The Elementary Cellular Automata A Journey into the COMPUTATIONAL WORLD

JONAKI B GHOSH & ROHIT ADSULE

"It's always seemed like a big mystery how nature, seemingly so effortlessly, manages to produce so much that seems to us so complex." – Stephan Wolfram

The topic of cellular automata has many interesting and wide ranging applications to real life problems emerging from areas such as image processing, cryptography, neural networks, developing electronic devices to modelling biological systems. In fact cellular automata can be a powerful tool for modelling many kinds of systems. They may be described as mathematical models for systems in which simple components act together to produce complicated patterns of behaviour. A large number of cellular automata (CA) models are used to study physical, biological, chemical or social phenomena involving interacting entities. Examples of such CA applications may be as diverse as modelling traffic flow, crystal growth, ant colony activity or forest fires. A popular and well-known example of CA is The Game of Life which was originally developed by John Conway, a British Mathematician in 1970. Though the game appears to be a simple 'toy' played by applying a few simple rules in a two-dimensional grid of square cells, it has been the springboard for the study of 'artificial life' systems because of the amazingly complex behaviour displayed by some of the patterns which emerge when this CA evolves. More details about the Game of Life may be found in [2].

Keywords: Cellular automata, neural network, ECA, Mathematica, computer algebra system, Boolean expression, Karnaugh map

The topic of Cellular Automata lends itself to interesting investigations which are well within the reach of high school students. As we hope to illustrate in this article, the ideas are simple and yet powerful. We shall describe briefly our attempts to investigate this unique and interesting topic.

Briefly defined, a cellular automaton is a collection of cells on a grid of a specified shape that evolves through discrete time steps according to a set of rules based on the state (or color) of the neighbouring cells. Cellular Automata may be one, two or three-dimensional. Here we will describe an exploratory project where we attempted to explore the one-dimensional Elementary Cellular Automata (ECA) as defined by Stephan Wolfram using Mathematica, a Computer Algebra System. A part of the explorations were also done using NICO [5] an open source software. Mathematica's extensive numeric as well as easy-to-use graphic capabilities along with inbuilt commands for cellular automata make it very conducive for exploring this topic. Readers with access to Mathematica may explore the topic using the commands described in a later section. Others may use the NICO software.

The aim of the project was to

a) Explore the evolution of the ECAs;

b) Represent the ECA rules in decimal, binary, and Boolean forms;

c) Classify all the ECAs into specific categories depending on the patterns emerging from their evolution. Mathematica programming and the NICO software were used to generate pictures (graphics) of the 256 ECAs and observe their patterns;

d) Explore the sensitivity of the ECAs to initial conditions.

Some Mathematical Preliminaries

As mentioned earlier, a cellular automaton is a collection of coloured cells on a grid of a specified shape that evolves through discrete time steps according to a set of rules based on the state (or color) of the neighbouring cells. One-dimensional cellular automata are usually found on triangular, square or hexagonal grids. We shall limit our discussion to cellular automata on a square grid, that is, on a grid of square cells.

The defining characteristics of cellular automata are as follows.

i) A cellular automaton develops on a grid of cells.

ii) Each cell has a state – dead or alive. Live cells are coloured black or assigned a value 1 whereas dead cells are white and assigned a value of 0. Colours other than black or white may also be used.

iii) Each cell in the grid has a neighbourhood. A neighbourhood of a given cell is a set of cells which are adjacent to it. This may be chosen in various ways. For example, if we consider a linear grid of square cells, then the neighbourhood of each cell may be the two adjacent cells – one to its left and the other to its right.

iv) Finally every cellular automaton must have a defining rule based on which it grows and evolves in discrete time steps. For example, in a square grid, each row of cells may be considered as a different generation of cells. Thus the first row is the initial generation (or generation 0) where each cell has a state (0 or 1). The state of each cell in the second row must be a function of its neighboring cells in the row above it (that is the initial row). This may be written as

$$(\text{Cell state}_t) = f(\text{Neighboring Cell state}_{t-1})$$

To begin with let us consider a linear grid of 8 cells where every cell has state 0 except the 5th cell which has state 1.

