Multiway Sequential Cellular Automata

Margaux H. Wong
margaux@margauxhwong.com

Cellular automata (CAs) are used to model rule-based evolutionary systems with standard CAs applying unitary, fixed rules to an entire generation at a time. A sequential updating asynchronous cellular automaton (CA) with more than one rule for each input sequence is studied. These multiway sequential CAs (MSCAs) can model complex systems with multiple branching rule sets where changes propagate through the system. This paper examines the case of one-dimensional, two-cell, two-branch MSCAs in order to better understand their structure and the impact of parameters. The complete set of 1296 M-type rule sets possible for this type of multiway sequential CA (MSCA) is applied to a full set of 32 initial conditions, representing all possibilities of a six-cell initial condition, generating 41 472 state graphs. Machine learning is used to classify a subset of these state graphs into 10 classes. Analytical data enables characterization of these classes of graphs and investigation of the role of rule sets in these state graphs. Target distribution analysis of the M-type rule sets is performed within each class of graphs to tease out intrinsic characteristics of the classes.

Keywords: cellular automata; multiway sequential cellular automata; MSCA; multiway systems; asynchronous cellular automata; nonhomogeneous cellular automata

1. Introduction

Cellular automata (CAs) are discrete models of computation also referred to as tessellation automata, cellular structures and tessellation structures [1–16]. CAs of many geometries have been studied [17–19]. CAs have been applied to a variety of applications, including viral spread [20], drug therapy [21], urban development [22, 23], pattern recognition [24–28], VLSI [29–31], image processing [32, 33], cryptology [34–39], bioinformatics [40], solitons [41] and fractals [42–45], to name a few.

In the one-dimensional binary case, a cellular automaton (CA) consists of an array of binary cells defined by an initial state and a list of evolutionary rules that describe how previous cell values determine current cell values. Figure 1 shows a space-time visualization of the behavior of the rule 30 CA, where each

https://doi.org/10.25088/ComplexSystems.34.3.325

row in the two-dimensional grid represents one application of the rule set to the row of cells. In other words, the vertical axis moving downward is the time axis.

While typical CAs update the entire array synchronously, the sequential CA (SCA) uses asynchronous updating [2]. The main difference in an SCA is that cell updates are based on new values of cells rather than old, which results in significantly different behavior than a standard CA. Figure 2 shows a spacetime visualization of the behavior of the rule 30 SCA.

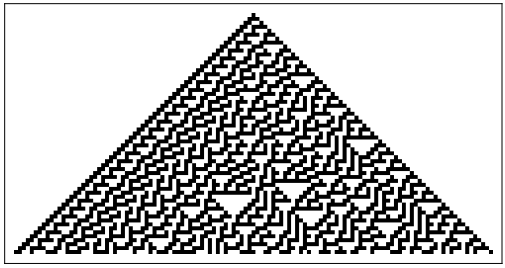


Figure 1. A spacetime visualization of the behavior of the rule 30 CA. The initial condition is an array of white cells with one black center cell. Each row represents one application of the rule set to the previous row of cells.

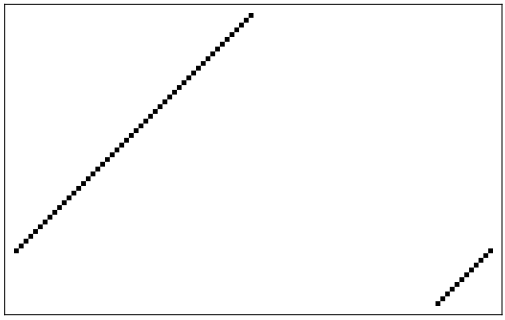


Figure 2. A spacetime visualization of the behavior of the rule 30 SCA. The initial condition is an array of white cells with one black center cell.

This paper presents a systematic analysis of multiway sequential CAs (MSCAs), which are an extension of traditional CAs and SCA models that allow for asynchronous updating of multibranching rule sets. This paper makes key contributions in graph representation, classification of graph structures and comprehensive rule set analysis of the MSCAs, potentially opening new directions for modeling systems where multiway propagation occurs, such as quantum field theory or biological distributed systems.

2. Sequential Cellular Automata and Multiway Sequential Cellular Automata

2.1 Sequential Cellular Automata

Sequential CAs (SCAs) can be defined to operate on r-dimensional arrays and have been described for a variety of geometries. One-dimensional arrays are the most common form; however, these evolutionary rule sets may operate on two-dimensional images, three-dimensional volumes or even higher-dimension arrays. In the SCA, one step is considered as the application of the rule set to a subset of cells, updating a portion of the r-dimensional array. One generation consists of the successive application of the rule set to the continually updated array, following a predetermined path covering the entire r-dimensional array once.

Examples of SCA rule sets are given in Figures 3–5. Figure 3 shows the well-known rule 30 rule set, in which three contiguous cells are used as the input to determine the new value of the center cell. Figure 4 shows an SCA that is discussed in more detail later, wherein two input cells yield two output cells. Both rule sets in Figures 3 and 4 are examples of $r = 1$ SCAs. Figure 5 shows an example of an $r = 2$ SCA where the rules are applied to a 2×2 portion of the array. Note that Figure 5 only displays a subset of the complete rule set for concision.

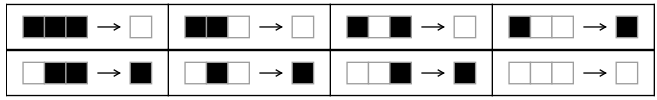


Figure 3. The SCA rule 30 rule set with three input cells and one output cell. The three input cells determine the new value of the center cell in this $r = 1$ SCA.

Figure 4. An SCA rule set with two input cells and two output cells. This is an $r = 1$ SCA.

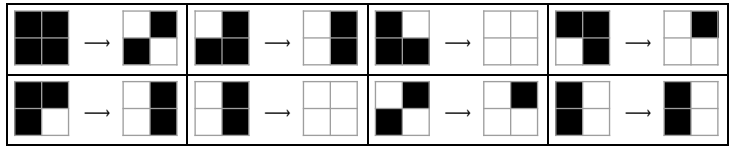


Figure 5. A subset of the rules comprising a two-dimensional SCA rule set, meaning $r = 2$. Each rule operates on a 2×2 portion of the two-dimensional array.

https://doi.org/10.25088/ComplexSystems.34.3.325

The SCAs considered in this paper operate on one-dimensional arrays, meaning $r = 1$, and have a fixed order of application; that is, the array is traversed in one direction. A step transforms a fixed number of cells, determined by the output size of the rule set. In the classic rule 30 example, this results in transforming a single cell of the array. A generation is the successive application of the rule set over the entire one-dimensional array, using updated values as the inputs after each step.

In order to define the evolutionary rule set for the SCA, let N be the length of the one-dimensional array upon which the SCA will operate. Let $g \in \mathbb{Z}$, $g \geq 0$, where g represents the current generation of the SCA. Let $s \in \mathbb{Z}$, $1 \leq s \leq N - 1$, where s denotes the current step of the SCA within generation g. Let $a_{s,g}$ be a finite array of length N that results after application of the rule set in step s of generation g of the SCA. Note that $a_{N-1,0}$ represents the initial condition, which is the original array input into the SCA. Let I be the input cell values, T be the target cell values, and D be the cardinality of I. Then the function f, which defines the rule set being applied by the SCA, is given by

$$f : \{I_k \to T_k \mid I_k \in I,\ T_k \in T,\ k \in \mathbb{Z},\ 1 \leq k \leq D\}. \tag{1}$$

Consider, for example, an SCA that operates on 2-cell inputs and yields 2-cell outputs (2:2), with binary values in the cells. The evolutionary update for the SCA for this 2:2 case is defined in equation (2). Note when $s = 1$, the updated values are based on the last step $s = N - 1$ of the previous generation, $g - 1$:

$$\begin{aligned}
(a_{s,g}[s],\ a_{s,g}[s+1]) &= f(a_{s-1,g}[s],\ a_{s-1,g}[s+1]) \\
&\qquad i = s,\ s+1,\ s > 1 \\
a_{s,g}[i] &= a_{s-1,g}[i] \\
&\qquad i \neq s,\ s+1,\ s > 1 \\
(a_{s,g}[s],\ a_{s,g}[s+1]) &= f(a_{N-1,g-1}[s],\ a_{N-1,g-1}[s+1]) \\
&\qquad i = s,\ s+1,\ s = 1 \\
a_{s,g}[i] &= a_{N-1,g-1}[i] \\
&\qquad i \neq s,\ s+1,\ s = 1.
\end{aligned} \tag{2}$$

The operation of this SCA starts from the leftmost cell of the array in a particular step s and generation g, $a_{s,g}[1]$. The SCA progresses one cell to the right at each successive step. When the rightmost cell in the array $a_{s,g}[N]$ has been updated, one generation is complete. As a result, the first and last cells of the array will be updated once, while all other cells will be updated twice, in two consecutive steps, in each generation. The rule set for an SCA will map each input cell pair to only one output cell pair.

The set of input cell values is the four ordered pairs $I = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. For clarity, these ordered pairs will be denoted $I = \{00, 01, 10, 11\}$. The set of target cell values is the same four binary options, $T = \{00, 01, 10, 11\}$. An example of the rule set is $f = \{00 \to 01, 01 \to 10, 10 \to 11, 11 \to 00\}$. Table 1 shows the first three steps of a 2:2 SCA on an array, a. The first row indicates the initial condition, with each column representing one element of the array. Each row represents a successive step in the SCA. The shaded cells indicate the cells that are transformed in each step, based on the values of those same two cells in the previous step. In Table 1, the letters α and β are appended to the index when there is a change in that value. A white cell indicates one whose value is copied from the cell above with no change.

Initial Condition	$a_{N-1,\,0}[1]$	$a_{N-1,\,0}[2]$	$a_{N-1,\,0}[3]$	$a_{N-1,\,0}[4]$	...
Step 1	$a_{1,\,1}[1\alpha]$	$a_{1,\,1}[2\alpha]$	$a_{1,\,1}[3]$	$a_{1,\,1}[4]$	...
Step 2	$a_{2,\,1}[1\alpha]$	$a_{2,\,1}[2\beta]$	$a_{2,\,1}[3\alpha]$	$a_{2,\,1}[4]$	...
Step 3	$a_{3,\,1}[1\alpha]$	$a_{3,\,1}[2\beta]$	$a_{3,\,1}[3\beta]$	$a_{3,\,1}[4\alpha]$	...

Table 1. Initial steps of a 2:2 SCA. Light blue regions show cells that are updated during that step. The letter α or β is appended to the index to indicate a change in that value.

As shown in Table 1, the first step applies the rule set to $a_{N-1,\,0}[1]$ and $a_{N-1,\,0}[2]$ to generate new values for those two cells, $a_{1,\,1}[1\alpha]$ and $a_{1,\,1}[2\alpha]$. Then, the second step applies the rule set to $a_{1,\,1}[2\alpha]$ and $a_{1,\,1}[3]$ and updates those two values, generating $a_{2,\,1}[2\beta]$ and $a_{2,\,1}[3\alpha]$. Note that the second element of the array has been updated twice, influenced by the values of $a_{N-1,\,0}[1]$, $a_{N-1,\,0}[2]$ and $a_{N-1,\,0}[3]$, generating $a_{2,\,1}[2\beta]$, the final value of the second element of the array in this generation. The influence of both neighbors on a cell's value mimics the three-cell rule sets of the most common regular CA but in a sequential manner where the rule set is applied twice and the intermediate value is operated on by the SCA. The differences become more obvious when the evolution of the SCA is examined further. The next step will apply the rule set to $a_{2,1}[3\alpha]$ and $a_{2,1}[4]$, updating those values, generating $a_{3,\,1}[3\beta]$ and $a_{3,\,1}[4\alpha]$. Note that the final value of the third cell is $a_{3,\,1}[3\beta]$, and it is also influenced by values of its neighboring cells. However, the difference between the SCA and CA is that in the traditional CA, the original values of $a_{N-1,\,0}[2]$, $a_{N-1,\,0}[3]$ and $a_{N-1,\,0}[4]$ impact the new

value of $a_{3,1}[3\beta]$, whereas in the SCA $a_{1,1}[2\alpha]$, $a_{2,1}[3\alpha]$ and $a_{2,1}[4]$, which has carried down the value of $a_{N-1,0}[4]$, govern the new value of $a_{3,1}[3\beta]$ instead. The incorporation of $a_{1,1}[2\alpha]$ is the key difference and highlights a major attribute of the SCA: propagation. As can be seen in Figure 2, the value of $a_{N-1,0}[1]$ impacts $a_{1,1}[2\alpha]$, the value of $a_{1,1}[2\alpha]$ impacts $a_{2,1}[3\alpha]$, and the value of $a_{2,1}[3\alpha]$ impacts $a_{3,1}[4\alpha]$. Therefore, the effects of $a_{N-1,0}[1]$ continue to propagate down the array within one generation. Similarly, the impact of each cell propagates to the rest of the cells in the array as the SCA evolves through a generation. The SCA is a variant of a CA as opposed to a Post tag system, as the evolution is still based on the previous values of the array and cell values are neighbor dependent, simply in a more complex and interwoven manner.

2.2 Multiway Sequential Cellular Automata

This paper introduces multiway SCAs (MSCAs) where additional complexity is added by using nonhomogeneous rule sets, or, in other words, an MSCA maps one input to several targets. Each state of an MSCA has several evolutionary possibilities, with each state propagating information from previous states due to its sequential nature. MSCAs can be defined to operate on r-dimensional arrays like SCAs. MSCAs operating on r-dimensional arrays have a fixed geometric order of application of the b-maximal branch rule sets. In other words, the r-dimensional array has a preset order in which the cells are traversed and updated. At each update, there are up to b different rules for updating the current cell, yielding up to b different outputs. For a particular rule set, each input will have a predetermined set of outputs. However, instead of a 1:1 mapping, one input might have three outputs, while another input might have one output. Note that this means in MSCAs, different inputs can yield different numbers of outputs as b is the maximum, not fixed, number of outputs.

