectangles of size $1 \times N$ is shown in Figure 10 and equals $2^{\lceil \mathrm{Log}_2 N - 1 \rceil}$ for $N > 1$.

The formula for the number of iterations needed for replication is an upper limit in a sense. As the 16th iteration in Figure 7 shows, fractal replication seems to be occurring earlier than expected for the 30×30 cell configuration that we started with. This is because the starting configuration (Figure 6(a)) is sparse, and we do not perceive the overlaps as being significant. If we start with a 30×30 cell smiley face

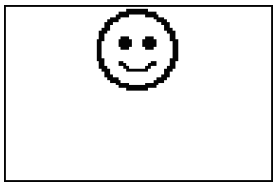


Figure 11. 30×30 smiley face as a starting configuration.

in Figure 11, then we clearly see that perfect nonoverlapping replicas only form on the 32nd iteration (Figure 12).

Shapes are fractally replicated even if we introduce them in between iterations. Figure 13 shows what happens when a 10×10 cell squiggle is introduced on the 10th step.

The 10×10 cell squiggle takes 16 iterations to replicate and maintains its identity as seen on the 26th iteration (16 iterations after its introduction). The 42nd (10 + 32) and 64th iterations of this system are shown in Figure 14. The 42nd clearly shows a replica of the squiggle while the 64th shows a replica of the smiley that was part of the initial field.

The properties of our modified 1-CA can be summarized as follows.

1. Any configuration of black cells in the field is replicated in a fractal layout.

2. Because the shapes grow at a fixed rate (one cell in each horizontal direction on each time step), the number of iterations required to replicate any shape is dependent on the size of the shape—larger shapes require more iterations (because they need to move further apart in order to have copies that do not overlap). An $N \times N$ square configuration of cells will take $2^{\lceil \mathrm{Log}_2 N \rceil}$ iterations to replicate. A $1 \times N$ rectangle takes $2^{\lceil \mathrm{Log}_2 N - 1 \rceil}$ iterations for $N > 1$.

3. Additional shapes introduced anywhere in the field or on any iteration will continue to replicate fractally even though their growth may look chaotic for a few iterations.

4. Shapes that overlap each other may look chaotic to begin with, but repeated iterations show that the shapes remain intact.

5. The field changes in a discontinuous way on the iteration after a sharp copy of the original configuration is seen.

An obvious question that comes to mind is whether this phenomenon is seen for every rule in a modified 1-CA or just for the equivalent of Rule 150.

In our update rule, the new state of each cell is given by the sum of the three cells being considered modulo 2. Of the 256 possible rules involving three cells (the cell being updated and its two neighbors), only one other rule, Rule 105, shows any kind of fractal replication.

