GENERATION OF ISOMORPHIC CELLULAR AUTOMATA

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

M.Tech

in

Computer Science and Engineering

By

TARUN DITTAKAVI (206122028)

COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI – 620015

JUNE 2024

BONAFIDE CERTIFICATE

This is to certify that the project titled **GENERATION OF ISOMORPHIC CELLULAR AUTOMATA** is a bonafide record of the work done by

Tarun Dittakavi (206122028)

in partial fulfillment of the requirements for the award of the degree of **Master of Technology** in **Computer Science and Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2023–24.

Kamalika Bhattacharjee

DR. KAMALIKA BHATTACHARJEE
Guide

DR. S MARY SAIRA BHANU
Head of the Department

Project Viva-voce held on _____

Internal Examiner

External Examiner

ABSTRACT

Finite cellular automata can be represented as graphs, using transition diagrams. Isomorphism in cellular automata is defined with respect to the structure of their transition diagrams. Two cellular automata are said to be isomorphic if their transition diagrams are isomorphic to each other. This thesis explores isomorphism in cellular automata and introduces a new concept, named *pseudo-isomorphism*. In this thesis, some methods for synthesizing (pseudo) isomorphic cellular automata have been discussed. These methods specifically focus on generating a set of (pseudo) isomorphic cellular automata, from a given cellular automaton, in such a way that the resulting isomorphisms preserve the neighborhood architecture and boundary condition of the original cellular automaton.

Additionally, a new mathematical notion, named *state-neighborhood relation* has been introduced and is used to determine if a particular set of cycles in the transition diagram can be reversed to generate a (pseudo) isomorphic cellular automaton. These results are then specialized to reversible finite cellular automata, to derive a condition for when an isomorphic automaton can be obtained by reversing its entire transition diagram. This is possible due to the fact that the transition diagrams of such cellular automata consists entirely of disjoint directed cycles.

Keywords : Isomorphism; Pseudo-Isomorphism; Reversibility; State-Neighborhood Relation; Transition Diagrams

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to the following people for guiding me through this course and without whom this project and the results achieved from it would not have reached completion.

Dr. Kamalika Bhattacharjee, Assistant Professor, Department of Computer Science and Engineering, for helping me and guiding me in the course of this project. Without her guidance, I would not have been able to successfully complete this project. Her patience and genial attitude is and always will be a source of inspiration to me.

Dr. S Mary Saira Bhanu, Head of the Department, Department of Computer Science and Engineering, for allowing me to avail the facilities at the department.

DPEC members — **Dr. Sridevi M**, **Dr. Oswald C** and **Dr. Sai Krishna M**, for their valuable suggestions given during the reviews.

I am also thankful to the faculty and staff members of the Department of Computer Science and Engineering, for their constant support and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Cellular automata (CAs) are dynamical systems that exhibit complex global behavior from simple local interactions [1]. A cellular automaton (CA) is made up of cells that are arranged in a regular network. Out of the finite set of possible states, each cell will be in any one of those states at a given instant. At the next time step, each cell updates its state based on its current state and the states of those cells that are surrounding it. These surrounding cells are also known as *neighbors* to the cell and need not be strictly adjacent to it. So there is a parameter of the CA known as *neighborhood vector*, that defines which cells are neighbors to any particular cell of the CA. Below is the formal definition of a CA, taken directly from Ref. [1].

Definition 1.1. *A CA is a quadruple* $(\mathcal{L}, \mathcal{S}, \mathcal{N}, f)$*, where:*

- $\mathcal{L} \subseteq \mathbb{Z}^D$*, is a D-dimensional cellular space. A set of cells are placed in the locations of* $\mathcal{L}$ *to form a regular network.*
- $\mathcal{S}$ *is the finite state set.*
- $\mathcal{N} = \langle \overrightarrow{v_1}, \overrightarrow{v_2}, \ldots, \overrightarrow{v_N} \rangle$*, is the neighborhood vector of* N *distinct elements of* $\mathcal{L}$*, which associates a cell with its neighbors. Generally, the neighbors of a cell are the nearest cells surrounding it. However when the neighborhood vector* $\mathcal{N}$ *is given, then the neighbors of a cell at location* $\overrightarrow{v} \in \mathcal{L}$ *are at locations* $(\overrightarrow{v} + \overrightarrow{v_i}) \in \mathcal{L}$*, for all* $i \in \{1, 2, \ldots, N\}$*.*
- $f : \mathcal{S}^N \to \mathcal{S}$*, is called the local rule of the CA. The next state of a cell is given by* $f(a_1, a_2, \ldots, a_N)$*, where* $a_1, a_2, \ldots, a_N$ *are the states of its* N *neighbors.*

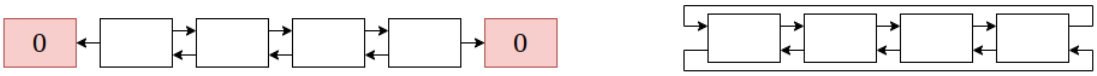
In the above definition, $\mathcal{S}^N$ (the domain of f) is the set of all neighborhood patterns a cell can have, based on the neighborhood vector $\mathcal{N}$. For ease of writing, this set of all neighborhood patterns will be denoted by $\mathcal{N}$, in the rest of this thesis.

1.1 Finite CAs and Boundary Conditions

Theoretically, the cellular space $\mathcal{L}$ is infinite, extending in all the D dimensions. There is no concept of boundary condition for such infinite spaces, as there would be no terminal cells. Every cell will have all of its neighbors, as defined by the neighborhood vector. However to simulate a CA in computational environments, we

need to limit the cellular space to a finite size. A CA is said to be a *finite CA*, if the cellular space $\mathcal{L}$ is finite. Otherwise, the CA is said to be an *infinite CA* [1].

Boundary conditions are used to specify the neighbors for terminal cells in a finite CA. The most widely used boundary conditions are the null and periodic boundary conditions. In CAs that follow null boundary condition, the missing neighbors of terminal cells are always considered to be in a fixed state of 0 (null). On the contrary, in CAs that follow periodic boundary condition, the cells are considered to be arranged in a cyclic manner. Hence there are technically no terminal cells in a CA following periodic boundary condition. Fig. 1.1 illustrates null and periodic boundary conditions for a finite 1D CA. A finite 1D CA having n cells is arranged in the form of a linear array, where each cell is addressed by its index i ($0 \leq i < n$).



(a) Null boundary condition (b) Periodic boundary condition

Figure 1.1: Boundary conditions in finite 1D CAs

Apart from the null and periodic boundary conditions, there are other boundary conditions, such as adiabatic, reflexive and intermediate boundary conditions. These are out of scope for the current thesis. Brief descriptions of these terms can be found in Ref. [1].

1.2 Elementary CAs

An elementary CA (ECA) is a binary ($\mathcal{S} = \{0, 1\}$) 1D CA, where a cell depends on the states of its immediate left and right neighbors, along with its current state, to transition into a new state.

1.3 Local Rule

The local transition function f, also known as the *local rule*, determines the next state of a cell, based on its current neighborhood pattern, which is defined by the neighborhood vector $\mathcal{N}$. Local rules are typically encoded as numbers for the ease of representation. The neighborhood pattern of a cell can be expressed as a d-ary string (d being the number of possible states of a cell), that is obtained by concatenating the states of all the cells as defined by $\mathcal{N}$.

In case of ECAs, the neighborhood vector $\mathcal{N} = \langle -1, 0, 1 \rangle$, where -1 represents the immediate left neighbor, 0 represents the current cell itself and 1 represents the immediate right neighbor. So the neighborhood pattern for a cell of an ECA can

be any of the following — $000, 001, 010, 011, 100, 101, 110, 111$. For any of these patterns, the next state of a cell is going to be either 0 or 1. This next state is determined by the local rule followed by the cell. Some possible local rules for ECAs are shown in Table 1.1.

Table 1.1: Sample ECA rules

Rule	**111**	**110**	**101**	**100**	**011**	**010**	**001**	**000**
30	0	0	0	1	1	1	1	0
90	0	1	0	1	1	0	1	0
150	1	0	0	1	0	1	1	0

Every ECA rule is assigned a number, which is the decimal equivalent of the binary string obtained by listing the resultant states of the rule for each of the eight neighborhood patterns — 111 to 000, which are also known as *rule min terms (RMTs).*

1.4 Uniform and Non-Uniform CAs

If all the cells of a CA follow the same local rule, then it is said to be a *uniform CA*. Otherwise, it is a *non-uniform CA*. The formal definition of a CA, as the quadruple $(\mathcal{L}, \mathcal{S}, \mathcal{N}, f)$, covers only uniform CAs. In case of finite 1D CAs of size n, the local rule f in the quadruple $(\mathcal{L}, \mathcal{S}, \mathcal{N}, f)$ is replaced with the rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1} \rangle$, to accommodate non-uniform CAs within the quadruple definition.

This non-uniformity can be further extended to allow different cells to have different neighborhood vectors. However such CAs are not considered in this thesis. This work only considers finite 1D CAs of size n, where the neighborhood vector of a cell consists of l cells immediately to its left, the cell itself and r cells immediately to its right. Here l is known as the *left radius* and r is known as the *right radius* of a cell. Each cell assumes any state from a finite set of states $\mathcal{S} = \{0, 1, \ldots, d-1\}$ and takes a rule from the rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, ..., \mathcal{R}_{n-1} \rangle$ to update its state. In case of uniform CAs, $\mathcal{R}_0 = \mathcal{R}_1 = \cdots = \mathcal{R}_{n-1}$.

1.5 Configurations and Global Transition Function

The states of all the cells of a CA, taken together at any point of time is known as its *configuration* at that instant. As a CA evolves over time (that is, as the cells update their states), it hops from one configuration to another.

The notion of viewing a CA in terms of configurations leads us to describe the CA using a *global transition function*. Let $\mathcal{C}$ represent the set of all configurations $\mathcal{S}^{\mathcal{L}}$ of a CA. The global transition function for this CA, $\mathcal{G} : \mathcal{C} \rightarrow \mathcal{C}$ takes a configuration as an input and yields the next configuration as the output. For a finite 1D CA with n cells, the configuration space is denoted by $\mathcal{C}_n$ and the global transition function is denoted by $\mathcal{G}_n$.

1.6 Synchronous and Asynchronous CAs

Traditionally CAs are considered to be synchronous, that is, there is a clock that synchronizes the state updates of all the cells. All cells of the CA update their states simultaneously, at every tick of the clock. Majority of the research on CAs deals with synchronous CAs itself. However over the past few decades, asynchronous CAs have been introduced, where each cell can update its state, independent of the other cells in the network. So in an asynchronous CA system there is no global clock that handles the synchronization of state updates of all the cells. Asynchronous CAs are out of scope for this thesis, as the current project only deals with synchronous CAs. Further discussion about asynchronous CAs can be found in Ref. [1].

1.7 Linear CAs

The local rule f of a CA $(\mathcal{L}, \mathcal{S}, \mathcal{N}, f)$ is said to be a *linear rule*, if $f : \mathcal{S}^{\mathcal{N}} \rightarrow \mathcal{S}$ can be expressed as $f(a_1, a_2, \ldots, a_N) = k_1.a_1 + k_2.a_2 \ldots k_N.a_N$, where, each $k_i \in \mathcal{S}$ is a constant. Linear rules are also known as additive rules. A non-uniform CA is said to be linear/additive, if the local rules for all of its cells are linear/additive.

Of particular interest are linear rules for ECAs. Of the 256 ECA rules (0 to 255), there are exactly 7 linear rules — 60, 90, 102, 150, 170, 204 and 240. This list excludes 0, which is trivially linear. One interesting property of these linear ECA rules is that they can be expressed as an XOR of the input variables. For example, rule 150 can be expressed as $S_i(t+1) = S_{i-1}(t) \oplus S_i(t) \oplus S_{i+1}(t)$, where $S_i(t)$ represents the state of i^{th} cell at time t.

Since these linear rules are expressed using XOR logic, their complements can be expressed using XNOR logic. For example, the complement of rule 60, which is (255 – 60) = 195, can be expressed as $S_i(t+1) = \overline{S_{i-1}(t) \oplus S_i(t)}$. Table 1.2 shows the XOR expressions for each of the linear ECA rules and the XNOR expressions for the corresponding complemented rules.