Examples of MSCA rule sets or subsets of these rules are shown in Figures 6–10. Figure 6 shows an MSCA that, like the SCA in Figure 3, maps three input cells to a single output cell. However, in Figure 6, there are two outputs for each input. This is an example of a 3:1 $r = 1$, $b = 2$ MSCA. Figure 7 shows an example of an MSCA rule set that has two input cells, two output cells and two output possibilities for each input cell. This is a 2:2 $r = 1$, $b = 2$ MSCA. Figure 8 shows a subset of rules of what will be referred to as the quad flex MSCA. In this case,

there are again two input cells and two output cells. However, there are between one and four outputs for each rule. This is thus a 2:2 $r = 1$, $b = 4$ MSCA. Figure 9 shows a subset of rules of a 5:3 $r = 1$, $b = 8$ MSCA. Finally, Figure 10 shows a two-dimensional case where rules are applied to a 2×2 subset of the array. The subset of rules shown is from an $r = 2$, $b = 2$ MSCA.

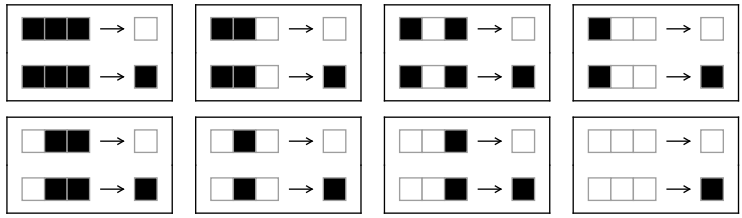


Figure 6. An $r = 1$ MSCA rule set where each rule has three input cells and one output cell. For this MSCA, $b = 2$, meaning there are up to two outputs for each input. This case has exactly two outputs for each input.

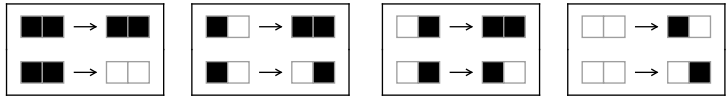


Figure 7. An $r = 1$ MSCA rule set, where each rule has two input cells and two output cells. For this MSCA, $b = 2$, meaning there are up to two outputs for each input. This case has exactly two outputs for each input.

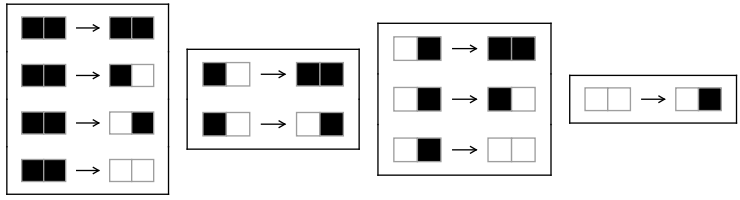


Figure 8. A subset of the rules of an $r = 1$ MSCA rule set where each rule has two input cells and two output cells. For this MSCA, $b = 4$, meaning there are between one and four outputs for each input. This case will be referred to as a quad flex MSCA.

https://doi.org/10.25088/ComplexSystems.34.3.325

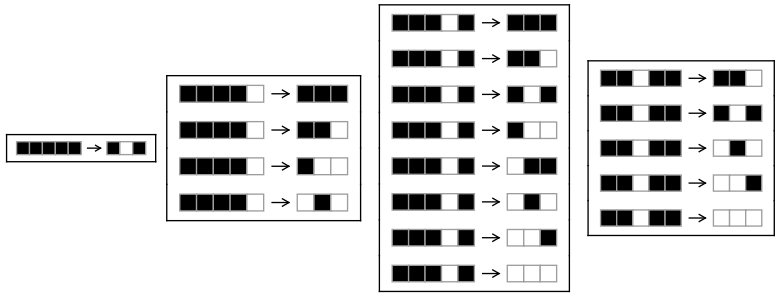


Figure 9. A selection of the rules comprising an $r = 1$ MSCA rule set where each rule has five input cells and three output cells. For this MSCA, $b = 8$, meaning there are between one and eight outputs for each input.

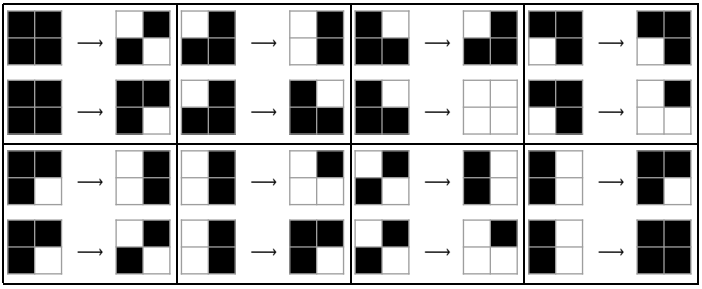


Figure 10. A selection of the rules comprising an $r = 2$ MSCA rule set. Each rule has a 2×2 array of input and output cells. For this MSCA, $b = 2$, meaning there are up to two outputs for each input. This case has exactly two outputs for each input.

Due to the b-maximal branch character of MSCAs, more complex definitions of step and generation are required. In MSCAs, when the rule set is applied, if there are two possible outputs, then there are now two versions of the updated r-dimensional array. Each time the rules are applied, up to b versions of r-dimensional arrays are created. Therefore, instead of a simple series of updated arrays, there is now a tree-like structure of updated arrays. However, further complicating this is that some of the updated arrays can be identical to arrays created elsewhere on the tree, yielding a graph structure where the nodes are versions of the r-dimensional arrays. Within this MSCA progression, a step is defined as the application of the rule set to the identical cells in every current array. The current arrays are those created in the previous step. For example, in a 2:2 MSCA rule set operating on a one-dimensional array, where the rule set

is being applied to the third and fourth cells of the array, a step would consist of applying the MSCA rule set to the third and fourth cells of every current one-dimensional array. A generation then is complete when the rule set has been applied along the prescribed path covering the entire r-dimensional array once. In the same 2:2 one-dimensional array example, a generation is complete when the application of the rule set to the last two cells of the array is completed.

The mapping h representing the rule set of a unique MSCA is shown in equation (3), where b is the maximum number of branches allowed at each step, k indicates the current element of the set of inputs, I_k is the k^{th} element of I (i.e., one of the inputs), J_k indicates the number of branches or mappings that occur for element I_k, and D is the cardinality of I. Note that the function h is defined as the complete rule set, including the varying number of branches for each input:

$$\begin{aligned} h : \{ I_k \rightarrow {T_k}^{J_k} \mid\ & I_k \in I,\ {T_k}^{J_k} \in T^{J_k}, \\ & J_k \in \{1,\ \ldots,\ b\},\ k \in \mathbb{Z},\ 1 \le k \le D, \}. \end{aligned} \tag{3}$$

This paper will examine MSCAs that operate on one-dimensional arrays with 2-cell inputs that yield 2-cell outputs (2:2), with binary values in the cells, similar to the SCA discussed earlier. Thus the updating scheme for the MSCA is shown in equation (4), where a new subscript has been added to the array element indicating the branch number. Again, j indicates the current branch being constructed, that is, $1 \le j \le J_k$. Let q be the branch of the tree from the previous generation, whose descendants are being constructed. Let k be determined by the value of input cells. Note when $s = 1$, the updated values are based on the last step $s = N - 1$ of the previous generation, $g - 1$:

$$\begin{aligned} \left(a_{j,s,g}[s],\ a_{j,s,g}[s+1]\right) &= h\left(a_{q,s-1,g}[s],\ a_{q,s-1,g}[s+1]\right) \\ & \qquad i = s,\ s+1,\ s > 1 \\ a_{j,s,g}[i] &= a_{q,s-1,g}[i] \\ & \qquad i \ne s,\ s+1,\ s > 1 \\ \left(a_{j,s,g}[s],\ a_{j,s,g}[s+1]\right) &= h\left(a_{q,N-1,g-1}[s],\ a_{q,N-1,g-1}[s+1]\right) \\ & \qquad i = s,\ s+1,\ s = 1 \\ a_{j,s,g}[i] &= a_{q,N-1,g-1}[i] \\ & \qquad i \ne s,\ s+1,\ s = 1. \end{aligned} \tag{4}$$

Due to the added b-maximal branch complexity, MSCA results cannot be displayed as a series of updated arrays forming an image. Thus, we introduce a graph structure in the next section for the display and analysis of MSCA results.

https://doi.org/10.25088/ComplexSystems.34.3.325

3. Multiway Sequential Cellular Automata Graph Structures

3.1 Evolutionary Graphs of Sequential Cellular Automata in the Context of Multiway Sequential Cellular Automata

In order to narrowly focus on studying the impact of rule sets on the final graph structure of the MSCA, the underlying SCA of the particular MSCA studied in this paper will remain a 2-cell input and output (2:2) structure; however, for every 2-cell input there will be multiple rules enabling several possible outputs. There is no longer a single output array for each generation, as found in the regular SCA, and as such, the standard two-dimensional grid representing several generations of output can no longer be used. The evolution of the MSCA can be visualized using a series of graphs that indicate branching due to the multiway rule sets. While a tree-like structure can be used to show the time component of evolution, this representation is not ideal for the MSCA, as it fails to show repetitive nodes. Therefore, a multiway state graph will be used, where each node represents a unique array and edges show the evolution pattern.

An MSCA has b-maximal branch rule sets, meaning at each application of rules, an input pattern will map to at most b possible output patterns. To distinguish them from regular CA rule sets, these MSCA rule sets will be called M-type rule sets. Since the M-type rule sets studied in this paper use a 2:2 input output structure, $I = \{00, 01, 10, 11\}$. Since each input has four possible outputs as shown in Table 2, there are 16 possible cell-level rules. Each of these rules has been color-coded as shown in Table 2 to allow the impact of these individual rules to be seen in the state graphs. An M-type rule set is a subset of these 16 rules. These 16 cell-level rules will be referred to in the paper with the color and mapping. For example, rule 1 will be represented in the form rule 1 (■: 00 → 00).

A table of state graphs will show one complete generation of the SCA or MSCA in a column, with each successive entry in the

Rule #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Color	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Input	00	00	00	00	01	01	01	01	10	10	10	10	11	11	11	11
Output	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11

Table 2. Cell-level rules that are used to form two-cell MSCA M-type rule sets. Colors are assigned to each rule to enable identification of rules in the state graphs. This table is for a one-dimensional MSCA, that is, operating on a single row of cells, with rules applied to blocks of two cells at a time, using a two-branch MSCA. This color legend is used for all of the subsequent figures in this paper.

column displaying the results of one step of the SCA or MSCA. For each step, a state graph is shown, with every node representing a particular state of the one-dimensional array, with white cells indicating a 0 value and black cells indicating a 1 value. These nodes are connected by colored edges to show the evolution of the MSCA. Step i of a generation transforms cells i and $i+1$. The colored edges are as defined in Table 2 to indicate which of the cell-level rules were applied to progress from one state to the next. Duplicate edges of the same color are not shown in these state graphs in order to maximize clarity, due to the high complexity of the multiway graphs shown in later sections.

The $b = 1$ case of the MSCA is simply the straightforward, deterministic SCA. This can be displayed as a multiway state graph where no branching occurs. Table 3 shows the step-by-step evolution of the state graph of an example of the most basic SCA, where only one rule changes cell values. The M-type rule set in this example includes rules 2 (■: 00 → 01), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11). Each step refers to the application of the rule set to two cells. In step 1, the first two cells of the initial condition (00110) are transformed from 00 to 01, resulting in a node of 01110. A pink arrow can be seen because this is the result of applying rule 2 (■: 00 → 01). The next step operates on the second and third cells of the current state (01110), transforming them using rule 16 (■: 11 → 11). Thus, the state maps back to itself, generating the orange loop pointing back to the same state. In step 3, rule 16 (■: 11 → 11) is applied. The third step in Table 3 shows no change because the third and fourth cells of the current state (01110) are 11 and the self-returning orange edge is already shown. Step 4 displays the addition of a bright green loop as the fourth and fifth cells of the current state (01110) are transformed using rule 11 (■: 10 → 10). As the end of the array has been reached, the first generation is complete. The second generation begins with step 1 operating on the first and second cells of the current state (01110). Applying rule 6 (■: 01 → 01) creates the blue loop and results in no change in the current state (01110). However, the convergence criterion is met, terminating the process. Convergence is achieved if all children states have been previously reached in the same step of a previous generation. It must be in the same step because the step number governs which cells will be processed next. If the state is reached in the same step of a previous generation, then the same rules will be applied in the same order on the same cells in future steps, yielding no new unique edges or states in the states graph. At this point, the algorithm concludes.

https://doi.org/10.25088/ComplexSystems.34.3.325

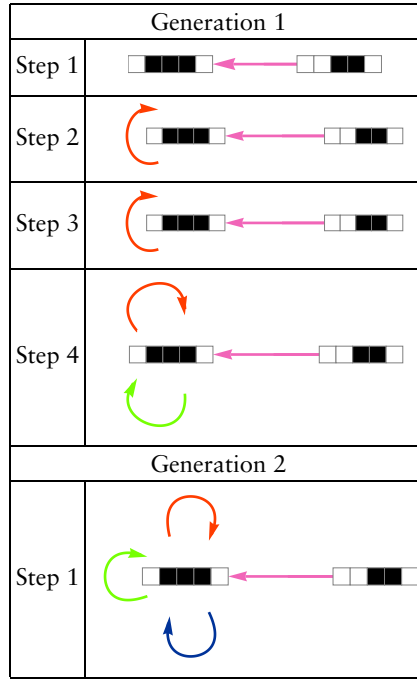
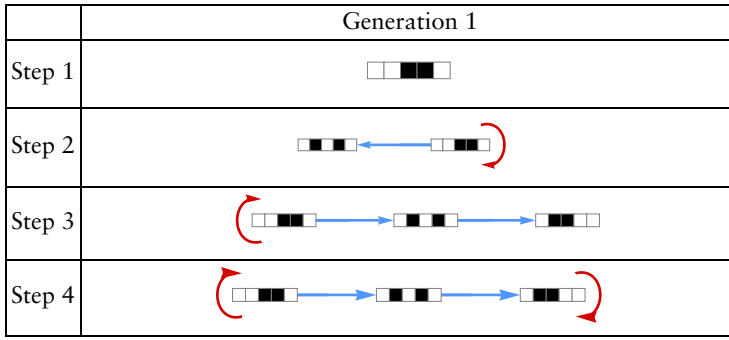
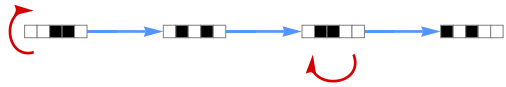
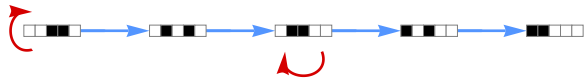
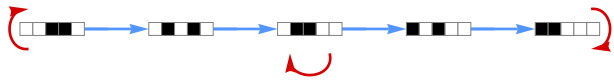
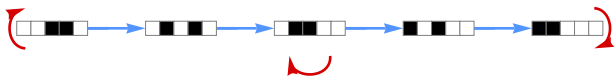


Table 3. Step-by-step state graphs for an SCA with two input, two output structure, using rules 2 (■: 00 → 01), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11) and starting with the initial condition 00110.

