CS-2349 Theory of Computation: Final Project Report

Shweta Prasad, Sehajpreet Kaur

Department of Computer Science, Ashoka University

1 Project Goals

We have 2 main goals for this project as given in Project Proposal:

- Simulation of all possible one dimensional cellular automata with 2 possible values ($k = 2$) and range 1 ($r = 1$) as in (1), *Statistical mechanics of cellular automata* and here: Tables of Cellular Automaton Properties and a detailed study of the rules that show interesting behaviour.

- Understanding the application of the density classification problem in (2), *Evidence for complex, collective dynamics and emergent, distributed computation in plants*

2 Introduction

Cellular automata are austere machines capable of showing complex emergent behavior based on small range interactions defined by local rules which may or may not be the same in all localities in a given space. This set up provides us with a way to model emergent behavior in nature, from natural pattern formation to computation in the absence of interconnecting entities such as a network of neurons. By virtue of its capability to do highly parallelized computation, it provides simple yet robust models to uncover the underlying rules of complex emergent phenomena, such as social network dynamics, collective behavior in animals, phantom vehicular congestion on roads, reaction-diffusion based pattern formation, and so on. One can use cellular automata to do parallel formal language recognition and parallel arithmetic, and based on defined rules of interaction between neighbors in both space and time, study the evolution of its behavior.

2.1 Cellular Automata

A cellular automaton is a model of a system of “cell” objects with the following characteristics:

- The cells live on a grid

- Each cell assumes one of finitely many states.

- Each cell has a defined neighbourhood. A cell’s neighbourhood consists of the cells that can affect its state in the next step of evolution. It is usually defined in terms of radius/range (r) or diameter.

- The rule of evolution, defined for a cell and its neighbourhood.

One can define a cellular automaton of any number of dimensions for the grid, any number of finite states per cell and define any kind of neighbourhood, wherein only the cells constituting the neighbourhood can influence the cell of interest’s state in the next time step of evolution. We describe a subset of automata – called elementary cellular automata – in more detail in the next sub-section.

2.2 Elementary Cellular Automata

Elementary cellular automata are one dimensional grids with only two possible states per cell: 1 or 0. One can think of it as bit strings where each bit occupies a cell in an array. Each cell has a radius/range $r = 2$ or equivalently, a diameter of $d = 3$ neighbourhood. This means that to determine the state of a cell in position p at time step $t + 1$, we will look at the states of cells in position $p - 1$, p, and $p + 1$ at time step t.

Since we are dealing with a $r = 2$ neighbourhood, the state of each cell in the automaton array depends on its two immediate neighbours, in addition to its own state; Thus, in order to obtain the configuration of a given cell in the state array at time step $t + 1$, the triple centered at the cell of interest (at the current time step t) is fed into the transition function. Since each cell in the array can only be in one of two states, we have $2^3 = 8$ possible configurations of the triple.

At a given time step, the transition rule is applied to every cell in the array (by feeding into the function the triple centred at that cell) to get the whole automaton’s configuration at the next time step. To represent the rules (used interchangeably with the transition function), we can make use of $8-$bit strings, the index of each bit in the string corresponding to the configuration of the 8 possible triples – $000, 001, 010, 011, 100, 101, 110, 111$. The value stored at each of these indices represent the output of the transition rule/function. Since each of theses 8 values is either 0 or 1 there are $2^{2^3} = 2^8 = 256$ possible rules that can be derived, based on possible configurations of the bit string. These are known as “elementary” rules.

2.3 Classification of Rules and Scale Invariance

On applying different transition rules on the same initialization vector (IV), one can notice several classes of behaviour in cellular automata. Some rules lead the automaton to converge to one state invariant to the IV; some do not alter the initial configuration at all through out the course of evolution; some settle into cyclic or periodic patterns; some show very random behavior; some rules are very sensitive to the initial conditions, unlike the ones that converge to the same state regardless of the IV; some show "complex" behavior, just at the critical point between ordered behavior and chaotic behavior, so that structured long-range correlations between cell states start to emerge, even though the cells are limited to a locality of at most 2 neighbours. This gives rise to structures such as triangles appearing in the plot, and sometimes even appearing to move across the plot over the course of time. Cellular automata that showcase complex behavior are of particular interest as most interesting as they show behaviour relevant to modelling biological and physical phenomena such as population activity in the brain, earthquakes, avalanches and so on.