Figure 1: A linear grid of 8 cells where the 5th cell is a live cell.

This linear grid of square cells will be referred to as **generation 0** (or row 0). The states of cells in

generation 1(that is row 1) will be determined by the neighborhood of each cell in row 0 which comprises of the three cells just above it. Clearly the states of these three cells may be any one of the following

000 001 010 100 011 101 110 111

The fact that there are three cells and the state of each cell is either 0 or 1 implies that there are $2^3 = 8$ ways of colouring these cells. Thus there are 8 neighbourhood configurations described by the triples of 0's and 1's as shown above. Conventionally, while defining an ECA, these neighbourhoods are taken in the following specific order.

111 110 101 100 011 010 001 000

Each of these configurations will determine the state of the middle cell of the three cells just below it in the next row, which may again be either 0 or 1. However the state of the leftmost corner cell in row 1 will be determined by the state of the cell just above it in row 0, its right neighbour, and the last cell in row 0. Similarly, the state of the rightmost corner cell in row 1 will be determined by the state of the cell just above it in row 0, its left neighbour and the first cell in row 0.

Let us now arbitrarily assign 0's and 1's to all 8 neighbourhood configurations as follows

111 110 101 100 011 010 001 000

0 0 0 1 1 1 1 0

Pictorially this may be represented as

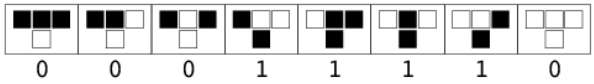


Figure 2: A rule set for a one-dimensional cellular automaton

This arbitrary assignment (also known as the rule set) will be the defining rule which will determine how this particular automaton will evolve. Note that this defining rule 00011110 may be treated as a binary number whose decimal representation may be obtained as follows

$0 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 30$

This kind of a rule set generates an elementary cellular automaton. The ECA which evolves from this rule set is referred to as **Rule 30**. However a different assignment of 0's and 1's would lead to a different rule set and a different ECA. Since each of the 8 groups of three cells may be assigned 0 or 1, this leads to $2^8 = 256$ possible assignments. Thus, in all, there are 256 ECA rules.

Equivalent expressions of ECA rules

We have seen that each ECA rule has a decimal representation as well as a binary representation. For example, let us consider rule 110. Its binary representation may be obtained by expanding 110 in powers of 2 as follows

$0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 110$

Here 110 has been written in base 2. Reading off the coefficients of the powers of 2, we get 01101110, which is the binary representation of 110.

Pictorially this translates to the following

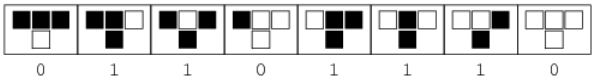


Figure 3: The elementary Cellular Automaton 110

Boolean expressions for ECA rules

Sometimes it is useful to represent the Cellular Automata rules using logical expressions. In such a case we need to find the Boolean expressions equivalent to the rule. This section is for the reader who is interested to understand the details of converting the ECA rules to Boolean expressions. However, the reader may skip this section and move to the next section on Mathematica explorations in case he or she wishes to omit the details. We will now highlight the method of finding Boolean expression for a given rule, say rule 110, using the concept of Karnaugh maps [4]. A detailed treatment of the method is described in [5].