As seen in Table 3, SCAs are often linear or at least have a linear component. Table 4 shows the mapping where only the 01 pair changes and the M-type rule set includes rules 1 (■: 00 → 00), 7 (■: 01 → 10), 11 (■: 10 → 10) and 16 (■: 11 → 11). Although it is more complex than the previous case and does not converge for three generations, it is still linear, augmented by loops that indicate a mapping back to the same state.



	Generation 2
Step 1	
Step 2	
Step 3	
Step 4	

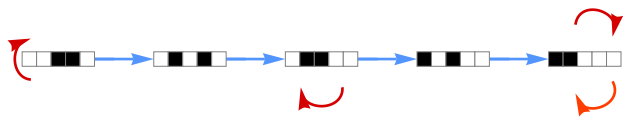
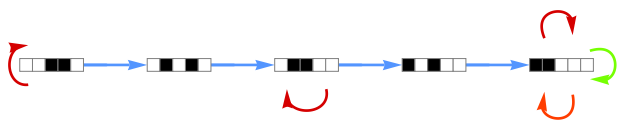
	Generation 3
Step 1	
Step 2	
Step 3	
Step 4	

Table 4. State graphs for the SCA with rules 1 (■: 00 → 00), 7 (■: 01 → 10), 11 (■: 10 → 10) and 16 (■: 11 → 11). The initial condition is 00110.

Not all SCA graphs are linear in appearance as shown in Table 5. However, the branching in SCA graphs is not due to multiway rule sets; rather, it is due to states that reoccur but at different steps of their respective generations. The step indicates which two cells are to be processed next, regardless of the generation. Thus, even if the same state is reached, if different cell pairs within the state are processed next, it can lead to different subsequent states. In Table 5, the evolution of an SCA with a rule set consisting of rules 2 (■: 00 → 01), 7 (■: 01 → 10), 12 (■: 10 → 11) and 13 (■: 11 → 00) displays a distinctly nonlinear pattern after initially appearing linear. Note that the rules are "circular," meaning each pair maps to a different pair, and no two pairs map to the same pair. Selected generations of the SCA evolution are shown, with this SCA converging in the ninth generation.

https://doi.org/10.25088/ComplexSystems.34.3.325

Generation 1	
Step 1	
Step 2	
Step 3	
Step 4	

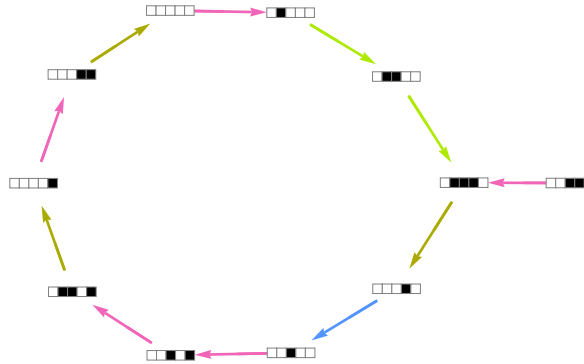
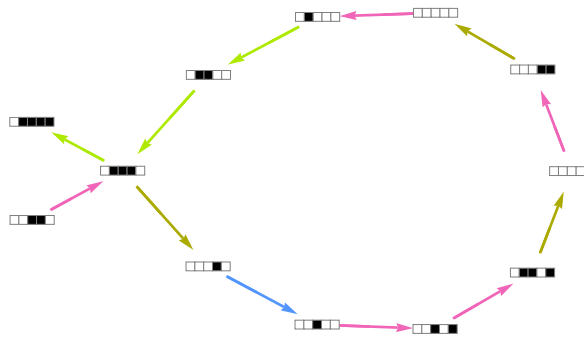
Generation 3	
Step 1	
Step 2	
Step 3	
Step 4	

Table 5. Generations 1 and 3.

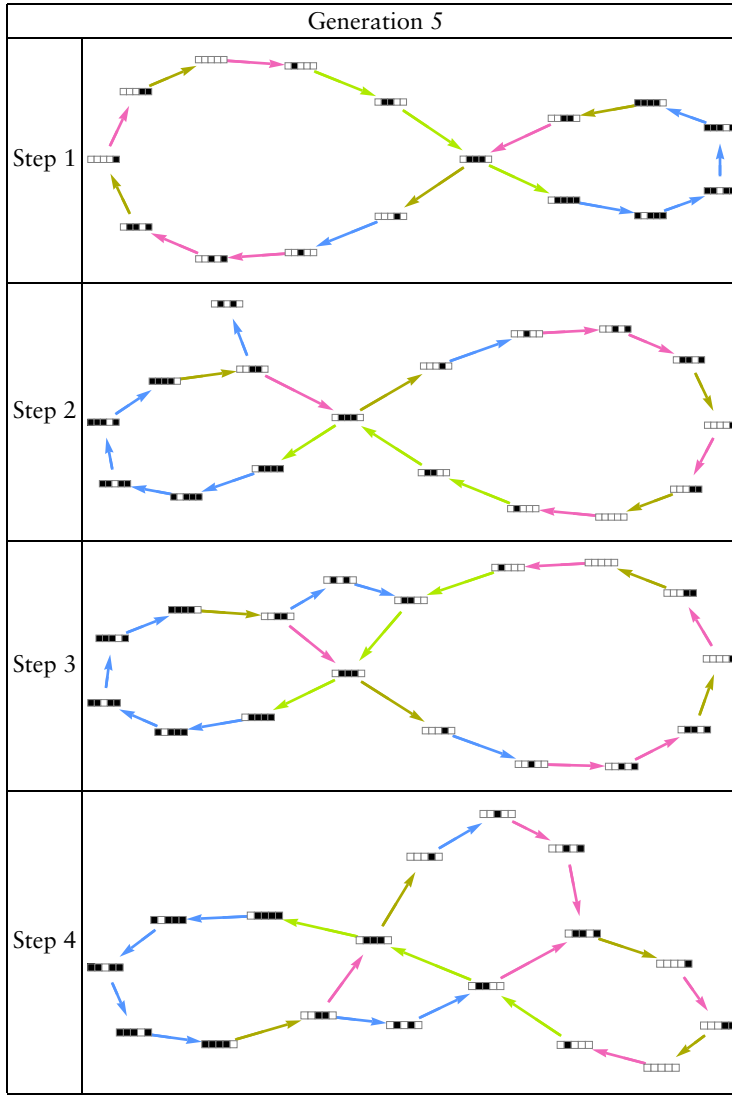


Table 5. Generation 5.

https://doi.org/10.25088/ComplexSystems.34.3.325

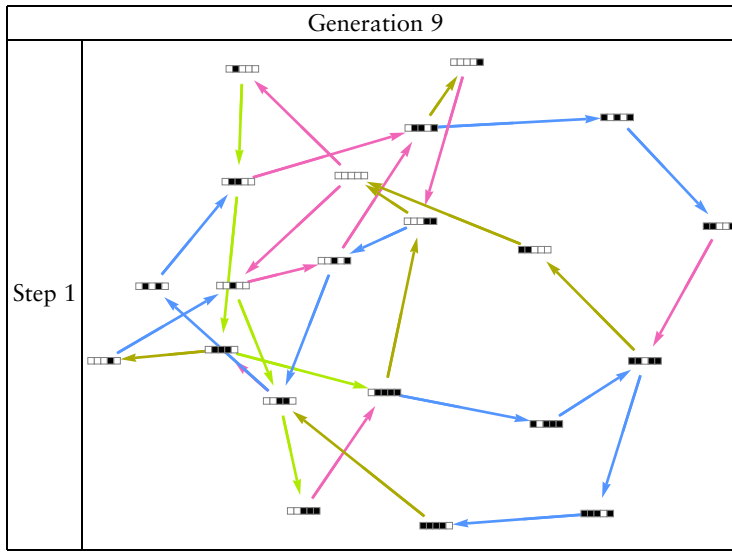
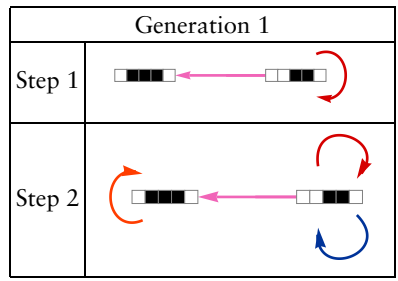


Table 5. Generations 1, 3, 5 and 9 of the evolutionary state graphs for the SCA with rules 2 (■: 00 → 01), 7 (■: 01 → 10), 12 (■: 10 → 11) and 13 (■: 11 → 00). The initial condition is 00110.

3.2 Evolutionary Graphs of Simple Multiway Sequential Cellular Automata

In this section, the most elementary MSCAs will be examined, where only one input does not map to itself. Consider first the most basic, one-dimensional, $b = 2$ MSCA, where only one rule has two possible outputs and all other inputs map back to themselves. Table 6 shows each step of the MSCA for a rule set with rules 1 (■: 00 → 00), 2 (■: 00 → 01), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11). The difference between this MSCA and the SCA examined in Section 2 is that 00 maps to two possible outputs rather than one. As only one of the options of the multiway does not map back to itself, the graph still appears linear, with loops returning to states.



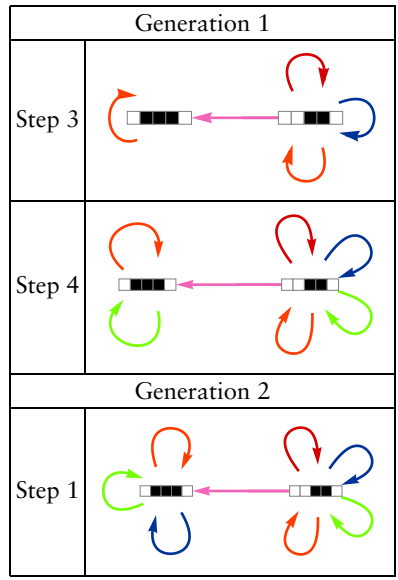


Table 6. State graphs for the MSCA with the rule set containing rules 1 (■: 00 → 00), 2 (■: 00 → 01), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11). The initial condition is 00110.

A rule set where the multiway options for one input pair do not include the identity mapping back to itself displays typical MSCA branching in its state graph. Table 7 shows the state graphs for the MSCA with a rule set containing rules 2 (■: 00 → 01), 3 (■: 00 → 10), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11). Immediately, the first step shows the initial condition branching into two new states. As 00 no longer appears in the states, the remaining steps merely add loops as rules 6, 11 and 16 are applied, as determined by the cell values as the MSCA traverses each child state graph.

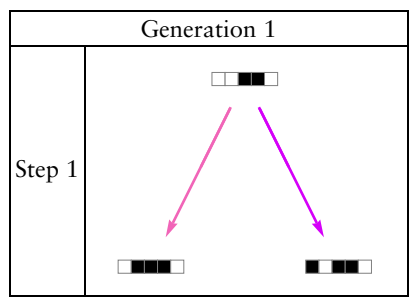


Table 7. (*continues*).

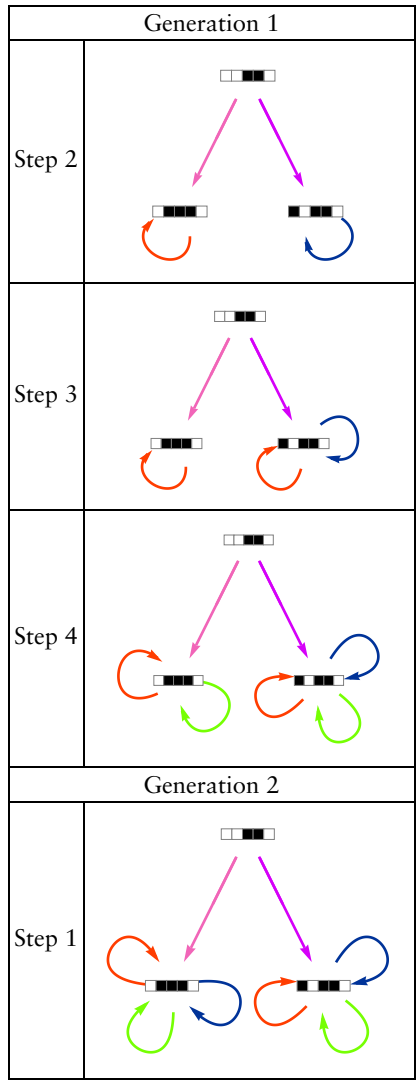


Table 7. State graphs for the MSCA with a rule set containing rules 2 (■: 00 → 01), 3 (■: 00 → 10), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11). The initial condition is 00110.

