CYCLIC PROPERTIES OF CELLULAR AUTOMATA

SUKANYA MUKHERJEE

DEPARTMENT OF INFORMATION TECHNOLOGY
INDIAN INSTITUTE OF ENGINEERING SCIENCE & TECHNOLOGY, SHIBPUR
HOWRAH, WEST BENGAL, INDIA-711103

2022

Cyclic Properties of Cellular Automata

Sukanya Mukherjee
Registration No. PhD/R/2015/0159

A report submitted in partial fulfillment for the degree of

Doctor of Philosophy
in
Engineering

Under the supervision of

Dr. Sukanta Das
Associate Professor & Head
Department of Information Technology
Indian Institute of Engineering Science and Technology, Shibpur

Department of Information Technology
Indian Institute of Engineering Science and Technology, Shibpur
West Bengal, India – 711103

February, 2022

Department of Information Technology
Indian Institute of Engineering Science and Technology, Shibpur
P.O. Botanic Garden,
Howrah–711103

CERTIFICATE OF APPROVAL

It is certified that, the thesis entitled ***"Cyclic Properties of Cellular Automata"*** is a record of bona fide work carried out under my guidance and supervision by **Sukanya Mukherjee** in the Department of Information Technology of Indian Institute of Engineering Science and Technology, Shibpur.

In my opinion, the thesis has fulfilled the requirements for the degree of Doctor of Philosophy in Engineering of Indian Institute of Engineering Science and Technology, Shibpur. The work has reached the standard necessary for submission and, to the best of my knowledge, the results embodied in this thesis have not been submitted for the award of any other degree or diploma.

Date:

(Dr. Sukanta Das)
Associate Professor & Head
Department of Information Technology
Indian Institute of Engineering Science and Technology,
Shibpur, Howrah, West Bengal, India–711103

Dedicated
to the hope that
every orphan may get proper education
and achieve his/her aspiration.

Acknowledgement

It is believed that there is a circle around a point which is same as to make a person educated both mentally and academically with the help of his or her well-wishers and guides. I am one of the persons whose achievements in life dedicated to my guides, my family members and my friends.

First of all I express my gratefulness and admiration to my respected supervisor Dr. Sukanta Das, Associate Professor & Head, Department of Information Technology, Indian Institute of Engineering Science and Technology (IIEST), Shibpur, who always encouraged me to be disciplined in research life and to think deeply for a problem statement. His regularity in cooperation helped me to identify my faultiness and gives an idea about to overcome the weakness. He taught me how to conduct research and how the outcomes of the research can be communicated to the society through writing the articles. I could not have completed this work without his continuous help and guidance in all the stages of preparing this dissertation.

I gratefully acknowledge my doctoral committee members for their valuable comments during yearly progress report presentation and pre-submission presentation. I convey my gratitude to the former Heads of the Department of Information Technology, IIEST, Shibpur, Prof. Hafizur Rahaman, Prof. Santi Prasad Maity and Prof. Arindam Biswas and all other respected professors for their immense support and kind cooperation for continuing my research smoothly. I am thankful to all the technical and non-technical staff members (Malay Da, Suman Da and Dinu Da) of this department for their support and service.

I extend my gratitude to my co-researchers and friends Kamalika, Sumit and Nazma. They were instrumental in developing some of the work that has been included into this dissertation. Kamalika and Nazma were the two persons with whom I have shared moments of anxiety as well as of excitement during these years; their presence was very important in an activity that is often felt as tremendously solitaire. I met some of the most interesting and warm individuals in these years of Ph.D. Thanks to my Ph.D. mates Souvik, Supreeti, Raju, Deborpriya, Biswanath-da and Tuhin for sharing some of the nice moments that I would cherish for long.

I would like to thank Prof. Debika Bhattacharyya, Prof. Sourav Saha, Prof. Mohuya Chakraborty and Prof. Indraneel Mukhopadhyay, the heads of the department of Computer Science and Engineering of Institute of Engineering & Management for their constant support and motivation for continuing my research work in different phases of this journey. I am also grateful to Nilanajnadi, Tamal-da, Ee-kian madam, Tufan, Shubhasri and Bavrabi, my fellows in the department of Computer Science and Engineering, Institute of Engineering &

Management for their constant moral support and encouraging words even when I went through some hard moments.

I would like to convey my admiration to my baba (father) who taught me the basic attribute of life, attitude of a person. My choice of career path was influenced by my father's hope as he wanted to see me as an academician. I want to mention about two most important women of my life, my didi (elder sister) and my maa (mother). I always respectful to them for their life long sacrifice to make me continue my study, and they are the ideal of my life. I am also thankful to Bubun-da (my elder sister's husband) for his constant encouragement and motivation on my study since my secondary education. I am also respectful to mamoni (my mother-in-law), bapi (my father-in-law) and dada (my brother-in-law) for their kind support and inspiration for continuing my education. A special thanks to Anamitra, my husband whose care and great companionship help me to get rid of any problem. Besides, technical discussions with him helped a lot in writing research manuscripts.

I am fortunate to receive one of the best supports in my life from my honourable teachers, ever since my childhood days. I express my utmost gratitude to them; without their immense endearments, profound knowledge and valuable thoughts, it would have been impossible for me to reach at this level of education.

I convey my message *I am thankful to you* to all the persons whom I meet at any phase of my life.

Dated:
Indian Institute of Engineering Science
and Technology, Shibpur
Howrah, West Bengal, India

....
(Sukanya Mukherjee)
[Reg. No: PhD/R/2015/0159]

Abstract

This dissertation studies the cyclic behaviour of one-dimensional non-uniform cellular automata (CAs) with finite number of cells under null boundary condition. Further, this dissertation also shows that the reversible CA can be used as an effective clustering tool.

We observe identification of cycles in a CA as a sub-problem of *reachability problem*. To study the latter problem, we develop some theories on the characterization tool, named as the *reachability tree*. Using these theories, we can identify some scenarios where a given configuration is not reachable from itself. This dissertation also studies the problem of finding the cycle structure of an arbitrary CA. Our first strategy evaluates the cycle structures efficiently for irreversible cellular automata, but it is computationally demanding when the cycle length is exponential. We, therefore, propose an alternative strategy for finding the cycle structures exclusively for reversible cellular automata. Though our second strategy has also worst case exponential running time, it performs efficiently in practice for many reversible CAs.

We next introduce the *decomposition* property of cycle structure of cellular automata, and accordingly define *reducible* and *irreducible* cellular automata. Furthermore, the reducible cellular automata are categorized into *strictly* and *weakly* reducible cellular automata. A number of propositions are reported which efficiently identifies the categories of reversible cellular automata.

We next target to find the (near) optimal reversible cellular automaton (i.e. CA with minimum possible size) for a given cycle length. To this effect, we introduce the notion of *large length cycle* CAs. In order to generate such CAs, we first classify the reversible CA rules into four categories - *completely dependent*, *partially dependent*, *weakly dependent* and *independent*. The CA is then constructed by selecting more rules from *completely dependent*, moderate number of rules from *partially dependent* and very few from *weakly dependent.* Empirically it is shown that this design of CA through a stochastic approach is capable of generating all possible cycle lengths from 2^{n-1} to $2^n - 1$ using different n-cell reversible cellular automata. However, in order to verify whether a CA (generated by the above stochastic approach) can produce a cycle of the given length, we need to evaluate its cycle structure; but this evaluation can be computationally intensive, specially when the CA size is large. We therefore *concatenate* the *component* (large length cycle) CAs of small sizes to generate a larger sized CA, with the objective that the latter CA generates cycle of the desired length. One drawback of this method is that the size of this CA (generated by concatenating large length cycle CAs) may be far from optimal for the input cycle length. We show that this is an optimization problem, and apply a novel biologically inspired algorithm to solve the problem. Here, we introduce a notion of *significant features* which play the vital role in the creation of offsprings. An offspring is created by the significant features of multiple parents. We experiment with different values of cycle lengths ranging from 2^{40} to 2^{20K}; we observe that across all runs, our approach produces solution values exceeding the minimum possible CA size for the corresponding instance by only 0.78 percentage at most.

As an application of large length cycle CAs, this research reports a design of pseudo-random number generators (PRNGs). Since finding the longest cycle in such a stochastically generated large length cycle CA is computationally demanding, we take a window of a configuration (a substring of the configuration rather than the entire configuration) for generating random number. The randomness quality of our CA based PRNG is verified by Diehard battery of tests, battery rabbit of TestU01 library and NIST statistical test suite. Finally, when compared on the same platform, our PRNGs outperform most of the well-known PRNGs existing today. Further, the n-cell irreducible cellular automata with large length cycles can generate all possible prime numbers between 2^{n-1} to $2^n - 1$ as cycle lengths.

Finally, we propose reversible cellular automata as an effective clustering tool. Traditionally, a cluster consists of *close* objects, which, in case of CA necessarily means that these objects belong to the same *cycle* and are *reachable* from one another. This research figures out CA properties based on "reachability" that make the clustering effective. An *iterative* strategy is proposed which can generate the desired number of clusters in the present level (iteration) by *merging* the objects of *closely reachable* clusters of previous level using an unique *auxiliary CA*. We show that our algorithm is at least at par with the best algorithms existing today when evaluated using standard metrics for benchmark datasets. Our clustering scheme is capable of creating same arrangement of clusters even if the alteration is allowed on the ordering of the features for a given set of objects.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Humans have been intrigued about natural phenomena from time immemorial. Any untutored event instigates them to deduce some inference rationally. It is a common observation that most systems are in a continuous process of aging that leads to their decay and wear out with time. However, this concept of aging seems relative to the perspective of view. For example, the sun and the solar system are constantly aging and shall eventually die out, but to a general observer these are stable.

Many of the efforts in natural science have been earmarked to categorize systems based on whether they seem stable or in flux relative to us. These mainly rely on some models that mathematically capture the essential features of the physical systems [1]. Let us consider the example of heat flow. The second law of thermodynamics states that heat never spontaneously flows from colder body to warmer body; thus the process of heat flow is *irreversible*. On the other hand, formulation of the energy conservation law in terms of forces between indivisible elements of the bodies establishes Newtonian mechanics as fundamentally *time-reversible*; mathematically, such transformation is *bijective*. Most natural phenomena are reversible when observed from the microscopic viewpoint.

Many physical phenomena (such as traffic flow, forest fire, growth of cancer cells) can be expressed as many-body systems. Here, instead of considering individual elements, the system is considered as a collection of components distributed over space and evolving in time through local interactions amongst themselves [2]. Often, these interactions can be represented by a set of rather simple rules that captures the evolution of a component based on its neighbors. Cellular automaton (CA) is one of the modeling tools introduced in this context to provide a simple description of the complex natural systems.

A Cellular Automaton (CA) consists of a grid of elements termed as *cells*; a cell has a discrete *state* and the cell's state space is finite. The collection of states of all cells at a time step is called as *configuration*. CA evolves in discrete time and space. The state of a cell gets updated over time steps based on a *rule*, which is a *computable* local transition function on the states of the cell and its neighbors. In other words, the rules modify synchronously the state of each cell according to its state and those of the adjacent cells. A CA with m-neighborhood architecture represents that a cell updates its state based on the state values of its m-neighbors (including the cell itself). Thus, in every time step, a configuration transits to its *next configuration* by applying a rule to every cell of CA; the former configuration is said to be the predecessor of the latter. A configuration is said to be *reachable* if it has at least one predecessor. Classically, a CA follows the same rule at all cells (*uniform* CA) but recently there has also been a focus on *non-uniform* CA that follows different rules at different cells.

The above principles make the design of CAs quite intuitive and ingenious to model natural phenomena. Thus CAs are found to exhibit several fundamental properties of the physical world: they are massively parallel, evolving over time through local interactions. Some contributory works of CA in connection to modeling physical phenomena are listed in [3–14]. CA finds its applications in natural computing [15, 16], immune system [17–21], detection of genetic disorder of cells [22], study of the nature of fish migration [23] and population growth of vegetables [24, 25]; CAs can also be used to control forest firing [26–30] and traffic modeling [31, 32, 32–40].

The aspect of reversibility of most natural phenomena can be mimicked by *reversible* CA, a class of CA that preserves information [41]. The study of reversible CA was initiated by Hedlund [42] and Richardson [43]. The field of reversibility of finite CA has gained immense popularity over the last few years [44–54].

In a reversible CA, every configuration is reachable. Clearly, the transition function of such a CA must be bijective, so that its inverse exists. Traditionally, CAs are defined over infinite lattice with infinite configurations. In practice, however, one can not work with an infinite lattice; thus the lattice size (number of cells) has to be *finite*. If a CA has n number of cells, we call such CA as n-cell CA and in other words, we can say that the *size* of the CA is n. In a reversible CA of finite size, every configuration can be reached from itself and thus *cycles* are formed; elseways, some configurations do not belong to any cycle and such a CA is termed as *irreversible* CA. Therefore, the configuration space of any reversible CA of finite size represents cyclic space. Many physical systems can be modelled by such cyclic spaces; for example, determining the steady state or dynamic equilibrium of a natural process in thermodynamics can be modeled using cyclic spaces. In this dissertation, we study the *cyclic behaviour of finite* 1*-dimensional three neighborhood binary CAs with null boundary condition.*

1.1 Motivation and Objectives of the Dissertation

During evolution, a CA hops from one configuration to another configuration; thus a sequence (or path) of configurations can be formed where a particular configuration in the sequence is reachable from any configuration appearing before it in the sequence. This brings the notion of a decision problem - *Configuration REachability Problem* (CREP). Given a source configuration and a destination configuration, the goal of CREP is to decide whether the destination configuration is reachable from the source one. CREP is undecidable for 1-dimensional infinite CAs [55]; so researchers considered this problem for finite CAs [55, 56] where CREP is shown to be PSPACE-complete [55]. It has also been shown that CREP is NP-intermediate for the CAs with additive rules [56]. However, all works on CREP focus on the uniform CAs only, this motivates us to study CREP for non-uniform CAs in this dissertation.

The special case of CREP with the same source and destination configurations confirms the existence of a cycle in CA configuration space. Similarly, in a reversible CA, the existence of a path among a set of given l configurations establishes the presence of a cycle of length at least l. However, CREP is computationally demanding, impelling us to find the *cycle structure* of CA through some alternate mechanisms. If a CA generates μ_1 cycles of length l_1, μ_2 cycles of length l_2, and so on, then the cycle structure of that CA is $[\mu_1(l_1), \mu_2(l_2), \cdots]$. The cycle structure of a CA is a well-studied problem (see [57–60]), with both theoretical interests and applications [44, 53, 58, 61–63]. CAs with large length cycles are in demand for generating Pseudo-random patterns, to test VLSI circuits (see [44, 58, 61–63]) etc., CAs which have cycles of equal length are used in block cipher generation [53]. All these works about cycle structure focus on linear CAs only. In a more recent work [64], both linear and non-linear CAs have been considered for the following problems: (i) a decision algorithm to figure out whether a given CA configuration belongs to a cycle or not of the input CA; (ii) the total number of configurations that belong to the cycles of a CA; (iii) synthesis of CAs with only *point state attractors* (cycles of length one) which have great contributions in pattern recognition. However, [64] mainly covers irreversible CAs, and reversible CAs have not been explored to a great extent. Moreover, though a small piece of work is done on the cyclic properties of non-linear CAs, till date, no method is available for evaluating the cycle structures of an arbitrary non-linear CA. This dissertation aims to devise some strategies which compute the cycle structures of both irreversible and irreversible non-linear CA efficiently.

Note that the problem of finding the cycle structure of a CA explained above returns a summarized representation of CA cyclic space. The converse problem is equally appealing and challenging, where the aim is to deduce some properties of a CA given its cycle structure. To this effect, we focus on the cyclic space of reversible cellular automata and characterize the cyclic spaces of CAs and categorize the CAs into different classes.

Our next objective is to study the lengths of cycles of cellular automata. There has been a theoretical challenge since late 1980s to the CA community to get the maximum possible cycle length for a CA. Prior works have addressed this problem by considering the *maximal length* CAs (a CA that can generate a cycle of length $2^n - 1$ where the CA has n number of cells) [58, 65]. It has already been seen that maximal length CAs are mainly non-uniform linear CAs, and generation of maximal length CAs is related to primitive polynomials. The maximum degree of available binary primitive polynomial known till date is limited (see [66]). Moreover, the complexity of generating the primitive polynomial is exponential. Hence, given the inherent hardness, we can only resort to finding near optimal solution for the problem, with the objective that the *displacement* (difference from the optimal solution) of the attained near optimal solution should be as small as possible. Formally, any CA that generates a cycle of length $l \in \mathbb{N}$ is of size at least $n = \lfloor log_2 l \rfloor + 1$; thus our objective is to find a binary reversible CA of size $n + \delta$, with δ as small as possible, that generates a cycle of length at least l. Furthermore, we want to observe a practical applicability of such CAs as Pseudo-random Number Generators (PRNGs).

One of the well-known general purpose PRNGs is *Mersenne Twister* and its variants [67–69] where the period length is a *Mersenne* prime. Similar cycles can also be generated by a maximal length CA of size n, where the corresponding cycle length $2^n - 1$ is prime, that is a Mersenne prime. Hence maximal length CAs can be used as a special category of prime generator - Mersenne prime generator. However, if the CA is not a maximal length, then the cycle length, if it is a prime, can not be a Mersenne prime, but some other primes. This observation steers us to think whether a CA is capable of generating a cycle of arbitrary prime length. We also explore if there exists a collection of CAs of size n which can generate cycles of lengths of all possible primes in between 2^{n-1} and $2^n - 1$.

Recall that in a cellular automaton (CA), the configurations within a cycle are reachable from one another, whereas the configurations of different cycles are not reachable. A *cluster* also establishes an intrinsic connection among the objects. This connectivity indicates us to think reversible cellular automata (CAs) as natural clustering tools. Therefore, a reversible CA can act as a function that maintains a bijective mapping among the configurations which are reachable or connected and gathers similar objects (configurations) into same cluster (cycle). Clustering can be viewed as an optimization problem where a trade-off between two facets - *limited* number of clusters and less *intra-cluster* distances among the target objects (the values for any particular feature of the objects in a cluster are close) exists. Therefore, our objective is to design CA based clustering such that CA has limited number of cycles and the configurations inside a cycle represent related objects.

To summarize, the objectives of this dissertation are listed below:

- Study of CREP and cycle structure for non-linear CAs by employing the Reachability Tree.
- Characterization of the cyclic space of reversible cellular automata.
- Finding reversible CA of near optimal size that generates a cycle of given length.
- Study of non-linear CA as PRNG.
- Investigating CAs of size n which can generate cycles of all possible prime lengths in between 2^{n-1} and $2^n - 1$.
- Application of reversible cellular automata in clustering.

1.2 Contributions of the Dissertation

Our contributions in this dissertation are listed below.

- We employ reachability tree as a characterization tool of CA. By developing theories on reachability tree, we can decide for some selective cases whether a given configuration is reachable from itself or not (in this dissertation we consider the special case of CREP where the source and destination configurations are same).
- Using the reachability tree, we design a mechanism for computing the cycle structure of a CA by filtering out configurations that do not belong to any cycle. This scheme performs well for irreversible CA, but is computationally intensive for reversible CAs.

- To address the above issue, in the reachability tree, we capture the *resemblance* among the *subtrees* based on which we decide whether to traverse a particular subtree or not. To this effect, we introduce the concept of *blocks*. This dissertation introduces a new notion of *partial cycle structure* which evaluates the numbers of cycles along with their lengths of a subtree or a collection of subtrees. By detecting the resemblance among the subtrees and using the partial cycle structures, we infer that not all nodes in the reachability tree need to be explored, resulting in increased efficiency of our algorithm for evaluating the cycle structure of a reversible cellular automaton.

- We characterize the cyclic spaces of finite reversible cellular automata based on which we categorize those CAs broadly into two classes - *reducible* and *irreducible*. Further, reducible CAs are sub-categorized into *strictly reducible* and *weakly reducible* cellular automata. In this chapter, we introduce the notion of *isomorphic* CAs based on their cycle structures. Given a cyclic structure, we show efficient mechanisms of identifying the category of the CA.

- To design a binary n-cell CA that generates a cycle of length at least $l \in \mathbb{N}$ such that n is as small as possible, the following contributions are summarized.

  - A stochastic method has been developed to synthesize CAs with *large* cycles. We introduce an n-cell CA as *large length cycle* CA if the CA generates a cycle of length l where $l > 2^{n-1}$.
  - An operator has been proposed which operates over two CAs of sizes n_1, n_2 and forms a new CA of size $n_1 + n_2$.
  - An evolutionary strategy has been developed to obtain near optimal CA for a given length of cycle. The novelty of this scheme is that it reduces not only the displacement, but also generates the best solution efficiently.

- We use the large length cycle CAs generated from the above mentioned stochastic process as a source of randomness and report a generalized scheme of using these CAs as window-based pseudo-random number generators. This scheme offers some advantages like ease of hardware implementation, robustness and portability. We have empirically tested these PRNGs for randomness using Diehard, TestU01 and NIST as the testbeds.

- We also demonstrate that irreducible CAs of size n can act as prime number generator for an arbitrary prime number in the range $[2^{n-1}, 2^n - 1]$.

- We report an *iterative* strategy which can generate *desired* number of clusters on demand. This work presents the performance analysis of our proposed cycle based clustering algorithm on some real datasets taken from ML repository. We also compare our proposed algorithm with some traditional benchmark clustering algorithms like *centroid based clusterings*, *hierarchical clusterings* [70, 71]. Our results indicate that, performance of our CA-based clustering technique is at least as good as the best known clustering algorithm existing today.

1.3 Organization of the Dissertation

The dissertation is organized as follows:

Chapter 2 presents an archive of CA. Here, the expedition of Cellular Automata (CAs) is described based on its attributes. Some seminal works on CA's reversibility property and CAs' role in technological advancements (specially in machine learning) have been discussed in this chapter. Besides, this chapter provides some fundamental concepts, terminologies, definitions and notations of CAs by giving priority on ECAs (1-dimensional binary CA with three-neighborhood architecture); a brief outline on *Reachability tree* (which is the characterization tool for non-linear CAs) and reversible CAs (which include non-linear non-uniform CAs) is presented.

Chapter 3 studies computation of the cycle structure of a CA. As finding a cycle can be thought of as a sub-problem of CREP, we start the journey of this chapter by studying CREP (where the input CA can be linear or additive or non-linear, and the CA can be either classical or non-uniform). Here, CREP considers that the source and destination configurations are same. Reachability tree is explored to develop some theories for affirming whether the given configuration is not reachable from itself without exploring up to leaf level of that tree. However, for the scenario where there exists a path of l configurations between the source and destination (in our case, the length of that cycle where the given configuration belongs, is l), there is no shortcut mechanism and we need to traverse at least l configurations to reach the given configuration from where it started and it can take exponential amount of time when l is exponential in the CA size n. Thus we are motivated to design an efficient mechanism to evaluate the cycle structure of CA. Our first strategy evaluates the cycle structure by discarding *acyclic* configurations before reaching to leaf level. Based on experimentation, we demonstrate that our scheme performs well in case of irreversible CA. However, the results are not very satisfactory for reversible CAs. This motivates us to figure out some notable properties of reachability tree which may help to reduce the computational cost of cycle structure. To this effect, few notions are introduced: an *intra-linked* subtree and an *inter-linked* collection of subtrees in the reachability tree. We identify the property of *isomorphism* for finding the resemblance either among the intra-linked subtrees or among the inter-linked collections of subtrees in the reachability tree. We figure out that out of these isomorphic subtrees, only one of them needs to be explored (some subtrees in the tree can be skipped for traversal purpose) for evaluating the cycle structure of CA. In this way, the cycle structure of CA is evaluated by the partial cycle structures of isomorphic subtrees. The concept of "block" has been introduced for representing the reachability tree as a collection of intra-linked subtrees and the inter-linked collections of subtrees. It is observed that the presence of *some* rules in the CA design confirms the existence of blocks and isomorphism in the reachability tree. Empirical investigation shows that the efficacy of our (modified) strategy depends on the design of CAs. The worst case time complexity is obviously still exponential for cases where the given CA does not exhibit isomorphism in the corresponding reachability tree; nevertheless, for many practical cases, our scheme significantly reduces the running time.

Chapter 4 reports on the characterization of reversible CAs based on their cyclic spaces. Here, we introduce the *decomposition* property of cycle structure and based on this property, the CAs are categorized. If the given cycle structure of a CA can not be decomposed, then such CA is called as *irreducible*; otherwise, the cycle structure of that CA can be *decomposed* into the cycle structures of *small* sized CAs and such cellular automata are called as *reducible*. Now, a reducible cellular automaton can be either *strictly reducible* or *weakly reducible* based on whether the cycle structure of an n-cell reversible cellular automaton can be decomposed into the cycle structures of n_1-cell and n_2-cell CAs where $n = n_1 + n_2$ or only n_1-cell CA (where $n_1 < n$) respectively. Given a cyclic structure, for deciding the CA category, we devise some mechanism based on the lengths of cycles as well as the count of cycles.

In **Chapter** 5, we aim to design a binary n-cell CA that can generate a cycle of length at least $l \in \mathbb{N}$ such that n is as small as possible. We begin with a stochastic approach to address this problem. The idea is as follows: we first classify the CA rules based on some *parameter*, which determine the dependence of the next state of a cell on the present states of its neighbors. We classify the CA rules into four categories - *completely dependent*, *partially dependent*, *weakly dependent* and *independent*. Our hypothesis is that if the rules of a CA possess a higher value of that parameter, then the CA is expected to have larger length cycle. Thus a CA constructed by selecting the rules randomly by assigning more priority to the rules that have higher values of these parameters is expected to generate *large length cycles*. However, as there is no efficient mechanism to decide if the CA has actually a cycle of length l, we need a brute force method to know the lengths of cycles of the CA. This implies, for given l, where $\lfloor log_2 l \rfloor + 1$

is large, determining CA using the above approach is infeasible in practice. As a solution to this issue, we introduce an operator which takes two CAs of cycle lengths l_1 and l_2, and produces a CA with a cycle of length $\texttt{lcm}(l_1, l_2)$. Our approach thus is to first stochastically synthesize small sized *component* CAs, and test if these CAs really generate large cycle (since these component CAs are of small sizes, it is computationally feasible to test whether they generate large cycles or not). Next we apply the operator on these already synthesized small sized CAs to generate our desired large sized CA. The solution obtained by the above approach works fine for many input cases. For the scenarios where it has large displacement, the solution is further improved by employing an evolutionary strategy which conforms some divergent of genetic algorithm. The notable change in our strategy with respect to genetic algorithm is that an offspring is not produced by the selected parents using genetic operators like crossover or mutation. Here, the creation of an offspring is influenced by multiple parents in such a way that the offspring owns the *significant* features of the selected parents. From the experimental results, we observe that our stochastic approach coupled by the evolutionary strategy drastically reduces the displacement of the solution. As an application of CAs with *large length cycles* (generated by the stochastic approach), we have reported a design of PRNGs. Our scheme offers benefits like portability, ease of hardware implementation, unpredictability, robustness, etc. We have verified the randomness quality of these CAs as PRNGs by using Diehard battery of tests, battery rabbit of TestU01 library and NIST statistical test suite. This chapter reports an application of irreducible CAs as prime number generator. It is observed that irreducible CAs of size n with large length cycles are the optimal candidate CAs which can generates a cycle of length l where l is an arbitrary prime number in the range $[2^{n-1}, 2^n - 1]$.

Chapter 6 reports a CA based clustering technique where we have used the reversible CAs to distribute the target objects among clusters. In a reversible CA, the configuration space is divided into a number of cycles where the configurations inside a cycle are *reachable* from each other. This "reachability" is exploited as our closeness metric to distribute the configurations of each cycle as a distinct cluster. In this way, reversible CA can aptly work as a natural clustering tool, where the target objects are encoded as configurations of a CA. However, any arbitrary reversible CA may not work as *effective* clustering technique for a given dataset. So, we have identified some properties which must be satisfied by a CA to be a *good* candidate for clustering. This CA can be called as an *auxiliary* CA. Such a CA can effectively cluster the target objects ensuring less intra-cluster and more inter-cluster distance. Nevertheless, sometimes the application may need a specific number clusters whose corresponding bijective function may not be represented as a CA. To deal with this, we have developed a level-wise iterative clustering algorithm, where instead of a single reversible

CA, we choose one reversible CA per level where each cluster of one level is treated as an object in the next level. In this approach, the *closely reachable* objects (clusters) of the previous level are *merged* using an unique auxiliary CA in the present level. We have tested our algorithm on some standard datasets based on three well-known benchmark validation indices. It can be observed that performance of our CA based iterative clustering technique is at par with the best result obtained by the state-of-the-art algorithms. This research also confirms that the arrangement of clusters outputted by our CA based clustering algorithm remains fixed even if the ordering of the features of a given set of objects is changed (verified with the real datasets).

In **Chapter** 7, the concluding remarks are made on theoretical aspects and experimental results of our work mentioned throughout this dissertation. This chapter also gives the direction for future work which are either unexplored or can lead to further improvement in our work.

CHAPTER 2

Cellular Automata: Definitions, Survey and Useful Terminologies

In this dissertation, we study the cyclic properties of 1-dimensional finite cellular automata (CAs). In particular, we explore the cyclic behaviour of reversible cellular automata to understand their intrinsic properties. Our work supports both classical (uniform) and non-uniform CAs. This dissertation also shows that reversible cellular automata can be very useful to clustering problem.

In this backdrop, the current chapter presents an archive of CAs - their initiation, a brief tour to the milestones, a report on global behaviour of CAs with special focus on *reversibility* and the contributions of CA in technological advancements. Since we investigate the potentiality of cellular automata in clustering problem, we survey the role of CAs in machine learning (Section 2.2). Section 2.3 introduces some special terms and terminologies, few fundamental concepts and notations are used throughout the thesis. Besides, a brief outline on the analysis and synthesis of finite (non-uniform) reversible CAs is presented, here.

2.1 Expedition of Cellular Automata (CAs)

A cellular automaton (CA) is a collection of an orderly fashioned elements that are termed as *cells*; a cell has a discrete *state* that is a member of a finite set. A cell can be thought of as a finite state machine which updates its state at every discrete time step to go to its *next state*. A cell uses a *local transition function* which takes the present states of the cell's neighbors as argument for the update.

Definition 2.1 *A cellular automaton (CA) is a quadruple ($\mathscr{L}, S, \mathcal{N}, \mathcal{R}$), where*

- $\mathscr{L} \subseteq \mathbb{Z}^D$ *is the* D*-dimensional cellular space.*
- S *is the finite set of states.*
- $\mathcal{N} = (\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_m)$ *is the neighborhood vector with* m *distinct elements of* $\mathscr{L}$ *which associates one cell to its* neighbors*. The neighbors of a cell at location* $\vec{v} \in \mathscr{L}$ *are the cells at locations* $(\vec{v} + \vec{v}_i) \in \mathscr{L}$*, for all* $i \in \{1, 2, \cdots, m\}$
- $\mathcal{R} : S^m \rightarrow S$ *is the local rule of the automaton which is used as the local transition function applied on cells of that automaton.*

All cells of a CA update their states at discrete time steps and the collection of state values of all cells at a particular time step is called as a *configuration.* Thus, a configuration of a CA is a mapping $\mathcal{A} : \mathscr{L} \rightarrow S$ that specifies the states of all cells. Therefore, $S^{\mathscr{L}}$, the set of all possible mappings from $\mathscr{L}$ to S is the set of all possible configurations.

During the evolution, CA transits from one configuration to its *next* configuration. Thus, a CA can be represented as a mapping $G : S^{\mathscr{L}} \rightarrow S^{\mathscr{L}}$. Let $\mathtt{x} = (x_{\vec{v}})_{\vec{v} \in \mathscr{L}}$ be the initial configuration of the CA and $\mathtt{y} = (y_{\vec{v}})_{\vec{v} \in \mathscr{L}}$ be the next configuration of $\mathtt{x}$. $\mathtt{y}$ can be evaluated as follows (for all $\vec{v} \in \mathscr{L}$):

$$y_{\vec{v}} = G(\mathtt{x})|_{\vec{v} \in \mathscr{L}} = \mathcal{R}(x_{\vec{v}+\vec{v}_1}, x_{\vec{v}+\vec{v}_2}, \cdots, x_{\vec{v}+\vec{v}_m}) \tag{2.1}$$

Here $\mathtt{y}$ is the *successor* of $\mathtt{x}$, and $\mathtt{x}$ is the *predecessor* of $\mathtt{y}$. G is called the *global transition function* of the CA which is induced by the local transition function, popularly called as the local *rule* applied on the cells of that CA. Classically, all cells of a CA follow the same rule. Applying the global transition function G repeatedly, we can get the evolution of the CA which generates an infinite sequence of configurations $G(\mathtt{x}), G^2(\mathtt{x}), G^3(\mathtt{x}), \cdots$.

The neighborhood architecture of a CA can alternatively be represented by the *radius* of that CA. For example, in 1-dimensional CA, let us consider the r_l and r_r neighbors at the left and right sides respectively of a cell. When $r_l = r_r = r$, then r is the number of consecutive distinct cells (called as neighbors) in each direction on which a cell depends on, and the neighborhood can be represented as $(-r, -r+1, \cdots, -1, 0, 1, \cdots, r-1, r)$. Therefore $\mathcal{R} : S^{2r+1} \rightarrow S$, and G of Equation 2.1 can be represented as follows.

$$y_i = G(\mathtt{x})|_{i \in \mathscr{L}} = \mathcal{R}(x_{i-r}, x_{i-r+1}, \cdots, x_{i+r-1}, x_{i+r}) \tag{2.2}$$

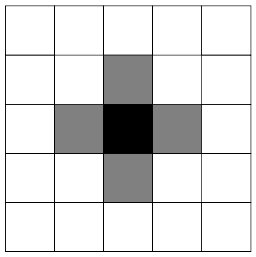

Figure 2.1: von Neumann neighborhood: Black cell is the cell under consideration and the shaded cell including itself are its neighbors.

The journey of CA was conceived by John von Neumann [72]. Starting from von Neumann's CA, the different variants of CA models are explored based on the key attributes of CAs, such as the states of the cell, the dimension of CA, the neighborhood of a cell, etc. Next, we revisit some work done in CA theory which are achieved by tuning these attributes.

2.1.1 A Tour on Attributes of CAs

The von Neumann's CA is a 2-dimensional CA which considers 29 states per cell with 5-neighborhood architecture (see Figure 2.1). In that CA, a cell (dark square in the figure) depends on its four neighbors (shaded squares) and on the cell itself. The construction of von Neumann's CA leads to the design of a simplified structure; some notable work in this aspect include Thatcher's model [73], Arbib's work [74], etc. The simplification was targeted either on the reduction of state count or on representation of neighborhood structure.

The notable change on CA's state count was initiated by Codd where the cells of CA use *eight* states [75]. In Banks' work on universal computability of CA [76], the number of CA states are *two*; moreover, models of self-reproducing CAs are introduced considering *four* states. All these CAs are 2-dimensional with 5-neighborhood architecture.

The seminal work on CA neighborhood after von Neumann's CA is *Moore neighborhood* [77] - 9-neighborhood dependency (see Figure 2.2). Moore neighborhood CA with binary states contributes to the design of the famous *Game of Life* developed by John Conway. Besides von Neumann and Moore neighborhood architectures, Margolus neighborhood [78] is also reported which introduces the concept of *block* cellular automata and *partitioning* cellular automata. There is always a correlation between the neighborhood of a CA and its state count. Smith [79] showed that the neighborhood size and state count are interrelated - a CA with higher neighborhood size can always be emulated by a CA of lesser neighborhood size having higher number of state count and vice-versa.

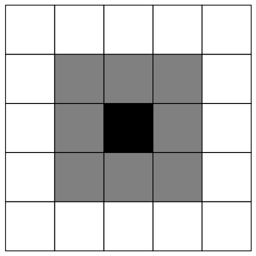

Figure 2.2: Moore neighborhood: The black cell has 9 neighbors including itself.

Traditionally each cell is considered as a square, but some alternatives have also been reported in prior art. Morita et al. studied CAs with hexagonal [80,81] and triangular [82,83] cells; CAs can also be defined in hyperbolic plane [84,85]. Varying cell shape results in different neighborhood dependencies.

Besides the study of primary properties and parameters of 2-dimensional CAs [86–89], researchers have also proposed cellular automata for higher dimensions [90–94]. Almost all decision problems are undecidable (see for example [55, 94]) in higher dimensional CAs; for example, reversibility is undecidable for higher dimensional CAs [87]. In this thesis, we focus on 1-dimensional CA which is well-studied in literature. In fact, in the world of cellular automata, the most popular is 1-dimensional CA - a large section of researchers are attracted in this domain [95–98]. Classically for a given 1-dimensional CA, the radius remains same in both the directions, though such constraint is relaxed in some work [99, 100]; *one-way* CA [101] is an example of such 1-dimensional CAs where the cells of CA depend on the neighbors of one side (either left side or right side). The most ground-breaking work in 1-dimension is Wolfram's CAs [102–104]; such CAs are also called as *Elementary cellular automata* or ECAs [104].

An *Elementary Cellular Automaton* (ECA) is a 1-dimensional CA consisting of the state set $S = \{0, 1\}$. The radius value is 1, which implies that the update of an ECA cell depends on the present states of its left neighbor, itself and its right neighbor; thus ECA supports 3-neighborhood architecture (Figure 2.3). Therefore, a local rule of ECA can be defined as $\mathcal{R} : \{0, 1\}^3 \rightarrow \{0, 1\}$. In evolution of an ECA, each configuration determines its next configuration in the following way.

$$y_i = G(\mathtt{x})|_{i\in\mathscr{L}} = \mathcal{R}(x_{i-i}, x_i, x_{i+1}) \tag{2.3}$$

The number of possible local rules are (2^{2^3}) 256. Each such rule is represented as an eight-bit sequence of an integer in the range 0 to 255; for example, let us consider rule 54 where the rules

$$\mathcal{R}(1, 1, 1) = 0, \mathcal{R}(1, 1, 0) = 0, \mathcal{R}(1, 0, 1) = 1, \mathcal{R}(1, 0, 0) = 1,$$

$$\mathcal{R}(0, 1, 1) = 0, \mathcal{R}(0, 1, 0) = 1, \mathcal{R}(0, 0, 1) = 1, \mathcal{R}(0, 0, 0) = 0,$$

Figure 2.3: Elementary cellular automaton: here, the black cell is the cell under consideration; the gray colored cell and the cell with diagonal stripes are the left and right neighbors respectively of the cell under consideration.

are obtained from the binary representation of $54 = (00110110)_2$. The local rules for ECAs can also be represented in a tabular form (see Table 2.1), where the present state combinations of a cell and its neighbors and the next state of that cell are represented by PS and NS respectively. In the table, PS actually refers to $x_{i-1}x_ix_{i+1}$, the sequence of the present states of cells $i-1$, i and $i+1$; whereas, NS represents $\mathcal{R}(x_{i-1}, x_i, x_{i+1})$ (thus $\mathcal{R}$ is applied to any cell i). It is an easy observation that there can exist 256 possible ECA rules. An ECA rule can be either *linear*, *non-linear* or *complemented.*

Definition 2.2 *An ECA rule $\mathcal{R}$ is called linear if $\mathcal{R}$ can be expressed as $\mathcal{R}(x_{i-1}, x_i, x_{i+1}) = ax_{i-1} + bx_i + cx_{i+1} \pmod 2$, where $a, b, c \in \{0, 1\}$. A rule which can be expressed as $1-\mathcal{R}(x_{i-1}, x_i, x_{i+1})$, where $\mathcal{R}$ is a linear rule, is called as complemented rule. Therefore, linear and complemented ECA rules can be expressed as XOR and XNOR respectively of their input variables (whereas a non-linear rule can never be expressed as XOR or XNOR of its input variables).*

The first column of Table 2.2 shows *eight* example rules 0, 60, 90, 102, 150, 170, 204 and 240 which are *linear.* These rules are also called as *additive*, since for binary CAs, the additive property is implied by linearity. Linear/additive ECA rules are expressed as XOR of their input variables. Let us consider a linear ECA rule 60 where $\mathcal{R}(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \oplus x_i$. Using the XORed expression, the values of a, b and c are represented as 1, 1 and 0 respectively for rule 60, whereas rule 150 represents $a = 1$, $b = 1$ and $c = 1$ as here $\mathcal{R}(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \oplus x_i \oplus x_{i+1}$. Also, the XORed expression of a linear rule contributes in interpreting the neighborhood dependency of that rule. For example, every cell with rule 90 is dependent of its left and right neighbors.

A complemented rule is the logical complement of a linear/additive rule, there-

Table 2.1: ECA rules 10, 60, 15, 180 and 20

PS	111	110	101	100	011	010	001	000	Rule
	0	0	0	0	1	0	1	0	10
	0	0	1	1	1	1	0	0	60
NS	0	0	0	0	1	1	1	1	15
	1	0	1	1	0	1	0	0	180
	0	0	0	1	0	1	0	0	20

fore, another set of *eight* rules represent complemented rules. The *second* column of Table 2.2 represents the complemented rule of the corresponding linear rule in the first column. The rule which does not belong to any of linear/additive and complemented categories, is called as *non-linear* rule. There are total 240 ECAs.

Table 2.2: Linear and Complemented ECA Rules ($\mathcal{R}(x_{i-1}, x_i, x_{i+1})$)

Linear Rules	Complemented Rules
$0(x_{i-1}, x_i, x_{i+1}) = 0$	$255(x_{i-1}, x_i, x_{i+1}) = 1$
$60(x_{i-1}, x_i, x_{i+1}) =$ $x_{i-1} + x_i \pmod 2$	$195(x_{i-1}, x_i, x_{i+1}) =$ $1 - 60(x_{i-1}, x_i, x_{i+1})$
$90(x_{i-1}, x_i, x_{i+1}) =$ $x_{i-1} + x_{i+1} \pmod 2$	$165(x_{i-1}, x_i, x_{i+1}) =$ $1 - 90(x_{i-1}, x_i, x_{i+1})$
$102(x_{i-1}, x_i, x_{i+1}) =$ $x_i + x_{i+1} \pmod 2$	$153(x_{i-1}, x_i, x_{i+1}) =$ $1 - 102(x_{i-1}, x_i, x_{i+1})$
$150(x_{i-1}, x_i, x_{i+1}) =$ $x_{i-1} + x_i + x_{i+1} \pmod 2$	$105(x_{i-1}, x_i, x_{i+1}) =$ $1 - 150(x_{i-1}, x_i, x_{i+1})$
$170(x_{i-1}, x_i, x_{i+1}) = x_{i+1}$	$85(x_{i-1}, x_i, x_{i+1}) =$ $1 - 170(x_{i-1}, x_i, x_{i+1})$
$204(x_{i-1}, x_i, x_{i+1}) = x_i$	$51(x_{i-1}, x_i, x_{i+1})$ $1 - 204(x_{i-1}, x_i, x_{i+1})$
$240(x_{i-1}, x_i, x_{i+1}) = x_{i-1}$	$15(x_{i-1}, x_i, x_{i+1}) =$ $1 - 240(x_{i-1}, x_i, x_{i+1})$

ECAs have attracted researchers from diverse domains, and the phase wise developments of ECA can be found in [44, 102, 105–115]. Next we dig into the behaviour of a CA by exploring its primary properties that can be mapped to the real-world problems.

2.1.2 Global Behaviour and Reversibility

A CA exhibits complex global behaviour by its local interaction and computation. The global behaviour of CAs such as universality, reversibility, laws of conservation etc., play a key role in establishing cellular automata to model physical systems and solve real-life problems. Let us now discuss the work done on some of these inherent properties of CAs [115].

2.1.2.1 Universality

A CA can perform universal computation; an arbitrary Turing machine can be simulated by a CA [106]. The universality property of a CA was conceptualized in [73, 74, 116]. Prior work [117, 118] have shown Turing machine can also be simulated by reversible CA. One-way CAs and totalistic CAs [102] can also perform universal computation [119]. Even using very simple rules [120, 121], CA can show its universal computational ability. The universal computation property has also been shown by 1-dimensional CA of radius 1 and 2 with state counts 7 and 4 respectively [122]. ECA rule 110 [123] is also computationally as strong as Turing machine [109]. It is found that a CA can simulate another CA, and one remarkable work [124] shows that a CA having 14 states can simulate any CA whose initial configuration and local rule are encoded as an initial configuration of the former CA. However, a CA is called intrinsically universal if it can simulate all CAs of the same dimension [114]. The smallest intrinsically universal CAs are found in both 1-dimensional and 2-dimensional spaces (see work in [125] and [116] respectively).

2.1.2.2 Reversibility and Garden-of-Eden

Reversibility in CAs was first studied in [42, 43]. A CA is *injective* if its global transition function G is one-to-one, and *surjective* when G is onto. If a CA is *invertible* [43] such that the CA is injective, therefore, every configuration of that CA has exactly one predecessor and one successor. A CA is *reversible* if its global transition function is *bijective*. All injective CAs are reversible, as shown in Hedlund's work and Richardson's work. Amoroso and Patt [95] contributed on deciding reversibility of 1-dimensional infinite CA. A decision algorithm on reversibility is also reported for CA with *finite* number of cells [126]. Sutner in his work [96] proposed an effective decision algorithm to test reversibility of a 1-dimensional CA. However for case of higher dimensions, like 2-dimensional CA, there is no algorithm which can decide reversibility of an arbitrary CA [87]. Toffoli in his work [127] showed that an arbitrary CA of dimension D can be simulated by a reversible CA with dimension $D+1$, whereas Morita et al. showed that 1-dimensional reversible CA is capable of simulating any 1-dimensional CA with finite configurations [118, 128]. Some other notable works on reversibility are reported in [88, 129–136].

The concept of non-reachability - *Garden-of-Eden* is introduced in the work of Moore [77] and Myhill [137]. These theorems are related to the *injectivity* and *surjectivity* properties of CA. A configuration is termed as Garden-of-Eden configuration if it does not have any predecessor. Such configuration is also called as *non-reachable* configuration. Moore has claimed that the existence of non-reachable configuration implies the existence of a configuration with more than one predecessor. The converse statement is claimed and proved by Myhill in [137]. A CA with finite configurations is called as *irreversible* if it has atleast one Garden-of-Eden configuration and CA is surjective if and only if the CA is bijective [138]. Thus the existence of non-reachable configuration infringes the injectivity property of CA. Moreover, a CA with injective property follows surjective property, but the converse is not true [114]. The injectivity and the surjectivity are not valid [139] for the CAs which are defined on hyperbolic plane [84, 85]. Some more seminal work on Garden-of-Eden and its extension are reported in [140–145].

2.1.2.3 Conservation Laws

Reversible cellular automata obey no loss of information. Besides reversibility, various conservation laws that are common in physics may also be exhibited by CAs [146–148]. Among them, number conservation is the most studied property in CA. In number conserving CAs, the count of different states in initial configuration remains unchanged during the evolution of CA. This class of cellular automata was first studied in [38] and revisited in [100,147–149]. Properties like universality, reversibility, etc. of such CAs have also been studied to analyze the global behaviour of this class of CAs [150–153]. Number conserving CAs have great contributions in highway traffic [38,154–159]; these automata are also used in particle conservation [160, 161].

In 2-dimension, the number conservation is supported by CA with Margolus neighborhood [162]. Besides number conserving CA, additive conserved quantities in CA is introduced and studied by Hattori and Takesue [163]. Non-uniform (where all cells of CA are not forced to follow the same local rule) number conserving CAs [164] have been first studied in [165] where ECA is considered as CA model. This work reports a linear time decision algorithm to figure out whether the given non-uniform CA is number conserving or not. The designing approach of non-uniform number conserving CA for a given number of cells is also reported here. The number conservation property has also been studied under asynchronous update [166–168].

Study of existence of *chaos* in CA [169] has also gained significant attention over the years. Some seminal works on the chaotic behaviour of CA are reported in [170–174]. Some CAs have the properties such that any initial configuration having more 1s (resp. 0s) than 0s (resp. 1s) must be attracted by a fixed point with all 1s (resp. 0s) configuration. This property is called as *density classification*; density classification is one of the most studied computational domains of CAs [175–180]. Another aspect is the *synchronization* problem, as studied in [181–186].

Traditionally CA is infinite and uniform (all cells of a CA follow same transition function or rule). We next revisit the developments of CAs by imposing restriction on lattice space $\mathscr{L}$ and injecting *non-uniformity* in cells' local rules.

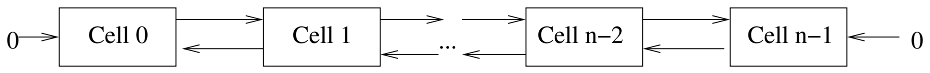

Figure 2.4: Null boundary CA

2.1.3 Finite Cellular Automata and Non-uniformity

Traditionally the cellular space $\mathscr{L}$ of CA is infinite and the sizes of the configurations are also infinite. However for many work, specially for practical implementation, restriction is imposed on the number of cells of CA and such CA must have boundaries.