Table 1.2: Linear ECA rules and their complements

Rule	XOR Expression	Rule	XNOR Expression
60	$S_i(t+1) = S_{i-1}(t) \oplus S_i(t)$	195	$S_i(t+1) = \overline{S_{i-1}(t) \oplus S_i(t)}$
90	$S_i(t+1) = S_{i-1}(t) \oplus S_{i+1}(t)$	165	$S_i(t+1) = \overline{S_{i-1}(t) \oplus S_{i+1}(t)}$
102	$S_i(t+1) = S_i(t) \oplus S_{i+1}(t)$	153	$S_i(t+1) = \overline{S_i(t) \oplus S_{i+1}(t)}$
150	$S_i(t+1) = S_{i-1}(t) \oplus S_i(t) \oplus S_{i+1}(t)$	105	$S_i(t+1) = \overline{S_{i-1}(t) \oplus S_i(t) \oplus S_{i+1}(t)}$
170	$S_i(t+1) = S_{i+1}(t)$	85	$S_i(t+1) = \overline{S_{i+1}(t)}$
204	$S_i(t+1) = S_i(t)$	51	$S_i(t+1) = \overline{S_i(t)}$
240	$S_i(t+1) = S_{i-1}(t)$	15	$S_i(t+1) = \overline{S_{i-1}(t)}$

1.8 Reversible and Irreversible CAs

A CA whose global transition function $\mathcal{G}$ is a bijection, is said to be a *reversible CA*. Otherwise it is an *irreversible CA*. A finite CA is reversible, if and only if, for every configuration $c \in \mathcal{C}$ there exists $k \in \mathbb{N}$ such that $\mathcal{G}^k(c) = c$. A finite reversible CA loops around a set of configurations again and again, in a cyclic manner, that is, the configurations of the CA group together into components, in such a way that when the CA is initialized with a configuration belonging to a component, all the configurations in that component get repeated after regular intervals of time.

1.9 Characteristic Matrices

A linear ECA of n cells can be represented as an $n \times n$ matrix $\mathcal{M}$, known as its *characteristic matrix*. The i^{th} row of $\mathcal{M}$ corresponds to the rule $\mathcal{R}_i$ of the i^{th} cell in the ECA, representing the cell's dependency on its neighbors.

In case of ECAs, this characteristic matrix $\mathcal{M}$ is a tri-diagonal matrix, whose entries are all either 0 or 1 (elements in the field $GF(2)$). Below is an expression that defines the value of each cell of the characteristic matrix $\mathcal{M}$.

$$\mathcal{M}[i,j] = \begin{cases} 1 & \text{if the next state of cell } i \text{ is based on present state of cell } j \\ 0 & \text{otherwise} \end{cases}$$

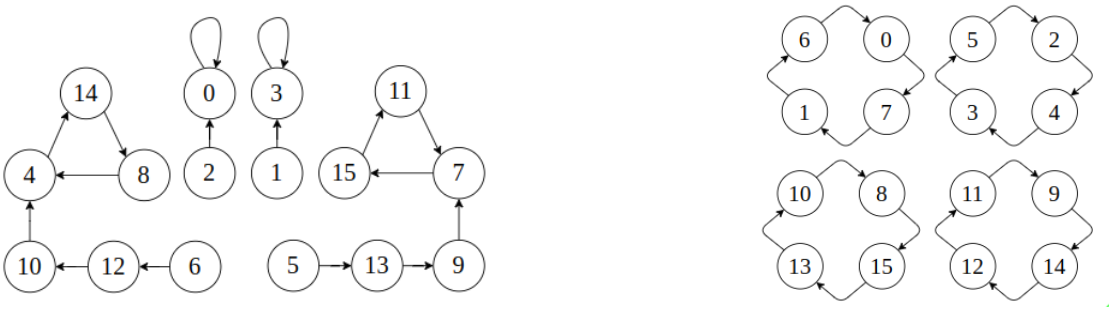
Below is an example of a characteristic matrix. It represents a 4-cell non-uniform ECA with rule vector $\langle 60, 170, 90, 150 \rangle$ under null boundary condition.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

From the characteristic matrix $\mathcal{M}$ of a linear ECA, its characteristic polynomial $\mathcal{P}$ can be computed, which can also be used to represent the ECA. It is to be noted that since the elements of $\mathcal{M}$ are elements of $GF(2)$, all the operations involved in computing $\mathcal{P}$ also occur in $GF(2)$. This means that the coefficients of $\mathcal{P}$ will also be elements of $GF(2)$, that is, 0 or 1.

1.10 Transition Diagrams

The evolution of finite CA can be represented as a directed graph where each vertex represents a particular configuration and each edge represents a transition from one configuration to another. For a finite 1D CA of size n, an edge from vertex $c_x = x_0x_1 \dots x_{n-1}$ to vertex $c_y = y_0y_1 \dots y_{n-1}$, for $c_x, c_y \in \mathcal{C}_n$, means that c_x is a predecessor of c_y. In the above notation x_i and y_i denote the state of i^{th} cell in configurations c_x and c_y respectively. Such a graph is known as the *transition diagram* of a CA. For a finite reversible CA, the transition diagram consists entirely of disjoint directed cycles. Figure 1.2 shows the transition diagrams (with configurations written in their decimal equivalents) of an irreversible ECA with rule vector $\langle 90, 60, 154, 70\rangle$ and a reversible ECA with rule vector $\langle 12, 147, 51, 65\rangle$ under null boundary condition.



(a) $\langle 90, 60, 154, 70\rangle$ (b) $\langle 12, 147, 51, 65\rangle$

Figure 1.2: Examples of transition diagrams under null boundary condition

1.11 Isomorphic CAs

The idea of representing CAs as graphs leads us to the concept of isomorphism in CAs. Firstly, two directed graphs G and H are said to be isomorphic to each other if there exists a bijection $f : V(G) \to V(H)$ from the vertex set of G to vertex set of H, such that two vertices u and v of G are adjacent if and only if the vertices $f(u)$ and $f(v)$ of H are adjacent and the direction of each edge is preserved. Two CAs are said to be isomorphic if their transition diagrams are

isomorphic [2]. Figure 1.3 shows that the ECAs with rule vectors $\langle 9, 142, 165, 65\rangle$ and $\langle 6, 212, 90, 65\rangle$ are isomorphic to each other under periodic boundary condition.

(a) $\langle 9, 142, 165, 65\rangle$ (b) $\langle 6, 212, 90, 65\rangle$

Figure 1.3: Example of isomorphic CAs

To obtain a graph that is isomorphic to the given graph, we just need to permute the labels of nodes in the original graph. This can be done trivially and we can have $n!$ such permutations, where n is the number of nodes in the graph. However all these $n!$ permutations do not necessarily produce a unique isomorphic graph. There can be repetitions among the graphs produced and this is highly dependent on the structure of the original graph itself. So we can say that for a graph of n nodes, there exist at most $n!$ isomorphic graphs.

In this thesis, we extend the concept of isomorphism by defining *pseudo-isomorphism* in CAs and explore schemes for generating (pseudo) isomorphic CAs, by maintaining the invariant of same neighborhood architecture and boundary condition.

CHAPTER 2

LITERATURE REVIEW

This chapter gives a brief overview of the existing literature that is relevant to the current thesis. We start with works describing computational bounds for the problem of graph isomorphism and then explore works on CAs that deal with synthesis of isomorphic CAs, linearity and synthesis of reversible CAs.

2.1 Computational Bounds for Graph Isomorphism

The problem of graph isomorphism aims to determine if two given graphs A and B are isomorphic to each other. The naive way for solving this problem would be to check if B can be obtained by relabeling the nodes of A. The time complexity of this approach would be $\mathcal{O}(n!)$, where n is the number of nodes in A.

Efficiently solving the problem of graph isomorphism is an ongoing field of research where the latest algorithm to check whether two graphs are isomorphic takes *quasi-polynomial* time of its number of vertices [3]. The purpose of this work is to give a guaranteed upper bound on the computational steps required to solve the problem, rather than provide a practical solution.

In case of transition diagrams, the number of vertices for a d-state n-cell CA is equal to the number of configurations of the CA, that is, d^n . So testing for isomorphism of two CAs using their transition diagrams cannot be done in time lesser than $\Omega(d^n)$. In this situation Ref. [2] describes two synthesis algorithms to generate a set of isomorphic CAs from the transition diagram of a given n-cell CA. These schemes are described next.

2.2 Synthesis Schemes on Isomorphism

Ref. [2] describes two synthesis algorithms to generate a set of isomorphic CAs from the transition diagram of a given n-cell CA. They are known as *state map method* and *exchanging cell states method*. Using state map method, the generated isomorphic CAs have the same neighborhood architecture and boundary condition as the original CA. However this invariant is not guaranteed with the exchanging cell states method.

2.2.1 State Map Method

For an n-cell 1D CA having $\mathcal{S} = \{0, 1, \ldots, d-1\}$, the state map method can be used to produce at most $(d!)^n - 1$ isomorphic CAs. This method works by defining n permutations (that is, bijections) π_0 to π_{n-1} on the set of states $\mathcal{S}$. For every configuration $c_x = (x_0, x_1, \ldots x_{n-1}) \in \mathcal{C}_n$, the mapping $\pi(c_x) = (\pi_0(x_0), \pi_1(x_1), \ldots, \pi_{n-1}(x_{n-1}))$ is computed. It is proved that the mapping $\pi : \mathcal{C} \to \mathcal{C}$ is a bijection, that is, for two configurations $c_x, c_y \in \mathcal{C}$, such that $c_x \neq c_y$, we will have $\pi(c_x) \neq \pi(c_y)$ and also every configuration in $\mathcal{C}$ will have a pre-image in $\mathcal{C}$, with respect to π. So applying π on each node of the transition diagram will result in a permutation of the nodes of the transition diagram. It is proved that this resulting permutation will correspond to a CA having the same neighborhood architecture and boundary condition. If the configuration c_x is a predecessor of the configuration c_y in the original CA, then the configuration $\pi(c_x)$ will be a predecessor of the configuration $\pi(c_y)$ in the obtained isomorphic CA, for all $c_x, c_y \in \mathcal{C}$.

Example 2.1. Consider a 3-cell ECA with rule vector $\langle 6, 90, 20 \rangle$ under null boundary condition. The permutations π_0, π_1 and π_2 are defined as below.

$$\pi_0(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{if } x = 1 \end{cases} \quad \pi_1(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{if } x = 1 \end{cases} \quad \pi_2(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{if } x = 1 \end{cases}$$

Using state map method with these permutations on the above ECA results in an isomorphic ECA with rule vector $\langle 9, 165, 65 \rangle$ under null boundary condition. Figure 2.1 demonstrates this example.

(a) $\langle 6, 90, 20 \rangle$ (b) $\langle 9, 165, 65 \rangle$

Figure 2.1: Example of state map method

2.2.2 Exchanging Cell States Method

For an n-cell 1D CA, two positions i, j are selected such that $i \neq j$ and $0 \leq i, j \leq (n-1)$. For every configuration $c_x = (x_0, x_1, \ldots, x_i, \ldots, x_j, \ldots, x_{n-1}) \in \mathcal{C}$, the states of cells at positions i, j are exchanged, so as to get the new configuration

as $c'_x = (x_0, x_1, \ldots, xj, \ldots, x_i, \ldots, x_{n-1})$. Applying this transformation on each node of the transition diagram results in an isomorphic image of the transition diagram. However as stated previously this newly obtained isomorphic transition diagram might not correspond to a CA with the same neighborhood architecture and boundary condition as that of the original one.

Example 2.2. Consider a 3-cell ECA with rule vector $\langle 6, 90, 20 \rangle$ under null boundary condition. Applying the exchanging cell states method on this ECA, with $i = 1$ and $j = 2$, does not result in an ECA under the same boundary condition, but results in a CA with rule vector $\langle 90, 5204, 60 \rangle$ under null boundary condition, where the first two cells have a left radius of 1 and right radius of 2, while the last cell has a left radius of 2 and right radius of 0. Figure 2.2 demonstrates this example.

(a) $\langle 6, 90, 20 \rangle$ (b) $\langle 90, 5204, 60 \rangle$

Figure 2.2: Example of exchanging cell states method

2.3 Cycle Structure of Complemented Linear CAs

Ref. [4] includes an analysis of linear ECAs, from an algebraic perspective. This analysis makes use of vector spaces and characteristic polynomials of CAs over the field $GF(2)$. One of the results regarding the cycle structure of linear ECAs is of particular interest with respect to isomorphism in CAs. Firstly, the term *complemented rule vector* needs to be defined. For an n-cell ECA with rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1} \rangle$, the rule vector obtained by replacing one or more rules by their corresponding complementary rules is known as a complemented rule vector for the original CA.

So for a CA of n cells, there will be 2^n–1 complemented rule vectors. Ref. [4] proved that the cycle structure of a linear ECA $\mathcal{CA}_1$ and the one containing any of the complemented rule vectors of $\mathcal{CA}_1$ as its rule vector, will be same if the characteristic polynomial of $\mathcal{CA}_1$ does not have a factor of $(x + 1)$. The exact theorem from Ref. [4] is stated as below.

Theorem 2.1. *The cycle structures of complemented CA and its corresponding linear CA are identical if the characteristic polynomial $f(x)$ of T matrix of the complemented CA does not have a factor $(x + 1)$.*

Remark 2.1. As mentioned previously, the transition diagram of a reversible ECA consists entirely of disjoint directed cycles. So if two reversible CAs have the same cycle structure, then they are isomorphic to each other. This implies that for an n-cell reversible linear ECA not having a factor of $(x+1)$ in its characteristic polynomial, all the 2^n-1 complemented ECAs are isomorphic to it, since they have the same cycle structure.

Example 2.3. The reversible linear ECA with rule vector $\langle 150, 240, 90, 60\rangle$ under null boundary condition has $\mathcal{P} = x^4+x^2+1$ as its characteristic polynomial. Since $\mathcal{P}$ does not have a factor of $(x+1)$, as per the above remark, all of its 15 complemented ECAs under null boundary condition are isomorphic to it.