Table 8 shows a $b = 3$ MSCA where 00 maps to the three other cell pairs and not back to itself. The other cell pairs map back to themselves. The rule set therefore contains rules 2 (■: 00 → 01), 3 (■: 00 → 10), 4 (■: 00 → 11), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11). The expected three-way branch is seen immediately in step 1. Again, as that is the only occurrence of the cell pair 00, no further branching is seen, only the addition of identity loops.

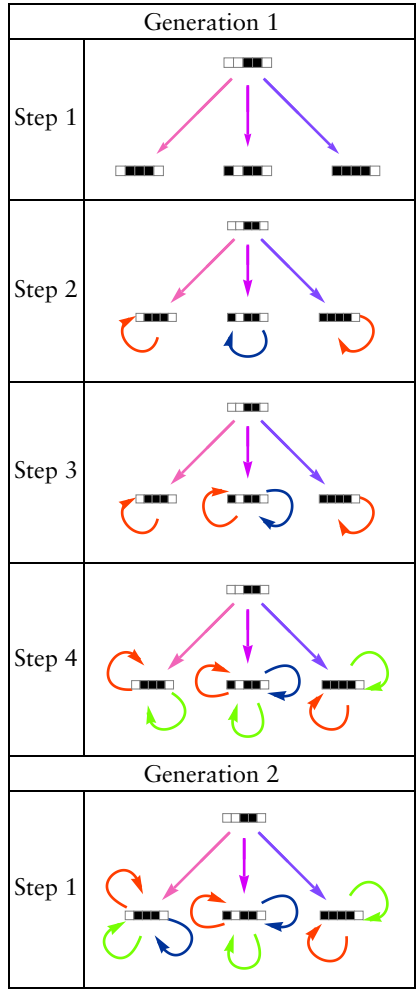


Table 8. State graphs for the MSCA with the rule set containing rules 2 (■: 00 → 01), 3 (■: 00 → 10), 4 (■: 00 → 11), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11). The initial condition is 00110.

Table 9 shows a $b = 4$ MSCA where 00 maps to all four cell pairs and the other cell pairs map back to themselves. The rule set therefore contains rules 1 (■: 00 → 00), 2 (■: 00 → 01), 3 (■: 00 → 10), 4 (■: 00 → 11), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11). The expected three-way branch is seen immediately in step 1, along with an identity loop. Again, as that is the only occurrence of the cell pair 00, no further branching is seen, only the addition of identity loops.

https://doi.org/10.25088/ComplexSystems.34.3.325

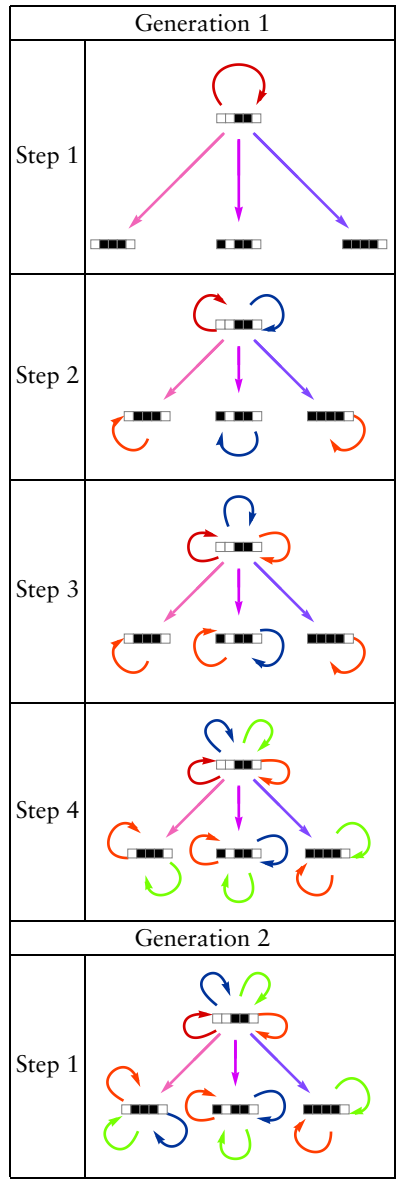


Table 9. State graphs for the MSCA with the rule set containing rules 1 (■: 00 → 00), 2 (■: 00 → 01), 3 (■: 00 → 10), 4 (■: 00 → 11), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11). The initial condition is 00110.

3.3 Two-Branch Multiway Sequential Cellular Automata

It is clear that when more than one input has multiple outputs, the graph's complexity increases significantly. To better understand the structure of these graphs and the relationship between

rule sets and graph structure, a more detailed analysis of the one-dimensional, 2:2 input output, $b = 2$, no repeated MSCA is discussed in this section. The reason for these choices is that $b = 2$ is the first-level multiway case. This allows the multiway effects to be seen most clearly in graphs, as the number of nodes and edges is smaller than in systems with higher b values. By enforcing two separate rules at each rule application, the full effect of the branching can be seen.

In Table 2, there are four possible outputs for each input. For a two-branch MSCA, each rule set contains two of these four possible outputs for each input. Since the two outputs are required to be distinct, there are two branches at each node. Thus, for the first output, there are four choices, and for the second output, there are three choices, resulting in 12 possible pairs of distinct outputs in the rule set. However, within these 12 possible pairs, for every pair of rule x and rule y, there is also a pair of rule y and rule x. These duplicates are eliminated, leaving six possible output cell pairs for each input cell pair. Since there are four input cell pairs, this results in 6^4 or 1296 possible rule sets. These 1296 rule combinations will be referred to as M-type rule sets, with a number specifying their identity to distinguish them from both the 16 possible cell-level rules shown in Table 2 and the standard CA rules [2]. More generally, for a one-dimensional MSCA with a b-branch multiway system, where z-size cell blocks of the current state are evaluated at a time, the number of possible M-type rule sets is given by

$$\text{number of rule sets} = \left(\begin{matrix} 2^z \\ b \end{matrix} \right)^{2^z}. \tag{5}$$

The MSCA diagrammed in Table 10 is a two-branch MSCA with a shorter initial condition 011 but with a more complex M-type rule set M1044. M1044 contains the rules 2 (■: 00 → 01), 4 (■: 00 → 11), 6 (■: 01 → 01), 8 (■: 01 → 11), 11 (■: 10 → 10), 12 (■: 10 → 11), 15 (■: 11 → 10) and 16 (■: 11 → 11). The initial condition of 011 is represented by the rightmost node in the first panel. After the first step of evolution, the multiway evolution of the state 011 yields the states 011 and 111. The blue arrow from 011 back to itself shows that rule 6 (■: 01 → 01) was applied to the initial condition. The cyan arrow from 011 to 111 shows that rule 8 (■: 01 → 11) was applied to the initial condition. Thus, the children states from this step of evolution are 011 and 111. In the second panel (Generation 1, Step 2), each of the children states from panel 1 has undergone multiway

https://doi.org/10.25088/ComplexSystems.34.3.325

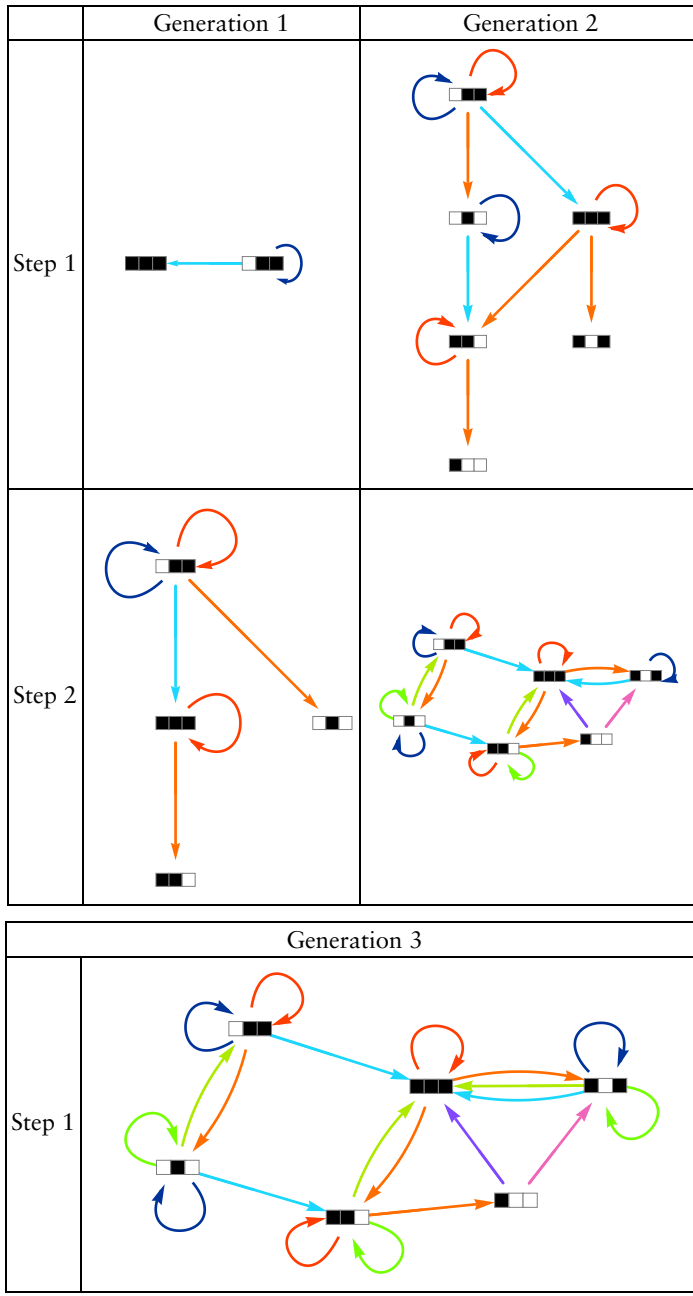


Table 10. Example of the evolution of a two-dimensional MSCA. The initial condition of this graph was 011 and the rule set was M1044, which contains rules 2 (■: 00 → 01), 4 (■: 00 → 11), 6 (■: 01 → 01), 8 (■: 01 → 11), 11 (■: 10 → 10), 12 (■: 10 → 11), 15 (■: 11 → 10) and 16 (■: 11 → 11).

evolution, so there are two new arrows emanating from both states 011 and 111. These are the dark orange (■) and light orange (■) edges. Note that there are two new states: 010 and 110. The evolution continues in generation 2, shown in the second column. The MSCA stops when convergence is reached.

4. Classification of the Multiway Sequential Cellular Automata Graphs

Considerable work has been done studying classification of regular CAs [46–58]. The nature of the MSCA is significantly different from the traditional CA. As shown in Section 3, the evolution of an MSCA rule set cannot be shown in a single spacetime visualization commonly used to examine and classify traditional CA characteristics; rather, the branching is displayed in a directed graph, or digraph, with nodes representing evolved states of the cells. The standard spacetime visualization is equivalent to a single directed walk on the graph that makes a choice at each node visited on which subsequent direction to take. There is no counterpart in traditional CA rules that could generate the resulting visualization unless for every input the same output is chosen, which is a single special case in the entire set of possible walks. Furthermore, each rule set in the MSCA yields a multitude of different visualizations corresponding to each directed walk. Especially for more complex MSCAs, for the same rule set, there can be a set of nodes traversed that yields a CA sensitive to initial conditions, orderly, and with high compression, while the same rule set with a different traversal of nodes may yield a CA with low sensitivity to initial conditions, chaotic behavior and low compression. Therefore, the classification that is examined in this paper is a classification of the multiway graph structure, not a classification of the typical cellular automata visualizations. It should also be noted that although each node represents a specific state of the array, the MSCA classification does not consider these states or cell values.

In order to characterize the MSCA and to study the impact of various parameters on the evolution of MSCA graphs, six different initial conditions with six cells each were tested, namely 000100, 001110, 010110, 101011, 110011 and 111001. These initial conditions represented a good variety of configurations, and were long enough to show the evolutionary impact of sequential evaluation of the rules yet short enough for most cases to converge reasonably quickly in the two-branch MSCA case. As will be seen in Section 5, the rule set is the dominating

factor as opposed to initial condition, thus six diverse initial conditions were sufficient for classification. The 1296 possible M-type rule sets were individually applied to each of the six initial conditions, yielding a robust set of 7776 cases.

Graphs from two initial conditions, a total of 2592 cases, were examined to determine unique graph classes. Ten different classes were manually identified based on features and structure such as node structure, node clustering and edge density: Class 1 Radials, Class 2 Central Clusters, Class 3 Skewed Clusters, Class 4 Arcs, Class 5 Bimodals, Class 6 Periodic Clusters, Class 7 Grids, Class 8 Even Grids, Class 9 Axials and Class 10 Lattices. The classes were distinct enough that manual training data was consistent and reproducible. Then, 962 of the 2592 manually classified graphs, or 12% of the total 7776 graphs, were used to form a training dataset. A machine learning classifier was created using the Wolfram Classify function. This function determines model specifics such as a method and hyperparameters through an automated procedure of experiments on training data subsets. For example, four method types were attempted: logistic regression, decision tree, nearest neighbors and random forest. The logistic regression learning curve with a limited memory Broyden–Fletcher–Goldfarb–Shanno optimization algorithm was best, thus was used for the final classifier. The logistic regression method uses linear combinations of numerical features to model the log probabilities of each class. This classifier was used to classify the remaining 5184 graphs into the 10 predetermined classes. 335 of the graphs that had been hand classified that were not used for the training data or parameter tuning were used to check the level of agreement between hand classification and the trained classifier function. Of these 335 graphs, 93.7% were classified the same compared to the model's predicted accuracy of $(94.4 \pm 2.2)\%$.

The 10 classes of graphs are presented in this section. As these graphs are more complex than the ones studied in the previous section, they will be simplified by showing the nodes as black circles instead of displaying the entire array at each node. As an example, the step-by-step evolution of an exemplar of the first class will be examined. Class 1 graphs are referred to as Radials and have two characteristic nodes in the center and a radial structure. Rings of evenly spaced nodes emanate outward from the center. Figure 11 is an exemplar of this class.