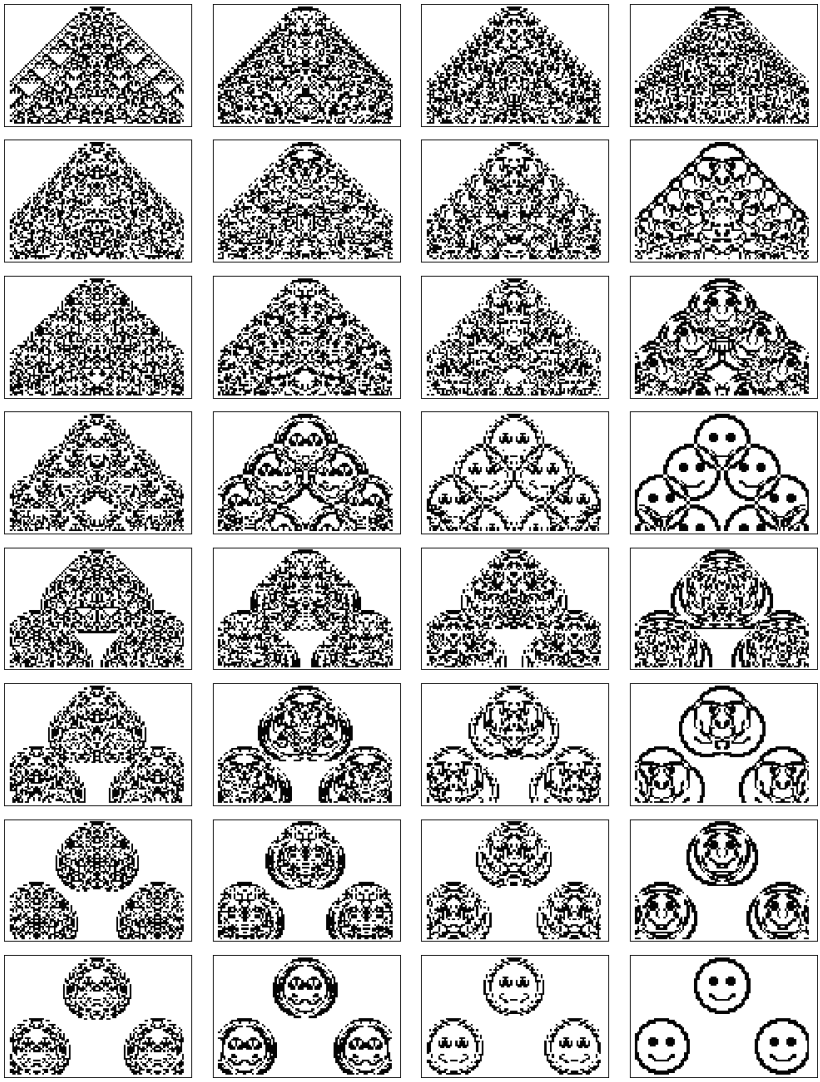


Figure 12. First 32 iterations (left to right, top to bottom) for the modified 1-CA field in Figure 11.

Figure 15 shows the first 16 iterations of the same starting configuration as in Figure 6 using Rule 105. Rule 105 in binary is {0,1,1,0,1,0,0,1}, which is the inverse of the binary representation of Rule 150—{1,0,0,1,0, 1,1,0}. Although the fractal replication behavior of Rule 105 is similar to Rule 150, it is not as perfect a replicator as Rule 150 because it introduces other artifacts as seen in the first 16 iterations. If the initial configuration had been made up of all black cells with just one white

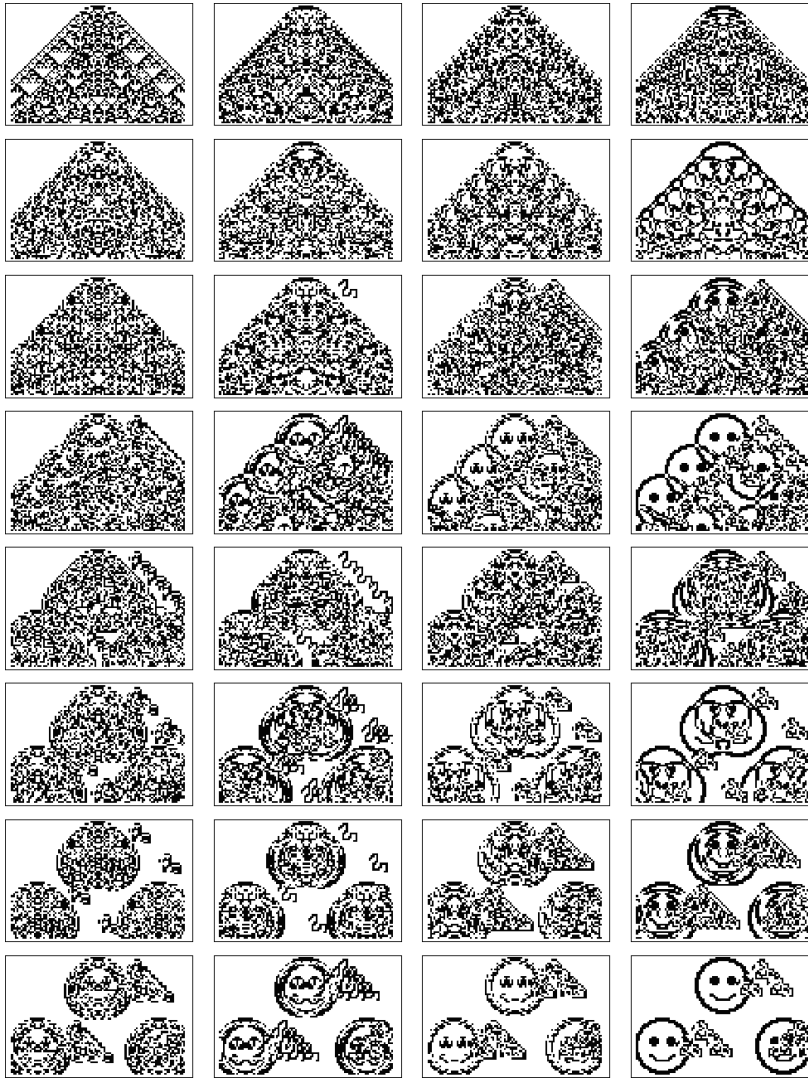
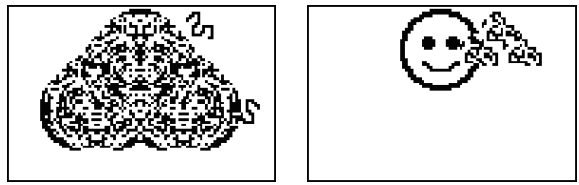