2.4 Synthesis Schemes on Reversibility

CAs having cycles in their transition diagrams are targeted by the *reversing cycles algorithm*, that will be described later in the thesis. For testing the implementation of this algorithm, such CAs need to be synthesized. The natural choice for CAs having cycles in their transition diagrams would be reversible CAs. To synthesize reversible ECAs under null and periodic boundary conditions, the respective schemes described in Ref. [5, 6] are used.

These synthesis schemes work by grouping the 70 balanced ECA rules[1] into classes, in such a way that the rule class of the current cell determines the rule class of the next cell. For null boundary condition the classification is done into 3 tables, one for the first cell, one for the intermediate cells and one for the last cell. Similarly, for periodic boundary condition the classification is done into 2 tables, one for the first cell and one for the remaining cells. The complete tables can be referred in the works cited above.

To generate a reversible ECA, we begin by randomly selecting a rule from the first cell rule table. This gives the rule class for the second cell. Now from the obtained class, we select any of the available rules in that class randomly, from the intermediate cell rule table. This gives the rule class for the next cell. This process is continued until the rules for all the cells have been determined. It is important to note that the rule for last cell of the CA should be selected from the last cell rule table and not the intermediate cell rule table, for null boundary condition.

ECAs obtained by this synthesis scheme are reversible under null boundary condition. However the synthesized ECAs are not always reversible under periodic boundary condition.

[1] An ECA rule is said to be *balanced*, if the next state is 0 for any of the 4 (out of 8) RMTs and 1 for the remaining 4 RMTs. So there are a total of ${}^8C_4 = 70$ possible balanced rules, out of the 256 ECA rules.

CHAPTER 3

METHODOLOGY

In this chapter we briefly describe the exhaustive search method for generating isomorphic CAs and then explore concepts of pseudo-isomorphism, along with the reversing cycles algorithm for generating pseudo-isomorphic CAs. Even though the exhaustive search method does not provide any algorithmic advantage, it is useful to describe it here to understand the hardness of the problem. Along with that, a subroutine used in exhaustive search for determining if a transition diagram corresponds to a CA or not will be used in reversing cycles as well.

3.1 Exhaustive Search for Isomorphisms

Searching for isomorphic CAs begins by exploring every possible permutation of the transition diagram G of a given CA $\mathcal{CA}_1$. Every permutation of the G will result in an isomorphic transition diagram H. However this H does not necessarily emulate a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$. So after generating H, the task would be to verify if it corresponds to a CA with the same neighborhood architecture and boundary condition or not. The transition diagram H corresponds to a CA, if every cell follows a *valid* local rule over the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$, while updating its state by following the transitions defined in H. A cell is said to follow a *valid* local rule, under a specified neighborhood architecture and boundary condition, if there does not exist any neighborhood pattern n_j under that condition, for which the cell transitions into more than one state.

To check if a cell i follows a valid local rule or not, every node (configuration) of H needs to be visited to obtain the neighborhood of i at that node, along with the state to which i is making a transition. The rule followed by i will be valid, if no collisions are encountered for any neighborhood of i. If any of the cells violate the local rule property, then H does not correspond to a CA having the same neighborhood architecture and boundary condition, and further processing of H can be terminated. On the other hand, if all the cells follow a local rule, then H corresponds to a CA (say $\mathcal{CA}_2$) and the rule vector of $\mathcal{CA}_2$ can be returned, which comprises of the rules extracted for each cell while checking for local rule compliance.

The code snippet in Listing 3.1 shows C++ implementation of this method. The

complete source code is not included here, so as to focus on the algorithm itself and not get distracted by implementation specific details. The actual code can be found in the Github repository mentioned in the next chapter. A flowchart demonstrating the high level overview of this algorithm is presented in Figure 3.1.

```
void
binary_1d_ca::print_isomorphisms()
{
  vector<unsigned short> this_graph = this->get_graph();
  vector<unsigned short> this_permutation;

  for (unsigned short i = 0; i < this->num_configs; i++)
  {
    this_permutation.push_back(i);
  }

  vector<unsigned short> current_graph(this->num_configs, 0);
  vector<unsigned short> current_rules(this->num_cells, 0);
  vector<unsigned short> current_permutation = this_permutation;

  while (current_permutation)
  {
    for (unsigned short j = 0; j < current_permutation.size(); j++)
    {
      unsigned short current_config = current_permutation[j];
      current_graph[current_config] = current_permutation[this_graph[j]];
    }

    bool is_valid_ca = this->extract_rules(current_graph, current_rules);

    if (is_valid_ca)
    {
      print(current_rules);
    }

    current_permutation = next_permutation(current_permutation);
  }
}

```

Listing 3.1: Implementation of exhaustive search

It can be seen in Listing 3.1 that a method named `extract_rules()` is used to determine if the current permutation of the transition diagram corresponds to a CA with the neighborhood architecture and boundary condition or not. It returns a boolean value indicating the same. It also does the work of extracting the local rule of each cell, if the permutation of the transition diagram does emulate a CA under the required conditions. So this method involves the logic of checking for local rule compliance for every cell. The code snippet implementing this `extract_rules()` method is shown in Listing 3.2.

3.1.1 Time Complexity of Exhaustive Search Algorithm

For a d-state 1D CA of n cells, there will be d^n nodes in its transition diagram G. So there will be a maximum of $d^n!$ distinct isomorphic permutations of the transition diagram. Every such permutation H needs to be checked if it would correspond to

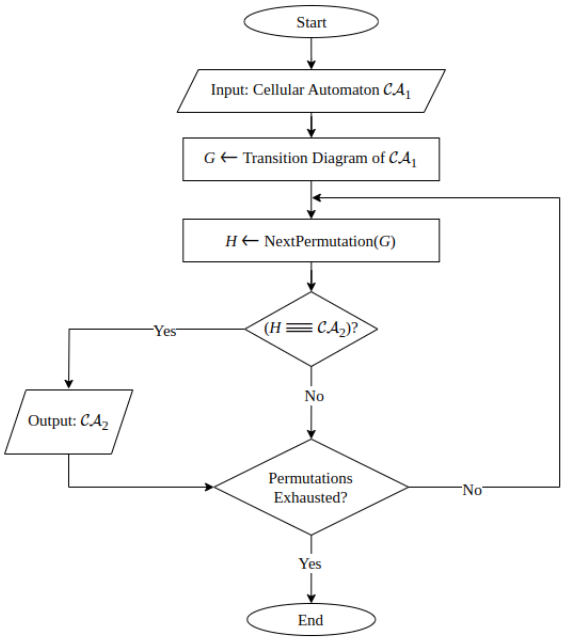


Figure 3.1: Exhaustive search algorithm

a CA with the same neighborhood architecture and boundary condition. This fact itself establishes the hardness of the problem, as we need to process each of the $d^n!$ permutations. To check if H corresponds to CA or not, we need to check if every cell follows a local rule. This means we need to process all the n cells that form the transition diagram H. To check if any cell follows a local rule, all the d^n nodes (configurations) of H should be evaluated. Time taken to evaluate each node of H for a particular cell's local rule compliance depends upon the neighborhood size of the CA under consideration. The maximum neighborhood size a CA can have is equal to the number of cells in it, which is n. So the time taken to evaluate a node for a particular cell's local rule compliance would be $\mathcal{O}(n)$. Hence, the overall time taken to check if H corresponds to CA with the same neighborhood architecture and boundary condition as that of the given CA would be $\mathcal{O}(n * (d^n * n))$, that is, $\mathcal{O}(n^2 * d^n)$. Since this operation has to be repeated for all the $d^n!$ permutations of G, the total time complexity of the algorithm would be $\mathcal{O}(d^n! * (n^2 * d^n))$.

3.1.2 Space Complexity of Exhaustive Search Algorithm

Storing a CA implies storing the states of all cells at a particular point of time and their local rules. For a d-state 1D CA of size n, this would require a space of $O(n)$. But since the CA is taken as input by the algorithm, space occupied by it can be ignored while computing the space complexity. The transition diagram G can be stored as a 1D array of size d^n, where the entry $G[c_x]$ corresponds to the configuration c_x. So $G[c_x] = c_y$ means that configuration c_y is a successor of

```
bool
binary_1d_ca::extract_rules(
  vector<unsigned short> &graph,
  vector<unsigned short> &rules
)
{
  for (unsigned short i = 0; i < this->num_cells; i++)
  {
    vector<char> current_rule_row((1U << this->num_neighbors), 'X');
    ostringstream current_rule_stream;

    for (unsigned short j = 0; j < this->num_configs; j++)
    {
      string current_config = to_binary_str(j, this->num_cells);
      string next_config = to_binary_str(graph[j], this->num_cells);
      string neighborhood = this->get_neighborhood(i, current_config);

      unsigned short neighborhood_num = parse_binary_str(neighborhood);
      char next_state = next_config_str[i];

      if (current_rule_row[neighborhood_num] == 'X')
      {
        current_rule_row[neighborhood_num] = next_state;
      }
      else if (current_rule_row[neighborhood_num] != next_state)
      {
        return false;
      }
    }

    unsigned short num_neighborhoods = current_rule_row.size();

    for (unsigned short j = 0; j < num_neighborhoods; j++)
    {
      char current_char = current_rule_row[num_neighborhoods - 1 - j];
      current_rule_stream << (current_char == 'X' ? '0' : current_char);
    }

    rules[i] = parse_binary_str(current_rule_stream.str());
  }

  return true;
}
```

Listing 3.2: Implementation for extracting rules from a transition diagram

configuration c_x. This means that while processing any permutation H of G, we would require an additional $\mathcal{O}(d^n)$ space to store H. Additionally, we would also require $\mathcal{O}(d^n)$ space (in the worst case, when neighborhood size is equal to n) to store the rule row (a mapping between neighborhood patterns and their resultant states, for a rule) of the cell which is currently being processed for checking the local rule compliance. So the overall space complexity would be $\mathcal{O}(d^n)$.

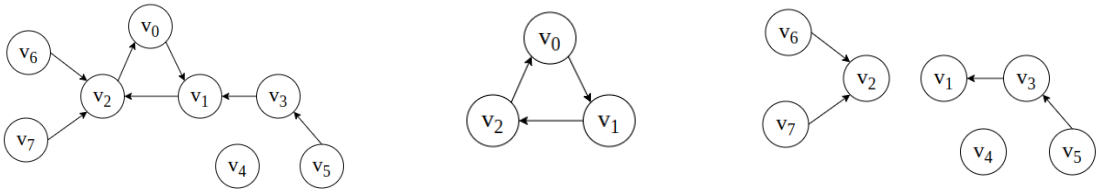
This completes the analysis of exhaustive search. Before exploring the concept of pseudo-isomorphism, we need to define a few graph operations that will be used later to prove some results.

3.2 Graph Operations

In this section we shall define two graph operations, namely *sub-graph removal* and *graph union*, that will be used later to prove the necessary and sufficient conditions for obtaining a pseudo-isomorphic CA by reversing a set of cycles in a given CA.

3.2.1 Sub-Graph Removal

Suppose G' is a sub-graph of G. The graph $G - G'$ is obtained by removing $E(G')$ — the set of all edges of G' — from the graph G and then removing all the nodes of G' with in-degree and out-degree of 0. Let $E_{in}(v)$ be the set of all incoming edges and $E_{out}(v)$ be the set of all outgoing edges of a node v. Then $E(G - G') = E(G) - E(G')$ and $V(G - G') = V(G) - \{v \in G' \mid E_{in}(v) \subseteq E(G')$ and $E_{out}(v) \subseteq E(G')\}$. Figure 3.2 shows an example of this operation.

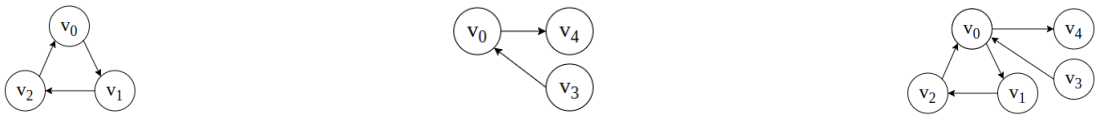


(a) Graph G (b) Graph G' (c) Graph $G - G'$

Figure 3.2: Example of sub-graph removal

3.2.2 Graph Union

Suppose G and H are two labeled graphs. The graph $G \cup H$ contains all the nodes and edges of both G and H, that is, $V(G \cup H) = V(G) \cup V(H)$ and $E(G \cup H) = E(G) \cup E(H)$. Figure 3.3 shows an example of this operation.



(a) Graph G (b) Graph H (c) Graph $G \cup H$

Figure 3.3: Example of graph union

3.3 Pseudo-Isomorphic CAs

When a cycle of some graph G is reversed, the resultant graph H need not be isomorphic to G. While the truth of this statement can be easily verified for generic

graphs, it is also true for transition diagrams, where the out-degree of every node is exactly 1.