Table 11 shows the step-by-step evolution of the exemplar graph for class 1. The graph in generation 2, step 2 is noteworthy for its bimodal nature. The graph converges by the fifth step of the third generation.

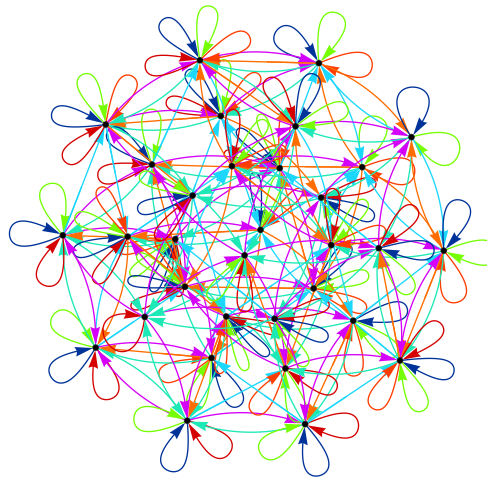
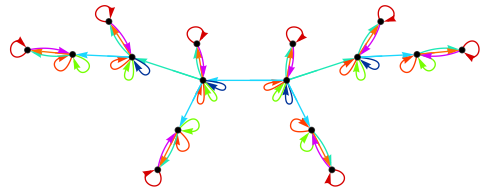


Figure 11. Example of an MSCA class 1 Radial graph generated from an initial condition of 010110 and the rule set M372 containing rules 1 (■: 00 → 00), 3 (■: 00 → 10), 6 (■: 01 → 01), 8 (■: 01 → 11), 9 (■: 10 → 00), 11 (■: 10 → 10), 15 (■: 11 → 10) and 16 (■: 11 → 11). Class 1 Radial graphs show a circular, radial structure.

Generation 1	
Step 1	
Step 2	
Step 3	
Step 4	

Table 11. Steps 1–4, Generation 1.

https://doi.org/10.25088/ComplexSystems.34.3.325

Generation 1	
Step 5	

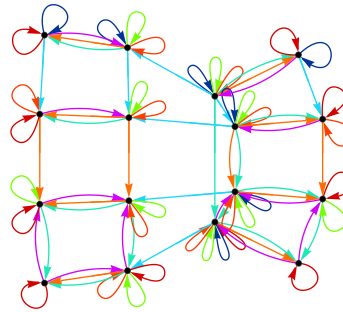
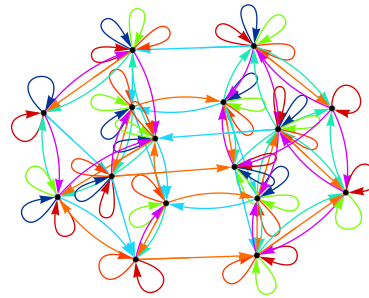
Generation 2	
Step 1	
Step 2	

Table 11. Step 5, Generation 1. Steps 1 and 2, Generation 2.

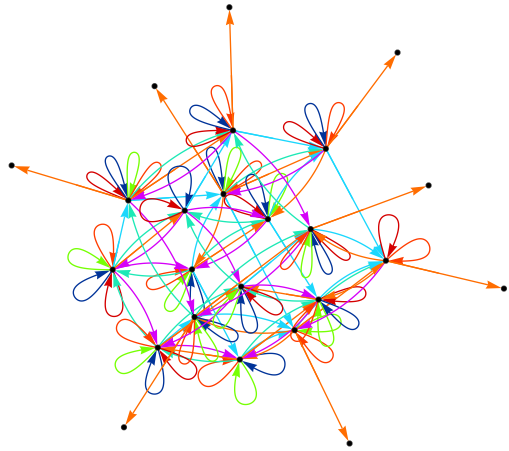
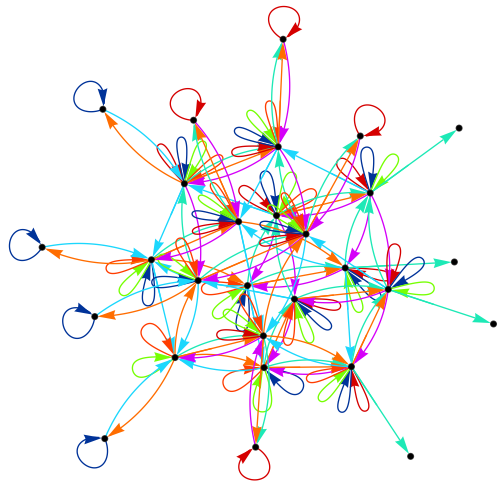
Generation 2	
Step 3	
Step 4	

Table 11. Steps 3 and 4, Generation 2.

https://doi.org/10.25088/ComplexSystems.34.3.325

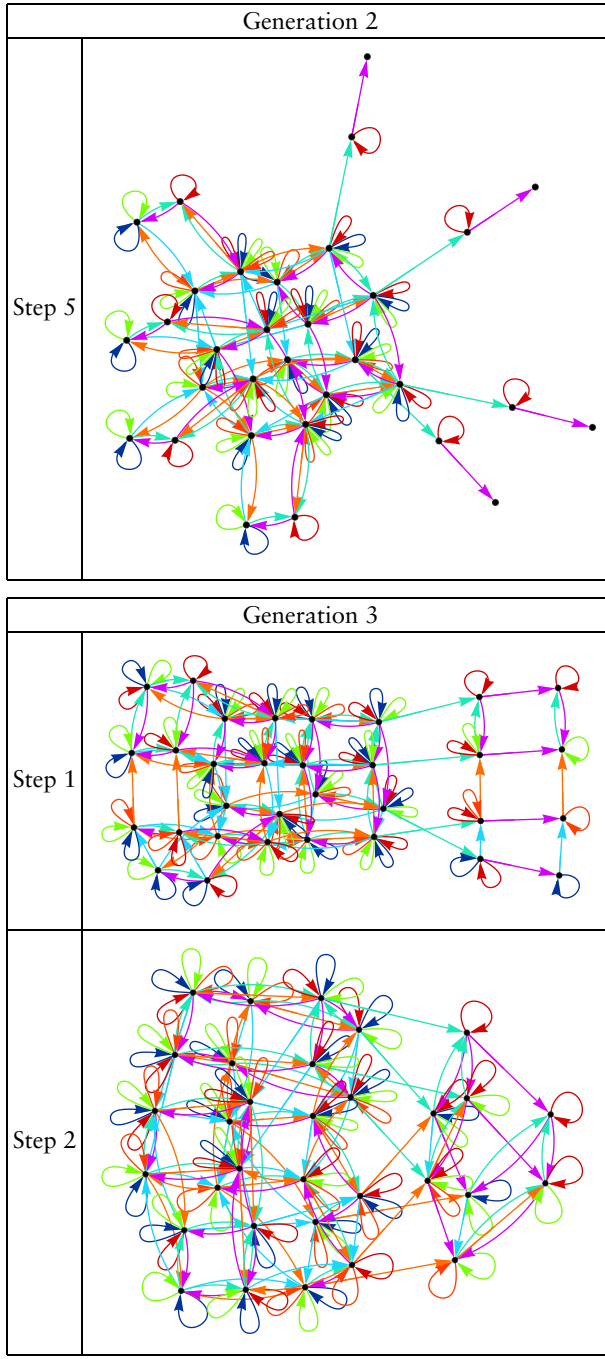


Table 11. Step 5, Generation 2. Steps 1 and 2, Generation 3.

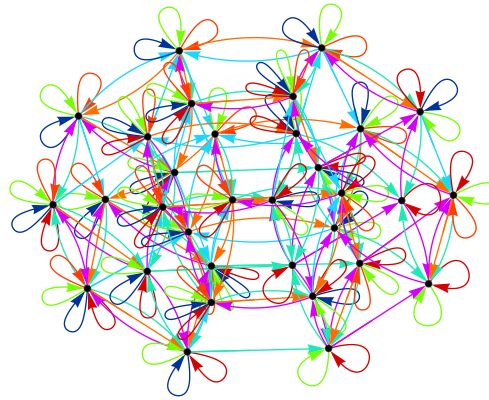
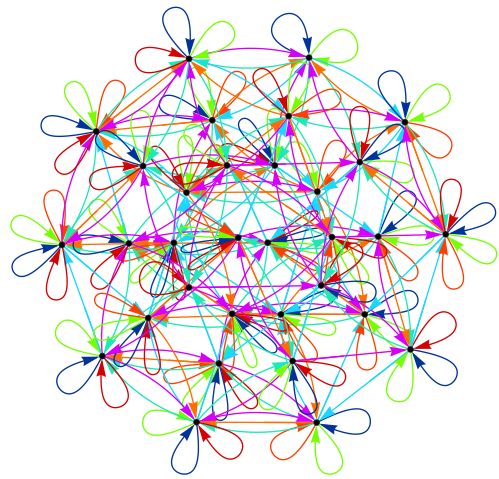
Generation 3	
Step 3	
Step 4	

Table 11. Steps 3 and 4, Generation 3.

https://doi.org/10.25088/ComplexSystems.34.3.325

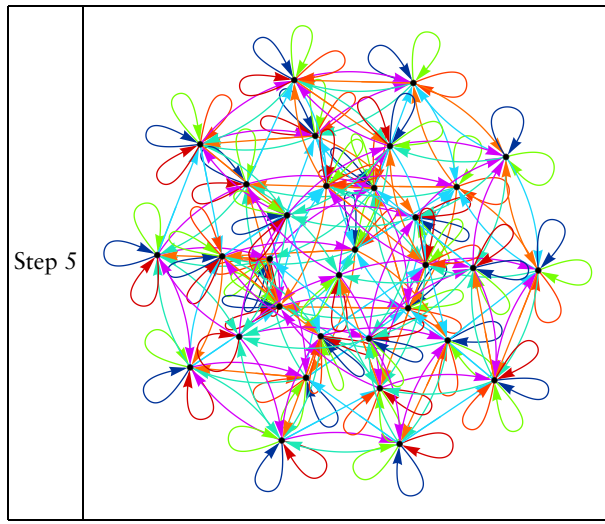


Table 11. Step-by-step evolution of the MSCA class 1 Radial exemplar with initial condition 010110 and rule set M372, whose constituent cell-level rules can be found in Table 13.

Table 12 shows exemplars of the 10 graph classes. Significant differences can be seen in the distribution of nodes and edges as well as overall structure. Table 13 provides the cell-level rules that form each of the M-type rule sets used to generate each of the exemplars in Table 12. As stated before, class 1 graphs, labeled Radials, have a radial structure and rings of evenly spaced nodes emanating outward from the center. Radials have fewer nodes, are symmetric, and the overall density of edges is lower. Class 2 Central graphs have a cluster of nodes in the center, evenly spaced nodes around the outside of the graph, and are symmetric in nature. The distribution of nodes in the exemplar is classic for this class of graphs. Class 3 Skewed graphs are asymmetric and have a larger concentration of edges and nodes on one side of the graph. Notable features include strong curvature on one side, resulting from a series of unidirectional edges connecting a series of nodes, as well as denser but smaller groups of edges on the more sparse side of the graph. Class 4 graphs, Arcs, have an arc-like shape on one side and a cluster of edges on the other that emanates from the center. Class 5 graphs, Bimodals, have two clusters of nodes and edges connected by a relatively sparse region of nodes and edges. Typically, the color-coded edges show two distinct groups of rules generating the edges that connect the two regions. Class 6 Periodic graphs are sparse and have an elongated structure with

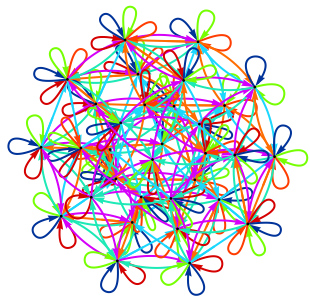
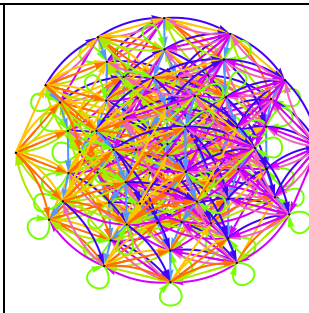
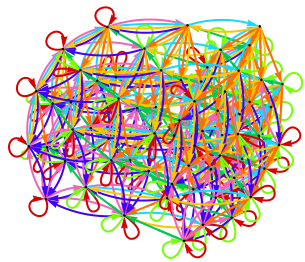
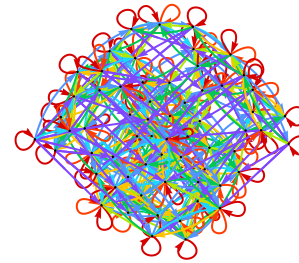
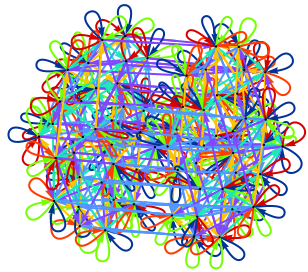
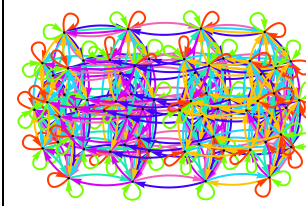
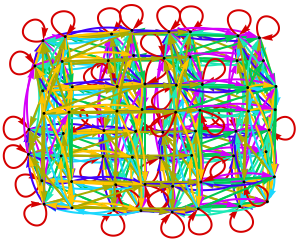
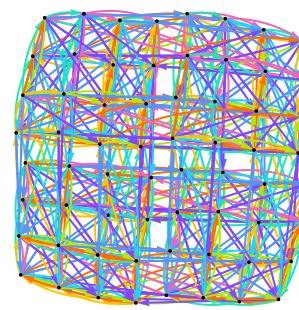
	
Class 1: Radial	Class 2: Central
	
Class 3: Skewed	Class 4: Arc
	
Class 5: Bimodal	Class 6: Periodic
	
Class 7: Grid	Class 8: Even

Table 12. (*continues*).