In this example three input variables can be combined in 8 different ways. Thus a truth table with three arguments p, q and r as input variables will have 8 rows. The 8 possible configurations of the truth values of p, q and r (where 1 corresponds to true and 0 corresponds to false) correspond to the 8 neighbourhood configurations of rule 110 as discussed above. Further the truth value of each row corresponds to the binary digits of 110 as follows:

Row number	p	q	r	f (p, q, r)
1	1	1	1	0
2	1	1	0	1
3	1	0	1	1
4	1	0	0	0
5	0	1	1	1
6	0	1	0	1
7	0	0	1	1
8	0	0	0	0

Table 1: Truth table for three arguments p, q and r

There are two different unsimplified boolean expressions representing the function f(p, q, r), using the Boolean variables p, q, r and their inverses. These are

$f(p, q, r) = \sum m_i,\ i \in \{2,3,5,6,7\}$ where m_i are the miniterms to map (that is, rows that have output 1 in the truth table).

$f(p, q, r) = \sum M_i,\ i \in \{1,4,8\}$ where M_i are the maxterms to map (that is, rows that have output 0 in the truth table).

Using the concept of Karnaugh maps (which are used to simplify Boolean expressions), we obtain Table 2 (placing 1's in the cells which correspond to 1 in the f(p, q, r) column). For example, row 7 corresponds to 001 (values of p, q and r respectively) and has truth value 1. Thus, in the following grid, we place 1 in the cell which corresponds to 001 (that is, in cell (2, 3)). Similarly we fill in the cells which correspond to rows with truth value 1 (note that there are five such rows and hence the following table has five 1's).

p/qr	00	01	11	10
0		1	1	1
1		1		1

Table 2: The Karnaugh map table

Now that the Karnaugh table has been constructed, we shall try to find the simplest Boolean expression to define the rule 110. To achieve this, we will group the 1's inside the table in sets of 2, 4, 8 respectively (that is in powers of 2). These groupings, referred to as miniterm groupings, must be done in such a way that adjacent 1's in the Karnaugh table are encircled (or grouped in rectangles). Thus the set of 2 groupings (since 4, 8, etc., are not possible) must include all the five 1's in the table and may be taken as follows:

Group 1: These are the two 1's in column 3

Group 2: These are the two 1's in column 5

Group 3: These are the two 1's appearing in the 3rd and 4th cells of row 2. (Instead we could have also taken the 1's appearing in the 2nd and 3rd cells of row 2. The groups may intersect as groups 2 and 3 here).

These translate to q'r (group 1- since the value of q is 0 and the value of r is 1), qr' (group 2 - since the value of q is 1 and the value of r is 0) and p'r (group 3 – value of p is 0 and that of r changes from 1 to 0).

The simplified Boolean expression is = p'r + q'r + qr'

Using the 'and', 'or' and 'not' notations of logic denoted by the symbols $\wedge$, $\vee$, and $\sim$ respectively the above expression may also be expressed as

$(\sim p \wedge r) \vee (\sim q \wedge r) \vee (q \wedge \sim r)$

(note that '+' is replaced by $\vee$, ' . ' is replaced by $\wedge$ and ' is replaced by $\sim$)

This may be further simplified using the XOR notation as

$(\sim p \wedge r) \vee (q \oplus r)$

(Note that XOR is referred to as the exclusive or argument in logic and is represented by the symbol ⊕. A ⊕ B is true when either A or B is true (or has value 1)).

In a similar manner using the concept of Karnaugh maps we were able to find Boolean expressions corresponding to several other ECA rules.

A *Mathematica* based exploration

Mathematica is a powerful Computer Algebra System which can be used to explore the Elementary Cellular Automata. In this section we shall use Mathematica to obtain the graphic (pictorial) representations of all the 256 ECAs. The aim is to observe the evolutionary pattern of each ECA through the first 100 iterations and to categorise them into specific classes based on the patterns manifested by them.

