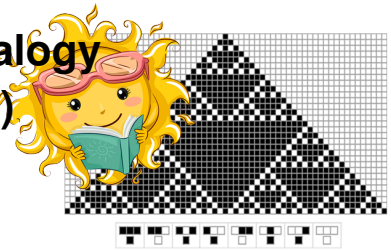
Information-Theoretic Analysis of ECA Rules (Genealogy Interceded Phenotypic Analysis (GIPA) of ECA rules)

R Goyal

August 24, 2022



This study demonstrates the grouping of the ECA Rules using information-theoretic measures in light of Wolfram's classification of ECA, with 88 rule equivalence classes. A formal qualitative behavior of the space-time diagrams for different rules gives rise to the macroscopic parameters, whose quantified (numerical) evaluation provides an opportunity for further analysis through machine learning-based approaches to extract interesting patterns. In this study, analysis has been done for a fixed single-cell scenario for different numbers of cells and iterations, with additional experiments for difference analysis with random inputs. This study is based on the prior studies by Borriello, and Chliamovitch. Adverting to the prior expositions, Hector Zenil has done pioneering work in algorithmic probability and analyzed information dynamics of cellular automata from different perspectives. This study, however, analyzes at the level of rules; therefore, we use the concept of BiEntropy, which was proposed by Grenville J. Croll to compute approximate information content of a binary string (see Eq. 1).

$$H(p) = \frac{1}{2^{n-1}-1}\sum_{k=0}^{n-2}(-p(k)\log_2 p(k) - (1-p(k))\log_2(1-p(k)))2^k \tag{1}$$

The amount of information processed by a CA rule in a space-time patch is captured through four measures.

- DiffEntropy (DE): Maximum absolute difference in the BiEntropy values of any reachable configuration (C_j) from the initial configuration (C_i), e.g.,

$$DE = Max(abs(BiEntropy(C_j) - BiEntropy(C_i))) \tag{2}$$

  This parameter captures the impact of the transformation carried out by a rule.

- SimConfigOrdered (SCO): Count the similar BiEntropy values (based on a threshold value, i,e, 0.01) of two configurations in a sorted list of entropy values.

$$SCO = BiEntropy(C_i) - BiEntropy(C_j) \tag{3}$$

  This parameter captures the frequency of similar (information content-wise) reachable configurations during processing.

- SimConfigImmediate (SCI): Count the similar BiEntropy values (based on a threshold value, i,e, 0.01) of two successive configurations.

$$SCI = BiEntropy(C_i) - BiEntropy(C_{i+1}) \tag{4}$$

- SimConfigFluctuation (SCF): Count the fluctuations in the BiEntropy values (where DE/2> 0.01) of two successive configurations.

$$SCF = abs(BiEntropy(C_i) - BiEntropy(C_{i+1})) > DE/2 \tag{5}$$

  This parameter captures the higher fluctuations (> DE/2) of successive reachable configurations during processing.

Category-I

- Entropy (BiEnrtropy values) does not change, though string may get change (like in Rule 51); because change still leads to a pattern (like 1's complement in the case of Rule 51), the entropy value remains the same.
- All rules (strong) of Category-I are characterized by the zero DE, SCI, and SCF values with very high SCO values.

Category-II

- Entropy (BiEntropy values) stabilizes (string either becomes all zero's or a well-observed pattern) from initial high or fluctuating entropy values.
- Rules with a strong association of Category-II characterized by the low to high DE and very high SCO values with zero and one value of SCI and SCF values respectively, whereas, for weak associations, DE and SCO values are usually high with low SCI and SCF values.

Category-III

- Entropy (BiEntropy values) values fluctuate or result in a periodic behavior (BiEntropy values usually remain high) from initial high or fluctuating entropy values.
- All Rules (strong and weak) of Category-III are characterized by the low to high DE and very high SCO values with low to medium SCI and SCF values values (except for Rule 1 and 33, wherein pretty high, e.g., 79, SCI and SCF values alternates)

Category-IV

- Entropy (BiEntropy values) exhibit chaotic beahiour with smaller cell sizes (value of n), which stabilizes as n increases (i.e., $n = 32$).
- All Rules of Category-IV are characterized by the low to high DE and high SCO values with very low SCI and low to high SCF values.

Category-V

- Rules in Category-V exhibit mixed behavior (of previous categories) with different values of cell length (value of n).
- None of these rules showed a zero value (zero indicates a high level of overlapping) of DTW analysis with any other rule.

Table 1: Category-wise variation in the four measurements (i.e., DE, SCO, SCI, and SCF)

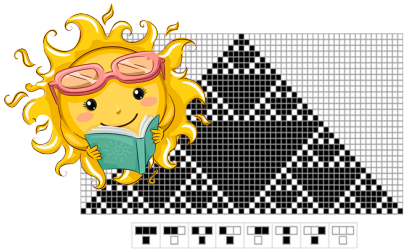
Category	DE	SCO	SCI	SCF
I	0	very high	0	0
II	low to high	very high	0	1
III	low to high	very high	low to medium	low to medium
IV	low to high	high	low	low to high
V	Mixed			

Cellular Automata as models in Social Sciences

Rezki Chemlal[†]

[†]Bejaia University

August 24, 2022



In this project we were interested in the use of 2 dimension cellular automata in social sciences , we were very flexible in the last point as we also inculed models in urban modeling.We also did not restrict ourselves to the formal definition of cellular automata as we considered also interesting models that may be considered close to cellular automata. We realized programs for all models

Schelling's model of segregation is a model developed by economist Thomas Schelling. Consider two social categories (State 1 , State 2). At each step a cell will check if the rate of its neighbors corresponding to the same social category is higher than a threshold B_a.If so than the cell remain unchanged otherwise the cell will migrate if an empty place is available. Many criterion may be applied to find a suitable migration place it ranges from just finding the nearest empty place to finding a complete ideal place i.e. where the cell has no more need to migrate.

Example 1 Update the indicated cell according to rates (1) 40% and (2) 75%. If we want to update the cell indicated in the Figure. 1a, we have the following situations :

1. Rate 40% : Here the cell has 4 neighbors in the same social category so in this case the cell will remain unchanged.
2. Rate 70% : Here our cell needs 7 neighbors in the same social category so in this case the cell will migrate.

While studying the dynamic Schelling established that there is a value B_{seg} such that if $B_a < B_{seg}$ then the population will be "uniformly" distributed i.e. the two categories are mixed and if $B_a \geq B_{seg}$ you have a compact block distribution i.e. islands type configurations.

The value B_{seg} is estimated in to be $B_{seg} = \frac{5}{9} \approx 0.55.56\ \%$.

Definition 0.1. Consider the cellular automaton on the alphabet $\{0,1,2\}$ defined by

$$Seg\left(x_{ij}\right) = \begin{cases} x_{ij} \text{ if } \sum_{k=i-1}^{i+1} \sum_{k=i-1}^{i+1} \chi_{\{x_{ij}\}}\left(x_{kl}\right) \geq r.9 \\ 0 \text{ else} \end{cases}$$

where $r \in [0,1]$ is the segre*gation rate and χ is the characteristic function* . Here 0 stands for empty state and 1,2 for social categories 1,2 respectively.