Figure 13. First 32 iterations of the modified 1-CA field in Figure 11, with a small squiggle introduced on the 10th iteration.

cell in the first row, then the result would have been the exact inverse of the modified 1-CA with Rule 150.

2.5 Summary

When we modify an elementary 1-CA to allow individual cells to look one step into the future and we allow the system to revisit the past over many iterations, the system shows fractal replication behavior.



(a) 42nd iteration (b) 64th iteration

Figure 14. 42nd and 64th iterations of modified 1-CA field in Figure 11.

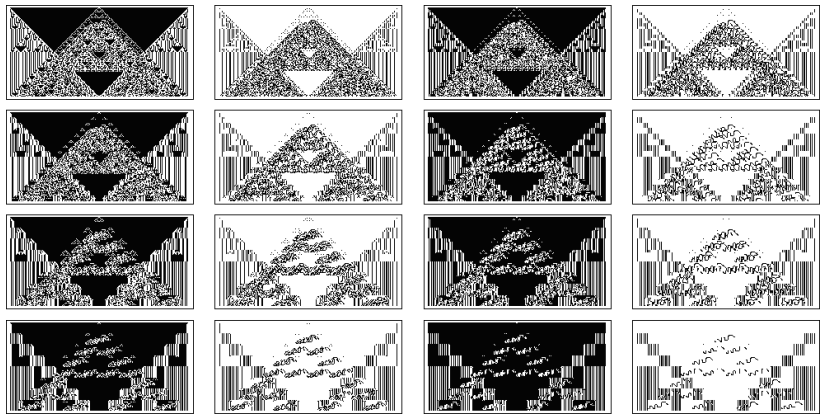


Figure 15. First 16 iterations of modified 1-CA system based on Rule 105.

Of the remaining 254 rules, Rules 60, 102, and 195 allow replication to occur, but not in a fractal layout.

Rule 150 is special in that it is the only rule that exhibits perfect fractal replication in our modified 1-CA systems. See [3], [6], and [7] for demonstrations of how self-similar nested structures (such as Sierpinski Triangles) can only occur when the rules are additive. The exclusivity of Rules 150 and 105 comes from the fact that they are the only rules which flip the result if any of the three inputs are flipped. This makes them the most specific additive rules, and the only ones capable of fractal replication behavior.

In section 3 we describe a more conventional 2-CA (without time manipulation) that also shows the fractal replication behavior seen in a modified Rule 150 1-CA.

3. Fractal replication in a two-dimensional cellular automaton

One way of bringing the nonstandard and seemingly arbitrary peeks into the future (and reiterations of the past) in line with conventional CA research is to study the time-modified 1-CA using a 2-CA.

The most well known 2-CA is the "Game of Life" devised by Conway and popularized in [10]. "Life," as it is popularly known, is a two state CA where the cells are arranged in a square grid, and each cell is updated based on a Moore neighborhood consisting of the cell itself and its eight immediate vertical, horizontal, or diagonal neighbors. The rule for updating the state of a cell on the next step is as follows: A cell in state 1 survives on the next time step if two or three of its neighbors are currently 1, and goes to state 0 otherwise; a cell in state 0 goes to state 1 if it is surrounded by exactly three neighbors in state 1.

There are many ways that we can set up the rules and neighborhood of a 2-CA in order to see fractal replication behavior. We describe one such system which is the exact analogue of the modified Rule 150 1-CA presented in section 2.