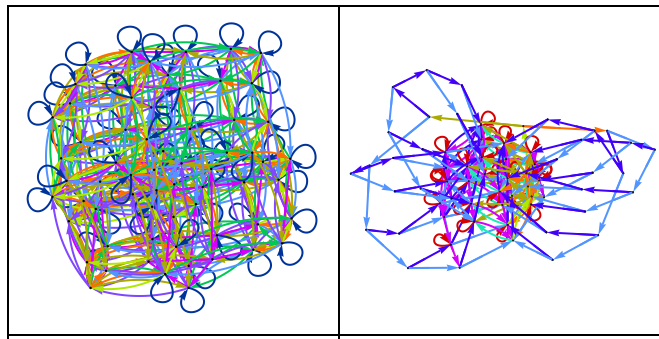


Table 12. Exemplars for all 10 classes of MSCA generated from an initial condition of 010110 with the rule sets indicated in Table 13.

three or four repeating dense regions that contain all of the nodes. Class 7 graphs, Grids, have a symmetric and orderly grid of 64 nodes. Each node belongs to two groups of nodes and edges, which are seen as columns and rows that are each two nodes wide. Class 8 graphs, Evens, appear as distorted grids of 64 nodes. The edges are more uniformly spread throughout the grid than in class 7 graphs, and the color is very uniform, with all colors in all areas of the graph. Class 9 Axial graphs are less uniform and orderly than class 6, 7 or 8 graphs. Axials feature two groups of denser edges that manifest as vertical and horizontal axes. The remaining nodes are sparsely scattered with fewer edges between them. Class 10 graphs, Lattice graphs, are more sparse and have a lattice-like structure surrounding a central cluster of nodes.

Cell-Level Rule	Radial	Central	Skewed	Arc	Bimodal	Periodic	Grid	Even	Axial	Lattice
	M372	M718	M94	M641	M551	M731	M289	M1060	M1214	M266
1 (■: 00 → 00)	x		x	x	x		x			x
2 (■: 00 → 01)		x	x			x		x		
3 (■: 00 → 10)	x	x				x	x		x	x
4 (■: 00 → 11)				x	x			x	x	
5 (■: 01 → 00)		x	x			x	x			x
6 (■: 01 → 01)	x				x				x	
7 (■: 01 → 10)		x		x	x			x	x	x
8 (■: 01 → 11)	x		x	x		x	x	x		
9 (■: 10 → 00)	x				x	x	x	x		x
10 (■: 10 → 01)			x	x			x		x	
11 (■: 10 → 10)	x	x	x		x	x				
12 (■: 10 → 11)		x		x				x	x	x
13 (■: 11 → 00)							x		x	x
14 (■: 11 → 01)		x	x	x	x	x	x	x		
15 (■: 11 → 10)	x	x	x					x	x	x
16 (■: 11 → 11)	x			x	x	x				

Table 13. Cell-level rules comprising the M-type rule sets used to generate each MSCA class exemplar in Table 12.

Recall that there are eight cell-level rules contained in each M-type rule set for the 2:2 two-branch, no repetition case of MSCAs being studied. Table 13 shows that there is a good distribution of cell-level rules among the M-type rule sets used to generate the MSCA exemplars. Each of the 16 cell-level rules is found in at least three exemplars, with a maximum of seven occurrences. Similarly, the rule sets show considerable variation in their underlying cell-level rules. There is a maximum correspondence of five cell-level rules between any two of the 10 rule sets in Table 13. The most common overlap is three cell-level rules. Thus, the distinct character of the exemplars is reflected in the rules that comprise the M-type rule sets from which the exemplars are generated.

5. Graph Analysis

5.1 Analysis of Initial Conditions

In order to study the impact of initial conditions on the graph structure, a total of 41 472 state graphs were generated by applying the 1296 M-type rule sets to 32 initial conditions. This represents all possible length-six binary initial arrays, because each of the 64 possibilities has a duplicate by zero/one symmetry, for example, 000 000 and 111 111. For each case, the MSCA was run until convergence. The 41 472 MSCA graphs were examined to determine if the initial condition impacted the final graph classification. The percentage of each class that originated from each initial condition was determined and shown in the bar charts in Figure 12. The distribution is fairly uniform for each class, indicating that initial condition is not a good indicator of final graph structure. The only outlier is the relatively fewer instances of the initial condition 000 000 in the Arc class, which is likely due to classification errors.

Another study was performed examining the resulting behaviors of each of the 1296 rules. Within the 41 472 graphs, there were 32 different initial conditions for each M-type rule set. The resulting graph structures were classified, then the number of initial conditions that yielded the same graph class was counted for each M-type rule. Of the 1296 rules, over 85% of the rule sets yielded at least 30 of 32 graphs in the same class. Over 93% yielded 26 or more graphs in the same class. Combined with Figure 12, this clearly indicates that the rule set is the dominant factor in final graph structure.

https://doi.org/10.25088/ComplexSystems.34.3.325

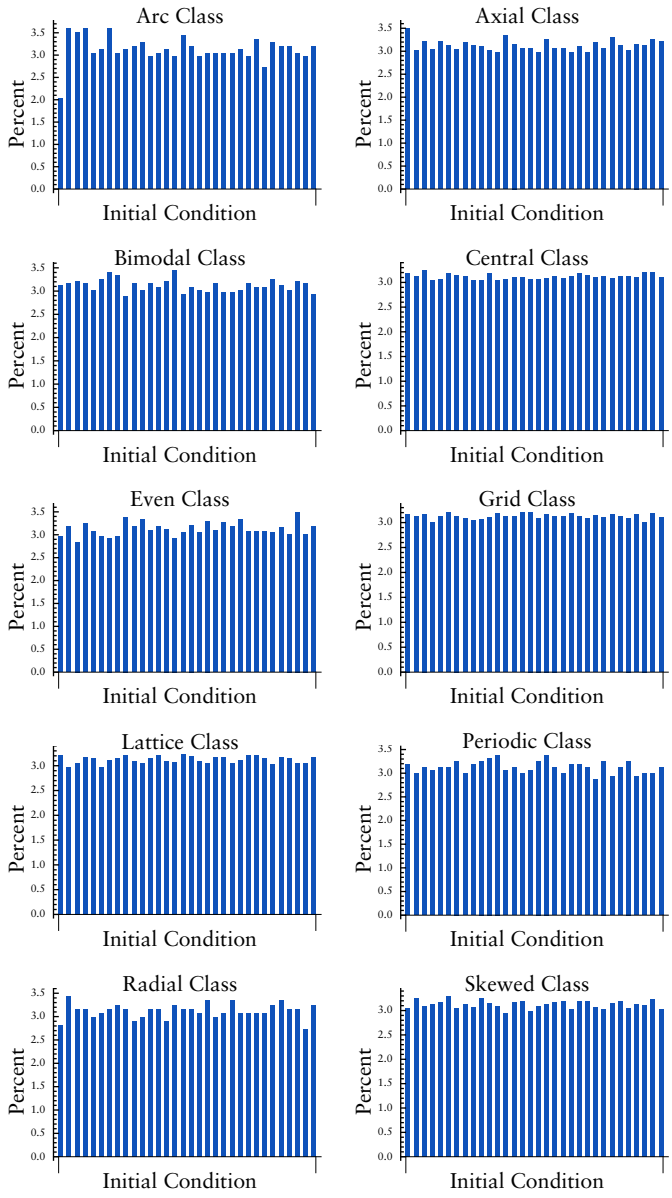


Figure 12. Bar charts for the 10 MSCA classes showing the percentage of graphs in that class that originates from each initial condition. A relatively even distribution is seen for every class.

5.2 Graph Analytics

As initial conditions have been shown to be less impactful than rule sets in determining the behavior of MSCAs, the more focused subset of 7776 graphs generated using the initial conditions 000100, 001110, 010110, 101011, 110011 and 111001 introduced in Section 4 was used for a deeper dive into the characteristics of the MSCA classes. Analytical data was calculated for each of the 7776 graphs, including vertex count, edge count, mean vertex degree and mean betweenness centrality as measures of graph structure. Vertex degree, also called valence, is the number of edges incident to the vertex. Betweenness centrality measures a node's centrality based on the number of shortest paths between other vertex pairs that include the node. The mean and standard deviation were calculated for each class for each graphical measure, as shown in Tables 14 and 15. Class 1 Radial graphs were a distinct outlier in all four measures. Radial graphs have roughly half the number of vertices as other graphs from other classes and fewer than half the edge count, which points to a far sparser graph. Class 8 Evens had the highest number of vertices and edges as well as the highest average mean vertex degree. This is apparent in the visual appearance of class 8 Even graphs, as there are no regions of sparse edges. There is remarkable similarity in vertex count between all the classes of graphs except the class 1 Radials. The higher standard deviation in all measures for the class 10 Lattices points to the far greater variation in Lattice graphs, suggesting possible subclasses within the Lattice class.

Class #	Class	Vertex Count μ	Vertex Count σ	Edge Count μ	Edge Count σ
1	Radial	32.0	0.152	260.0	19.1
2	Central	63.9	2.160	589.0	39.0
3	Skewed	62.6	4.910	549.0	64.1
4	Arc	63.0	1.430	549.0	28.6
5	Bimodal	64.0	0.541	578.0	30.0
6	Periodic	60.9	7.510	520.0	74.5
7	Grid	63.3	4.470	587.0	54.3
8	Even	64.0	0.000	613.0	21.3
9	Axial	63.7	1.590	560.0	40.2
10	Lattice	52.4	10.60	315.0	87.5

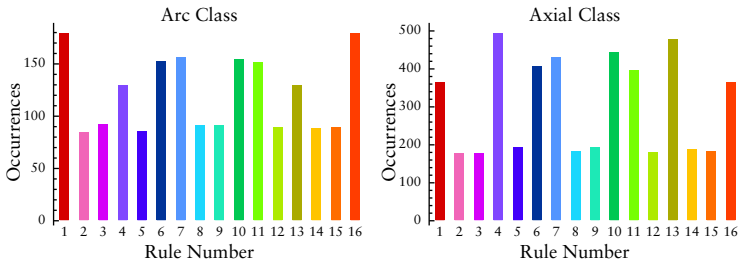
Table 14. Vertex and edge counts for each MSCA class.

Class #	Class	Mean Vertex Degree μ	Mean Vertex Degree σ	Mean Betweenness Centrality μ	Mean Betweenness Centrality σ
1	Radial	16.2	1.19	49.9	7.64
2	Central	18.4	0.988	129.0	17.3
3	Skewed	17.5	1.21	129.0	22.7
4	Arc	17.4	0.769	126.0	14.2
5	Bimodal	18.1	0.901	106.0	14.8
6	Periodic	17.0	0.753	122.0	31.9
7	Grid	18.5	0.916	104.0	13.5
8	Even	19.2	0.664	104.0	8.76
9	Axial	17.6	1.04	112.0	14.9
10	Lattice	11.9	1.85	109.0	35.4

Table 15. Broader data analytics for each MSCA class.

5.3 Rule Analysis

The 7776 state graphs were used to examine the impact of specific cell-level rules within M-type rule sets, as described in Table 2, on resulting graph behaviors. The goal is to determine if particular cell-level rules or sets of cell-level rules lead to specific classes of graphs. Furthermore, similarities between classes can be illuminated, as well as classes that are outliers in behavior. For each class, the various M-type rules responsible for generating the graphs were inventoried. The 16 cell-level rules comprising each of these M-type rule sets were then collated. The bar charts of the resulting rule frequencies are shown for each class in Figure 13. It should be noted that the bar charts show the number of occurrences of each cell-level rule within the M-type rule sets, not the number of applications of these cell-level rules in the evolution of the MSCA.



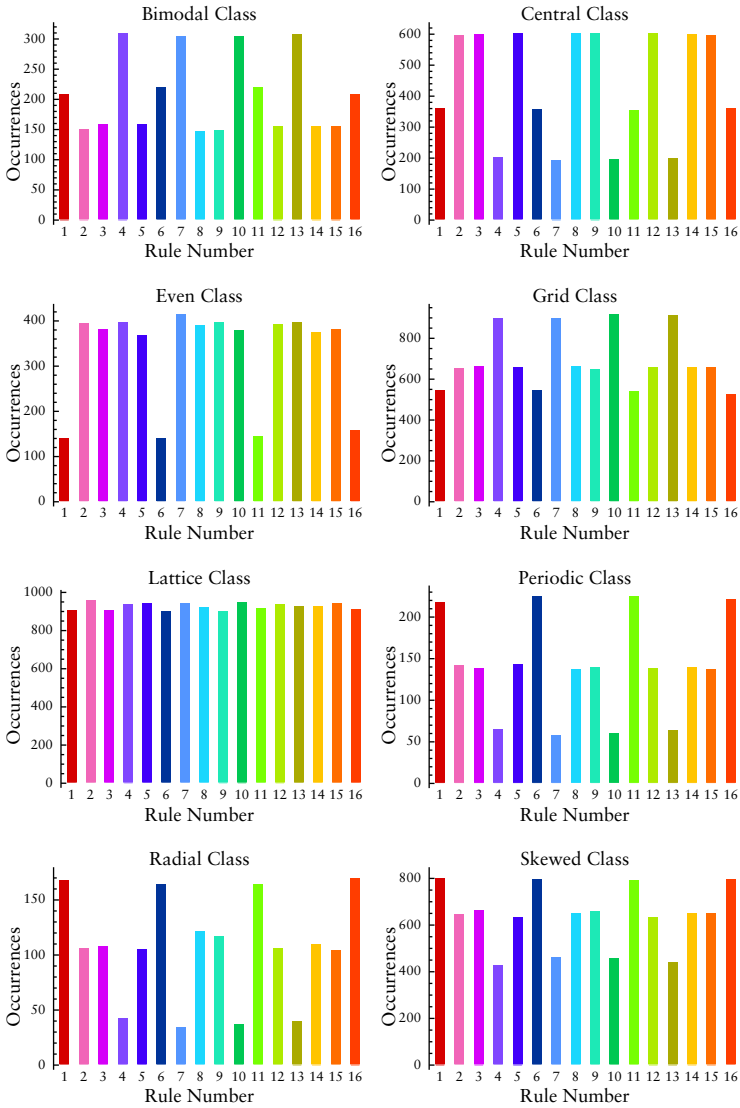


Figure 13. Bar charts for the 10 MSCA classes showing the relative impact of the 16 possible rules from Table 2. All 1296 M-type rule sets were run until convergence. Comparison across the bar charts shows the similarities between various classes.