Definition 2.3 *A CA ($\mathscr{L}, S, r, \mathcal{R}$) is called as a finite cellular automaton if $\mathscr{L}$ is finite.*

In finite cellular automaton, the evolution of CA is a finite sequence of configurations where the size of configurations are also finite. For 1-dimensional CA, $\mathscr{L} = \{0, 1, 2, \cdots, n-1\}$ where n is the number of cells in a configuration. For finite CAs, the boundary cells are treated specially. Two boundary conditions are generally considered - *open* boundary and *periodic* boundary. The former one follows some fixed states at the missing neighbors of the extreme cells and the later one uses the states of some extreme cells at the neighbors of the extreme cells. In open boundary condition, the most popular choice is *null* boundary [187] where the missing neighbors are assigned as state value 0 (null) (Figure 2.4). In periodic boundary CAs, the missing neighbor of the rightmost cell is the state value of left most cell and the missing neighbor of leftmost cell is the state value of right most cell (Figure 2.5). Like 1-dimensional CAs, the higher dimensional CAs with periodic boundary CAs have also been studied [81, 91, 188]. There exist some more variants on boundary conditions - such as *adiabatic* boundary, *reflexive* boundary and *intermediate* boundary. Adiabatic boundary assigns the state value of the corresponding extreme cell at its missing neighbors; in reflexive boundary the missing neighbors follows the state value of the immediate neighbors of boundary cells - for example the missing neighbor of the rightmost cells follows the state value of the left neighbor of the rightmost cell. In intermediate boundary [189] the missing neighbor of the boundary cell uses the state value of next to next neighbor's of the boundary cell. Another boundary type is stochastic boundary [190] where the missing neighbor uses the

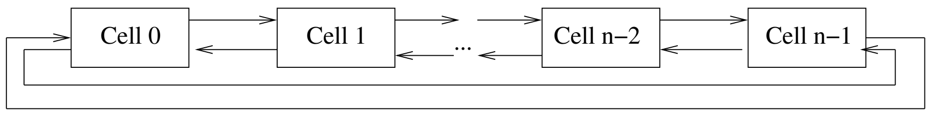

Figure 2.5: Periodic boundary CA

state in a stochastic manner.

Traditionally, cellular automata maintain three foundational properties which are as follows.

- *uniformity:* same rule is followed by every cell of that CA.
- *locality:* the rule of the CA acts locally and cells of the CA follow similar neighborhood dependency.
- *synchronicity:* the CA updates state values of all cells' simultaneously in each discrete time step.

Relaxations on the above mentioned properties can lead to different variants of CAs [115]. A CA violating synchronization in the update is called as *Asynchronous cellular automata* (ACAs) [191]. The cells of CA may use different neighborhood dependency i.e. a CA cell can have varying number of neighbors; this concept gave birth to *Automata Network* [192]. As stated before, *Non-uniform cellular automaton* does not enforce *uniformity* in applying the local rules at cells, i.e., cells of this type of CA use different local transition rules.

The concept of non-uniform cellular automata was first introduced in 1986 (see [97]) where the focus was on group properties of 1-dimensional finite CA under null and periodic boundary conditions. Some ground breaking work on non-uniform CAs are mentioned in [44–46, 193–195]. Recent research focus on special types of non-uniform CAs where the CA cells allow non-uniformity in applying the local rules along with the neighborhood structure and the global properties of non-uniform CAs have also been studied [164, 196–198]. We next focus on non-uniform 1-dimensional CAs.

Definition 2.4 *A non-uniform cellular automaton is a quadruple $(\mathscr{L}, S, r, \mathcal{R}_i)$ where $\mathscr{L} \subseteq \mathbb{Z}$, $i \in \mathscr{L}$, r is the radius, and $\mathcal{R}_i : S^{2r+1} \rightarrow S$ is the rule of cell i.*

The set of local rules $\{\mathcal{R}_i\}_{i\in\mathbb{Z}}$ induces the global transition function G and therefore, G can be defined as follows.

$$y_i = G(\mathtt{x})|_{i\in\mathscr{L}} = \mathcal{R}_i(x_{i-r}, x_{i-r+1}, \cdots, x_{i+r-1}, x_{i+r}) \tag{2.4}$$

When the cellular space $\mathscr{L}$ is finite, then the automaton $(\mathscr{L}, S, r, \mathcal{R}_i)$ is defined as a sequence of rules, defined as the *rule vector* of the CA.

Definition 2.5 *The rule vector of an n-cell non-uniform CA $(\mathscr{L}, S, r, \mathcal{R}_i)$ is $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1}\rangle$, where $|\mathscr{L}| = n$ and $\mathcal{R}_i$ is the rule used by cell $i \in \mathscr{L}$.*

If $\mathcal{R}_i$ ($\forall i \in \mathscr{L}$) is an ECA rule, then the automaton $(\mathscr{L}, S, r, \mathcal{R}_i)$ with finite size n such that $|\mathscr{L}| = n$ is called as *non-uniform ECA*. For example, let us consider $\mathscr{L} = \{0, 1, 2, 3\}$ and the rule vector $\mathcal{R} = \langle 6, 86, 15, 68 \rangle$ where cell 0 uses ECA rule 6. Similarly, cells 1, 2 and 3 use respectively 86, 15 and 68. The rule vector $\langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ can also be called as n-cell CA.

An n-cell non-uniform ECA is called as linear/additive CA if all rules $\langle \mathcal{R}_i \rangle_{0 \le i \le n-1}$ are linear/additive. A linear non-uniform ECA of size n can be represented as a binary characterization matrix of order n. Let us consider A to be the characterization matrix. The i^{th} row of that matrix represents the dependency of cell i to its neighboring cell j. Thus,

$$A[i,j] = \begin{cases} 1, \text{if the next state of cell } i \text{ depends on the present state of cell } j \\ 0, \text{otherwise} \end{cases}$$

As we consider 3-neighborhood automata, $A[i, j]$ is 1 when $j \in \{i - 1, i, i + 1\}$. Let us consider a 4-cell linear CA $\langle 60, 240, 150, 240 \rangle$ with null boundary condition. Here at the 0^{th} row, since rule 60 is considered, we get $A(0, 0)$ and $A(0, 1)$ as 1 and 0 respectively; note that for rule 60, the cell is dependent on its left neighbor and itself. As null boundary has been considered, the left neighboring cell's state of cell 0 is always 0; the same condition is to be maintained for the right neighbor of cell 3. Thus, the characterization matrix corresponding to the CA is given as:

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

Similar to linear CAs, we can get complemented CA; here at least one rule in $\mathcal{R}_i$s ($0 \le i \le n - 1$) must be a complemented rule, while the remaining are linear. As an example, we can consider the CA $\langle 60, 15, 150, 240 \rangle$ where only cell 1 follows complemented rule 15. As before, a complemented CA can also be represented by matrix algebra and inversion vector. The inversion vector is defined as mentioned below.

$$F_i = \begin{cases} 1, \text{if cell } i \text{ uses complemented rule} \\ 0, \text{otherwise} \end{cases}$$

Therefore, CA $\langle 60, 15, 150, 240 \rangle$ can represented as -

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad F = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

If an ECA is neither linear nor complemented, then such a CA is called as *non-linear*. In a non-linear ECA, at least one $\mathcal{R}_i$ is non-linear. For example, $\langle 60, 166, 150, 240 \rangle$ is a 4-cell non-linear CA.

Non-uniform CAs use mostly the same rule space used in elementary cellular automata. When we use non-uniform ECAs, we get 256^n possible cellular automata of size n. Non-uniform CA has gained attraction both for its theoretical aspect and its capability to be applied in real-life scenarios.

2.1.4 Applications of Cellular Automata

Since its birth, CA has been shown to be capable of simulating biological and physical systems, and handing real-life problems. We now aim to present a snapshot of application of CAs in real-life problems.

2.1.4.1 Role of CAs in Biological and Physical Systems

The introduction of CA was an effort to explain the biological event of reductionist theory. Application of CA in biological field was reported in [199]; here Lindenmayer uses CAs where the cells appear or disappear dynamically, and the objective was to model the growth of filamentary organisms. Such a system is called as L-system, and has great contributions in modeling plant life [200].

Though the modeling of artificial life was initiated by von Neumann, the implementation of the construction was done by Pesavento [201]. Besides, CA can be modelled as to study the spontaneous emergence of self-replicating systems [202] - each cell of that CA is used to (i.) store the state of the cell and (ii.) indicate the bonding of the cell with its neighbors (left or right). Some notable work of CA in connection to biology are discussed in [203–205].

CA also has a strong application in physics [206]. Some contributory work of CA in connection to physical phenomena are listed in [3–14]. CA has significant contributions in natural computing [15, 16] as well as in physical systems; some of them are as follows. CA's contribution in immune system are listed in [17–21]. CA can be used in detection of genetic disorder of cells [22], to study the nature of fish migration [23], population growth of vegetables [24, 25]. CA model can be used to control forest firing [26–30]; CA has contributions in managing traffic control system, urban and crowd [31–34, 36–40].

2.1.4.2 Contribution of CAs in Parallel Processing

One of the key features of CA is the ability to use parallelism to solve complex problems, as discussed in the work of Toffoli and Margolus [78]. Prior work also shows the efficacy of CA's power in parallelism - CAs can act as parallel processor [207], multiplier [208, 209]. CA can be utilized for prime number sieves [210], sorting [211], fault tolerant computing machine [212], nanometer-scale classical computer [213] etc.

2.1.4.3 CA as Source of Randomness

CAs are a good design choice for researchers who love to deal with hardware implementation, mainly due to the key properties of simplicity, parallelism and locality; besides, CA is good source of pseudo randomness [214] which makes CA a distinguishable tool in domains like VLSI [44,45,61,193,215,216], cryptography [217–221]. The pseudo randomness property of CAs is mainly correlated with cryptography. CA can generate pseudo random number and pseudo random pattern. In case of pseudo random *number* generation (PRNG), ECA rule 30 has a great contribution [222]. Some other notable work on pseudo random number generation are listed in [223–229]. It is also observed that by increasing the state count to *three*, we can get good quality PRNGs [230,231]. This has instigated the researchers to extend the number of states up to *ten*, and cellular automata can be used as decimal PRNGs [52]. It has already been shown that decimal cellular automata based PRNGs [232] are at least as good as SFMT19937-64 in terms of the empirical tests (where SFMT19937-64 is the best ranked Pseudo random number generator). CAs also contribute to pseudo random *pattern* generation (PRPG) where non-uniform finite cellular automata play a key role [45, 233]. A pattern is an n length configuration where the corresponding finite CA uses n cells. The main aim of generating PRPG is to figure out minimum cell length - [45] shows that a 45-cell non-linear ECA based PRPG is capable of beating the existing all PRPGs.

2.1.4.4 Contribution of CAs in VLSI Design

CA's properties are very appealing for the design of VLSI architecture; mainly we use non-uniform CAs for this target branch of applications. In particular, non-uniform CA based pseudo random pattern generator (PRPG) has been proposed for built-in self-test (BIST) in VLSI circuits [61]. As stated before, CA has been used as PRPG [45, 62, 215]; CA can also be modelled as BIST structure [45, 234–237]. Moreover, CA has a great role in test-pattern generator [45, 234, 238, 239]. Non-uniform ECAs are also capable of generating patterns without PPS (Prohibited Pattern Set) [45,233,240,241]. Using CA, a *UBIST* (Universal BIST) can be designed [242, 243]. Some additional notable work on CA based VLSI design are listed in [97, 244–247].

2.1.4.5 CAs in Secured Communication

Maintaining a large volume of data *securely* is a great challenge, so is transmitting them in a secured manner. CA plays an important role in this aspect through its application in the design of ciphers, encryption and decryption techniques, signature analysis and error correcting codes. Some of the notable work in cryptography are [53, 218, 221, 248–262]. An allied field is the study of error correcting codes; some notable work are mentioned in [263, 264, 264]. CA's contribution in signature analysis is also remarkable [234, 265–267].

2.1.4.6 Role of CA in Compression

CAs are effective for data compression. In Lafe's work [268], the significance of CA is shown not only for its data encryption ability, but also for its application in digital image compression. CA has notable utilization in text compression [269, 270], document compression [271], compression and decompression of images [272–276], video compression [277] and encompression [278, 279].

Other than the above mentioned traditional fields of applications, the use of cellular automata in social science [280–284] and medical science [285–298] are noteworthy.

2.2 Cellular Automata in Machine Learning

Cellular automata have great impact in the fields of machine learning, particularly, in pattern recognition and classification, clustering and image processing.

2.2.1 Pattern Recognition and Classification

CA, a decentralized computing tool, is cost-effective to be implemented in hardware and used in different domains. One of the important fields of its diverse applications is in pattern recognition and classification domain. A given pattern is either recognized or classified by CA structure. Research field demands competent identification of patterns of interest from its background.

The characteristics of CA attract researchers for designing systems to classify or recognize the patterns efficiently [44, 299, 300]. CA started its journey as pattern classifier and recognizer a long back as a language recognizer [301] and CAs contributions in language recognition is noteworthy [302–304]. CA can accept context-free language [302], non context-free language [305] and CA can also be thought of as context-sensitive language acceptor [101]. In Mahajan's dissertation [303], it is reported that several types of CAs like real-time one way

CA, one way CA, real time CA, linear CA etc. can play the role of language recognizer. Besides the role as language recognizer, the ability of cellular automata for classification problem had already been addressed in Tzionas's works. One such work is that a multi-valued pattern classification scheme had been proposed which uses the hybrid model of 2-dimensional additive CA and a single layer perceptron architecture [306] and in another work [307], CA based pattern classifier on nearest neighborhood discriminant has been reported.

By introducing the associative memory, the time to recognize the pattern from the available pool of patterns is independent of the total count of patterns available in that pool. The sparse structure of CA is justified to design CA based associative memory and such model play a key role in pattern recognition [169, 299, 300, 306, 308–310]. In [311], it was observed that MACA (Multicycle Attractor CA), a special class of CA behaves like natural classifier and effective for VLSI applications. MACA's unique state transition behavior leads the researchers to use MACA for modeling associative Memory and MACA can be used as pattern classifier [312]. The unique property of MACA state transition behavior is formalized based on a new concept of Hash Family referred to as Hamming Hash Family (HHF) [313]. Some of the popular works on MACAs (multiple attractor CAs) based design where CAs are used to model associative memory are mentioned in [314–316]. In Maji's dissertation [317], the complexity of MACA based pattern classification algorithm has been improved from $O(n^3)$ to $O(n)$, where n is the number of bits in a pattern; this MACA based classifier has great contributions in data mining, image compression, fault diagnosis, etc. Another class of CAs named as generalized MACA (GMACA) has been used for pattern classification and it performs better than Hopfield network [318]. Using the concept of fuzzy CAs, efficient classifier can be designed [319, 320].

Non-uniform non-linear ECAs also have been explored for design of efficient classifier. In [321], an efficient non-linear non-uniform ECAs based pattern classifier is reported where the candidate CAs are PE (Pseudo-Exhaustive) bits based single length cycle CA; the extended work on non-linear non-uniform ECAs based pattern classification reported in [64, 322]) are also noteworthy. Besides synchronous update, CA with asynchronous update also contributes in pattern classification [323]. An elegant CA based classification algorithm has been reported in [324] for satellite images. Some more remarkable work on CA based pattern classifier are mentioned in [325–328].

2.2.2 Image Processing

Similar to other fields, CA is also popular in the domain of image processing. CA is capable of handling all key tasks of image processing such as translation, zooming, rotation, segmentation, thinning, compression, edge detection and noise reduction, etc. (see [329, 330]). The features of 2-dimensional CAs mainly attract the researchers to solve the various issues in image processing [331]. In [329], CAs are trained to perform noise filtering, thinning and convex hulls. An advancement of [329] is reported in [332] where number cells states are increased to 3 and intensity of images are considered. Noise reduction and edge detection, two important aspects of image processing are handled by CAs [273, 333–339]. Besides CA's role in noise filtering, thinning, edge detection of images, CA based image compression technique is also reported in [273] for both grey level and colour images. CA performs image analysis efficiently [340]. CA based classifier can be used for improving the classification accuracy rate in satellite images [324]. In [341], a notable approach is presented to classify Wolfram's CAs based on the texture description captured in the evolved space-time diagrams. For characterizing different sort of textures of images, a linear CA based technique has been proposed in [342]. A texture descriptor which is generated by using corrosion modeling and cellular automata, is reported in [343] for classifying synthetic and natural texture images.

2.2.3 Clustering

Cellular automata have also been applied as effective tools for clustering. By conjugating with intelligence, CAs act efficiently for data clustering [344]. An artificial ant algorithm of clustering (A^4C) based on Ants Sleeping Model (ASM) has been proposed [345] where the corresponding ASM has been built on the principle of cellular automata in artificial life. In [346], an ant-based data clustering technique integrated with cellular automata is reported which is used to represent complex datasets in meaningful visual clusters. Generalized cellular automata (GCA) based clustering is a stochastic process over the configuration space on a GCA array [347]. In [348], a seminal clustering technique has been proposed by connecting social segregation models based on CA and ant clustering algorithm. Here, linear cellular automaton is used as CA model. Recently, a stochastic cellular automata based clustering algorithm, *SCA-clustering* has been devised which does not use any distance metric whereas the generation of clusters is influenced by the heat transfer process in nature [349]. A non-uniform ECA based clustering is reported in [350] where the candidate CA is MACA, a special class of CA which shows its capability of forming clusters using binary patterns by considering hamming distance as similarity metric [350].

Next, we introduce some concepts, notations, definitions and terminologies mainly related to ECAs which are relevant to the work done in the subsequent chapters.

Table 2.3: A Non-uniform CA with four cells $\langle 10, 10, 240, 20 \rangle$

PS (RMT)	111 (7)	110 (6)	101 (5)	100 (4)	011 (3)	010 (2)	001 (1)	000 0	Rule
	0	0	0	0	1	0	1	0	10
	0	0	0	0	1	0	1	0	10
NS	1	1	1	1	0	0	0	0	240
	0	0	0	1	0	1	0	0	20

2.3 Definitions and Terminologies

This section notes down some definitions and terminologies in cellular automata which are required for our discussion. As this work studies the cyclic behaviour of 1-dimensional CA with *finite* cells, let us recall some fundamental concepts in ECAs - 1-dimensional CA with binary state and 3-neighborhood architecture.

Let the size of CA be n; such CA is called as either an n-cell CA or a CA of size n. The cells are denoted as cell 0 to cell $n-1$. In ECA, every cell i contains its present state value x_i as either 0 or 1. Therefore, y_i, the next state of cell i is computed as $y_i = \mathcal{R}_i(x_{i-1}, x_i, x_{i+1})$ where x_{i-1} and x_{i+1} are the left and right neighbors of cell i respectively; $\mathcal{R}_i$ is the rule applied on cell i. This dissertation considers null boundary condition, that is, $x_{-1} = x_n = 0$.

2.3.1 Rule Min Term, Rule and Rule vector

It is already mentioned in Table 2.1, `PS` refers an input (x_{i-1}, x_i, x_{i+1}) to $\mathcal{R}_i$ and `NS` represents the output of $\mathcal{R}_i(x_{i-1}, x_i, x_{i+1})$. Let $\mathcal{R}_i[\mathtt{r}]$ denote $\mathcal{R}_i(x_{i-1}, x_i, x_{i+1})$ where $\mathtt{r}$ is the decimal equivalent of (x_{i-1}, x_i, x_{i+1}). Therefore, $\mathcal{R}_i(1,1,1)\mathcal{R}_i(1,1,0)\cdots\mathcal{R}_i(0,0,0)$ represents a 8-bit binary string "$\mathcal{R}[7]\mathcal{R}[6]\cdots\mathcal{R}[0]$" which can be called as *rule*; its equivalent decimal number is also called as rule (see Table 2.3). Next, we introduce *Rule Min Term* (RMT).

Definition 2.6 *The sequence of $x_{i-1}x_ix_{i+1}$ with respect to $\mathcal{R}_i(x_{i-1}, x_i, x_{i+1})$, where $\mathcal{R}_i$ is a next state function and $x_{i-1}, x_i, x_{i+1} \in \{0,1\}$, is termed as Rule Min Term (RMT). An RMT is generally presented by its decimal value $\mathtt{r}_i = 4x_{i-1} + 2x_i + x_{i+1}$. Therefore, $\mathcal{R}_i[\mathtt{r}_i] = \mathcal{R}_i(x_{i-1}, x_i, x_{i+1})$.*

Let us represent RMT (x_{i-1}, x_i, x_{i+1}) as a sequence $x_{i-1}x_ix_{i+1}$. The 001 of the first row of Table 2.1 is the RMT 1, next state against which is 1 for rule 10 and 0 for rule 240. As we write $\mathcal{R}_i[\mathtt{r}_i]$ denotes its next state, therefore, $10[1] = 1$, $240[1] = 0$ (see Table 2.1). This $\mathtt{r}_i$ is the RMT of cell i. RMTs 0, 2, 4 and 6 are called as *even* RMTs whereas 1, 3, 5 and 7 are called as *odd* RMTs.

The concept of *rule vector* is already introduced in Section 2.1.3 (see Definition 2.5). In a rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ of an n-cell non-uniform ECA, every $\mathcal{R}_i$ is an ECA rule. For designing CA rule vector, out of 256 rules, the applicable rules at cell 0 and cell $n-1$ are restricted to *sixteen*. The reason is followed as: due to the null boundary condition, the next states for cell 0 and cell $n-1$ are determined only by the present states $(0, x_0, x_1)$ and $(x_{n-2}, x_{n-1}, 0)$ respectively, as the left cell and right cell will never be *one* for them. Thus, we consider $\mathcal{R}_0(1, x_0, x_1) = \mathcal{R}_{n-1}(x_{n-2}, x_{n-1}, 1) = 0$ (see for example, the first row and the last row of Table 2.4). Therefore, each of the terminal cells considers only $2^4 = 16$ distinct rules. Nevertheless, if we take any rule for these two terminal cells for which this condition is not satisfied, then also its behaviour will be equivalent to these 16 rules. For example, for cell 0, behaviour of ECA 150(10010110) and ECA 6(00000110) are equivalent.

A rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ can also be rewritten as $\mathcal{R} = \langle \mathcal{R}_0^{a_0}, \mathcal{R}_1^{a_1}, \cdots, \mathcal{R}_m^{a_m} \rangle$; it means that first a_0 cells use $\mathcal{R}_0$, next a_1 cells use $\mathcal{R}_1$, and so on. Obviously, $a_0 + a_1 + \cdots + a_m = n$. For example, let us consider an 8-cell CA $\mathcal{R} = \langle 9, 233, 233, 42, 90, 90, 90, 20 \rangle$. Here, the *first* cell uses rule 9. Next, cells 1 to 2 use rule 233, cell 3 uses rule 42 and cells 4 to 6 use rule 90. The last cell of $\mathcal{R}$ uses rule 20. Therefore, $\mathcal{R}$ can be rewritten as $\langle 9, 233^2, 42, 90^3, 20 \rangle$. Similarly, $\mathcal{R} = \langle \cdots, (\mathcal{R}_i^{a_i}, \mathcal{R}_{i+1}^{a_{i+1}}, \cdots, \mathcal{R}_{i+j}^{a_{i+j}})^p, \cdots \rangle$ represents that the substring of rules $\langle \mathcal{R}_i^{a_i}, \mathcal{R}_{i+1}^{a_{i+1}}, \cdots, \mathcal{R}_{i+j}^{a_{i+j}} \rangle$ repeats for p times consecutively in the rule vector. Let us explain the representation with some example. In an 10-cell CA with rule vector $\mathcal{R} = \langle 6, 44, 128, 128, 90, 44, 128, 128, 90, 68 \rangle$ a substring of rules $\langle 44, 128^2, 90 \rangle$ repeats consecutively for 2 times. Therefore, $\mathcal{R}$ can be rewritten as $\mathcal{R} = \langle 6, (44, 128^2, 90)^2, 68 \rangle$.

2.3.2 Reachable Configuration, RMT Sequence and Cycles

By applying rules at all cells at a time step, we get a *sequence* of updated state values of cells (from cell 0 to cell $n-1$) which is called as *configuration*. Let `x` be a configuration which can be represented as $(x_i)_{0 \le i \le n-1}$ where each $x_i \in \{0, 1\}$.

Table 2.4: A 4-cell CA $\langle 6, 86, 15, 68 \rangle$

PS (RMT)	111 (7)	110 (6)	101 (5)	100 (4)	011 (3)	010 (2)	001 (1)	000 (0)	Rule
	0	0	0	0	0	1	1	0	6 ($\mathcal{R}_0$)
	0	1	0	1	0	1	1	0	86 ($\mathcal{R}_1$)
NS	0	0	0	0	1	1	1	1	15 ($\mathcal{R}_2$)
	0	1	0	0	0	1	0	0	68 ($\mathcal{R}_3$)

Any substring of a configuration is called as *sub-configuration*. We also define the special cases of sub-configurations which are used in this dissertation.

Definition 2.7 *The prefix sub-configuration of* $\mathtt{x}$ *for* k *cells given by* $\mathtt{prefix}(\mathtt{x}, k) = x_0 x_1 \cdots x_{k-1}$.

Definition 2.8 *The suffix sub-configuration of* $\mathtt{x}$ *for* k *cells given by* $\mathtt{suffix}(\mathtt{x}, k) = x_{n-k} x_{n-k+1} \cdots x_{n-1}$.

Let us consider a configuration $\mathtt{x} = 10101100$ of an 8-cell cellular automaton; the sub-configuration 10101 is the *prefix* sub-configuration of $\mathtt{x}$ for 5 cells and given as $\mathtt{prefix}(10101100, 5) = 10101$. In the same way, we can get $\mathtt{suffix}(10101100, 5) = 01100$.

Now, the switching from current state values of all cells to the next state values of cells is nothing but the transition of configurations to evolve. Thus, the evolution of CA is determined by a *global transition function*. Let us consider G_n be the global transition function of an n-cell CA such that $G_n : C_n \to C_n$ where $C_n = \{0, 1\}^n$ (as $S = \{0, 1\}$ and $|\mathscr{L}| = n$) represents configuration space of an n-cell CA. Therefore, CA can be thought of as a function G_n. Hence, $\mathtt{y} = G_n(\mathtt{x})$ where $\mathtt{x}, \mathtt{y} \in C_n$ and $\mathtt{y} = (y_i)_{0 \le i \le n-1}$, and an y_i is computed as $\mathcal{R}_i(x_{i-1}, x_i, x_{i+1})$. Using G_n, next we explain *reachability* and *cycle*.

Definition 2.9 *Let* $\mathtt{x}, \mathtt{y} \in C_n$ *be two configurations of a CA* $G_n : C_n \to C_n$. *The configuration* $\mathtt{y}$ *is called reachable from* $\mathtt{x}$ *if there exists* $l \in \mathbb{N}$ *such that* $G_n^l(\mathtt{x}) = \mathtt{y}$*; otherwise it is not reachable from* $\mathtt{x}$.

If for all $\mathtt{x} \in C_n$, $G_n^{l'}(\mathtt{x}) \neq \mathtt{y}$, *where* $l' \in \mathbb{N}$, *then* $\mathtt{y}$ *is a non-reachable configuration; otherwise, it is reachable (from some) configuration.*

Here, Figure 2.6 represents the *configuration transition diagram* of CA $\langle 10, 10, 240, 20 \rangle$. The configuration transition diagram of an n-cell CA represents the evolution of that CA over time and also the reachability among the CA's configurations. In Figure 2.6, configuration 1010 is *reachable* from configuration 1111, but configuration 0110 is *not reachable* from it. Moreover, configuration 0110 is *non-reachable* configuration as it is not reachable from any configuration. If $\mathtt{y}$ is reachable from $\mathtt{x}$, then $\mathtt{x}$ is the *source* configuration and $\mathtt{y}$ is the *destination* configuration. Let us introduce *successor* and *predecessor*. Here, in $\mathtt{y} = G_n(\mathtt{x})$, $\mathtt{y}$ is the *successor* of $\mathtt{x}$, and $\mathtt{x}$ is the *predecessor* of $\mathtt{y}$. In Figure 2.6, configuration 1010 is the successor of configuration 1111. Therefore, we can find l^{th} ($l \in \mathbb{N}$) successor of a configuration $\mathtt{x}$ by applying G_n on $\mathtt{x}$ for l time. In this way, the reachability of any two configurations can be decided - such problem is known as Configuration REachability Problem (CREP), which was already studied in the domain of linear/additive CAs [56].

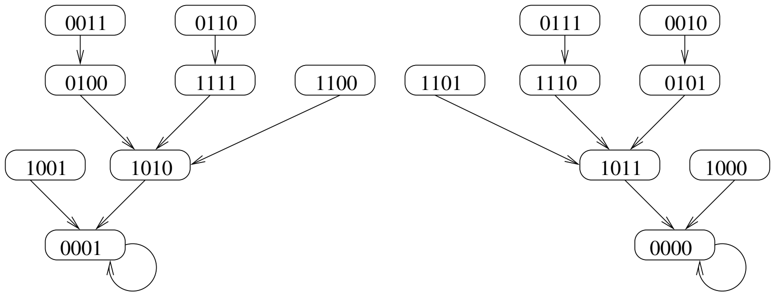

Figure 2.6: Configuration transition diagram of CA $\langle 10, 10, 240, 20\rangle$

During evolution, when configuration $\mathtt{y}$ is reachable from another configuration $\mathtt{x}$ such that $G_n(\mathtt{x}) = \mathtt{y}$, then the configuration $\mathtt{y}$ can also be thought as a *sequence of RMTs*. Let us find the connection between a configuration and its predecessor by the *sequence of RMTs*.

$$\begin{aligned} G_n(\mathtt{x}) &= G_n((x_0x_1\cdots x_{n-1})) \\ &= \mathcal{R}_0(0,x_0,x_1)\mathcal{R}_1(x_0,x_1,x_2)\cdots\mathcal{R}_{n-1}(x_{n-2},x_{n-1},0) \\ &= \mathcal{R}_0[\mathtt{r}_0]\mathcal{R}_1[\mathtt{r}_1]\cdots\mathcal{R}_{n-1}[\mathtt{r}_{n-1}] \\ &= (y_0y_1\cdots y_{n-1}) = \mathtt{y} \end{aligned}$$

Here, $\mathtt{r}_0, \mathtt{r}_1, \cdots, \mathtt{r}_{n-1}$ are the RMTs which are the decimal equivalents of $(0x_0x_1)$, $(x_0x_1x_2)$, $\cdots$, $(x_{n-2}x_{n-1}0)$ combinations respectively and $\mathtt{x}$ is the *predecessor* of $\mathtt{y}$. Hence, the configuration $\mathtt{y} = (y_0y_1\cdots y_{n-1})$ can also be represented as a sequence of RMTs $(\mathtt{r}_0\mathtt{r}_1\cdots\mathtt{r}_{n-1})$; such sequence is called as *RMT Sequence* (RS).

Definition 2.10 *Let $\mathtt{x} = (x_i)_{0\le i\le n-1}$ be a configuration of an 1-dimensional n-cell CA. The RMT sequence of $\mathtt{x}$, denoted as $\rho(\mathtt{x})$, is $(\mathtt{r}_i)_{0\le i\le n-1}$ where $\mathtt{r}_i$ is the RMT of cell i.*

Let us consider $\mathtt{x} = 0011$ be a configuration of an 4-cell ECA. Therefore, the corresponding RMT sequence is $\rho(\mathtt{x}) = 0136$. In this dissertation, RMT sequence and RS are used interchangeably. Thus, any *reachable* configuration can alternatively be represented as a *RMT Sequence*. Moreover, in a RS any two consecutive RMTs are *related*. For ECAs, $\mathtt{r}_i$, the RMT for cell i considers an 3-bit window which takes the values as in a sequence of (x_{i-1}, x_i and x_{i+1}). To find $\mathtt{r}_{i+1}$, the same window shifts 1-bit right over the configuration and the window is loaded with x_i, x_{i+1} and x_{i+2}. Therefore, $\mathtt{r}_{i+1}$ can be either $\{2\mathtt{r}_i \pmod 8$ or $2\mathtt{r}_i + 1 \pmod 8\}$ (in such case the present state combination can be either $(x_i, x_{i+1}, 0)$ or $(x_i, x_{i+1}, 1)$). For example if $\mathtt{r}_i$ is 6 (110), therefore, $\mathtt{r}_{i+1}$ can be either 4 (100) or 5 (101). Thus we get the consecutive RMTs for cells i and $(i+1)$

Table 2.5: Relation between RMTs of cell i and cell $(i+1)$

$\mathtt{r}_i$	0	1	2	3	4	5	6	7
$\mathtt{r}_{i+1}$	$0,1$	$2,3$	$4,5$	$6,7$	$0,1$	$2,3$	$4,5$	$6,7$

and Table2.5 refers desired information. Next, we report some basic properties of RMTs.

Definition 2.11 *Two RMTs* $\mathtt{r}_i$ *and* $\mathtt{r}_j$ *(*$\mathtt{r}_i \neq \mathtt{r}_j$*) are said to be sibling if* $\mathtt{r}_i = 2\mathtt{r} \pmod 8$ *and* $\mathtt{r}_j = 2\mathtt{r}+1 \pmod 8$ *where* $\mathtt{r}$ *is a RMT.*

RMTs 0 and 1 are sibling RMTs. Similarly, RMTs 2 and 3, RMTs 4 and 5, and RMTs 6 and 7 are sibling to each other.

Definition 2.12 *Two RMTs* $\mathtt{r}_i$ *and* $\mathtt{r}_j$ *(*$\mathtt{r}_i \neq \mathtt{r}_j$*) are said to be equivalent if* $2\mathtt{r}_i \pmod 8 = 2\mathtt{r}_j \pmod 8$ *and* $2\mathtt{r}_i+1 \pmod 8 = 2\mathtt{r}_j+1 \pmod 8$*.*

Therefore, RMTs 0 and 4 are equivalent to each other. Similarly, RMTs 1 and 5, RMTs 2 and 6, and RMTs 3 and 7 are also equivalent to each other.

Definition 2.13 *In a CA, cell* i *restores its previous state if* $\mathcal{R}_i(x_{i-1}, x_i, x_{i+1}) = \mathcal{R}_i[\mathtt{r}_i] = x_i$*. Such* $\mathtt{r}_i$ *is called as self replicating RMT.*

For example, rule 15 (see Table 2.1) has *four* self replicating RMTs - 2, 3, 4 and 5. The number of self replicating RMTs for a rule plays a major role in cycle formation. Next we focus on *cycles* of CA.

Let $\mathtt{x}, \mathtt{y} \in C_n$ be two configurations of CA G_n. Let $\mathtt{y}$ is *reachable* from $\mathtt{x}$ and $\mathtt{x}$ is also *reachable* from $\mathtt{y}$, then they are in same cycle.

Definition 2.14 *A configuration of a CA* G_n *is said to be cyclic if* $\mathtt{x} = G_n^l(\mathtt{x})$ *for* $l \in \mathbb{N}$*.* $\mathtt{x}$ *is also called as a member of a cycle of length* l*. If a configuration is not a member of cycle, then such configuration is called as acyclic configuration.*

A cycle of length l *is a collection of* l *cyclic configurations where each configuration has exactly one predecessor. Such cycle is called as* l*-cycle.*

Figure 2.6 presents the two cyclic configurations - 0000 and 0001; each of them represent a cycle of length *one*. Therefore, CA $\langle 10, 10, 240, 20\rangle$ refers *two* 1-cycle. Let $\mathtt{c}$ denote a cycle of length l of an n-cell CA. Therefore, $\mathtt{c}$ can be represented as $\{\mathtt{x}, G_n(\mathtt{x}), G_n^2(\mathtt{x}), \cdots, G_n^{l-1}(\mathtt{x})\}$. In this way, we can get $\mathtt{c}_1, \mathtt{c}_2, \cdots, \mathtt{c}_w$ cycles; obviously, $|\mathtt{c}_1| + |\mathtt{c}_2| + \cdots + |\mathtt{c}_w| \leq |C_n|$.

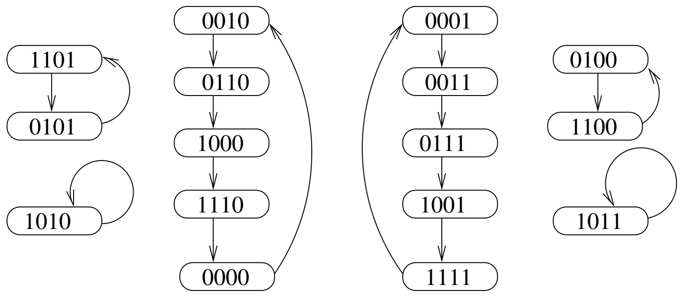

Figure 2.7: Configuration transition diagram of CA $\langle 6, 86, 15, 68\rangle$

Definition 2.15 *The cyclic space of an n-cell CA is the collection of all cycles generated in that CA.*

Let $\mathbb{C}_{\mathcal{R}}$ be the cyclic space of an n-cell CA $\mathcal{R}$. Let $\mathsf{c}_1, \mathsf{c}_2, \cdots, \mathsf{c}_w$ be the generated cycles in $\mathcal{R}$. Therefore, $\mathbb{C}_{\mathcal{R}} = \cup_{i=1}^{w} \mathsf{c}_i \subseteq C_n$.

Definition 2.16 *Let us consider an n-cell CA with global transition function $G_n : C_n \to C_n$ and that CA generates $\mathsf{c}_1, \mathsf{c}_2, \cdots, \mathsf{c}_w$ cycles. Such CA is reversible if $\mathsf{c}_1 \cup \mathsf{c}_2 \cup \cdots \cup \mathsf{c}_w = C_n$ and $\mathsf{c}_1 \cap \mathsf{c}_2 \cap \cdots \cap \mathsf{c}_w = \varnothing$. Here, G_n is bijective. Otherwise, the CA is irreversible.*

Figure 2.7 represents a reversible CA which generates *six* cycles of different lengths. Next, we introduce the concept of cycle structure.

Definition 2.17 *[44, 60] Cycle structure (CS) of a CA ($\mathcal{R}$) is the collection of the number of cycles of the CA along with their lengths. It is defined as $\mathtt{CS}_{\mathcal{R}} = [\mu_1(l_1),\ \mu_2(l_2),\ \cdots,\ \mu_m(l_m)]$, where μ_i is the number of l_i-cycle and $l_1 > l_2 > \cdots > l_m$. Each $\mu_i(l_i)$ is called as cyclic component. Here, l_m and l_1 are called as largest and smallest cycle lengths respectively.*

In Figure 2.7, the cycle structure of the CA $\langle 6, 86, 15, 68\rangle$ is $[2(1), 2(2), 2(5)]$. Here, 2(1), 2(2) and 2(5) are the cyclic components. Here, 5 is the largest cycle length and 1 is the minimum cycle length. Cycle structure of CA has already been studied in the domain of linear CAs [44, 57–59, 312].

CAs can be classified based on the lengths of cycles. A CA $\mathcal{R}$ of size n is said to be *Maximal length* CA if $\mathtt{CS}_{\mathcal{R}} = [1(1), 1(2^n - 1)]$. Similarly, if $\mathtt{CS}_{\mathcal{R}} = [\mu(1)]$, then such CA is called as *Single length cycle* CA (see for example Figure 2.6 whose cycle structure is [2(1)]). If the cycle structure of a CA is represented as $[\mu(l)]$, then such CA is known as *equal length cycle* CA [351]. CA $\langle 10, 10, 240, 20\rangle$ is *equal length cycle* CA (see Figure 2.6).

As the aim of this dissertation is to study the cyclic properties of CA, non-uniform reversible ECAs are used very frequently in the subsequent chapters.

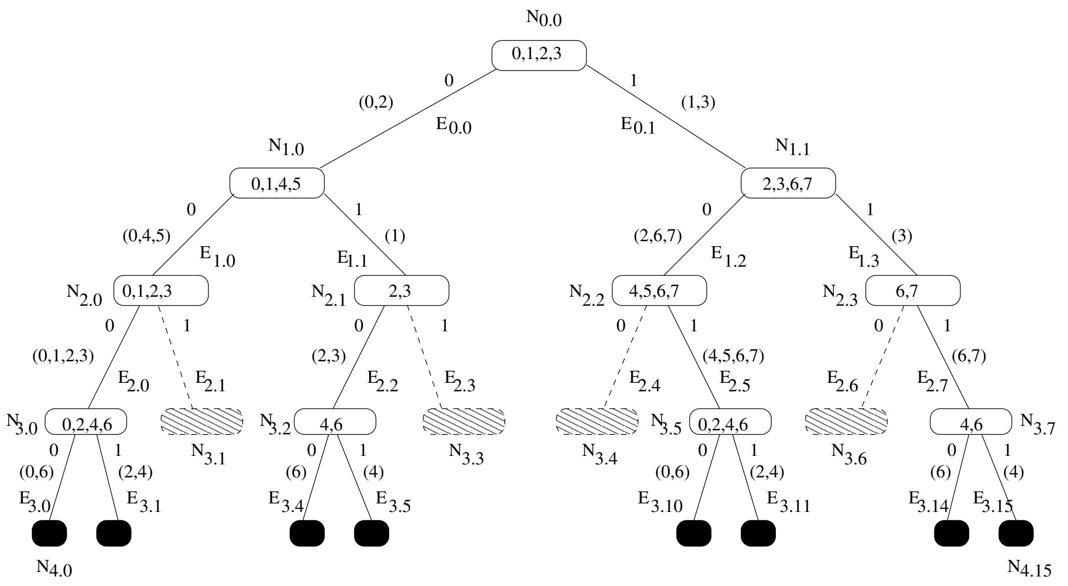

Figure 2.8: Reachability tree of CA $\langle 10, 10, 240, 20 \rangle$

Such ECAs represent linear and non-linear CAs both. Linear CAs are characterized by matrix algebra and polynomial theory [44] whereas specially for non-linear CA, *Reachability tree* [45] is used as the characterization tool. Therefore, let us introduce reachability tree.

2.4 Reachability Tree and Reversibility

Reachability tree [45, 64, 322] is a binary tree which describes the configurations along with their RMT sequences in pictorial representation. Reachability tree is a characterization tool for 1-dimensional CA, which is used for uniform/non-uniform and linear/non-linear CAs. This tree was first used for non-uniform ECAs with null boundary condition [352]. Later it was used for periodic boundary non-uniform ECAs also [46]. Though reachability tree first designed for binary CA, a generalized version has also been introduced in [52] under periodic boundary condition.

2.4.1 Reachability Tree

For an n-cell ECA, there are $n + 1$ levels starting from root (level 0) to leaf (level n). Each node of the tree is represented by $N_{i.j}$, where i $(0 \leq i \leq n)$ is the level index and j $(0 \leq j \leq 2^i - 1)$ is the node number at i^{th} level and each node represents two edges $E_{i.2j}$ and $E_{i.2j+1}$ are called as 0-edge and 1-edge where binary values 0 and 1 are described respectively. In reachability tree for null boundary CA, $\{0, 1, 2, 3\}$ and $\{0, 2, 4, 6\}$ are the sets of effective RMTs for $\mathcal{R}_0$ and $\mathcal{R}_{n-1}$ respectively. As this dissertation focuses on null boundary CA, let us define formally the reachability tree for a null boundary CA.

Definition 2.18 *Reachability tree of an n-cell CA $\langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_i, \cdots, \mathcal{R}_{n-1} \rangle$ under null boundary condition is a rooted and edge-labeled binary tree with $n+1$ levels. The j^{th} node at level i is denoted by $N_{i.j}$ $(0 \le i \le n,\ 0 \le j \le 2^i - 1)$. The edges of level i from $N_{i.j}$ $(i < n)$ are $E_{i.2j}$ and $E_{i.2j+1}$; $E_{i.2j}$ connects $N_{i.j}$ and $N_{i+1.2j}$, while $E_{i.2j+1}$ connects $N_{i.j}$ and $N_{i+1.2j+1}$. $E_{i.2j}$ and $E_{i.2j+1}$ from node $N_{i.j}$ are called as* 0-edge *and* 1-edge *respectively. $E_{i.2j}$ and $E_{i.2j+1}$ are labelled respectively $l_{i.2j}$ and $l_{i.2j+1}$. Therefore, $E_{i.2j} = (N_{i.j}, N_{i+1.2j}, l_{i.2j})$ and $E_{i.2j+1} = (N_{i.j}, N_{i+1.2j+1}, l_{i.2j+1})$. $N_{i.j}$, $l_{i.2j}$ and $l_{i.2j+1}$ are in fact multi-sets $(l_{i.2j}, l_{i.2j+1} \subseteq N_{i.j})$ maintaining the following relations:*

1. *[For root] $N_{0.0} = \{0, 1, 2, 3\}$.*
2. *$\forall \mathtt{r} \in N_{i.j}$, RMT $\mathtt{r}$ of $\mathcal{R}_i$ is in $l_{i.2j}$ (resp. $l_{i.2j+1}$), if $\mathcal{R}_i[\mathtt{r}] = 0$ (resp. 1). That means, $l_{i.2j} \cup l_{i.2j+1} = N_{i.j}$ $(0 \le i \le n-1,\ 0 \le j \le 2^i - 1)$.*
3. *$\forall \mathtt{r} \in l_{i.j}$, RMTs $2\mathtt{r} \pmod 8$ and $2\mathtt{r} + 1 \pmod 8$ of $\mathcal{R}_{i+1}$ are in $N_{i+1.j}$ $(0 \le i \le n-3,\ 0 \le j \le 2^{i+1} - 1)$.*
4. *[For level $n-1$] $\forall \mathtt{r} \in l_{n-2.j}$, RMT $2\mathtt{r} \pmod 8$ of $\mathcal{R}_{n-1}$ is in $N_{i+1.j}$ $(0 \le j \le 2^{n-1} - 1)$.*
5. *[For level n] Leaf node $N_{n.j} = \varnothing$, for any j, $0 \le j \le 2^n - 1$.*

The reachability tree for an n-cell CA is constructed in the given manner. The root node of reachability tree contains RMTs 0, 1, 2 and 3. From node $N_{0.0}$, two edges are formed where left edge places such RMT $\mathtt{r}$ from $\{0, 1, 2, 3\}$ where $\mathcal{R}_0[\mathtt{r}] = 0$. The remaining RMTs are placed on right edge of $N_{0.0}$. As RMT $\mathtt{r}$ is on the left edge of $N_{0.0}$, therefore, $N_{1.0}$ must have RMTs $2r \pmod 8$ and $2r + 1 \pmod 8$. In this way, $N_{1.1}$ also be constructed. By developing the tree level wise in this manner, we get level n, where the leaf nodes are empty. Only at level $n - 1$, the nodes store even RMTs as $\{0, 2, 4, 6\}$.

Figure 2.8 shows the reachability tree of CA $\langle 10, 10, 240, 20 \rangle$. For simplicity, we only show the distinct elements of each multi-set in the figure. Here, $N_{0.0} = \{0, 1, 2, 3\}$. Set $l_{0.0}$ (resp. $l_{0.1}$) contains RMTs $\mathtt{r}$ where $\mathtt{r} \in N_{0.0}$ and $\mathcal{R}_0[\mathtt{r}] = 0$ (resp. $\mathcal{R}_0[\mathtt{r}] = 1$). Thus $l_{0.0} = \{0, 2\}$ (since $\mathcal{R}_0[0] = \mathcal{R}_0[2] = 0$). Similarly, $l_{0.1} = \{1, 3\}$. Thus $E_{0.0}$ (0-edge at level 0) is labelled $l_{0.0}$. By progressing in this manner level wise, the corresponding tree is constructed. If an edge of reachability tree does not carry any RMT $\mathtt{r}$ when $l_{i.j} = \varnothing$ for some i and j, then the corresponding edge is known as *non-reachable*. In this tree, $l_{2.1}$ does not carry any RMT such that $l_{2.1} = \varnothing$. Thus $E_{2.1}$ non-reachable (dotted edge in Figure 2.8) and it can not connect any node of next level (such nodes are shaded in Figure 2.8).

The reachability tree of a CA gives the information about reachable and non-reachable configurations. An edge sequence from root to leaf (configuration) in the tree represents a reachable configuration. Such edge sequence is associated with some RMT sequence it means that every edge of that sequence carries some RMT. In Figure 2.8, edge sequence $E_{0.0}E_{1.1}E_{2.2}E_{3.4}$ refers configuration 0100 which is a reachable configuration and is associated with an RMT sequence 0136. To get reachable configurations, we have to traverse the tree up to level $n-1$, but the non-reachable configurations sometimes can be computed before reaching to level $n-1$. We can get a collection of non-reachable configurations using an edge sequence from root to a non-reachable edge $E_{i.j}$ where $i < n-1$, where 0- and 1-edges are replaced by 0 and 1 respectively, and the rest bits are filled up arbitrarily. In Figure 2.8, by the edge sequence $(E_{0.0}E_{1.0}E_{2.1})$, we can get the set $\{0010,\ 0011\}$ of non-reachable configurations. If in a reachability tree, there does not exist any non-reachable edge at any level, therefore, there is no non-reachable configuration which implies that the corresponding tree refers a reversible CA. As this dissertation studies the cyclic properties of CA, let us revisit the mechanism of generating n-cell reversible cellular automaton using reachability tree where the ECA rules are used as the candidate rules.