If we treat the vertical dimension of a 2-CA as the analogue of the time steps in our earlier 1-CA, then the rules and neighborhood of the 2-CA will be as follows.

- Number of states: $k = 2$.
- Neighborhood: 2—North-East (NE) and North-West (NW) neighbors.
- Update rule:
  $\text{Cell}_{t+1} = 1$ If $\text{Cell}_t = 1$ and Count $(\text{Neighborhood}_t) = 0$ or 2
  $\text{Cell}_{t+1} = 0$ If $\text{Cell}_t = 1$ and Count $(\text{Neighborhood}_t) = 1$
  $\text{Cell}_{t+1} = 1$ If $\text{Cell}_t = 0$ and Count $(\text{Neighborhood}_t) = 1$
  $\text{Cell}_{t+1} = 0$ If $\text{Cell}_t = 0$ and Count $(\text{Neighborhood}_t) \neq 1$.

In other words, a cell "survives" (remains in state 1) on the next step if it has 0 or 2 neighbors in state 1 on the current step, and an inactive cell (state 0) is born (becomes state 1) on the next step if it has exactly one neighbor in state 1 on this step. In the "Weighted Life" rule syntax developed in [8], this would be listed as "NW1, NN0, NE1, WW0, ME0, EE0, SW0, SS0, SE0, HI0, RS0, RS2, RB1." The first nine values in this list specify the weights of the neighbors being considered. "HI0" specifies that there are no history states or intermediate states between a cell's 1 and 0 states. "RS" and "RB" signify the rules for survival and birth respectively. Thus, in this rule, a cell in state 1 survives if surrounded by zero or two cells in state 1 while a cell in state 0 comes alive if surrounded by exactly one neighbor in state 1. The corresponding Wolfam number for this 2-CA is 10, with the NE and NW neighbors weighted as 1, and all other Moore neighbors weighted as 0.

If the initial configuration ($t = 0$) is a single 1 cell, then this 2-CA will evolve over the first four steps as shown in Figure 16.

Just as in the case of our modified 1-CA, any shape (a smiley face in Figure 17) "reproduces" in a self-similar way, laid out fractally as a Sierpinski Triangle.

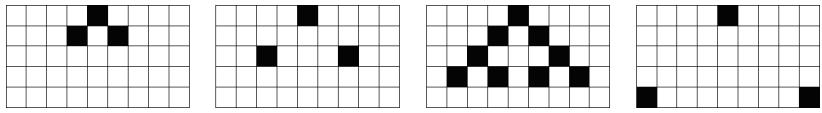


Figure 16. First four steps in a 2-CA analogue of a modified 1-CA.