By comparing the bar charts of Figure 13, relationships between state graphs of various classes can be observed. The bar chart for Radial and Periodic graph rule frequencies shows remarkable relative similarity for all rules, which is surprising

given the nature of the graphs is quite different. This points to underlying evolutionary similarities not evident in the final graph structure. Similar to the Radial and Periodic bar charts, the bar chart for Skewed graphs has relatively higher occurrences of rules 1 (■: 00 → 00), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11) and relatively lower occurrences of rules 4 (■: 00 → 11), 7 (■: 01 → 10), 10 (■: 10 → 01) and 13 (■: 11 → 00). However, the relative differences between rule frequency are not as great as those of Radial and Periodic graphs. Again, the Skewed graphs bear little resemblance to the Radial and Periodic classes of graphs, making the similarity of bar charts notable.

Another three classes that are worth comparing are the Arc, Axial and Bimodal classes. Their bar charts all have greater rule frequency for rules 1 (■: 00 → 00), 4 (■: 00 → 11), 6 (■: 01 → 01), 7 (■: 01 → 10), 10 (■: 10 → 01), 11 (■: 10 → 10), 13 (■: 11 → 00) and 16 (■: 11 → 11). The Axial and Bimodal are more similar than the Arc bar chart, although in the Axial bar chart, the rules that occur with higher frequency occur significantly more than the others in comparison to the Bimodal bar chart. Again, the graphs of these three classes do not appear to physically resemble each other.

The Grid bar chart has some similarity with the Bimodal bar chart. Most notable is that these classes have high frequency of rules 4 (■: 00 → 11), 7 (■: 01 → 10), 10 (■: 10 → 01) and 13 (■: 11 → 00). While the Bimodal has moderately high frequencies of rules 1 (■: 00 → 00), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11) making it relate to the Arc and Axial bar charts, the Grid bar chart shows those four rules occurring with the lowest frequency. Once again, despite the similarity in bar charts, the graphs do not appear similar in the Grid and Bimodal classes.

Finally, the Central, Even and Lattice classes all have distinct bar charts. In fact, the Central bar chart appears to be the inverse of the Bimodal bar chart with high- and low-frequency rules reversed and moderate rules remaining the same. The Even Grid bar chart shows rules 1 (■: 00 → 00), 6 (■: 01 → 01), 11 (■: 10 → 10) and 16 (■: 11 → 11) to be distinctly lower in frequency than the other rules, which is the opposite of the group of Periodic, Radial and Skewed bar charts, in which those are the rules of highest frequency. Finally, all rules occur with roughly the same frequency in the Lattice bar chart, which suggests that possibly there are subclasses to the Lattice class.

5.4 Target Analysis

The preceding analysis examined the frequency of usage of each of the 16 cell-level rules in Table 2 within the M-type rule sets and the correlation to the graph classes. This analysis instead examines the distribution of the output, or targeted cell pairs of the cell-level rules in each rule set. Since the MSCAs studied are two-block, there are four possible output targets cell pairs, $T = \{00, 01, 10, 11\}$. Eight rules from Table 2 comprise each M-type rule set in the two-branch MSCAs examined in this paper. These cell-level rules map each of the four possible input cell pairs $I = \{00, 01, 10, 11\}$ to two targets from T. In other words, there are eight targets for every rule set. Note these eight targets must have duplicates, given T has a cardinality of four. Let the counter V_w be the number of occurrences of the target T_w in the rule set h:

$$V_w = \sum 1_{T_w \in h} \text{ for } 1 \leq w \leq 4. \tag{6}$$

Define a target distribution (TD) for a rule set as follows, where the TD is the concatenation, not the multiplication, of the digits of V_w:

$$\text{TD} = V_0 \,||\, V_1 \,||\, V_2 \,||\, V_3. \tag{7}$$

There are two examples that demonstrate the utility of the TD. Consider the case $\{00 \to 00, 00 \to 01, 01 \to 00, 01 \to 01, 10 \to 00, 10 \to 01, 11 \to 00, 11 \to 01\}$, where every input cell pair is mapped to either 00 or 01. Then, out of the total eight target cell pairs, half are 00 and half are 01. In other words, four tar get cell pairs are 00, four are 01, zero are 10 and zero are 11. Thus, that case has a TD = 4400, where the TD's digits display the number of occurrences of each type of target in the rule set. Consider a second case $\{00 \to 00, 00 \to 01, 01 \to 00, 01 \to 01, 10 \to 10, 10 \to 11, 11 \to 10, 11 \to 11\}$, where two of the target blocks are 00, two are 01, two are 10, and two are 11, hence TD = 2222. In the previous 4400 case, where there were only two targets, the behavior of the MSCA was limited by the TD as reflected by the state graph's final structure. Contrastingly, in the 2222 case, there is an even distribution of the targeted cell pairs, thus fewer limitations on the behavior of the MSCA. Examining the relationship between the TD and graph classes gives different insight into the fundamental graph evolution.

Each M-type rule set contains eight cell-level rules, as described in Table 2. Thus, each TD must be four digits that sum to eight. Since the two targets for each input cell pair must be distinct, there at most are four occurrences of one target,

https://doi.org/10.25088/ComplexSystems.34.3.325

meaning each digit must be between zero and four. Therefore, the possible digit combinations are the permutations of TDs 2222, 3221, 3311, 3320, 4211, 4220, 4310 and 4400. This yields 85 different TD possibilities, one of which corresponds to each of the 1296 M-type rule sets. However, it is the frequency of targets, rather than the frequency of specific targets, that is fundamental to this analysis. Consider TD 4040 and TD 0404. The distinction between them is simply which specific targets are in the rule set four times. This is unnecessary when examining the frequency of targets in a generalized sense. As such, the 85 TDs are grouped into the original eight base TD cases: 2222, 3221, 3311, 3320, 4211, 4220, 4310 and 4400.

Let the number of rule sets with a particular TD_u be the number of rule sets with a TD that is any permutation of TD_u. Then, define the *normalized base target distribution value* (NBTD) as follows in equation (8), where TD_u is the u^{th} TD:

$$\mathrm{NBTD}_u = \frac{\#\text{ of rule sets with } \mathrm{TD}_u}{\text{total } \#\text{ of rule sets}} \text{ for } 1 \le u \le 8. \tag{8}$$

The NBTD describes the likelihood of any given M-type rule set having a certain TD. The number of graphs corresponding to each TD for every class was determined, then these counts were normalized by the size of the class to find the *normalized class target distribution value* (NCTD). Since this analysis focuses on TD, the size of the class is not of interest:

$$\mathrm{NCTD}_{c,u} = \frac{\#\text{ of graphs with } \mathrm{TD}_u \text{ in class } c}{\#\text{ of graphs in class } c} \tag{9}$$

for $1 \le u \le 8$ and $1 \le c \le 10$.

Each NCTD was normalized by the NBTD to achieve the *normalized target distribution ratio* (NTDR):

$$\mathrm{NTDR}_{c,u} = \frac{\mathrm{NCTD}_{c,u}}{\mathrm{NBTD}_u} \tag{10}$$

for $1 \le u \le 85$ and $1 \le c \le 10$.

An NTDR value of one for a particular TD indicates that the number of graphs in the specified class with that TD aligns with the expected number given the NBTD. A larger NTDR demonstrates a higher number of graphs than expected, while a lower value means fewer graphs resulted from the TD than expected. Ultimately, analyzing the NTDR as a function of graph class enables analysis of the relationship between particular TDs and classes, along with highlighting similarities and differences between classes. The NTDR for each of the eight base TD cases for each class is shown in Figure 14.

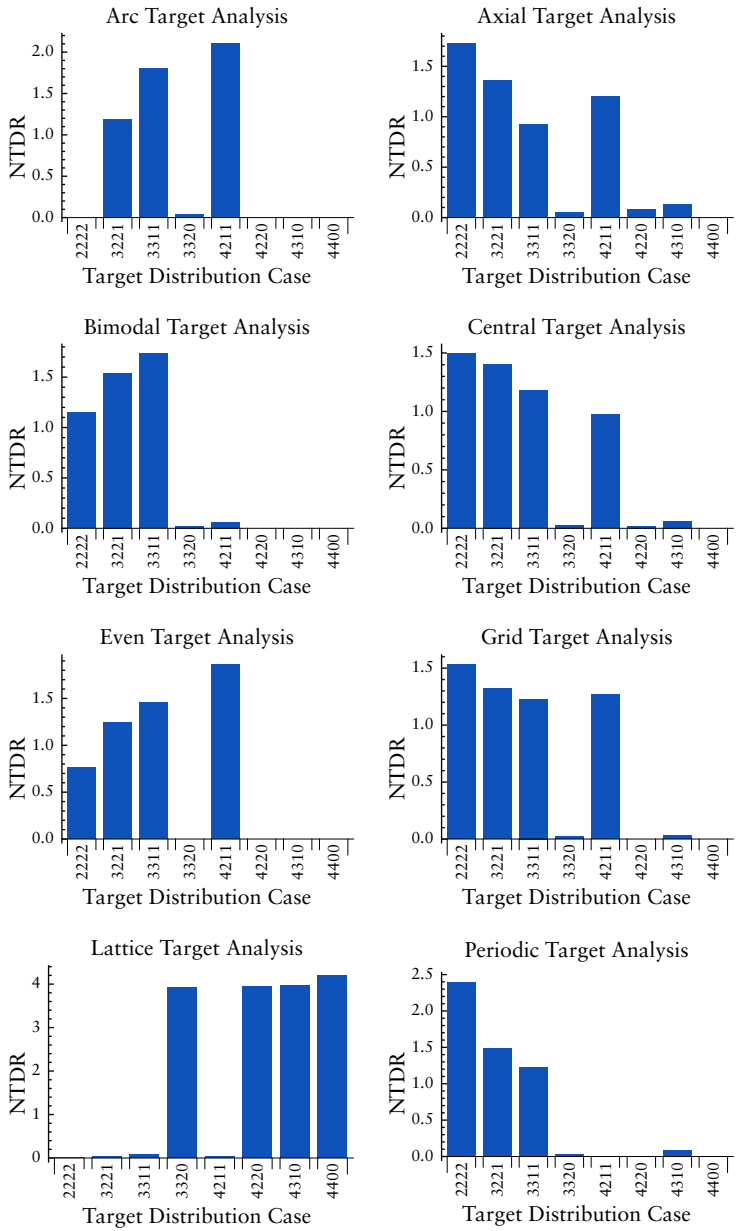


Figure 14. (*continues*).

https://doi.org/10.25088/ComplexSystems.34.3.325

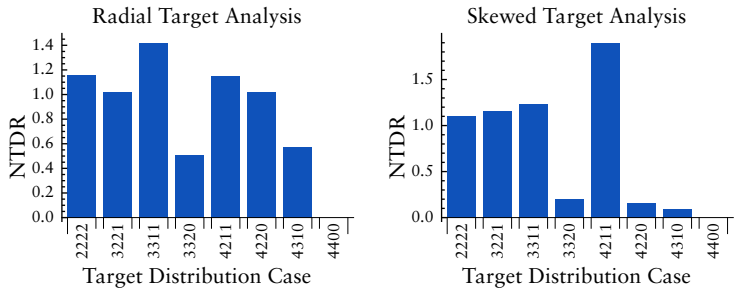


Figure 14. Bar charts of the NTDR for each of the eight base target distributions for each class of graphs. The case number shows the target cell pair distribution among the four possible cell pairs (00, 01, 10, 11), with each digit representing a cell pair. All permutations of a base target distribution case are included in the same data point. For each class, the NTDR shows on a relative basis how many of the M-type rules had the indicated target distribution.

Figure 14 shows that there are distinct differences between classes of graphs in terms of the targeting of output cell pairs. Recall that an NTDR of one is the expected value. The y axis of the majority of bar charts has the same range; however, Arc and Periodic classes have slightly larger range and the Lattice class has a significantly larger range.

In looking at the targeting data, the Axial, Central, Grid, Skewed and Even classes share strong influence of cases 2222, 3221, 3311 and 4211. The difference between these classes is in the relative amounts of each TD case. Axial, Central and Grid classes are weighted toward 2222 and 3221, whereas Skewed and Even classes are weighted toward the 4211 case. The Periodic bar chart also resembles that of the Axial, Central and Grid classes but notably lacks influence from the 4211 case. Similarly, the Bimodal class is similar to the Even and Skewed classes; however, it also lacks influence from the 4211 case. The Arc bar chart also resembles that of the Even and Skewed classes; however, it lacks participation from the 2222 case.

Of note are the completely distinct natures of the Lattice and Radial bar charts. The Lattice is the opposite, in terms of case participation, of the five classes discussed above: Axial, Central, Grid, Skewed and Even. The Radial bar chart is the only one where all cases except for the 4400 case participate in strong numbers. In fact, the 4400 case is only seen in the Lattice case and would appear to be a strong predictor of that class. Recall that this is the only case where there are only two targets. Ultimately, the distinct similarities and differences between classes

indicate that target analysis can reveal intrinsic characteristics of each class.

6. Conclusions and Future Work

A systematic analysis of multiway sequential cellular automata (MSCAs), an asynchronous updating, multi-branching extension of traditional cellular automata (CAs) that allows for propagation of effects, has been presented. A complete set of 32 initial conditions was run through the entire group of 1296 possible M-type rule sets for a two-cell input, two-cell output, two-branch base case of the MSCA, generating 41 472 state graphs. A machine learning classifier was trained and used to determine 10 classes with distinct characteristics such as node structure, node clustering and edge density.