Definition 0.2. Consider the function $Mig : \{0,1,2\}^{\mathbb{Z}} \to \{0,1,2\}^{\mathbb{Z}}$ defined by :

$$Mig\left(x\right)_{i,j} = \begin{cases} if\ Seg\left(x_{ij}\right) \neq 0 \text{ then } \left| \begin{array}{l} r = 1 \\ if\ \exists x_{lm} = 0 \in V_r\left(x_{ij}\right) \text{ then } x_{lm} \leftarrow x_{ij}\ and\ x_{ij} \leftarrow 0 \\ else\ r \leftarrow r+1 \end{array} \right. \\ x_{ij}\ if\ Seg\left(x_{ij}\right) = 0 \end{cases}$$

notice that we are using a local rule with unbounded radius.

In our program we used the following color scheme Green = social group 1 ; White = social group 2, we added another state empty (0 = red square) If after applying the local rule a cell a_{ij} has to migrate then we find an empty cell a_{mn} elsewhere, then a_{mn} takes the value a_{ij} and a_{ij} is switched to empty state.The following figure represents two typical configurations we find in simulations after some iterations and if the rate is higher than 55% the configuration space is successively homogenized (see in Figure. 1b and Figure. 1c).

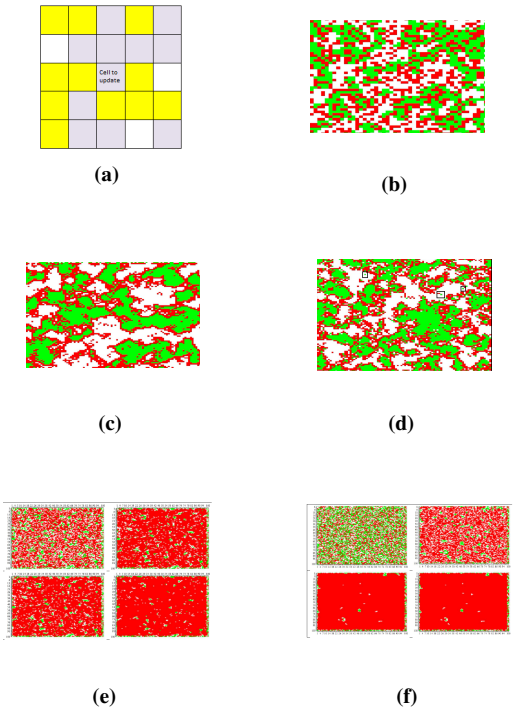
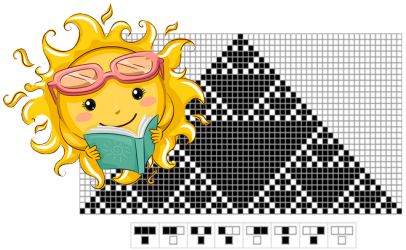


Figure 1: (b) Rate 45% Number of iterations : 40; (c) Rate 80% Number of iterations : 40 ; (d) Black boxes highlights surrounded cells; (e) From upper line from left the 4 first iterations of the model for $S_b = 10$ red pixel $= Dl$, green pixel $= SL$; (f)From upper line from left the 4 first iterations of the model for $S_b = 10$ red pixel = Dl, green pixel = SL;

We considered an asynchronous version of the Schelling model, In this model we consider that each cell decide to migrate depending on time. In our program, we used a random time generator for each cell and a program parameter named "$time$" when the random generator give a value greater than "$time$" then the state cell is updated. During simulations and because of randomness of migration the configurations will not be homogenized as in the synchronous model , some cells may remain unchanged despite begin "surrounded" by islands of different social categories, see the Figure. 1d.

Considering the case of Algeria in many geographical regions there is a social pressure to maintain the amazing language (Kabylie region, Chaouia region and in the desert among Touareg population) the pressure is somewhat less or in existent in other regions. Here we consider a partition of the lattice depending of the geography of the country, on each element of the partition we define a version of the cellular automaton S but with adequate parameters.

Modeling the Spread of Covid-19 with 2-D Cellular Automata



Subrata Paul[†]
Mentor: Kamalika Bhattacharjee[*]

[†]Indian Institute of Engineering Science and Technology, Shibpur, India
[*]National Institute of Technology Tiruchirappalli,Tiruchirappalli, India

August 20, 2022

This project focuses on developing a model that can accurately simulate COVID-19's global dissemination patterns. We utilize cellular automaton (CA) to build such a model. In this project, a new variant of CA called *Temporary Stochastic Cellular Automata* (TSCA) is used, where two rules is being utilized, one of which serves as a default rule and the other of which is a probabilistic rule that is applied with some probability. To consider the mutation of the COVID-19 rule, we employ a set of TSCAs. Each TSCA is considered as $(f, g)[\tau]$ and at a time the applied TSCA is chosen from the set. The model evolves using two TSCA rules f and g, where f is represented as the propagation of the virus, g is represented as recovery function and g is applied with probability τ. The model is validated on the basis of a real-time dataset of spreading Coronavirus (SARS-COVID-19) over the world. This proposed model depicts the spreading scenario of the novel Coronavirus which has caused a global pandemic.

We consider periodic boundary condition, where the first and last row of cells are neighbors to each other, whereas, the first and last column of cells are neighbors to each other. Temporally stochastic cellular automata (TSCA) where at a time step, a cell can be updated using one of the two rules f and g. Here, f is the *virus spreading* rule for the CA, whereas, g is the *immunity* and is applied with some probability. That is, rule g is applied with probability $\tau \in [0, 1]$ whereas the rule f is applied with probability $(1-\tau)$. We call τ as the *rate of immunity*. This way of looking at these rules makes both of them temporally stochastic. Therefore,

$$y = \begin{cases} G_g(x) & \text{with probability } \tau \\ G_f(x) & \text{with probability } 1-\tau \end{cases}$$

where, $G_g(x)|_i = g(s_1, s_2, s_3, \cdots, s_{10})$ and $G_f(x)|_i = f(s_1, s_2, s_3, \cdots, s_{10})$. The proposed system specification is written as $(f, g)[\tau]$. The rate of transmission of the virus varies depending on the time of year, sometimes being quite high and other times being very low. As a result, we have considered a set of TSCAs and each of the TSCAs that we have employed for varying periods of time represents $(f, g)[\tau]$ in our model. We have used a set of TSCA rules, to simulate our model, $(f_1, g_1)[0]$, $(f_1, g_1)[0.3]$, $(f_2, g_2)[0.3]$, $(f_3, g_1)[0.3]$, $(f_3, g_3)[0.3]$.

We assume that, initially, the state of all the cells is considered to be 0 and only one cell is set to state 1. Here, we consider *white cell* for *state 0*, *black cell* for *state 1*, Fig. 1a shows the initial configuration for the model. Fig. 1b, Fig. 1c and Fig. 1d show the intermediate configurations for the simulation which demonstrates several COVID-19 spread scenarios at various time. *white cell* indicates un-infected cells whereas *black cell* indicates infected cells. Fig **??** shows the graph representation of our simulation, where Fig 1e shows the growing of infected cells (switches to *state 1*) with respect to time. Fig 1f depicts the number of infected cells over the grid for different time period.