In literature, one dimensional cellular automata rules are broadly divided into 4 classes based on this variety of behaviour that we can observe. *Class 1* rules give rise to homogeneous states where all cells in the automaton end up with the same state. Rules $250, 254$ are examples of *Class 1* rules. *Class 2* rules lead to stable structures or simple periodic patterns; for example, rules $4, 108$. *Class 3* rules lead to seemingly chaotic, non-periodic behavior, as can be seen in the cases of rules $30, 90$. *Class 4* is of particular interest as these rules show complex patterns that have structures that propagate locally in the grid/lattice, sometimes called solitions. For an automaton to be able to do any kind of computation, information needs to be propagated, and solitions are thought to be particles of information preserved through time and moving from one place to the other. Therefore, *Class 4* rules are thought to be capable of computation. Wolfram conjectured that this class of CA are equivalent in computational power to a universal Turing Machine (3). Mathew Cook and Wolfram showed that rule 110 was Turing Complete (4; 5). Examples of rules in this class are $54, 110$.

As a result of the influence of their locality, cellular automaton rules do not define any intrinsic scale. However, over the course of being evolved with certain rules, structure is observable, such as in case of solitions, and these structures are often characterised by long-range correlations; Such structure is said to be "emergent" and the overall entropy of the system decreases as a result of emergent structure. Long-range correlations arising from a system with locally defined rules is a marker for scale invariance, self-similarity and fractal dimensions. Further review about this aspect of cellular automata is outside the scope of this report.

3 Simulation

To implement the cellular automaton, we must define the IV, the transition rule applied and the number of generations it is evolved for.

Once the size (here: length) of the automaton is defined, we initialize an array of 1s and 0s of the given length either randomly, based on some a given density of 1s or with set patterns of 1s and 0s. It is also possible to simply accept any binary bit-string as the IV and make note of its length.

We used the bit string representation of rules as explained in Section 2.2 in our implementation. The rule number R is a decimal integer in the range $[1-256]$ and it is mapped to an $8-$bit binary string R_b as follows :

$$R_b = f_{d\to b}(R-1)$$

where $f_{d\to b}$ converts numbers from their decimal representation to their binary representation.

While applying the rules, two considerations need to be made: the handling of the edge cells, and how the rule itself is applied to the cells on the grid.

The transition rule is applied to every cell in the grid by centering the transition function on the cell of interest, and then computing the next state of that cell by taking into account the states of its immediate neighbours to the left and right; Essentially, triplets centered on the cell of interest are passed to the transition function. All triplets have to be passed into the transition function and this can either be done sequentially or parallely. One must note that all computation should be done based on the states of cells in the previous time-step. Cells cannot be updated in place. Instead, a new state array needs to be made from the newly computed states of the individual cells. After all cells have been populated with updated states, the next generation is produced by applying the transition rule on this newly computed state array. This is repeated to produce all generations of the automaton.

We handled edge cases by treating the array as an infinite, circular grid, by 'attaching' the first cell to the last cell like you would a strip of paper to form a ring. Therefore, the leftmost two cells become the neighbourhood of the rightmost two cells and vice versa.

To produce a visual trace of how the IV evolves in time, we take as an input parameter the number of generations we want to observe in the output. At each time step in evolution, we plot the state array of the automaton as a horizontal row of black or white pixels based on whether the value is (respectively) 1 or 0 in the cells corresponding to the pixels. These rows are then stacked vertically to provide the final plot to show how the entire array evolves in time for a given number of generations. All code associated with this report is available at this Github repository.