These classes were analyzed to gain insight into the parameters that yielded each class and the characteristic of each class. Analytical data on the graphs of each class were calculated, highlighting the variance in the tenth Lattice class, hinting at the existence of subclasses. For the short arrays studied, initial conditions were shown to be of little consequence through both analysis of the percentage of each class that resulted from each of the 32 initial conditions and a study on how many initial conditions resulted in the same graph class for a given rule set. By analyzing the frequency of usage of cell-level rules, relationships between various classes and supporting evidence for the existence of subclasses were found. Finally, the target distribution of the M-type rules was studied to determine a target distribution profile for each class of MSCA. This analysis relates graph structure to structure of the rule set.

The MSCA holds promise for modeling systems with multiple updating schemes. Future work will focus on the distribution of color in graphs, which represents the specific cell-level rules within each graph. It has been noted that some graphs show concentrations of color, while others show colors linked to specific graph features. Additionally, a study is underway on the quad flex MSCA with each input cell pair mapping to a varying number of output cells, ranging from one to four. This case yields additional complexities as well as non-branching nodes. Also of interest is examining the impact of the length of the array on graph characteristics.

https://doi.org/10.25088/ComplexSystems.34.3.325

Acknowledgments

I would like to express my deepest appreciation to Dr. Stephen Wolfram for the inspiration to work on this topic and his continued encouragement. I would like to thank Hatem Elshatlawy, Lyman Hurd, James Boyd, Xerxes Arsiwalla, James Wiles and Adiba Shaikh for invaluable mentorship and Rory Foulger for unbridled support.

References

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

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

[3] N. Fatès, "A Guided Tour of Asynchronous Cellular Automata," in *Cellular Automata and Discrete Complex Systems (AUTOMATA 2013)* Giessen, Germany (J. Kari, M. Kutrib and A. Malcher, eds.), Berlin, Heidelberg: Springer, 2013 pp. 15–30. doi:10.1007/978-3-642-40867-0_2.

[4] J. L. Schiff, *Cellular Automata: A Discrete View of the World*, Hoboken, NJ: Wiley-Interscience, 2008.

[5] E. F. Codd, *Cellular Automata*, New York: Academic Press, 1968.

[6] T. M. Li, ed., *Cellular Automata*, New York: Nova Science Publishers, 2011.

[7] A. Adamatzky, ed., *Cellular Automata: A Volume in the Encyclopedia of Complexity and Systems Science*, New York: Springer, 2018.

[8] J. Hawkins, *The Mathematics of Cellular Automata*, Providence, RI: American Mathematical Society, 2024.

[9] K.-P. Hadeler and J. Müller, *Cellular Automata: Analysis and Applications*, Cham: Springer International Publishing, 2017.

[10] H. V. McIntosh, *One Dimensional Cellular Automata*, London: Luniver Press, 2009.

[11] M. Mitchell, *Complexity: A Guided Tour*, New York: Oxford University Press, 2009.

[12] J. Kari, "Theory of Cellular Automata: A Survey," *Theoretical Computer Science*, **334**(1–3), 2005 pp. 3–33. doi:10.1016/j.tcs.2004.11.021.

[13] J. von Neumann, *Theory of Self-Reproducing Automata* (A. W. Burks, ed.), Urbana, IL: University of Illinois Press, 1966.

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

[15] S. Wolfram, "Random Sequence Generation by Cellular Automata," *Advances in Applied Mathematics*, **7**(2), 1986 pp. 123–169. doi:10.1016/0196-8858(86)90028-X.

[16] E. R. Berlekamp, J. H. Conway and R. K. Guy, *Winning Ways for Your Mathematical Plays*, New York: Academic Press, 1982.

[17] J.-P. Allouche, M. Courbage and G. Skordev, "Notes on Cellular Automata," *Cubo*, **2**(3), 2001 pp. 213–244.

[18] M. Sablik and G. Theyssier, "Topological Dynamics of Cellular Automata: Dimension Matters," *Theory of Computing Systems*, **48**(3), 2011 pp. 693–714. doi:10.1007/s00224-010-9255-x.

[19] C. Bays, "Cellular Automata in Triangular, Pentagonal and Hexagonal Tessellations," *Encyclopedia of Complexity and Systems Science* (R. A. Meyers, ed.), New York: Springer, 2018 pp. 892–900. doi:10.1007/978-0-387-30440-3_58.

[20] A. S. Fauci, G. Pantaleo, S. S. Stanley and D. Weissman, "Immunopathogenic Mechanisms of HIV Infection," *Annals of Internal Medicine*, **124**(7), 1996 pp. 654–663. doi:10.7326/0003-4819-124-7-199604010-00006.

[21] J. Hawkins and D. Molinek, "Markov Cellular Automata Models for Chronic Disease Progression," *International Journal of Biomathematics*, **8**(6), 2015 1550085. doi:10.1142/S1793524515500850.

[22] I. Santé, A. M. García, D. Miranda and R. Crecente, "Cellular Automata Models for the Simulation of Real-World Urban Processes: A Review and Analysis," *Landscape and Urban Planning*, **96**(2), 2010 pp. 108–122. doi:10.1016/j.landurbplan.2010.03.001.

[23] W. Shi, M. F. Goodchild, M. Batty, M.-P. Kwan and A. Zhang, *Urban Informatics*, New York: Springer, 2021.

[24] Brady, Raghavan and Slawny, "Probabilistic Cellular Automata in Pattern Recognition," in *International 1989 Joint Conference on Neural Networks*, Washington, DC, Piscataway: Institute of Electrical and Electronics Engineers, 1989 pp. 177–182. doi:10.1109/IJCNN.1989.118577.

[25] S. Das, S. Mukherjee, N. Naskar and B. K. Sikdar, "Characterization of Single Cycle CA and Its Application in Pattern Classification," *Electronic Notes in Theoretical Computer Science*, **252**, 2009 pp. 181–203. doi:10.1016/j.entcs.2009.09.021.

[26] P. Maji and P. P. Chaudhuri, "Fuzzy Cellular Automata for Modeling Pattern Classifier," *IEICE Transactions on Information*, **88**(4), 2005 pp. 691–702. doi:10.1093/ietisy/e88-d.4.691.

[27] P. Maji, C. Shaw, N. Ganguly, B. K. Sikdar and P. P. Chaudhuri, "Theory and Application of Cellular Automata for Pattern Classification," *Fundamenta Informaticae*, **58**(3–4), 2003 pp. 321–354. dl.acm.org/doi/abs/10.5555/1006455.1006463.

[28] C. D. Thompson and H. T. Kung, "Sorting on a Mesh-Connected Parallel Computer," *Communications of the ACM*, **20**(4), 1977 pp. 263–271. doi:10.1145/359461.359481.

[29] A. R. Khan, P. P. Choudhury, K. Dihidar, S. Mitra and P. Sarkar, "VLSI Architecture of a Cellular Automata Machine," *Computers & Mathematics with Applications*, **33**(5), 1997 pp. 79–94. doi:10.1016/S0898-1221(97)00021-7.

[30] G. Ch. Sirakoulis, I. Karafyllidis, A. Thanailakis and V. Mardiris, "A Methodology for VLSI Implementation of Cellular Automata Algorithms Using VHDL," *Advances in Engineering Software*, **32**(3), 2001 pp. 189–202. doi:10.1016/S0965-9978(00)00085-5.

[31] Pries, Thanailakis and Card, "Group Properties of Cellular Automata and VLSI Applications," *IEEE Transactions on Computers*, **35**(12), 1986 pp. 1013–1024. doi:10.1109/TC.1986.1676709.

[32] L. Diosan, A. Andreica and A. Enescu, "The Use of Simple Cellular Automata in Image Processing," *Informatica*, **62**(1), 2017 pp. 5–14. doi:10.24193/subbi.2017.1.01.

[33] P. L. Rosin, "Training Cellular Automata for Image Processing," in *Image Analysis: 14th Scandinavian Conference (SCIA 2005)*, Joensuu, Finland (H. Kalviainen, J. Parkkinen and A. Kaarna, eds.), Berlin, Heidelberg: Springer, 2005 pp. 194–204. doi:10.1007/11499145_22.

[34] P. Guan, "Cellular Automaton Public-Key Cryptosystems," *Complex Systems*, **1**(1), 1987 pp. 51–57. complex-systems.com/pdf/01-1-4.pdf.

[35] H. Gutowitz, "Cryptography with Dynamical Systems," *Cellular Automata and Cooperative Systems* (N. Boccara, E. Goles, S. Martinez and P. Picco, eds.), Dordrecht: Springer, 1993 pp. 237–274. doi:10.1007/978-94-011-1691-6_21.

[36] M. Tomassini and M. Perrenoud, "Nonuniform Cellular Automata for Cryptography," *Complex Systems*, **12**(1), 2000 pp. 71–81. complex-systems.com/pdf/12-1-3.pdf.

[37] B. Harish and K. Sadiq, "Efficient Cryptography Using Cellular Automata Rules," *International Journal of Emerging Engineering Research and Technology*, **3**(12), 2015 pp. 18–25. www.ijeert.ijrsset.org/pdf/v3-i12/3.pdf.

[38] S. Nandi, B. K. Kar and P. P. Chaudhuri, "Theory and Applications of Cellular Automata in Cryptography," *IEEE Transactions on Computers*, **43**(12), 1994 pp. 1346–1357. doi:10.1109/12.338094.

[39] S. Wolfram, "Cryptography with Cellular Automata," in *Advances in Cryptology: Proceedings of CRYPTO '85* (H. C. Williams, ed.), Berlin, Heidelberg: Springer, 1985 pp. 429–432. doi:10.1007/3-540-39799-X_32.

[40] K. S. Pokkuluri, R. B. Inampudi and S. S. S. N. Usha Devi Nedunuri, "IN-MACA-MCC: Integrated Multiple Attractor Cellular Automata with Modified Clonal Classifier for Human Protein Coding and Promoter Prediction," *Advances in Bioinformatics*, **1**, 2014 261362. doi:10.1155/2014/261362.

[41] K. Steiglitz, I. Kamal and A. Watson, "Embedding Computation in One-Dimensional Automata by Phase Coding Solitons," *IEEE Transactions on Computers*, **37**(2), 1988 pp. 138–145. doi:10.1109/12.2143.

[42] F. V. Haesler, H. O. Peitgen and G. Skordev, "Cellular Automata, Matrix Substitutions and Fractals," *Annals of Mathematics and Artificial Intelligence*, **8**(3), 1993 pp. 345–362. doi:10.1007/BF01530797.

[43] K. Culik II and S. Dube, "Fractal and Recurrent Behavior of Cellular Automata," *Complex Systems*, **3**(3), 1989 pp. 253–267. complex-systems.com/pdf/03-3-3.pdf.

[44] B. Martin, "Inherent Generation of Fractals by Cellular Automata," *Complex Systems*, **8**(5), 1994 pp. 347–366. complex-systems.com/pdf/08-5-4.pdf.

[45] S. J. Willson, "Cellular Automata Can Generate Fractals," *Discrete Applied Mathematics*, **8**(1), 1984 pp. 91–99. doi:10.1016/0166-218X(84)90082-9.

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

[47] G. J. Martinez, J. C. Seck-Tuoh-Mora and H. Zenil, "Computation and Universality: Class IV versus Class III Cellular Automata," *Journal of Cellular Automata*, **7**(5–6), 2012 pp. 393–430.

[48] H. Zenil, "Compression-Based Investigation of the Dynamical Properties of Cellular Automata and Other Systems," *Complex Systems*, **19**(1), 2010 pp. 1–28. doi:10.25088/ComplexSystems.19.1.1.

[49] H. Zenil, N. A. Kiani and J. Tegnér, "Low-Algorithmic-Complexity Entropy-Deceiving Graphs," *Physical Review E*, **96**(1), 2017 012308. doi:10.1103/PhysRevE.96.012308.

[50] A. Adamatzky, *Identification of Cellular Automata*, Bristol, PA: Taylor and Francis, 1994.

[51] W. Li and N. Packard, "The Structure of the Elementary Cellular Automata Rule Space," *Complex Systems*, **4**(3), 1990 pp. 281–297. complex-systems.com/pdf/04-3-3.pdf.

[52] Y. Aizawa and I. Nishikawa, "Toward the Classification of the Patterns Generated by One-Dimensional Cellular Automata," *Dynamical Systems and Nonlinear Oscillators* (G. Ikegami, ed.), World Scientific Press, 1986 pp. 210–222.

[53] K. Sutner, "Classification of Cellular Automata," *Encyclopedia of Complexity and Systems Science*, (R. A. Meyers, ed.), New York: Springer, 2009 pp. 755–768.

[54] L. P. Hurd, J. Kari and K. Culik, "The Topological Entropy of Cellular Automata Is Uncomputable," *Ergodic Theory and Dynamical Systems*, **12**(2), 1992 pp. 255–265. doi:10.1017/S0143385700006738.

[55] J. T. Baldwin and S. Shelah, "On the Classifiability of Cellular Automata," *Theoretical Computer Science*, **230**(1–2), 2000 pp. 117–129. doi:10.1016/S0304-3975(99)00042-0.

[56] R. H. Gilman, "Classes of Linear Automata," *Ergodic Theory and Dynamical Systems*, **7**(1), 1987 pp. 105–118. doi:10.1017/S0143385700003837.

[57] P. Kurka, "Languages, Equicontinuity and Attractors in Cellular Automata," *Ergodic Theory and Dynamical Systems*, **17**(2), 1997 pp. 417–433. doi:10.1017/S014338579706985X.

[58] P. Kurka, "Topological Dynamics of Cellular Automata," *Codes, Systems, and Graphical Models* (B. Marcus and J. Rosenthal, eds.), New York: Springer, 2001 pp. 447–485. doi:10.1007/978-1-4613-0165-3_25.