For the study, we choose COVID-19 spreading data set which are taken from `https://www.kaggle.com/datasets/imdevskp/corona-virus-report`

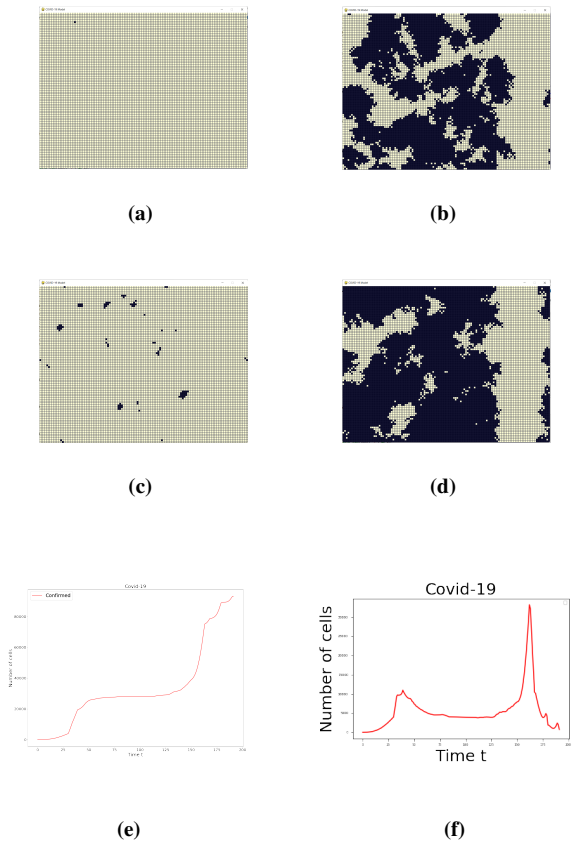
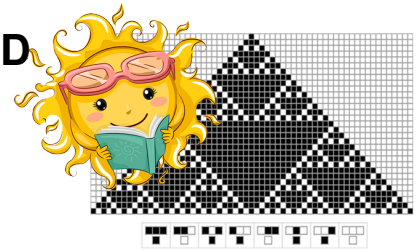


Figure 1: (a) Initial configuration of the CA; (b) Configuration shows the first wave of spreading virus; (d) Configuration shows the rising of second wave; (e) Number of confirmed cases (switches to *state 1*); (f) Shows the wave of spreading the virus;

, we employed three cases: confirmed cases, recovered cases and the wave. The spreading accuracy of the proposed model is compared with real time data sets. We observed that our proposed TSCA model displays the identical circumstance like a real-time scenario and it becomes effective and performs effectively for other circumstances as well.

We may consider this model as a generalized model of spreading of this sort of viruses in future by only changing the set of TSCA rules (tune the parameters f, g, τ to form a TSCA).

Finding the Inverse Cellular Automata of a given 1-D Reversible CA



Khitish Kumar Gadnayak
Mentor: Kamalika Bhattacharjee

August 23, 2022

Reversibility is one of the important characteristic in the domain of cellular automata since it gives the notion of preserving the information during the evolution. This property leads to give wide aspect of applications in many real life scenarios. The main objective of this project work is to analyze the bijectivity property of the reversible CA with the help of state transition diagram (STD). This report also describes the modeling approach by mapping of bits of transition states with the rule minterm (RMT) sequences to obtain the inverse function for the reversible and semi-reversible CAs. The reversibility property of the cellular automata describes the preservation of information during the evolution of the cellular automata. The reversibility property also signifies that the each configuration has an preimage or predecessor. Due to the inherent characteristics the reversible CAs are widely used in pattern generation, cryptography, pseudo-random number generation , language recognition.

In our project work, we consider the nature of transition of states of the reversible cellular automata with the help of state transition diagrams and establish the relationship between the different rules of ECA and their inverse functions. We also map the transition of the states in order to analyze the injective and surjective properties of the reversible cellular automata. Figure 1 describes the state transition diagram and the mapping of states to show the bijective property that the CA with rule number 15 which is a reversible CA for a 4-cell under periodic boundary condition.

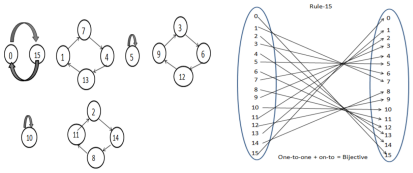


Figure 1: (a)State Transition Diagram (b)Mapping Function

In the next phase of our project work we propose an approach to find the inverse function of the reversible CA for a 4-cell perodic boundary configuration by considering the transition states in a reverse order. In the mapping we consider a window of size three for 3-neighborhood structure for two successive configurations and then we trace the change in the corresponding bit of the window for a map of that changed bit in the RMT sequences.After mapping of all bits in RMT sequences we observe the inverse function rule by finding the decimal equivalent of the RMT bits arranged from left to right with MSB in the left and the LSB in the right. Figure. 2 describes the mapping process for finding the inverse function.

As in the initial phase, we consider the CA under periodic boundary condition for cell size of 4. Then in successive cases we investigate the inverse rule finding approach for cell sizes of $1, 2, 3, 5, 6$ and 7 respectively. Therfore we propose a generalized approach for the finding of inverse rule for reversible CA of length n.

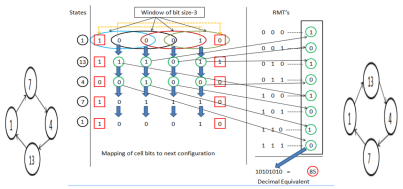


Figure 2: Rule mapping for inverse function(Rule-15)

We propose a generalized appraoch for find the inverse rule for a reversible CA for length of size n. From the observations and work analysis of all the ECA rules, the reversible CAs of length n under periodic boundary conditions are listed in Table. 1.

Table 1: List of Reversible ECAs

Reversible ECAs		
Rule No.	Inverse Function Rule	Relationship
15	85	Both rules are inverse of each other
170	240	Both rules are inverse of each other
51	51	Inverse functioned rule of itself
204	204	Inverse functioned rule of itself

The previous paragraph describes the finding of inverse rule for reversible CA for ECA rules of cell size n under null boundary conditions and in this section we extend our analysis further to

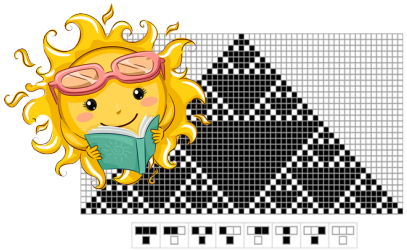
find the inverse rule for the semi-reversible CA.

In our analysis, we consider different non-trivial semi-reversible CA rules like rule-45, rule 154 and rule 105. The non-trivial rule 45 and rule-154 are reversible for odd values of n that means, for $n = 1, 3, 5, 7, \cdots$ where as rule-105 is a reversible CA for n values like $n = 1, 2, 4, 5, 7, 8, \cdots$ and $n \neq 3k, \forall k \in N$. We try to consider the same mapping processes discussed in previous section to

nd the inverse rule for the above mentioned semi-reversible cases. From the work analysis we observe some salient features those are listed below:

- The algorithmic approach for reversible CA is true for semi-reversible CA for $n = 1$ and $n = 3$ and the inverse rule is rule-101 with a three neighborhood mapping process.
- imilarly for rule-154, the inverse rule mapping approach is true for $n = 1$ and $n = 3$ and the inverse rule is rule-180 with a three neighborhood mapping process.

ISOMORPHISM IN CELLULAR AUTOMATA



Vicky Vikrant[†]
Mentor: Sukanya Mukherjee[*]

[†]National Institute of Technology, Tiruchirappalli, India
[*]Institute of Engineering & Management, Kolkata, India

August 22, 2022

This work focuses on the isomorphism of cellular automata (CAs). Two cellular automata are said to be isomorphic if their configurations evolve in the similar way. As a model, here we use non-uniform elementary cellular automata under null boundary and discover few inherent properties of CAs to decide whether the given cellular automata are isomorphic.