2.4.2 Reversible CAs

A short survey on the reversibility property of cellular automata mainly for infinite lattice size is reported in Section 2.1.2.2. The major works that had been done on reversibility are for infinite CAs; in higher dimensions, there is no decision algorithm for an arbitrary CA [87]. Thus, researchers are interested in 1-dimensional CAs with finite size.

It is already established that reversibility is undecidable in 2-dimensional CA [87, 133]; therefore, the researchers are also interested the same for 1-dimensional CAs. The reversibility in 1-dimension has already been studied for non-uniform CAs [197]. Besides, reversibility for finite CAs have been explored extensively [44, 46–48, 50, 51, 353–356]. The reversibility of finite CA addressed in [45,46,353] cover those CAs which follow non-linear rules also. Here, the non-linear rules are elementary rules and moreover, the cells are not forced to follow same rules. Therefore, these reversible CAs are non-linear as well non-uniform also. As we focus on finite CAs, so reversibility is checked for both null and periodic boundary conditions. Most of the CAs are binary [46, 50, 51]. Recently reversibility on d state cellular automata is mentioned in [357]. The concept of

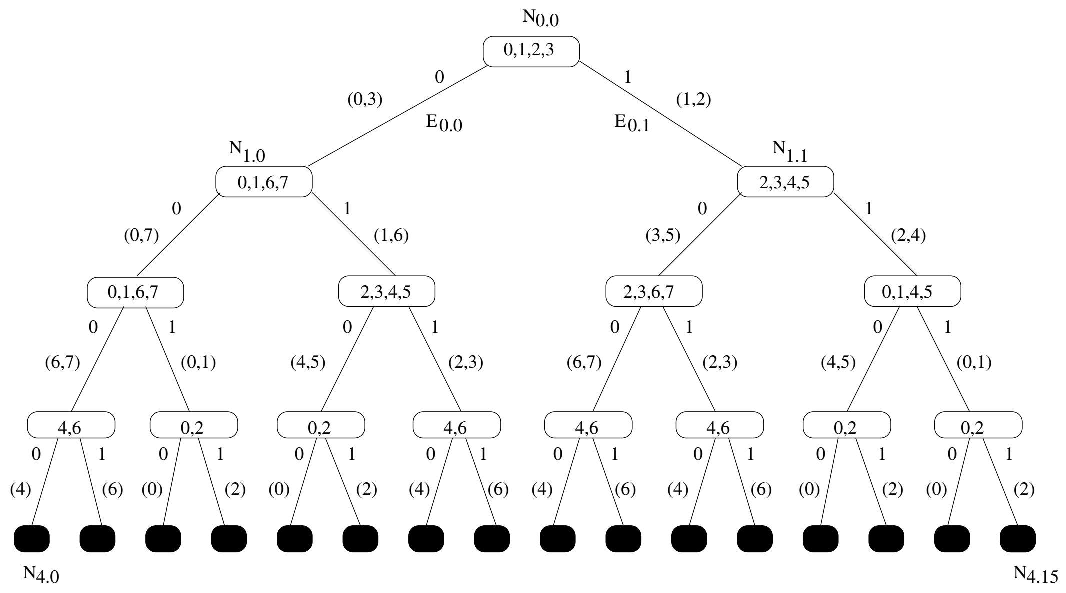

Figure 2.9: Reachability tree of CA $\langle 6, 86, 15, 68 \rangle$

reversibility and semi-reversibility of CAs and the relation between finite and infinite CAs in terms of reversibility are discussed in [52].

As this dissertation is on 1-dimensional CA with finite lattice size, let us discuss the mechanism of generating reversible CA using reachability tree [45,46]. To decide a CA is reversible or not by the help of reachability tree, the following theorem plays the vital role.

Theorem 2.1 *The reachability tree for a reversible CA is Complete. [45]*

In the reachability tree of the 4-cell reversible CA $\langle 6, 86, 15, 68 \rangle$ (see Figure 2.9), $N_{0.0} = \{0, 1, 2, 3\}$. $E_{0.0}$ contains RMTs 0 and 3 as $\mathcal{R}_0[0] = \mathcal{R}_0[3] = 0$ whereas $E_{0.1}$ refers RMTs 1 and 2 as $\mathcal{R}_0[1] = \mathcal{R}_0[2] = 1$. Therefore, $l_{0.0} = \{0, 3\}$ and $l_{0.1} = \{1, 2\}$. The two children nodes $N_{1.0}$ and $N_{1.1}$ of node $N_{0.0}$ represent the sets of RMTs $\{0, 1, 6, 7\}$ and $\{2, 3, 4, 5\}$ respectively as $l_{0.0} = \{0, 3\}$ and $l_{0.1} = \{1, 2\}$ respectively. By repeating the process of generating the children nodes, the RT is constructed up to level 4 where the nodes of the leaf level (here it is level 4) are *empty*. As every $l_{i.j_1}$ $(0 \leq i \leq 3$ and $0 \leq j_1 \leq 2^{i+1} - 1)$ is *non-empty*, therefore, 2^4 children nodes are generated at level 4 which shows the RT is *complete* [45] also.

A linear time algorithm has been devised in [45] which decides the given CA is reversible or irreversible CA using reachability tree. Next, the contributory theorems and corollary are reported for designing the algorithm.

Table 2.6: Balanced reversible ECA rules

15,	23,	27,	30,	39,	43,	45,	51,	53,	54,	57,	58,	60,
75,	77,	78,	83,	85,	86,	89,	90,	92,	99,	101,	102,	105,
106,	108,	113,	114,	120,	135,	141,	142,	147,	149,	150,	153,	154,
156,	163,	165,	166,	169,	170,	172,	177,	178,	180,	195,	197,	198,
201,	202,	204,	210,	212,	216,	225,	228,	232,	240			

Theorem 2.2 *The reachability tree of an ECA with null boundary is complete if each edge, except the leaf edges, is resulted from exactly two RMTs of the corresponding rule.* [45]

Corollary 2.1 *All the nodes except leaves of the reachability tree for a reversible CA is constructed with 4 RMTs.* [45]

Theorem 2.3 *At each level, except the root, of the reachability tree for a reversible CA, there are 2 or 4 unique nodes.* [45]

It is observed that all rules do not contribute in the generation of reversible CA. This has motivated the researchers to figure out the effective rules for reversible and irreversible CAs. The presence of a *irreversible* rule in a rule vector makes the CA irreversible [46].

Theorem 2.4 *An unbalanced rule is an irreversible rule.* [45]

Therefore, for an n-cell reversible CA, all n rules must have to be *balanced* [45]. A balanced rule has equal number of 0s and 1s. Though there are 70 balanced ECA rules but only 62 (see Table 2.6) of them are effective of reversible by CAs which is established by the following theorem.

Theorem 2.5 *A balanced rule with same value for the RMTs* 0, 2, 3, 4, *or RMTs* 0, 4, 6, 7, *or RMTs* 0, 1, 2, 6, *or RMTs* 0, 1, 3, 7 *is an irreversible rule.* [45]

As null boundary CA is considered, therefore, the first and last rules are to be selected separately. The balanced property is checked only for *four* corresponding effective RMTs both for first rule and last rule. For $\mathcal{R}_0$, the balanced group rules are $3, 5$, 6, 9, 10 and 12; similarly, $\mathcal{R}_{n-1}$ is to be selected from the set $\{5, 17, 20, 65, 68, 80\}$.

Even we design a rule vector by selecting $\mathcal{R}_0$ from $\{3, 5, 6, 9, 10, 12\}$, $\mathcal{R}_1$ to $\mathcal{R}_{n-2}$ from Table 2.6 and $\mathcal{R}_{n-1}$ from $\{5, 17, 20, 65, 68, 80\}$, then also it is not guaranteed that the generated CA is reversible. There must a *relation* for selecting $\mathcal{R}_{i+1}$ based on $\mathcal{R}_i$ $(0 \le i \le n-2)$.

In reversible CA, as there is no non-reachable edge, therefore, at every level i, there exists 2^i nodes. According to Corollary 2.1 and Theorem 2.3, the possible nodes for a level are any of *three* possible conditions when the count of unique nodes of that level is *two*: $\{0,1,2,3\}$ & $\{4,5,6,7\}$, $\{0,1,4,5\}$ & $\{2,3,6,7\}$ and $\{0,1,6,7\}$ & $\{4,5,2,3\}$. When a level of reachability tree considers *four* unique nodes, then the nodes are any one of the following *three* combinations.

$\{0,1,2,3\}$ & $\{4,5,6,7\}$ and $\{0,1,4,5\}$ & $\{2,3,6,7\}$,
$\{0,1,2,3\}$ & $\{4,5,6,7\}$ and $\{0,1,6,7\}$ & $\{4,5,2,3\}$,
$\{0,1,4,5\}$ & $\{2,3,6,7\}$ and $\{0,1,6,7\}$ & $\{4,5,2,3\}$.

Thus for each such six combinations of nodes at level i, we get a subset of rules from the rule-space of 62 group rules. Therefore, those 62 are classified into *six* classes which are uniquely identified as Class I to Class VI. So, for every $\mathcal{R}_i$, we get two parts - (i.) the class name of $\mathcal{R}_i$ and (ii.) the next class name from where $\mathcal{R}_{i+1}$ is to be selected.

Table 2.7: Rules to generate a reversible CA

(a) Class relationship of $\mathcal{R}_i$ and $\mathcal{R}_{i+1}$

Class of $\mathcal{R}_i$	$\mathcal{R}_i$	Class of $\mathcal{R}_{i+1}$
I	51, 204, 60, 195	I
	85, 90, 165, 170	II
	102, 105, 150, 153	III
	53, 58, 83, 92, 163, 172, 197, 202	IV
	54, 57, 99, 108, 147, 156, 198, 201	V
	86, 89, 101, 106, 149, 154, 166, 169	VI
II	15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240	I
III	51, 204, 15, 240	I
	85, 105, 150, 170	II
	90, 102, 153, 165	III
	23, 43, 77, 113, 142, 178, 212, 232	IV
	27, 39, 78, 114, 141, 177, 216, 228	V
	86, 89, 101, 106, 149, 154, 166, 169	VI
IV	60, 195	I
	90, 165	IV
	105, 150	V
V	51, 204	I
	85, 170	II
	102, 153	III
	86, 89, 90, 101, 105, 106, 149, 150, 154, 165, 166, 169	VI
VI	15, 240	I
	105, 150	IV
	90, 165	V

(b) First Rule Table

Rules for $\mathcal{R}_0$	Class of $\mathcal{R}_1$
3, 12	I
5, 10	II
6, 9	III

(c) Last Rule Table

Rule class for $\mathcal{R}_{n-1}$	Rule set for $\mathcal{R}_{n-1}$
I	17, 20, 65, 68
II	5, 20, 65, 80
III	5,17, 68, 80
IV	20, 65
V	17, 68
VI	5, 80

The classification of these rules into different classes are shown in Table 2.7a [45]. Here, separate tables are used for the rules at cell 0 (Table 2.7b) and cell $n-1$ (Table 2.7c), as, each of them is selected from a set of only six rules different from the 62 rules of Table 2.7a. The process of synthesizing a non-uniform reversible CA is as follows:

1. Select a rule for the first cell i.e., $\mathcal{R}_0$ from the rules of first column of Table 2.7b. This rule dictates the class (second column of Table 2.7b) of the next rule $\mathcal{R}_1$ which has to be chosen from Table 2.7a.

2. For each i, $1 \leq i \leq n-2$, choose a rule $\mathcal{R}_i$ from the second column of Table 2.7a, where the first column of that row has the class indicated by rule $\mathcal{R}_{i-1}$. The third column of the row defines the class for the next rule $\mathcal{R}_{i+1}$.

3. The last rule $\mathcal{R}_{n-1}$ is picked up from Table 2.7c based on the class directed by $\mathcal{R}_{n-2}$.

Let us design a 4-cell reversible CA. To select an arbitrary $\mathcal{R}_0$, the first column of Table 2.7b is taken into consideration. Let rule 6 be selected as $\mathcal{R}_0$ from Table 2.7b (the third row and first column of Table 2.7b). As the class information of the next rule of rule 9 is class III (the second column and third row of Table 2.7b), therefore, $\mathcal{R}_1$ can be anyone from the pool of CA rules of class III from Table 2.7a. Let rule 86 be chosen as $\mathcal{R}_1$ (the first column of Table 2.7a for rule 86 is class III). Now, the third column and the row corresponding to rule 86 in Table 2.7a is class VI, therefore, $\mathcal{R}_2$ is to be selected from class VI from Table 2.7a. By repeating the same process, let us choose $\mathcal{R}_2$ as 15. Therefore, the class information for rule $\mathcal{R}_3$ is class I. However, as this is the last rule, we need to select this rule from Table 2.7c. Let $\mathcal{R}_3$ is 68 (second column and first row of Table 2.7c). Therefore, the reversible CA is $\langle 6, 86, 15, 68 \rangle$ (see Figure 2.7 and Figure 2.9).

2.5 Summary

This chapter has described a short tour on cellular automata. In this tour, we have seen the different milestones of the developments on cellular automata such as various types of CA based on its attributes like neighborhood structure, dimension, state count etc., CA's reversibility property, CA as technology and a special mention of CA in machine learning. From this tour, it has been observed that though reversibility of CA has been studied extensively but the cyclic properties of CA have not been explored much. Besides, in very few literature, we have found the contributions of CA in clustering problem, though CA (specially, single length cycle CA) is well known for pattern classification problem. This dissertation aims to study - (i.) the cyclic properties of CAs in broader aspects and (ii.) the effectiveness of CAs in clustering problem. In next chapters, the following problems statements are studied.

- How do we compute the cycle structure of any arbitrary CA efficiently?
- How do we characterize the cyclic spaces of CAs?
- For a given length cycle, how do we determine the best possible CA?
- Does reversible CA contribute effectively in clustering problem?

CHAPTER 3

Cellular Automata - Reachability, Cycles and Cycle Structure

This chapter presents the mechanism of identifying *cycles* in 1-dimensional finite cellular automaton (CA). We begin by viewing this problem as a special case of the *reachability problem*. *Reachability tree* is employed as a characterization tool on the basis of which an algorithm is reported to compute the cycle structure of a CA. However, the exponential complexity (in worst case) of this algorithm drives us to investigate some notable properties of reachability tree, keeping an eye on finding cycle structure of the reversible cellular automata (CAs). This chapter also devises an algorithm for evaluating the cycle structure of reversible CAs. We demonstrate that this algorithm performs efficiently for a collection of reversible cellular automata.

3.1 Introduction

During the evolution of CA, a configuration is *reachable* from a set of configurations, called predecessors, which makes the configuration space as a relation to represent the reachability among the configurations. This brings the notion of Configuration REachability Problem (CREP): Given a CA, CREP decides whether the destination configuration is reachable from the source configuration. The CREP is undecidable for infinite CAs (of dimension 1) [55], so researchers considered this problem for finite CAs [55, 56]. CREP is P-complete, NP-complete and PSPACE-complete depending on the types of CAs [55]. It has also been shown that CREP is NP-intermediate for the CAs with additive rules [56]. Wolfram's rule 102, for example, is an additive rule [358], and so to decide reachability of a destination configuration in a ECA 102 of finite size from a given source configuration, super polynomial time is needed. CREP has been studied for uniform cellular automata [55, 56] and remained unexplored in the domain of non-uniform CAs.

In the configuration space of a CA, the configurations are ultimately attracted to some *cycles*. A naive approach of determining the cycles of a CA is as follows: for every pair of configurations $(\mathtt{x}_s, \mathtt{x}_d)$, if $\mathtt{x}_s$ and $\mathtt{x}_d$ are reachable from each other, then they belong to same cycle by which ultimately all the cycles can be detected. Thus, identifying cycles in CA can be thought of as a sub-problem of CREP. Let us consider that $\mathtt{x}_s$ and $\mathtt{x}_d$ are respectively the source and destination configurations. Now, the CREP can be formulated in the following directions.

1. Does there exist a *path* from $\mathtt{x}_s$ to $\mathtt{x}_d$?
2. Does there exist a *path* from $\mathtt{x}_s$ (resp. $\mathtt{x}_d$) to itself?

This chapter studies CREP in connection to detecting a cycle of an n-cell cellular automaton. The computational complexity of CREP (and cycle determination) instigates us to dig into the problem of finding the *cycle structure* of a CA efficiently. In this chapter, the expedition of cycle structure starts from irreversible CA and its computational issue drives us to cultivate the non-trivial, challenging problem - finding cycle structure of reversible CA.

Hereafter, in this dissertation, by "CA" we refer to non-uniform 1-dimensional 3-neighborhood binary cellular automaton with finite lattice size and null boundary condition. In this chapter, as the target rule space consists of linear and non-linear rules both, *reachability tree* is used as the characterization tool for studying the above mentioned problem statements and all the necessary properties which are significant for designing the solution spaces efficiently, are developed using the reachability tree.

Reachability tree grows exponentially. And, one has to explore the entire tree for finding the cycle structure of a given CA. Targeting to reduce the search time for detecting cycles, the following contributions are reported in this chapter.

1. An effective mechanism for identifying acyclic configurations in a CA (an acyclic configuration does not belong to any cycle of the CA).
2. The estimation on the cycle structure of an n-cell reversible CA at level i of the reachability tree where $i < n$.
3. A methodology for finding the cycle structure of a reversible CA by detecting the similarity among the subtrees (will be defined in Section 3.4.1 of this chapter on page 60) in the reachability tree of that CA.

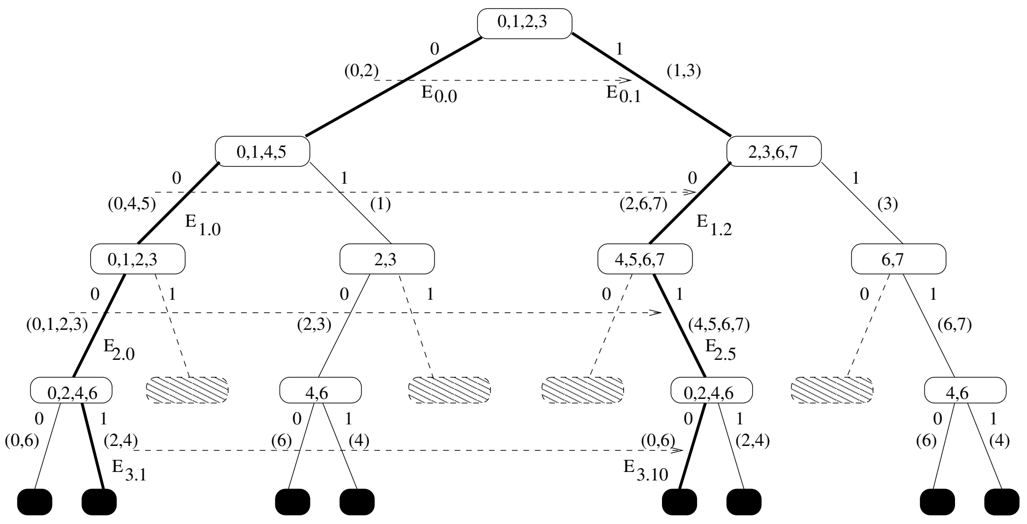

Figure 3.1: Predecessor of 0001 in CA $\langle 10, 10, 240, 20 \rangle$ by linking the edges

3.2 Reachability Analysis and Cycles in Cellular Automata

This chapter uses reachability tree (see Section 2.4.1 of Chapter 2 on page 33) as a tool for detecting the cycles of a CA. As identification of cycles is a special case of CREP, as a first step, our target is to figure out the intrinsic properties of reachability tree which analyze the reachability among the configurations using it. For finding a path from $\mathtt{x}_s$ to $\mathtt{x}_d$, we need to determine whether $\mathtt{x}_s$ is the l^{th} predecessor of $\mathtt{x}_d$ (where $l \in \mathbb{N}$). However, the tree guides us to find the predecessors of the CA configurations by establishing relation among edges. To find the relations among the edges, we use the concept of "link" in the next section. Let us first describe it intuitively.

Let us consider $\mathtt{x} = (x_0 \cdots x_i \cdots x_{n-1})$ be a configuration of an n-cell CA. In the reachability tree of that CA, $\mathtt{x}$ can be represented as a sequence of edges (from level 0 to level $n-1$). Let $(E_{0.j_0} \cdots E_{i.j_i} \cdots E_{n-1.j_{n-1}})$ be the sequence of edges which represents $(x_0 \cdots x_i \cdots x_{n-1})$ such that edge $E_{p.j_p}$ $(0 \le p \le n-1)$ of the tree refers the corresponding x_p (the state value of cell p). Let us consider this $(E_{0.j_0} \cdots E_{i.j_i} \cdots E_{n-1.j_{n-1}})$ to be associated with an RMT sequence $\rho(\mathtt{x}) = (\mathtt{r}_0 \cdots \mathtt{r}_i \cdots \mathtt{r}_{n-1})$. The configuration corresponding to $\rho(\mathtt{x})$ can also be represented by a sequence of edges in the tree, let it be $(E_{0.j'_0} \cdots E_{i.j'_i} \cdots E_{n-1.j'_{n-1}})$ (the corresponding configuration is $\mathtt{x}'$). Thus if we *link* $E_{i.j_i}$ to $E_{i.j'_i}$ for every i, it would be possible to reach directly to a predecessor of a configuration in the tree. Hence, to observe predecessors of all configurations, we can form such links using all RMT sequences. Following example illustrates this idea.

Example 3.1 Figure 3.1 is the reachability tree of the CA with rule vector $\langle 10, 10, 240, 20\rangle$. It shows that the sequence of edges representing configuration 0001 (*i.e.*, $(E_{0.0}E_{1.0}E_{2.0}E_{3.1})$) and the edge sequence representing the configuration corresponding to its associated RS 2524 (*i.e.*, $(E_{0.1}E_{1.2}E_{2.5}E_{3.10})$) are different. Let us now form *links* between the edges as shown by the dotted lines in Figure 3.1. However, the sequence $(E_{0.1}E_{1.2}E_{2.5}E_{3.10})$ represents the configuration 1010. Hence, 1010 is one of the predecessors of 0001. See Figure 2.6 for verification.

From Example 3.1, we get that a sequence of edges in a reachability tree can be *linked to* another sequence of edges if the former one refers to a reachable configuration. If a sequence of edges represents a *non-reachable* configuration, then such sequence is not linked to any sequence of edges in that tree. Though it can be *linked from* another sequence of edges. In fact, every sequence of edges in the tree is linked from a distinct sequence of edges in that tree. That means, at a particular level of the tree, every edge is linked from some edge(s) of that level.

3.2.1 Linking Configurations in Reachability Tree and Reachability Analysis

Now, we formalize the rules of forming links in the tree. When we traverse the reachability tree of a CA, at every level i $(0 \le i \le n-1)$, each RMT $\mathtt{r} \in l_{i.j}$ on edge $E_{i.j}$ $(0 \le j \le 2^i - 1)$ forms a *link* with an *edge* $E_{i.k}$. The physical meaning of a *link* is the connectivity between *two* edges using some RMT. Let $E_{i.j}(\mathtt{r}) \rightarrow E_{i.k}$ be a link at level i where $E_{i.j}$ and $E_{i.k}$ are called as respectively *source* and *destination* edges. When $j = k$, then such link is called as *self* link where $\mathtt{r}$ is *self replicating*; otherwise, the link is called as either *forward* link $(j < k)$ or *backward* link $(j > k)$. Next, we report the mechanism of linking the edges of a level in the tree.

Let us consider an edge $E_{i.k}$ at level i of the reachability tree of an n-cell CA, with a link from $E_{i.j}$ to $E_{i.k}$ due to RMT $\mathtt{r}$. When k is from $\{0, 4, 8, 12, \cdots,\}$ then corresponding $\mathtt{r}$ is from $\{0, 1\}$. Similarly, we can get $\mathtt{r} \in \{2, 3\}$ if $k \in \{1, 5, 9, 13, \cdots,\}$, $\mathtt{r} \in \{4, 5\}$ if $k \in \{2, 6, 10, 14, \cdots,\}$ and $\mathtt{r} \in \{6, 7\}$ if $k \in \{3, 7, 11, 15, \cdots,\}$. Therefore, $\mathtt{r}$ can be either $2k \pmod 8$ and $2k + 1 \pmod 8$. Only at level $n - 1$, $\mathtt{r} = 2k \pmod 8$. Therefore, j can be computed by a recursive function $\mathtt{src}(i, k, \mathtt{r})$ as follows.

$$\mathtt{src}(i, k, \mathtt{r}) = \mathtt{src}(i - 1, \lfloor \frac{k}{2} \rfloor, \lfloor \frac{\mathtt{r}}{2} \rfloor) + \mathcal{R}_i[\mathtt{r}]$$

The base case of the recursion is given by:

$$\texttt{src}(0, k, \texttt{r}) = \mathcal{R}_0[\texttt{r}] \text{ where } k \in \{0, 1\}$$

A link $E_{i.j}(\texttt{r}) \to E_{i.k}$ is *derived* from a link $E_{i-1.\lfloor j/2 \rfloor}(\texttt{r}') \to E_{i-1.\lfloor k/2 \rfloor}$ if $\texttt{r}$ is either $2\texttt{r}' \pmod 8$ or $2\texttt{r}'+1 \pmod 8$. As the links help to trace the predecessor of a configuration, the repeated use of linking technique may lead to reach the source configuration from the destination configuration. For studying CREP, let us take a tour on the well-established concepts, lemmas, properties etc., of reachability tree.

Lemma 3.1 *There exist only two links to $E_{i.k}$ from any one or two edges for RMTs $\texttt{r}$ and $\texttt{s}$ when $0 \leq i < n-1$ and $\texttt{r}$ and $\texttt{s}$ are sibling to each other, and only one link when $i = n-1$ in a reachability tree $(0 \leq j \leq 2^{i+1} - 1)$. [322]*

Property 3.1 *A link present at i^{th} level triggers two links at level $i+1$, where $0 \leq i \leq n-3$; a link of $(n-2)^{th}$ level derives one link at $(n-1)^{th}$ level [64].*

This is obvious, because an RMT $\texttt{r}$ at a node/label of level i contributes two RMTs - $2\texttt{r} \pmod 8$ and $2\texttt{r}+1 \pmod 8$ in node/label(s) of level $i+1$. Both the RMTs participate in links, depending upon the link caused by RMT $\texttt{r}$. However, a link at level $n-2$ triggers only one link at last level, as RMT $2r+1 \pmod 8$ is not considered in that level.

Property 3.2 *In the reachability tree, every level i except level $n-1$ has 2^{i+2} links $(0 \leq i \leq n-2)$, whereas level $n-1$ has 2^{i+1} links [64].*

In the reachability tree, every level starting from root to $(n-1)^{th}$ level represents 2^{i+1} edges where $0 \leq i \leq n-1$ and every edge is linked from the same edge or another edge(s) *twice* when $i < n-1$; therefore, 2^{i+2} links are present at every level. Only exception occurs at level $i = n-1$ where the number of total links is reduced to 2^{i+1} as each edge is linked from only *one* edge.

Through the links, the edges of a particular level of reachability tree can be connected. If $E_{i.j_1}$ is linked to $E_{i.j_2}$ for RMT $\texttt{r}_1 \in l_{i.j_1}$ and $E_{i.j_2}$ is linked to $E_{i.j_3}$ for RMT $\texttt{r}_2 \in l_{i.j_2}$ $(0 \leq j_1, j_2, j_3 \leq 2^{i+1} - 1)$, then there is a link from $E_{i.j_1}$ to $E_{i.j_3}$ $(0 \leq i \leq n-1)$ using the *transitivity* property [64] of links which is mentioned below.

- If $E_{i.j_1}(\texttt{r}_1) \to E_{i.j_2}$ and $E_{i.j_2}(\texttt{r}_2) \to E_{i.j_3}$, then
- $E_{i.j_1}(\texttt{r}_1) \to E_{i.j_2}(\texttt{r}_2) \to E_{i.j_3}$.

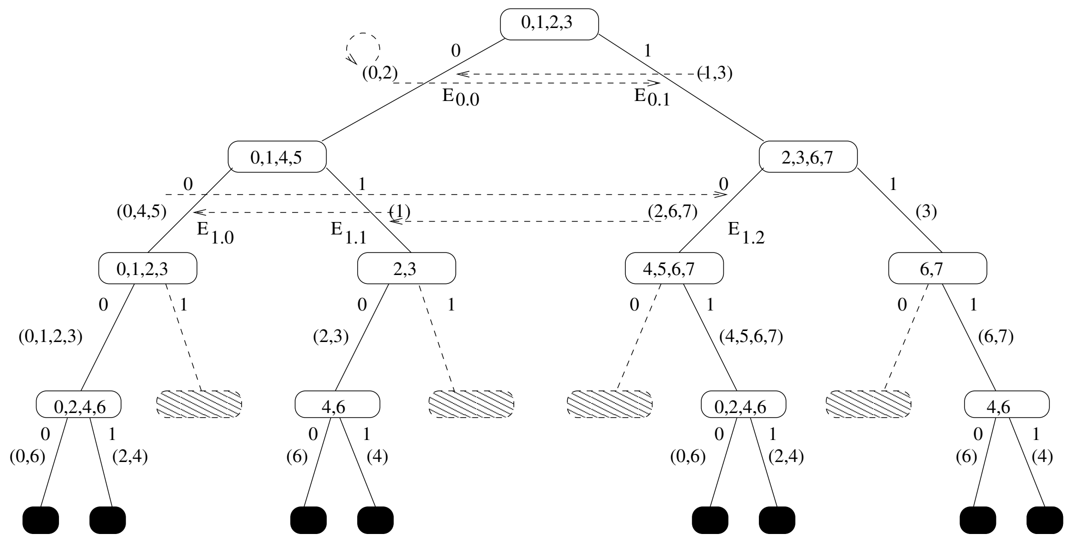

Figure 3.2: Links in CA $\langle 10, 10, 240, 20 \rangle$ up to level 1

Thus using the transitive property, a sequence of links $E_{i.j_1}(\mathtt{r}_1) \rightarrow E_{i.j_2}(\mathtt{r}_2) \rightarrow E_{i.j_3}$ can be obtained. The length of a sequence of links is determined by the number of RMTs *involved* in that sequence. If there is a link from edge $E_{i.j}$ to edge $E_{i.k}$, there are m number of RMTs (that is, m number of edges), then we write: $length(E_{i.j}, E_{i.k}) = m$. We write, $length(E_{i.j}, E_{i.k}) = \infty$ if there is no link between $E_{i.j}$ and $E_{i.k}$. In Figure 3.2, there exists a connection from $E_{1.0}$ to $E_{1.2}$: $E_{1.0}(5) \rightarrow E_{1.2}(2) \rightarrow E_{1.1}$. So, $length(E_{1.0}, E_{1.1}) = 2$. Here, the sequence is composed of forward links and backward links. A self link always forms a loop of length one. Sometimes, a sequence of links may form loop. A loop of links of length greater than one is called as *cross link* [64].

Definition 3.1 *A sequence of links of length l (> 1) at level i that forms a loop is called as cross link. Let $E_{i.j_1}(\mathtt{r}_1) \rightarrow E_{i.j_2}(\mathtt{r}_2) \rightarrow \cdots \rightarrow E_{i.j_l}(\mathtt{r}_l) \rightarrow E_{i.j_1}$ be a cross link of level i; then $E_{i.j_w}$ is involved in that cross link for RMT $\mathtt{r}_w$ for every $w \in [1, l]$.*

In Figure 3.2, at level 1, consider a sequence of links - $E_{1.0}(5) \rightarrow E_{1.2}(2) \rightarrow E_{1.1}(1) \rightarrow E_{1.0}$; this sequence forms a cross link of length 3. The cross links are classified into two categories: elementary cross link and compound cross link [64].

Definition 3.2 *[64] A cross link is said to be elementary cross link if each of the edges is unique in the sequence of edges of the cross link. Otherwise, it is a compound cross link.*

In an elementary cross link, an edge is involved as source edge in the sequence of links only for *once*; whereas a compound cross link allows an edge as source edge in the sequence more than one time. Even, a self link may also be involved in compound cross link. In Figure 3.2, $E_{1.0}(5) \rightarrow E_{1.2}(2) \rightarrow E_{1.1}(1) \rightarrow E_{1.0}$ is an elementary cross link and $E_{0.0}(2) \rightarrow E_{0.1}(1) \rightarrow E_{0.0}(0) \rightarrow E_{0.0}$ is a compound cross link where edge $E_{0.0}$ is used as source edges twice. The cross links and self links of the reachability tree of a CA play the vital for detecting the cycles of CA.

Theorem 3.1 *For an n-cell CA, a cross link of length l at level $n-1$ implies a cycle of length l of the CA [322].*

Corollary 3.1 *An n-cell CA contains μ cycle(s) of length one, if and only if $(n-1)^{th}$ level contains μ self links. [64]*

Let us now define *path* between two edges of a level - $E_{i.j_1}$ and $E_{i.j_l}$. We say that there exists a path from $E_{i.j_1}$ to $E_{i.j_l}$ if there is a sequence of links from $E_{i.j_1}$ to $E_{i.j_l}$, that is, if $length(E_{i.j_1}, E_{i.j_l})$ is finite. Otherwise, there is no path between $E_{i.j_1}$ and $E_{i.j_l}$. If a path exists, we write it as the following: $E_{i.j_1}(\mathtt{r}_1) \rightarrow E_{i.j_2}(\mathtt{r}_2) \rightarrow \cdots \rightarrow E_{i.j_l}$.

To detect the reachability from $\mathtt{x}_s = (x_{s_0}x_{s_1}\cdots x_{s_{n-1}})$ (the source configuration) to $\mathtt{x}_d = (x_{d_0}x_{d_1}\cdots x_{d_{n-1}})$ (the destination configuration), we have to identify a path from $E_{i.d_i}$ to $E_{i.s_i}$ $(0 \leq i \leq n-1)$ in the reachability tree where $(E_{i.s_i})_{0\leq i\leq n-1}$ and $(E_{i.d_i})_{0\leq i\leq n-1}$ represent the sequences of edges for $\mathtt{x}_s$ and $\mathtt{x}_d$ respectively. Let us introduce the mechanism for determining the path from $\mathtt{x}_d$ to $\mathtt{x}_s$ intuitively using reachability tree.

Example 3.2 Suppose, $\mathtt{x}_s = 1111$ and $\mathtt{x}_d = 0001$ for the CA $\langle 10, 10, 240, 20\rangle$. Now, from Figure 3.3, we see that $\mathtt{x}_s$ and $\mathtt{x}_d$ are represented by the sequences of edges $(E_{0.1}E_{1.3}E_{2.7}E_{3.15})$ and $(E_{0.0}E_{1.0}E_{2.0}E_{3.1})$ respectively. To detect the reachability for (1111, 0001), first, we need to detect a path from $E_{0.0}$ to $E_{0.1}$; next target is to find a path from $E_{1.0}$ to $E_{1.3}$ and in this way a path needs to be identified from $E_{3.1}$ to $E_{3.15}$. From the links in the tree, we can get the path - $E_{3.1}(4) \rightarrow E_{3.10}(6) \rightarrow E_{3.15}$ ($length(E_{3.1}, E_{3.15}) = 2$). See Figure 2.6 for verification. Therefore, 0001 is reachable from 1111 in CA $\langle 10, 10, 240, 20\rangle$. For the same CA, if $\mathtt{x}_s = 0001$ and $\mathtt{x}_d = 0010$, then the corresponding sequences of edges are $(E_{0.0}E_{1.0}E_{2.0}E_{3.1})$ and $(E_{0.0}E_{1.0}E_{2.1}E_{3.2})$ respectively. It is shown in Figure 3.3 that there is no path from $E_{3.2}$ to $E_{3.1}$ (as $E_{2.1}$ is non-reachable edge, then there is no existence of links from $E_{2.1}$ and $E_{3.2}$). So, 0010 is not reachable from 0001.

Next section targets to study CREP when $\mathtt{x}_s = \mathtt{x}_d$.

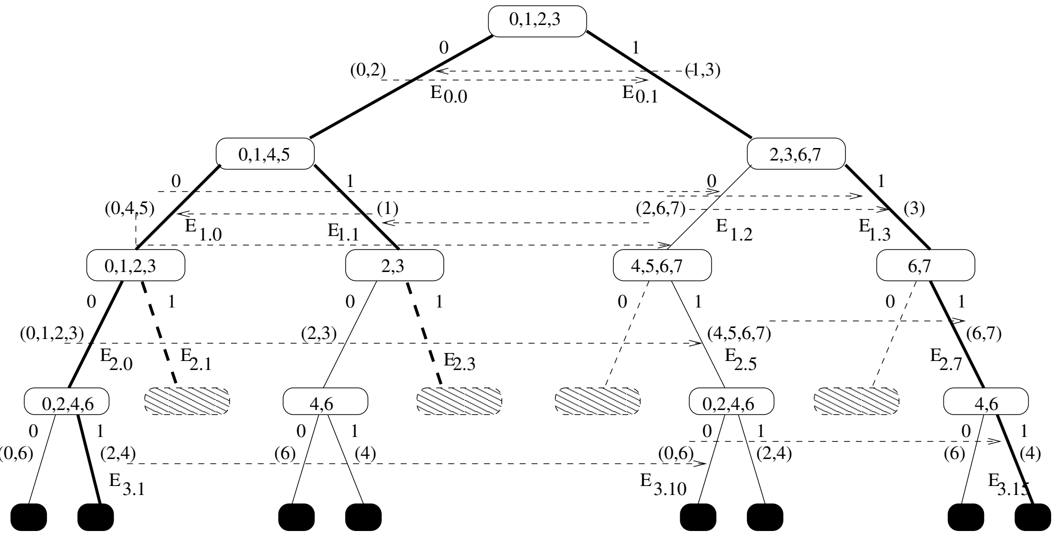

Figure 3.3: Reachability analysis among the configurations is explained in CA $\langle 10, 10, 240, 20\rangle$. The detection of acyclic configuration is also explained (based on Condition 1) using CA $\langle 10, 10, 240, 20\rangle$. Corollary 3.4 is also verified by this figure.

3.2.2 Study of CREP for Identification of Cycles

This section reports whether the given configuration is reachable from itself. Let us consider $\mathtt{x} = (x_i)_{0\leq i\leq n-1}$ be a configuration of an n-cell CA. Let us also consider the sequence of edges corresponds to x in the reachability tree as $(E_{0.j_{01}}E_{1.j_{11}}\cdots E_{n-1.j_{(n-1)1}})$. For determining the reachability of x from itself, a path has to be figured out from $E_{i.j_{i1}}$ to itself at every level i $(0 \leq i \leq n-1)$ of the tree. To reach $E_{n-1.j_{(n-1)1}}$ from itself, if the path length is greater than one, obviously this path forms a cross link.

Theorem 3.2 *In the reachability tree of an n-cell CA, let $(E_{0.j_{01}}E_{1.j_{11}}\cdots E_{n-1.j_{(n-1)1}})$ be the sequence of edges corresponds to the configuration* x *of that CA. The configuration* $\mathtt{x} = (x_0x_1\cdots x_{n-1})$ *is reachable from itself, if and only if there exists a cross link which involves $E_{n-1.j_{(n-1)1}}$.*

Proof: Suppose there exists a cross link of length l at level $n-1$: $E_{n-1.j_{(n-1)1}}(\mathtt{r}_{(n-1)1}) \rightarrow E_{n-1.j_{(n-1)2}}(\mathtt{r}_{(n-1)2}) \rightarrow \cdots \rightarrow E_{n-1.j_{(n-1)l}}(\mathtt{r}_{(n-1)l}) \rightarrow E_{n-1.j_{(n-1)1}}$ and $E_{n-1.j_{(n-1)1}}$ represents x_{n-1}. Now, we can prove that x is reachable from itself. Hence, we can get a sequence of edges from root to $E_{n-1.p}$ for each $p \in \{j_{(n-1)1}, j_{(n-1)2}, \cdots, j_{(n-1)l}\}$ which represents a reachable configuration. Here, two reachable configurations which are represented by edge sequences that end with $E_{n-1.j_{(n-1)q}}$ and $E_{n-1.j_{(n-1)(q+1)}}$ respectively are two consecutive configurations where each of them associated with at least one RS. Hence, we can get a sequence of consecutive configurations that forms a cycle involving the x.

Since there is a cross link, the sequence of configurations forms a path (cross link) involving the RMTs $\mathtt{r}_{(n-1)1}, \mathtt{r}_{(n-1)2}, \cdots, \mathtt{r}_{(n-1)l}$. Hence, $\mathtt{x}$ is reachable from itself.

Now suppose, $\mathtt{x}$ is reachable from itself. Obviously, there is a cross link involving $E_{i.j_{i1}}$, $0 \le i \le n-1$. Hence the proof follows. $\square$

Corollary 3.2 *For an n-cell CA,* $\mathtt{x}$ *is reachable from itself with path length one, if and only if every* $E_{i.j_{i1}}$ $(0 \le i \le n-1)$ *forms a self link.*

Other than identifying the cross link(s) or self link(s) at every level of the reachability tree for $\mathtt{x}$, the reachability of $\mathtt{x}$ from itself can also be determined by figuring out whether $\mathtt{x}$ is *acyclic* or not.

Theorem 3.3 *An RMT* $\mathtt{r} \in l_{i.j}$ *can not be a part of a cycle, if the RMT is not involved in a self link or cross link* $(0 \le i \le n-1,\ 0 \le j \le 2^{i+1}-1)$. *[322]*

If a link $E_{i.j}(\mathtt{r}) \to E_{i.k}$ is not in a loop at level i, therefore, RSs which have the member RMT $\mathtt{r}$ of $E_{i.j}$ always direct acyclic configurations. Here, we are motivated to figure out some conditions which determine that $\mathtt{x}$ is acyclic by traversing minimum number of edges at some level i. Obviously, these conditions decide that the given $\mathtt{x}$ is not reachable from itself.

Condition 1 *For an n-cell CA, let* $(E_{i.j_{i1}})_{0 \le i \le n-1}$ *be the sequence edges in the reachability tree for a configuration* $\mathtt{x} = (x_i)_{0 \le i \le n-1}$. *If* $E_{i.j_{i1}}$ *is non-reachable where* $0 \le i \le n-1$, *then* $\mathtt{x}$ *is acyclic.*

Reason: From Theorem 3.2 and Corollary 3.2 respectively, we know that, $\mathtt{x}$ is reachable from itself if there exists a cross link or self link involving $E_{n-1.j_{(n-1)1}}$. And from Property 3.1, we can say that the path at level $n-1$ is triggered from the root. That is, if there exists a cross link or self link which involves $E_{n-1.j_{(n-1)1}}$, then, there exist cross links or self links at levels 0, 1, $\cdots$ and $n-2$ which involve $E_{0.j_{01}}$, $E_{1.j_{11}}$, $\cdots$, and $E_{n-2.j_{(n-2)1}}$ respectively. If at any level, $E_{i.j_{i1}}$ is non-reachable, then there is no link from this edge. Hence, there is no cross link or self link which involves $E_{i+1.j_{(i+1)1}}$, $\cdots$, and $E_{n-1.j_{(n-1)1}}$. Therefore, $\mathtt{x}$ is acyclic.

Condition 2 *For an n-cell CA, let* $(E_{i.j_{i1}})_{0 \le i \le n-1}$ *be the sequence edges in the reachability tree for a configuration* $\mathtt{x} = (x_i)_{0 \le i \le n-1}$. *Let us also consider* $E_{i.j_{i1}}(\mathtt{r}_{j_{i1}}) \to E_{i.k_{i1}}$. *If* $|l_{i.j_{i1}}| = 1$ *and* $E_{i.k_{i1}}$ *is non-reachable where* $0 \le i \le n-1$, *then* $\mathtt{x}$ *is acyclic.*

Reason: As $|l_{i.j_{i1}}| = 1$, therefore, there is only one link from $E_{i.j_{i1}}$. Now, that link is $E_{i.j_{i1}}(\mathtt{r}_{j_{i1}}) \to E_{i.k_{i1}}$. As $E_{i.k_{i1}}$ is non-reachable, therefore, $E_{i.k_{i1}}$ can not be linked to any edge (according to Condition 1), then $E_{i.j_{i1}}$ can not be involved any sequence of links other than $E_{i.j_{i1}}(\mathtt{r}_{j_{i1}}) \rightarrow E_{i.k_{i1}}$. So, $E_{i.j_{i1}}$ can not be involved in any cross link or self link. Hence, $\mathtt{x}$ is acyclic.

Condition 3 *For an n-cell CA, let $(E_{i.j_{i1}})_{0\le i\le n-1}$ be the sequence of edges in the reachability tree for a configuration $\mathtt{x} = (x_i)_{0\le i\le n-1}$. Let us also consider that $E_{i.j_{i1}}(\mathtt{r}_{j_{i1}}) \rightarrow E_{i.k_{i1}}$. If $|l_{i.j_{i1}}| = |l_{i.k_{i1}}| = 1$ and $E_{i.k_{i1}}$ is self linked where $0 \le i \le n-2$, then $\mathtt{x}$ is acyclic.*

Reason: As $|l_{i.j_{i1}}| = 1$, therefore, there is only one link from $E_{i.j_{i1}}$. Now, that link is $E_{i.j_{i1}}(\mathtt{r}_{j_{i1}}) \rightarrow E_{i.k_{i1}}$. As $E_{i.k_{i1}}$ is self linked only for one RMT, therefore, $E_{i.k_{i1}}$ is not linked to any edge excluding itself. Therefore, there is no path $E_{i.k_{i1}}$ to $E_{i.j_{i1}}$. Hence, $E_{i.j_{i1}}$ can not be involved in any cross link or self link and $\mathtt{x}$ is acyclic.

Let us explain the above mentioned conditions using the following example.

Example 3.3 Let us consider the CA $\langle 10, 10, 240, 20\rangle$ with given configuration $\mathtt{x} = 0110$. In the reachability tree of CA $\langle 10, 10, 240, 20\rangle$, $\mathtt{x} = 0110$ can be represented as the sequence of edges $(E_{0.0}E_{1.1}E_{2.3}E_{3.6})$. In Figure 3.3, $E_{0.0}$ is involved in a self link: $E_{0.0}(0) \rightarrow E_{0.0}$. Besides, this edge is also involved in a cross link: $E_{0.0}(2) \rightarrow E_{0.1}(1) \rightarrow E_{0.0}$. At the first level, there exist more than one cross link which involve $E_{1.1}$ (for example, $E_{1.1}(1) \rightarrow E_{1.0}(4) \rightarrow E_{1.2}(2) \rightarrow E_{1.1}$ and $E_{1.1}(1) \rightarrow E_{1.0}(4) \rightarrow E_{1.2}(6) \rightarrow E_{1.3}(3) \rightarrow E_{1.1}$). Now, at the second level, $E_{2.3}$ is non-reachable edge. So, there is no path from $E_{2.3}$ to anywhere. Therefore, 0110 is *acyclic* by satisfying Condition 1 (see Figure 3.3).

Next, let us consider $x = 1010$ for CA $\langle 10, 146, 128, 16\rangle$. The corresponding sequence of edges for configuration 1010 in the reachability tree of CA $\langle 10, 146, 128, 16\rangle$ is $(E_{0.1}E_{1.2}E_{2.5}E_{3.10})$ as depicted in Figure 3.4. In the tree, $E_{0.1}$ is involved both in self link and cross link respectively $E_{0.1}(3) \rightarrow E_{0.1}$ and $E_{0.1}(1) \rightarrow E_{0.0}(2) \rightarrow E_{0.1}$ (in Figure 3.4). Now, at the next level, $E_{1.2}$ is involved in more than one cross link. At level 2, the target edge is $E_{2.5}$ and $|l_{2.5}| = 1$. Now, $E_{2.5}$ is linked to $E_{2.3}$ such that $E_{2.5}(7) \rightarrow E_{2.3}$. Here, $E_{2.3}$ is *non-reachable*; so, $E_{2.3}$ can not form a link with any edge (according to Condition 1). Therefore, $E_{2.5}$ can not be involved in any sequence of links other than $E_{2.5}(7) \rightarrow E_{2.3}$. So, $E_{2.5}$ is not involved in any cross link by satisfying Condition 2 (see Figure 3.4).

Next, let us consider $\mathtt{x} = 1101$ for the same CA (see Figure 3.4). The sequence of edges corresponds to $\mathtt{x}$ is $(E_{0.1}E_{1.3}E_{2.6}E_{3.13})$. Here also, $E_{0.1}$ forms both cross link and self link. Now, at the next level, the target edge $E_{1.3}$ is involved in self link only ($E_{1.3}(7) \rightarrow E_{1.3}$). At level 2, $E_{2.6}$ is involved in one link and that link is a forward link: $E_{2.6}(6) \rightarrow E_{2.7}$. As this $E_{2.7}$ is only linked to itself so there is no path from $E_{2.7}$ to $E_{2.6}$. Therefore, $E_{2.6}$ is not involved in cross link. So, 1101 is *acyclic* by satisfying Condition 3 (see Figure 3.4).