The inbuilt *Mathematica* command for exploring cellular automata is

```
CellularAutomaton[rule, init, t]
```

Here:

rule stands for the decimal representation of the rule set in binary form. For example, 26 is the decimal representation of the rule 0 0 0 1 1 0 1 0.

init represents the initial condition or generation 0.

t denotes the number of steps

For example, the **rule 30** elementary cellular automata after 10 iterations, with an initial condition of 9 cells, where only the 5th cell is alive may be computed as follows

```
CellularAutomaton[30, {0, 0, 0,
0, 1, 0, 0, 0, 0}, 10]
```

Note that the output is displayed as a list of lists where each list is the state of the automaton at time t.

```
{{0, 0, 0, 0, 1, 0, 0, 0, 0},
{0, 0, 0, 1, 1, 1, 0, 0, 0}, {0,
0, 1, 1, 0, 0, 1, 0, 0}, {0, 1,
1, 0, 1, 1, 1, 1, 0}, {1, 1, 0,
0, 1, 0, 0, 0, 1}, {0, 0, 1, 1,
1, 1, 0, 1, 1}, {1, 1, 1, 0, 0,
0, 0, 1, 0}, {1, 0, 0, 1, 0, 0,
1, 1, 0}, {1, 1, 1, 1, 1, 1, 1,
0, 0}, {1, 0, 0, 0, 0, 0, 0, 1,
1}, {0, 1, 0, 0, 0, 0, 1, 1, 0}}
```

By adding the //**TableForm** command at the end of the code, the output is obtained in the form of a table or matrix.

```
CellularAutomaton[30, {0, 0,
0, 0, 1, 0, 0, 0, 0}, 10]//
TableForm
000010000
000111000
001100100
011011110
110010001
001111011
111000010
100100110
111111100
100000011
010000110
```

However, if we wish to create the automaton with a larger number of cells in row 0 having a single live cell in the middle we may replace **init** by **{{1}, 0}**. By doing this Mathematica adjusts the number of cells in row 0 according to the number of iterations required. Thus

```
CellularAutomaton[30, {{1}, 0},
10]//TableForm
```

outputs 10 rows (iterations) with an initial row consisting of a single live cell in the centre (with value 1) and 10 dead cells (value 0) on either side.

```
00000000010000000000
00000000111000000000
00000001100100000000
00000011011110000000
00000110010001000000
00001101111011100000
00011001000010010000
00110111100111111000
```

```
00110010001110000010 0
01101111011001000111 0
11001000010111101100 1
```

Whereas

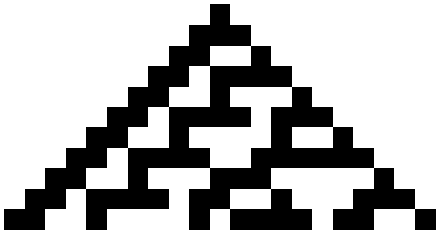
```
CellularAutomaton[30, {{1}, 0},
20]//TableForm
```

will display the output with the initial row having 1 live middle cell and 20 cells of value 0 on either side.

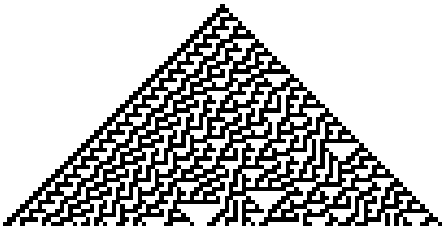
The above codes display the output only in binary form and do not help us to visualise them. For obtaining the visual image of the automata we use the **ArrayPlot** command.

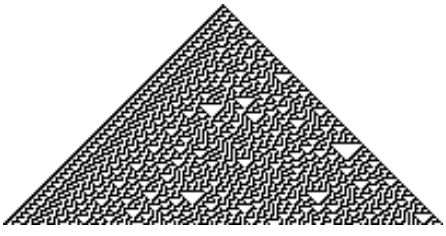
Thus the first 10 iterations of Rule 30 with a single live cell in the initial condition is as follows. Note that all cells with state 1 are black whereas the ones with state 0 remain white.

```
ArrayPlot[CellularAutomaton[30,
{{1}, 0}, 10]]
```