Cellular automata are discrete dynamical systems which produce complex global behaviour using simple local computation. The configurations of a cellular automaton (CA) evolves with time. A configuration is said to be *reachable* if it has some predecessor configuration; otherwise, the configuration is *non-reachable*. Let G be a CA and C be the configuration space. Let $G(x) = y$ and $G(z) \neq x$ where y is reachable configuration and y is reachable from x; Now, c is not reachable from any configuration z. Therefore, x is non-reachable. In finite CA, every configuration ultimately reaches to some *cycle*. The configuration which is used to form a cycle, is called a *cyclic* configuration. The length of a cycle is determined by the number of cyclic configurations it possesses. The configuration which is not cyclic, is called as *acyclic*. The cycle structure of a CA is the collection of the number of cycles along with their lengths. If all the configurations of a CA are cyclic, then such CA is *reversible*; otherwise, the CA is *irreversible*.

Let G_1 and G_2 be two ECAs of same size having same configuration space C. G_1 and G_2 and are said to be *isomorphic* if there exists a bijective mapping $\pi : C \to C$ such that $G_1(x) = y$ iff $G_2(\pi(x)) = \pi(y)$ where $\forall x, y \in C$. It is very challenging to figure out π and not much work has been found on the isomorphism in cellular automata. Thus we are motivated to find some intrinsic properties of cellular automata which play the instrumental role to decide of isomorphism in CAs. In our work, we use *reachability tree*, which is a rooted and edge-labelled binary tree that decides the reachable and non-reachable configurations of CA. This tool is used to develop some properties on the isomorphism in cellular automata.

Property1: Two CAs G_1 and G_2 are said to be *isomorphic* if both be reversible or both be irreversible but the converse is not always true.

If **Property1** is not satisfied by the given G_1 and G_2, then they are not isomorphic. Therefore, we test the reversibility of the given non-uniform CAs (using the algorithm of $O(n)$) and if we find that G_1 is reversible CA and G_2 is irreversible CA, then they are not isomorphic. Let $(10, 150, 90, 20)$ and $(2, 150, 90, 20)$ are given CAs. Here, $(10, 150, 90, 20)$ is reversible CA but $(2, 150, 90, 20)$ is irreversible CA as 2 can not be the first rule of any reversible CA. Therefore, Let $(10, 150, 90, 20)$ and $(2, 150, 90, 20)$ are not isomorphic.

As **Property1** is a necessary condition to decide isomorphism in cellular automata, we need to figure out some more properties on CAs when G_1 and G_2 both be reversible or both be irreversible.

Now, in reversible CAs, as all configurations are cyclic, for deciding isomorphism, we need to check whether they have same number of cycles along with same lengths.

Property2: Let G_1 and G_2 be reversible cellular automata. They are said to be isomorphic iff both the CAs have the same cycle structure.

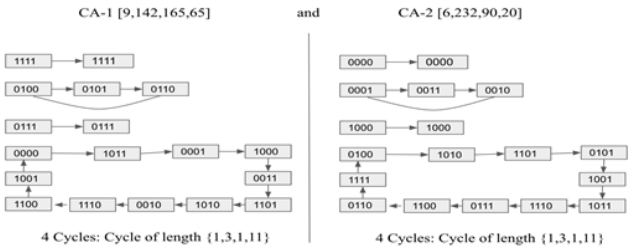


Figure 1: Reversible CAs

Here, $(9, 142, 165, 65)$ and $(6, 232, 90, 20)$ are isomorphic CAs as they have same cycle structure $[2(1), 1(3), 1(11)]$. Though few works have been reported on the computing of cycle structure of reversible CAs but its inherent hardness still makes it as a challenging problem. So, we are motivated to figure out some properties of isomorphic reversible CAs.

Next we focus on irreversible CAs. To study the isomorphism in irreversible CAs, other than cycle structures of those CAs, the count of acyclic configurations play an instrumental role.

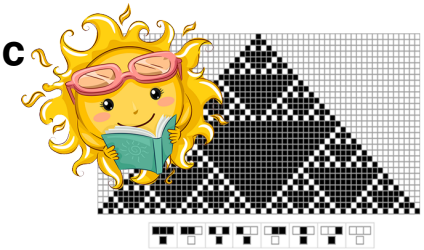
Property3: Let G be a CA of size n. In the reachability tree of that CA, if k^i denote the number of non-reachable edge(s) at the level i of that tree, then the total number of non-reachable configurations(s) of that CA is $\sum_{i=0}^{n-1} k^i \times 2^{n-1-i}$.

Property4: Let G_1 and G_2 be irreversible cellular automata. They are said to be **isomorphic** if both the CAs have the same number of non-reachable configurations but the converse is not always true.

If **Property4** is not satisfied by the given G_1 and G_2, then they are not isomorphic. To check the count of non-reachable configurations, we should use the reachability tree as a tool.

Here, $(1, 135, 92, 5)$ and $(10, 60, 86, 20)$ are not isomorphic cellular automata as they have total number of non reachable configurations 3 and 4 respectively. Next, we need to deduce some more properties of CAs for determining isomorphism when they have sane number of non-reachable configurations.

Multi-seed Image Segmentation for the Microscopic Image



Tanisha Ayach[†], Suchitra Behera[†], Sagarika Padhi[†]
Mentor: Nazma Naskar[⁕]

[†]Institute of Mathematics & Applications, Bhubaneswar, India
[⁕]Kalinga Institute of Industrial Technology, Odisha, India

August 24, 2022

Image segmentation is one of the most important image processing techniques for biological images. Till now, various Cellular Automata(CA) rules have been used for segmentation. For updation of states of the cell, conditional rules using Von Neumann neighbourhood and Moore neighbourhood were used. In this work, instead of taking one seed point per image, multiple seed points are used for segmentation. A dataset of microscopic images has been taken and various segementation rules have been used for analysis.

A significant number of diagnostic procedures in modern clinical practise, as well as medical and biological research, now depend on the whole range of medical imaging technology. The manual procedure of screening microscopic slides requires subjectivity. Finding the boundaries of cells, cell nuclei, and histological structure in images of stained tissue with sufficient accuracy is referred to as segmentation in microscopic images. The goal of segmentation is to make an image representation simpler while also making it more significant and easier to examine. The regions must be relevant to shown objects or features of interest in order to be significant and valuable for microscopic image analysis and interpretation. The difficulty in accurately segmenting each cell, as well as the wide variability in the features of each component, make this a difficult job. Here in this proposed work, an existing dataset has been used for analysing multi-seed segmentation where the updation rule considers Von Neumann and extended Moore neighbourhood. In order for segmentation to be more prominent depending on the quality of the input images, thresholding plays a crucial role in image segmentation. Here, the updation rule using Von Neumann neighbourhood is based upon thresholding, where the same using Moore neighbourhood relies on maximum and minimum state. Results have been analysed from the images, segmented using these rules.

Segmentation simply refers to breaking up an image into various items or regions. The region of interest is labelled as the foreground in the segmentation, and the remaining portions of the image are labelled as the background. It takes a lot of time and effort to manually segment. The work is typically accomplished by marking the object of interest. However, this approach does not always produce correct results. We require an automatic image segmentation technique that produces precise results with minimal user input. Cellular automata are utilised in this automated procedure. An image is made up of pixels that can be thought of as cells, which is the basic idea behind employing cellular automata for automatic segmentation. The segmentation of an image is based on the image's characteristics, such as intensity and other properties that are derived from intensity.The classification of a pixel as foreground or background also depends on the neighbouring pixels. This is because the segmentation of the image also depends on the value of the intensities of the pixels in the pixel's neighbourhood. Edge detection and thresholding are the fundamental concerns in image segmentation. Various threshold values may result in segmented images with varying levels of clarity.