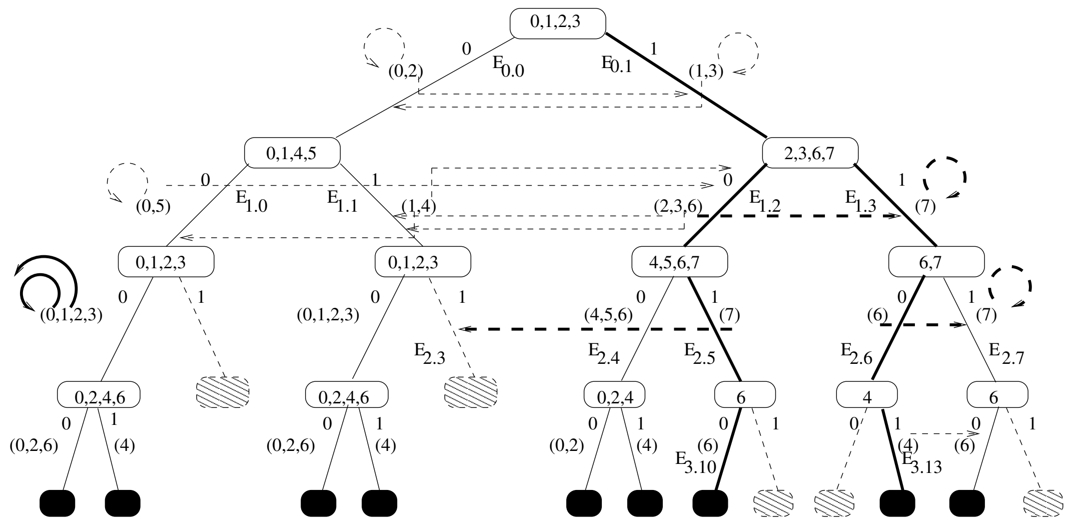

Figure 3.4: Reachability analysis of acyclic configurations is explained based on Conditions 2 and 3 in CA $\langle 10, 146, 128, 16\rangle$. Corollaries 3.3 and 3.5 are also verified by this figure.

Using all three above mentioned conditions which are explained by Example 3.3, the reachability may be decided much before than encountering the last rule. As an example, we can see that if $\mathtt{x} = 11(0+1)^{n-2}$, and if the first two rules of the CA are 10 and 16, then $\mathtt{x}$ is not reachable from itself for any value of $n \geq 2$ (by satisfying Condition 1). Therefore, based on a substring of the rule vector $\langle \mathcal{R}_0, \cdots, \mathcal{R}_{n_1-1}\rangle$ of an n-cell CA, we can decide if $\mathtt{x}$ is acyclic where $n_1 < n$. However, when the destination configuration is reachable from the source one, we will be able to detect it only at level $n-1$. Moreover, for determining existence of the path from $E_{n-1.j_{(n-1)1}}$ to itself, a path has to be found out from $E_{i.j_{i1}}$ to itself at every level i where $0 \leq i \leq n-2$. It is observed from Example 3.3 that an $E_{i.j_{i1}}$ may be involved in self link and cross link both when $i < n-1$. Even an edge can also be involved in multiple cross links of level i. The mechanism of identifying the cross link has already been reported in [64] and it is computationally demanding. This instigates us to develop some theories which help to keep only those edges in the reachability tree at level i, which are involved in either self link(s) or cross link(s). Thus we get ultimately only the cycles of CA at level $n-1$ of the tree. Moreover, the count of cycles of an n-cell CA can also be determined without traversing all 2^n edges of level $n-1$. This incites us to deal the problem of cycle structure (see Definition 2.17 of Chapter 2 on page 32). Now the question is, can we devise an efficient strategy which can find the cycle structure (we can call it as "CS" also) of an n-cell arbitrary CA? Let us start our journey with irreversible CAs.

3.3 Finding Cycle Structure (CS) of Cellular Automata

During evolution of a cellular automaton, it approaches to a set of configurations which form cycles. The cycles, which are also called as attractors, may have different lengths. In a CA, we can get μ_1 cycles of length l_1, μ_2 cycles of length l_2, and so on. We can write down these information of cycles as $[\mu_1(l_1), \mu_2(l_2), \cdots]$, which is popularly known as cycle structure of the CA. It is already noted that detecting cycles can be considered as a sub problem of CREP and CREP is PSPACE-complete depending on the types of CAs [56]. So, we are motivated to study the problem of cycle structure (CS) of an n-cell CA. Next we explain the cycle structure of an n-cell CA.

By cycle structure (CS) of an n-cell CA, we mean here the collection of number of cycles along with their lengths [44, 60]. Let us consider $\texttt{CS}_{\mathcal{R}}$ be the cycle structure of an n-cell CA $\mathcal{R}$ where $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \mu_2(l_2), \cdots, \mu_m(l_m)]$ and $l_1 > l_2 > \cdots > l_m$. Therefore, the CA has m distinct cycle lengths and total $\sum_{i=1}^{m} \mu_i$ number of cycles. Here, $\mu_i(l_i)$ $(1 \le i \le m)$ refers the μ_i cycle(s) of length l_i and this $\mu_i(l_i)$ is called as a *cyclic component* of that CA. In reversible CA, $\sum_{i=1}^{m} \mu_i \cdot l_i = 2^n$. For some special cases like single length cycle CAs and equal length cycle CAs, the cycle structures of CAs are in the given form $[\mu(l)]$; for single length cycle CA, $l = 1$.

The cycle structure of a CA is a well-studied problem [57–59] and has great contributions in different applications [44]. CAs which can generate *large* length cycles have great contributions in generating pseudo-random patterns, in testing of VLSI circuits, etc., (see [44,58,61–63]); the CAs which have equal length cycles are used in block cypher generation [53]; CAs with only point state attractors (cycles of length one) have great contributions in pattern recognition [64, 312, 317]. The candidate CAs of the above works are *non-uniform* and *linear* where the cells use Wolfram's rules. The reason for focusing on linear CAs for studying the cycle structure was, matrix algebra, polynomial theory, etc., can be used for these CAs, and established theories of these fields can be utilized to get the cycle structures. On the other hand, for an arbitrary non-linear CA, no method is known to get its CS.

In this scenario, this dissertation devises a mechanism which finds the cycle structures of finite 1-dimensional CAs and the candidate CAs are designed with linear/additive and non-linear rules. For getting cycle structure, only those links are to be present in the reachability tree which are involved either in self link(s) or in cross link(s). Therefore, the links which are associated with some acyclic configurations, are to be discarded. Already Section 3.2.2 contributed on deciding an acyclic configuration (using either Condition 1 or Condition 2 or Condition 3). Next, we report how to eliminate those links at level i of the reachability tree which are associated with acyclic configurations.

3.3.1 Identification of Acyclic Configurations

In an n-cell CA, as the number of non-reachable configurations is exponential [45], obviously, the count of acyclic configurations is also exponential. Therefore, at level $n-1$ of the reachability tree of the CA, there exist exponential number of links which are not *useful* for generating cycle(s) of the CA. So, our focus is to find such links at level i. Next, we report some corollaries which help us to find acyclic configurations .

Corollary 3.3 *If $E_{i.j}(\mathtt{r}) \rightarrow E_{i.k}$ where $j \neq k$ and $l_{i.k} = \varnothing$, then RMT $\mathtt{r}$ is involved in those RSs which represent acyclic configurations.*

Proof: Let us consider $E_{i.j}(\mathtt{r}) \rightarrow E_{i.k}$. As $l_{i.k} = \varnothing$, then, all the edges generated from $E_{i.k}$ up to level $n-1$ are non-reachable. So, any edge sequence where $E_{i.k}$ is involved represents non-reachable configuration. Now, every derived link of $E_{i.j}(\mathtt{r}) \rightarrow E_{i.k}$ has source edge as $E_{(i+v).j_v}$ (where $1 \leq v \leq n-1-i$ and $2^v.j \leq j_v < 2^v.(j+1)$) destination edge as $E_{(i+v).k_v}$ $(2^v.k \leq k_v < 2^v.(k+1))$ where $E_{(i+v).k_v}$ is non-reachable edge. Therefore, an edge sequence involves $E_{i.j}$, which represents reachable configuration with RMT sequence $(\cdots \mathtt{r}'\mathtt{r}\mathtt{r}'')$ (where $\mathtt{r}$ is either $2\mathtt{r}' \pmod 8$ or $2\mathtt{r}'+1 \pmod 8$ and $\mathtt{r}''$ is either $2\mathtt{r} \pmod 8$ or $2\mathtt{r}+1 \pmod 8$), has a predecessor which is an acyclic (non-reachable) configuration. The proof follows. □

Example 3.4 In Figure 3.4, at level 2, one of the links is $E_{2.5}(7) \rightarrow E_{2.3}$. Now, $l_{2.3} = \varnothing$. So, $E_{2.3}$ can not establish a link with any other edge. Therefore, RMT 7 of $E_{2.5}$ is involved in a RS 1376 which refers an acyclic (non-reachable) configuration (0111).

Corollary 3.4 *If an edge $E_{i.j}$ is self linked for two sibling RMTs $\mathtt{r}$, $\mathtt{s} \in l_{i.j}$, then the RSs that involve RMT $\mathtt{q} \in l_{i.j} \setminus \{\mathtt{r}, \mathtt{s}\}$ represent acyclic configurations $(0 \leq i \leq n-2,\ 0 \leq j \leq 2^{i+1}-1)$. [322]*

Example 3.5 In Figure 3.3, edge $E_{2.0}$ is self linked for RMTs 0 and 1 which are derived from RMT 0 of $E_{1.0}$. RMT 4 of $E_{1.0}$ also generates RMT 0 and 1 which belong to $l_{2.0}$. Other RMTs of $l_{2.0}$ are 2 and 3, but, no edge is linked to $E_{2.0}$ other than itself. Therefore, four RSs represent acyclic configurations - 2400 (1000), 2536 (1011) and 2412 (1001), 2524 (1010) (for verification see Figure 2.6).

Corollary 3.5 *If $E_{i.j}(\mathtt{r}) \rightarrow E_{i.k}$ and $E_{i.k}$ is self linked for one RMT* `s` *where* `r` *and* `s` *are sibling RMTs. If $|l_{i.k}| = 1$, then the RSs that involve RMT $\mathtt{r} \in l_{i.j}$ represent acyclic configurations* $(0 \leq i \leq n-2,\ 0 \leq j \leq 2^{i+1}-1)$.

Proof: As $E_{i.k}$ is self linked for RMT `s` therefore, $E_{i.k}$ can not be involved in any cross link, neither it can be linked to any other edge. Moreover, as $|l_{i.k}| = 1$, there is no RMT at $E_{i.k}$ which participates in any cross link. Now, `r` (where $\mathtt{r} \in l_{i.j}$ is the sibling RMT of `s`) is linked to $E_{i.k}$, so `r` also can not be part of any cross link. Therefore, RSs involve RMT `r` represent acyclic configurations. This completes the proof. □

Example 3.6 In Figure 3.4, $E_{1.2}(6) \rightarrow E_{1.3}$. Now, $|l_{1.3}| = 1$ and $E_{1.3}$ is self linked with RMT 7. Here, $E_{1.3}$ is not linking to any other edge other than itself. Therefore, RMT 6 of $E_{1.2}$ is not involved in any loop. Thus RSs 3764 and 3776 represent acyclic configurations 1110 and 1111 respectively.

From the theories mentioned above, it is clear that if a link $E_{i.j}(\mathtt{r}) \rightarrow E_{i.k}$ is not useful for generating cross links or self links at level i, then it has to be *removed* from the tree, and at next level onwards, no link is triggered from that discarded link. Thus due to the non-useful link at level i, 2^{n-i-1} links are not generated at level $n-1$. In this way, we can reduce the number of links as well as edges at level $n-1$ which are to be traversed to find the cycles of CA.

The following points are used sequentially to discard all acyclic configurations from the reachability tree.

1. The non-reachable edges do not participate in self or cross links. So they are obviously not processed in the tree.

2. Based on Corollary 3.3, there may exist an edge $E_{i.j}$ and an RMT `r`, such that $E_{i.j}$ can not be involved in any self link or cross link for RMT `r`. If this happens, `r` is removed from $l_{i.j}$. We check this condition for every edge and RMT.

3. Applying point 2 recursively, a number of edges (as well as RMTs) can be removed that can not participate in either self link or cross link.

4. Finally, the RMTs that are part of RSs which represent acyclic configurations (Corollary 3.4, Corollary 3.5) are removed.

Algorithm 1 Algorithm to find the CS of an n-cell CA

```
    Input: $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$
    Output: $\mathtt{CS}_{\mathcal{R}}$
 1: Set $l_{0.0} = \{r | \mathcal{R}_0[r] = 0\}$, $l_{0.1} = \{r | \mathcal{R}_0[r] = 1\}$, $\mathcal{J} = \{0, 1\}$.
 2: Initialize $lc \leftarrow 4$ and $ec \leftarrow |\mathcal{J}|$, where for any level, $ec$ and $lc$ denote the number of edges and the number of links among edges respectively.
 3: for ($i = 0$ to $n-1$)        (loop across levels) do
 4:    Create edges of level $i$ and links among edges of level $i$ (and remove all edges and links of previous levels if applicable).
 5:    for $j \in \mathcal{J}$ do
 6:       if $|l_{0.j}| \geq 3$ and $E_{i.j}$ is self linked with the sibling RMTs $r, s \in l_{0.j}$ then
 7:          Set $lc = lc - |l_{0.j}| + 2$; $l_{0.j} = \{r, s\}$.
 8:       end if
 9:       if $|l_{0.j}| = 1$ and $E_{i.j}$ is self linked with $r \in l_{0.j}$ then
10:          Set $l_{0.j'} = l_{0.j'} \setminus \{s\}$, where $s$ is sibling RMT of $r$, and $E_{i.j'}(s) \rightarrow E_{i.j}$.
11:          Set $lc = lc - 1$.
12:       end if
13:    end for    (edges of a particular level. Loop-1)
14:    Set $\mathcal{J}' = \{j | j \in \mathcal{J}$ and $l_{0.j} = \varnothing\}$ and $\mathcal{J} = \mathcal{J} \setminus \mathcal{J}'$, $\mathcal{J}'' = \varnothing$, $ec = |\mathcal{J}|$.
15:    while $\mathcal{J}' \neq \varnothing$ do
16:       for $j \in \mathcal{J}'$ do
17:          Set $l_{0.j'} = l_{0.j'} \setminus \{r\}$ and $l_{0.j''} = l_{0.j''} \setminus \{s\}$, where $r$ and $s$ are sibling RMTs, and $E_{i.j'}(r) \rightarrow E_{i.j}$ and $E_{i.j''}(s) \rightarrow E_{i.j}$.
18:          Set $lc = lc - 2$.
19:          if $l_{0.j'} = \varnothing$ then $\mathcal{J}'' = \mathcal{J}'' \cup \{ j' \}$
20:          end if
21:          if $l_{0.j''} = \varnothing$ then $\mathcal{J}'' = \mathcal{J}'' \cup \{ j'' \}$
22:          end if
23:       end for
24:       $\mathcal{J}' = \mathcal{J}''$; $ec = ec - |\mathcal{J}'|$; $\mathcal{J}'' = \varnothing$. $\mathcal{J} = \mathcal{J} \setminus \mathcal{J}'$
25:    end while    (edges of a particular level. Loop-2)
26:    $lc \leftarrow 2 \cdot lc$ if $i < n-1$. Set $ec \leftarrow 2 \cdot ec$.
27:    $\mathcal{J}' = \varnothing$, $k = 0$.
28:    for $j \in \mathcal{J}$ do
29:       If $i < n-1$, set $L_j = \{2r \pmod 8 | r \in l_{0.j}\} \cup \{(2r+1) \pmod 8 | r \in l_{0.j}\}$, else set $L_j = \{2r \pmod 8 | r \in l_{0.j}\}$.
30:       Update $l_{1.k} = \{r' | r' \in L_j$ and $\mathcal{R}_{i+1}[r'] = 0\}$, $l_{1.k+1} = \{r' | r' \in L_j$ and $\mathcal{R}_{i+1}[r'] = 1\}$. $k = k + 2$. $\mathcal{J}' = \mathcal{J}' \cup \{2j\} \cup \{2j+1\}$.
31:    end for
32:    Set $\mathcal{J} = \mathcal{J}'$, $l_{0.} = l_{1.}$.
33: end for    (level of tree)
34: Report the count of self links and cross links along with their lengths at level $n-1$.
```

3.3.2 An Effective Strategy for Computing the CS of a CA

For finding the cycle structure of a CA of size n, traditionally, we need to traverse all the configurations (2^n) of that CA. Obviously, the complexity of such technique is exponential. Now, in reachability tree, we can detect non-reachable configurations without exploring the entire tree if non-reachable edge can be found before reaching level $n-1$. So, a large number of acyclic (non-reachable) configurations can be discarded from the tree. Besides, reachable acyclic configurations can also be detected before reaching level $n-1$, see Corollary 3.3, Corollary 3.4 and Corollary 3.5 for reference. Based on the theories developed in Section 3.3.1, after removing all unnecessary edges (which leads to generate acyclic configurations) in reachability tree, at every level we get only those edges which are involve either in cross link or self link. The self and cross links at level $n-1$ of the tree contribute in evaluating the CS of the CA (Theorem 3.1, Corollary 3.1). For example, if there are one self link, one cross link of length 3 and two cross links of length 6 at level 3 of the tree, then the CS of the corresponding 4-cell CA is $[1(1), 1(3), 2(6)]$. Hence, our task is to find the lengths of cross links at level $n-1$, and to count the cross links of same length for reporting the cycle structure.

Combining all the above points, we now present the steps of our scheme to find the cycle structure of a CA in Algorithm 1. We use two variables lc and ec for storing respectively the number of edges and the number of links among edges of a particular level of the reachability tree. The data structure $l_{0.j}$ (where $0 \leq j \leq ec-1$) maintains label of an edge $E_{i.j}$ at the current level i of the reachability tree. Line numbers 6−8 and 9−12 follow from Corollary 3.4 and Corollary 3.5 respectively to identify links that are not useful and accordingly reduce lc. The loop from lines 15−25 follows from Corollary 3.3 to remove links which may further generate non-reachable edges. Lines 26−32 update the data structures for the next level (note that the same data structures are reused across all levels). Let us take the following example to illustrate the above scheme.

Example 3.7 See Figure 3.3 for the CS of CA $\mathcal{R} = \langle 10, 10, 240, 20\rangle$. Here, $l_{0.0} = \{0, 2\}$ and $l_{0.0} = \{1, 3\}$ as $\mathcal{R}_0[0] = \mathcal{R}_0[2] = 0$ and $\mathcal{R}_0[1] = \mathcal{R}_0[3] = 1$ where $\mathcal{R}_0 = 10$. At level 0 of the tree, $lc = 4$ and $ec = 2$. As $|l_{0.0}| = |l_{0.1}| = 2$, therefore, lines 6−8, 9−12 and 15−25 are not executed. Therefore, no link is removed from the tree at level 0. Therefore, the updated values of lc and ec are respectively 8 and 4 (Using line 26). Next, we get $l_{1.0} = \{0, 4, 5\}$, $l_{1.1} = \{1\}$, $l_{1.2} = \{2, 6, 7\}$ and $l_{1.3} = \{3\}$ and again no link is removed from level 1 and the updated values of lc and ec are respectively 16 and 8. At level 2, $l_{2.0} = \{0, 1, 2, 3\}$, $l_{2.2} = \{2, 3\}$, $l_{2.5} = \{4, 5, 6, 7\}$ and $l_{2.7} = \{6, 7\}$. Here, $l_{2.0}$ and $l_{2.5}$ are actually represent multi-sets $\{0, 1, 0, 1, 2, 3\}$ and $\{4, 5, 4, 5, 6, 7\}$ respectively. As $E_{2.0}$ is self linked for RMTs 0 and 1 which are derived from RMT 0 of $l_{1.0}$, therefore,

Table 3.1: Experimental results for evaluating cycle structures using Algorithm 1.

CA size n	The rule vector $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$	Cycle structure $\texttt{CS}_{\mathcal{R}}$
10	$\langle 10, 150, (90, 150)^3, 90, 20 \rangle$	$[1(1), 1(1023)]$
12	$\langle 14, 30^3, 45^2, 30^4, 45, 5 \rangle$	$[1(1), 2(3), 1(4), 1(8)]$
18	$\langle 10, 150, 90, 150, 240^2, (90, 150)^2, 240^2, (90, 150)^2, 240, 80 \rangle$	$[1(1), 1(3), 2(6), 20(12), 160(24)]$
25	$\langle (1, 32, 76, 44, 89)^4, 1, 32, 76, 44, 81 \rangle$	$[297(2), 360(4), 1341(6), 216(12)]$
28	$\langle 6, 195, 8, 10, (102, 195, 8, 10)^5, 102, 195, 8, 0 \rangle$	$[1(1)]$
30	$\langle (8^2, 4^2, 90^2)^4, 8^2, 4^2, 90, 80 \rangle$	$[1024(1), 15878(2)]$
35	$\langle (4, 186, 15, 240, 80)^7 \rangle$	$[1280(4)]$
42	$\langle (8, 45, 30, 120, 64, 48)^6, 8, 45, 30, 120, 64, 16 \rangle$	$[1(1), 5461(3)]$
55	$\langle (1, 21, 35, 95, 48)^{10}, 1, 21, 35, 95, 16 \rangle$	$[1024(2)]$
100	$\langle 8^{50}, 48^{49}, 16 \rangle$	$[1(1)]$
500	$\langle 10, 90, 240^2, (90^2, 240^2)^{123}, 90^2, 240, 80 \rangle$	$[1(1)]$

$lc = 16 - 6 + 2 = 12$ ($|l_{2.0}| = 6$) and $l_{2.0} = \{0, 1\}$ (using lines 6−8). Next, we get $\mathcal{J}' = \{1, 3, 4, 6\}$ at the same level; hence, $ec = 8 - 4 = 4$ (using line 14). Now, using lines 15−25, $\mathcal{J}'' = \{2, 5, 7\}$. Therefore, $lc = 12 - 10 = 2$ and $ec = 4 - |\mathcal{J}''| = 1$. Thus, $l_{3.0} = \{0\}$ and $l_{3.1} = \{2\}$. So, the number of cycles is two (Corollary 3.1) and those cycles have the same length i.e. 1. So, the corresponding cycle structure is represented as $\texttt{CS}_{\mathcal{R}} = [2(1)]$.

To understand the performance of our scheme, we have extensively experimented with randomly synthesized CAs rule vectors. A sample result of the experimentation is given in Table 3.1. The first column of the table represents CA size. The second and third columns of the table refer rule vector of a CA and the cycle structure of that CA. We have observed that in many cases, the proposed scheme efficiently gives the output.

Analysis of the scheme: We next figure out the time taken by each step of Algorithm 1. Clearly line 1 and line 2 of the pseudocode take constant amount of time. The 'for' loop (lines 3-33) is executed for each level of the tree. We will fix a particular level i, and analyse the running time of our algorithm for that level i. Let us begin by defining some variables. For level i, let lc_i^I and ec_i^I respectively denote the initial number of links and the number of edges at level i (line 4). In course of the algorithm, we remove redundant links and edges and let lc_i^F and ec_i^F respectively be the final number of links (line 18) and the number of edges (line 24) at level i. Following are some obvious relations: (i) $ec_i^I \leq lc_i^I$ (ii) $ec_i^F \leq ec_i^I$ (iii) $lc_i^F \leq lc_i^I$ (iv) $lc_{i+1}^I = 2 \cdot lc_i^F$ (v) $ec_{i+1}^I \leq 2 \cdot ec_i^F$.

Line 4 of the algorithm creates edges of level i and links between edges of level i and thus takes $O(ec_i^I + lc_i^I)$ time. The 'for' loop (from line 5 to 13) takes constant time for each edge belonging to $\mathcal{J}$, thus total time is $O(ec_i^I)$. Next we analyse the running time of the 'while' loop (line 15 to 25 of the algorithm). The first iteration of the while loop checks for each element j of $\mathcal{J}'$ if $l_{0.j}$ is empty and accordingly sets $l_{0.j'}$ and $l_{0.j''}$ to empty for appropriate indices j' and j''; thus it takes $O(|\mathcal{J}'|)$ time. It can be shown that all the remaining iterations of the while loop combined also take $O(|\mathcal{J}'|)$ time. Note that across all the iterations of the while loop, an index j can be inserted to the set $\mathcal{J}''$ at most once. This statement can be proved as follows. Let us assume that the index j is inserted to the set $\mathcal{J}''$ at the k^{th} iteration of the while loop. This implies that during iteration k of the while loop, removal of some RMTs from $l_{0.j}$ makes $l_{0.j}$ empty for the first time. Since, the procedure in the while loop does not add back RMTs to any set, the set $l_{0.j}$ will remain empty from that point onwards. Thus j will be inserted to the set $\mathcal{J}''$ only at the k^{th} iteration of the while loop. Thus across all the remaining iterations of the while loop, each edge will be considered at most once. Hence the running time of all the remaining iterations is at most $O(ec_i^I)$. Lines 28 to 31 of the algorithm will take $O(2.ec_i^F)$ time, which by the above relations can be bounded by $O(ec_i^I)$. Finally, returning the cycle structure (line 34) will take $O(ec_{n-1}^F)$ time. Thus total running time of our algorithm is $\sum_{i=0}^{n-1} O(lc_i^I + ec_i^I)$.

The analysis of the space complexity is similar. Line 4 of the algorithm creates lc_i^I number of links and ec_i^I number of edges for level i. Redundant links and edges are then removed to reduce these numbers respectively to lc_i^F and ec_i^F. Thus the space complexity of our algorithm is $O(\max_i (lc_i^I + ec_i^I))$.

Worst case analysis: The worst case in Algorithm 1 occurs if $ec_{n-1}^F = 2^n$. Thus, the space requirement is exponential and obviously, time requirement is also exponential.

An algorithm for detecting cyclic configuration(s) in an n-cell CA has already been reported in [64]. Using that algorithm, at a level of the tree an RMT `r` is removed from an edge if that edge is linked to a non-reachable edge due to that `r` and ultimately by removing such RMT a new non-reachable edge may be generated. Such property is supported by Corollary 3.3. The improvement of our technique (Algorithm 1) over the algorithm mentioned in [64] is that the RMT `r` of an edge can also be removed even if that edge is not linked to a non-reachable edge due to `r` (by Corollary 3.4 and Corollary 3.5).

It is already shown that even if we use irreversible CA where the count of acyclic configurations is exponential, then also the complexity of Algorithm 1 can be exponential if the number of available ec_{n-2}^F is 2^{n-1}. Thus the detection of acyclic configurations at early levels may reduce the complexity in space and time both. The hardness of this non trivial problem leads us to look into the cycle structure of reversible CA - where no edges can be discarded at any level

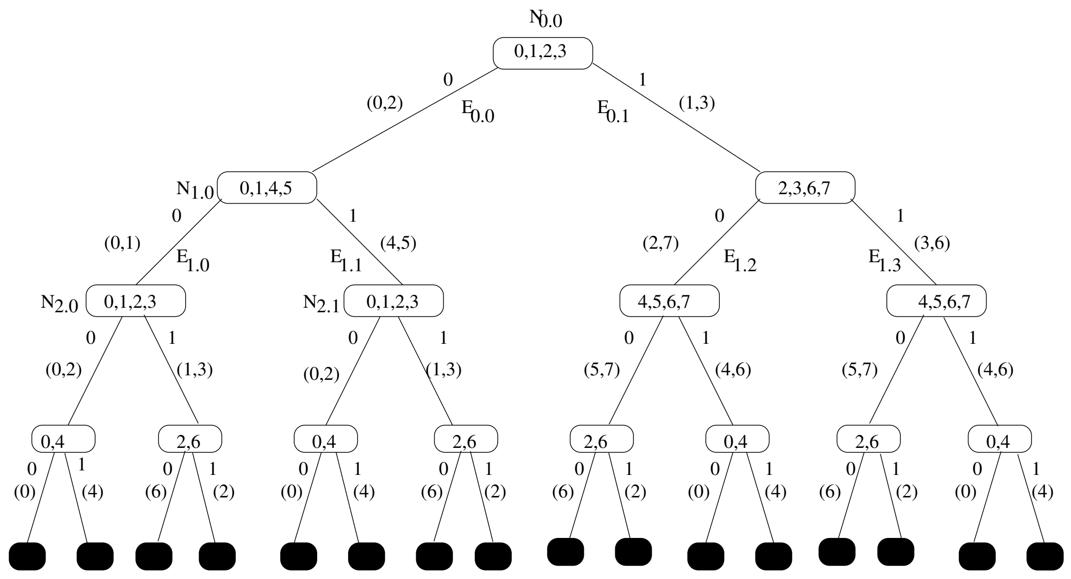

Figure 3.5: Header nodes in the reachability tree of CA $\langle 10, 120, 90, 20\rangle$

by the properties of acyclic configurations.

3.4 Cycle Structure of Reversible CAs

All configurations are cyclic in a reversible CA. Therefore, the technique mentioned in Section 3.3.2 requires the reachability tree to be explored upto level $n-1$ for evaluating the CS of an n-cell reversible CA. In every level, we get a collection of cross and self links, and all of these links are used to detect the CS. In this section, we figure out some properties of the links which contribute to the cycle structures of reversible CAs. In particular we shall focus on:

- Estimation of CS of an n-cell reversible CA by exploring the reachability tree only upto level i where $i < n$.
- Finding the exact CS of an n-cell reversible CA without exploring the entire tree.

We address these aspects by identifying certain special nodes in the reachability tree, which we term as *header* nodes, and then viewing the reachability tree as a collection of *subtrees* at those header nodes. Our target is to estimate or evaluate the number of cycles or lengths of cycles of a CA by exploring *minimum* number of such subtrees. Next, the concept of a subtree at a header node will be introduced to express its effectiveness in estimating or evaluating the cycle structure of a CA.

3.4.1 The Subtree

We begin with some useful definitions. All these definitions assume reversibility of the cellular automata under consideration.

Definition 3.3 *A node $N_{i.j_i}$ $(1 \leq i \leq n-1, 0 \leq j_i \leq 2^i - 1)$ in the reachability tree is said to be header node if the label $l_{i-1.j_i}$ of the edge $E_{i-1.j_i}$, connecting nodes $N_{i-1.\lfloor \frac{j_i}{2} \rfloor}$ to $N_{i.j_i}$, contains only one pair of sibling RMTs.*

Example 3.8 In the reachability tree of CA $\langle 10, 120, 90, 20 \rangle$ referred in Figure 3.5, nodes $N_{2.0}$ and $N_{2.1}$ are the *header* nodes as $l_{1.0}$ and $l_{1.1}$ respectively contain sibling RMT pairs $\{0, 1\}$ and $\{4, 5\}$.

Definition 3.4 *A subtree at header node $N_{i.j_i}$ $(1 \leq i \leq n-1, 0 \leq j_i \leq 2^i - 1)$ is given as the tree rooted at $N_{i.j_i}$, and is denoted by $\mathtt{st}(N_{i.j_i})$. The node $N_{i-1.\lfloor \frac{j_i}{2} \rfloor}$ and the edge $E_{i-1.j_i}$ are respectively called as the parent node and parent edge of header node $N_{i.j_i}$.*

Example 3.9 Figure 3.6 represents the *subtree* at header node $N_{2.0}$ of the reachability tree of CA $\langle 10, 120, 90, 20 \rangle$ (see Figure 3.5). This subtree denoted by $\mathtt{st}(N_{2.0})$, is marked by the dotted lines. The *parent node* is $N_{1.0}$ and the *parent edge* is $E_{1.0} = (N_{1.0}, N_{2.0}, l_{1.0})$.

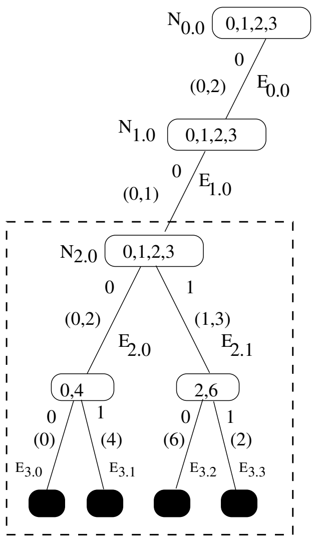

Figure 3.6: Subtree at header node $N_{2.0}$ in the reachability tree of CA $\langle 10, 120, 90, 20 \rangle$ is presented inside the dotted lines. Here, $N_{1.0}$ and $E_{1.0}$ are respectively the parent node and parent edge of $N_{2.0}$.

Now we present the effectiveness of subtrees at header nodes in evaluating the cycle structure of an n-cell CA in an intuitive manner with the help of an example. Let us consider a 4-cell reversible CA $\mathcal{R} = \langle 6, 204, 90, 20 \rangle$ having $\texttt{CS}_{\mathcal{R}} = [2(1), 1(2), 2(3), 1(6)]$. For evaluating this CS, Algorithm 1 explores all possible links and cross links at any level of the reachability tree of $\langle 6, 204, 90, 20 \rangle$ (see Figure 3.7). We observe that the reachability tree includes four subtrees at header nodes $N_{2.0}$, $N_{2.1}$, $N_{2.2}$ and $N_{2.3}$. Now the links generated at level $n-1$ in $\texttt{st}(N_{2.0})$, form *one* cross link $E_{3.1}(4) \rightarrow E_{3.2}(6) \rightarrow E_{3.3}(2) \rightarrow E_{3.1}$ of length *three* and *one* self link $E_{3.0}(0) \rightarrow E_{3.0}$. Another subtree $\texttt{st}(N_{2.2})$ generates the same length cross link $E_{3.9}(4) \rightarrow E_{3.10}(6) \rightarrow E_{3.11}(2) \rightarrow E_{3.9}$ and same number of self link i.e. one. Thus we observe that two subtrees have *resemblance* in terms of cyclic components, that is they produce identical number of cycles with identical lengths. In such case, it makes sense to process only one subtree to evaluate the *partial* cycle structures of multiple subtrees. Intuitively, the partial cycle structure of the subtree at a header node refers to the collection of the number of cycles along with the cycle lengths of that subtree. We formally define it in Section 3.5.

Next we focus on the header nodes $N_{2.1}$ and $N_{2.3}$. We see that $E_{1.1}$, the parent edge of $N_{2.1}$, is linked only to $E_{1.3}$, the parent edge of $N_{2.3}$ and vice versa. Therefore, the lengths of the elementary cross links formed among $E_{1.1}$ and $E_{1.3}$ is *two.* We will prove that the cycles generated by $\texttt{st}(N_{2.1})$ and $\texttt{st}(N_{2.3})$ will be of length that is multiple of this number. For instance in this CA, the cross links generated at level $n-1$ among $\texttt{st}(N_{2.1})$ and $\texttt{st}(N_{2.3})$ have lengths *two* and *six.* Hence when multiple subtrees at header nodes of level i are connected (via links), the cycles generated by those subtrees have lengths which are multiple of the elementary cross link's length generated at level $i-1$ among those subtrees.

Therefore, the self links and cross links of subtrees contribute in estimating or finding the cycle structure of a CA. Thus, we get that by processing subtrees, we can evaluate the cycle structure of the CA by processing the tree *partially.* The immediate question is, can we identify the existence of subtrees at header nodes in an arbitrary n-cell reversible CA without processing its reachability tree? To answer this question, let us analyse the CA rules for generating subtrees at header nodes. For ease of representation, we refer subtree at header node also by "subtree".

3.4.2 Effective CA Rules for Subtrees in the Reachability Tree

From Definition 3.4, we get that the presence of a header node $N_{i.j_i}$ at level i in the reachability tree of a CA, confirms the generation of subtrees in that

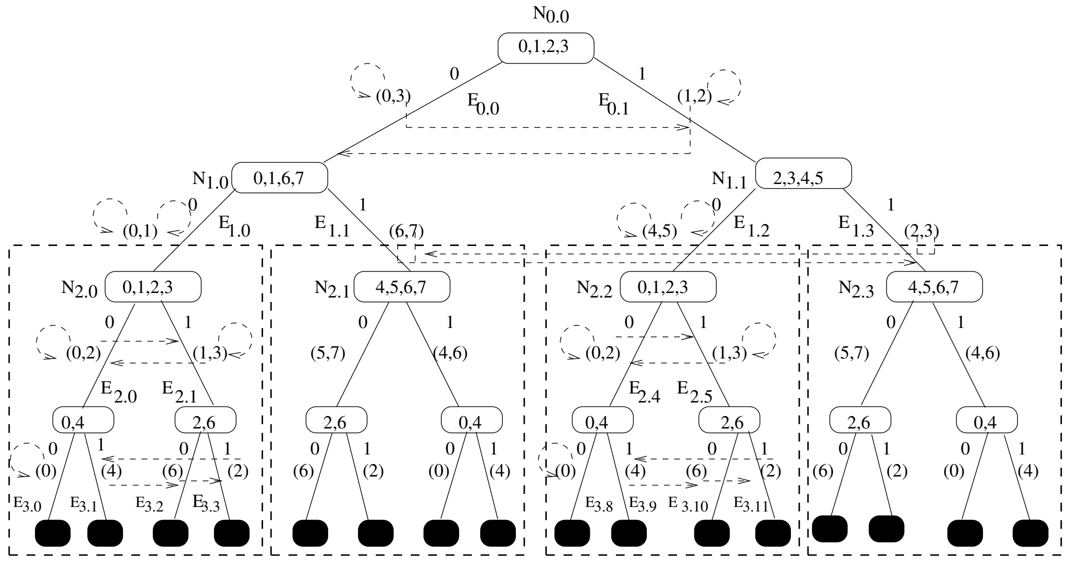

Figure 3.7: The reachability tree of CA $\langle 6, 204, 90, 20\rangle$ can be viewed as a collection of four subtrees. Here, both the subtrees at header nodes $N_{2.0}$ and $N_{2.2}$ produce one 1-cycle and one 3-cycle.

reachability tree. Now, $N_{i.j_i}$ is a header node if $l_{i-1.j_i} = \{\mathtt{r}, \mathtt{s}\}$ where $\mathtt{r}$ and $\mathtt{s}$ are sibling RMTs. Therefore, every $\mathcal{R}_{i-1}$ of the rule vector of an n-cell CA $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{i-1}, \cdots, \mathcal{R}_{n-1}\rangle$ decides whether a header node is present at level i in the reachability tree of that CA.

The header nodes can be generated from level 1 to level $n-1$. If $N_{1.0}$ is the header node, then $N_{1.1}$ is also another header node as $l_{0.0}$ contains one pair of self replicating RMTs, then $l_{0.1}$ also contains another pair of self replicating RMTs since every $l_{i.j}$ $(0 \le j \le 2^{i+1}-1)$ at each level i $(0 \le i \le n-1)$ is non-empty in the reachability tree of a reversible CA. Thus $\mathcal{R}_0$ can be either 3 or 12 if $N_{1.0}$ and $N_{1.1}$ are header nodes. In analogous manner, for other level $i \in \{2, 3, \cdots, n-1\}$, if $N_{i.j_i}$ is a header node, then $N_{i.j_i+1}$ be also the header node when j_i is an even number. It is already mentioned in Chapter 2 that the number of unique nodes at level i in the reachability tree of a reversible CA is either 2 or 4 (according to Theorem 2.3). Let us concentrate on the design of $\mathcal{R}_{i-1}$ to generate subtrees at header nodes in the reachability tree.

- When at level $i-1$, the number of unique nodes is 2 and the contents of the nodes of level $i-1$ are $\{0,1,2,3\}$ & $\{4,5,6,7\}$, consider the following cases.

  – If $N_{i.j_i}$ & $N_{i.j_i+1}$ are header nodes for some j_i where $0 \le j_i \le 2^i - 1$, then, either $l_{i-1.j_i} = \{0,1\}$ & $l_{i-1.j_i+1} = \{2,3\}$ or $l_{i-1.j_i} = \{2,3\}$ &

  $l_{i-1.j_i+1} = \{0, 1\}$ when $N_{i-1.\lfloor \frac{j_i}{2} \rfloor} = \{0, 1, 2, 3\}$. Therefore, $\mathcal{R}_{i-1}$ is any one from $\{83, 92, 99, 108, 147, 156, 163, 172\}$.
  - If $N_{i.j_i}$ & $N_{i.j_i+1}$ are header nodes for some j_i where $0 \le j_i \le 2^i - 1$, then, either $l_{i-1.j_i} = \{4, 5\}$ & $l_{i-1.j_i+1} = \{6, 7\}$ or $l_{i-1.j_i} = \{6, 7\}$ & $l_{i-1.j_i+1} = \{4, 5\}$ when $N_{i-1.\lfloor \frac{j_i}{2} \rfloor} = \{4, 5, 6, 7\}$. Therefore, $\mathcal{R}_{i-1}$ is any one from $\{53, 54, 57, 58, 197, 198, 201, 202\}$.
  - If all nodes of level i are header nodes, then, $\mathcal{R}_{i-1}$ is any one from $\{51, 60, 195, 204\}$.

- When at level $i - 1$, the number of unique nodes is 2 and the contents of the nodes of level $i - 1$ are $\{0, 1, 4, 5\}$ & $\{2, 3, 6, 7\}$, consider the following cases.
  - If $N_{i.j_i}$ & $N_{i.j_i+1}$ are header nodes for some j_i where $0 \le j_i \le 2^i - 1$, then, either $l_{i-1.j_i} = \{0, 1\}$ & $l_{i-1.j_i+1} = \{4, 5\}$ or $l_{i-1.j_i} = \{4, 5\}$ & $l_{i-1.j_i+1} = \{0, 1\}$ when $N_{i-1.\lfloor \frac{j_i}{2} \rfloor} = \{0, 1, 4, 5\}$. Therefore, $\mathcal{R}_{i-1}$ is any one from $\{75, 120, 135, 180\}$.
  - If $N_{i.j_i}$ & $N_{i.j_i+1}$ are header nodes for some j_i where $0 \le j_i \le 2^i - 1$, then, either $l_{i-1.j_i} = \{2, 3\}$ & $l_{i-1.j_i+1} = \{6, 7\}$ or $l_{i-1.j_i} = \{6, 7\}$ & $l_{i-1.j_i+1} = \{2, 3\}$ when $N_{i-1.\lfloor \frac{j_i}{2} \rfloor} = \{2, 3, 6, 7\}$. Therefore, $\mathcal{R}_{i-1}$ is any one from $\{30, 45, 210, 225\}$.
  - If all nodes of level i are header nodes, then, $\mathcal{R}_{i-1}$ is any one from $\{15, 60, 195, 240\}$.

- When at level $i - 1$, the number of unique nodes is 2 and the contents of the nodes of level $i - 1$ are $\{0, 1, 6, 7\}$ & $\{2, 3, 4, 5\}$, consider the following cases.
  - If $N_{i.j_i}$ & $N_{i.j_i+1}$ are header nodes for some j_i where $0 \le j_i \le 2^i - 1$, then, either $l_{i-1.j_i} = \{0, 1\}$ & $l_{i-1.j_i+1} = \{6, 7\}$ or $l_{i-1.j_i} = \{6, 7\}$ & $l_{i-1.j_i+1} = \{0, 1\}$ when $N_{i-1.\lfloor \frac{j_i}{2} \rfloor} = \{0, 1, 6, 7\}$. Therefore, $\mathcal{R}_{i-1}$ is any one from $\{23, 27, 39, 43, 212, 216, 228, 232\}$.
  - If $N_{i.j_i}$ & $N_{i.j_i+1}$ are header nodes for some j_i where $0 \le j_i \le 2^i - 1$, then, either $l_{i-1.j_i} = \{2, 3\}$ & $l_{i-1.j_i+1} = \{4, 5\}$ or $l_{i-1.j_i} = \{4, 5\}$ & $l_{i-1.j_i+1} = \{2, 3\}$ when $N_{i-1.\lfloor \frac{j_i}{2} \rfloor} = \{2, 3, 4, 5\}$. Therefore, $\mathcal{R}_{i-1}$ is any one from $\{77, 78, 113, 114, 141, 142, 177, 178\}$.
  - If all nodes of level i are header nodes, then, $\mathcal{R}_{i-1}$ is any one from $\{15, 51, 204, 240\}$.

- When at level $i - 1$, the number of unique nodes is 4, consider the following cases.

- Let us consider the contents of the nodes of level $i-1$ to be $\{0,1,2,3\}$ & $\{4,5,6,7\}$ and $\{0,1,4,5\}$ & $\{2,3,6,7\}$. If $N_{i.j_i}$ & $N_{i.j_i+1}$ are header nodes for some j_i where $0 \leq j_i \leq 2^i - 1$, therefore, each of $l_{i-1.j_i}$ and $l_{i-1.j_i+1}$ represents a pair of sibling RMTs. As a pair of sibling RMTs are the contents of two distinct unique nodes, the label of every edge of level $i-1$ contains only one pair of sibling RMTs. Therefore, all nodes of level $i-1$ are header nodes. Thus $\mathcal{R}_{i-1}[0] = \mathcal{R}_{i-1}[1] = \mathcal{R}_{i-1}[6] = \mathcal{R}_{i-1}[7]$ and $\mathcal{R}_{i-1}[2] = \mathcal{R}_{i-1}[3] = \mathcal{R}_{i-1}[4] = \mathcal{R}_{i-1}[5]$. Therefore, $\mathcal{R}_{i-1}$ is any one from $\{60, 195\}$.
- Let us consider the contents of the nodes of level $i-1$ to be $\{0,1,2,3\}$ & $\{4,5,6,7\}$ and $\{0,1,6,7\}$ & $\{2,3,4,5\}$. If $N_{i.j_i}$ & $N_{i.j_i+1}$ are header nodes for some j_i where $0 \leq j_i \leq 2^i - 1$, therefore, each of $l_{i-1.j_i}$ and $l_{i-1.j_i+1}$ represents a pair of sibling RMTs. As a pair of sibling RMTs are the contents of two distinct unique nodes, the label of every edge of level $i-1$ contains only one pair of sibling RMTs. Therefore, all nodes of level $i-1$ are header nodes. Thus $\mathcal{R}_{i-1}[0] = \mathcal{R}_{i-1}[1] = \mathcal{R}_{i-1}[4] = \mathcal{R}_{i-1}[5]$ and $\mathcal{R}_{i-1}[2] = \mathcal{R}_{i-1}[3] = \mathcal{R}_{i-1}[6] = \mathcal{R}_{i-1}[7]$. Therefore, $\mathcal{R}_{i-1}$ is any one from $\{51, 204\}$.
- Let us consider the content of the nodes of level $i-1$ to be $\{0,1,4,5\}$ & $\{2,3,6,7\}$ and $\{0,1,6,7\}$ & $\{2,3,4,5\}$. If $N_{i.j_i}$ & $N_{i.j_i+1}$ are header nodes for some j_i where $0 \leq j_i \leq 2^i - 1$, then, each of $l_{i-1.j_i}$ and $l_{i-1.j_i+1}$ represents a pair of sibling RMTs. As a pair of sibling RMTs are in two distinct unique nodes, the label of every edge of level $i-1$ contains only one pair of sibling RMTs. Therefore, all nodes of level $i-1$ are header nodes. Thus $\mathcal{R}_{i-1}[0] = \mathcal{R}_{i-1}[1] = \mathcal{R}_{i-1}[2] = \mathcal{R}_{i-1}[3]$ and $\mathcal{R}_{i-1}[4] = \mathcal{R}_{i-1}[5] = \mathcal{R}_{i-1}[6] = \mathcal{R}_{i-1}[7]$. Therefore, $\mathcal{R}_{i-1}$ is any one from $\{15, 240\}$.

In the above mentioned way, by the presence of $\mathcal{R}_{i-1}$ (when $i > 0$), we can decide whether the reachability tree of the given CA contains the subtrees at header nodes. In such case $\mathcal{R}_{i-1}$ should be selected from the set of 46 CA rules which are mentioned in Table 3.2. Let $\mathcal{H}$ denote that set. For example, the reachability tree of a 4-cell CA $\langle 10, 120, 90, 20 \rangle$ contains two subtrees at header nodes $N_{2.0}$ and $N_{2.1}$ at level 1 as $\mathcal{R}_1 = 120 \in \mathcal{H}$ (see Figure 3.5).