4 The Problem of Density Classification

We have seen that cellular automata are governed by local interaction rules. We also observe several emergent phenomena after evolving the CA for several time steps, such as moving solitions, long range correlations, stationary structures such as triangles, and so on. Considering that it is capable of a number of complex behaviors, can we have a cellular automaton that over the course of its evolution, can decide the majority state of the cells in its state array – that is, decide whether 1s or 0s are the majority? This is known as the Density Classification Problem, or the Majority Problem, where the automaton is tasked to somehow classify the initial configuration fed to it as having a majority of 1s or 0s, with only local interaction rules in place.

4.1 Classical problem statement and approaches using 1-D automata

Very briefly : Is there an automaton [rule] that can accurately perform majority voting (or density classification) by converging to a homogeneous state where the state array consists only of the majority element, in all cases? It is important to note that in this statement of the problem, there is great emphasis on finding an automaton that will converge to the majority element. We will see in later sections that the statement of the problem makes a difference to its decidability. One must also remember that this problem is not trivial, since we aim to let the automaton do the computation, and not rely on an external counter to output the number of ones or zeros in a given bit-string.

4.2 The GKL Cellular Automaton

The density classification problem for cellular automata, as stated above, was introduced in 1978 by Gacs, Kurdyumov and Levin (GKL)(6). The GKL automaton is a 2-state model ($k = 2$) with a $r = 3$ neighbourhood.

The GKL CA is a finite $1-$D array of $n \geq 4$ cells (under periodic boundary conditions). The transition function $\Phi : \{0,1\}^n \rightarrow \{0,1\}^n$ that defines the rule of evolution of this CA is as follows:

$$X^{t+1} = \Phi(X^t) = \begin{cases} maj(x^t_{i-3}, x^t_{i-1}, x^t_i) & \text{if } x^t_i = 0 \\ maj(x^t_i, x^t_{i+1}, x^t_{i+3}) & \text{if } x^t_i = 1 \end{cases} \quad (1)$$

where, X^t is the state array of the CA at time step t, x^t_i represents the i^{th} cell in the state array at the same time step and $maj(a,b,c) = \lfloor \frac{(a+b+c)}{2} \rfloor$ where $a, b, c \; \epsilon \; \{0,1\}$. This rule is called the majority rule. The GKL CA is not an elementary CA.

The classification works as $X^t \rightarrow \mathbf{0} = (0, \ldots, 0)$ or $X^t \rightarrow \mathbf{1} = (1, \ldots, 1)$ depending on which symbol was in majority. This formalises the convergence of the state array to a homogeneous state that reflects the majority element.

Some important questions we can ask at this point are: Will all initial configurations converge to a homogeneous state? How long will it take to converge, if it does converge? Can we identify cases where it will not converge? The problem statement does not require that a consensus be reached in case of $\rho = 0.5$, however, it requires that for $\rho_1 > 0.5$ or $\rho_1 < 0.5$ a consensus be reached. It has been shown that the GKL classifier accurately classifies the initial configuration within 100 generations for densities that are not very close to or equal to 0.5

4.3 Undecidability of $\rho = 0.5$

$\rho = 0.5$ implies that the IV has an equal number of 0s and 1s. In order to eventually converge to one state, the GKL rule changes the constitution of the state array at every step of the automaton's evolution. This means that the densities of the symbols in the IV are not retained over the course of evolution, and instead, the classifier changes the densities of the individual symbols in the state array such that the majority element is favoured; Consequently, with each time step, the density of the majority element increases, and that of the minority decreases, until the state array entirely converges to the majority state. However, if the density of the IV is exactly 0.5, at the succeeding time step of evolution, it is meaningless to move in either direction. Due to the nature of the rule applied, the constitution of the state array can change such that at the succeeding time step, the density of one symbol is greater than that of the other. Consequently, a majority is established and the classifier will converge to this new majority symbol that does not reflect the densities of the original IV. It is also possible, depending on the spatial distribution of the symbols in the IV, that the transition rule settles into a limit cycle and simply does not converge. Therefore, at $\rho = 0.5$, we cannot decide whether or not the machine will converge to a symbol. This is still the case for densities very close to 0.5, particularly when the IV length is even.