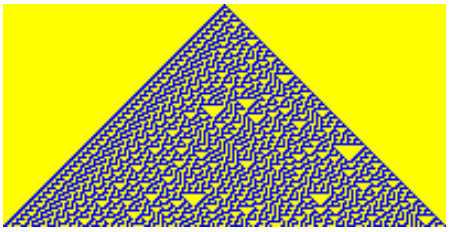
Increasing the iterations to 50 and 100 (by changing 10 to 50 and 100 respectively in the above code) yields the following outputs:





We can also add colour to the output by using the **ColorRules** option within the **ArrayPlot** command as follows

```
ArrayPlot[CellularAutomaton[30,
{{1}, 0}, 100], ColorRules->
{1→Blue, 0->Yellow}]
```



We observe that the evolution of rule 30 is random. There appears to be a continuous border on the left-hand side of the 'triangle'. However there is no fixed pattern in the yellow triangles which are randomly spread across the pattern.

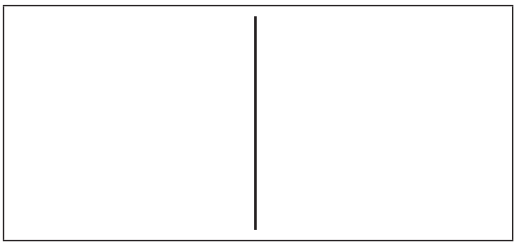
Classification and sensitivity analysis of the ECAs

Using *Mathematica* and NICO software (details are mentioned later in the article) we were able to explore the evolution of all the 256 ECA rules and classify them into four major categories which are also mentioned in the research literature associated with cellular automata.

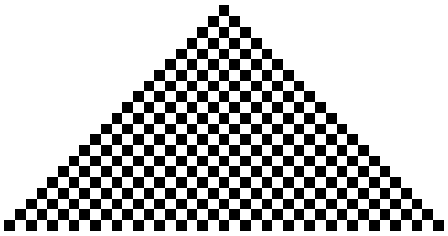
1. **Uniform:** where all cells are either black or white.
2. **Repetitive:** Some patterns are repetitive having a regular alternating pattern or a block of cells which repeat themselves throughout.
3. **Nested or Fractal–like:** These automata lead to Sierspinski triangle like fractal patterns exhibiting clear self- similarity or other nested patterns.
4. **Random or chaotic:** These are patterns which cannot be placed in any of the above three categories. There is no fixed pattern in these automata and their evolution is highly unpredictable.

Here are some examples of ECAs which evolve from one live cell in the centre of the top row of the grid

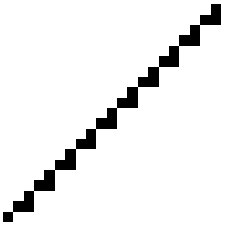
Rule 151: Uniform – all cells are black



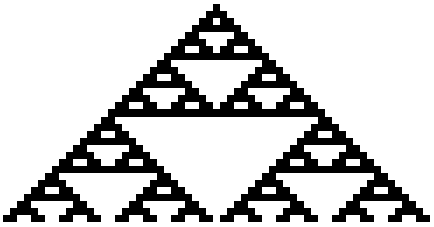
Rule 4: Repetitive: stationary – all cells are black in the same location



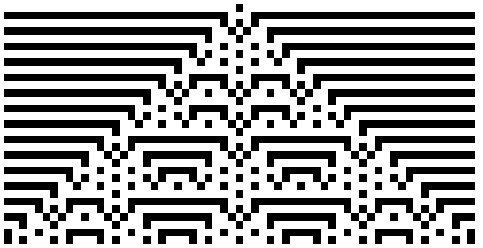
Rule 50: Repetitive: alternating black and white cells remain the same throughout



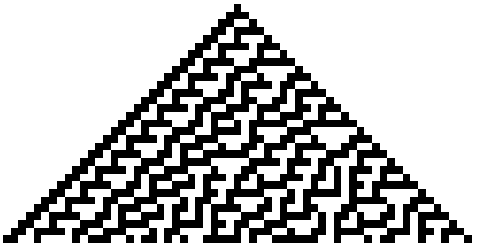
Rule 6: Repetitive: non-stationary: the same pattern is repeated in a different location.