Table 3.2: The CA rules for Subtrees at Header nodes

204, 51, 60, 195, 53, 58, 83, 92, 163, 172, 197, 202, 54, 57, 99, 108,
147, 156, 198, 201, 15, 30, 45, 75, 120, 135, 180, 210, 225, 240, 23,
43, 77, 113, 142, 178, 212, 232, 27, 39, 78, 114, 141, 177, 216, 228

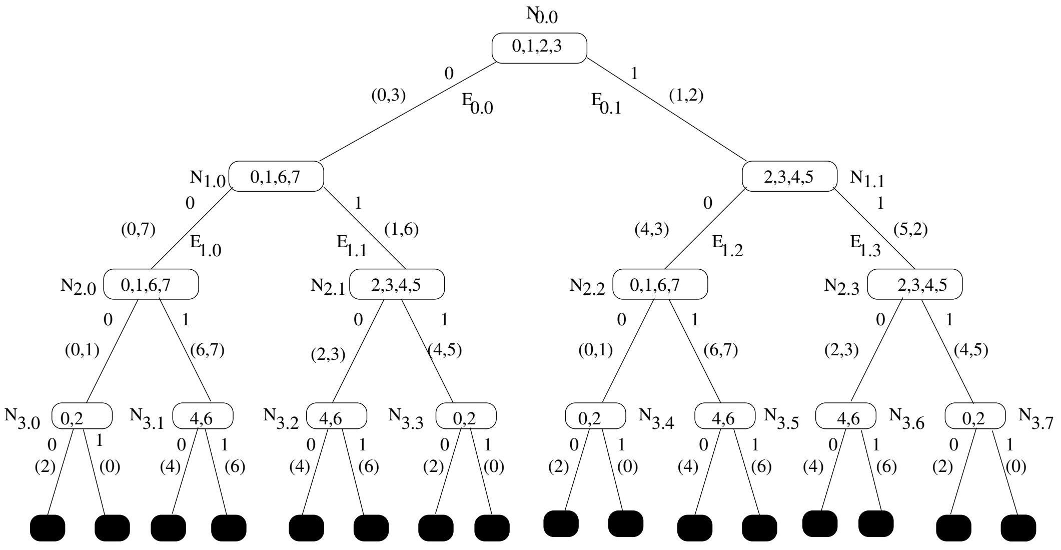

Figure 3.8: Reachability tree of CA $\langle 6, 102, 240, 65\rangle$

Here after, we focus on only those reversible cellular automata whose rule vectors contain at least one rule from $\mathcal{H}$. For all the conditions, lemmas, corollaries etc., reported next, the CAs use rules from $\mathcal{H}$ in their rule vectors.

3.4.3 The Intrinsic Properties of Subtrees

Let $N_{i.j_i}$ be a header node at level i of the reachability tree (where j_i is an even number). Obviously, $N_{i.j_i+1}$ is another header node. Therefore, in the reachability tree of a CA, the number of header nodes is always multiple of *two*. The following conditions are used to evaluate the number of subtrees at header nodes in the tree.

Condition 4 *Let $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{i-1}, \cdots, \mathcal{R}_{n-1}\rangle$ be the rule vector of an n-cell reversible CA. If $\mathcal{R}_{i-1} \in \{15, 240, 51, 204, 60, 195\}$, then there exist 2^i subtrees at header nodes of level i.*

Reason: Let $N_{i.j_i}$ be a header node in the reachability tree of an n-cell reversible CA $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1}\rangle$ (where $1 \le i \le n-1$ and $0 \le j_i \le 2^i - 1$). Let us also consider $\mathcal{R}_{i-1} \in \{15, 240, 51, 204, 60, 195\}$. So, every $l_{i-1.j_i}$ contains a pair of sibling RMTs. Thus for each j_i, $N_{i.j_i}$ is header node. Hence the number of header nodes at level i is 2^i.

Condition 5 *Let $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{i-1}, \cdots, \mathcal{R}_{n-1}\rangle$ be the rule vector of an n-cell reversible CA. If $\mathcal{R}_{i-1} \in \mathcal{H} \setminus \{15, 240, 51, 204, 60, 195\}$ where $\mathcal{H}$ is the set*

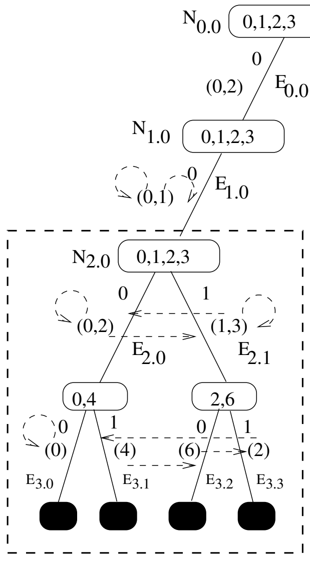

Figure 3.9: intra-linked subtree: $\texttt{st}(N_{2.0})$ with links

of rules given in Table 3.2, then there exist 2^{i-1} *subtrees at header nodes of level* i.

Reason: Let us consider $N_{i.j_i}$ and $N_{i.j_i+1}$ to be the header nodes in the reachability tree of an n-cell reversible CA $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ (where $1 \le i \le n-1$, $0 \le j_i \le 2^i - 1$ and j_i is an even number). Let us also consider $\mathcal{R}_{i-1} \in \mathcal{H} \setminus \{15, 240, 51, 204, 60, 195\}$. So, the number of unique nodes at level $i-1$ is 2. Since the total number of nodes at level $i-1$ is 2^{i-1}, $\frac{2^{i-1}}{2}$ nodes contain the contents of node $N_{i-1.\lfloor \frac{j_i}{2} \rfloor}$. As each of $l_{i-1.j_i}$ and $l_{i-1.j_i+1}$ contains a pair of sibling RMTs, $\frac{2^i}{2}$ header nodes are generated at level i (as the total number of nodes at level $i-1$ is $\frac{2^{i-1}}{2}$).

Example 3.10 Consider the reachability tree of CA $\langle 10, 120, 90, 20 \rangle$ as in Figure 3.5. As $\mathcal{R}_1 = 120$, according to Condition 5, *two* subtrees are generated. Figure 3.5 shows that $N_{2.0}$ and $N_{2.1}$ are *two* header nodes at level 2, out of 4 nodes. Therefore, subtrees $\texttt{st}(N_{2.0})$ (see Figure 3.6) and $\texttt{st}(N_{2.1})$ are generated in that tree. Figure 3.8 represents that at level 3 of the reachability tree of CA $\langle 6, 102, 240, 65 \rangle$, all the nodes are header nodes as $\mathcal{R}_2 = 240$ (according to Condition 4). Therefore, the reachability tree contains *eight* subtrees which are denoted as $\texttt{st}(N_{3.0})$ to $\texttt{st}(N_{3.7})$.

If we get more than one rule from $\mathcal{H}$ (which is the set of CA rules mentioned in Table 3.2) in the rule vector of a reversible CA, then the subtrees are generated at different levels. For example, the 5-cell CA $\langle 6, 178, 105, 204, 20 \rangle$ generates header nodes at levels 2 and 4 as $\mathcal{R}_1 = 178$ and $\mathcal{R}_3 = 204$ are from $\mathcal{H}$. Here, at level 2, the header nodes are $N_{2.2}$ and $N_{2.3}$. At level 4, all *sixteen* nodes are header nodes (according to Condition 4). Therefore, this reachability tree contains *two* subtrees at header nodes of level 2 and *sixteen* subtrees at header nodes of level 4. Out of the sixteen header nodes of level 4, $N_{4.8}$ to $N_{4.11}$ and $N_{4.12}$ to $N_{4.15}$ belong to $\texttt{st}(N_{2.2})$ and $\texttt{st}(N_{2.3})$ respectively. This brings the notion that a subtree can be *contained* in another subtree. Formally, for two header nodes $N_{i.j_i}$ $(1 \leq i \leq n-1$ and $0 \leq j_i \leq 2^i - 1)$ and $N_{i_1.j_{i_1}}$ $(i < i_1)$ of the reachability tree, the subtree at $N_{i_1.j_{i_1}}$ is *contained* in the subtree at $N_{i.j_i}$ if $j_i = \lfloor j_{i_1}/2^{i_1-i} \rfloor$. In CA $\langle 6, 178, 105, 204, 20 \rangle$, the subtrees $\texttt{st}(N_{4.9})$ to $\texttt{st}(N_{4.11})$ are contained in $\texttt{st}(N_{2.2})$ and similarly, $\texttt{st}(N_{4.12})$ to $\texttt{st}(N_{4.15})$ are contained in $\texttt{st}(N_{2.3})$.

Till now, we have discussed about the subtrees and the set of CA rules for generating the subtrees. Let us go back to estimate or evaluate the cycle structure of an n-cell reversible cellular automaton. To this effect, we introduce some useful terminologies on subtrees that will be used throughout this chapter.

Definition 3.5 *A subtree* $\texttt{st}(N_{i.j_i})$ $(1 \leq i \leq n-1, 0 \leq j_i \leq 2^i - 1)$ *is called as intra-linked subtree if* $E_{i-1.j_i}$ *is only involved in self link(s). It is denoted as* $\texttt{sta}(N_{i.j_i})$.

Example 3.11 Let us consider a subtree $\texttt{st}(N_{2.0})$ (Figure 3.9) with links in the reachability tree of CA $\langle 10, 120, 90, 20 \rangle$ (shown in Figure 3.5). At level 1, two self links are formed: $E_{1.0}(0) \rightarrow E_{1.0}$ and $E_{1.0}(1) \rightarrow E_{1.0}$. Therefore, $E_{1.0}$ is involved only in self links. Thus, $\texttt{st}(N_{2.0})$ is *intra-linked subtree.*

Obviously, an intra-linked subtree generates some cycles of CA. For CA $\langle 10, 120, 90, 20 \rangle$, the subtree $\texttt{st}(N_{2.0})$ (see Figure 3.9) generates *two* cycles where every configuration involved in those cycles has 00 as the *prefix* sub-configuration for 2 cells. Now, for $\texttt{st}(N_{i.j_i})$ to be an intra-linked subtree, we require $E_{i-1.j_i}$ to be self linked only; thus for $\mathcal{R}_{i-1}$, there exists at least one pair of self replicating sibling RMTs. But out of 46 rules, we get some rules whose sibling RMTs are not self replicating and these rules have no contribution in generating intra-linked subtrees in reachability tree. Hence, $\mathcal{R}_{i-1}$ is selected from a proper subset of $\mathcal{H}$ which is represented as $\mathcal{H}_1 = \{204, 60, 195, 92, 172, 197, 202, 108, 156, 198, 201, 15, 30, 45, 75, 120, 135, 180, 210, 225, 240, 77, 142, 212, 232, 78, 141, 216, 228\}$. Therefore, out of 46 rules which create subtrees in the reachability trees of non-uniform reversible cellular automata, only 29 rules are used to create reachability tree with intra-linked subtrees. We may also get a collection of subtrees which are connected via links.

Definition 3.6 *A collection of subtrees* $\texttt{st}(N_{i.j_i^1})$, $\texttt{st}(N_{i.j_i^2})$, $\cdots$, $\texttt{st}(N_{i.j_i^q})$ $(1 \leq i \leq n-1, 0 \leq j_i^1, j_i^2, \cdots, j_i^q \leq 2^i - 1)$ *is said to be inter-linked if* $E_{i-1.j_i^1}$ *is only linked to* $E_{i-1.j_i^2}$*,* $E_{i-1.j_i^2}$ *is only linked to* $E_{i-1.j_i^3}$ *and in this way,* $E_{i-1.j_i^q}$ *is only linked to* $E_{i-1.j_i^1}$*. It is denoted by* $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$*. Any subtree belonging to* $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ *is termed as a component of the collection.*

Example 3.12 Let us consider the subtrees $\texttt{st}(N_{3.1})$, $\texttt{st}(N_{3.7})$ and $\texttt{st}(N_{3.2})$ (Figure 3.10) of CA $\langle 6, 102, 240, 65 \rangle$ (shown in Figure 3.8). Here, at level 2, we get that $E_{2.1}$ is linked to only $E_{2.7}$, $E_{2.7}$ is only linked to $E_{2.3}$ and $E_{2.3}$ is linked to only $E_{2.1}$. Thus $E_{2.1}$, $E_{2.7}$ and $E_{2.3}$ are involved to form elementary cross links of length three. Therefore, we get an inter-linked collection of subtrees and is denoted as $\texttt{str}(N_{3.1}, N_{3.7}, N_{3.2})$. Here, $\texttt{st}(N_{3.1})$, $\texttt{st}(N_{3.7})$ and $\texttt{st}(N_{3.2})$ are the components of the collection.

Here, $\texttt{str}(N_{3.1}, N_{3.7}, N_{3.2})$ (in Figure 3.10) generates a cycle of length *six* whereas the lengths of the elementary cross links among the parent edges of $N_{3.1}$, $N_{3.7}$ and $N_{3.2}$ are *three.* Thus the cycles generated by $\texttt{str}(N_{3.1}, N_{3.7}, N_{3.2})$ is multiple of the length of the elementary cross links among the parent edges of $N_{3.1}$, $N_{3.7}$ and $N_{3.2}$. Moreover, in the cycle generated by $\texttt{str}(N_{3.1}, N_{3.7}, N_{3.2})$, a sequence of *prefix* sub-configurations for 3 cells $(001, 111, 010)$ repeats.

It is already established that to get intra-linked subtree, only 29 rules are used. Like that, let us see whether all the members of $\mathcal{H}$ are used to generate reachability trees with the inter-linked collection of subtrees. Let us consider an inter-linked collection of q subtrees $\texttt{st}(N_{i.j_i^1}), \texttt{st}(N_{i.j_i^2}), \cdots, \texttt{st}(N_{i.j_i^q})$. Based on the above definition, the parent edges of the header nodes $N_{i.j_i^1}$, $N_{i.j_i^2}$, $\cdots$ and $N_{i.j_i^q}$ are linked in the given manner: $E_{i-1.j_i^1}$ is linked to $E_{i-1.j_i^2}$, $E_{i-1.j_i^2}$ is linked to $E_{i-1.j_i^3}$, and in this way $E_{i-1.j_i^q}$ is linked to $E_{i-1.j_i^1}$. Let $l_{i-1.j_i^1} = \{\texttt{r}_i^1, \texttt{s}_i^1\}$ and $l_{i-1.j_i^q} = \{\texttt{r}_i^q, \texttt{s}_i^q\}$. We next consider two possible scenarios. For the first one, let $N_{i.j_i^1} = N_{i.j_i^q}$. This implies either $l_{i-1.j_i^1} = l_{i-1.j_i^q}$, where $\texttt{r}_i^1 = \texttt{r}_i^q$ and $\texttt{s}_i^1 = \texttt{s}_i^q$ are self replicating RMTs or $l_{i-1.j_i^1} \neq l_{i-1.j_i^q}$ where $\texttt{r}_i^1$ and $\texttt{r}_i^q$ are equivalent RMTs as well as $\texttt{s}_i^1$ and $\texttt{s}_i^q$ are also equivalent RMTs. Thus $\mathcal{R}_{i-1}$ is designed in such a manner that at least one pair of sibling RMTs is self replicating. For the other scenario, $N_{i.j_i^1} \neq N_{i.j_i^q}$. This implies $l_{i-1.j_i^1} \neq l_{i-1.j_i^q}$. In this case, $l_{i-1.j_i^1}$ and $l_{i-1.j_i^1+1}$ (where j_i^1 is an even number) always contain two different pairs of sibling RMTs. Similarly, $l_{i-1.j_i^q}$ and $l_{i-1.j_i^q+1}$ (such that j_i^q is an even number) also contain different pairs of sibling RMTs. In such case, $\mathcal{R}_{i-1}$ is selected from $15, 240, 51, 205, 60, 195$. Thus $\mathcal{R}_{i-1}$ is selected from a pool of 30 rules which is a proper subset of $\mathcal{H}$ and represented as $\mathcal{H}_2 = \{51, 204, 60, 195, 92, 172, 197, 202, 108, 156, 198, 201, 15, 30, 45, 75, 120, 135, 180, 210, 225, 240, 77, 142, 212, 232, 78, 141, 216, 228\}$. Therefore, though 46 rules contribute in generating reachability trees containing subtrees, but at most 30 rules from them are contributing in generating intra-linked subtrees and the inter-

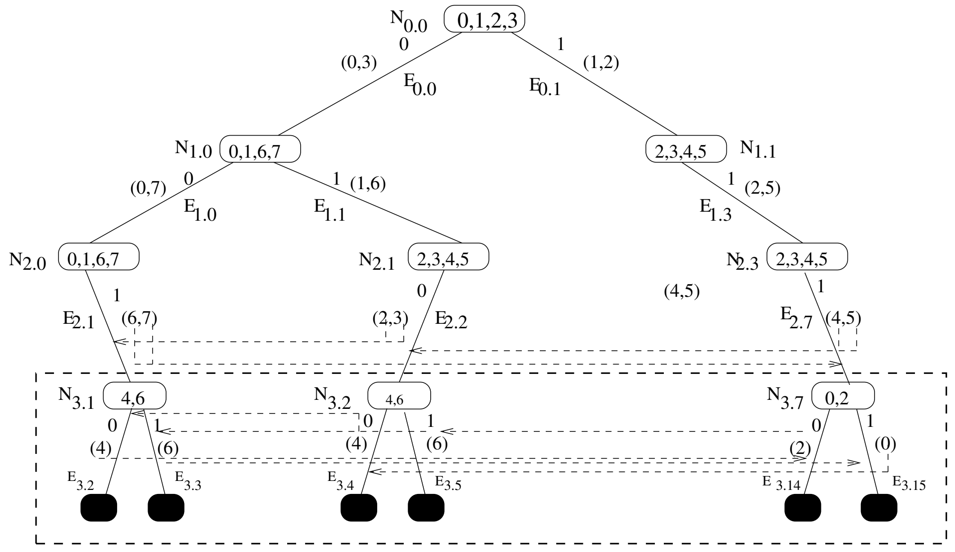

Figure 3.10: An inter-linked collection of subtrees in the reachability tree of CA $\langle 6, 102, 240, 65\rangle$. By linking the subtrees $\texttt{st}(N_{3.1})$, $\texttt{st}(N_{3.7})$ and $\texttt{st}(N_{3.2})$, we present $\texttt{str}(N_{3.1}, N_{3.7}, N_{3.2})$ inside the dotted lines.

linked collections of subtrees.

We shall next focus on how the intra-linked subtrees or inter-linked collection of subtrees are useful for estimating the cycle structures of reversible cellular automata.

3.4.4 Estimation of CS of Reversible CA

One way to detect the CS of a reversible CA is to explore the reachability tree till level $n-1$ and identify the cross links and self links at level $n-1$. However, it may not be always necessary to explore upto all the levels, and we can target to estimate the cycle structure of n-cell CA by discovering some properties of links at some earlier level i $(< n)$. Let us start the exploration with self links - the count of self links at level i in estimating CS of a CA.

Corollary 3.6 *Suppose there does not exist any self link at level i $(< n)$ in the reachability tree of an n-cell CA $\mathcal{R}$. Then such CA does not contribute any single length cycle.*

Proof: Suppose every link $E_{i.j}(\mathtt{r}) \rightarrow E_{i.k}$ at a particular level i $(< n)$ is either forward link or backward link. Therefore, every link derived from each $E_{i.j}(\mathtt{r}) \rightarrow E_{i.k}$ at level $i+1$ is also either forward link or backward link and it continues upto level $n-1$. Thus no cycle of length one (single length cycle) can be generated by $\mathcal{R}$ as no self link is present at level $n-1$ (according to Corollary 3.1). Moreover, as every link of a level is involved in loop formation, then, the links of level $n-1$ are only involved in cross links. Therefore, the minimum length of the cross link is 2. Thus the cycle structure of that CA is $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \mu_2(l_2), \cdots, \mu_m(l_m)]$ where $l_1 > 1$. Hence the proof follows. $\square$

By Corollary 3.6, it is determined that the CA generates only cycles of length more than one. Further, we target to estimate the count of cycles or the lengths of cycles of a CA by the help of intra-linked subtrees and inter-linked collection of subtrees without exploring the entire tree.

The parent edge of an intra-linked subtree is always linked to itself. Similarly, every parent edge of an inter-linked collection of subtrees is always linked to some unique parent edge of that collection. Thus the presence of intra-linked subtree or inter-linked collection of subtrees partitions the reachability tree horizontally. The existence of intra-linked subtrees or inter-linked collection of subtrees and the lengths of the elementary cross links among the parent edges of inter-linked collection of subtrees guide us to estimate either the total number of cycles or the largest cycle length of a reversible CA.

Lemma 3.2 *Let the reachability tree of an n-cell reversible CA $\mathcal{R}$ contain only one intra-linked subtree* $\texttt{sta}(N_{i.j_i})$ *where* $1 \le i < n-1$ *and* $0 \le j_i \le 2^i - 1$. *Let* $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \cdots, \mu_m(l_m)]$ *be the cycle structure of CA $\mathcal{R}$. Then,* $(\mu_1 + \cdots + \mu_m) \ge 2$ *and* $l_m \le 2^n - 2^{n-i-1}$.

Proof: As $\texttt{sta}(N_{i.j_i})$ be an intra-linked subtree, $E_{i-1.j_i}$ (the parent edge of $N_{i.j_i}$) generates only two self links. Therefore, at level $n-1$, total 2^{n-i-1} links are derived in $\texttt{sta}(N_{i.j_i})$ (following Property 3.1). All these links are involved to form cross links and self links only among themselves. Thus $\texttt{st}(N_{i.j_i})$ generates at least *one* cycle. The remaining $2^n - 2^{n-i-1}$ links are used to generate cross link(s) and self link(s). Since $\texttt{sta}(N_{i.j_i})$ is the only intra-linked subtree, all 2^{n-i-1} links generated in $\texttt{sta}(N_{i.j_i'})$ (such that $\lfloor \frac{j_i}{2} \rfloor = \lfloor \frac{j_i'}{2} \rfloor$) at level $n-1$ are involved in μ' cross link(s) where $\mu' \ge 1$. Thus the maximum length of a cross link is at most $\le 2^n - 2^{n-i-1}$. This implies $(\mu_1 + \cdots + \mu_m) > \mu' + 1 \ge 2$ and $l_m \le 2^n - 2^{n-i-1}$. Hence the proof follows. $\square$

Corollary 3.7 *Let the reachability tree of an n-cell reversible CA $\mathcal{R}$ contain v (> 1) intra-linked subtrees rooted at v header nodes of level i where* $1 \le i < n-1$. *Let* $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \cdots, \mu_m(l_m)]$ *be the cycle structure of CA $\mathcal{R}$. Therefore,* $\mu_1 + \cdots + \mu_m \ge v + 1$ *and* $l_m \le \texttt{max}(2^n - v \cdot 2^{n-i-1}, 2^{n-i-1})$.

Proof: According to Lemma 3.2, if one intra-linked subtree generates at least one cycle, therefore, v such subtrees generate at least v cycles. As every subtree generates 2^{n-i-1} links at level $n-1$, therefore, these v cycles are formed using $v * 2^{n-i-1}$ links. The remaining v subtrees which are not intra-linked subtrees, generate only cross links. As $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \cdots, \mu_m(l_m)]$, therefore, $\mu_1 + \cdots + \mu_m \geq v+1$. The largest length of a cross link is at most of the maximum of 2^{n-i-1} or $2^n - v \cdot 2^{n-i-1}$. This completes the proof. □

Lemma 3.3 *Let the reachability tree of an n-cell reversible CA $\mathcal{R}$ contain $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ where $1 \leq i < n-1$ and $0 \leq j_i^1, j_i^2, \cdots, j_i^q \leq 2^i - 1$. Let $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \cdots, \mu_m(l_m)]$ be the cycle structure of CA $\mathcal{R}$. Then, there exists at least one k $(1 \leq k \leq m)$ for which l_k is multiple of q and $\mu_k(l_k)$ be a cyclic component of $\texttt{CS}_{\mathcal{R}}$.*

Proof: The parent edges $E_{i-1.j_i^1}, E_{i-1.j_i^2}, \cdots, E_{i-1.j_i^q}$ of respectively header nodes $N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q}$ always form elementary cross links of length q and compound cross links of length multiple of q. Now, for any cross link generated at level $n-1$ in $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$, if an edge is selected from $\texttt{st}(N_{i.j_i^1})$, therefore, after q times, edge is to be selected from $\texttt{st}(N_{i.j_i^1})$ again. Therefore, $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ generates at least one cross link of a length l_k where l_k is multiple of q. As an intra-linked collection of subtrees generates a cycle of CA, therefore, there exist μ_k cycles of length l_k where $\mu_k \geq 1$. Thus k should be some value from 1 to m where $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \cdots, \mu_m(l_m)]$. Hence the proof follows. □

Corollary 3.8 *Let the reachability tree of an n-cell reversible CA $\mathcal{R}$ contain $(v > 1)$ inter-linked collections of subtrees at level i where $1 \leq i < n-1$ and the first collection has q_1 subtrees, second collection has q_2 subtrees and in this way v^{th} collection has q_v subtrees. Let $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \cdots, \mu_m(l_m)]$ be the cycle structure of CA $\mathcal{R}$. Therefore, there exist at least v cycles where the length of any of those v cycles is multiple of some p and that p is from $\{q_1, q_2, \cdots, q_v\}$ and there exists $\mu_k(l_k)$ at least for one $k \in \mathbb{N}$, for which l_k is multiple of p.*

Proof: Let for the first inter-linked collection of subtrees, q_1 subtrees are considered. According to Lemma 3.3, any cycle generated in the first inter-linked collection of subtrees is of length of multiple of q_1. Let l_{11} be a length of an arbitrary cycle which is generated by within the first inter-linked collection of subtrees and obviously, l_{11} is multiple of q_1. In analogous manner, in the second inter-linked collection of subtrees, we get an arbitrary cycle of length l_{12} which is multiple of q_2. Thus in the v^{th} inter-linked collection of subtrees, we also get an arbitrary cycle of length l_{1v} which is multiple of q_v. Therefore, we get at least v cycles having lengths $l_{11}, l_{12}, \cdots, l_{1v}$. As $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \cdots, \mu_m(l_m)]$, then there exists a cyclic component $\mu_k(l_k)$ in $\texttt{CS}_{\mathcal{R}}$ at least for one k $(1 \leq k \leq m)$ such that l_k is multiple of $q \in \{q_1, q_2, \cdots, q_v\}$. Hence the proof follows. □

It is already established from the above lemmas and corollaries that by detecting the existence of intra-linked subtrees or the inter-linked collections of subtrees at level i ($< n$), we can estimate the cycle structure of a CA. Moreover, every such subtree or collection of subtrees contains some cycles of the CA. Therefore, our target is to figure out the intrinsic properties of reachability tree for evaluating the cycle structure of an n-cell reversible CA by exploring the reachability tree *partially*.

3.5 Partial Cycle Structure

Let $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \mu_2(l_2), \cdots, \mu_m(l_m)]$ be the cycle structure of an n-cell reversible CA. Let $\texttt{sta}(N_{i.j_1})$ be contained in the reachability tree of $\mathcal{R}$ and it generates $\mu_1^{j_1}$ cycles of length $l_1^{j_1}$, $\mu_2^{j_1}$ cycles of length $l_2^{j_1}, \cdots$, and in the same way $\mu_{m_1}^{j_1}$ cycles of length $l_{m_1}^{j_1}$. Therefore, $\texttt{sta}(N_{i.j_1})$ produces a collection of number of cycles based on the cycle lengths of the CA i.e., $\texttt{sta}(N_{i.j_1})$ refers a collection of cyclic components. This brings the notion of partial cycle structure.

Definition 3.7 *The collection of cyclic components of an intra-linked subtree or an inter-linked collection of subtrees is called as partial cycle structure of the respective intra-linked subtree or an inter-linked collection of subtrees. Suppose, $\mu_1^{j_1}(l_1^{j_1}), \mu_2^{j_1}(l_2^{j_1}), \cdots, \mu_{m_1}^{j_1}(l_{m_1}^{j_1})$ be the cyclic components of an intra-linked subtree $\texttt{sta}(N_{i.j_1})$. Then the partial cycle structure of $\texttt{sta}(N_{i.j_1})$ is represented as $\texttt{PCS}_{\texttt{sta}(N_{i.j_1})} = [\mu_1^{j_1}(l_1^{j_1}), \mu_2^{j_1}(l_2^{j_1}), \cdots, \mu_{m_1}^{j_1}(l_{m_1}^{j_1})]$. Similar to intra-linked, we can also get the partial cycle structure of an inter-linked collection of subtrees and it is denoted by $\texttt{PCS}_{\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})}$ where $j_i^1, j_i^2, \cdots, j_i^q$ are some unique integer values from 0 to $2^i - 1$.*

We define a relation $\leq$ between $\texttt{PCS}_{\texttt{sta}(N_{i.j_1})}$ and the CS of an n-cell CA $\mathcal{R}$ as follows: we say that $\texttt{CS}_{\mathcal{R}} \leq \texttt{PCS}_{\texttt{sta}(N_{i.j_1})}$, if for every element of form $\mu_{k_1}^{j_1}(l_{k_1}) \in \texttt{PCS}_{\texttt{sta}(N_{i.j_1})}$ where $1 \leq k_1 \leq m_1$, there exists a cyclic component $\mu_k(l_k)$ $(1 \leq k \leq m)$ in $\texttt{CS}_{\mathcal{R}}$ where $\mu_k \geq \mu_{k_1}^{j_1}$ and $l_k = l_{k_1}^{j_1}$. Let there exist another intra-linked subtree $\texttt{sta}(N_{i.j_2})$ in the reachability of that n-cell CA $\mathcal{R}$. Let $\texttt{PCS}_{\texttt{sta}(N_{i.j_2})} = [\mu_1^{j_2}(l_1^{j_2}), \mu_2^{j_2}(l_2^{j_2}), \cdots, \mu_{m_2}^{j_2}(l_{m_2}^{j_2})]$. Using $\texttt{PCS}_{\texttt{sta}(N_{i.j_1})}$ and $\texttt{PCS}_{\texttt{sta}(N_{i.j_2})}$, we can find the resultant partial cycle structure. Let $+$ be the operator which is defined on $\texttt{PCS}_{\texttt{sta}(N_{i.j_1})}$ and $\texttt{PCS}_{\texttt{sta}(N_{i.j_2})}$ to figure out the total cyclic components of $\texttt{sta}(N_{i.j_1})$ and $\texttt{sta}(N_{i.j_2})$. Therefore, $\texttt{PCS}_{\texttt{sta}(N_{i.j_1})} + \texttt{PCS}_{\texttt{sta}(N_{i.j_2})}$ can be defined as $[\mu_1(l_1), \mu_2(l_2), \cdots, \mu_{m'}(l_{m'})]$ where m' is a positive integer and $m' \leq m_1 + m_2$. Let $\mu_{k'}$ be the count of cycle(s) of length $l_{k'}$ and $\mu_{k'}$ is defined as follows .

$$\mu_{k'} = \begin{cases} \mu_{k_1}^{j_1} + \mu_{k_2}^{j_2} & \text{if } l_{k_1}^{j_1} = l_{k_2}^{j_2} = l_{k'} \\ \mu_{k_1}^{j_1} & \text{if } l_{k'} = l_{k_1}^{j_1} \text{ and } l_{k'} \neq l_i^{j_2}, \forall i \\ \mu_{k_2}^{j_2} & \text{if } l_{k'} = l_{k_2}^{j_2} \text{ and } l_{k'} \neq l_i^{j_1}, \forall i \end{cases}$$

Thus the cycle structure of a reversible CA can be found using the partial cycle structures of intra-linked subtree(s) and (or) the inter-linked collection(s) of subtrees are contained in the reachability tree of that CA.

Example 3.13 Let us consider $\mathtt{sta}(N_{3.0})$ and $\mathtt{str}(N_{3.1}, N_{3.7}, N_{3.2})$ (depicted in Figure 3.8) in the reachability tree of CA $\mathcal{R} = \langle 6, 102, 240, 65\rangle$. Here, $\mathtt{PCS}_{\mathtt{sta}(N_{3.0})} = [1(2)]$ and $\mathtt{PCS}_{\mathtt{str}(N_{3.1},N_{3.7},N_{3.2})} = [1(6)]$. Then the resultant cycle structure is $\mathtt{PCS}_{\mathtt{sta}(N_{3.0})} + \mathtt{PCS}_{\mathtt{str}(N_{3.1},N_{3.7},N_{3.2})} = [1(2), 1(6)]$.

As the partial cycle structure of an intra-linked subtree or an inter-linked collection of subtrees can generate some cycles of CA, thus we are motivated to find some *resemblance* among the intra-linked subtrees or among the inter-linked collections of subtrees with the objective that the cycles generated in one intra-linked subtree or in an inter-linked collection of subtrees can guide us to evaluate the cycle structure of an n-cell reversible CA. To identify the resemblance, first we have to figure out the resemblance among links.

Definition 3.8 *Let $E_{i.j_1}(\mathtt{r}_1) \rightarrow E_{i.j_2}$ and $E_{i.j_3}(\mathtt{r}_2) \rightarrow E_{i.j_4}$ be two links at level i and these two links are said to be identical links if either $\mathtt{r}_1$ and $\mathtt{r}_2$ are equivalent RMTs or $\mathtt{r}_1 = \mathtt{r}_2$.*

Example 3.14 *In CA $\langle 6, 102, 240, 65\rangle$ (Figure 3.11), $E_{1.1}(6) \rightarrow E_{1.3}$ and $E_{1.3}(2) \rightarrow E_{1.1}$ are identical links. The self links $E_{1.0}(0) \rightarrow E_{1.0}$ and $E_{1.2}(4) \rightarrow E_{1.2}$ are also identical.*

Inference 3.1 *Let $E_{i.j_1}(\mathtt{r}_1) \rightarrow E_{i.j_2}$ and $E_{i.j_3}(\mathtt{r}_2) \rightarrow E_{i.j_4}$ be the identical links. The derived links $E_{i+1.j_1'}(\mathtt{r}_1') \rightarrow E_{i+1.j_2'}$ and $E_{i+1.j_3'}(\mathtt{r}_2') \rightarrow E_{i+1.j_4'}$ are also identical links where $\mathtt{r}_1' = \mathtt{r}_2'$.*

In this way, we get the identical links among the subtrees rooted at some header nodes.

3.5.1 Isomorphism in Subtrees

If a link $E_{i.j_1}(\mathtt{r}_1) \rightarrow E_{i.j_2}$ is identical to $E_{i.j_3}(\mathtt{r}_2) \rightarrow E_{i.j_4}$, then we get some resemblance in the reachability tree as every link of a particular level (from level

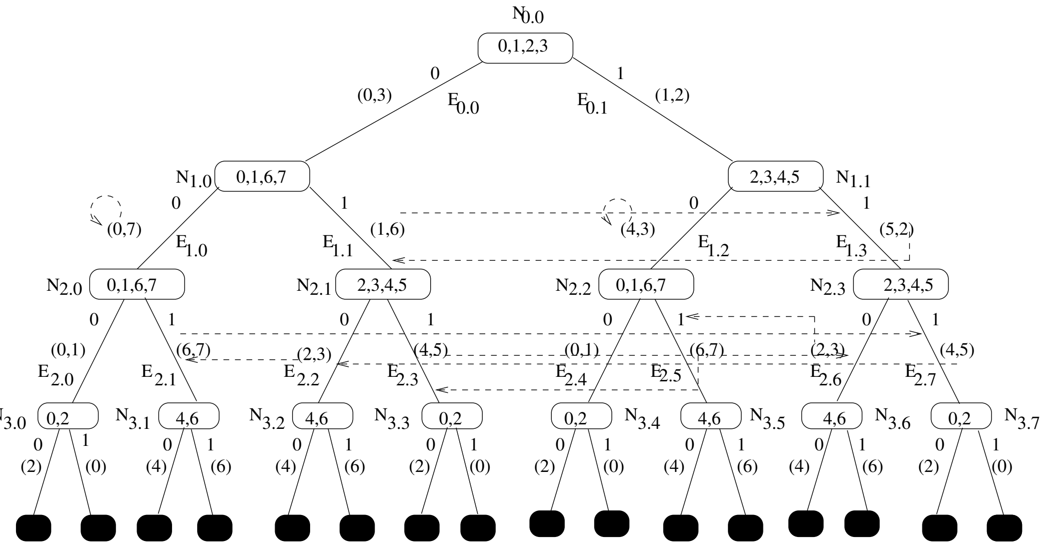

Figure 3.11: Some identical links in CA $\langle 6, 102, 240, 65\rangle$

$i+1$ to level $n-1$) derived from $E_{i.j_1}(\mathtt{r}_1) \rightarrow E_{i.j_2}$ is identical to an unique link of the derived links of $E_{i.j_3}(\mathtt{r}_2) \rightarrow E_{i.j_4}$ at that corresponding level (from Inference 3.1). This brings the notion of *isomorphism* among the subtrees rooted at header nodes.

Definition 3.9 *Let* $\mathtt{sta}(N_{i.j_i^1})$ *and* $\mathtt{sta}(N_{i.j_i^2})$ *be two intra-linked subtrees. We say* $\mathtt{sta}(N_{i.j_i^2})$ *is isomorphic to* $\mathtt{sta}(N_{i.j_i^1})$ *iff* $E_{i-1.j_i^1}(\mathtt{r}_1) \rightarrow E_{i-1.j_i^1}$ *and* $E_{i-1.j_i^2}(\mathtt{r}_2) \rightarrow E_{i-1.j_i^1}$ *are identical links for all values of* $\mathtt{r}_1, \mathtt{r}_2$*.*

Example 3.15 In Figure 3.12, $\mathtt{sta}(N_{3.4})$ is *isomorphic* to $\mathtt{sta}(N_{3.0})$ as $E_{2.0}(0) \rightarrow E_{2.0}$ and $E_{2.4}(0) \rightarrow E_{2.4}$ are identical links and $E_{2.0}(1) \rightarrow E_{2.0}$ and $E_{2.4}(1) \rightarrow E_{2.4}$ are also identical links.

Using identical links, we can detect the resemblance among the cross links also.

Definition 3.10 *Two cross links* $E_{i.j_{00}}(\mathtt{r}_{00}) \rightarrow E_{i.j_{01}}(\mathtt{r}_{01}) \rightarrow \cdots \rightarrow E_{i.j_{0(l-1)}}(\mathtt{r}_{0(l-1)}) \rightarrow E_{i.j_{00}}$ *and* $E_{i.j_{10}}(\mathtt{r}_{10}) \rightarrow E_{i.j_{11}}(\mathtt{r}_{11}) \rightarrow \cdots \rightarrow E_{i.j_{1(l-1)}}(\mathtt{r}_{1(l-1)}) \rightarrow E_{i.j_{10}}$ *are said to be identical cross links if there exists some integer* m *between* 0 *and* $l-1$ *such that* $\mathtt{r}_{0q}$ *and* $\mathtt{r}_{1((q+m) \pmod l)}$ *are either same or equivalent RMTs for every* q $(0 \leq q \leq l-1)$*.*

Example 3.16 In Figure 3.11, in the reachability tree of CA $\langle 6, 102, 240, 65\rangle$, $E_{2.1}(6) \rightarrow E_{2.7}(4) \rightarrow E_{2.2}(2) \rightarrow E_{2.1}$ and $E_{2.3}(4) \rightarrow E_{2.6}(2) \rightarrow E_{2.5}(6) \rightarrow E_{2.3}$ are the identical cross links (in this case $m = 2$).

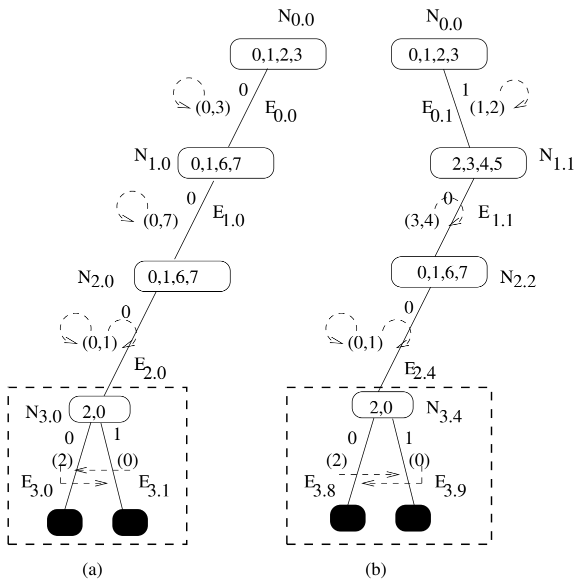

Figure 3.12: Isomorphic intra-linked subtrees for CA $\langle 6, 102, 240, 65\rangle$. Here, (a) and (b) represent $\texttt{sta}(N_{3.0})$ and $\texttt{sta}(N_{3.4})$ respectively.

Like intra-linked subtrees, we can get the resemblance among the inter-linked collections of subtrees also. For that, identical cross links play the instrumental role.

Definition 3.11 *An inter-linked collection of subtrees is* isomorphic *to another inter-linked collection of subtrees iff for every cross link at level $i-1$ in the later, there exists a unique identical cross link at level $i-1$ of the former one.*

Example 3.17 From Figure 3.11, we get $\texttt{str}(N_{3.1}, N_{3.7}, N_{3.2})$ is *isomorphic* to $\texttt{str}(N_{3.3}, N_{3.6}, N_{3.5})$.

Now, the objective of detecting isomorphism among the subtrees, to figure out the number of cycles of CA without exploring the entire reachability tree.

Lemma 3.4 *Let us consider two isomorphic intra-linked subtrees (resp. isomorphic inter-linked collection of subtrees). Then they generate same number of cycles having same lengths.*

Proof: Let us consider $\texttt{sta}(N_{i.j_i^1})$ and $\texttt{sta}(N_{i.j_i^2})$ to be isomorphic. Then, for every link from level i to $n-1$ in $\texttt{sta}(N_{i.j_i^1})$, there exits an identical link in $\texttt{sta}(N_{i.j_i^2})$ and vice versa. Therefore, any self link or cross link generated at level $n-1$ in $\texttt{sta}(N_{i.j_i^1})$, there exists an identical self link or cross link in $\texttt{sta}(N_{i.j_i^2})$ at level $n-1$. Thus $\texttt{sta}(N_{i.j_i^1})$ and $\texttt{sta}(N_{i.j_i^2})$ both generate same number cycles having same lengths.

Let us consider $\mathtt{str}(N_{i.j_i^{11}}, N_{i.j_i^{12}}, \cdots, N_{i.j_i^{1q}})$ and $\mathtt{str}(N_{i.j_i^{21}}, N_{i.j_i^{22}}, \cdots, N_{i.j_i^{2q}})$ $(0 \le j_i^{11}, j_i^{12}, \cdots, j_i^{1q}, j_i^{21}, j_i^{22}, \cdots, j_i^{2q} \le 2^i - 1)$ to be isomorphic. Hence, for every cross link from levels i to $n-1$ in $\mathtt{str}(N_{i.j_i^{11}}, N_{i.j_i^{12}}, \cdots, N_{i.j_i^{1q}})$, there exists an identical cross link in $\mathtt{str}(N_{i.j_i^{21}}, N_{i.j_i^{22}}, \cdots, N_{i.j_i^{2q}})$. Thus $\mathtt{str}(N_{i.j_i^{11}}, N_{i.j_i^{12}}, \cdots, N_{i.j_i^{1q}})$ and $\mathtt{str}(N_{i.j_i^{21}}, N_{i.j_i^{22}}, \cdots, N_{i.j_i^{2q}})$ both generate same number cycles having same lengths. This completes the proof. □

Corollary 3.9 : *Let the reachability tree of a reversible CA $\mathcal{R}$ contain v isomorphic intra-linked subtrees (resp. inter-linked collections of subtrees). Let $[\mu_1^1(l_1^1), \mu_2^1(l_2^1), \cdots, \mu_{m_1}^1(l_{m_1}^1)]$ be the partial cycle structure of an intra-linked subtree (resp. inter-linked collection of subtrees) out of those v intra-linked subtrees (resp. inter-linked collections of subtrees). Then, $\mathtt{CS}_{\mathcal{R}} \le [v.\mu_1^1(l_1^1), v.\mu_2^1(l_2^1), \cdots, v.\mu_{m_1}^1(l_{m_1}^1)]$.*

Proof: According to Lemma 3.4, every of v intra-linked subtrees (resp. inter-linked collections of subtrees) generates μ_1^1 cycles of length l_1^1, μ_2^1 cycles of length l_2^1, $\cdots$ and $\mu_{m_1}^1$ cycles of length $l_{m_1}^1$. Therefore, by considering all v isomorphic intra-linked subtrees (resp. inter-linked collections of subtrees), we get at least $v \cdot \mu_1^1$ cycles of length l_1^1, $v \cdot \mu_2^1$ cycles of length l_2^1 and in this way we also get $v \cdot \mu_{m_1}^1$ cycles of length $l_{m_1}^1$. Then $\mathtt{CS}_{\mathcal{R}} \le [v.\mu_1^1(l_1^1), v.\mu_2^1(l_2^1), \cdots, v.\mu_{m_1}^1(l_{m_1}^1)]$. The proof follows. □

In the reachability tree of an n-cell reversible CA, if all the nodes of a level are header nodes and the tree can be viewed as a collection of isomorphic intra-linked subtrees and isomorphic intra-linked collection of subtrees, then the cycle structure can be evaluated in reduced complexities. Let us verify it intuitively. To get the cycle structure of $\langle 6, 102, 240, 65 \rangle$, at level 3, we can find all nodes are header nodes. Moreover, we get $\mathtt{sta}(N_{3.0})$ and $\mathtt{sta}(N_{3.4})$ are isomorphic and $\mathtt{str}(N_{3.1}, N_{3.7}, N_{3.2})$, and $\mathtt{str}(N_{3.3}, N_{3.6}, N_{3.5})$ are also isomorphic. Therefore, we can evaluate the cycle structure of $\langle 6, 102, 240, 65 \rangle$ in the given manner.

$$\mathtt{CS}_{\mathcal{R}} = \mathtt{PCS}_{\mathtt{sta}(N_{3.0})} + \mathtt{PCS}_{\mathtt{sta}(N_{3.4})} + \mathtt{PCS}_{\mathtt{str}(N_{3.1}, N_{3.7}, N_{3.2})} + \mathtt{PCS}_{\mathtt{str}(N_{3.3}, N_{3.6}, N_{3.5})}$$

Here, $\mathtt{PCS}_{\mathtt{sta}(N_{3.0})} = \mathtt{PCS}_{\mathtt{sta}(N_{3.4})} = [1(2)]$ and $\mathtt{PCS}_{\mathtt{str}(N_{3.1}, N_{3.7}, N_{3.2})} = \mathtt{PCS}_{\mathtt{str}(N_{3.3}, N_{3.6}, N_{3.5})} = [1(6)]$. Therefore, $\mathtt{CS}_{\mathcal{R}} = [2(2), 2(6)]$ (using Corollary 3.9). Therefore, by processing $\mathtt{PCS}_{\mathtt{sta}(N_{3.0})}$ and $\mathtt{PCS}_{\mathtt{str}(N_{3.1}, N_{3.7}, N_{3.2})}$, we can evaluate the partial cycle structures of $\mathtt{PCS}_{\mathtt{sta}(N_{3.4})}$ and $\mathtt{PCS}_{\mathtt{str}(N_{3.3}, N_{3.6}, N_{3.5})}$ respectively also. In this example, though only one intra-linked subtree and one inter-linked collection of subtrees are traversed, but that subtrees grow exponentially. Next, we target to evaluate the partial cycle structure of an inter-linked collection of subtrees without traversing all the components of that collection.

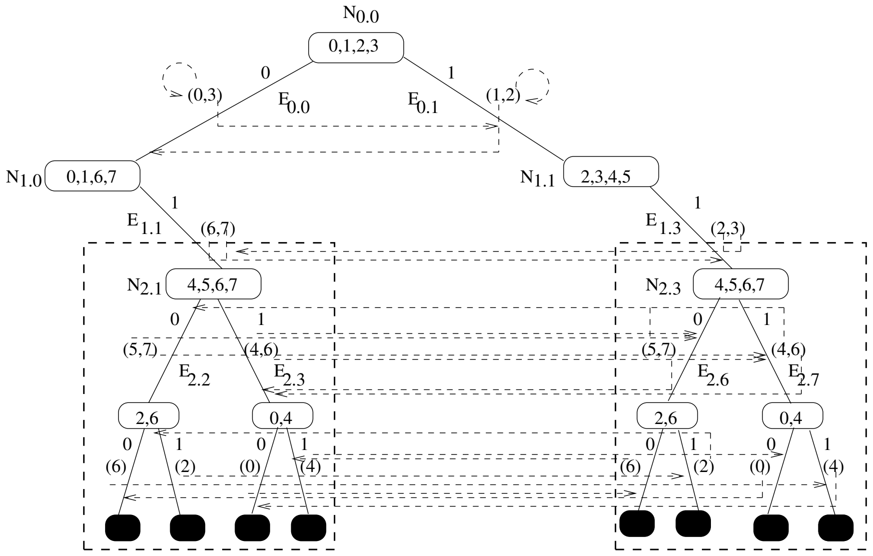

Figure 3.13: For $\texttt{str}(N_{2.1}, N_{2.3})$ of the reachability tree of CA $\langle 6, 204, 90, 20\rangle$, $\texttt{cla}(N_{2.1}, 2)$ is used as the coalesced subtree.