4.4 Modified problem statement and the Rule 184 CA

Land and Belew (7), in their 1995 paper prove that there exists no perfect two-state cellular automaton for the density classification problem. The GKL rule, as seen above, achieves an accuracy of 97.8%. It is almost always successful on configurations with high or low densities but its accuracy drops when the initial configuration has densities close to 0.5.

Capcarrere et al. (8)showed that if the problem statement is modified a little bit, the rule 184 elementary automaton is able to do the density classification task with 100% accuracy. The output configuration in the original problem is a homogeneous state array consisting only of the majority element. It is observed that the $2-$state, $r = 1$, rule 184 CA, upon presentation of an arbitrary initial configuration, relaxes the grid to a limit cycle within $\lceil N/2 \rceil$ time steps where N is the number of cells. While the output configuration is not a homogeneous convergence to the majority element, the density classification can be observed as follows: If the density of 1s$> 0.5 (< 0.5)$, then the final configuration consists of one or more blocks of at least two consecutive 1s (0s), interspersed by an alternation of 0s and 1s. For an initial density of 0.5, the final configuration is simply an alternation of 0s and 1s. (proof for 184)

The function governing the behaviour of rule 184 can be shown to achieve this desired configuration within a given number of time steps with a 100% accuracy (8). Furthermore, as opposed to classifiers solving the original problem, the output configuration for an initial density of 0.5 is well defined. The complexity of this novel approach as compared to the fixed-point output approach can be quantified in two ways:

1. Kolmogorov Complexity: For a finite string, this is concerned with the size of the shortest program that computes or produces the string. Since we start with an arbitrary initial configuration, this complexity is quite high. However, there is a notable reduction using both the fixed-point output and the novel "blocks" output.

2. Computational Complexity: The complexity of the final configuration in the fixed-point system is that of a simple regular language. Even in the novel approach, the complexity of the output is that of a regular language (identification of a block of two state-0 or state-1 cells).

Thus, by modifying the problem statement, we find a $2-$state, $r = 1$ cellular automaton to solve the density classification problem that is not only as viable as the original one (GKL classifier) in terms of complexity but surpasses it in terms of accuracy.

4.5 Why does rule 184 work?

A detailed proof is given in (8). Very briefly, here are important features of the rule that allows it to classify without error: First, the rule does not change the density of the state array. This means that at every time step of evolution, the state array has a constitution with the same density as that in case of the IV. Instead, the rule changes the configuration of the symbols at every time step. Second, the rule reorganizes the state array in such a way that (as a result of the pigeon hole principle) only the majority element appears in a non-alternating fashion, Therefore, if a symbol appears in a contiguous sequence of length greater than or equal to 2, then that symbol is the local majority. Third, these contiguous sequences move to the right (if the sequence is of 0s) or to the left (if the sequence is of 1s) at every time step of evolution, and if two such streams ever meet at a given time step, then they combine in an alternating fashion.

4.6 Discussion: Why not this approach?

While the rule 184 classifier has a 100% accuracy, it still requires that there be some form of external computation in order to determine whether or not the machine has settled into a limit cycle, and if there is a majority element, detect where they occur as contiguous clumps of cells in the same state. The external computation can be done in several ways: by keeping a counter, using a regular expression query and so on. However, this is not a viable model for physical systems as there are no external counters in such systems that keep track of the states of all individual components in the network. Therefore, while rule 184 is clearly capable of computing the majority, the majority element is only evident to an external observer. For this reason, rule 184 is not preferred, neither is this problem statement.