Rule 126: Nested: looks like the Sierpinski triangle pattern



Rule 105: Nested: nested behaviour appearing in symmetrical patterns



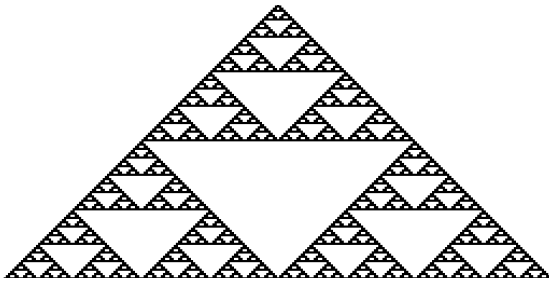
Rule 30: Random: completely random behaviour

Exploring the ECAs online

The NICO'S ELEMENTARY CELLULAR AUTOMATA software may be accessed through link https://sciencevsmagic. net/eca/#. It provides a wonderful opportunity to explore the ECAs even if one doesn't have access to a sophisticated computer algebra system such as Mathematica.

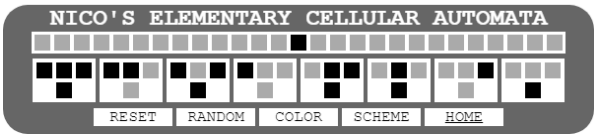
In order to explore a particular ECA rule one may enter the rule number after the # symbol.

Thus https://sciencevsmagic. net/eca/#126 will lead to the Sierpinski triangle like pattern shown below.



The user may also obtain an ECA by specifying the initial condition and the defining rule of the automaton by clicking on the relevant cells in the

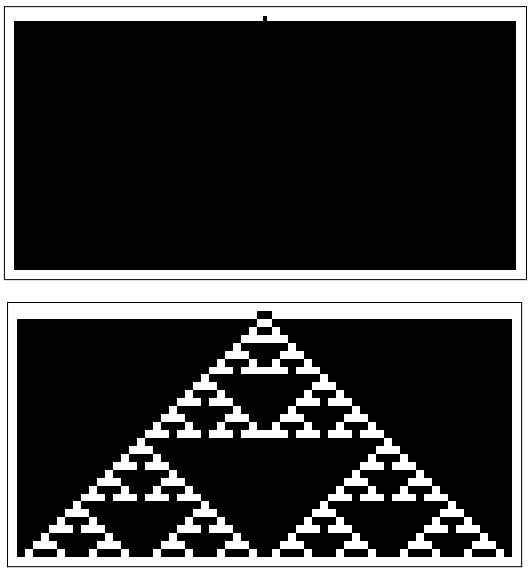
options provided at the bottom of the screen as shown.



Sensitivity analysis of the ECAs

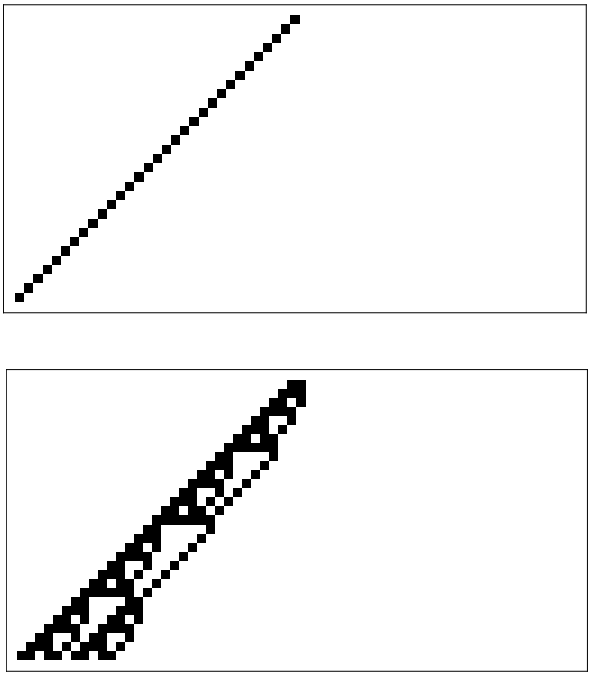
We tried to explore the sensitivity of each ECA rule to the specified initial conditions using both *Mathematica* as well as NICO. In this section we have included the Mathematica images. In the previous section the automata have been generated by considering one live cell positioned in the centre (value 1) of a linear grid of cells. We were interested to see if making a slight variation in the initial condition significantly impacted the evolution of the cellular automata.