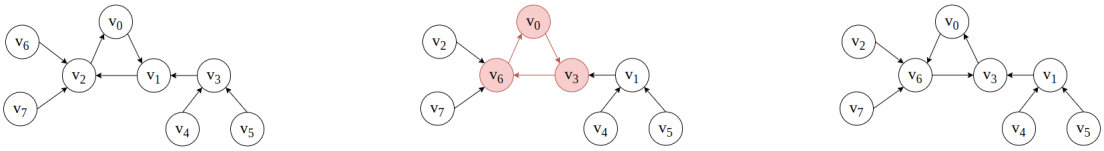
(a) Graph G (b) Graph H

Figure 3.4: Reversing a cycle need not result in an isomorphic graph

Figure 3.4 demonstrates the case where a possible transition diagram G is not isomorphic to H, which is obtained by reversing a cycle of G. Here even though G and H look almost similar, they are not isomorphic because there does not exist any bijection $f : V(G) \rightarrow V(H)$, such that there is an edge from node u to node v in G, if and only if, there is an edge from node $f(u)$ to node $f(v)$ in H. This leads us to the definition of *pseudo-isomorphism* in graphs.

Definition 3.1 (Pseudo-Isomorphic Graphs)**.** *Two graphs G and H are said to be pseudo-isomorphic to each other, if G is isomorphic to H_r, where H_r is obtained by reversing zero or more cycles of H and leaving the other edges of H untouched.*

Figure 3.5 shows an example of pseudo-isomorphic graphs. Here graphs G and H are pseudo-isomorphic to each other, since G is isomorphic to H_r, which is obtained by reversing a cycle of H.



(a) Graph G (b) Graph H_r (c) Graph H

Figure 3.5: Example of pseudo-isomorphic graphs

Lemma 3.1. *Isomorphism implies pseudo-isomorphism, that is, two graphs G and H that are isomorphic to each other, are pseudo-isomorphic to each other as well. But, the converse is not true, that is, two graphs G and H that are pseudo-isomorphic to each other, need not be isomorphic to each other.*

It is obvious that isomorphism implies pseudo-isomorphism. To show that pseudo-isomorphism does not imply isomorphism, it suffices to give an example that demonstrates a case where graphs G and H are pseudo-isomorphic, but not isomorphic to each other. The example graphs given in Figure 3.4 demonstrates exactly the same.

Remark 3.1. From the definition of pseudo-isomorphism, it is clear that whenever a set of cycles of a graph G are reversed, the resulting graph H will be pseudo-isomorphic to G. But, if the reversed cycles are separate components of their own, then the graph H will not only be pseudo-isomorphic, but will also be isomorphic to G.

Having defined pseudo-isomorphism with respect to graphs, we can apply this concept to CAs. Two CAs are said to be pseudo-isomorphic to each other, if their transition diagrams are pseudo-isomorphic to each other. Figure 3.6 shows that ECAs with rule vectors $\langle 51, 105, 4, 132\rangle$ and $\langle 3, 60, 236, 84\rangle$ are pseudo-isomorphic, but not isomorphic under null boundary condition. Similarly, Figure 3.7 shows that ECAs with rule vectors $\langle 95, 18, 5, 111\rangle$ and $\langle 219, 3, 10, 86\rangle$ are pseudo-isomorphic, but not isomorphic under periodic boundary condition.

(a) $\langle 51, 105, 4, 132\rangle$ (b) $\langle 3, 60, 236, 84\rangle$

Figure 3.6: Example of pseudo-isomorphic CAs under null boundary condition

(a) $\langle 95, 18, 5, 111\rangle$ (b) $\langle 219, 3, 10, 86\rangle$

Figure 3.7: Example of pseudo-isomorphic CAs under periodic boundary condition

3.4 State-Neighborhood Relation

To obtain pseudo-isomorphic CAs by reversing cycles, we introduce a new mathematical notion named *state-neighborhood relation* as defined below:

Definition 3.2 (State-Neighborhood Relation)**.** *Consider a CA, with $\mathbb{S}$ being the finite set of states a cell can be in, l being the left radius, r being the right radius and $\mathcal{N}$ being the set of all possible neighborhood patterns a cell can have. Clearly, $\mathcal{N} = \mathbb{S}^{l+r+1}$. The state-neighborhood relation $\mathbb{SN}_i \subseteq \mathbb{S} \times \mathcal{N}$ for a cell i of the CA, is a relation from the set of states $\mathbb{S}$ to the set of neighborhood patterns $\mathcal{N}$, such*

that $(s_j, n_j) \in \mathcal{SN}_i$*, if there exist configurations* $c_1, c_2 \in \mathcal{C}$*, such that* $\mathcal{G}(c_1) = c_2$*, the state of* i *in* c_1 *is* s_j*, and neighborhood of* i *in* c_2 *is* n_j*.*

Remark 3.2. Let G' be a sub-graph of the transition diagram G of a CA. The state-neighborhood relation for a cell i, restricted over G' would be $\mathcal{SN}_i|_{G'} \subseteq \mathcal{SN}_i$, such that $(s_j, n_j) \in \mathcal{SN}_i|_{G'}$, if there exist configurations $c_1, c_2 \in V(G')$, such that $\mathcal{G}(c_1) = c_2$, the state of i in c_1 is s_j, and neighborhood of i in c_2 is n_j. On the same note, $\mathcal{R}_i$ restricted over the sub-graph G' would be

$$\mathcal{R}_i|_{G'}(n_j) = \begin{cases} \mathcal{R}_i(n_j) & \text{if } \exists\ c \in V(G') \text{ such that neighborhood of } i \text{ in } c \text{ is } n_j \\ \textit{undefined} & \text{otherwise} \end{cases}$$

Remark 3.3. $\mathcal{SN}_i|_G = \mathcal{SN}_i$ and $\mathcal{R}_i|_G = \mathcal{R}_i$.

Remark 3.4. $(\mathcal{SN}_i|_{G'})^{-1} = \{(n_j, s_j) \mid (s_j, n_j) \in \mathcal{SN}_i|_{G'}\}$.

Figure 3.8 shows the transition diagram of an ECA with rule vector $\langle 201, 51, 195, 204\rangle$ under periodic boundary condition and the state-neighborhood relation $\mathcal{SN}_0$, for the first cell of the CA. Consider the successive configurations $c_1 = 0000$ and $c_2 = 1110$. The state of first cell in c_1 is 0 and its neighborhood in c_2 is 011. So $(0, 011) \in \mathcal{SN}_0$. The other elements of $\mathcal{SN}_0$ are computed similarly.

(a) $\langle 201, 51, 195, 204\rangle$ (b) $\mathcal{SN}_0$

Figure 3.8: Example of state-neighborhood relation for periodic boundary condition

3.5 Pseudo-Isomorphic CAs by Reversing Cycles

Definition 3.1 says that reversing a set of cycles in the transition diagram G of a CA $\mathcal{CA}_1$ and leaving the remaining edges untouched, results in a pseudo-isomorphic transition diagram H. This transition diagram H may or may not correspond to a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$. If it does, then the CA corresponding to H (say $\mathcal{CA}_2$) is pseudo-isomorphic to $\mathcal{CA}_1$. The transition diagram H corresponds to a CA, if every cell follows a *valid*

local rule over the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$, while updating its state by following the transitions defined in H. A cell is said to follow a *valid* local rule, under a specified neighborhood architecture and boundary condition, if there does not exist any neighborhood pattern n_j under that condition, for which the cell transitions into more than one state.

To check if a cell i follows a valid local rule or not, every node (configuration) of H needs to be visited to obtain the neighborhood of i at that node, along with the state to which i is making a transition. The rule followed by i will be valid, if no collisions are encountered for any neighborhood of i. Assuming k to be the number of cycles in the transition diagram G of the input CA, there will be a total of 2^k possible cycle reversal patterns, out of which one would correspond to G itself. Thus, a maximum of $2^k - 1$ pseudo-isomorphic CAs can be obtained by this reversing cycles algorithm. A flowchart demonstrating the high level overview of this algorithm is presented in Figure 3.9. Listing 3.3 shows a C++ implementation of this algorithm.

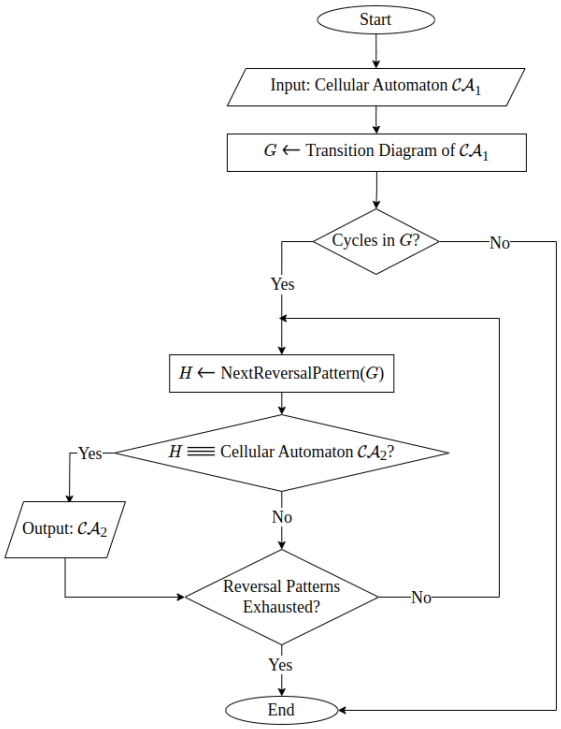


Figure 3.9: Reversing cycles algorithm

It can be seen in Listing 3.3 that the implementation of reversing cycles algorithm uses a method named `get_cycles()` to capture all the cycles in a given transition diagram. Listing 3.4 shows the implementation of this method.

Remark 3.5. The transition diagram G of a reversible CA is composed entirely of disjoint directed cycles. This implies that each cycle is a component of its own.

```
void
binary_1d_ca::print_reversed_pseudoisomorphisms()
{
  vector<set<unsigned short>> cycles = get_cycles(this->graph);

  if (cycles.size() == 0)
  {
    print("No cycles to reverse");
    return;
  }

  for (long i = 0; i < (1UL << cycles.size()); i++)
  {
    vector<unsigned short> current_graph(this->num_configs, USHRT_MAX);
    vector<unsigned short> current_rules(this->num_cells, 0);

    long current_index = i;
    short current_cycle = 0;
    unsigned long index_mask = (1UL << cycles.size()) - 1;

    while (index_mask)
    {
      if (current_index & 1)
      {
        for (const auto &config : cycles[current_cycle])
        {
          current_graph[this->graph[config]] = config;
        }
      }
      else
      {
        for (const auto &config : cycles[current_cycle])
        {
          current_graph[config] = this->graph[config];
        }
      }

      current_index >>= 1;
      index_mask >>= 1;
      current_cycle += 1;
    }

    for (unsigned short i = 0; i < current_graph.size(); i++)
    {
      if (current_graph[i] == USHRT_MAX)
      {
        current_graph[i] = this->graph[i];
      }
    }

    if (this->extract_rules(current_graph, current_rules))
    {
      print(current_rules);
    }
  }
}

```

Listing 3.3: Implementation of reversing cycles algorithm

So the transition diagram H, obtained by reversing a set of cycles of G, will be isomorphic to G. If H corresponds to a CA, then the CA corresponding to H will be isomorphic to the CA corresponding to G.

The immediate question that follows is, *when can a selected set of cycles in the transition diagram of a CA be reversed, so as to obtain a pseudo-isomorphic*

```
vector<set<unsigned short>>
get_cycles(vector<unsigned short> &graph)
{
  unsigned short current_node = 0;
  set<unsigned short> visited_nodes;
  set<unsigned short> cycled_nodes;
  vector<set<unsigned short>> cycles;

  for (unsigned short i = 0; i < graph.size(); i++)
  {
    while (true)
    {
      if (cycled_nodes.find(current_node) != cycled_nodes.end())
      {
        current_node = i + 1;
        break;
      }

      if (visited_nodes.find(current_node) != visited_nodes.end())
      {
        set<unsigned short> current_cycle;
        unsigned short start_node = current_node;
        bool is_cycle = true;
        current_cycle.insert(start_node);

        while ((current_node = graph[current_node]) != start_node)
        {
          if (current_node in cycled_nodes)
          {
            is_cycle = false;
            break;
          }

          current_cycle.insert(current_node);
        }

        if (!is_cycle)
        {
          current_node = i + 1;
          break;
        }

        cycles.push_back(current_cycle);

        for (const auto &node : current_cycle)
        {
          cycled_nodes.insert(node);
        }
      }

      visited_nodes.insert(current_node);
      current_node = graph[current_node];
    }
  }

  return cycles;
}

```

Listing 3.4: Implementation for capturing cycles in a transition diagram

CA? This can be answered by the following theorem, which uses the concept of state-neighborhood relation, introduced in the earlier section.