Wongthanavasu and Sadananda suggested a method based on a conditional rule for updating the state of the states. This rule can be represented as:

$$v_{c+} = \begin{cases} 0 & \text{if } \ v_c \leq v_{max} - v_{min} \\ v_{max} - v_{min}, & \text{otherwise} \end{cases}$$

In order to approach an edge detection strategy, A. Popovici and D.Popovici took into account the different state differences between the neighbouring pixels in accordance with the Von-Neumann idea and the central pixel. Here, all absolute state differences have been considered in the comparison. The central pixel's state will be 0 if the differences are more than the given threshold. Otherwise, it remains unchanged.This rule can be expressed as follows:

$$v_{c+} = \begin{cases} 0 & \text{if } \ |v_i - v_c| \leq \epsilon \\ v_c, & \text{otherwise} \end{cases}$$

The region of interest can be segmented more successfully by using cellular automata-based segmentation approaches. By using various transition rules, characteristics from various imaging modalities can be retrieved that could be useful in biological images.

Applying these above updation rules to multiple seeds,it is found that, the updation rule using Von Neumann neighbourhood often gives better result than the Moore neighbourhood e.g.,

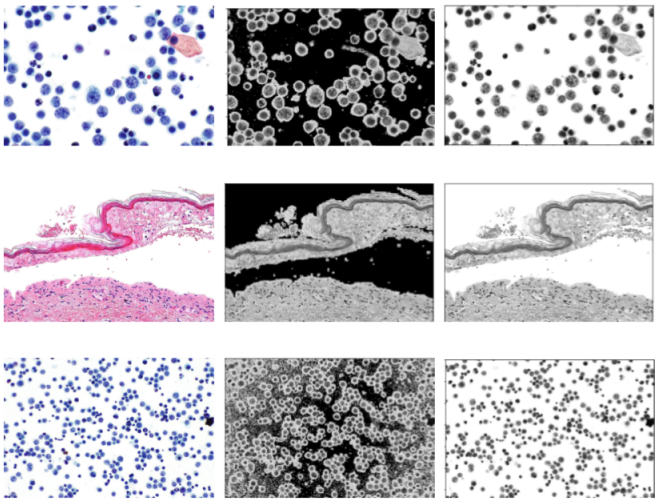
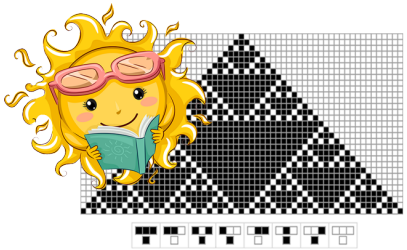


Figure 1: Original image, segmentation using Von Neumann CA & Moore neighbourhood CA

Hexagonal Cellular Automata Simulation of Intergranular Cracking in Polycrystalline Materials



Tarun Kumaar M K, Deeraj H and Suvetha M
Mentor: Prince Gideon Kubendran Amos

September 12, 2022

In our everyday life, we can find a diverse range of materials that each have their own characteristic properties that decide which applications they can be employed in. Thus, we often group materials by their shared characteristics into groups such as metals, fabrics, glasses and so on. The properties of a material are closely correlated with its structure on the microscopic scale. Hence, it is very important to study and understand how the microstructure of a given material affects its response to certain conditions and thus, affects its characteristic properties.

Materials are composed of atoms or molecules and how these elements are arranged in three-dimensional space affects their interactions and their properties as well. When there is a regular ordered arrangement of atoms over a long distance, that material can be called crystalline and when the ordered arrangement of atoms is observed over a much shorter range, the material is deemed amorphous. Crystalline structures often arise when the material in consideration undergoes solidification from liquid to solid state. As the material is cooled, small nuclei that each have a particular orientation of crystalline structure expand to form large grains. The boundary separating two adjacent grains of different orientation is called the grain boundary.

Intergranular cracking occurs when a crack propagates along the grain boundaries of a boundary, usually when these boundaries are weakened. This can be compared to a wall of bricks where cracking takes place in the mortar that joins these bricks together. Intergranular cracking is likely to occur if there is a hostile environmental influence and is favoured by larger grain sizes and higher stresses. Though there are several mechanisms for intergranular fracture, all of them revolve around grain boundary orientation and the presence of solutes and impurities.

Our model takes an input of the desired number of nuclei, which are then distributed at random throughout the domain. For this system, we are considering 18 different grain orientations associated with their own unique colours. Conventionally, cellular automata are associated with square discretization but here we have implemented hexagonal discretization and attempt to undertake a qualitative comparison between them.

By applying a specific rule to evolve the automata synchronously, we can observe the nuclei that were introduced at the start of the evolution grow into large grains and develop grain boundaries. The rule implemented here is such that, the cell under consideration will also solidify or become alive if there is at least one live cell in the neighbourhood or radius 1. The use of hexagonal cells gives us a unique grain boundary shape compared to square cells that may appear more jagged upon close observation. By testing different neighbourhoods and rules, we were able to identify a few combinations that resulted in a satisfactory model for a polycrystalline system. One important thing to note is that once cells become “alive” or nucleated they should not be able to return to their *dead* state. After obtaining this polycrystalline structure, we are able to apply a rule to model the crack as it propagates along the grain boundary between two or more grains. This rule is capable of deciding between two paths where a junction between three or more grains is formed in the domain. Thus, the model is able to create a polycrystalline structure which can then be used as a sample to effectively model the growth of a crack across the domain as shown below.

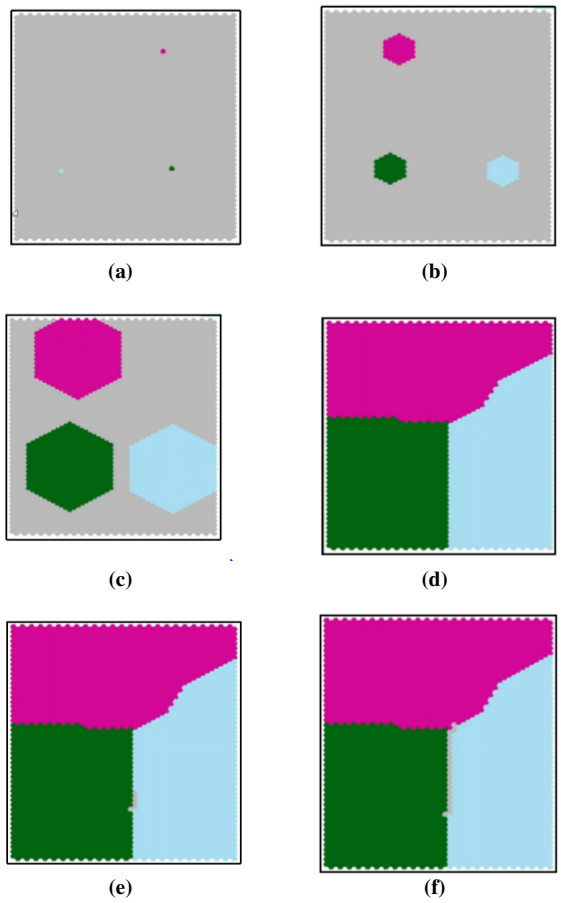
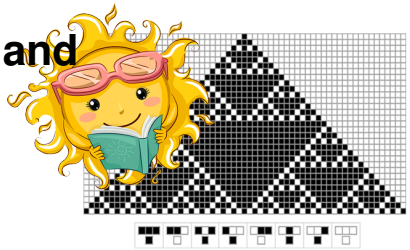


Figure 1: (a), (b), (c), (c) Three Nuclei developing into Polycrystalline System; (e), (f) Intergranular Crack Initiation and Propag ation along the green and blue boundarie;

It is also worthwhile to note that this model is not limited to materials science related applications. As the model is able to select a path at a junction of two or more possible options, it may be used as a path finding algorithm. If one were to consider the grains to be mountains with the centre of each grain being its peak, and the grain boundaries the narrow valleys at ground level; the model can be construed as a path finding algorithm that can be applied in mountainous terrain. We hope to expand this project to also model intergranular crack propagation within the grains and search for other potential applications outside the domain of materials science.