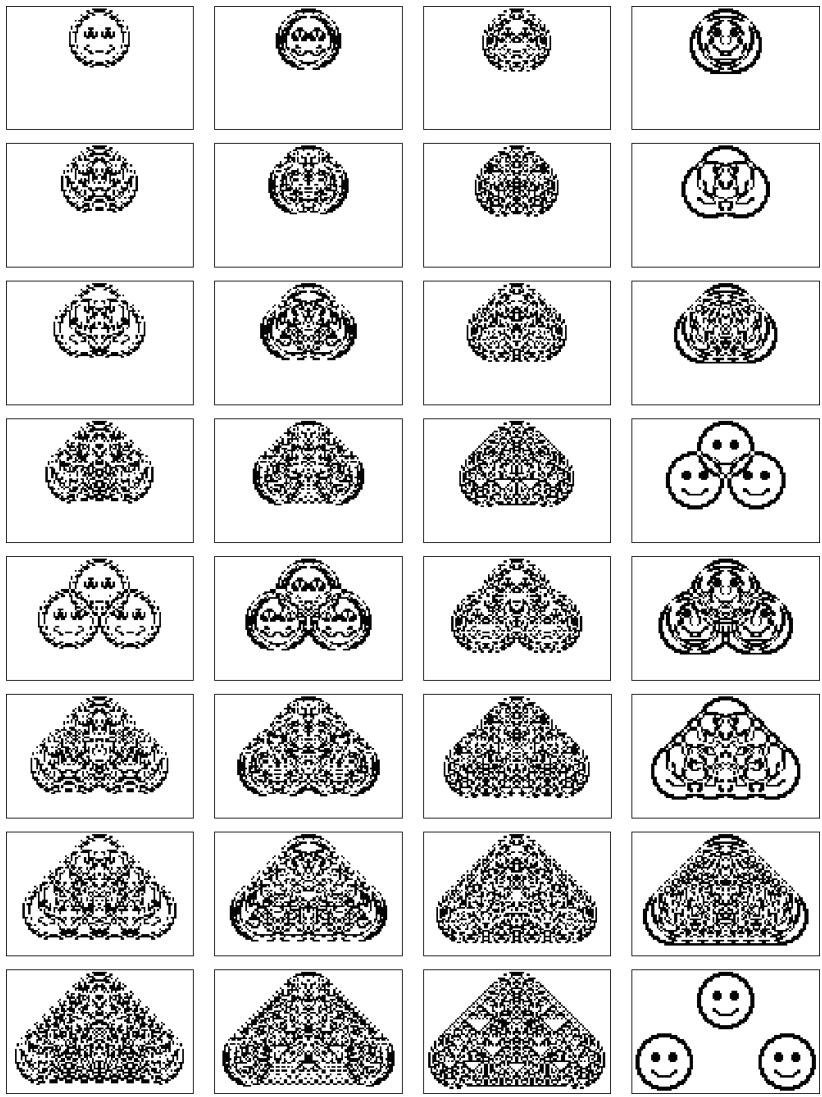


Figure 17. First 32 steps (left to right, top to bottom) of the 2-CA in Figure 15, initialized with a smiley face instead of a single active cell.

The key features of fractal replication in this 2-CA are the same as the features of fractal replication in the modified 1-CA.

4. Discussion

In both of these cellular automaton (CA) systems, perfect copies of the original configuration occur every 2^n steps (where n is dependent on the size of the configuration). The main difference between the two CAs is the way that the field changes from one iteration or step to the next. In the modified one-dimensional CA (1-CA), the iteration after the stage with the perfect replica is the most chaotic. In contrast, for the two-dimensional CA (2-CA), the step before the step with the perfect replica is the most chaotic looking.

Beginning with a starting configuration of a single active cell, both of these CA systems result in perfect Sierpinski Triangles. The modified 1-CA fills the field in the first iteration itself (the field cannot be infinite because we have to revisit the first row after an iteration is complete). On the other hand, the 2-CA described here grows infinitely.

Thus, the two systems are not identical. However, their fractal replication behavior is the same and is a direct consequence of the fact that the update rule is essentially the same and is additive in nature. In both cases, the states of three cells are considered before the update is done and if the number of active cells out of these three is odd, then the updated cell is active, else it is inactive.

In both systems the number of iterations (for the modified 1-CA) or time steps (for the 2-CA) is a step function dependent on the size of the cell configuration.

Rule 150 is the only rule for which our modified elementary 1-CA shows perfect fractal replication. However, there are several 2-CAs (with different neighborhoods and rules) that fractally reproduce any shape present at the start. For a list of 2-CA replicators (based on the "Weighted Life" model in [8]) with square, rectangular, and triangular layouts, see [11].

Our focus in this paper was on the effect of modifying the time domain of a 1-CA. The 2-CA is just a way to demonstrate that what we have done is not some artifact of the updating scheme. Although additive rules have been described before [3], in traditional 1-CAs they form nested self-similar structures only for very simple initial conditions. For more complex initial states, such systems do not exhibit fractally laid out copies. By allowing individual cells to look ahead in our 1-CA, and by allowing the modified system to revisit the past, we see qualitatively different behavior that is not immediately intuitive. Shapes can grow across each other without losing their identity, which is a well-known property of additive rules such as Rule 90. We also see dramatic phase transitions from sharp replicas to chaotic jumbles (and *vice versa*, in the