5 An Application: Stomatal patchiness as a form of emergent computation in leaves

The following section is a brief review of (2) as a possible application of cellular automata in understanding physical phenomena. Gaseous exchange in leaves is facilitated by stomata; Stomata are discrete entities on surfaces of leaves that must collectively solve the problem of adjusting their apertures such that the leaf can conduct optimal gaseous exchange: take up sufficient carbon dioxide while minimising water loss by transpiration through the open apertures. This is in the face of spatially and temporally heterogeneous environmental conditions, and the problem is compounded by the fact that decisions have to be made at the organism level, in the absence of any kind of neural tissue that could provide long-range connections.

Under certain conditions, stomatal apertures in leaves show synchronized activity in patches that exhibit complex dynamics analogous to behaviours of cellular automata that are capable of computation. While it has been suggested that some biological processes are equivalent to computation, the quantitative evidence for the view is somewhat weak. This paper shows that spatial and temporal correlations in stomatal dynamics show values that are statistically indistinguishable from those found in the dynamics of automata that compute. The results of this study suggest that plants engage in a form of emergent, distributed computation.

The paper hitches on the density classification problem as an apt example of emergent, distributed computation that the grid of stomatal apertures are purported here to be]capable of doing. The paper does so by comparing the statistical dynamics of the $1-D$ GKL classifier and the $2-D$ cellular automaton that simulates sandpiles. There is a qualitative similarity between the $2-D$ sandpile automata that model the spreading of avalanches of activity, the $1-D$ automata that perform density classification by means of growing patches of synchronized states, and stomatal patchiness. The authors purport that if the analogy between the surface of a leaf and the density classifier is apt, then the rarely appearing long-transient episodes of stomatal patchiness correspond to hard problems that the leaves struggle to solve. In terms of the GKL classifier, this corresponds to the classification of densities very close to 0.5, as they take the longest time to converge.

Considering that leaf surfaces would only facilitate such long-transient patchy activity if some particle of information is transmissible, the search is then directed towards identifying the presence of solitions. The study was able to show that the classifier CA did indeed produce solitions much like that seen in leaves showing stomatal patchiness, as visualised using chlorophyll fluorescence. To note is that this visualization is a reliable indicator of stomatal aperture state (open or close) and the relationship is given as fluorescence inversely proportional to stomatal conductance (gaseous exchange is permitted when the aperture is open, and chlorophyll fluoresces when the aperture is closed).

The paper thus takes the first step in giving stronger evidence showing that biological phenomena such as stomatal patchiness involve computations essentially identical to those of some cellular automata that show emergent and distributed computation.

References

[1] S. Wolfram, “Statistical mechanics of cellular automata,” *Rev. Mod. Phys.*, vol. 55, pp. 601–644, Jul 1983. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.55.601

[2] D. Peak, J. D. West, S. M. Messinger, and K. A. Mott, “Evidence for complex, collective dynamics and emergent, distributed computation in plants,” *Proceedings of the National Academy of Sciences*, vol. 101, no. 4, pp. 918–922, 2004. [Online]. Available: https://www.pnas.org/content/101/4/918

[3] S. Wolfram, *A New Kind of Science.* Wolfram Media, 2002. [Online]. Available: https://www.wolframscience.com

[4] M. Cook, “Universality in elementary cellular automata,” *Complex Systems*, vol. 15, pp. 1–40, 2004.

[5] A. Ilachinski, *Cellular Automata.* WORLD SCIENTIFIC, 2001. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/4702

[6] P. Gacs, G. L. Kurdyumov, and L. A. Levin, “One-dimensional uniform arrays that wash out finite islands,” *Problems Inform. Transmission*, vol. 14, pp. 223–226, 1978. [Online]. Available: https://zbmath.org/?q=an:0393.68060

[7] M. Land and R. K. Belew, “No perfect two-state cellular automata for density classification exists,” *Physical Review Letters*, vol. 74, no. 25, p. 5148–5150, 1995.

[8] M. S. Capcarrere, M. Sipper, and M. Tomassini, “Two-state, $r = 1$ cellular automaton that classifies density,” *Phys. Rev. Lett.*, vol. 77, pp. 4969–4971, Dec 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.77.4969