Reversibility in α-asynchronous Cellular Automata and Cryptography

Sourasish Das, Mainak Shaw, Aravind P
Mentor : Souvik Roy



August 24, 2022

Reversible computation has garnered a lot of attention over the years,Though studied extensively in a great variety of synchronous computation models, it is virtually unexplored in an asynchronous framework. While discussing asynchronous frameworks, alpha asynchronous cellular automata is a niche topic in the discussion of reversibility. Alpha Asynchronous cellular automata update their cells according to the value of alpha, since most of the updation process is based on probability, the experiments to be done to decide on the reversibility was done over a range of CA sizes. Further 88 rules were identified which depicted the characteristics of the 255 rules , and a sample of 10 probabilities were taken and experiments were conducted for all initial configurations for a given CA size and given value of alpha.

Definition 0.1. An elementary cellular automaton is a one-dimensional cellular automaton where there are two possible states (labeled 0 and 1) and the rule to determine the state of a cell in the next generation depends only on the current state of the cell and its two immediate neighbors.

Definition 0.2. A reversible cellular automaton is a cellular automaton in which every configuration has a unique predecessor.

Definition 0.3. An α-asynchronous cellular automaton is able to update individual cells independently, depending on the probability α

The set of 255 rules were reduced down to 88 characteristic rules, and a sample set of 10 probabilities were taken and for a range of CA sizes all initial configurations were checked for reversibility.While classifying rules each of the configurations were converted into their decimal representation and were plotted as a network of states and for the theoretical results a list of all the possible configurations for a given initial configuration and if the configuration list contains the initial configuration more than once then it can be declared as being reversible for that configuration , for a given alpha value and given CA size. **Probability Values :** 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9

Rules : 0 , 1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15 ,18 ,19 ,22 ,23 ,24 ,25 ,26 ,27 ,28 ,29 ,30 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,40 ,41 ,42 ,43 ,44 ,45 ,46 ,50 ,51 ,54 ,56 ,57 ,58 ,60 ,72 ,73 ,74 ,76 ,77 ,78 ,90 ,104 ,105 ,106 ,108 ,128 ,129 ,130 ,131 ,132 ,133 ,134 ,136 ,137 ,138 ,140 ,142 ,146 ,150 ,152 ,154 ,156 ,160 ,161 ,162 ,164 ,168 ,170 ,172 ,178 ,184 ,200 ,204 ,232

Sizes :3 ,4 ,5 ,6 ,7 ,8 ,9 ,10

For a given CA size, a given rule the classification is as follows:

Reversible:

- For all alpha values
- If all configurations are reversible then it is Reversible

Irreversible:

- For all alpha values
- If some configuration is irreversible then it is Irreversible

Partially Reversible:

- For some alpha values
- If all configurations are reversible then it is partially reversible

Now generalizing this classification over all values of alpha and CA sizes. The rules can be classified as:

Reversible:

- For all alpha values
- If all configurations are reversible then it is Reversible
- Example: 204

Irreversible:

- For all alpha values
- If all configurations are irreversible then it is irreversible
- Example: 0,2,4,6,8

Cell Size-Partially Reversible:

- For some alpha values
- If all configurations are reversible then it is Cell Size-Partially Reversible
- Example: 1,3,7,9,11

Cell Size-Reversible:

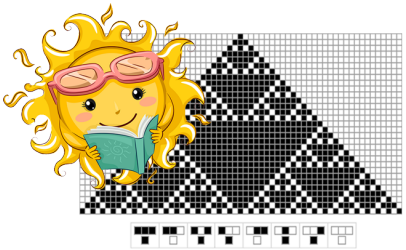
- For some CA sizes
- For all alpha values
- If all configurations are reversible then it is Cell Size-Reversible
- Example: 19,27,33

Table 1: Observations

Irreversible	0,2,4,5,6,7,8,10,12,13,14,15, 18,24,26,29,32,34,36,38, 40, 42,44,50,72,77,78,104,106,128, 130,132,133,136,138,150,152,154, 160,162,168,170,178,184,232
Reversible	204
Cell Size-Reversible	19,27,33,35,37,41,43, 45,51,57,105,142
Cell Size-Partially-Reversible	1,3,7,9,11,22,23,25,28,30,46,54, 58,60,73,74,76,90,10 8,129, 131,134,137,140,146,156,161,164, 172,200

All 88 rules that are representative of the 256 rules were classified and the only completely reversible rule is Rule Number 204. Other rules are dependent on the CA size and alpha values. The ratio of completely irreversible rules to reversible rules(partially, CA size reversible and completely reversible) are 1:1 as there are 43 rules that are reversible to an extent but the other 45 are completely irreversible. In conclusion except 1 rule (204) all other rules are either partially reversible / irreversible.

Characterization of 2D Linear Cellular Automata by using Linear Algebra



Tina Das
Mentor: Kamalika Bhattacharjee

August 30, 2022

In this project we studied the behavior of $2D$ linear Automaton generated by linear rule under the null boundary condition over the field $\mathbb{Z}^2$ transition matrix or rule matrix of this cellular Automata by represent each cell of block into linear form $1, 0$ according to dependency and independency of each cell respectively determined by applied rule .viz,for 2×2 block there is 4 cell, we identified each cell by the number $1, 2, 3, 4$ and for each cell like cell number 1,we put 1,0 if it is dependent or independent to particular that cell in appropriate order respectively, corresponding to each cell we get one representation and ultimately we obtain a 4×4 matrix, for 3×3 matrix we obtain 9×9 order matrix. We are characterize this Automata by using matrix rule and able to Identify Reversibility of this Automata by calculating the Determinant of the Rule matrix. Lastly, we checked Reversibility of 2×2 block and 3×3 block for all 32 rules over $\mathbb{Z}^2$ field.

Table 1: Results for 2×2 Block

Rule	Reversible/Irreversible
Rule 0	Irreversible
Rule 1	Reversible
Rule 2	Irreversible
Rule 3	Irreversible
Rule 4	Irreversible
Rule 5	Irreversible
Rule 6.	Irreversible
Rule 7.	Reversible
Rule 8.	Irreversible
Rule 9.	Reversible
Rule 10.	Irreversible
Rule 11.	Reversible
Rule 12.	Reversible
Rule 13.	Irreversible
Rule 14.	Reversible
Rule 15.	Reversible
Rule 16.	Irreversible
Rule 17.	Reversible
Rule 18.	Reversible
Rule 19.	Reversible
Rule 20.	Irreversible
Rule 21.	Reversible
Rule 22.	Irreversible
Rule 23.	Irreversible
Rule 24.	Irreversible
Rule 25.	Irreversible
Rule 26.	Irreversible
Rule 27.	Irreversible
Rule 28.	Reversible
Rule 29.	Reversible
Rule 30.	Reversible
Rule 31.	Reversible.