3.5.2 Coalesced Subtree and Blocks in CA

Let us consider $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$. Then at level $n-1$, a total of $2^{n-i-1}\cdot q$ links need to be processed for finding the cycles. Let $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ generate a cycle of length l. According to Lemma 3.3, every cycle generated in $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ has a length of multiple of q. Therefore, l is multiple of q. Suppose $N_{i.j_i^1} = N_{i.j_i^2} = \cdots = N_{i.j_i^q}$. Therefore, at level i, there exists a permutation π such that edge k of $\texttt{st}(N_{i.j_1})$ is linked to edge $\pi(k)$ of $\texttt{st}(N_{i.j_2})$, edge k of $\texttt{st}(N_{i.j_2})$ is linked to edge $\pi(k)$ of $\texttt{st}(N_{i.j_3})$ and so. In general, edge k of $\texttt{st}(N_{i.j_i^p})$ is linked to the edge $\pi(k)$ of $\texttt{st}(N_{i.j_i^{p \pmod q)+1}})$. So, at level i, for every link which connects two consecutive components of $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$, there exist $q-1$ identical links. Let for the cycle of length l, if a particular edge k at level $n-1$ is selected in one component, the same edge k is to be selected for every component. Therefore, if edge k is selected in every component after l' times therefore, l is also multiple of l'. The identical links among these q subtrees in $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ instigate us to introduce the notion of *coalesced* subtree.

In $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$, as $N_{i.j_i^1} = N_{i.j_i^2} = \cdots = N_{i.j_i^q}$, we consider only one subtree out of q and we process only that subtree. Such subtree is called as *coalesced*. We create a *coalesced* subtree as follows: (i) the nodes and edges of this coalesced subtree are same as those of $\texttt{st}(N_{i.j_1^1})$ (ii) edge k of this coalesced subtree is linked to edge $\pi(k)$ of itself. It is denoted by $\texttt{cla}(N_{i.j_1^1}, q)$. Here, $\texttt{st}(N_{i.j_1^1})$ refers the first subtree in $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ and q is the number of subtrees involved in $\mathtt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$. Let us consider the reachability tree of CA $\langle 6, 204, 90, 20\rangle$ (see Figure 3.7). The subtrees at header nodes $\mathtt{st}(N_{2.1})$ and $\mathtt{st}(N_{2.3})$ are used to form an inter-linked collection of subtrees which can be denoted by $\mathtt{str}(N_{2.1}, N_{2.3})$. As $N_{2.1} = N_{2.3}$, the coalesced subtree is given by $\mathtt{cla}(N_{2.1}, 2)$ (depicted in Figure 3.13). In Figure 3.13, $E_{2.2}$ is linked to $E_{2.6}$ and $E_{2.7}$ respectively for RMTs 5 and 7. Now, with respect to $\mathtt{st}(N_{2.1})$, $E_{2.2}$ is the first edge of $\mathtt{st}(N_{2.1})$ at level 2. Similarly, $E_{2.6}$ and $E_{2.7}$ are the first and second edges respectively of $\mathtt{st}(N_{2.3})$ at level 2. Therefore, as $\mathtt{st}(N_{2.1})$ is considered as the coalesced subtree, therefore, $E_{2.2}$ is to be linked to itself and $E_{2.3}$ for the respective RMTs 5 and 7. Thus by considering $\mathtt{st}(N_{2.1})$ only, we can figure out the partial cycle structure of $\mathtt{str}(N_{2.1}, N_{2.3})$. Let us intuitively explain it. Since we get $E_{2.2}(5) \rightarrow E_{2.2}$, $E_{2.2}(7) \rightarrow E_{2.3}$, $E_{2.3}(4) \rightarrow E_{2.2}$ and $E_{2.3}(6) \rightarrow E_{2.3}$ by considering $\mathtt{cla}(N_{2.1}, 2)$, the corresponding derived links at level 3 are considered for finding the number of cycles along with their lengths for $\mathtt{str}(N_{2.1}, N_{2.3})$. The derived links are $E_{3.4}(6) \rightarrow E_{3.7}$, $E_{3.5}(2) \rightarrow E_{3.5}$, $E_{3.6}(0) \rightarrow E_{3.4}$ and $E_{3.7}(4) \rightarrow E_{3.6}$. Now, every component of $\mathtt{str}(N_{2.1}, N_{2.3})$ has four edges at level 3 of the reachability tree. Now, $E_{3.4}(6) \rightarrow E_{3.7}$ refers that the *first* edge of the current component is linked to the *fourth* edge of the next component, $E_{3.6}(0) \rightarrow E_{3.4}$ represents that the *third* edge of the current component is linked to the *first* edge of the next component and using $E_{3.7}(4) \rightarrow E_{3.6}$, we get that *fourth* edge of the current component is linked to the *third* edge of the next component. Therefore, using $E_{3.4}(6) \rightarrow E_{3.7}(4) \rightarrow E_{3.6}(0) \rightarrow E_{3.4}$, we can find the connection among the edges of $\mathtt{st}(N_{2.1})$ and $\mathtt{st}(N_{2.3})$ as follows: the first edge of $\mathtt{st}(N_{2.1})$ is linked to the fourth edge of $\mathtt{st}(N_{2.3})$, the fourth edge of $\mathtt{st}(N_{2.3})$ is linked to the third edge of $\mathtt{st}(N_{2.1})$, the third edge of $\mathtt{st}(N_{2.1})$ is linked to the first edge of $\mathtt{st}(N_{2.3})$ and the same sequence repeats from $\mathtt{st}(N_{2.3})$ as started with $\mathtt{st}(N_{2.1})$. Thus we get a cross link of length *six*. Therefore, any cycle length of $\mathtt{str}(N_{2.1}, N_{2.3})$ can be computed using the length of some cross link of $\mathtt{cla}(N_{2.1}, 2)$.

Next, we report the lemmas, corollaries based on coalesced subtree which may help to reduce the computation cost of evaluating the cycle structures of CAs.

Lemma 3.5 *Let l be the length of a cycle in $\mathtt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ where $N_{i.j_i^1} = N_{i.j_i^2} = \cdots = N_{i.j_i^q}$. Let l' be the length of a cross link generated in $\mathtt{cla}(N_{i.j_i^1}, q)$. Therefore, $l = \mathtt{lcm}(l', q)$.*

Proof: If $\mathtt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ generates a cycle of length l at level $n-1$, according to Lemma 3.3, that l is multiple of q. Let $\mathtt{cla}(N_{i.j_i^1}, q)$ generate a cycle of length l'. Therefore, edge k is selected from every subtree after l' time. Therefore, edge k of $\mathtt{st}(N_{i.j_i^1})$ is selected again after the multiple of l' times. Thus l is multiple of q as well as l'. Therefore, $l = \mathtt{lcm}(l', q)$. Hence the proof follows. □

Corollary 3.10 : *Let l be the length of a cycle in $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ where $N_{i.j_i^1} = N_{i.j_i^2} = \cdots = N_{i.j_i^q}$. Let l' be the length of a cross link generated in $\texttt{cla}(N_{i.j_i^1}, q)$. Therefore, $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ generates $\texttt{gcd}(l', q)$ number of cycles of length l.*

Proof: As $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ generates a cycle of length l at level $n-1$, according to Lemma 3.3, that l is multiple of q. Let $\texttt{cla}(N_{i.j_i^1}, q)$ generate a cross link of length l'. Therefore, $l = \texttt{lcm}(l', q)$ (according to Lemma 3.5). Thus the total number of links at level $n-1$ is $l' \cdot q$ for detecting the cross link(s) of length l; therefore, number of cross links of length l is given by $\frac{l' \cdot q}{\texttt{lcm}(l', q)} = \texttt{gcd}(l', q)$. This completes the proof. $\square$

Corollary 3.11 *Let us consider $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ where $N_{i.j_i^1} = N_{i.j_i^2} = \cdots = N_{i.j_i^q}$. Let $\texttt{cla}(N_{i.j_i^1}, q)$ generate μ_1^q cycles of length l_1^q, μ_2^q cycles of length l_2^q and so on. Then $\texttt{PCS}_{\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})} = [\mu_1^q \cdot \texttt{gcd}(l_1^q, q)(\texttt{lcm}(l_1^q, q)), \mu_2^q \cdot \texttt{gcd}(l_2^q, q)(\texttt{lcm}(l_2^q, q)), \cdots]$.*

Proof: For every cycle of length l_1^q, $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ generates $\texttt{gcd}(l_1^q, q)$ cycles (using Corollary 3.10) of length $\texttt{lcm}(l_1^q, q)$ (according to Lemma 3.5). As for every cycle of length l_1^q, $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ generates $\texttt{gcd}(l_1^q, q)$ cycles, then a total of $\mu_1^q \cdot \texttt{gcd}(l_1^q, q)$ cycles of length $\texttt{lcm}(l_1^q, q)$ is generated. In analogous manner, $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ also generates $\mu_2^q \cdot \texttt{gcd}(l_2^q, q)$ cycles of length $\texttt{lcm}(l_2^q, q)$. Then, $\texttt{PCS}_{\texttt{str}(N_{i.j_1}, N_{i.j_2}, \cdots, N_{i.j_q})} = [\mu_1^q \cdot \texttt{gcd}(l_1^q, q)(\texttt{lcm}(l_1^q, q)), \mu_2^q \cdot \texttt{gcd}(l_2^q, q)(\texttt{lcm}(l_2^q, q)), \cdots]$. Hence the proof follows. $\square$

By the above mentioned lemma and corollaries, the partial cycle structure can be evaluated without exploring the entire subtree, only one is explored upto level $n-1$. These q nodes have same contents. If the nodes have not the same set of RMTs, can we reduce the computation cost for finding the CS of an reversible CA? Let us figure out the answer. Let the header nodes of $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ may have different contents. Let also consider, edge k of $\texttt{st}(N_{i.j_i^p})$ is linked to the edge $\pi(k)$ of $\texttt{st}(N_{i.j_i^p \pmod q)+1})$. Now at level i, the subtrees can produce identical links if for every RMT $\texttt{r}$ belongs to $l_{i.2j_i^p}$ (resp. $l_{i.2j_i^p+1}$) there exists the same or equivalent RMT in $l_{i.2j_i^p \pmod q)+1}$ ($l_{i.2j_i^p \pmod q)+1}+1}$). Then for a link at level i, there exist $q-1$ identical links in $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ and it holds for level i to level $n-1$. Thus $\mathcal{R}_i$ should be designed in such a manner that the sibling RMTs have same values, i.e. $\mathcal{R}_i[0] = \mathcal{R}_i[4]$, $\mathcal{R}_i[1] = \mathcal{R}_i[5]$, $\mathcal{R}_i[2] = \mathcal{R}_i[6]$ and $\mathcal{R}_i[3] = \mathcal{R}_i[7]$. Therefore, such $\mathcal{R}_i$ should be selected from $\{51, 204, 85, 170, 102, 153\}$. Then by considering coalesced subtree, we can figure out the partial cycle structure of $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$.

Corollary 3.12 *Let* $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ *be an inter-linked collection of* q *subtrees of CA* $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_i, \cdots, \mathcal{R}_{n-1}\rangle$. *Let* $\mathcal{R}_i$ *be selected from* $\{51, 204, 85, 170, 102, 153\}$. *Let also consider* $\texttt{cla}(N_{i.j_i^1}, q)$ *generate* μ_1^q *cycles of length* l_1^q, μ_2^q *cycles of length* l_2^q *and so on. Then,* $\texttt{PCS}_{\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})} = [\mu_1^q \cdot \texttt{gcd}(l_1^q, q)(\texttt{lcm}(l_1^q, q)), \mu_2^q \cdot \texttt{gcd}(l_2^q, q)(\texttt{lcm}(l_2^q, q)), \cdots]$.

Proof: Let $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ be an inter-linked collection of CA $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_i, \cdots, \mathcal{R}_{n-1}\rangle$. Let also consider $\mathcal{R}_i \in \{51, 204, 85, 170, 102, 153\}$. Therefore, for every RMT $\texttt{r}$ belongs to $l_{i.2j_i^p}$ (resp. $l_{i.2j_i^p+1}$) there exists the same or equivalent RMT in $l_{i.2j_i^{p \pmod q)+1}}$ ($l_{i.2j_i^{p \pmod q)+1}+1}$). Thus edge k of every component $\texttt{st}(N_{i.j_i^p})$ of $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$ is linked to the edge $\pi(k)$ of $\texttt{st}(N_{i.j_i^{p \pmod q)+1}})$. Therefore, $\texttt{cla}(N_{i.j_i^1}, q)$ is used as the coalesced subtree of $\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})$. Let $\texttt{cla}(N_{i.j_i^1}, q)$ generate μ_1^q cycles of length l_1^q, μ_2^q cycles of length l_2^q and so on. Then $\texttt{PCS}_{\texttt{str}(N_{i.j_i^1}, N_{i.j_i^2}, \cdots, N_{i.j_i^q})} = [\mu_1^q \cdot \texttt{gcd}(l_1^q, q)(\texttt{lcm}(l_1^q, q)), \mu_2^q \cdot \texttt{gcd}(l_2^q, q)(\texttt{lcm}(l_2^q, q)), \cdots]$ (according to Corollary 3.11). Hence the proof follows. □

Corollary 3.13 *Let the reachability tree of a CA* $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_i, \cdots, \mathcal{R}_{n-1}\rangle$ *contain an inter-linked collection of* q_1 *subtrees, another inter-linked collection of* q_2 *subtrees and so on. If* $\mathcal{R}_i \in \{51, 204, 85, 170, 102, 153\}$, *then every inter-linked collection of subtrees considers one unique coalesced subtree.*

Proof: Let us consider $\texttt{sta}(N_{i.j_{11}}, N_{i.j_{21}}, \cdots, N_{i.j_{q1}})$, $\texttt{sta}(N_{i.j_{12}}, N_{i.j_{22}}, \cdots, N_{i.j_{q2}})$ and so on. Let also consider $\mathcal{R}_i \in \{51, 204, 85, 170, 102, 153\}$. Therefore, for every link in $\texttt{st}(N_{i.j_{11}})$, there exist $q-1$ identical links in other components of $\texttt{sta}(N_{i.j_{11}}, N_{i.j_{21}}, \cdots, N_{i.j_{q1}})$; similarly, for every link in $\texttt{st}(N_{i.j_{12}})$, there exist $q-1$ identical links in other components of $\texttt{sta}(N_{i.j_{12}}, N_{i.j_{22}}, \cdots, N_{i.j_{q2}})$ and so on. Therefore, $\texttt{st}(N_{i.j_{11}})$, $\texttt{sta}(N_{i.j_{12}})$ can be considered as coalesced subtrees of respectively $\texttt{sta}(N_{i.j_{11}}, N_{i.j_{21}}, \cdots, N_{i.j_{q1}})$, $\texttt{st}(N_{i.j_{12}})$ and $\texttt{sta}(N_{i.j_{12}}, N_{i.j_{22}}, \cdots, N_{i.j_{q2}})$. As $\mathcal{R}_i \in \{51, 204, 85, 170, 102, 153\}$, therefore, $\texttt{cla}(N_{i.j_{11}})$ and $\texttt{cla}(N_{i.j_{12}})$ produce same collection of cycles. Therefore, only $\texttt{cla}(N_{i.j_{11}})$ is considered. This completes the proof. □

Using all the lemmas and corollaries on coalesced subtree, we explore only one component of an inter-linked collection of subtrees upto level $n-1$. Even for one component subtree also, the subtree has to be explored exponentially. To reduce the computational cost, we aim to find whether a subtree is *contained* in another subtree. Let $\texttt{st}(N_{i.j_i})$ contain $\texttt{st}(N_{i'.j_i'})$. Therefore, $\texttt{st}(N_{i.j_i})$ is explored upto level i' and it is needed to be checked whether isomorphism occurs or not. If isomorphism is detected, then computation cost will be reduced. Therefore, if we traverse level wise, we may get more than one levels where all nodes of those

Algorithm 2 Algorithm to find the CS of an n-cell reversible CA

```
    Input: $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$
    Output: $CS$
 1: $CS \leftarrow$ PCS_intra($\mathcal{R}_0, ...\mathcal{R}_{n-1}, N_{0.0}$).
 2: Change $CS$ in appropriate format.
 3: $CS \leftarrow \varnothing$.
 4: while ($CS$ is non empty) do
 5:    Let the first element of $CS$ be $(\mu_1, \ell_1)$.
 6:    $val = 0$.
 7:    for each element $e$ in $CS$ of form $(\mu', \ell_1)$ in CS do
 8:       $val = val + \mu'$
 9:       Remove element $e$ from $CS$
10:    end for
11:    $CS \leftarrow CS \cup val(\ell_1)$.
12: end while
       Return $CS$.
```

levels are header nodes. In this way we can get multiple levels in the reachability tree, where we get all nodes are header nodes. The reachability tree can already be partitioned into some disjoint subtrees or collections of subtrees. Next, we can see that a reachability tree is a collection of *blocks*. When all nodes are header nodes in some level in the reachability tree, the corresponding CA rule is from $\{15, 240, 51, 204, 60, 195\}$. Thus a CA can be a collection of *blocks*.

Definition 3.12 *Let* $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{i_1}, \cdots, \mathcal{R}_{i_2}, \cdots, \mathcal{R}_{i_k}, \cdots, \mathcal{R}_{n-1} \rangle$ *where* $i_1, i_2, \cdots, i_k$ *positions use rules from* $\{15, 240, 51, 204, 60, 195\}$. *Now, this* $\mathcal{R}$ *represents* $k+1$ *blocks where each block represents a sequence of rules where the last rule of that block is from* $\{15, 240, 51, 204, 60, 195\}$ *(except the last block). The number of blocks in a CA is denoted as* `b`. *The length of a block which can be called as block length is determined by the number of rules involved in that block. A block can be denoted by* b_i *where* b_i *is the* i^{th} *block (where* $i \geq 0$*).*

Let $\mathcal{R} = \langle 6, 178, 90, 60, 54, 86, 15, 153, 39, 101, 80 \rangle$ be an 11-bit CA which also represents *three* blocks b_0, b_1 and b_2. Here, b_0 refers a sequence of rules $\langle 6, 178, 90, 60 \rangle$. Similarly, b_1 and b_2 represent $\langle 54, 86, 15 \rangle$ and $\langle 153, 39, 101, 80 \rangle$ respectively. The lengths of blocks b_0, b_1 and b_2 are 4, 3 and 4 respectively. To get blocks in CA, there exists at least one rule from $\{15, 240, 51, 204, 60, 195\}$. To generate an arbitrary n-cell reversible CA, we select the rules from Table 2.7b, Table 2.7a and Table 2.7c respectively for cell 0, cell i $(1 \leq i \leq n-2)$ and cell $n-1$ respectively (reported in Section 2.4.2 of Chapter 2 on page 38). The computational cost for finding the cycles or cycle structure in CA depends on the number of blocks. If the number blocks is more, it is expected that block

Algorithm 3 PCS_intra

```
Input: $\mathcal{R} = \langle \mathcal{R}_q, \mathcal{R}_{q+1}, \cdots, \mathcal{R}_{n-1} \rangle$, $N_{q.h}$
Output: $CS$.
1: Initialize $CS$ as an empty list.
2: for (every level $i = q$ to $n-2$) do
3:     Create edge $E_{i.j}$ of level $i$ (where $2^{i-q+1} \cdot h \leq j < 2^{i-q+2} \cdot h$) and links among
       edges of level $i$ (and remove all edges and links of previous levels if applicable).
4:     if ($\mathcal{R}_i \in \mathcal{H}_2$) then
5:         Find the collection of intra-linked subtrees $AL$ and the inter-linked collections
           of subtrees $GL$ using Algorithm 5.
6:         $L_1 \leftarrow Isomorphic(AL)$ using the procedure in Algorithm 6.
7:         $L_2 \leftarrow Isomorphic(GL)$ using the procedure in Algorithm 6.
8:     end if
9:     for ($(j, p) \in L_1$) do
10:        temp ← PCS_intra ($\langle \mathcal{R}_i, \cdots, \mathcal{R}_{n-1} \rangle$, $N_{i.j}$).
11:        for $(\mu, \ell) \in$ temp do
12:            Append $(p \cdot \mu, \ell)$ in $CS$.
13:        end for
14:    end for
15:    for ($(\{j_1, j_2, \cdots, j_k\}, p) \in L_2$) do
16:        temp ← PCS_inter ($\langle \mathcal{R}_i, \cdots, \mathcal{R}_{n-1} \rangle$, $\{N_{i.j_1}, N_{i.j_2}, \cdots, N_{i.j_k}\}$),
           $\{E_{i-1.j_1}, E_{i-1.j_2}, \cdots, E_{i-1.j_k}\}$).
17:        for $(\mu, \ell) \in$ temp do
18:            Append $(p \cdot \mu, \ell)$ in $CS$.
19:        end for
20:        Discard all edge indices $e$ in $AL$ and all edge indices $e \in CL$ (where $CL$ is in
           $GL$) for further processing in subsequent levels.
21:    end for
22: end for
23: Find out the cross links and self links at level $n-1$ and let temp1 denote the
    corresponding collection of cyclic components.
24: Append every element of temp1 in $CS$.
25: Return $CS$.
```

size should be less. The presence of blocks in CA shows that a subtree can be contained in another subtree. It is already established that by partial cycle structures we can evaluate the CS of CA. Now, to evaluate the partial cycle structure of an inter-linked or an intra-linked collection of subtrees, we may not process the subtree exponentially if there is the existence of blocks. Let us discuss the procedure formally.

Algorithm 2 computes the cycle structure of an n-cell reversible CA. It takes a reversible CA $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ as an input, while the output CS refers to the cycle structure of $\mathcal{R}$. A routine is called at line 1 of Algorithm 2 to

Algorithm 4 PCS_inter

```
Input: $\mathcal{R} = \langle \mathcal{R}_q, \mathcal{R}_{q+1}, \cdots, \mathcal{R}_{n-1}\rangle$, $\{N_{q.h_1}, N_{q.h_2}, \cdots, N_{q.h_k}\}$, $\{E_{q-1.h_1}, E_{q-1.h_2}, \cdots, E_{q-1.h_k}\}$) where $E_{q-1.h_1}$ is only linked to $E_{q-1.h_2}$, $E_{q-1.h_2}$ is only linked to $E_{q-1.h_3}$, $\cdots$, $E_{q-1.h_k}$ is only linked to $E_{q-1.h_1}$.
Output: $CS$.
 1: Initialize $\mathcal{B} = \{51, 204, 85, 170, 102, 153\}$.
 2: Initialize $CS$ as an empty list.
 3: for (every level $i = q$ to $n-2$) do
 4:     Create edge $E_{i.j_v}$ (where $1 \le v \le k$) of level $i$ (where $2^{i-q+1} \cdot h_v \le j < 2^{i-q+2} \cdot h_v$) and links among edges of level $i$ (and remove all edges and links of previous levels if applicable).
 5:     if ($\mathcal{R}_i \in \mathcal{H}_2$) then
 6:         Find the inter-linked collections of subtrees $GL$ using Algorithm 5.
 7:         $L_2 \leftarrow Isomorphic(GL)$ using the procedure in Algorithm 6.
 8:     end if
 9:     for $((\{j_1, j_2, \cdots, j_w\}, p) \in L_2)$ do
10:         if (All $j_v$s are odd) or (all $j_v$s are even) or ($\mathcal{R}_{i+1} \in \mathcal{B}$) then
11:             temp $\leftarrow$ PCS_intra ($\langle \mathcal{R}_i, \cdots, \mathcal{R}_{n-1}\rangle$, $N_{i.j_1}$).
12:             for ($\mu$, $\ell$) $\in$ temp do
13:                 Append $(p \cdot \mu \cdot \texttt{gcd}(\ell, w), \texttt{lcm}(\ell, w))$ in $CS$.
14:             end for
15:         else
16:             temp $\leftarrow$ PCS_inter ($\langle \mathcal{R}_i, \cdots, \mathcal{R}_{n-1}\rangle$, $\{N_{i.j_1}, N_{i.j_2}, \cdots, N_{i.j_w}\}$), $\{E_{i-1.j_1}, E_{i-1.j_2}, \cdots, E_{i-1.j_w}\}$).
17:             for ($\mu$, $\ell$) $\in$ temp do
18:                 Append $(p \cdot \mu, \ell)$ in $CS$.
19:             end for
20:         end if
21:     end for
22:     Discard all edge indices $e \in CL$ (where $CL$ is in $GL$) for further processing in subsequent levels.
23: end for
24: Find out the cross links and self links at level $n-1$ and let temp1 denote the corresponding collection of cyclic components.
25: $CS \leftarrow CS \cup$ temp1.
26: Return $CS$.
```

evaluate the CS of the CA. At level i, if the $\mathcal{R}_i$ is selected from $\mathcal{H}_2$, then using Algorithm 5, we need to find whether an intra-linked subtree or an inter-linked collection of subtrees is generated. In Algorithm 5, we get two lists AL and GL. The AL stores those edge indices which form self links only (see lines 4 to 7). The GL is a collection of cross links where CL, a cross link is formed using the parent edges of the header nodes generated at level $i+1$. Lines 8 to 21 detect

Algorithm 5 Algorithm to detect intra-linked subtree and inter-linked collection of subtrees

```
Input: $\mathcal{J}'$ and $E_{i.j}$s such that $j \in \mathcal{J}'$.
Output: $AL$ and $GL$.
1: Set $p_3 \leftarrow 0$, $p_4 \leftarrow 0$;
2: Let $AL$ and $GL$ denote the lists of sets initialized to empty list.
3: while ($\mathcal{J}' \neq \varnothing$) do
4:    Choose the first element $j$ of $\mathcal{J}'$.
5:    if ($E_{i.j}$ is only linked to itself)    (Detection of intra-linked subtrees at level $i+1$) then
6:       Append set $\{j\}$ to $AL$; $\mathcal{J}' \leftarrow \mathcal{J}' \setminus \{j\}$.
7:    end if
8:    if ($E_{i.j}$ forms either forward link or backward link only with $E_{i.k}$)    (Detection of inter-linked collection of subtrees at level $i+1$) then
9:       Set $org \leftarrow j$, $CL \leftarrow \{j\}$, $src \leftarrow k$;
10:      while ($E_{i.src}$ is only linked to $E_{i.dest}$) do
11:         Set $CL \leftarrow CL \cup \{src\}$, $src \leftarrow dest$;
12:         if ($org = src$) then
13:            Set $flag \leftarrow 1$;
14:            break;
15:         end if
16:      end while
17:      $\mathcal{J}' \leftarrow (\mathcal{J}' \setminus CL) \setminus \{src\}$;
18:      if ($flag = 1$) then
19:         Append set $CL$ to $GL$.
20:      end if
21:   end if
22: end while
23: Return $AL$ and $GL$.
```

the inter-linked collection of subtrees at level $i+1$. Algorithm 6 is called if AL or GL is non empty. If k intra-linked subtrees or inter-linked collections of subtrees are isomorphic then only the js of one such set and k is appended in list L_1 or L_2 respectively. If L_1 is non empty list, then the routine for finding the partial cycle structure of an intra-linked subtree (`PCS`_intra ($\langle \mathcal{R}_i, \cdots, \mathcal{R}_{n-1} \rangle$, $N_{i.j}$)) is called (lines 9 to 14 of Algorithm 3). Similarly, if L_2 is non empty list, then the routine for finding the partial cycle structure of an inter-linked collection of subtrees (`PCS`_inter ($\langle \mathcal{R}_i, \cdots, \mathcal{R}_{n-1} \rangle$, $\{N_{i.j_1}, N_{i.j_2}, \cdots, N_{i.j_k}\}$), $\{E_{i-1.j_1}, E_{i-1.j_2}, \cdots, E_{i-1.j_k}\}$)) is called (lines 15 to 21 of Algorithm 3). Both these routines are recursive. Using `PCS`_intra ($\langle \mathcal{R}_i, \cdots, \mathcal{R}_{n-1} \rangle$, $N_{i.j}$), the partial cycle structure of a subtree is evaluated (lines 11 to 13) by following Corollary 3.9. In the routine for evaluating the partial cycle structure of an inter-linked collection of subtrees, the inter-linked collection of subtrees can be coalesced also (lines 9 to 13). Here

Algorithm 6 Isomorphic: Detect isomorphism among intra-linked subtrees or the inter-linked collection of subtrees

```
Input: $GL$ - list of sets where each set is either singleton or represents a collection of parent edges which form cross link.
Output: A list $L$ constructed as follows: for every $\ell_1, \ell_2, \cdots \ell_z \in GL$ that are isomorphic to each other, add $(\ell_1, z)$ to $L$.
1: while $GL$ is non-empty do
2:     Suppose $\ell$ be the first set in $GL$. Let $m = |\ell|$.
3:     Find the collection $I_\ell$ of sets in $GL$ that are isomorphic to $\ell$ using Steps 3-8.
4:     $I_\ell = [\ell]$.
5:     for $J \in GL$ do
6:         if there exists an integer $k$ such that for all $i$ $(0 \leq i, k \leq m)$, element $i$ of $\ell$
       and element $(i + k) \pmod{m}$ of $J$ are both odd or both even then
7:             Append $J$ to $I_\ell$.
8:         end if
9:     end for
10:    Remove $I_\ell$ from $GL$.
11:    Append $(\ell, |I_\ell|)$ to $L$.
12: end while
13: Return $L$.
```

Table 3.3: Performance analysis

n	$\mu_1 + \mu_2 + \cdots + \mu_m$	t (in secs)
10	74	0
40	57347007334	0.04
64	281547222833763994	0.13
100	441952910909022165458233306	0.4
160	60926000609516717046230280747285239981777105878	1.79
480	13007285626316635515341352899890624777023186191339 896175190273145105715557841569008383460876276454987 983261221300586092873103323199703456221095424	43.87
10	60	0
15	1066	0.09
20	16340	3.66
25	236700	273.75

the edges used in AL and GL are removed for further processing.

The worst case complexity of Algorithm 2 is exponential if no isomorphism exists among the intra-linked subtrees or inter-linked collection of subtrees. If Algorithm 3 and Algorithm 4 are called frequently for a CA, the CS of such CA can be evaluated in reduced time.

Thus by introducing the concepts of blocks, isomorphism among the subtrees and the arrangements of rules within the blocks, we obtain some results after doing an extensive experimentation. Though our algorithm can evaluate the CS of CA, here, in experimentation, we report only the total number of cycles $\sum_{i=1}^{m} \mu_i$ and the execution time to compute the total number of cycles of the given CA. All the experiments are performed on an Intel(R) i3 2.40GHz system. Here, Table 3.3 is used to represent effectiveness of Algorithm 2. The first column of the table refers the CA size n. For a given n, we take a reversible CA and find the total number of cycles. The second column refers the total number of cycles and the third column shows the execution time denoted by t (in seconds) to figure out the total number of cycles. The following cases show how the design of an n-cell reversible CA using blocks are effective to get the cycle structure in faster execution.

- Case I: If the first rule of every block (excluding the first block) in an n-cell CA is selected from $\{51, 204, 85, 170, 102, 153\}$ [rows $2-7$ of Table 3.3] and the maximum block size is restricted to 4.

- Case II: If the first rule of every block (excluding the first block) in an n-cell CA is selected from other than $\{51, 204, 85, 170, 102, 153\}$ [rows $8-11$ of Table 3.3] and the maximum block size is restricted to 3.

3.6 Summary

This chapter has presented detection of cycles as a sub problem of reachability problem. An effective strategy has been devised to figure out the cycle structure of an n-cell CA. In case of irreversible CA, our algorithm performs well but it has exponential complexity for reversible CAs. To deal with this issue, we have estimated the CS of a CA at early level in the reachability tree. Next, we introduced the concept of partial cycle structure of subtree. To evaluate the CS of reversible CA, we use the partial cycle structures of inter-linked subtrees and the inter-linked collections of subtrees. Moreover, we have introduced the concept of isomorphism among the subtrees in the reachability tree. Using isomorphism among the subtrees, only one subtree's partial cycle structure is evaluated to find the partial cycle structures of the remaining subtrees. Obviously, the CS of a reversible CA can be evaluated efficiently if more number of isomorphic subtrees exist in the reachability tree.

CHAPTER 4

Cyclic Spaces of Reversible Cellular Automata

The problem of evaluating the cycle structures (CSs) of reversible cellular automata has been addressed in the previous chapter and Algorithm 2 (reported in Section 3.5 on page 81) has been used to find the CS of a reversible CA. This chapter discovers some fascinating properties of the cycle structures of reversible cellular automata to study another interesting problem which can be mentioned as follows: Can we *decompose* the cycle structure of an arbitrary reversible CA of finite size? Accordingly, we introduce the concepts of *reducible* and *irreducible* reversible cellular automata.

4.1 Introduction

Traditionally, a reversible cellular automaton is defined over infinite lattice size. Nevertheless, in finite reversible cellular automaton, each configuration reappears after a period of time. Therefore, the configuration space of any reversible CA represents the *cyclic space* of the CA (Definition 2.15 of Chapter 2 on page 32) and the cycle structure of a CA is evaluated from the cyclic space of the CA. However, often it is found that the cycle structure of a CA of a particular size is related to the CS of another CA of smaller size. This brings us to the notion of *decomposition* of a cycle structure. Intuitively, the cycle structure of a large sized CA is decomposed into the cycle structure of a small sized CA if the length of any cycle of the former CA is always a multiple of the length of some cycle of the small sized CA.

Recall the mechanism introduced in Chapter 3 for finding the CS of any reversible CA. Consider the reachability tree of an n-cell reversible CA $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$. Suppose an inter-linked collection of q subtrees is found at the level n_1 ($n_1 < n$) of the reachability tree. This implies that some cycles generated by $\mathcal{R}$ are always of length that is multiple of q, and q is the length of the elementary cross links (see Definition 3.2 of Chapter 3 on page 46) at level $n_1 - 1$ which involve the parent edges of those q header nodes (according to Lemma 3.3 of Chapter 3 on page 71). If every edge of level $n_1 - 1$ contains sibling RMTs in the corresponding edge label, then such edge either forms one self link or is involved in one cross link where the effective RMT is *even*. Thus length of any cycle of $\mathcal{R}$ is found to be a multiple of the length of some cycle of an n_1-cell CA $\mathcal{R}^{n_1} = \langle \mathcal{R}^1_0, \mathcal{R}^1_1, \cdots, \mathcal{R}^1_{n_1-1} \rangle$, where $\mathcal{R}^1_i = \mathcal{R}_i$, for $i = 0, 1, \cdots, n_1 - 2$, and $\mathcal{R}^1_{n_1-1}[\mathtt{r}] = \mathcal{R}_{n_1-1}[\mathtt{r}]$ is defined only for every even RMT $\mathtt{r}$ (and all odd RMTs are ignored). In other words, the cycle structure of $\mathcal{R}$ is decomposed into the cycle structure of $\mathcal{R}^{n_1}$.

Generating a CA from a given cyclic/acyclic space for linear cellular automata was first attempted in [359]. However, the scheme reported in [359] fails to generate CAs for all valid considerations of cycle structure. The limitations of the technique mentioned in [359] has been addressed in [312]. Here, for a given valid cycle structure, a CA is always generated using an efficient mechanism with polynomial time complexity [312]. The research reported in [312] is framed as follows: (1) the CS of an n-cell CA can be obtained from the cycle structures of small sized CAs, (2) from a given valid cycle structure, the corresponding CA can be obtained if the input cycle structure is *decomposed* into the CSs of small sized CAs. Though the scheme mentioned in [312] covers linear/additive CAs, no literature has been found on the decomposition of cycle structure for *non-linear* cellular automata. This chapter studies whether the given cycle structure of a non-linear reversible CA can be decomposed or not.

For deciding the decomposition of cycle structure, we investigate the following factors: (i) the presence of particular rules or the arrangement of CA rules in the rule vector and (ii) some notable properties on the cyclic components or the length(s) of the cycle(s) in the cycle structure of the CA. We also introduce the notion of **irreducible** cellular automaton (CA whose cycle structure can not be decomposed into the cycle structure of any small sized CA) and **reducible** CAs (all CAs which are not irreducible). This chapter develops the theories for deciding whether the given cellular automaton is irreducible by considering its cycle structure only. We further show that our theory is effective for a fairly wide range of non-linear CAs. The contributions of this chapter are mentioned below.

- Classification of non-linear reversible cellular automata based on their cycle structures.

- An effective mechanism to decide whether the given cycle structure can be decomposed.

4.2 The Intrinsic Properties of Cycle Structure

As this chapter focusing on the decomposition of the cycle structure of an n-cell reversible CA, let us formally introduce it.

Definition 4.1 *Let us consider two CAs $\mathcal{R}$ and $\mathcal{R}^{n_1}$ of sizes n and n_1 $(< n)$ respectively with respective cycle structures $\mathtt{CS}_{\mathcal{R}} = [\mu_1(l_1), \mu_2(l_2), \cdots, \mu_m(l_m)]$ and $\mathtt{CS}_{\mathcal{R}^{n_1}} = [\mu_1^1(l_1^1), \mu_2^1(l_2^1), \cdots, \mu_{m_1}^1(l_{m_1}^1)]$. Then $\mathtt{CS}_{\mathcal{R}}$ is decomposed to $\mathtt{CS}_{\mathcal{R}^{n_1}}$ if the following conditions hold.*

1. *For every cycle of length l_i^1 of $\mathcal{R}^{n_1}$, there exist some unique cycles $\mathtt{c}_1, \mathtt{c}_2, \cdots, \mathtt{c}_k$ in $\mathcal{R}$, the lengths of which are multiple of l_i^1.*
2. *The summation of the lengths of $\mathtt{c}_j$s equals to $l_i^1 \times 2^{n-n_1}$.*

Let us consider the following examples.

Example 4.1 Let us consider a 6-cell CA $\langle 6, 86, 15, 57, 86, 240\rangle$. Cycle structure of this CA is $[4(5), 3(8), 1(20)]$. Let us also consider another 3-cell CA $\langle 6, 86, 5\rangle$ whose CS is $[1(1), 1(2), 1(5)]$. Now, every cycle length in the 6-cell CA is multiple of some cycle length of the 3-cell CA (satisfying the first condition of Definition 4.1). Besides, the lengths of the cycles of $\langle 6, 86, 15, 57, 86, 240\rangle$ satisfies the second condition of Definition 4.1. Consider the following cases for $\langle 6, 86, 15, 57, 86, 240\rangle$. Let us take five cycles with lengths 20, 5, 5, 5 and 5. The summation of these lengths is $5 \times 2^{6-3}$ where 5 is a cycle length of $\langle 6, 86, 5\rangle$. In the same manner, we get two cycles of length 8 such that $2 \times 2^{6-3}$ and one more cycle of same length 8 which equals to $1 \times 2^{6-3}$. Therefore, the CS of $\langle 6, 54, 15, 57, 86, 240\rangle$ is decomposed to the CS of $\langle 6, 54, 5\rangle$. There may exist multiple CAs of size 3 with same cycle structure of $\langle 6, 54, 5\rangle$. For instances we can consider $\langle 5, 30, 20\rangle$ and $\langle 6, 169, 80\rangle$. Therefore, $[4(5), 3(8), 1(20)]$ can be decomposed to the CSs of multiple CAs.

Example 4.2 The cycle structure of a 5-cell CA $\langle 5, 105, 54, 101, 5\rangle$ is $[1(1), 1(2), 1(6), 1(23)]$. This cycle structure can never be decomposed as one cycle length 23 is a prime number. Even if we consider 1-cell CA with two cycles of length 1, the CS of $\langle 5, 105, 54, 101, 5\rangle$ can not be decomposed as $1 \times 2^{5-1} < 23$ (the second condition of Definition 4.1 is not satisfied). Thus the CS of $\langle 5, 105, 54, 101, 5\rangle$ can not be decomposed into the CS of any small sized CA.

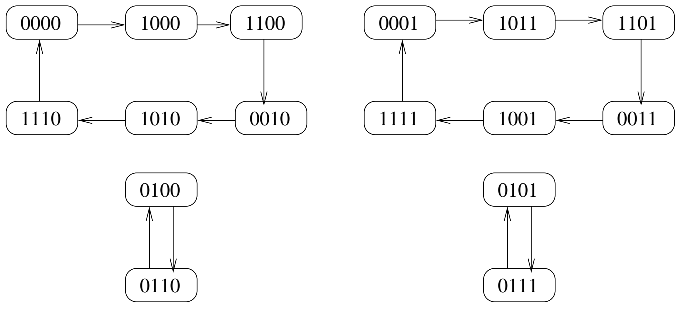

Figure 4.1: The cyclic space of CA $\langle 5, 60, 54, 68\rangle$

For deciding whether the CS of the given n-cell CA can be decomposed or not, we need to get small sized CAs. We may get a large pool of n_1-cell CAs where $n_1 < n$, but the challenge is how do we identify the appropriate small sized CAs for which the conditions mentioned in Definition 4.1 are satisfied. So, to get rid of the problem, we start either to look into the cyclic components of the n-cell CA for discovering some notable properties on the cycle structure decomposition (we can consider Example 4.2 as a reference) or scanning the rule vector for figuring out the significant CA rule for the decomposition of cycle structure. Let us understand how the presence of rules are significant for deciding the decomposition of the CS of an n-cell CA using an example. Let us consider a 4-cell CA $\mathcal{R} = \langle 5, 60, 54, 68\rangle$ (see Figure 4.1). In Figure 4.1, two cycles of length 2 are generated where every configuration involved in those cycles has **01** as the *prefix* (see Definition 2.7 in Chapter 2) sub-configuration for 2 cells. A prefix sub-configuration of $\mathtt{x} = (x_0x_1x_2x_3)$ for 2 cells is represented by (x_0x_1). CA $\mathcal{R}$ also generates two cycles of length 6 where a collection of prefix sub-configurations $(\mathbf{00}, \mathbf{10}, \mathbf{11})$ for 2 cells reappears in each of the cycles. Now, $(\mathbf{00}, \mathbf{10}, \mathbf{11})$ and $(\mathbf{01})$ can be considered as the cycles of a reversible CA size 2. Here, $(\mathbf{00}, \mathbf{10}, \mathbf{11})$ and $(\mathbf{01})$ are the cycles of CA $\mathcal{R}^1 = \langle 5, 20\rangle$ where $20[\mathtt{r}] = 60[\mathtt{r}]$ for all RMTs $\mathtt{r} \in \{0, 2, 4, 6\}$ and all odd RMTs $(\{1, 3, 5, 7\})$ are ignored. Therefore, $[2(2), 2(6)]$, the CS of $\mathcal{R}$ can be *decomposed* into the CS of $\mathcal{R}^1$ i.e., $[1(1), 1(3)]$. Here, $\mathcal{R}_1 = 60$ is the rule for which the CS of $\mathcal{R}$ can be decomposed. Let us consider another 4-cell CA $\langle 5, 120, 85, 20\rangle$ where the same collection of sub-configurations $(\mathbf{00}, \mathbf{10}, \mathbf{11})$ and $(\mathbf{01})$ reappear in the cycles of $\langle 5, 120, 85, 20\rangle$ (see Figure 4.2). Here, each of sub-configurations is *suffix* (see Definition 2.8 in Chapter 2) sub-configuration for 2 cells. A suffix sub-configuration of $\mathtt{x} = (x_0x_1x_2x_3)$ for 2 cells is represented by (x_2x_3). Now, $(\mathbf{00}, \mathbf{10}, \mathbf{11})$ and $(\mathbf{01})$ can be considered as the cycles of a reversible CA of size 2 $\langle 5, 20\rangle$ where $5[\mathtt{s}] = 85[\mathtt{s}]$ for all RMTs $\mathtt{s} \in \{0, 1, 2, 3\}$ and RMTs 4, 5, 6 and 7 are ignored. The cycle structures of $\langle 5, 20\rangle$ and $\langle 5, 120, 85, 20\rangle$ are respectively $[1(1), 1(3)]$ and $[2(3), 1(4), 1(6)]$. Now, every cycle length of CA $\langle 5, 120, 85, 20\rangle$ is multiple of some cycle length of CA $\langle 5, 20\rangle$. Therefore, the cycle structure of $\langle 5, 120, 85, 20\rangle$ can be *decomposed* into the CS of $\langle 5, 20\rangle$ and

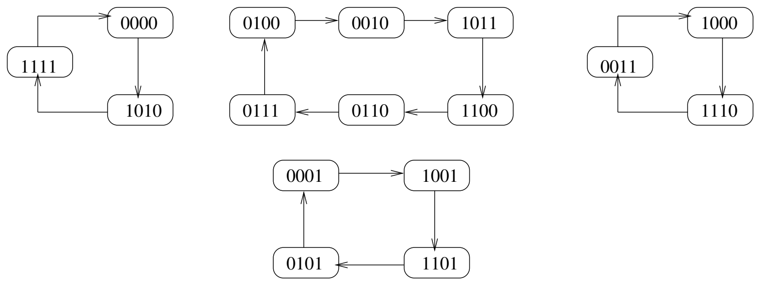

Figure 4.2: The cyclic space of CA $\langle 5, 120, 85, 20\rangle$

$\mathcal{R}_2 = 85$ in $\mathcal{R}$ is the rule for which $\texttt{CS}_{\mathcal{R}}$ is decomposed. As the cycle structures of CAs $\langle 5, 60, 54, 68\rangle$ and $\langle 5, 120, 85, 20\rangle$ both can be decomposed, therefore, the cyclic spaces of both the CAs *contain* the cyclic spaces of small sized CAs. By exploring the cyclic spaces of CAs, it can also be observed that two different CAs of same size possess the same cycle structure. As an example, we can consider CAs $\langle 6, 232, 90, 20\rangle$ and $\langle 9, 142, 165, 65\rangle$ as both of them have the same cycle structure $[2(1), 1(3), 1(11)]$ (Figure 4.3). Next, we introduce some terminologies which are useful to deduce some properties on the decomposition of CSs of CAs.

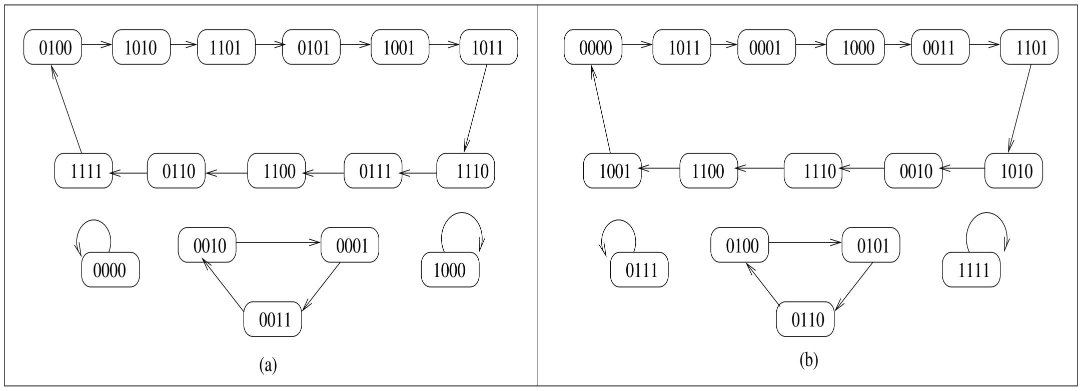

Figure 4.3: CAs $\langle 6, 232, 90, 20\rangle$ and $\langle 9, 142, 165, 65\rangle$ have same cycle structure $[2(1), 1(3), 1(11)]$.

4.2.1 Some Terminologies

Formally, a CA of size n can be interpreted as a function $G_n : C_n \to C_n$ where $C_n = \{0, 1\}^n$ is the collection of all possible configurations of length n. Let $\mathtt{x} = (x_0 x_1 \cdots x_{n-1})$ denote a configuration of the n-cell CA, where each x_i is the present state of cell i. A CA of size n forms a cycle of length $l \in \mathbb{N}$, where l is in between of 1 to $2^n - 1$ (where $n > 3$), if $G_n^l(\mathtt{x}) = \mathtt{x}$ for some $\mathtt{x} \in C_n$. As we use reversible CA, therefore, $G_n : C_n \to C_n$ be a bijective function. Let us define a cycle of a CA as follows.

Definition 4.2 *Let $G_n : C_n \to C_n$ be a bijective function such that $G^l_n(\mathtt{x}) = \mathtt{x}$ for every $\mathtt{x} \in C_n$ for some $l \in \mathbb{N}$. Here, $G_n(\mathtt{x}) = G_n(x_0x_1 \cdots x_{n-1}) = x^1_0x^1_1 \cdots x^1_{n-1}$ where every $x^1_i \in \{0, 1\}$ $(0 \leqslant i \leqslant l - 1)$ and $\mathtt{c} = (\mathtt{x}, G_n(\mathtt{x}), G^2_n(\mathtt{x}), \cdots, G^{l-1}_n(\mathtt{x}))$ represents the cycle of length l generated by G_n.*

Let G_n and $G_{n'}$ be two bijective functions which are defined over respectively $\{0, 1\}^n$ and $\{0, 1\}^{n'}$.

Definition 4.3 *Let G_n and $G_{n'}$ be the global transition functions of two reversible CAs of same size $(n = n')$. We can say that G_n and $G_{n'}$ are **isomorphic** to each other if for every cycle $\mathtt{c}_j$ of length l_j in G_n there exists a unique cycle $\mathtt{c}'_j$ in $G_{n'}$ of the same length l_j and vice versa.*

Let $G_{n_1} : \{0, 1\}^{n_1} \to \{0, 1\}^{n_1}$ be another bijective function where $n_1 < n$.