For example, Rule 151 with one black cell in its initial row leads to all black cells (as we have seen above). However if we add another black cell to the initial condition we get a Sierpinski-like nested pattern.



Rule 151 for different initial conditions which differ by only one black cell.

Similarly Rule 106 displays a repetitive non-stationary evolution where the same pattern is repeated in a different location. However by making a slight change in its initial condition we get a random behaviour.



Rule 106 for different initial conditions which differ by only one black cell.

We tested all 256 ECAs for sensitivity using Mathematica and NICO. The findings have been summarised in Table 1.

Category	Rule Number(s)	Characteristics	Sensitivity to Initial Conditions
Uniform	0, 8, 32, 40, 64, 72, 96, 104, 128, 136, 160, 168, 192, 200, 224, 232	All cells are white.	Not sensitive to initial conditions.
Uniform	151, 159, 183, 191, 215, 223, 233, 235, 237, 239, 247, 249, 251, 253, 255	All cells are black.	Rule 151 is very sensitive and leads to randomness and Sierpinski-like structures. Rule 183 leads to nestedness.
Repetitive (Stationary)	1, 4, 5, 7, 12, 19, 21, 23, 29, 31, 33, 36, 37, 44, 51, 55, 63, 68, 71, 76, 87, 91, 95, 100, 108, 119, 123, 127, 132, 140, 164, 172, 196, 201, 203, 204, 205, 207, 217, 219, 221, 228, 236	The same pattern is repeated in the same location.	Same structure retained with minor variations.
Repetitive (Single Triangular pattern)	50, 54, 58, 77, 94, 109, 114, 122, 133, 147, 158, 163, 177, 178, 179, 186, 190, 214, 222, 242, 246, 250, 254	A single triangle is formed in which the same pattern continues throughout although inner variations may occur.	109 and 133 are very sensitive to initial conditions, and lead to randomness. Rule 122 leads to Sierpinski-like structure.
Repetitive (Non-Stationary)	2, 3, 6, 9, 10, 11, 14, 15, 16, 17, 20, 24, 25, 27, 34, 35, 38, 39, 41, 42, 43, 46, 47, 48, 49, 52, 53, 56, 59, 61, 65, 66, 67, 74, 80, 81, 83, 84, 85, 88, 97, 98, 103, 106, 107, 111, 112, 113, 115, 116, 117, 120, 121, 125, 130, 134, 138, 139, 142, 143, 144, 148, 152, 155, 162, 166, 170, 171, 173, 174, 175, 176, 180, 184, 185, 187, 189, 194, 202, 208, 209, 211, 212, 213, 216, 226, 227, 229, 231, 234, 240, 241, 243, 244, 245, 248	The same pattern is repeated in a different location.	Rules 106, 120 are very sensitive and lead to randomness.
One-Sided Pattern	13, 28, 60, 69, 70, 78, 79, 92, 93, 102, 110, 124, 137, 141, 153, 156, 157, 188, 193, 195, 197, 198, 199, 206, 220, 230, 238, 252	Forms a definite shape on only one side.	Some rules, such as 124 and 137 displayed chaotic behaviour.
Nested	89, 105, 150	Shows nested patterns	
Sierpinski Triangle	18, 22, 26, 82, 90, 126, 129, 146, 154, 161, 165, 167, 181, 182, 210, 218	Lead to a Sierpinski-Triangle structure	Rules 22 and 182 are very sensitive to initial conditions and lead to randomness.
Multiple Patterns	57, 62, 73, 99, 118, 131, 145	Contains two patterns in the same rule. Many of these are symmetric.	In Rule 73, the symmetry disappears.
Random	30, 45, 75, 86, 89, 101, 135, 149, 169, 225	The evolution seems to be random.	Randomness is retained