Working procedure: Rule 31 :This is an example of a Reversible Automata

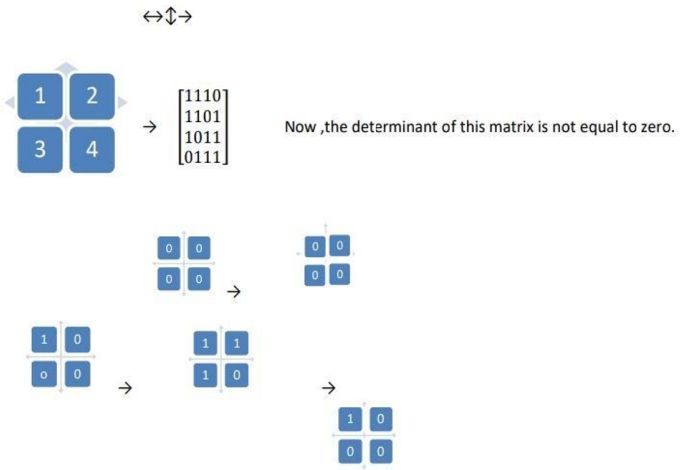
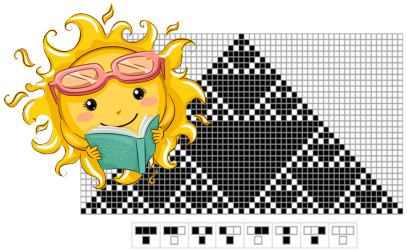


Figure 1: Working procedure of Rule 31; This is an example of a Reversible Automata ;

UNDERSTANDING THE CONVERGENCE IN ALPHA-ASYNCHRONOUS CELLULAR AUTOMATA



Bheemakonda Raja Maheswar† and Revanth D. Rampal†
Mentor: Souvik Roy*

†RGUKT-AP, RK Valley, Andhra Pradesh, India
*Indian Institute of Engineering Science and Technology, Shibpur, India

August 29, 2022

This paper contains the keen observations and research on factors of convergence, converging rules and attractors of Alpha-Asynchronous Cellular automata. Our experimentation will be on elementary cellular automata from alpha values 0.1 to 1.0 for a set of 88 minimal rules. In this process our main focus is to track converging rules and record them. To get this information we developed a program that takes alpha values and rules as input and returns above mentioned information as output. Our second focus will be on recording the attractors. In this project we consider 2-state (0 or 1) 2-neighborhood one Dimensional Cellular automata. In which cell state transition takes place with help of $0 - 255$ Binary rules. In simple words an initial configuration is set and some rule is applied from the given rules to it to reproduce further configurations and self replication. For every possible neighborhood there will be a respective state assigned to it. According to this evolution takes place in configurations.

Generally in Cellular Automata cell updation or state transition takes place synchronously considering time as discrete, but in case of asynchronous Cellular Automata the cell updation takes place asynchronously irrespective of time. Convergence is a phenomenon of where a system cannot come back to its initial configuration after moving out of the configuration. Rather the system, in course of its evolution, settles down to a configuration (convergence point) or to a small set of configurations. A range of physical systems show this convergence phenomenon. During the evolution of α-asynchronous cellular automata at one particular stage all configurations converge to a fixed point (fixed configuration). We call this fixed point. Our main interest is on type 4 (see in Figure. 1),

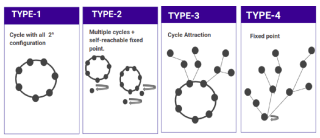


Figure 1: 4 types of evolutions observed in α- asynchronous CA

to recognize all of the converging rules under different alpha values and we record them. We developed a program which generates output that consists of CA sizes that shows convergence for every alpha value of a set of rules under periodic boundary conditions. There are 256 rules in ECA but some of the rules show the evolution of the same behavior. So we take rules of unique behavior and avoid the similar types. The program takes rules as input. And outputs two text files. First File consists of information about for every rule under all alpha values whether the rule shows convergence or not. Second File consists of all attractors with respect to rule only if that rule Shows convergence. From the information obtained by the first File we plot a table that shows convergence properties of the given 88 rules under all alpha values i.e. from 0.1 to 1.0 with CA sizes 4 to 10. From the Figure. 2, the information we took the rules as convergible if and only if they are totally covergible under all given conditions. To show an example of table, In Figure. 2 rule 0 characteristics

Rule Number	Alpha	CA Size 4	CA Size 5	CA Size 6	CA Size 7	CA Size 8	CA Size 9	CA Size 10
RULE 0	0.1	✓	✓	✓	✓	✓	✓	✓
	0.2	✓	✓	✓	✓	✓	✓	✓
	0.3	✓	✓	✓	✓	✓	✓	✓
	0.4	✓	✓	✓	✓	✓	✓	✓
	0.5	✓	✓	✓	✓	✓	✓	✓
	0.6	✓	✓	✓	✓	✓	✓	✓
	0.7	✓	✓	✓	✓	✓	✓	✓
	0.8	✓	✓	✓	✓	✓	✓	✓
	0.9	✓	✓	✓	✓	✓	✓	✓
	1.0	✓	✓	✓	✓	✓	✓	✓

Figure 2: Rule-0 satisfying all the given condition.

are shown under every condition i.e., Alpha value and CA size the rule shows convergence.(converged represents tick mark, first tick tells for alpha 0.1 and CA size 4 the evolution is converged). Because of this we finalize that Rule 0 shows convergence. After performing multiple tests and program executions we found the following rules that show convergence in α -Asynchronous Cellular Automata.The list of rules (47 rules) show complete convergence- 0, 2, 4, 8, 10, 12, 22, 24, 32, 36, 38, 40, 42, 44, 54, 56, 72, 74, 76, 78, 90, 104, 106, 108, 128, 130, 132, 136, 138, 140, 142, 146, 150, 152, 154, 156, 160, 162, 164, 168, 170, 172, 178, 184, 200, 204, 232. Attractors of a rule cannot be recorded because we use a *randomizer function* to the cells of eca, because of this every time we execute the program, the randomizer function may produce 0th or originally existing state. Due to this phenomena, the evolution of cellular automata is different for every cycle of execution.

If the random value generated by the randomizer function is less than alpha value for a cell in Lattice then that cell state is set to zero in Alpha Asynchronous CA. (Remaining cells follow the transition function in other words, RULE).

$$Random(0.1\ to\ 1.0) < \alpha \tag{1}$$

Here we are taking an example of CA size 12 assuming some alpha value. When we execute the program 2 times, we look two scenario (see in Figure. 3).

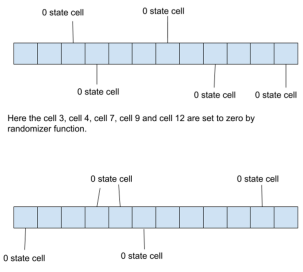


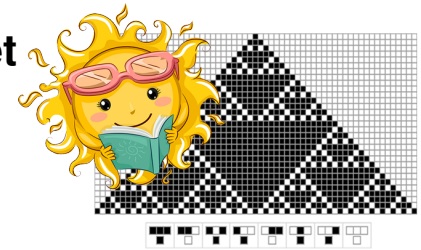
Figure 3: Execution of the program for 2 times

In second case (see Figure 3, Cell 1, Cell 4, Cell 5, Cell 6, Cell 11 are set to state 0. Due to this phenomena it is impossible to get the same attractors every time. So with this reason we are unable to record attractors.