Definition 4.4 *G_{n_1} is said to be **left** quasi isomorphic to G_n if $G_n(\mathtt{x}) = \mathtt{y}$ implies $G_{n_1}(\mathtt{prefix}(\mathtt{x}, n_1)) = \mathtt{prefix}(\mathtt{y}, n_1)$ for every $\mathtt{x} \in \{0, 1\}^n$.*

Definition 4.5 *G_{n_1} is said to be **right** quasi isomorphic to G_n if $G_n(\mathtt{x}) = \mathtt{y}$ implies $G_{n_1}(\mathtt{suffix}(\mathtt{x}, n_1)) = \mathtt{suffix}(\mathtt{y}, n_1)$ for every $\mathtt{x} \in \{0, 1\}^n$.*

Definition 4.6 *G_{n_1} is said to be quasi isomorphic to G_n if G_{n_1} is either left or right quasi isomorphic to G_n.*

4.2.2 Influential CA Rules for the Decomposition of Cycle Structures

It is already mentioned that the cycle structure of n-cell reversible CA is decomposed into an n_1-cell reversible CA if the cyclic space of n_1-cell CA is contained in the cyclic space of n-cell CA. Therefore, every cycle of that n_1-cell CA reappears in some cycle(s) of the n-cell CA either once or multiple times. As reversible cellular automata can be represented as bijective functions, thus the n_1-cell CA is *quasi isomorphic* to n-cell CA. Now, we focus on the design of rule vector of n-cell CA, for which there exists a quasi isomorphic CA.

It is already shown that the presence of rules like 60, 85 in the rule vectors of the CAs guarantee the decomposition of the cycle structure of those CAs. The immediate question raised is why the presence of those rules in a CA are definitive for decomposing its cycle structure. Let us give the answer in affirmative way.

Let $\mathtt{x}_1$ can be represented as $\mathtt{prefix}(\mathtt{x}, n_1)$ where $\mathtt{x}$ and $\mathtt{x}_1$ are the configurations of CAs $\mathcal{R}$ of size n and $\mathcal{R}^{n_1}$ of size n_1 respectively. Let $\mathtt{x}_1$ be a member of cycle $\mathtt{c}$ of $\mathcal{R}^{n_1}$. If every $\mathtt{x}_1$ comes back into view in a cycle of $\mathcal{R}$ after a time period which is multiple of the length of $\mathtt{c}$, therefore, state change only at cell n_1 has no influence in at cell $n_1 - 1$. Therefore, cell $n_1 - 1$ is *independent* of its *right* neighbor. Analogously, if $\mathtt{x}_1$ is represented as $\mathtt{suffix}(\mathtt{x}, n_1)$, therefore, state change only at cell $n - n_1 - 1$ has no effect on cell $n - n_1$. Thus cell $n - n_1$ is *independent* of its *left* neighbor. When cell $n_1 - 1$ is independent of its right neighbor, a sequence of sub-configuration(s) reappears from cell 0 to cell $n_1 - 1$ in every cycle of $\mathcal{R}$; such CA brings the notion of *left quasi isomorphic* CA. Similarly, if cell $n - n_1$ of n-cell CA is independent of its left neighbor, then a sequence of sub-configuration(s) reappears from cells $n - n_1$ to $n - 1$ in every cycle of $\mathcal{R}$. Therefore, for such CA, there exists a *right quasi isomorphic* CA. The (left or right) quasi isomorphic CA is designed in the following manner.

Proposition 4.1 *An n_1-cell ($n < n_1$) CA $\mathcal{R}^{n_1} = \langle \mathcal{R}^1_0, \mathcal{R}^1_1, \cdots, \mathcal{R}^1_{n_1-1}\rangle$ is said to be left quasi isomorphic to $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n_1-1}, \cdots, \mathcal{R}_{n-1}\rangle$ if the following conditions are satisfied.*

- $\mathcal{R}_0[0yz] = \mathcal{R}^1_0[0yz]$ $(\forall y, z \in \{0,1\})$
- $\mathcal{R}_j[xyz] = \mathcal{R}^1_j[xyz]$ $(\forall x, y, z \in \{0,1\}$ *and* $\forall j \in \{1, 2, \cdots, n_1 - 2\})$
- $\mathcal{R}_{n_1-1}[xy0] = \mathcal{R}_{n_1-1}[xy1] = \mathcal{R}^1_{n_1-1}[xy0]$ $(\forall x, y \in \{0,1\})$

Proof: Let $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n_1-1}, \cdots, \mathcal{R}_{n-1}\rangle$ be an n-cell CA and $\mathcal{R}^{n_1} = \langle \mathcal{R}^1_0, \mathcal{R}^1_1, \cdots, \mathcal{R}^1_{n_1-1}\rangle$ be an n_1 cell CA where $n_1 < n$. Let us consider the following cases.

- **Case 1:** If $\mathcal{R}_j[0yz] = \mathcal{R}^1_j[0yz]$ $\forall y, z \in \{0,1\}$ and $j = 0$, then state transition in $\mathcal{R}$ and $\mathcal{R}^{n_1}$ are identical at cell 0.
- **Case 2:** If $\mathcal{R}_j[xyz] = \mathcal{R}^1_j[xyz]$ $\forall x, y, z \in \{0,1\}$ and $\forall j \in \{1, 2, \cdots, n_1 - 2\}$, then $\mathcal{R}$ and $\mathcal{R}^{n_1}$ follow same state transitions from cells 1 to $n_1 - 2$.
- **Case 3:** At cell $n_1 - 1$, if $\mathcal{R}_j[xyz] = \mathcal{R}^1_j[xy0]$ $\forall x, y, z \in \{0,1\}$ and $j = n_1 - 1$, then $\mathcal{R}$ and $\mathcal{R}^{n_1}$ follow same state transition at cell $n_1 - 1$.

By considering all three cases, cells 0 to $n_1 - 1$ follow identical state transitions in $\mathcal{R}$ and $\mathcal{R}^{n_1}$. Thus $\mathcal{R}^{n_1}$ is left quasi isomorphic to $\mathcal{R}$. □

Proposition 4.2 *An n_1-cell ($n < n_1$) CA $\mathcal{R}^{n_1} = \langle \mathcal{R}_0^1, \mathcal{R}_1^1, \cdots, \mathcal{R}_{n_1-1}^1 \rangle$ is said to be right quasi isomorphic to $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n_1-1}, \cdots, \mathcal{R}_{n-1} \rangle$ if the following conditions are satisfied.*

- $\mathcal{R}_{n-1}[xy0] = \mathcal{R}_{n_1-1}^1[xy0]$ $(\forall x, y \in \{0,1\})$
- $\mathcal{R}_{n-n_1+j}[xyz] = \mathcal{R}_j^1[xyz]$ $(\forall x, y, z \in \{0,1\}$ *and* $j \in \{1, 2, \cdots, n_1 - 2\})$
- $\mathcal{R}_{n-n_1}[0yz] = \mathcal{R}_{n-n_1}[1yz] = \mathcal{R}_0^1[0yz]$ $(\forall y, z \in \{0,1\})$

Proof: Let $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n_1-1}, \cdots, \mathcal{R}_{n-1} \rangle$ be an n-cell CA and $\mathcal{R}^{n_1} = \langle \mathcal{R}_0^1, \mathcal{R}_1^1, \cdots, \mathcal{R}_{n_1-1}^1 \rangle$ be an n_1 cell CA where $n_1 < n$. Let us consider the following cases.

- **Case 1:** If $\mathcal{R}_{n-n_1}[xyz] = \mathcal{R}_0^1[0yz]$ $\forall x, y, z \in \{0,1\}$, then state transition at cell $n - n_1$ of $\mathcal{R}$ is identical to the state transition at cell 0 in CA $\mathcal{R}^{n_1}$.
- **Case 2:** If $\mathcal{R}_{n-n_1+j} = \mathcal{R}_j^1[xyz]$ $\forall x, y, z \in \{0,1\}$ and $\forall j \in \{1, 2, \cdots, n_1 - 2\}$, then, $\mathcal{R}$ and $\mathcal{R}^{n_1}$ follow same state transitions respectively from cells $n - n_1 + 1$ to $n - 2$ and 1 to $n_1 - 2$.
- **Case 3:** If $\mathcal{R}_{n-1}[xy0] = \mathcal{R}_{n_1-1}^1[xy0]$ $\forall x, y \in \{0,1\}$ then, $\mathcal{R}$ and $\mathcal{R}^{n_1}$ follow identical state transition at cells $n - 1$ and $n_1 - 1$ respectively.

By considering all three cases, cells 0 to $n_1 - 1$ in $\mathcal{R}^{n_1}$ follow identical state transitions of cells $n - n_1$ to $n - 1$ in $\mathcal{R}$. Thus $\mathcal{R}^{n_1}$ is right quasi isomorphic to $\mathcal{R}$. □

If a rule is independent of the state change of its neighbor(s), then such rule is called as *independent* rule.

Definition 4.7 *If $\mathcal{R}[xy0] = \mathcal{R}[xy1]$ $\forall x, y \in \{0,1\}$, then $\mathcal{R}$ is a right independent rule. If $\mathcal{R}[0yz] = \mathcal{R}[1yz]$ $\forall y, z \in \{0,1\}$, then $\mathcal{R}$ is a left independent rule.*

Here, out of 62 rules for non-uniform reversible CAs, only 15, 51, 60, 195, 204, and 240 are right independent rules if the cell position is from 1 to $n - 2$. For $\mathcal{R}_0$, we can get only a set of two right independent rules which are 3 and 12. Similarly, an $\mathcal{R}_i$ $(1 \le i \le n - 2)$ can be any of CA rules 51, 85, 102, 153, 170, and 204 if $\mathcal{R}_i$ is left independent rule in an n-cell reversible CA. For cell $n - 1$, we get a set of left independent rules which consisting of only 17 and 64. The presence of right or left independent rule is instrumental for deciding the decomposition of the CS of a CA.

Lemma 4.1 *If a cell of an n-cell reversible CA $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ follows either right independent rule or left independent rule, then the cycle structure of $\mathcal{R}$ is decomposed.*

Proof: Let us consider $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$. If cell i in $\mathcal{R}$ follows right independent rule, for any l length cycle of $\mathcal{R}$, there exists an l' (where l' divides l) length sequence of sub-configuration(s) from cell 0 to cell $i-1$ that appears l/l' times in that cycle. Therefore, the cycle structure of $\mathcal{R}$ is decomposed into the cycle structure of $(i+1)$-cell CA $\langle \mathcal{R}_0^1, \mathcal{R}_1^1, \cdots, \mathcal{R}_i^1 \rangle$ where $\mathcal{R}_j^1 = \mathcal{R}_j$ $\forall j \in \{0, \cdots, i-1\}$ and $\mathcal{R}_i^1[\mathtt{r}] = \mathcal{R}_i[\mathtt{r}]$ $\forall \mathtt{r} \in \{0, 2, 4, 6\}$. Analogously, if cell i in $\mathcal{R}$ follows left independent rule, the cycle structure of $\mathcal{R}$ is decomposed into the cycle structure of $(n-i)$-cell CA $\langle \mathcal{R}_0^1, \mathcal{R}_1^1, \cdots, \mathcal{R}_{(n-i)-1}^1 \rangle$ where $\mathcal{R}_j^1 = \mathcal{R}_{i+j}$ $\forall j \in \{1, \cdots, (n-i)-1\}$ and $\mathcal{R}_0^1[\mathtt{s}] = \mathcal{R}_i[\mathtt{s}]$ $\forall \mathtt{s} \in \{0, 1, 2, 3\}$. □

Corollary 4.1 : *An n-cell reversible CA $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ has $(i+1)$-cell (resp. $(n-i)$-cell) left (resp. right) quasi isomorphic CA if $\mathcal{R}_i$ is right independent (resp. left independent) rule.*

Proof: In $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$ if $\mathcal{R}_i$ is right independent rule, therefore, the $(i+1)$-cell CA $\mathcal{R}^{(i+1)} = \langle \mathcal{R}_0^1, \mathcal{R}_1^1, \cdots, \mathcal{R}_i^1 \rangle$ is designed in the following manner.

- $\mathcal{R}_0[0yz] = \mathcal{R}_0^1[0yz]$ $(\forall y, z \in \{0, 1\})$
- $\mathcal{R}_j[xyz] = \mathcal{R}_j^1[xyz]$ $(\forall x, y, z \in \{0, 1\}$ and $\forall j \in \{1, 2, \cdots, i-1\})$
- $\mathcal{R}_i[xy0] = \mathcal{R}_i[xy1] = \mathcal{R}_i^1[xy0]$ $(\forall x, y \in \{0, 1\})$

Using Proposition 4.1, we get $\mathcal{R}^{(i+1)}$ as the left quasi isomorphic to $\mathcal{R}$. Analogously, we can get $(n-i)$-cell right quasi isomorphic CA using Proposition 4.2. □

The presence of left or right independent rule in a CA guarantees the decomposition of its cycle structure. A large section of reversible CAs maintain this property. For the remaining, there is no such cell in $\mathcal{R}$ which follows left or right independent rule. Therefore, the CA $\mathcal{R}$ without left independent or right independent rule can not have any quasi isomorphic CA. The immediate question is can the CSs of all those CAs not be decomposed? The next section gives the answer in an affirmative manner.

4.3 Classification of Reversible CAs

An appealing observation is that the cycle structure of a CA can be decomposed even if there is no right or left independent rule in that CA. For instance, consider the 5-cell CA $\mathcal{R} = \langle 9, 89, 90, 105, 5 \rangle$. The cycle structure of the CA is $[1(2), 1(30)]$. Now, there are two possibilities. The lengths 2 and 30 can be multiple of 1 and 15 respectively. On the other hand, the lengths 2 and 30 both are multiple of 2. Therefore, we may get an 4-cell quasi isomorphic CA whose cycle structure is $[1(1), 1(15)]$ for the former case; whereas, for later case the CS of the 1-cell quasi isomorphic CA is $[1(2)]$. Therefore, either $\mathcal{R}_3$ is right independent or $\mathcal{R}_1$ is left independent rule in $\mathcal{R}$ for the 4-cell quasi isomorphic CA. Similarly, either $\mathcal{R}_0$ is right independent or $\mathcal{R}_4$ is left independent in $\mathcal{R}$ when 1-cell quasi isomorphic CA is considered. But no rule in $\mathcal{R}$ is either right independent or left independent. Therefore, the cyclic space of $\langle 9, 89, 90, 105, 5\rangle$ does not contain the cyclic space of small sized CA. On the other hand, another 5-cell CA $\langle 9, 89, 105, 60, 65\rangle$ has the same cycle structure $[1(2), 1(30)]$. In this CA, rule 60 is the *right independent* rule and the CS of $\langle 9, 89, 105, 60, 65\rangle$ is decomposed into the cycle structure of 4-cell left quasi isomorphic CA $\langle 9, 89, 105, 20\rangle$ where $20[\mathtt{r}] = 60[\mathtt{r}]$ for every even RMT $\mathtt{r}$ and the CS of $\langle 9, 89, 105, 20\rangle$ is $[1(1), 1(15)]$. Thus, a valid cycle structure of a CA can be decomposed even if the CA has no quasi isomorphic CA but it has an *isomorphic* CA which has a quasi isomorphic CA. This brings a new notion - *reducible* cellular automata.

Definition 4.8 *An n-cell reversible CA $\mathcal{R}$ is said to be **reducible** if any of the following conditions holds.*

1. *The rule vector $\mathcal{R}$ contains at least one right or left independent rule.*
2. *There exists an isomorphic CA of $\mathcal{R}$, say $\mathcal{R}'$ which contains at least one right or left independent rule.*

The reducible CA is detected by the presence of any *left* or *right independent* rule. Sometimes a reducible CA contains both right and left independent rules at cell $i-1$ and i such that the given CA has both left and right isomorphic CAs of sizes i and $(n-i)$ respectively. This type of CA is a special case of reducible CA. Let us categorize the reversible cellular automata based on the quasi isomorphic CAs.

Let $\mathcal{R}$ be an n-cell CA. Let $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ be respectively n_1 and n_2-cell CAs which are left quasi isomorphic and right quasi isomorphic to $\mathcal{R}$.

Definition 4.9 *An n-cell CA $\mathcal{R}$ is said to be **strictly reducible** if any of the following conditions holds.*

1. *There exists an n_1-cell CA $\mathcal{R}^{n_1}$ and an n_2-cell CA $\mathcal{R}^{n_2}$ which are respectively left and right quasi isomorphic to $\mathcal{R}$ and $n = n_1 + n_2$.*
2. *$\mathcal{R}$ does not follow the above condition but there exists an isomorphic CA of $\mathcal{R}$ which does.*

Definition 4.10 *An n-cell CA $\mathcal{R}$ is said to be **weakly reducible** if any of the following conditions holds.*

1. *$\mathcal{R}$ is not strictly reducible CA.*

2. *There exists an n_1-cell CA $\mathcal{R}^{n_1}$ which is left quasi isomorphic (resp. right quasi isomorphic) to $\mathcal{R}$ but there does not exist any n_2-cell right quasi isomorphic (resp. left quasi isomorphic) CA to $\mathcal{R}$ such that $n = n_1 + n_2$.*

3. *$\mathcal{R}$ does not follow the above condition but there exists an isomorphic CA of $\mathcal{R}$ which does.*

Definition 4.11 *An n-cell CA $\mathcal{R}$ is said to be **irreducible** if the following conditions hold.*

1. *$\mathcal{R}$ is neither strictly nor weakly reducible.*

2. *There does not exist any n_1-cell CA $\mathcal{R}^{n_1}$ which is quasi isomorphic to $\mathcal{R}$.*

3. *There does not exist any isomorphic CA of $\mathcal{R}$ which has n_1-cell quasi isomorphic CA.*

4.3.1 Strictly and Weakly Reducible Cellular Automata

From Definition 4.9 and Definition 4.10, it is clear that any reducible CA is either strictly reducible or weakly reducible. This section first figures out the properties of strictly reducible CAs. Suppose the first condition is true for an n-cell strictly reducible CA $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$. Therefore, cell $n_1 - 1$ in $\mathcal{R}$ is independent of its right neighbor as the CA has n_1-cell left quasi isomorphic CA and cell $n - n_2$ of $\mathcal{R}$ is independent of its left neighbor as there exists an n_2-cell right quasi isomorphic CA in $\mathcal{R}$. In strictly reducible CA, $n_1 = n - n_2$ and the following lemma reports the arrangement of CA rules in the rule vector $\mathcal{R}$ for getting strictly reducible CA.

Lemma 4.2 *Let $\mathcal{R} = (\mathcal{R}_0, \mathcal{R}_1, \cdots \mathcal{R}_{i-1}, \mathcal{R}_i, \cdots \mathcal{R}_{n-1})$ be an n-cell reversible CA. Then $\mathcal{R}$ is said to be strictly reducible CA if $\mathcal{R}_{i-1}$ and $\mathcal{R}_i$ are respectively right and left independent rules for some i $(1 \leq i \leq n-1)$.*

Proof: Let $\mathcal{R}$ be an n-cell reversible CA where $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \cdots \mu_m(l_m)]$. If $\mathcal{R}_{i-1}$ is right independent rule, there exists an i-cell left quasi isomorphic CA to $\mathcal{R}$ (according to Corollary 4.1). Similarly, if $\mathcal{R}_i$ is left independent rule, then $\mathcal{R}$ has $(n-i)$-cell right quasi isomorphic CA (according to Corollary 4.1). Therefore, $i + (n-i) = n$ and $\mathcal{R}$ is strictly reducible CA. □

Thus the presence of right and left independent rules at consecutive cell positions in the rule vector of a reversible CA guarantees the generation of strictly reducible CA. Now, the relation between $CS_{\mathcal{R}^{n_1}}$ and $CS_{\mathcal{R}^{n_2}}$ is presented for determining the $CS_{\mathcal{R}}$ when $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ are respectively left and right quasi isomorphic CAs to $\mathcal{R}$. So, we introduce an operator for establishing the relation between $CS_{\mathcal{R}^{n_1}}$ and $CS_{\mathcal{R}^{n_2}}$.

4.3.1.1 A Generic Operator

Let $G_{n_1} : X \to X$ be a bijective function such that $G^{l'}_{n_1}(x) = x$, for all $x \in X$ where $l' \in \mathbb{N}$. Also let for an $x \in X$, any element of X can be expressed as $G^t_{n_1}(x)$, for some t between 0 and $l' - 1$. Thus X can be expressed as $(x, G_{n_1}(x), G^2_{n_1}(x), \cdots, G^{l'-1}_{n_1}(x))$. Similarly, let $G_{n_2} : Y \to Y$ be another bijective function such that $G^{l''}_{n_2}(y) = y$, for all $y \in Y$ where $l'' \in \mathbb{N}$. Then Y can also be expressed as $(y, G_{n_2}(y), G^2_{n_2}(y), \cdots, G^{l''-1}_{n_2}(y))$ for an $y \in Y$.

Define an operator $\texttt{conc} : X \times Y \to Z$, which is *one-to-one* if and only if
(i) $x \texttt{ conc } y \neq x' \texttt{ conc } y$, for any $x' \neq x$.
(ii)$x \texttt{ conc } y \neq x \texttt{ conc } y'$, for any $y' \neq y$.
Define $Z = \{x \texttt{ conc } y \mid x \in X, y \in Y\}$. Thus $\texttt{conc} : X \times Y \to Z$ is *onto* as well; that is, it is a bijection. Hence, given any $z \in Z$, it is possible to uniquely determine the particular $x \in X$ and the $y \in Y$ such that $z = x \texttt{ conc } y$. Therefore, we can define the inverse operators $\texttt{concinv}_1$ and $\texttt{concinv}_2$ as $x = \texttt{concinv}_1(z)$ and $y = \texttt{concinv}_2(z)$, when $z = x \texttt{ conc } y$. Then we have the following result.

Lemma 4.3 *For any $a \in Z$, there exists a positive integer l such that $h^l(a) = a$, where the function $h : Z \to Z$ is defined as*

$$h(a) = G_{n_1}(\texttt{concinv}_1(a)) \texttt{ conc } G_{n_2}(\texttt{concinv}_2(a)), \textit{ for all } a \in Z.$$

Proof: Any element a of Z can be expressed of form $G^i_{n_1}(x) \texttt{ conc } {G_{n_2}}^j(y)$, for some i in 0 to $l' - 1$, and j between 0 to $l'' - 1$. Thus

$$\begin{aligned} h(a) &= G_{n_1}(\texttt{concinv}_1(a)) \texttt{ conc } G_{n_2}(\texttt{concinv}_2(a)) \\ &= G_{n_1}(\texttt{concinv}_1(G^i_{n_1}(x) \texttt{ conc } G^j_{n_2}(y))) \\ &\qquad \texttt{conc } G_{n_2}(\texttt{concinv}_2(G^i_{n_1}(x) \texttt{ conc } G^j_{n_2}(y))) \\ &= G^{i+1}_{n_1}(x) \texttt{ conc } {G_{n_2}}^{j+1}(y) \end{aligned}$$

Thus $h^l(a) = G^{i+l}_{n_1}(x) \texttt{ conc } {G_{n_2}}^{j+l}(y)$, for any l.

For $h^l(a)$ to be equal to a, we require (i) $G^{i+l}_{n_1}(x)$ must equal $G^i_{n_1}(x)$ which implies l must be a multiple of l' and (ii) $G^{j+l}_{n_2}(y)$ must equal $G^j_{n_2}(y)$ which implies l must be a multiple of l''. Thus l must be a multiple of both l' and l'', hence the least value of l which satisfies $h^l(a) = a$ is $\texttt{lcm}(l', l'')$. For $l = \texttt{lcm}(l', l'')$, $h^l(a) = G^{i+\texttt{lcm}(l',l'')}_{n_1}(x) \texttt{ conc } G^{j+\texttt{lcm(l',l'')}}_{n_2}(y) = G^i_{n_1}(x) \texttt{ conc } G^j_{n_2}(y) = a$. □

From Lemma 4.3 we also observe that for any $a \in Z$, the set $\{a, h(a), h^2(a), \cdots, h^{l-1}(a)\} \subseteq Z$. Thus we have the following result.

Corollary 4.2 *Z can be partitioned into $\texttt{gcd}(l', l'')$ disjoint subsets where each subset is of the form $\{a, h(a), h^2(a), \cdots, h^{l-1}(a)\}$, for $l = \texttt{lcm}(l', l'')$ and some $a \in Z$.*

Proof: Clearly $\mid Z \mid = l' \cdot l''$. As stated before, for any $a \in Z$, the set $\{a, h(a), h^2(a), \cdots, h^{l-1}(a)\}$ of cardinality $l = \texttt{lcm}(l', l'')$ is a subset of Z. Hence Z can be partitioned into $\frac{l' \cdot l''}{\texttt{lcm}(l', l'')} = \texttt{gcd}(l', l'')$ disjoint subsets, each of which is of the form $\{a, h(a), h^2(a), \cdots, h^{l-1}(a)\}$ for some $a \in Z$. $\square$

4.3.1.2 Application of the Operator to CAs

Now, we describe how a CA of size n can be formed from two CAs of size n_1 and n_2 ($n = n_1 + n_2$) such that the new CA generates a cycle of length larger than the cycles of the latter CAs. To do this, we need to appropriately establish the sets X, Y and the operator `conc` defined already.

Let the two CAs of sizes n_1 and n_2 be respectively interpreted as functions $G_{n_1} : C_1 \to C_1$ and $G_{n_2} : C_2 \to C_2$, where C_1 and C_2 are respectively the set of configurations of size n_1 and n_2. Consider a particular cycle of length l' of the CA G_{n_1}, and let $\texttt{x}$ be a configuration that belongs to this cycle. Then the set of configurations in this cycle can be represented as $X = \{\texttt{x}, G_{n_1}(\texttt{x}), G^2_{n_1}(\texttt{x}), \cdots, G^{l'-1}_{n_1}(\texttt{x})\}$. In the same manner, if $\texttt{y}$ be a configuration belonging to a cycle of length l'' of the CA G_{n_2}, then the set of configurations in this cycle can be represented as $Y = \{\texttt{y}, G_{n_2}(\texttt{y}), G^2_{n_2}(\texttt{y}), \cdots, G^{l''-1}_{n_2}(\texttt{y})\}$.

We now define the `conc` operator on every configuration $\texttt{x} \in X$ and $\texttt{y} \in Y$. Suppose $\texttt{x} = (x_0 x_1 \cdots x_{n_1-1})$ and $\texttt{y} = (y_0 y_1 \cdots y_{n_2-1})$. Then

$$\texttt{x conc y} = (x_0 x_1 \cdots x_{\texttt{n}_1-1} y_0 y_1 \cdots y_{\texttt{n}_2-1}).$$

Let Z be the set defined as $Z = \{\texttt{x conc y} \mid \texttt{x} \in X, \texttt{y} \in Y\}$.

Note that `conc` is a bijective operator, thus the inverse operators $\texttt{concinv}_1$ and $\texttt{concinv}_2$ can be defined as:

$$\texttt{concinv}_1(x_0 x_1 \cdots x_{n_1-1} y_0 y_1 \cdots y_{n_2-1}) = (x_0 x_1 \cdots x_{n_1-1})$$

$$\texttt{concinv}_2(x_0 x_1 \cdots x_{n_1-1} y_0 y_1 \cdots y_{n_2-1}) = (y_0 y_1 \cdots y_{n_2-1}).$$

Then Corollary 4.2 implies that there exists $\texttt{gcd}(l', l'')$ disjoint subsets of Z, each of which consists of configurations that belong to a cycle having length

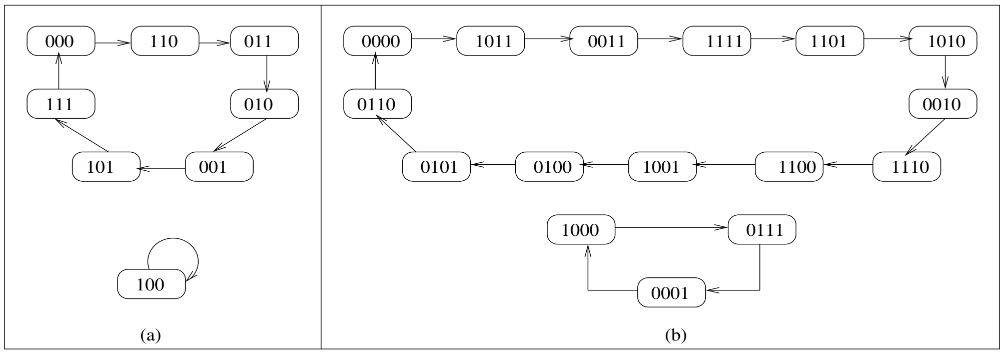

Figure 4.4: Configuration transition diagrams of 3-cell reversible CA $\langle 5, 105, 20\rangle$ and 4-cell reversible CA $\langle 5, 150, 45, 65\rangle$

$l = \texttt{lcm}(l', l'')$. The configurations of these cycles belong to a CA h of size $n = n_1 + n_2$, where h is determined as given in Lemma 4.3. Thus,

$$\begin{aligned} & h(x_0x_1 \cdots x_{n_1-1}y_0y_1 \cdots y_{n_2-1}) \\ = \;& G_{n_1}(\texttt{concinv}_1(x_0x_1 \cdots x_{n_1-1}y_0y_1 \cdots y_{n_2-1})) \quad \texttt{conc} \\ & \qquad G_{n_2}(\texttt{concinv}_2(x_0x_1 \cdots x_{n_1-1}y_0y_1 \cdots y_{n_2-1})) \\ = \;& G_{n_1}(x_0x_1 \cdots x_{n_1-1}) \quad G_{n_2}(y_0y_1 \cdots y_{n_2-1}). \end{aligned}$$

This CA h is said to be generated through *concatenation* of the two CAs G_{n_1} and G_{n_2}. We have thus established the main result of this section using corollary 4.2.

Corollary 4.3 : *Let l' and l'' be the lengths of any two cycles of the CAs of size n_1 and n_2 respectively. Then a CA of size $n_1 + n_2$ can be constructed which generates $\texttt{gcd}(l', l'')$ cycles, each of which is of length $\texttt{lcm}(l', l'')$.*

Example 4.3 For 4-cell CA $(9, 150, 45, 65)$, consider the cycle consisting of set of configurations $X = \{1000, 0111, 0001\}$ (Figure 4.4(b)). We consider another cycle $Y = \{000, 110, 011, 010, 001, 101, 111\}$ of 3-cell CA $(5, 105, 20)$ (Figure 4.4(a)). Using the operator `conc` on every pair of configurations, one from set X and another from Y, $Z = \{$**1000**000, **0111**110, **0001**011, **1000**010, **0111**001, **0001**101, **1000**111, **0111**000, **0001**110, **1000**011, **0111**010, **0001**001, **1000**101, **0111**111, **0001**000, **1000**110, **0111**011, **0001**010, **1000**001, **0111**101, **0001**111$\}$ (see Figure 4.5). For clarity, we have marked the first four states of every configuration in bold to denote that they correspond to cycle X as well. Here, $l' = 3$ and $l'' = 7$. So, $l = 21$ (using $\texttt{lcm}(l', l'')$). Here, the generated CA of size 7 generates only one (using $\texttt{gcd}(l', l'')$) cycle of length 21.

Let $\mathcal{R}$, $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ be the rule vectors of CAs h, G_{n_1} and G_{n_2} respectively. Let $\mathcal{R}^{n_1} = \langle \mathcal{R}^1_0, \mathcal{R}^1_1, \cdots \mathcal{R}^1_{n_1-1}\rangle$, $\mathcal{R}^{n_2} = \langle \mathcal{R}^2_0, \mathcal{R}^2_1, \cdots \mathcal{R}^2_{n_2-1}\rangle$ and $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1,$

$\cdots \mathcal{R}_{n-1}\rangle$. Since the CA h is generated by concatenation of G_{n_1} and G_{n_2}, the next state of $(n_1 - 1)^{th}$ cell of $\mathcal{R}$ should not to be dependent on the present state of the ${n_1}^{th}$ cell, for any configuration. That is, the $(n_1 - 1)^{th}$ cell must be independent of its right neighbor. Similarly, the ${n_1}^{th}$ cell of any configuration of $\mathcal{R}$ must be independent of its left neighbor. We define a new operator $\odot$ on CAs and denote $\mathcal{R} = \mathcal{R}^{n_1} \odot \mathcal{R}^{n_2}$ if the following conditions hold.

- Set $\mathcal{R}_i = \mathcal{R}_i^1$ for $0 \le i \le n_1 - 2$.
- Set $\mathcal{R}_{i+n_1} = \mathcal{R}_i^2$ for $1 \le i \le n_2 - 1$.
- For any $x, y \in \{0, 1\}$, $\mathcal{R}_{n_1-1}[xy0] = \mathcal{R}_{n_1-1}[xy1] = \mathcal{R}_{n_1-1}^1[xy0]$.
- For any $x, y \in \{0, 1\}$, $\mathcal{R}_{n_1}[0xy] = \mathcal{R}_{n_1}[1xy] = \mathcal{R}_0^2[0xy]$.

If the above conditions hold, the CAs $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ are said to be concatenated to form $\mathcal{R}$. Let us consider the following example for illustration.

Example 4.4 Suppose our objective is to generate a 7-cell CA through concatenation of a 4-cell CA $\langle 9, 150, 45, 65\rangle$ and a 3-cell CA $\langle 5, 105, 20\rangle$. Let $\mathcal{R}_0, \mathcal{R}_1, \cdots \mathcal{R}_6$ be the rule vector of the 7-cell CA. Therefore, $\mathcal{R}_3$ of 7-cell CA is selected in such a way that $\mathcal{R}_3[xy0] = \mathcal{R}_3[xy1] = 65[xy0]$ where $x, y \in \{0, 1\}$. It means that $\mathcal{R}_3[000] = \mathcal{R}_3[001] = 65[000] = 1$, $\mathcal{R}_3[010] = \mathcal{R}_3[011] = 65[010] = 0$, $\mathcal{R}_3[100] = \mathcal{R}_3[101] = 65[100] = 0$ and $\mathcal{R}_3[110] = \mathcal{R}_3[111] = 65[110] = 1$. Therefore, $\mathcal{R}_3 = 195$ which is equivalent to 65. Similarly, $\mathcal{R}_4$ of 7-cell CA

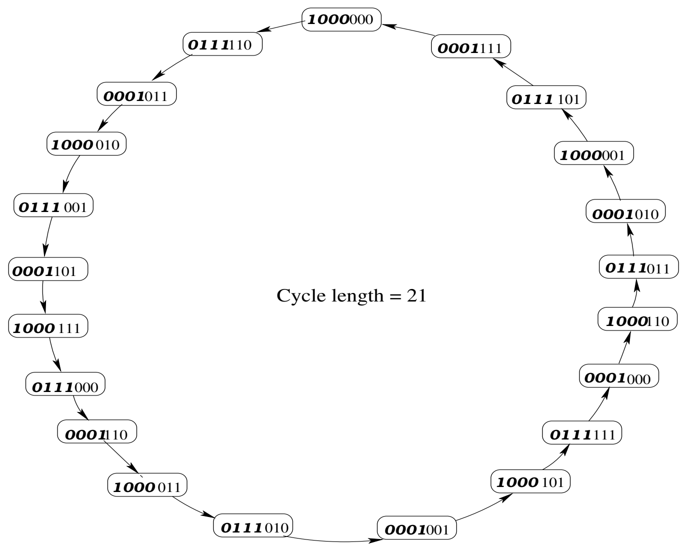

Figure 4.5: Configuration transition diagram (partial) of $\langle 9, 150, 45, 195, 85, 105, 20\rangle$

Algorithm 7 Algorithm to find the CS of an n-cell reversible CA formed by concatenating two CAs of sizes n_1 and n_2 respectively.

```
    Input: $\texttt{CS}_{\mathcal{R}^{n_1}} = [\mu_1^1(l_1^1), \mu_2^1(l_2^1), \cdots, \mu_{m_1}^1(l_{m_1}^1)]$, $\texttt{CS}_{\mathcal{R}^{n_2}} = [\mu_1^2(l_1^2), \cdots, \mu_{m_2}^2(l_{m_2}^2)]$
    Output: $\texttt{CS}_{\mathcal{R}} = \texttt{CS}_{\mathcal{R}^{n_1}} \times \texttt{CS}_{\mathcal{R}^{n_2}}$
1:  Compute $n_{ij} = \mu_i^1 \cdot \mu_j^2 \cdot \texttt{gcd}(l_i^1, l_j^2)$ and $l_{ij} = \texttt{lcm}(l_i^1, l_j^2)$, for $i \in \{1, 2, \cdots, m_1\}$ and
    $j \in \{1, 2, \cdots, m_2\}$.
2:  Create a set $L$ with the distinct $l_{ij}$s obtained above.
3:  for each element $l$ in $L$ do
4:      $\mu_l = \sum_{ij|l_{ij}=l} n_{ij}$
5:  end for
6:  Return $[\mu_l(l)]_{l \in L}$
```

must be selected in such a way that $\mathcal{R}_4[0xy] = \mathcal{R}_3[1xy] = 5[0xy]$. Here, we get $\mathcal{R}_4 = 85$. Therefore, the resultant CA formed by concatenation of $\langle 9, 150, 45, 65\rangle$ and $\langle 5, 105, 20\rangle$ is $\langle 9, 150, 45, 195, 85, 105, 20\rangle$.

We can extend Corollary 4.3 for multiple cycle lengths of two CAs. Therefore, we can find the lengths of all cycles of a CA of size $n_1 + n_2$ formed by concatenation of two CAs of sizes n_1 and n_2 respectively. Let $\mathcal{R}^{n_1}$ has cycles of lengths l_1^1, l_2^1, $\cdots$, $l_{m_1}^1$. Similarly, $\mathcal{R}^{n_2}$ has cycles of lengths l_1^2, l_2^2, $\cdots$, $l_{m_2}^2$. Therefore, $\mathcal{R}$ generates $\texttt{gcd}(l_i^1, l_j^2)$ cycles, each of which is of length $\texttt{lcm}(l_i^1, l_j^2)$ where $i = 1, 2, \cdots m_1$ and $j = 1, 2, \cdots m_2$. Let $\texttt{CS}_{\mathcal{R}^{n_1}} = [\mu_1^1(l_1^1), \mu_2^1(l_2^1), \cdots, \mu_{m_1}^1(l_{m_1}^1)]$ and $\texttt{CS}_{\mathcal{R}^{n_2}} = [\mu_1^2(l_1^2), \mu_2^2(l_2^2), \cdots, \mu_{m_2}^2(l_{m_2}^2)]$. As cycles of lengths l_i^1 and l_j^2 generate $\texttt{gcd}(l_i^1, l_j^2)$ cycles with unique length $\texttt{lcm}(l_i^1, l_j^2)$, therefore, $\mu_i^1(l_i^1)$ and $\mu_j^2(l_j^2)$ generate $\mu_i^1 \cdot \mu_j^2 \cdot \texttt{gcd}(l_i^1, l_j^2)$ cycles with unique length $\texttt{lcm}(l_i^1, l_j^2)$. Formally, $\texttt{CS}_{\mathcal{R}}$ is evaluated from $\texttt{CS}_{\mathcal{R}^{n_1}}$ and $\texttt{CS}_{\mathcal{R}^{n_2}}$ using the Algorithm 7 and is denoted using the following equation.

$$\begin{aligned}
\texttt{CS}_{\mathcal{R}} &= \texttt{CS}_{\mathcal{R}^{n_1}} \times \texttt{CS}_{\mathcal{R}^{n_2}} \\
&= [\mu_i^1 \cdot \mu_j^2 \cdot \texttt{gcd}(l_i^1, l_j^2)(\texttt{lcm}(l_i^1, l_j^2))] \ \forall (i \in \{1, \cdots, m_1\}, j \in \{1, \cdots, m_2\}) \\
&= [\mu_1(l_1), \mu_2(l_2), \cdots, \mu_m(l_m)] \qquad (4.1)
\end{aligned}$$

Therefore, the CS of any n-cell strictly reducible CA can be decomposed in the form of $[\mu_i^1 \cdot \mu_j^2 \cdot \texttt{gcd}(l_i^1, l_j^2)(\texttt{lcm}(l_i^1, l_j^2))] \ \forall (i \in \{1, \cdots, m_1\}, j \in \{1, \cdots, m_2\})$.

Till now, as per our discussion, the rule vector of a strictly reducible CA contains right independent and left independent rules at two consecutive position. But according to Definition 4.9, if the cycle structure of a CA satisfies Equation 4.1, then also the corresponding CA is strictly reducible. Now, the challenge is how do we decide the given valid arbitrary CS is of some strictly reducible CA? Though a scheme had been reported in [312], but that covers only linear/additive CAs. This work reports some properties on the cycle structure for arbitrary reversible CAs.

Proposition 4.3 *Let $\mathcal{R}$ be an n-cell reversible CA. Let $\texttt{CS}_{\mathcal{R}} = [1(l_1), 1(l_2)]$ where l_1 and l_2 are even numbers. If $\texttt{gcd}(l_1, l_2) = 2^i$ for some i between 1 to 3 (both inclusive), then $\mathcal{R}$ is strictly reducible.*

Proof: Since $\mathcal{R}$ is an n-cell reversible CA, $l_1 + l_2 = 2^n$. Let $l_1^1 = l_1/2^i$ and $l_1^2 = l_2/2^i$, then $l_1^1 + l_1^2 = 2^{n-i}$. Also since $\texttt{gcd}(l_1, l_2) = 2^i$, therefore, l_1^1 and l_1^2 are mutually prime and odd. Let us consider an n_1-cell reversible CA $\mathcal{R}^{n_1}$ and an $n_2 = i$-cell reversible CA $\mathcal{R}^{n_2}$ such that $\texttt{CS}_{\mathcal{R}^{n_1}} = [1(l_1^1), 1(l_1^2)]$ and $\texttt{CS}_{\mathcal{R}^{n_2}} = [1(2^{n_2})]$. Obviously, $\texttt{CS}_{\mathcal{R}^{n_2}}$ exists since $n_2 = i$ is one of $1, 2, 3$. Therefore, $\texttt{CS}_{\mathcal{R}}$ can be obtained using $\texttt{CS}_{\mathcal{R}^{n_1}} \times \texttt{CS}_{\mathcal{R}^{n_2}}$. Therefore, $\mathcal{R}$ is strictly reducible. □

Proposition 4.4 *Let $\mathcal{R}$ be an n-cell reversible CA. Let $\texttt{CS}_{\mathcal{R}} = [1(1), \mu(l)]$ for some $\mu > 1$. If $l = 2^{n_1} - 1$ for some n_1 ($1 \le n_1 < n$), then $\mathcal{R}$ is strictly reducible.*

Proof: Clearly, $\mu = (2^n - 1)/l = (2^n - 1)/(2^{n_1} - 1)$. Since μ is an integer, $(2^{n_1} - 1)$ shall divide $(2^n - 1)$, and so n_1 must divide n. Suppose $n = kn_1$ for some integer $k > 1$. Let us consider an n_1-cell reversible CA $\mathcal{R}^{n_1}$ such that $\texttt{CS}_{\mathcal{R}^{n_1}} = [1(1), 1(2^{n_1} - 1)]$. Therefore,

$$\begin{aligned}
\texttt{CS}_{\mathcal{R}^{n_1}} \times \texttt{CS}_{\mathcal{R}^{n_1}} &= [1(1), 1(2^{n_1} - 1)] \times [1(1), 1(2^{n_1} - 1)] \\
&= [1(1), 1(2^{n_1} - 1), 1(2^{n_1} - 1), (2^{n_1} - 1)(2^{n_1} - 1)] \\
&= [1(1), (\frac{2^{2n_1} - 1}{2^{n_1} - 1})(2^{n_1} - 1)]
\end{aligned}$$

Similarly,

$$\begin{aligned}
\texttt{CS}_{\mathcal{R}^{n_1}} \times \texttt{CS}_{\mathcal{R}^{n_1}} \times \texttt{CS}_{\mathcal{R}^{n_1}} &= [1(1), (\frac{2^{2n_1} - 1}{2^{n_1} - 1})(2^{n_1} - 1)] \times [1(1), 1(2^{n_1} - 1)] \\
&= [1(1), 1(2^{n_1} - 1), (\frac{2^{2n_1} - 1}{2^{n_1} - 1})(2^{n_1} - 1), (2^{2n_1} - 1)(2^{n_1} - 1)] \\
&= [1(1), (\frac{2^{3n_1} - 1}{2^{n_1} - 1})(2^{n_1} - 1)]
\end{aligned}$$

Proceeding in this manner,

$$\begin{aligned}
\texttt{CS}_{\mathcal{R}^{n_1}} \times \cdots (k \text{ terms}) \cdots \times \texttt{CS}_{\mathcal{R}^{n_1}} &= [1(1), (\frac{2^{kn_1} - 1}{2^{n_1} - 1})(2^{n_1} - 1)] \\
&= [1(1), \mu(l)] = \texttt{CS}_{\mathcal{R}}
\end{aligned}$$

Therefore, $\mathcal{R}$ is strictly reducible. □

Proposition 4.5 *Let $\mathcal{R}$ be an n-cell ($n > 4$) reversible CA where $\texttt{CS}_{\mathcal{R}} = [\mu(l)]$. If $\mu \ge 4$, then $\mathcal{R}$ is strictly reducible CA.*

Proof: Since all cycles of CA $\mathcal{R}$ has length l, l is of form 2^i, where $0 \leq i < n$ (the maximum length of cycle for an n-cell CA is $2^n - 1$ for $n > 3$). Let $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ be respectively 1-cell reversible CA and $(n-1)$-cell reversible CA with cycle structures $\texttt{CS}_{\mathcal{R}^{n_1}} = [2(1)]$ and $\texttt{CS}_{\mathcal{R}^{n_2}} = [\mu'(l)]$ where $\mu' = 2^{n-1-i}$. It is easy to verify that $\texttt{CS}_{\mathcal{R}} = \texttt{CS}_{\mathcal{R}^1} \times \texttt{CS}_{\mathcal{R}^2}$. Therefore, $\mathcal{R}$ is strictly reducible. $\square$

The above mentioned properties consider only those CAs whose cycle structures are to be computed by Equation 4.1 and we can detect a small number of n-cell strictly reducible CAs. In such cases, CA rules have no significance. Here, only the cycle structure is given where we need to figure out the nature of CA. We also intended to figure out some CAs whose cycle structures can not be satisfied by Equation 4.1.

Proposition 4.6 *Let $\mathcal{R}$ be an n-cell reversible CA. Let $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \mu_2(l_2), \cdots, 1(l_m)]$. If $l_m > 2^{n-1}$ and $l_m = 2^i \cdot l$ where $i > 3$ and l is a prime number, then $\mathcal{R}$ is not strictly reducible.*

Proof: Let if possible $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ be respectively n_1-cell reversible CA and n_2-cell reversible CA with cycle structures $\texttt{CS}_{\mathcal{R}^{n_1}}$ and $\texttt{CS}_{\mathcal{R}^{n_2}}$ such that $n = n_1 + n_2$ and $\texttt{CS}_{\mathcal{R}} = \texttt{CS}_{\mathcal{R}^1} \times \texttt{CS}_{\mathcal{R}^2}$. Without loss of generality, we can assume that the largest cycle in one of these two CAs, say $\mathcal{R}^{n_1}$, is of length 2^j for some j $(0 \leq j \leq i)$, while the largest cycle in the other CA, say $\mathcal{R}^{n_2}$, is of length $2^k l$, for some k $(0 \leq k \leq i)$. Then we require $\texttt{gcd}(2^j, 2^k \cdot l)$ to be 1; this implies at least one of j and k must be 0. However we also require $\texttt{lcm}(2^j, 2^k \cdot l)$ to be $2^i \cdot l$; thus $j = 0$ is not possible since this would imply $k = i$ contradicting the fact that an $n_2 < n$-cell CA can generate a cycle of length $l_m > 2^{n-1}$. Hence $k = 0$, and accordingly $j = i$ for $\texttt{lcm}(2^j, 2^k \cdot l)$ to be $2^i \cdot l$.

Again since $l_m > 2^{n-1}$, $l > 2^{n-1-i}$. This implies $n_2 \geq n - i$. Moreover, since $i > 3$, $n_1 \geq i + 1$. Hence $n_1 + n_2 \geq n + 1$, leading to a contradiction. $\square$

Proposition 4.7 *Let $\mathcal{R}$ be an n-cell reversible CA. Let $\texttt{CS}_{\mathcal{R}} = [\mu_1(l_1), \mu_2(l_2), \cdots, 1(l_m)]$. Let $l_m > 2^{n-1}$ and $l_m = 2^i \cdot l$ where $1 \leq i \leq 3$ and l is a prime number. If l_j is not divisible by 2^i for some j, then $\mathcal{R}$ is not strictly reducible CA.*

Proof: The proof follows from the proof of Proposition 4.6. Using a similar approach, we can show that if there exist two CAs $\mathcal{R}^{n_1}$ (of n_1 size) and $\mathcal{R}^{n_2}$ (of $n_2 = n - n_1$ size) such that $\texttt{CS}_{\mathcal{R}} = \texttt{CS}_{\mathcal{R}^1} \times \texttt{CS}_{\mathcal{R}^2}$, then the maximum length cycles of $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ are respectively 2^i and l. Since $l_m > 2^{n-1}$, $l > 2^{n-1-i}$. This implies $n_2 \geq n - i$, and so $n_1 \leq i$. Also, since $\mathcal{R}^{n_1}$ generates a cycle of length 2^i, $n_1 \geq i$. Hence, $n_1 = i$ and $\texttt{CS}_{\mathcal{R}^1} = [1(2^i)]$. Thus,

$$\texttt{CS}_{\mathcal{R}} = [1(2^i)] \times \texttt{CS}_{\mathcal{R}^2},$$

Table 4.1: A collection of 8-cell reversible cellular automata which represent strictly reducible CAs and not strictly reducible CAs both.