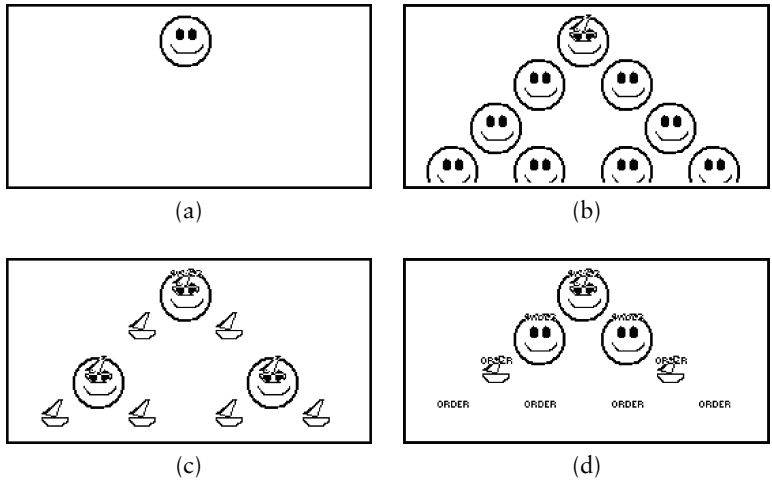


Figure 18. The Modified 1-CA is able to separate out images placed in the field at different points of time. (a) Starting with a smiley face. (b) After 32 iterations a small boat is placed at the top of the field, overlapping the first smiley face. (c) After 64 iterations a bitmap of the word "ORDER" is placed in the same location, but it is illegible because all black cells in the bitmap are flipped to white if it is overlapping a cell that is already black. (d) After 96 iterations the bitmap of "ORDER" is clearly visible, and the remaining shapes will also emerge on their own over the next 32 iterations.

case of the 2-CA described here). If shapes are of different sizes, or they are introduced on different iterations, then they sharpen at different times. Perfect and sharp replicas of all shapes are seen every 2^n steps (n depends on the size of the shape).

The additivity of the rule allows shapes to be introduced at different points in different parts of the field and on different iterations. Even though the shapes grow and overlap each other, they maintain their identity over many iterations. In Figure 18, new shapes are introduced on every 32nd iteration and placed in the same part of the field with the condition that a black cell in the new shape is flipped to white if the cell below is already black. After 96 iterations, we see how this modified 1-CA seems to be recovering the images placed in the field.

Computers typically store and retrieve images from memory using explicit allocation and retrieval instructions. Our modified 1-CA system is storing many copies of each image with no higher-order programming instructions. The only point at which an external entity writes data to the cells is on the top of the CA field. Computer memories, on the other hand, would have to seek out an empty block of memory cells, write the new data, remember where it wrote the data, and then recover it later as the result of an external command.

Organic neurons have little in common with digital cells in a CA, except for one condition. The state of a neuron in a biological neural network is dependent on the (past) states of other neurons connected to it. Such is the case in our modified 1-CA. It is tempting to speculate that the principles underlying memory storage and retrieval in neural networks may be similar to our modified 1-CA.

References

[1] E. F. Codd, *Cellular Automata* (Academic Press, New York, 1968).

[2] J. von Neumann, "Theory of Self-Reproducing Automata," in *Essays on Cellular Automata*, edited by A. W. Burks (University of Illinois Press, Urbana, 1970).

[3] S. Wolfram, *A New Kind of Science* (Wolfram Media Inc., Champaign, IL, 2002).

[4] P. Sarkar, "A Brief History of Cellular Automata," *ACM Computing Surveys*, **32**(1) (2000) 80–107.

[5] M. Delorme, "An Introduction to Cellular Automata," in *Cellular Automata: A Parallel Model*, edited by M. Delorme and J. Mazoyer (Kluwer, Dordrecht, 1999).

[6] E. Fredkin, "Digital Mechanics: An Informational Process Based on Reversible Universal CA," *Physica D*, **45** (1990) 254–270.

[7] S. J. Willson, "Cellular Automata Can Generate Fractals," *Discrete Applied Mathematics*, **8** (1984) 91–99.

[8] M. Wótowicz, "Cellular Automata Rules Lexicon," available at: www.mirwoj.opus.chelm.pl/ca/rullex_vote.html (2001, September 15).

[9] S. Wolfram, "Statistical Mechanics of Cellular Automata," *Reviews of Modern Physics*, **55** (1983) 601–644.

[10] M. Gardner, *Wheels, Life and other Mathematical Amusements* (W. H. Freeman, New York, 1983).

[11] S. Kumar, "2-CA Fractal Replicators," available at: ylog.org/complex/replicators.html (2005, March).