Cellular Automata based Modelling of Stock Market

Krishnendu Dasgupta
Mentor: Souvik Roy

August 23, 2022



In this work, it has been tried to showcase the Dynamics of the Stock Market with the help of a 2D Cellular Automata. The Active Traders are characterised by the states +1(Buying Stocks) and -1(Selling Stocks). The Inactive Traders are characterised by 0 State. Some Simulation Rules(For changing the State of the Traders from Active to Inactive and Vice Versa) are applied and the simulation is done. Based on that Simulation, some Graphs are plotted. After that, Simulation rules are Tweaked. Instead of checking the Neighbours and keeping the condition on the basis of at least 1 Neighbour in the Simulation Rules, K and l Neighbours are respectively checked for the Rule(1) and Rule(2) of Simulation where K and l are taken from the Set 1,2,3,4. Graphs are obtained and along with that some Drastic Changes are also observed in the Dynamics of the Market. We witness the Strictly Increasing Graphs of the Simulation to become sometimes Strictly Decreasing when the K and l values are Increased. In the later part of the analysis, we take The Global Neighbourhood Condition for the Simulation to resemble the Real Life Stock Market. These Neighbours are Fully Random in nature. Since, we are considering 512×128 Grid, so we have 65,536 cells and out of those cells, these Global Neighbours are Randomly Chosen and preferably they don't collide with the Local Neighbours. There we observe some Interesting Changes in the Dynamics of the Market. The Graphs remain Strictly Increasing for a very long extent of time as compared to the Most Initial Model. A 512×128 Grid is taken where each cell can have 3 states, precisely, 0,+1 and -1. 0 indicates an Inactive Trader who is not involved in the Market; $+1$ indicates an Active Trader who is buying some stocks; -1 indicates an Active Trader who is selling some stocks.

Now, initially a Random Configuration is taken and then the simulation is continued for generations to get some Interesting Observations. The Initial Random Configuration guarantees to have not more than 27% of Active Traders. Von Neumann Neighbourhood is considered for the simulation. The rule is as follow,

1. If a Cell is in State 0 and at least 1 of its Neighbours is in State 1, then with Probability PH, it gets converted to State (+1 or -1).
2. If a Cell is in State 1 and at least 1 of its Neighbours is in State 0, then with Probability PD, it gets converted to State 0.
3. If a Cell is in State 0, then with Pc Probability, it gets converted to State (+1 or -1).

In Figure. 1, Black Cells represent Inactive Traders and Non-Black Cells represent Active Traders. As per the Instructions and values given in the Paper, we do the fine Tuning of the Value of P_H. We take Fixed values for $P_C = 0.0001$ and $P_D = 0.05$. Then we take 5 Different Values of P_H as $[0.0493, 0.0490, 0.0488, 0.0485, 0.0475]$. Then we simulate using these values and plot the Graph between Number of Active Traders(Y-Axis) vs Number of Generations(X-Axis) (see in Figure. 1c). The Graph obtained is Strictly Increasing and finally settles and becomes almost constant within the range of $35,000 - 40,000$.

In the Previous Main Model, as per the Given Conditions (1) and (2), only 1 Opposite Neighbour can make the cell toggle from its state to another on a certain Probability. But, if we changed that quantity of 1 to K and l maybe respectively for the Rules, then a very Interesting Result comes in Picture. Where $K, l \in 1, 2, 3, 4$

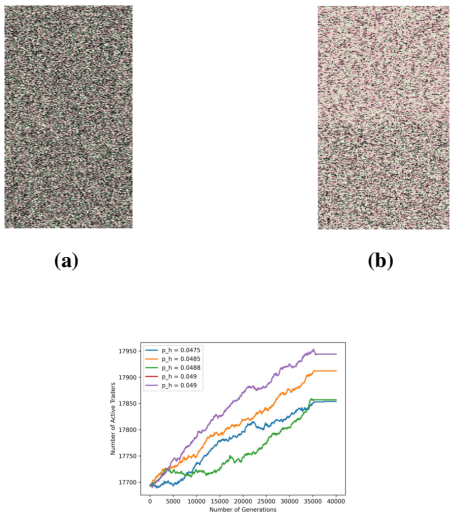


Figure 1: Diagram for $P_D = 0.5, P_H = 0.6, P_H = 0.3$ and simulated for 1,00,000 Generations. (a) Initial State; (b) Final state; (c) The Graph;

Revised Rule(1): If a Cell is in State 0 and at least K of its Neighbours is in State 1, then with Probability P_H, it gets converted to State (+1 or -1).

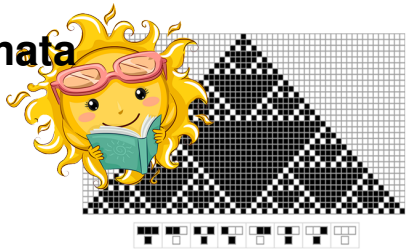
Revised Rule(2): If a Cell is in State 1 and at least l of its Neighbours is in State 0, then with Probability P_D, it gets converted to State 0.

Till now, we have seen the Neighbours to be Local Von Neumann Neighbours to be considered for Simulation. But in this second version of the Extended Model, Global Neighbours have been taken into picture so as to get some Insights about the Effect of the Market as a whole to a Particular Cell(A Trader to be precise). But in Real Life, the Stock Market's Traders are not only dependent on their Local Neighbours, for being converted from Active to Inactive and vice versa, but also somehow dependent on the Global Neighbours. Since, we are considering 512×128 Grid, so we have $65,536$ cells and out of those cells, these Global Neighbours are Randomly Chosen and preferably they don't collide with the Local Neighbours. Following are the two cases Considered with respect to Global Neighbourhood Condition:

- 2 Local Neighbours(Randomly chosen from 4 Von Neumann Neighbours) and 2 Global Neighbours(Randomly chosen from the whole Board).
- 3 Local Neighbours(Randomly chosen from 4 Von Neumann Neighbours) and 1 Global Neighbour(Randomly chosen from the whole Board).

Convergence in 3-state Non-uniform Cellular Automata and Pattern Classification

Aryan Pareek
Mentor : Nazma Naskar



August 24, 2022

1-dimensional three-neighborhood binary irreversible cellular automata (also called elementary reversible cellular automata or ECA) is a well explored field both with respect to theory and applications. The irreversible behavior of such CA, convergence property for point attractors has been studied in detail for null boundary CA, as well as periodic boundary conditions. The feature set of this cellular automata consists of only binary numbers 0, 1. As a result, we may believe that this may lead to some restrictions of such a pattern classifier in its ability, accuracy, capacity and range of classification. Also, when compared to the real world data, a 2-feature CA really falls short as most data has the number of features spread across the discrete number space.

To solve this problem and test out the pattern behavior, classification and dynamics of the d-state cellular automata, we wanted to first use 3-state 1-dimensional CA (uniform and non-uniform) for pattern classification.

Since there was no pre-existing tool for studying the dynamics of 3-state cellular automata, finding the number of point attractors for each rule and the rule space being too large and unfiltered to work with, we devised a 2-step methodology to study the dynamics of the CA as well as reduce the rule space to some extent to reduce complexity.

1. We derived a mathematical formula to subgroup d-state CA into d^{n+1} such groups based on some conditions, which we proved correct as well as extended along the discrete number space for any d-state CA.
2. We then used the rule numbers from within the selected groups that would be a good fit for our classifier by studying the dynamics of the 3-state CA rules using a self-designed algorithm. We only select those rules for classification that give only point attractors as reversible configuration can give misleading results for our classifier.

As future scope for our research, we wish to extend our work and implement a pattern classifier using these selected rules from the reduced rule space. We also plan to find more rules in lower number groups that may be suitable for implementing pattern classifiers.