Theorem 3.1. *Consider a CA* $\mathcal{CA}_1$ *with* n *cells, having rule vector* $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1} \rangle$. *Let* G *be its transition diagram. Let* $\mathcal{SN}_i$ *be the state-neighborhood relation of the cell* i *and* $\mathcal{SN}_i|_{G'}$ *be the restriction of* $\mathcal{SN}_i$

over a set of cycles G' of G. Let $\mathcal{R}_i|_{G-G'}$ be the restriction of $\mathcal{R}_i$ over the sub-graph $G - G'$. The set of cycles G' can be reversed to produce a pseudo-isomorphic CA (say $\mathcal{CA}_2$) having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$, if and only if, for each cell i, both the following conditions are satisfied:

1. *$\mathcal{SN}_i|_{G'}$ is one-to-one or one-to-many, that is, there does not exist any neighborhood pattern n_j, such that $(s_a, n_j) \in \mathcal{SN}_i|_{G'}$ and $(s_b, n_j) \in \mathcal{SN}_i|_{G'}$, for states $s_a \neq s_b$.*

2. *$(\mathcal{SN}_i|_{G'})^{-1}$ and $\mathcal{R}_i|_{G-G'}$ do not have any conflicts, that is, there does not exist any neighborhood pattern n_j, such that $\mathcal{R}_i|_{G-G'}(n_j)$ is defined and $(n_j, s_a) \in (\mathcal{SN}_i|_{G'})^{-1}$, for $s_a \neq \mathcal{R}_i|_{G-G'}(n_j)$.*

Proof. Let G'_r be the sub-graph that results from reversing G'. The transition diagram H obtained by reversing G' would be $H = (G - G') \cup G'_r$.

If: Consider that both the conditions stated in the theorem hold true. It needs to be shown that H would correspond to a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$.

The local rule for cell i to transition into a new state within the sub-graph G'_r would be given by $(\mathcal{SN}_i|_{G'})^{-1}$. Since $\mathcal{SN}_i|_{G'}$ is one-to-one or one-to-many, its inverse, $(\mathcal{SN}_i|_{G'})^{-1}$, is one-to-one or many-to-one. Since $(\mathcal{SN}_i|_{G'})^{-1}$ is a one-to-one or many-to-one relation, no neighborhood will be mapped to more than one state by $(\mathcal{SN}_i|_{G'})^{-1}$, making it a valid local rule for a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$, within the sub-graph G'_r.

Similarly, the local rule for cell i within the sub-graph $G - G'$ would be given by $\mathcal{R}_i|_{G-G'}$. Since $\mathcal{R}_i$ is a valid local rule for $\mathcal{CA}_1$, $\mathcal{R}_i|_{G-G'}$ is a valid local rule for a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$, within the sub-graph $G - G'$. Therefore, the local rule followed by cell i, to transition into a new state within the graph $H = (G - G') \cup G'_r$ is given by

$$\mathcal{R}'_i(n_j) = \begin{cases} (\mathcal{SN}_i|_{G'})^{-1}(n_j) & \text{within the sub-graph } G'_r \\ \mathcal{R}_i|_{G-G'}(n_j) & \text{within the sub-graph } G - G' \end{cases}$$

Since there is no conflict between $(\mathcal{SN}_i|_{G'})^{-1}$ and $\mathcal{R}_i|_{G-G'}$, there does not exist any neighborhood pattern n_j, such that $\mathcal{R}_i|_{G-G'}(n_j)$ is defined and $(n_j, s_a) \in (\mathcal{SN}_i|_{G'})^{-1}$, for $s_a \neq \mathcal{R}_i|_{G-G'}(n_j)$. So $\mathcal{R}'_i$ is a valid local rule for a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$, within the graph $H = (G - G') \cup G'_r$. Hence, cell i does not violate the local rule property within the graph H. Hence, if both the above stated conditions hold true for every

cell of the CA $\mathcal{CA}_1$, then H would emulate a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$.

Only If: Consider that any one of the conditions stated in the theorem does not hold true for a cell i of $\mathcal{CA}_1$.

Suppose the first condition does not hold, that is, $\mathcal{SN}_i|_{G'}$ is neither one-to-one nor one-to-many. This implies that $\mathcal{SN}_i|_{G'}$ is many-to-one or many-to-many. So its inverse, $(\mathcal{SN}_i|_{G'})^{-1}$, is one-to-many or many-to-many. The local rule followed by cell i to transition into a new state within the sub-graph G'_r would be given by $(\mathcal{SN}_i|_{G'})^{-1}$. Since $(\mathcal{SN}_i|_{G'})^{-1}$ is one-to-many or many-to-many, there exists a neighborhood pattern n_j, such that $(n_j, s_a) \in (\mathcal{SN}_i|_{G'})^{-1}$ and $(n_j, s_b) \in (\mathcal{SN}_i|_{G'})^{-1}$, for states $s_a \neq s_b$. So $(\mathcal{SN}_i|_{G'})^{-1}$ is not a valid local rule for a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$, within the sub-graph G'_r. Hence, clearly cell i violates the local rule property in the graph $H = (G - G') \cup G'_r$.

Suppose the first condition holds, but the second one does not, that is, $(\mathcal{SN}_i|_{G'})^{-1}$ and $\mathcal{R}_i|_{G-G'}$ have at least one conflict. The local rule followed by cell i to transition into a new state within the graph $H = (G - G') \cup G'_r$ is given by $\mathcal{R}'_i$, as defined previously. Since there is at least one conflict between $(\mathcal{SN}_i|_{G'})^{-1}$ and $\mathcal{R}_i|_{G-G'}$, there exists a neighborhood pattern n_j, such that $\mathcal{R}_i|_{G-G'}(n_j)$ is defined and $(n_j, s_a) \in (\mathcal{SN}_i|_{G'})^{-1}$, for $s_a \neq \mathcal{R}_i|_{G-G'}(n_j)$. So $\mathcal{R}'_i$ is not a valid local rule for a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$ within the graph $H = (G - G') \cup G'_r$.

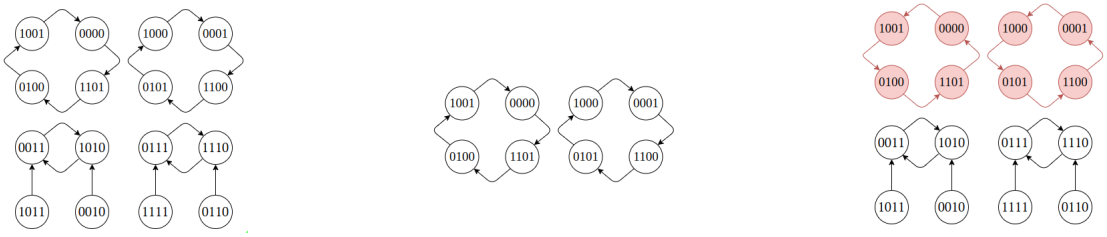
So if any of the above stated conditions does not hold true for any cell of $\mathcal{CA}_1$, then H would not emulate a CA having the same neighborhood architecture and boundary condition as that of $\mathcal{CA}_1$. □

Corollary 3.1. *By reversing the entire transition diagram G of a reversible CA, an isomorphic CA having the same neighborhood architecture and boundary condition can be obtained, if and only if the state-neighborhood relation $\mathcal{SN}_i$ for every cell i, is either one-to-one or one-to-many.*

Remark 3.6. By reversing cycles of length less than 3, the obtained transition diagram H is exactly same as the transition diagram G of the input CA. So H would correspond to the input CA itself. A CA is said to have *non-trivial reversed cycle pseudo-isomorphisms*, if a pseudo-isomorphic CA having the same neighborhood architecture and boundary condition can be obtained by reversing one or more cycles of length ≥ 3.

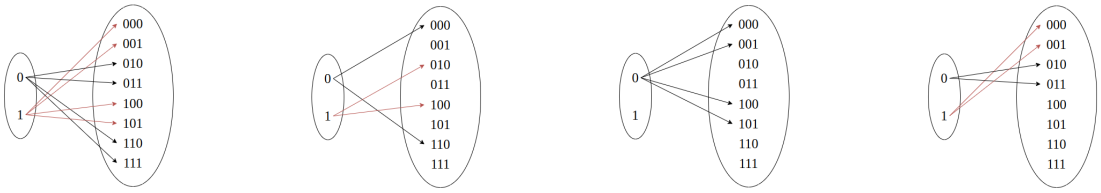
Example 3.1. Consider an ECA with rule vector $\mathcal{R} = \langle 51, 201, 204, 163 \rangle$ under periodic boundary condition. As per the notations used in the above theorem, G is its transition diagram, G' is a sub-graph of selected set of cycles, G'_r is the

sub-graph obtained by reversing G' and H is $(G - G') \cup G'_r$. Figure 3.10 shows the graphs G, G' and H. Similarly, $\mathcal{SN}_i|_{G'}$, the state-neighborhood relation for each cell i, restricted over the set of cycles G' is shown in Figure 3.11. It can be observed that for all i, $\mathcal{SN}_i|_{G'}$ is one-to-many and $(\mathcal{SN}_i|_{G'})^{-1}$ does not have any conflicts with $\mathcal{R}_i|_{G-G'}$. Since both the conditions stated in the theorem are true, the transition diagram $H = (G - G') \cup G'_r$ corresponds to an ECA with rule vector $\mathcal{R}' = \langle 51, 156, 204, 163 \rangle$ under periodic boundary condition. Each rule $\mathcal{R}'_i$ is computed as per the formula mentioned in the proof of the theorem.



(a) Graph G (b) Graph G' (c) Graph H

Figure 3.10: G, G' and H of $\langle 51, 201, 204, 163 \rangle$ under periodic boundary condition



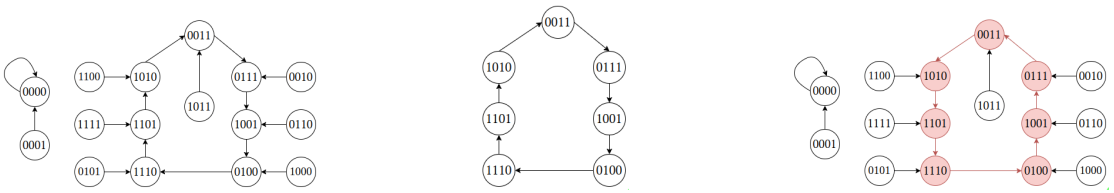
(a) $\mathcal{SN}_0|_{G'}$ (b) $\mathcal{SN}_1|_{G'}$ (c) $\mathcal{SN}_2|_{G'}$ (d) $\mathcal{SN}_3|_{G'}$

Figure 3.11: $\mathcal{SN}_i|_{G'}$ for $i = 0$ to 3 of $\langle 51, 201, 204, 163 \rangle$ under periodic boundary condition

Example 3.2. Consider an ECA with rule vector $\mathcal{R} = \langle 90, 150, 60, 90 \rangle$ under null boundary condition. Its transition diagram G, a set of cycles G' of G and H computed as $(G - G') \cup G'_r$ (where G'_r is the sub-graph obtained by reversing G') are shown in Figure 3.12. Similarly, $\mathcal{SN}_i|_{G'}$, the state-neighborhood relation for each cell i, restricted over the set of cycles G' is shown in Figure 3.13. It can be observed that $\mathcal{SN}_0|_{G'}$ and $\mathcal{SN}_3|_{G'}$ are many-to-many relations, thereby violating the first condition of the theorem. As a result, the transition diagram $H = (G - G') \cup G'_r$ does not correspond to any ECA under null boundary condition.

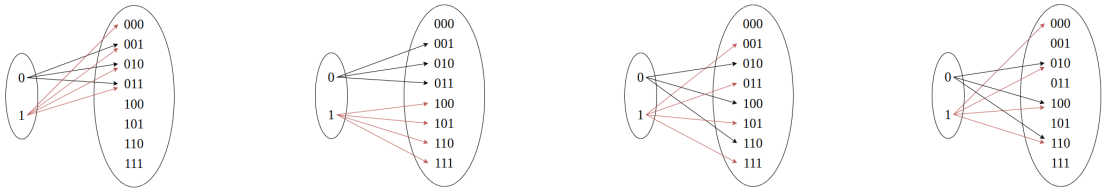
3.5.1 Time Complexity of Reversing Cycles Algorithm

For a d-state 1D CA of size n, there will be d^n nodes in its transition diagram G. By using depth first search (DFS) algorithm, the time complexity of capturing cycles in the transition diagram would be proportional to the number of nodes in



(a) Graph G (b) Graph G' (c) Graph H

Figure 3.12: G, G' and H of $\langle 90, 150, 60, 90\rangle$ under null boundary condition



(a) $\mathcal{SN}_0|_{G'}$ (b) $\mathcal{SN}_1|_{G'}$ (c) $\mathcal{SN}_2|_{G'}$ (d) $\mathcal{SN}_3|_{G'}$

Figure 3.13: $\mathcal{SN}_i|_{G'}$ for $i = 0$ to 3 of $\langle 90, 150, 60, 90\rangle$ under null boundary condition

the graph, that is, $\mathcal{O}(d^n)$[1]. Generating a graph H that would be pseudo-isomorphic to G, by reversing a combination of cycles in G, would take time proportional to the number of nodes in G, that is, $\mathcal{O}(d^n)$. The time complexity of checking if H would correspond to a CA having the same neighborhood architecture and boundary condition as that of the given CA would be $\mathcal{O}(n^2 * d^n)$, as discussed in the analysis of exhaustive search algorithm. So the total time complexity of generating pseudo-isomorphic CAs using reversing cycles algorithm would be $\mathcal{O}(2^k * (n^2 * d^n))$.