Table 3: Classification and sensitivity analysis of the 256 ECAs

Closing Remarks

In this article we have discussed the basics of the one-dimensional Elementary Cellular Automata as described by Stephan Wolfram. Our explorations have convinced us that simple rules can lead to interesting and complicated evolution patterns. These can be classified into the four categories: Uniform, Repetitive, Nested and Random, as described in the earlier section. However some of these ECAs are quite sensitive to initial conditions. Changing the state of one single cell in the initial row of the automata may lead to a completely different pattern. Thus the ECA may show transition from one category to another.

The topic of cellular automata offers tremendous scope for investigations. In the subsequent article, we shall detail elementary cellular automata which we have created by defining our own CA rules and we hope to highlight some of their interesting properties.

References

1. Gage, D., Laub, E., McGarry, B. Cellular Automata: Is rule 30 random? extracted from https://www. cs. indiana. edu/~dgerman/2005midwestNKSconference/dgelbm. pdf
2. Conway's Game of Life extracted from https://en. wikipedia. org/wiki/Conway%27s_Game_of_Life
3. Georgiadis, E., (2007). A Note on Minimal Boolean Formula Size of One-Dimensional Cellular Automata, Massachusetts Institute of Technology, Cambridge, MA 02139, U. S. A
4. Karnaugh maps retrieved from https://en. wikipedia. org/wiki/Karnaugh_map
5. House, A. (2003). Introduction to Karnaugh Maps extracted from http://www. eecg. toronto. edu/~ahouse/mirror/engi3861/kmaps. pdf
6. Nico's Elementary Cellular Automata retrieved from https://sciencevsmagic. net/eca/#176
7. Wolfram, S., (1985). Two-Dimensional Cellular Automata In *Journal of Statistical Physics*, 38, 901- 946.
8. Wolfram, S., (1986). Cellular Automata and Cryptography In Crypto '85 Proceedings, Lecture Notes in ComputerScience, 218, 429-432 (Springer-Verlag, 1986).
9. Yang, Y., Billings, S. A. (2000). Extracting Boolean Rules from CA Patterns In *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics,* 30 (4), 575 - 581.

JONAKI GHOSH is an Assistant Professor in the Dept. of Elementary Education, Lady Sri Ram College, University of Delhi where she teaches courses related to math education. She obtained her Ph.D. in Applied Mathematics from Jamia Milia Islamia University, New Delhi, and her M.Sc. from IIT Kanpur. She has taught mathematics at the Delhi Public School, R K Puram, where she set up the Math Laboratory & Technology Centre. She has started a Foundation through which she conducts professional development programmes for math teachers. Her primary area of research interest is in the use of technology in mathematics instruction. She is a member of the Indo Swedish Working Group on Mathematics Education. She regularly participates in national and international conferences. She has published articles in journals and authored books for school students. She may be contacted at jonakibghosh@gmail.com.

ROHIT ADSULE is a 12th grade student from The Shri Ram School Aravali, Gurugram, Haryana. His hobbies include chess, piano and football. He loves integrating his skills in mathematics with concepts related to computer science, and it was thus he came across the topic of cellular automata. He wishes to explore and understand the connection of mathematics to seemingly unrelated fields such as music theory and behavioural mechanics.