3.5.2 Space Complexity of Reversing Cycles Algorithm

A space proportional to the number of nodes in the transition diagram G, that is, $\mathcal{O}(d^n)$, is needed to store the cycles in G. Similarly, $\mathcal{O}(d^n)$ space is needed to store H, the permutation of G that is being currently checked if it would correspond to a CA having the same neighborhood architecture and boundary condition as the given CA. As discussed in the analysis of exhaustive search algorithm, this operation requires a space of $\mathcal{O}(d^n)$. So the total space complexity of generating pseudo-isomorphisms by reversing cycles would be $\mathcal{O}(d^n)$.

[1]For a graph $G = (V, E)$, the time complexity of DFS is $\mathcal{O}(|V| + |E|)$. Since the number of edges in a transition diagram is equal to the number of nodes in it, the time complexity of DFS in such a case would be $\mathcal{O}(|V|)$.

CHAPTER 4

RESULTS AND DISCUSSION

All the algorithms described in the previous chapter have been implemented using C++ and tested using GCC 12.3.0 compiler, on a Linux (Ubuntu 22.04) machine, running on an Intel i5 (12 th generation) processor containing 16 cores and having a RAM of 16 GB. The source code for a sample application demonstrating these algorithms is placed in the Github repository, `https://github.com/gilfoyle-bertram/CellularAutomata.git`. Guidelines needed for compiling and executing the files have been included in the repository itself. This chapter discusses the results obtained by implementing the algorithms described in the previous chapter.

4.1 Results of Exhaustive Search Algorithm

On observing the results for multiple 3-cell ECAs, it is found that for a 3-cell ECA under periodic boundary condition, all permutations of the transition diagram correspond to an ECA under the same boundary condition. The reason for this behavior is, for a 3-cell ECA under periodic boundary condition, every cell depends on every other cell to make a transition to the next state. So in any permutation of the transition diagram, a particular neighborhood pattern occurs exactly once for every cell. So there is no way for a cell to violate the local rule property, in any permutation of the transition diagram. This automatically implies that every permutation of the transition diagram will correspond to an ECA under the same boundary condition. This result can be generalized as below.

Remark 4.1. For a CA to have the maximum number of isomorphic CAs, it should have the neighborhood architecture and boundary condition in such a way that every cell depends on every other cell to make a transition to the next state. In such a case, every permutation of the transition diagram will correspond to an isomorphic CA having the same neighborhood architecture and boundary condition.

4.2 Results of Rule Complementation

As mentioned in Remark 2.1, for an n-cell reversible linear ECA not having a factor of $(x+1)$ in its characteristic polynomial, all the $2^n - 1$ complemented ECAs are isomorphic to it.

Tables 4.1, 4.2, 4.3 and 4.4 list a few reversible linear ECAs of sizes $3, 4, 5$ and 6, respectively, under null boundary condition, that do not have a factor of $(x + 1)$ in their characteristic polynomials. Similarly, Tables 4.5, 4.6, 4.7 and 4.8 list a few reversible linear ECAs of sizes $3, 4, 5$ and 6, respectively, under periodic boundary condition, that do not have a factor of $(x + 1)$ in their characteristic polynomials.

Table 4.9 shows the number of reversible linear ECAs of sizes $3, 4, 5$ and 6, under null and periodic boundary conditions, not having a factor of $(x + 1)$ in their characteristic polynomials.

Table 4.1: Some 3-cell reversible linear ECAs under null boundary condition not having a factor of $(x + 1)$ in their characteristic polynomials

ECA	**Characteristic Polynomial**
$\langle 150, 90, 90\rangle$	$x^3 + x^2 + 1$
$\langle 102, 90, 240\rangle$	$x^3 + x^2 + 1$
$\langle 90, 150, 60\rangle$	$x^3 + x + 1$
$\langle 150, 90, 240\rangle$	$x^3 + x^2 + 1$
$\langle 90, 90, 60\rangle$	$x^3 + x^2 + 1$
$\langle 150, 150, 90\rangle$	$x^3 + x + 1$
$\langle 102, 150, 240\rangle$	$x^3 + x + 1$
$\langle 170, 90, 150\rangle$	$x^3 + x^2 + 1$
$\langle 102, 90, 90\rangle$	$x^3 + x^2 + 1$
$\langle 150, 150, 240\rangle$	$x^3 + x + 1$

Table 4.2: Some 4-cell reversible linear ECAs under null boundary condition not having a factor of $(x + 1)$ in their characteristic polynomials

ECA	**Characteristic Polynomial**
$\langle 150, 240, 170, 60\rangle$	$x^4 + x^2 + 1$
$\langle 102, 90, 150, 90\rangle$	$x^4 + x + 1$
$\langle 170, 150, 90, 90\rangle$	$x^4 + x^3 + x^2 + x + 1$
$\langle 150, 150, 90, 150\rangle$	$x^4 + x^3 + 1$
$\langle 150, 240, 102, 240\rangle$	$x^4 + x^2 + 1$
$\langle 90, 150, 90, 150\rangle$	$x^4 + x + 1$
$\langle 90, 90, 90, 240\rangle$	$x^4 + x^2 + 1$
$\langle 150, 240, 90, 60\rangle$	$x^4 + x^2 + 1$
$\langle 90, 90, 150, 240\rangle$	$x^4 + x^3 + x^2 + x + 1$
$\langle 102, 90, 102, 90\rangle$	$x^4 + x^2 + 1$

4.3 Results of Reversing Cycles Algorithm

The algorithm for reversing cycles is used to check the number of ECAs having non-trivial reversed cycle pseudo-isomorphisms over reversible ECAs under null boundary condition. To test the algorithm, the synthesis scheme described in Ref. [5]

Table 4.3: Some 5-cell reversible linear ECAs under null boundary condition not having a factor of $(x+1)$ in their characteristic polynomials

ECA	Characteristic Polynomial
$\langle 102, 90, 102, 90, 240\rangle$	x^5+x+1
$\langle 102, 90, 102, 150, 90\rangle$	x^5+x^4+1
$\langle 170, 150, 150, 150, 150\rangle$	x^5+x^2+1
$\langle 170, 150, 60, 150, 90\rangle$	x^5+x^4+1
$\langle 150, 90, 90, 150, 60\rangle$	$x^5+x^4+x^3+x^2+1$
$\langle 170, 60, 102, 150, 90\rangle$	x^5+x^4+1
$\langle 170, 90, 90, 150, 150\rangle$	$x^5+x^3+x^2+x+1$
$\langle 90, 150, 150, 90, 90\rangle$	x^5+x^3+1
$\langle 90, 90, 150, 102, 240\rangle$	x^5+x+1
$\langle 102, 90, 90, 150, 60\rangle$	$x^5+x^4+x^3+x^2+1$

Table 4.4: Some 6-cell reversible linear ECAs under null boundary condition not having a factor of $(x+1)$ in their characteristic polynomials

ECA	Characteristic Polynomial
$\langle 150, 240, 150, 240, 170, 60\rangle$	$x^6+x^5+x^3+x+1$
$\langle 150, 150, 150, 150, 150, 240\rangle$	$x^6+x^5+x^4+x^2+1$
$\langle 102, 90, 102, 90, 150, 90\rangle$	$x^6+x^5+x^4+x^3+1$
$\langle 102, 90, 150, 150, 90, 60\rangle$	x^6+x^4+1
$\langle 150, 150, 240, 90, 90, 60\rangle$	$x^6+x^5+x^4+x^3+x^2+x+1$
$\langle 150, 150, 240, 90, 150, 60\rangle$	x^6+x^2+1
$\langle 102, 90, 170, 150, 90, 90\rangle$	$x^6+x^4+x^3+x^2+1$
$\langle 170, 60, 102, 90, 102, 90\rangle$	$x^6+x^5+x^3+x+1$
$\langle 170, 60, 170, 90, 150, 240\rangle$	$x^6+x^4+x^3+x^2+1$
$\langle 90, 60, 102, 90, 102, 90\rangle$	$x^6+x^5+x^3+x+1$

Table 4.5: Some 3-cell reversible linear ECAs under periodic boundary condition not having a factor of $(x+1)$ in their characteristic polynomials

ECA	Characteristic Polynomial
$\langle 60, 60, 150\rangle$	x^3+x^2+1
$\langle 60, 60, 240\rangle$	x^3+x+1
$\langle 170, 150, 60\rangle$	x^3+x+1
$\langle 170, 150, 90\rangle$	x^3+x^2+1
$\langle 90, 170, 170\rangle$	x^3+x+1
$\langle 240, 240, 90\rangle$	x^3+x+1
$\langle 90, 170, 150\rangle$	x^3+x^2+1
$\langle 102, 90, 150\rangle$	x^3+x+1
$\langle 150, 60, 60\rangle$	x^3+x^2+1
$\langle 240, 102, 150\rangle$	x^3+x+1

is used to generate 1000 random reversible ECAs for $n \in \{3, 4, 5\}$. In that experiment, for ECAs of size 3, on an average, 18% of the generated reversible ECAs are found to contain non-trivial reversed cycle pseudo-isomorphisms. But the per-

Table 4.6: Some 4-cell reversible linear ECAs under periodic boundary condition not having a factor of $(x+1)$ in their characteristic polynomials

ECA	Characteristic Polynomial
$\langle 60, 60, 60, 150\rangle$	x^4+x^2+1
$\langle 60, 60, 60, 240\rangle$	$x^4+x^3+x^2+x+1$
$\langle 102, 90, 102, 170\rangle$	x^4+x+1
$\langle 150, 240, 170, 60\rangle$	x^4+x^2+1
$\langle 170, 102, 170, 150\rangle$	x^4+x+1
$\langle 240, 60, 240, 90\rangle$	$x^4+x^3+x^2+x+1$
$\langle 240, 150, 240, 60\rangle$	x^4+x+1
$\langle 240, 60, 240, 60\rangle$	x^4+x^2+1
$\langle 150, 150, 90, 240\rangle$	x^4+x+1
$\langle 170, 102, 170, 102\rangle$	x^4+x^2+1

Table 4.7: Some 5-cell reversible linear ECAs under periodic boundary condition not having a factor of $(x+1)$ in their characteristic polynomials

ECA	Characteristic Polynomial
$\langle 60, 60, 60, 60, 150\rangle$	$x^5+x^4+x^3+x^2+1$
$\langle 170, 102, 170, 150, 90\rangle$	x^5+x^3+1
$\langle 170, 102, 170, 150, 102\rangle$	$x^5+x^4+x^2+x+1$
$\langle 60, 60, 60, 60, 240\rangle$	x^5+x+1
$\langle 170, 102, 170, 150, 170\rangle$	x^5+x^2+1
$\langle 240, 240, 240, 240, 90\rangle$	x^5+x^3+1
$\langle 240, 60, 240, 60, 60\rangle$	$x^5+x^4+x^3+x^2+1$
$\langle 60, 102, 240, 170, 90\rangle$	x^5+x+1
$\langle 240, 240, 240, 60, 60\rangle$	x^5+x^3+1
$\langle 102, 170, 102, 102, 170\rangle$	$x^5+x^4+x^3+x^2+1$

Table 4.8: Some 6-cell reversible linear ECAs under periodic boundary condition not having a factor of $(x+1)$ in their characteristic polynomials

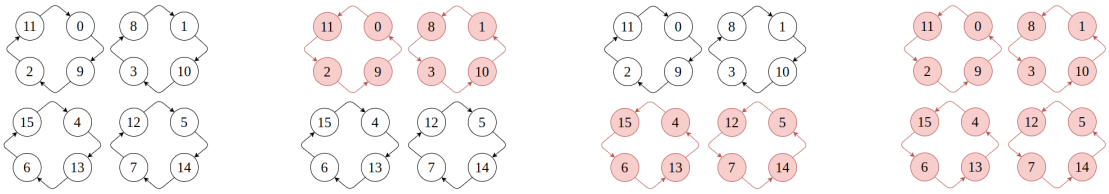
ECA	Characteristic Polynomial
$\langle 60, 240, 60, 240, 90, 60\rangle$	$x^6+x^5+x^3+x^2+1$
$\langle 60, 60, 60, 60, 60, 150\rangle$	x^6+x^2+1
$\langle 60, 150, 60, 150, 90, 60\rangle$	$x^6+x^5+x^4+x^3+1$
$\langle 102, 90, 102, 90, 102, 170\rangle$	$x^6+x^5+x^4+x^3+x^2+x+1$
$\langle 102, 170, 102, 170, 150, 90\rangle$	$x^6+x^5+x^4+x^3+1$
$\langle 240, 60, 240, 60, 240, 240\rangle$	x^6+x^4+1
$\langle 102, 90, 102, 90, 150, 170\rangle$	$x^6+x^5+x^3+x^2+1$
$\langle 102, 170, 102, 170, 170, 90\rangle$	x^6+x^2+1
$\langle 150, 240, 150, 240, 240, 90\rangle$	x^6+x^3+1
$\langle 240, 150, 240, 170, 150, 102\rangle$	$x^6+x^5+x^3+x+1$

centage quickly drops to 7% for ECAs of size 4 and to 2% for ECAs of size 5. Since these ECAs are reversible, the ECAs obtained by reversing cycles are not just pseudo-isomorphic, but are also isomorphic to the original ECA.

Table 4.9: No. of reversible linear ECAs not having a factor of $(x+1)$ in their characteristic polynomials

ECA Size	Null Boundary Condition	Periodic Boundary Condition
3	16	48
4	80	272
5	240	1240
6	976	5960

Figure 4.1 shows an example of a reversible ECA with rule vector $\langle 3, 204, 102, 17\rangle$ under null boundary condition. Isomorphic ECAs with rule vectors $\langle 3, 204, 105, 17\rangle$ and $\langle 3, 204, 150, 17\rangle$ under null boundary condition are obtained by reversing proper sub-graphs of the transition diagram, while an isomorphic ECA with rule vector $\langle 3, 204, 153, 17\rangle$ under null boundary condition is obtained by reversing the entire transition diagram itself. It is known that for an ECA with rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1}\rangle$ under null boundary condition, an ECA with rule vector $\mathcal{R}^{ref} = \langle \mathcal{R}^{ref}_{n-1}, \mathcal{R}^{ref}_{n-2}, \ldots, \mathcal{R}^{ref}_0\rangle$ is isomorphic to it, where $\mathcal{R}^{ref}_i$ refers to the reflection transformation [7] of $\mathcal{R}_i$. The ECAs obtained by reversing cycles may or may not include the one with rule vector $\mathcal{R}^{ref}$. For $\mathcal{R} = \langle 3, 204, 102, 17\rangle$, $\mathcal{R}^{ref} = \langle 3, 60, 204, 17\rangle$. In this case, none of the isomorphic ECAs obtained by reversing cycles corresponds to an ECA with rule vector $\mathcal{R}^{ref}$.

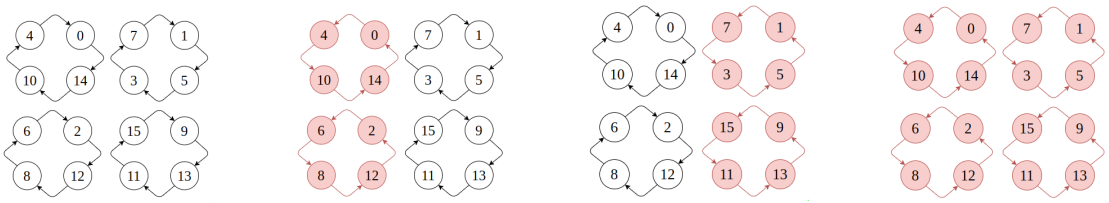


(a) $\langle 3, 204, 102, 17\rangle$ (b) $\langle 3, 204, 105, 17\rangle$ (c) $\langle 3, 204, 150, 17\rangle$ (d) $\langle 3, 204, 153, 17\rangle$

Figure 4.1: Example of reversing cycles under null boundary condition

Figure 4.2 shows an example of a reversible ECA with rule vector $\langle 201, 51, 105, 204\rangle$ under periodic boundary condition. Isomorphic ECAs with rule vectors $\langle 198, 51, 60, 204\rangle$ and $\langle 201, 51, 195, 204\rangle$ under periodic boundary condition are obtained by reversing proper sub-graphs of the transition diagram, while an isomorphic ECA with rule vector $\langle 198, 51, 150, 204\rangle$ under periodic boundary condition is obtained by reversing the entire transition diagram itself. It is known that for an ECA with rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1}\rangle$ under periodic boundary condition, ECAs with rule vectors $\mathcal{R}^{conj} = \langle \mathcal{R}^{conj}_0, \mathcal{R}^{conj}_1, \ldots, \mathcal{R}^{conj}_{n-1}\rangle$, $\mathcal{R}^{ref} = \langle \mathcal{R}^{ref}_{n-1}, \mathcal{R}^{ref}_{n-2}, \ldots, \mathcal{R}^{ref}_0\rangle$ and $\mathcal{R}^{c.r.} = \langle \mathcal{R}^{c.r.}_{n-1}, \mathcal{R}^{c.r.}_{n-2}, \ldots, \mathcal{R}^{c.r.}_0\rangle$ are isomorphic to it, where $\mathcal{R}^{conj}_i$, $\mathcal{R}^{ref}_i$ and $\mathcal{R}^{c.r.}_i$ refer to the conjugation, reflection and (conjugation + reflection) transformations [7] of $\mathcal{R}_i$, respectively. The ECAs obtained by reversing cycles may

or may not include those with rule vectors $\mathcal{R}^{conj}$, $\mathcal{R}^{ref}$ and $\mathcal{R}^{c.r.}$. For $\mathcal{R} = \langle 201, 51, 105, 204\rangle$, $\mathcal{R}^{conj} = \langle 108, 51, 105, 204\rangle$, $\mathcal{R}^{ref} = \langle 204, 105, 51, 201\rangle$ and $\mathcal{R}^{c.r.} = \langle 204, 105, 51, 108\rangle$. In this case, none of the isomorphic ECAs obtained by reversing cycles corresponds to those with rule vectors $\mathcal{R}^{conj}$, $\mathcal{R}^{ref}$ or $\mathcal{R}^{c.r.}$.



(a) $\langle 201, 51, 105, 204\rangle$ (b) $\langle 198, 51, 60, 204\rangle$ (c) $\langle 201, 51, 195, 204\rangle$ (d) $\langle 198, 51, 150, 204\rangle$

Figure 4.2: Example of reversing cycles under periodic boundary condition

The synthesis scheme described in Ref. [6] is used for generating reversible ECAs under periodic boundary condition. However not all the ECAs obtained through the scheme are found to be reversible. Out of the obtained reversible ECAs of sizes 3 and 4, under null and periodic boundary conditions, it has been observed that whenever an isomorphic ECA exists by reversing a proper sub-graph G' of the transition diagram G (that is, $G' \neq G$), an isomorphic ECA also exists by reversing G as well. Figures 4.1 and 4.2 serve as examples for this statement. However this is not always the case and a counter example has been found for the 5-cell reversible ECA with rule vector $\langle 5, 165, 204, 102, 17\rangle$ under null boundary condition. For this ECA, reversing a proper sub-graph of the transition diagram results in an isomorphic ECA with rule vector $\langle 5, 165, 204, 105, 17\rangle$, but reversing the entire transition diagram does not result in an isomorphic ECA under null boundary condition. Figure 4.3 demonstrates this example. Similarly, under periodic boundary condition, the reversible ECA with rule vector $\langle 51, 108, 204, 58, 105\rangle$ acts as a counter example. For this ECA, reversing a proper sub-graph of the transition diagram results in an isomorphic ECA with rule vector $\langle 51, 198, 204, 58, 105\rangle$, but reversing the entire transition diagram does not result in an isomorphic ECA under periodic boundary condition. Figure 4.4 demonstrates this example.

(a) $\langle 5, 165, 204, 102, 17\rangle$ (b) $\langle 5, 165, 204, 105, 17\rangle$

Figure 4.3: Example of reversing cycles under null boundary condition

Applying the reversing cycles algorithm on uniform ECAs of sizes 4 & 5 under null boundary condition, it is found that none of them have non-trivial reversed

(a) $\langle 51, 108, 204, 58, 105 \rangle$ (b) $\langle 51, 198, 204, 58, 105 \rangle$

Figure 4.4: Example of reversing cycles under periodic boundary condition

cycle pseudo-isomorphisms. However this is not the case with uniform ECAs under periodic boundary condition. Tables 4.10 and 4.11 list the uniform ECAs of sizes 4 & 5 respectively, that have non-trivial reversed cycle pseudo-isomorphisms under periodic boundary condition.

Remark 4.2. On observing the transition diagrams of uniform ECAs with rule vectors $\langle 2, 2, 2, 2 \rangle$ and $\langle 16, 16, 16, 16 \rangle$ under periodic boundary condition, it has been found that they are pseudo-isomorphic as well. But they are not listed in Table 4.10 because one cannot be obtained from the other by just reversing a set of cycles and leaving the remaining edges untouched. As shown in Figure 4.5, along with reversing a set of cycles, some non-cyclic edges need to be modified as well. Hence these cases are not covered by the reversing cycles algorithm.

(a) $\langle 2, 2, 2, 2 \rangle$ (b) $\langle 16, 16, 16, 16 \rangle$

Figure 4.5: Pseudo-isomorphic CAs under periodic boundary condition that cannot be obtained by just reversing a set of cycles and leaving the remaining edges untouched

Table 4.10: List of 4-cell uniform ECAs having non-trivial reversed cycle pseudo-isomorphisms under periodic boundary condition

Input CA	**Pseudo-Isomorphic CA**
$\langle 3, 3, 3, 3\rangle$	$\langle 17, 17, 17, 17\rangle$
$\langle 15, 15, 15, 15\rangle$	$\langle 85, 85, 85, 85\rangle$
$\langle 35, 35, 35, 35\rangle$	$\langle 49, 49, 49, 49\rangle$
$\langle 49, 49, 49, 49\rangle$	$\langle 35, 35, 35, 35\rangle$
$\langle 58, 58, 58, 58\rangle$	$\langle 114, 114, 114, 114\rangle$
$\langle 59, 59, 59, 59\rangle$	$\langle 115, 115, 115, 115\rangle$
$\langle 62, 62, 62, 62\rangle$	$\langle 118, 118, 118, 118\rangle$
$\langle 63, 63, 63, 63\rangle$	$\langle 119, 119, 119, 119\rangle$
$\langle 131, 131, 131, 131\rangle$	$\langle 145, 145, 145, 145\rangle$
$\langle 163, 163, 163, 163\rangle$	$\langle 177, 177, 177, 177\rangle$
$\langle 170, 170, 170, 170\rangle$	$\langle 240, 240, 240, 240\rangle$

Table 4.11: List of 5-cell uniform ECAs having non-trivial reversed cycle pseudo-isomorphisms under periodic boundary condition

Input CA	**Pseudo-Isomorphic CA**
$\langle 15, 15, 15, 15, 15\rangle$	$\langle 85, 85, 85, 85, 85\rangle$
$\langle 35, 35, 35, 35, 35\rangle$	$\langle 49, 49, 49, 49, 49\rangle$
$\langle 59, 59, 59, 59, 59\rangle$	$\langle 115, 115, 115, 115, 115\rangle$
$\langle 170, 170, 170, 170, 170\rangle$	$\langle 240, 240, 240, 240, 240\rangle$

Table 4.12: Some 4-cell non-uniform ECAs having non-trivial reversed cycle pseudo-isomorphisms under null boundary condition

Input CA	**Reversible**	**Pseudo-Isomorphic CA(s)**
$\langle 220, 140, 134, 49\rangle$	False	$\langle 12, 140, 137, 17\rangle$
$\langle 196, 132, 22, 49\rangle$	False	$\langle 4, 132, 25, 17\rangle$
$\langle 179, 62, 132, 19\rangle$	False	$\langle 3, 107, 132, 17\rangle$
$\langle 172, 38, 63, 68\rangle$	False	$\langle 12, 21, 243, 68\rangle$
$\langle 38, 204, 85, 89\rangle$	False	$\langle 6, 204, 102, 21\rangle$
$\langle 243, 54, 196, 188\rangle$	False	$\langle 3, 99, 196, 20\rangle$
$\langle 233, 12, 6, 17\rangle$	False	$\langle 9, 12, 9, 17\rangle$
$\langle 255, 104, 51, 198\rangle$	False	$\langle 15, 152, 51, 68\rangle$
$\langle 227, 148, 128, 25\rangle$	False	$\langle 3, 193, 128, 17\rangle$
$\langle 243, 132, 89, 155\rangle$	False	$\langle 3, 132, 86, 17\rangle$
$\langle 3, 121, 221, 147\rangle$	False	$\langle 3, 211, 221, 17\rangle$
$\langle 179, 134, 220, 108\rangle$	False	$\langle 3, 44, 220, 68\rangle$
$\langle 188, 17, 178, 236\rangle$	False	$\langle 12, 34, 43, 68\rangle$
$\langle 135, 180, 204, 25\rangle$	False	$\langle 11, 45, 204, 17\rangle$
$\langle 19, 130, 252, 229\rangle$	False	$\langle 3, 40, 252, 69\rangle$
$\langle 6, 15, 204, 20\rangle$	True	$\langle 5, 195, 204, 20\rangle$
$\langle 134, 15, 102, 208\rangle$	True	$\langle 5, 195, 170, 20\rangle$
$\langle 9, 204, 105, 17\rangle$	True	$\langle 9, 204, 102, 17\rangle$, $\langle 9, 204, 153, 17\rangle$, $\langle 9, 204, 150, 17\rangle$
$\langle 12, 51, 195, 68\rangle$	True	$\langle 12, 51, 150, 68\rangle$, $\langle 12, 51, 105, 68\rangle$, $\langle 12, 51, 60, 68\rangle$
$\langle 3, 60, 51, 17\rangle$	True	$\langle 3, 153, 51, 17\rangle$, $\langle 3, 102, 51, 17\rangle$, $\langle 3, 195, 51, 17\rangle$

Table 4.13: Some 5-cell non-uniform ECAs having non-trivial reversed cycle pseudo-isomorphisms under null boundary condition

Input CA	Reversible	Pseudo-Isomorphic CA(s)
$\langle 163, 130, 254, 173, 112\rangle$	False	$\langle 3, 40, 254, 173, 80\rangle$
$\langle 131, 134, 207, 205, 196\rangle$	False	$\langle 3, 44, 207, 205, 68\rangle$
$\langle 236, 179, 72, 86, 57\rangle$	False	$\langle 12, 179, 72, 89, 17\rangle$
$\langle 163, 147, 204, 63, 78\rangle$	False	$\langle 3, 57, 204, 63, 68\rangle$
$\langle 19, 68, 192, 169, 25\rangle$	False	$\langle 3, 68, 192, 166, 17\rangle$
$\langle 211, 57, 205, 193, 198\rangle$	False	$\langle 3, 147, 205, 193, 68\rangle$
$\langle 211, 225, 204, 6, 177\rangle$	False	$\langle 3, 180, 204, 9, 17\rangle$
$\langle 35, 28, 64, 60, 153\rangle$	False	$\langle 3, 73, 64, 60, 17\rangle$
$\langle 163, 134, 207, 203, 238\rangle$	False	$\langle 3, 44, 207, 203, 68\rangle$
$\langle 12, 54, 220, 98, 59\rangle$	False	$\langle 12, 54, 220, 146, 17\rangle$
$\langle 10, 120, 204, 57, 17\rangle$	True	$\langle 10, 120, 204, 54, 17\rangle$
$\langle 10, 15, 85, 195, 68\rangle$	True	$\langle 5, 240, 102, 15, 68\rangle$
$\langle 3, 198, 204, 89, 5\rangle$	True	$\langle 3, 108, 204, 89, 5\rangle$
$\langle 10, 180, 204, 108, 17\rangle$	True	$\langle 10, 180, 204, 156, 17\rangle$
$\langle 6, 51, 60, 51, 65\rangle$	True	$\langle 9, 51, 195, 51, 20\rangle$
$\langle 12, 156, 204, 105, 17\rangle$	True	$\langle 12, 156, 204, 102, 17\rangle$, $\langle 12, 156, 204, 153, 17\rangle$, $\langle 12, 156, 204, 150, 17\rangle$
$\langle 12, 204, 51, 150, 68\rangle$	True	$\langle 12, 204, 51, 195, 68\rangle$, $\langle 12, 204, 51, 60, 68\rangle$, $\langle 12, 204, 51, 105, 68\rangle$
$\langle 3, 150, 204, 108, 17\rangle$	True	$\langle 3, 195, 204, 108, 17\rangle$, $\langle 3, 60, 204, 156, 17\rangle$, $\langle 3, 105, 204, 156, 17\rangle$
$\langle 3, 57, 204, 198, 17\rangle$	True	$\langle 3, 147, 204, 198, 17\rangle$, $\langle 3, 57, 204, 201, 17\rangle$, $\langle 3, 147, 204, 201, 17\rangle$
$\langle 12, 51, 204, 153, 17\rangle$	True	$\langle 12, 51, 204, 150, 17\rangle$, $\langle 12, 51, 204, 105, 17\rangle$, $\langle 12, 51, 204, 102, 17\rangle$

Table 4.14: Some 4-cell non-uniform ECAs having non-trivial reversed cycle pseudo-isomorphisms under periodic boundary condition

Input CA	Reversible	Pseudo-Isomorphic CA(s)
$\langle 204, 151, 51, 55\rangle$	False	$\langle 204, 103, 51, 55\rangle$
$\langle 207, 155, 51, 131\rangle$	False	$\langle 207, 107, 51, 41\rangle$
$\langle 72, 182, 51, 57\rangle$	False	$\langle 72, 185, 51, 57\rangle$
$\langle 168130231248\rangle$	False	$\langle 152160235188\rangle$
$\langle 16, 91, 97, 31\rangle$	False	$\langle 32, 31, 81, 91\rangle$
$\langle 236, 51, 22, 136\rangle$	False	$\langle 236, 51, 67, 136\rangle$
$\langle 98, 51, 131, 236\rangle$	False	$\langle 146, 51, 41, 236\rangle$
$\langle 127, 123, 242, 84\rangle$	False	$\langle 127, 187, 182, 84\rangle$
$\langle 1, 69, 127, 159\rangle$	False	$\langle 16, 69, 127, 95\rangle$
$\langle 106, 15, 170, 165\rangle$	True	$\langle 85, 210, 90, 240\rangle$
$\langle 170, 30, 166, 240\rangle$	True	$\langle 86, 240, 170, 180\rangle$
$\langle 51, 54, 204, 198\rangle$	True	$\langle 51, 99, 204, 201\rangle$
$\langle 170, 180, 101, 15\rangle$	True	$\langle 106, 240, 85, 135\rangle$
$\langle 86, 15, 85, 75\rangle$	True	$\langle 85, 135, 101, 15\rangle$
$\langle 166, 15, 85, 165\rangle$	True	$\langle 85, 210, 90, 15\rangle$
$\langle 102, 240, 51, 51\rangle$	True	$\langle 170, 60, 51, 51\rangle$
$\langle 54, 51, 57, 204\rangle$	True	$\langle 57, 51, 57, 204\rangle$, $\langle 54, 51, 147, 204\rangle$, $\langle 57, 51, 147, 204\rangle$
$\langle 51, 198, 204, 54\rangle$	True	$\langle 51, 108, 204, 54\rangle$, $\langle 51, 198, 204, 57\rangle$, $\langle 51, 108, 204, 57\rangle$
$\langle 54, 204, 156, 51\rangle$	True	$\langle 99, 204, 156, 51\rangle$, $\langle 54, 204, 108, 51\rangle$, $\langle 99, 204, 108, 51\rangle$
$\langle 204, 51, 51, 195\rangle$	True	$\langle 204, 51, 51, 150\rangle$, $\langle 204, 51, 51, 105\rangle$, $\langle 204, 51, 51, 60\rangle$

Table 4.15: Some 5-cell non-uniform ECAs having non-trivial reversed cycle pseudo-isomorphisms under periodic boundary condition

Input CA	**Reversible**	**Pseudo-Isomorphic CA(s)**
$\langle 99, 204, 156, 51, 150\rangle$	False	$\langle 99, 204, 108, 51, 150\rangle$
$\langle 114, 204, 102, 51, 170\rangle$	False	$\langle 114, 204, 105, 51, 170\rangle$
$\langle 78, 64, 169, 51, 51\rangle$	False	$\langle 78, 64, 166, 51, 51\rangle$
$\langle 51, 204, 150, 204, 106\rangle$	False	$\langle 51, 204, 150, 204, 154\rangle$
$\langle 153, 240, 153, 204, 51\rangle$	True	$\langle 170, 195, 153, 204, 51\rangle$
$\langle 51, 150, 114, 204, 108\rangle$	True	$\langle 51, 150, 114, 204, 156\rangle$
$\langle 204, 149, 15, 51, 54\rangle$	True	$\langle 204, 149, 15, 51, 99\rangle$
$\langle 102, 204, 85, 195, 51\rangle$	True	$\langle 102, 204, 102, 15, 51\rangle$
$\langle 170, 85, 170, 85, 170\rangle$	True	$\langle 240, 240, 15, 240, 15\rangle$
$\langle 51, 147, 204, 204, 150\rangle$	True	$\langle 51, 57, 204, 204, 105\rangle$
$\langle 204, 60, 204, 85, 240\rangle$	True	$\langle 204, 60, 204, 170, 15\rangle$
$\langle 51, 150, 51, 85, 60\rangle$	True	$\langle 51, 150, 51, 153, 15\rangle$
$\langle 54, 204, 106, 240, 51\rangle$	True	$\langle 99, 204, 106, 240, 51\rangle$
$\langle 51, 57, 204, 108, 51\rangle$	True	$\langle 51, 147, 204, 156, 51\rangle$
$\langle 85, 60, 51, 156, 204\rangle$	True	$\langle 153, 15, 51, 201, 204\rangle$
$\langle 85, 170, 170, 170, 85\rangle$	True	$\langle 15, 15, 240, 240, 240\rangle$
$\langle 51, 51, 60, 204, 57\rangle$	True	$\langle 51, 51, 105, 204, 54\rangle$, $\langle 51, 51, 150, 204, 57\rangle$, $\langle 51, 51, 195, 204, 54\rangle$
$\langle 102, 204, 51, 153, 51\rangle$	True	$\langle 102, 204, 51, 60, 51\rangle$, $\langle 102, 204, 51, 195, 51\rangle$, $\langle 102, 204, 51, 102, 51\rangle$
$\langle 51, 201, 204, 108, 51\rangle$	True	$\langle 51, 156, 204, 108, 51\rangle$, $\langle 51, 201, 204, 156, 51\rangle$, $\langle 51, 156, 204, 156, 51\rangle$
$\langle 201, 204, 156, 51, 51\rangle$	True	$\langle 156, 204, 156, 51, 51\rangle$, $\langle 201, 204, 108, 51, 51\rangle$, $\langle 156, 204, 108, 51, 51\rangle$

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

Isomorphism in CAs has been explored in this project. A new concept named *pseudo-isomorphism* is introduced, along with a mathematical notion named *state-neighborhood relation*. The concept of pseudo-isomorphism is based on the premise that reversing a cycle of graph does not necessarily result in a isomorphic graph. An algorithm for obtaining pseudo-isomorphic CAs is proposed here, that works by reversing cycles in a transition diagram. Results obtained by rule complementation and reversing cycles have been discussed at the end of this thesis.

5.1 Contributions and Novelty

This thesis makes the following contributions to the ongoing research of CAs:

- Defines the concept of *pseudo-isomorphism* in graphs and CAs.
- Introduces a new mathematical notion named *state-neighborhood relation*.
- Devises an algorithm for obtaining pseudo-isomorphic CAs by reversing cycles in transition diagrams.
- Proves the necessary and sufficient conditions for obtaining a pseudo-isomorphic CA by reversing a set of cycles.

5.2 Scope for Future Work

As a possible extension to this project, efficient algorithms for the problem of *graph isomorphism* (checking if two given graphs are isomorphic) can be explored, which can then be directly used for checking isomorphism of CAs. There are numerous studies on this problem of graph isomorphism, that prove the upper bounds of computation for various graph topologies, including planar graphs. The transition diagrams of CAs discussed in this project fall under the category of planar graphs.

Similarly, the properties of pseudo-isomorphic graphs and pseudo-isomorphic CAs can be studied as a continuation of this project. Another possible area of work is to design new and efficient algorithms for synthesizing isomorphic CAs, in such a way that the algorithms can be applied on a wide variety of CAs and produce a reasonable subset of the possible isomorphisms.

REFERENCES

[1] Kamalika Bhattacharjee, Nazma Naskar, Souvik Roy, and Sukanta Das. "A survey of cellular automata: types, dynamics, non-uniformity and applications". In: *Natural Computing: An International Journal* 19.2 (2020), 433–461. ISSN: 1567-7818. URL: https://doi.org/10.1007/s11047-018-9696-8.

[2] Sukanya Mukherjee, Vicky Vikrant, and Kamalika Bhattacharjee. "Isomorphism in Cellular Automata". In: *Proceedings of Second Asian Symposium on Cellular Automata Technology*. Ed. by Sukanta Das and Genaro Juarez Martinez. Singapore: Springer Nature Singapore, 2023, pp. 193–206. ISBN: 978-981-99-0688-8. URL: https://doi.org/10.1007/978-981-99-0688-8_15.

[3] László Babai. "Graph isomorphism in quasipolynomial time [extended abstract]". In: STOC '16. Cambridge, MA, USA: Association for Computing Machinery, 2016, 684–697. ISBN: 9781450341325. URL: https://doi.org/10.1145/2897518.2897542.

[4] Niloy Ganguly, Biplab K. Sikdar, and P. Pal Chaudhuri. "Exploring Cycle Structures of Additive Cellular Automata". In: *Fundam. Inf.* 87.2 (2008), 137–154. ISSN: 0169-2968. URL: https://dl.acm.org/doi/abs/10.5555/1487722.1487724.

[5] Sukanta Das and Biplab K. Sikdar. "Classification of CA Rules Targeting Synthesis of Reversible Cellular Automata". In: *Cellular Automata*. Ed. by Samira El Yacoubi, Bastien Chopard, and Stefania Bandini. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 68–77. ISBN: 978-3-540-40932-8. URL: https://doi.org/10.1007/11861201_11.

[6] Sukanta Das and Biplab K. Sikdar. "Characterization of 1-d Periodic Boundary Reversible CA". In: *Electronic Notes in Theoretical Computer Science* 252 (2009). 15th International Workshop on Cellular Automata and Discrete Complex Systems, pp. 205–227. ISSN: 1571-0661. URL: https://doi.org/10.1016/j.entcs.2009.09.022.

[7] Stephen Wolfram. "Tables of Cellular Automaton Properties". In: *Theory and Applications of Cellular Automata (Including Selected Papers 1983-1986)*. Advanced Series on Complex Systems 1. World Scientific Publishing, 1986, pp. 485–557. URL: https://content.wolfram.com/sw-publications/2020/07/cellular-automaton-properties.pdf.