$\mathcal{R}$	$\texttt{CS}_{\mathcal{R}}$	Strictly Reducible	Remark
$\langle 9, 105, 180, 150, 150, 180, 54, 17\rangle$	$[1(126), 1(130)]$	Yes	Proposition 4.3
$\langle 9, 105, 165, 150, 101, 165, 149, 80\rangle$	$[1(24), 1(232)]$		
$\langle 5, 120, 105, 101, 90, 89, 150, 20\rangle$	$[1(30), 1(226)]$		
$\langle 5, 105, 150, 150, 105, 105, 105, 65\rangle$	$[1(1), 17(15)]$	Yes	Proposition 4.4
$\langle 5, 120, 60, 90, 30, 99, 166, 5\rangle$	$[4(64)]$	Yes	Proposition 4.5
$\langle 5, 90, 105, 165, 39, 169, 105, 65\rangle$	$[1(2), 3(10), 1(20), 2(22), 1(160)]$	No	Proposition 4.6
$\langle 5, 105, 90, 30, 101, 165, 106, 5\rangle$	$[1(6), 1(34), 1(40), 1(176)]$		
$\langle 10, 45, 165, 90, 165, 225, 105, 80\rangle$	$[1(16), 1(18), 1(30), 1(192)]$		
$\langle 5, 90, 165, 180, 150, 101, 150, 20\rangle$	$[1(1), 1(3), 1(6), 1(16), 1(22), 1(208)]$		
$\langle 10, 135, 169, 165, 89, 150, 150, 17\rangle$	$[1(4), 2(14), 1(224)]$		
$\langle 5, 90, 105, 141, 90, 90, 90, 5\rangle$	$[1(1), 1(3), 2(6), 1(7), 1(9), 1(30), 1(194)]$	No	Proposition 4.7
$\langle 5, 165, 105, 165, 89, 90, 86, 80\rangle$	$[1(4), 1(9), 1(17), 1(226)]$		
$\langle 5, 105, 90, 165, 165, 135, 83, 20\rangle$	$[1(1), 1(4), 1(117), 1(134)]$		

which implies that all cycle lengths in $\texttt{CS}_{\mathcal{R}}$ are divisible by 2^i, leading to a contradiction. □

Properties 4.6 and 4.7 represent a proper subset of n-cell non strictly reducible cellular automata. Besides, Properties 4.3, 4.6 and 4.7 consider n-cell CAs with maximum cycle length $l_m \geq 2^{n-1}$. Next we present in Table 4.1 few CAs which demonstrate the properties 4.3 to 4.7. As an instance, we have taken $n = 8$. The first column of the table consider the rule vector $\mathcal{R}$. The second column refers the cycle structure of $\mathcal{R}$. The third column shows whether the given CA is strictly reducible or not and the fourth column refers the corresponding property satisfied for validating the category of the CA.

The next question is if the cycle structure of CA can not be supported by Equation 4.1, what is the category of the CA? The answer depends on the decomposition of the cycle structure of that CA. If the cycle structure of the CA can be decomposed but can not be computed by Equation 4.1, then such CA is also reducible and called as *weakly reducible* (Definition 4.10). Though one way of detecting strictly reducible CAs is based on their rule vectors, but for weakly reducible CAs there is no such mechanism. Therefore, we concentrate on the valid cycle structures for figuring out the properties of weakly reducible cellular automata.

Proposition 4.8 *Let $\mathcal{R}$ be an n-cell reversible CA. Let $\texttt{CS}_{\mathcal{R}} = [1(l_1), 1(l_2)]$ where l_1 and l_2 are even numbers. If $\texttt{gcd}(l_1, l_2) = 2^i$ for some $i > 3$, then $\mathcal{R}$ is weakly reducible CA.*

Proof: We first show that $\mathcal{R}$ can't be a strictly reducible CA. Let if possible, there exist two CAs $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ of sizes n_1 and n_2 such that $n = n_1 + n_2$ and

$\mathtt{CS}_{\mathcal{R}} = \mathtt{CS}_{\mathcal{R}^{n_1}} \times \mathtt{CS}_{\mathcal{R}^{n_2}}$. Since cycle structure of $\mathcal{R}$ has only two cycles of distinct length, one of $\mathtt{CS}_{\mathcal{R}^{n_1}}$ and $\mathtt{CS}_{\mathcal{R}^{n_2}}$ must have two cycles of different lengths, while the other must have a single cycle. Without loss of generality, let us assume $\mathtt{CS}_{\mathcal{R}^{n_1}} = [1(l_1^1), 1(l_2^1)]$ and $\mathtt{CS}_{\mathcal{R}^{n_2}} = [1(2^l)]$, for some l. Since each cycle of $\mathtt{CS}_{\mathcal{R}}$ has distinct length, $\mathtt{gcd}(l_1^1, 2^l) = \mathtt{gcd}(l_2^1, 2^l) = 1$. This implies both l_1^1 and l_2^1 are odd. Moreover, $l_1 = \mathtt{lcm}(l_1^1, 2^l) = 2^l \cdot l_1^1$, and $l_2 = \mathtt{lcm}(l_2^1, 2^l) = 2^l \cdot l_2^1$. Thus, $\mathtt{gcd}(l_1, l_2) = 2^l$, which implies that l must equal i. But since $i > 3$, $n_2 = i + 1$. Again we have $l_1 + l_2 = 2^n$; thus $l_1^1 + l_2^1 = 2^{n-i}$ which implies that n_1 must be at least $n - i$, leading to a contradiction.

We now proceed to prove that $\mathcal{R}$ is weakly reducible. Let $\mathcal{R}^{n_1}$ be a n_1-cell CA such that $\mathtt{CS}_{\mathcal{R}^{n_1}} = [1(l')]$, where $n_1 = 2$ and $l' = 4$ (we can also set $n_1 = 3$ and $l' = 8$). It is easy to verify that the properties of decomposition of CS hold for $\mathtt{CS}_{\mathcal{R}}$, namely, (i) each cycle length of $\mathtt{CS}_{\mathcal{R}}$ is a multiple of l' and (ii) sum of the cycle lengths of $\mathtt{CS}_{\mathcal{R}}$ is $l' \cdot 2^{n-n_1}$. Therefore, $\mathcal{R}$ is weakly reducible. □

Proposition 4.9 *Let $\mathcal{R}$ be an n-cell reversible CA. Let $\mathtt{CS}_{\mathcal{R}} = [\mu(l)]$. If $\mu = 2$ and $n > 4$, then $\mathcal{R}$ is weakly reducible CA.*

Proof: Since $\mu = 2$, $l = 2^{n-1}$. We first show that $\mathcal{R}$ can't be a strictly reducible CA. Let if possible, there exist two CAs $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ of sizes n_1 and n_2 such that $n = n_1 + n_2$ and $\mathtt{CS}_{\mathcal{R}} = \mathtt{CS}_{\mathcal{R}^{n_1}} \times \mathtt{CS}_{\mathcal{R}^{n_2}}$. Since cycle structure of $\mathcal{R}$ has cycles of unique length, each of $\mathtt{CS}_{\mathcal{R}^{n_1}}$ and $\mathtt{CS}_{\mathcal{R}^{n_2}}$ shall also have cycle(s) of unique length. Let $\mathtt{CS}_{\mathcal{R}^{n_1}} = [\mu_1^1(l_1^1)]$ and $\mathtt{CS}_{\mathcal{R}^{n_2}} = [\mu_1^2(l_1^2)]$. Then $l = \mathtt{lcm}(l_1^1, l_1^2)$ and $\mu = \mu_1^1 \cdot \mu_1^2 \cdot \mathtt{gcd}(l_1^1, l_1^2)$. This can only happen when one of l_1^1 and l_1^2 is 2^{n-1}, while the other is 1 or 2. Without loss of generality, let us assume $l_1^1 = 2^{n-1}$. However, the only situation when a CA of size $n_1 < n$ can generate cycles of length 2^{n-1} corresponds to $n_1 = n - 1$ and $n_1 \leq 3$. This implies $n \leq 4$ leading to a contradiction. Thus $\mathcal{R}$ can't be a strictly reducible CA.

We now proceed to prove that $\mathcal{R}$ is weakly reducible. Let $\mathcal{R}^{n_1}$ be a n_1-cell CA such that $\mathtt{CS}_{\mathcal{R}^{n_1}} = [1(l')]$, where $n_1 = 2$ and $l' = 4$ (we can also set $n_1 = 3$ and $l' = 8$). It is easy to verify that the properties of decomposition of CS hold for $\mathtt{CS}_{\mathcal{R}}$, namely, (i) each cycle length of $\mathtt{CS}_{\mathcal{R}}$ is a multiple of l' and (ii) sum of the cycle lengths of $\mathtt{CS}_{\mathcal{R}}$ is $l'.2^{n-n_1}$. Therefore, $\mathcal{R}$ is weakly reducible. □

Table 4.2 represents some of the weakly reducible cellular automata. Every row of the table described as follows. The first and second columns refer the rule vector of an 8-cell CA and its cycle structure respectively; the third column mentions the property which validates that the corresponding CA of the row is weakly reducible. Though the above mentioned properties cover a subset of weakly reducible CAs, similar to strictly reducible CA, there does not exist any mechanism which can decide that $\mathtt{CS}_{\mathcal{R}}$ can not be satisfied by Equation 4.1. It is already mentioned that for a reducible CA, either the rule vector of the

Table 4.2: Some 8-cell weakly reducible automata

$\mathcal{R}$	$\texttt{CS}_\mathcal{R}$	Remark
$\langle 5, 120, 90, 225, 165, 150, 54, 17\rangle$	$[1(80), 1(176)]$	Proposition 4.8
$\langle 5, 90, 90, 30, 165, 30, 165, 20\rangle$	$[1(48), 1(208)]$	
$\langle 5, 225, 195, 99, 166, 90, 149, 80\rangle$	$[1(16), 1(240)]$	
$\langle 5, 120, 105, 89, 90, 169, 105, 20\rangle$	$[2(128)]$	Proposition 4.9

CA itself or the rule vector of its isomorphic CA contains at least one right or left independent rule; but there is no such rule in case of **irreducible** CA (Definition 4.11). Our next focus is on irreducible CAs.

4.3.2 Irreducible Cellular Automata

Now we present the behaviour of the irreducible CAs whose cycle structures can not be decomposed. As a reducible CA has no quasi isomorphic CA, therefore, all ten right and left independent rules are not considered in designing the rule vectors of n-cell irreducible CAs and their isomorphic CAs also (for cell positions 1 to $n-2$). Similarly, no right or left independent rule is to be selected for cell 0 and cell $n-1$ in the design of the rule vectors for the CAs under considerations. It is already seen that some CAs are strictly or weakly reducible even though their rule vectors contain neither right or left independent rules (for reference, see rows 3, 4 and 5 of Table 4.1 and row 3 and 5 of Table 4.2). Therefore, the arrangement of CA rules in rule vector do not decide whether the given CA is irreducible and we are motivated to figure out some intrinsic properties of cycle structure based on its either cycle lengths or number of cycles or cyclic components similar approaches taken in considerations for strictly and weakly reducible cellular automata. The following properties lead to figure out irreducible cellular automata.

Proposition 4.10 *Let $\mathcal{R}$ be an n-cell reversible CA. Let us consider $\texttt{CS}_\mathcal{R} = [\mu_1(l_1), \mu_2(l_2), \cdots 1(l_m)]$ be the cycle structure of $\mathcal{R}$. If $l_m > 2^{n-1}$ and l_m is a prime number, then $\mathcal{R}$ is an irreducible CA.*

Proof: We prove the lemma by contradiction. Suppose there exists $\mathcal{R}^{n_1}$ (a CA of size $n_1 < n$) such that $\texttt{CS}_\mathcal{R}$ can be decomposed to $\texttt{CS}^{n_1}_\mathcal{R}$. Since $l_m > 2^{n-1}$, length of every cycle of $\mathcal{R}^{n_1}$ is strictly less than l_m. Furthermore, l_m must be a (proper) multiple of some cycle length of $\mathcal{R}^{n_1}$. This leads to a contradiction since l_m is prime. So, $\mathcal{R}$ is an irreducible CA. $\square$

Proposition 4.11 *Let $\mathcal{R}$ be an n-cell reversible CA. Let the cycle structure of $\mathcal{R}$ be $\texttt{CS}_\mathcal{R} = [1(1), 1(l)]$, where $l = 2^n - 1$. Then $\mathcal{R}$ is an irreducible CA.*

Table 4.3: A collection of 8-cell irreducible cellular automata

$\mathcal{R}$	$\texttt{CS}_{\mathcal{R}}$	Remark
$\langle 5, 90, 165, 165, 165, 225, 90, 20\rangle$	$[1(1), 1(3), 1(5), 1(45), 1(71), 1(131)]$	Proposition 4.10
$\langle 5, 105, 90, 225, 165, 45, 86, 80\rangle$	$[1(1), 2(4), 2(55), 1(137)]$	
$\langle 5, 90, 150, 105, 165, 201, 86, 5\rangle$	$[2(1), 1(4), 1(13), 2(31), 1(36), 1(139)]$	
$\langle 5, 90, 165, 210, 150, 90, 169, 5\rangle$	$[1(6), 2(7), 1(9), 1(78), 1(149)]$	
$\langle 5, 90, 105, 90, 165, 114, 105, 5\rangle$	$[1(1), 1(7), 1(97), 1(151)]$	
$\langle 5, 90, 105, 228, 165, 165, 89, 80\rangle$	$[1(1), 1(3), 1(5), 2(7), 1(9), 1(67), 1(157)]$	
$\langle 5, 90, 150, 154, 150, 105, 149, 80\rangle$	$[1(93), 1(163)]$	
$\langle 5, 90, 147, 150, 150, 165, 90, 65\rangle$	$[1(1), 1(12), 1(19), 1(28), 1(29), 1(167)]$	
$\langle 5, 90, 105, 150, 150, 156, 101, 5\rangle$	$[2(1), 1(19), 2(31), 1(173)]$	
$\langle 5, 90, 105, 165, 165, 178, 90, 65\rangle$	$[1(1), 1(26), 1(50), 1(179)]$	
$\langle 5, 90, 147, 89, 105, 150, 89, 80\rangle$	$[1(17), 1(27), 1(31), 1(181)]$	
$\langle 5, 105, 90, 150, 147, 149, 105, 65\rangle$	$[1(2), 1(30), 1(33), 1(191)]$	
$\langle 5, 90, 54, 101, 105, 165, 165, 20\rangle$	$[1(1), 2(4), 1(13), 1(41), 1(193)]$	
$\langle 5, 105, 92, 105, 101, 90, 101, 80\rangle$	$[1(3), 1(6), 1(9), 1(12), 1(29), 1(197)]$	
$\langle 5, 90, 150, 90, 149, 150, 90, 65\rangle$	$[1(1), 1(6), 1(9), 1(13), 1(28), 1(199)]$	
$\langle 5, 90, 165, 210, 54, 150, 105, 65\rangle$	$[1(1), 1(3), 2(8), 1(25), 1(211)]$	
$\langle 5, 90, 165, 150, 147, 154, 105, 20\rangle$	$[1(33), 1(223)]$	
$\langle 5, 105, 90, 45, 57, 105, 150, 20\rangle$	$[1(1), 2(6), 1(7), 1(9), 1(227)]$	
$\langle 5, 90, 150, 150, 225, 150, 105, 80\rangle$	$[1(1), 1(7), 1(19), 1(229)]$	
$\langle 5, 90, 165, 90, 201, 165, 105, 20\rangle$	$[1(1), 1(3), 2(6), 1(7), 1(233)]$	
$\langle 5, 90, 105, 150, 135, 163, 90, 65\rangle$	$[1(1), 1(5), 1(11), 1(239)]$	
$\langle 5, 90, 150, 150, 180, 90, 75, 20\rangle$	$[2(1), 1(13), 1(241)]$	
$\langle 5, 90, 163, 165, 90, 90, 165, 20\rangle$	$[1(5), 1(251)]$	
$\langle 5, 90, 150, 150, 135, 169, 150, 20\rangle$	$[1(1), 1(255)]$	Proposition 4.11
$\langle 5, 90, 105, 165, 150, 75, 163, 65\rangle$	$[1(3), 1(25), 1(43), 1(185)]$	Proposition 4.13(ii)
$\langle 5, 90, 83, 150, 150, 90, 149, 80\rangle$	$[1(3), 1(28), 1(38), 1(187)]$	
$\langle 5, 90, 150, 150, 225, 154, 105, 20\rangle$	$[2(4), 1(47), 1(201)]$	
$\langle 5, 90, 83, 150, 89, 90, 106, 5\rangle$	$[1(4), 2(7), 1(12), 1(17), 1(209)]$	
$\langle 5, 90, 105, 177, 149, 165, 90, 80\rangle$	$[1(5), 1(6), 1(32), 1(213)]$	
$\langle 5, 90, 165, 225, 90, 225, 154, 5\rangle$	$[1(6), 1(10), 1(25), 1(215)]$	
$\langle 5, 90, 163, 150, 166, 165, 90, 80\rangle$	$[1(4), 1(6), 1(7), 1(8), 1(12), 1(219)]$	
$\langle 5, 90, 165, 225, 58, 150, 105, 80\rangle$	$[1(2), 1(14), 1(19), 1(221)]$	
$\langle 5, 90, 150, 228, 165, 165, 106, 5\rangle$	$[1(1), 1(3), 2(5), 1(7), 1(235)]$	
$\langle 5, 150, 165, 180, 166, 90, 165, 80\rangle$	$[1(2), 1(3), 1(14), 1(237)]$	
$\langle 5, 90, 165, 180, 150, 150, 120, 65\rangle$	$[1(1), 1(3), 1(5), 1(247)]$	
$\langle 5, 105, 166, 165, 169, 90, 169, 5\rangle$	$[1(7), 1(249)]$	
$\langle 5, 90, 54, 154, 90, 150, 150, 20\rangle$	$[1(3), 1(253)]$	
$\langle 5, 5, 90, 165, 45, 165, 105, 165, 65\rangle$	$[1(1), 1(13), 1(29), 1(32), 1(52), 1(129)]$	Proposition 4.13(i)
$\langle 5, 210, 165, 210, 90, 30, 90, 80\rangle$	$[1(8), 1(9), 1(10), 2(11), 1(13), 1(61), 1(133)]$	
$\langle 5, 225, 169, 165, 90, 165, 149, 80\rangle$	$[1(20), 1(23), 1(72), 1(141)]$	
$\langle 5, 90, 105, 105, 225, 54, 86, 5\rangle$	$[1(1), 1(4), 1(7), 3(20), 1(41), 1(143)]$	
$\langle 5, 120, 150, 165, 23, 165, 90, 20\rangle$	$[1(111), 1(145)]$	
$\langle 5, 90, 165, 180, 197, 105, 90, 80\rangle$	$[1(1), 1(3), 2(6), 1(7), 1(17), 1(61), 1(155)]$	
$\langle 5, 90, 105, 150, 180, 163, 90, 65\rangle$	$[1(1), 1(7), 1(89), 1(159)]$	
$\langle 5, 30, 165, 135, 150, 169, 105, 20\rangle$	$[1(1), 1(5), 2(15), 1(59), 1(161)]$	
$\langle 5, 90, 108, 105, 90, 90, 105, 20\rangle$	$[2(1), 1(2), 1(4), 3(8), 1(16), 1(31), 1(177)]$	
$\langle 5, 225, 150, 90, 86, 90, 105, 5\rangle$	$[1(1), 1(5), 1(67), 1(183)]$	
$\langle 5, 90, 90, 75, 99, 86, 105, 20\rangle$	$[1(1), 2(2), 4(6), 2(12), 1(203)]$	
$\langle 5, 90, 105, 165, 165, 150, 135, 20\rangle$	$[2(1), 2(7), 1(35), 1(205)]$	
$\langle 5, 90, 90, 30, 105, 27, 90, 5\rangle$	$[1(3), 2(4), 1(13), 1(15), 1(217),]$	
$\langle 5, 90, 150, 150, 135, 165, 30, 65\rangle$	$[1(10), 1(18), 1(59), 1(169)]$	Proposition 4.12
$\langle 5, 225, 165, 210, 57, 90, 150, 20\rangle$	$[1(13), 1(243)]$	Proposition 4.12, Proposition 4.14

Proof: We prove this lemma also by contradiction. Suppose there exists $\mathcal{R}^{n_1}$ (a CA of size $n_1 < n$) such that $\texttt{CS}_{\mathcal{R}}$ can be decomposed to $\texttt{CS}_{\mathcal{R}}^{n_1}$. Thus, l is a multiple of l' (where $l' < l$), and l' is length of some cycle in $\mathcal{R}^{n_1}$. Now $l' \leq 2^{n_1} - 1$ (since $l' = 2^{n_1}$ would imply l to be a multiple of 2^{n_1} leading to a contradiction). Moreover, $l \leq l' \cdot 2^{n-n_1}$, which implies $l \leq 2^n - 2^{n-n_1}$, which leads to the contradiction. So, $\mathcal{R}$ is an irreducible CA. $\square$

Proposition 4.12 *Let $\mathcal{R}$ be an n-cell reversible CA. Let $\texttt{CS}_\mathcal{R} = [\mu_1(l_1), \mu_2(l_2), \cdots 1(l_m)]$ be the cycle structure of $\mathcal{R}$ such that $l_m > 2^{n-1}$ and $l_m = p^q$, where $p > 2$ is a prime number and q is a positive integer. If there does not exist any l_j $(1 \leq j \leq m-1)$ which is multiple of p, then $\mathcal{R}$ is an irreducible CA.*

Proof: If $q = 1$, the proof follows from Proposition 4.10. Otherwise, $q \geq 2$. Let $\mathcal{R}$ can be reduced to n_1-cell CA $\mathcal{R}^{n_1}$ and $\texttt{CS}_{\mathcal{R}^{n_1}} = [\mu^1_1(l^1_1), \mu^1_2(l^1_2), \cdots \mu^1_{m_1}(l^1_{m_1})]$. Thus l_m is a multiple of some l^1_i, where $l^1_i < 2^{n_1} \leq 2^{n-1}$. Hence l^1_i must be of form p^r where $r < q$. Now, since none of the l_js $(1 \leq j \leq m-1)$ is multiple of p, $p^r \cdot 2^{n-n_1}$ must equal l_m, which is p^q. This is impossible since $p > 2$. So, $\mathcal{R}$ is irreducible. $\square$

Proposition 4.13 *Let $\mathcal{R}$ be an n-cell reversible CA. Let $\texttt{CS}_\mathcal{R}$ be the cycle structure of $\mathcal{R}$ such that $\texttt{CS}_\mathcal{R} = [\mu_1(l_1), \mu_2(l_2), \cdots 1(l_m)]$ where $l_m > 2^{n-1}$. Let $l_m = p_1 \cdot p_2$, where p_1 and p_2 are prime numbers. Then $\mathcal{R}$ is an irreducible CA if either of the following conditions holds: (i) none of p_1 or p_2 divides any other l_j $(1 \leq j \leq m-1)$ (ii) $\lceil \log_2 p_1 \rceil + \lceil \log_2 p_2 \rceil > n$.*

Proof: Let $\mathcal{R}$ be reduced to $\mathcal{R}^{n_1}$ (a CA of size n_1). Since there is one cycle with the largest length of $p_1 \cdot p_2 > 2^{n-1}$ in $\texttt{CS}_\mathcal{R}$, $\texttt{CS}_{\mathcal{R}^{n_1}}$ should have cycle of length either p_1 or p_2; without loss of generality, let there be a cycle of length p_1 in $\texttt{CS}_{\mathcal{R}^{n_1}}$. To prove the first part, note that only the cycle of length $p_1 \cdot p_2$ in $\texttt{CS}_\mathcal{R}$ is a multiple of this cycle length in $\texttt{CS}_{\mathcal{R}^{n_1}}$. From the second property of decomposition, the length of this cycle should be $p_1 \cdot 2^i$ for some $i > 1$. This leads to a contradiction since 2^i can never equal a prime number p_2.

To prove the second part, we see that $n_1 \geq \lceil \log_2 p_1 \rceil$. Again from the second property of decomposition, the sum of lengths of all cycles in $\texttt{CS}_\mathcal{R}$ that are multiple of the p_1 length cycle of $\texttt{CS}_{\mathcal{R}^{n_1}}$ is $p_1 2^{n-n_1}$. Hence $l_m \leq p_1 2^{n-n_1}$ which implies $p_2 \leq 2^{n-n_1}$. This requires n to be greater than or equal to $n_1 + \lceil \log_2 p_2 \rceil$, leading to a contradiction. $\square$

Proposition 4.14 *Let $\mathcal{R}$ be an n-cell reversible CA. If $\texttt{CS}_\mathcal{R} = [1(l_1), 1(l_2)]$ where l_1 and l_2 are odd numbers, then $\mathcal{R}$ is an irreducible CA.*

Proof: We prove this lemma by contradiction. Suppose there exists $\mathcal{R}^{n_1}$ (a CA of size $n_1 < n$) such that $\texttt{CS}_\mathcal{R}$ can be decomposed to $\texttt{CS}^{n_1}_\mathcal{R}$. Since $\texttt{CS}_\mathcal{R}$ contains two cycles of distinct lengths, $\texttt{CS}^{n_1}_\mathcal{R}$ shall also have only two cycles of different lengths. Thus $\mathtt{CS}^{n_1}_{\mathcal{R}} = [1(l^1_1), 1(l^1_2)]$. Without loss of generality, we can assume that l_1 is a multiple of l^1_1 and l_2 is a multiple of l^1_2. Furthermore, from the second property of decomposition, we have $l_1 = 2^{n-n_1}l^1_1$ and $l_2 = 2^{n-n_1}l^1_2$. This implies both l_1 and l_2 are even, leading to a contradiction. □

The properties 4.10 to 4.14 are used to cover a wide range of n-cell irreducible cellular automata where the maximum cycle length is larger than $2^n - 1$. As an instance, we take the CAs of a particular CA size $n = 8$. Table 4.3 refers some irreducible CAs of size 8 where the maximum cycle length is larger than 128. The first and the second columns of the table represent 8-cell irreducible CAs and their cycle structures. The third column refers the properties of which is satisfied by those irreducible cellular automata.

4.4 Summary

In this chapter, we have studied the property of decomposition of the cycle structures of reversible cellular automata. This property classifies the finite reversible CA family into two broad categories - reducible and irreducible. A large section of reversible CAs are reducible and they can be recognized by the presence of left or right independent rules. The remaining section where no such left or right independent rule is present in the rule vectors of the CAs, the CAs are also either reducible or irreducible. The reducible CAs are categorized into strictly reducible cellular automata and weakly reducible cellular automata. The equal length cycle cellular automata of size larger than four are either strictly reducible or weakly reducible CAs whereas the maximal length CAs are irreducible. The irreducible CAs are also attractive as such CAs can generate cycles where the corresponding lengths are prime numbers, power of prime numbers, multiple of prime numbers.

CHAPTER 5

Reversible Cellular Automata with Large Cycles and their Applications

In the journey of studying the cyclic properties of reversible cellular automata, Chapter 3 proposed an effective mechanism for finding the cycle structure of an arbitrary n-cell reversible CA and in the previous chapter, the fundamental properties of cycle structures have been explored which categorize the CAs broadly into two classes - reducible and irreducible. It is already found from the previous discussions that n-cell reversible CAs can produce cycles of lengths greater than 2^{n-1}. Such observation motivates us to study another problem - can we obtain a cycle of an arbitrary length l by a reversible CA of size n where $n = O(\log_2 l)$? This chapter initially focuses on the design of n-cell reversible CA which can produce l-cycle where n is "near optimal". Such n-cell CA can be called as *large length cycle* CA. As large *period* (cycle) length and randomness are the key points for the designing issues of good Pseudo Random Number Generators (PRNGs), this chapter discusses the contributions of non-linear large length cycle CAs in PRNGs. This chapter also studies an application of irreducible large length cycle CAs in number theory: whether a collection n-cell CAs can generate all possible prime numbers between $2^{n-1}+1$ to 2^n-1.

5.1 Introduction

In case of a finite cellular automaton, the lengths of cycles are finite and depend on local rule. For non-uniform cellular automata, where different cells may follow different rules, the lengths of cycles depend on two things - (1) the local rules and (2) arrangement of local rules. This has been a theoretical challenge since late 1980s to the CA community to intelligently arrange some rules to get a maximum possible cycle length for an n-cell non-uniform CA.

Prior work have addressed this problem by considering the *maximal length* CAs. Using maximal length CA we can get cycle of length $2^n - 1$ where the size of the CA is n ($n > 3$). Therefore, the maximal length CA of size $\lfloor \log_2 l \rfloor + 1$ is considered as the minimum (optimal) sized CA for generating a cycle of length l. It has already been seen that maximal length CAs are mainly linear CAs and generating maximal length CAs is related to primitive polynomials; this implies that there exists an n-cell binary maximal length CA iff the characteristic polynomial of the CA is primitive (over $GF(2)$), see [44, 65]. Now, these linear maximal-length CAs have some serious drawbacks. Firstly, Linear maximal length sequences induce some security related issues. Besides, there is a limited number of n-degree primitive polynomials available till date [58,66,360]. Moreover, the complexity of generating the primitive polynomial is exponential. Given this fact, one may be interested to get a CA with cycle of much larger length. However, with non-linear rules, the tools of matrix algebra and characteristics polynomial can no longer be applied to study the cyclic behaviour of the CAs. The only option known till date to get the cycle lengths of such a CA is to exhaustively search for it. This theoretical challenge makes the problem non-trivial. Hence, given the inherent hardness, we can only resort to finding near optimal solution for the problem, with the objective that the *displacement* (difference from the optimal solution) of the attained near optimal solution should be as small as possible. Therefore, in this chapter, our first goal is to deal with the following problem:

- *Given any $l \in \mathbb{N}$, find a binary n-cell reversible CA that generates a cycle of length at least l and n is as small as possible.*

Since there is no deterministic method known to solve the aforementioned problem, we explore the properties of these CAs which can lead to large length cycle(s) in their configuration space. We begin with a stochastic approach to address this problem. The idea is as follows: we first classify the CA rules based on some *parameters*, which determine the dependence of the next state of a cell on the present states of its neighbors. Our hypothesis is that if the rules of a CA possess higher parameter value, then the CA is expected to have larger length cycle. Thus a CA constructed by selecting the rules randomly by assigning more priority to the rules that have higher values of these parameters is expected to generate large length cycles. Algorithm 2 (mentioned in Section 3.5 on page 81) assures the presence of a cycle of length l using low computation cost; however, it works for some special rule vectors which possess left and right independent rules at consecutive positions. Therefore, in worst case, we need a brute force method to know the lengths of cycles of the CA. This implies, for given l, where $\lfloor \log_2 l \rfloor + 1$ is large, obtaining of CA using the above approach is infeasible in practice.

As a solution to this issue, we apply the generic operator $\odot$ (already been introduced in Section 4.3.1 of Chapter 4) which takes two CAs of cycle lengths l_i^1 and l_j^2, and produces a CA with a cycle of length $\texttt{lcm}(l_i^1, l_j^2)$. Our approach thus is to first synthesize small sized CAs in stochastic manner, and test if these CAs really generate large length cycle (since these CAs are of small sizes, testing whether they generate large cycles or not is computationally feasible). Next we apply the operator on these already synthesized small sized CAs to generate our desired large sized CA. The solution obtained by the above approach works fine for many input cases. For the scenarios where it has large displacement, the solution is further improved by employing an evolutionary strategy which conforms some divergent of genetic algorithm. The notable change in our strategy with respect to genetic algorithm is that an off-spring is not produced by the selected parents using genetic operators like crossover or mutation. Here, the creation of an off-spring is influenced by multiple parents in such a way that the off-spring owns the *significant* features of the selected parents.

Due to the inherent simplicity of the cellular automata, CAs of finite size have been widely explored as technologies including Test Pattern Generator for VLSI circuits, Pseudo-random Number generators (PRNGs), CAs based cryptosystems, Encompression etc., [44]. Many of these technologies, however, demand CAs which are reversible and have large length cycles. For a good PRNG, large length cycle is one of the essential criteria. Therefore, the large length cycle cellular automata which are generated by the aforementioned process can be the good candidates for the design of PRNGs. Besides large period length, another key requirement for good PRNG is randomness quality. Therefore, the evolution of the CA needs to be unpredictable, that is, one can not predict the future configurations of the CA from its present configuration, if the rule vector is not known. Hence, the stochastically synthesized large length cycle CAs are good choice for designing effective PRNGs. These CAs are not only generating large length cycles, but also possesses more randomness in the cycle as more priority is given to the rules that have higher values of the parameters which means these rules are highly influenced by their neighbors and the next configuration is rarely predictable from the current configuration. Moreover, we get a wide range of non-linear large length cycles cellular automata. Therefore, the second target of this chapter is mentioned as follows.

- *Can the non-linear large length cycle reversible cellular automata be applied as window-based pseudo-random number generators for obtaining better performance?*

In number theory, one of the most fascinating areas is the study of prime number due to its intrinsic difficulty. The chaotic behaviour and self organizing property enable cellular automata (CAs) to mathematically model natural phenomena. When $2^n - 1$ is prime, then the corresponding maximal length CA can be used as a special category of *prime* generator - *Mersenne* prime generator. This property of CA steers us to think whether CA is capable of generating a cycle of a prime length other than Mersenne prime. Moreover, from the experimental results of Chapter 4 (see Table 4.3 on page 108), it is found that a collection of *irreducible* cellular automata (see Definition 4.3.2 on 107 of Section 4.3) generate large length cycles and the corresponding cycle lengths are prime numbers. This motivates us to investigate a very challenging and hard problem of whether an n-cell reversible CA can generate a cycle of an arbitrary prime length l.

Therefore, the third and final problem statement of this chapter can be framed as follows.

- *For any prime $l < 2^n$, does there exist an n-cell cellular automaton which can generate a cycle of length l?*

Next, we start our expedition with first problem statement - generation of large length cycle cellular automata.

5.2 Reversible Cellular Automata with Large Length Cycles

As stated in the introduction, the first goal of this chapter is finding the *optimal* cellular automaton which generates the cycle of a given length. A CA is said to be *optimal* if it can generate a cycle of length l and the corresponding CA size is $\lfloor \log_2 l \rfloor + \delta$ where δ $(\in \mathbb{N})$ is as minimum as possible. Such CA is also called as *large length cycle* CA when $\delta = 1$.

Definition 5.1 *An n-cell CA is said to be large length cycle CA if the largest cycle length is greater than 2^{n-1}.*

Irreversible CAs in general produce a large number of acyclic configurations - the number of acyclic configurations in an irreversible CAs is exponential (already mentioned in [45]). This motivates us to employ reversible CAs in our work. As there is no deterministic process to design reversible CA which has a large cycle, we, therefore, develop a stochastic scheme of designing reversible CA which is expected to have large cycle.

We first present the intuitive notion behind our approach. In a 3-neighborhood CA, the next state of a cell depends on the present states of the cell and its neighbors; nevertheless, for some rules, this dependency may be limited. For example, the rules of a CA may be such that the next state of every cell depends only on its present state, and is totally independent of the present state of its neighbours. It is not difficult to show that every cycle of such CAs is of small length (if $f_i(x_{i-1}, x_i, x_{i+1}) = x_i$ for all i, then every cycle is of length one; whereas, if $f_i(x_{i-1}, x_i, x_{i+1}) = 1 - x_i$ for all i, every cycle is of length two). This observation motivates us to study the cycle structure of a CA based on the dependency of the next state function on the present state of the neighbors of a cell. To this effect, we quantify the extent to which the next state of a cell is *influenced* by the state of its neighbors, with the help of some parameters, which we define next.

5.2.1 Parameters for Measuring Dependence of a Cell on its Neighbors

Let $(x_0x_1 \cdots x_i \cdots x_{n-1})$ denote a configuration of a CA, where $\mathcal{R}_i(x_{i-1}, x_i, x_{i+1}) = \mathcal{R}_i(x_{i-1}, x_i, 1 - x_{i+1})$, for any of x_i and x_{i-1}. This means, the present state of cell $i + 1$ can not influence the next state of cell i, that is, cell i is *independent* of its right neighbor. In the similar way, if for all x_i, x_{i+1}, $\mathcal{R}_i(x_{i-1}, x_i, x_{i+1}) = \mathcal{R}_i(1 - x_{i-1}, x_i, x_{i+1})$, it indicates that cell i is independent of its left neighbor. Let us now define the *degree of dependence* on the neighbor of a cell:
Let $\alpha_{rd}(x_{i-1} = \mathtt{x}, x_i = \mathtt{y})$ be the dependence of cell i on its right neighbor when the present states of x_{i-1} and x_i are $\mathtt{x}$ and $\mathtt{y}$ respectively. Here, $\mathtt{x}, \mathtt{y} \in \{0, 1\}$.

Equation 5.1 $$\alpha_{rd}(x_{i-1} = \mathtt{x}, x_i = \mathtt{y}) = \begin{cases} 1 & \textit{if } \mathcal{R}_i(\mathtt{x}, \mathtt{y}, x_{i+1}) \neq \mathcal{R}_i(\mathtt{x}, \mathtt{y}, 1 - x_{i+1}) \\ 0 & \textit{otherwise} \end{cases}$$

For this rule $\mathcal{R}_i$, the parameter *degree of right dependence* measures the dependence of any cell on its neighbors. It is defined as the ratio of the number of combinations of values of x_i and x_{i-1} for which the transition function on x_i depends on x_{i-1} and is denoted by $\mathtt{P_r}(\mathcal{R}_i)$. Formally,

$$\mathtt{P_r}(\mathcal{R}_i) = \frac{\sum_{\mathtt{x} \in \{0,1\}} \sum_{\mathtt{y} \in \{0,1\}} \alpha_{rd}(x_{i-1} = \mathtt{x}, x_i = \mathtt{y})}{4}$$

Clearly, $\mathtt{P_r}(\mathcal{R}_i)$ can take values 0, 0.5 or 1 (since balanced rules are considered). Similarly, let the present states of x_i and x_{i+1} are $\mathtt{x}$ and $\mathtt{y}$ respectively and the dependence of cell i on its left neighbor is denoted by $\alpha_{ld}(x_i = \mathtt{x}, x_{i+1} = \mathtt{y})$.

Table 5.1: Categories of reversible CA rules on parameter $\mathtt{P_r}$

Category	$\mathcal{R}_i$
Completely Right Dependent	102, 153, 85, 170, 90, 165, 150, 105, 86, 89, 101, 106, 149, 154, 166, 169
Partially Right Dependent	53, 58, 83, 92, 163, 172, 197, 202, 54, 57, 99, 108, 147, 156, 198, 201, 23, 43, 77, 113, 142, 178, 212, 232, 27, 39, 78, 114, 141, 177, 216, 228, 30, 45, 75, 120, 135, 180, 210, 225
Right Independent	51, 204, 60, 195, 15, 240

Equation 5.2 $\alpha_{ld}(x_i = \mathtt{x}, x_{i+1} = \mathtt{y}) = \begin{cases} 1 & \textit{if}\ \mathcal{R}_i(x_{i-1}, \mathtt{x}, \mathtt{y}) \neq \mathcal{R}_i(1 - x_{i-1}, \mathtt{x}, \mathtt{y}) \\ 0 & \textit{otherwise} \end{cases}$

Now, the parameter $\mathtt{P_l}(\mathcal{R}_i)$, called the *degree of left dependence* for rule $\mathcal{R}_i$ determines the dependence of a cell i on its left neighbor. It is the ratio of the number of combinations of values of x_i and x_{i+1} for which the transition function on x_i depends on x_{i+1}.

$$\mathtt{P_l}(\mathcal{R}_i) = \frac{\sum_{\mathtt{x}\in\{0,1\}} \sum_{\mathtt{y}\in\{0,1\}} \alpha_{ld}(x_i = \mathtt{x}, x_{i+1} = \mathtt{y})}{4}$$

Example 5.1 Let us consider the rule 54. It can be observed that for this rule, $\alpha_{rd}(x_{i-1} = \mathtt{0}, x_i = \mathtt{0}) = 1$, $\alpha_{rd}(x_{i-1} = \mathtt{0}, x_i = \mathtt{1}) = 1$, while we have $\alpha_{rd}(x_{i-1} = \mathtt{1}, x_i = \mathtt{0}) = 0$, $\alpha_{rd}(x_{i-1} = \mathtt{1}, x_i = \mathtt{1}) = 0$. Therefore, $\mathtt{P_r}(\mathtt{54})$ is 0.5. On the other hand, $\alpha_{ld}(x_i = \mathtt{0}, x_{i+1} = \mathtt{0}) = 1$, $\alpha_{ld}(x_i = \mathtt{1}, x_{i+1} = \mathtt{0}) = 1$, while $\alpha_{ld}(x_i = \mathtt{1}, x_{i+1} = \mathtt{1}) = 0, \alpha_{ld}(x_i = \mathtt{1}, x_{i+1} = \mathtt{0}) = 0$. Therefore, $\mathtt{P_l}(54)$ is 0.5. Similarly, for rules 90 and 60, $\mathtt{P_r}(90) = 1$ and $\mathtt{P_r}(60) = 0$. For rules 150 and 170, we get $\mathtt{P_l}(150) = 1$ and $\mathtt{P_l}(170) = 0$.

Therefore, depending on the parameter $\mathtt{P_r}$, we can classify the rules of reversible CAs into three categories – *completely right dependent*, *partially right dependent* and *right independent* corresponding to right dependence degree values of 1, 0.5 and 0 respectively (see Table 5.1). In Table 5.1, the second column represents the possible rules that belong to each category, for any cell $(i)_{1\leq i\leq n-2}$.

In analogous manner, Table 5.2 refers to the classification of the rules depending on the parameter $\mathtt{P_l}$. Here, the three categories *completely left dependent*, *partially left dependent* and *left independent* consist of the CA rules for $\mathtt{P_l} =$ 1, 0.5 and 0 respectively. For any category, the second column represents the possible rules of that category at cell $(i)_{1\leq i\leq n-2}$.

Table 5.2: Categories of reversible CA rules on parameter $\mathtt{P_l}$

Category	$\mathcal{R}_i$
Completely Left Dependent	60, 195, 15, 240, 90, 165, 150, 105, 30, 45, 75, 120, 135, 180, 210, 225
Partially Left Dependent	53, 58, 83, 92, 163, 172, 197, 202, 54, 57, 99, 108, 147, 156, 198, 201, 23, 43, 77, 113, 142, 178, 212, 232, 27, 39, 78, 114, 141, 177, 216, 228, 86, 89, 101, 106, 149, 154, 166, 169
Left Independent	51, 204, 85, 170, 102, 153

As already discussed, a CA is expected to have a cycle of large length, when its rules are dependent on both of the left and right neighbors, so that, a change in any cell affects neighbors in both directions. For a rule $\mathcal{R}_i$, the degree of dependence on both the neighbors, denoted as $\mathtt{P}(\mathcal{R}_i)$, can be measured by the product of $\mathtt{P_r}(\mathcal{R}_i)$ and $\mathtt{P_l}(\mathcal{R}_i)$.

$$\mathtt{P}(\mathcal{R}_i) = \mathtt{P_r}(\mathcal{R}_i) * \mathtt{P_l}(\mathcal{R}_i).$$

Clearly, $\mathtt{P}(\mathcal{R}_i)$ can take values 0, 0.25, 0.5 or 1. Based on this parameter, we can further classify the rules of reversible CAs into four categories – *completely dependent*, *partially dependent*, *weakly dependent* and *independent* corresponding to the $\mathtt{P}$ values of 1, 0.5, 0.25 and 0 respectively (see Table 5.3).

However, because of the null boundary condition, the rules of Table 5.3 based on the parameter $\mathtt{P}$ as defined above, are not suitable as $\mathcal{R}_0$ and $\mathcal{R}_{n-1}$, the rules for the terminal cells. So, we redefine all these parameters for $\mathcal{R}_0$ and $\mathcal{R}_{n-1}$. Let us start with $\mathcal{R}_0$. Let $\alpha_{rd}(x_{-1} = \mathtt{0}, x_0 = \mathtt{y})$ denotes the dependence of cell 0 on its right neighbor where $x_{-1} = \mathtt{0}$, and x_0 is $\mathtt{y}$. Here, $\mathtt{y}$ can be either 0 or 1.

$$\alpha_{rd}(x_{-1} = \mathtt{0}, x_0 = \mathtt{y}) = \begin{cases} 1 & \text{if } \mathcal{R}_0(\mathtt{0}, \mathtt{y}, x_1) \neq \mathcal{R}_0(\mathtt{0}, \mathtt{y}, 1 - x_1) \\ 0 & \text{otherwise} \end{cases}$$

Therefore,

$$\mathtt{P_r}(\mathcal{R}_0) = \frac{\sum_{\mathtt{y} \in \{0,1\}} \alpha_{rd}(x_{-1} = \mathtt{0}, x_0 = \mathtt{y})}{2}$$

Clearly, $\mathtt{P_r}(\mathcal{R}_0)$ can be either 0 or 1. Table 5.4a shows the categories of the rules at cell 0 depending on $\mathtt{P_r}$. According to the definition of $\mathtt{P_l}$, it is undefined for $\mathcal{R}_0$ as α_{ld} can not be measured for $\mathcal{R}_0$. Therefore, $\mathcal{R}_0$ belongs to the categories of either right independent or completely right dependent.

Similarly, for $\mathcal{R}_{n-1}$, only α_{ld} can be measured which evaluates to $\mathtt{P_l}$. Here, $\alpha_{ld}(x_{n-1} = \mathtt{x}, x_n = \mathtt{0})$ denotes the dependence of cell $n-1$ on its left neighbor where $x_n = \mathtt{0}$ and $x_{n-1} = \mathtt{x} \in \{0, 1\}$.

Table 5.3: Four categories of reversible CA rules on the parameter P

Category	$\mathcal{R}_i$
Completely Dependent	90, 165, 150, 105
Partially Dependent	30, 45, 75, 120, 135, 180, 210, 225, 86, 89, 101, 106, 149, 154, 166, 169
Weakly Dependent	53, 58, 83, 92, 163, 172, 197, 202, 54, 57, 99, 108, 147, 156, 198, 201, 23, 43, 77, 113, 142, 178, 212, 232, 27, 39, 78, 114, 141, 177, 216, 228,
Independent	51, 204, 85, 170, 102, 153, 60, 195, 15, 240

$$\alpha_{ld}(x_{n-1} = \mathtt{x}, x_n = \mathtt{0}) = \begin{cases} 1 & \text{if } \mathcal{R}_{n-1}(x_{n-2}, \mathtt{x}, \mathtt{0}) \neq \mathcal{R}_{n-1}(1 - x_{n-2}, \mathtt{x}, \mathtt{0}) \\ 0 & \text{otherwise} \end{cases}$$

Therefore,

$$\mathtt{P}_1(\mathcal{R}_{n-1}) = \frac{\sum_{\mathtt{x}\in\{0,1\}} \alpha_{ld}(x_{n-1} = \mathtt{x}, x_n = \mathtt{0})}{2}$$

Clearly, $\mathtt{P}_1(\mathcal{R}_{n-1}) \in \{0, 1\}$. Therefore, $\mathcal{R}_{n-1}$ is the member of either completely left dependent or left independent category (see Table 5.4b). Now, we can calculate P for $\mathcal{R}_0$ and $\mathcal{R}_{n-1}$. Both of them are categorized into two categories: completely dependent and independent. Table 5.4c shows the rules for the terminal cells belonging to these categories based on the value of P.

In the next subsection, we present a procedure to use this parameter to synthesize non-linear reversible CAs with large length cycle(s).

5.2.2 Synthesis of Large Length Cycle CAs

From the previous sections we can observe that, if there is more influence of the neighbors on the change of state of each cell, there is better chance of obtaining a large cycle. This influence is measured by the parameter P, which classifies the intermediate rules associated with the rule vector of a reversible CA into four categories – *completely dependent*, *partially dependent*, *weakly dependent* and *independent*. Any rule belonging to the *completely dependent* class of Table 5.3 is linear and totally dependent on both the neighbors; that means, for these rules, every change of state of any neighbor of cell i induces change in the state

Table 5.4: Categories of rules for terminal cells

(a) Categories of $\mathcal{R}_0$ on $\mathtt{P_r}$

Category	$\mathcal{R}_0$
Completely Right Dependent	5, 6, 9, 10
Right Independent	3, 12

(b) Categories of $\mathcal{R}_{n-1}$ on $\mathtt{P_l}$

Category	$\mathcal{R}_{n-1}$
Completely Left Dependent	5, 20, 65, 80
Left Independent	17, 68

(c) Categories of $\mathcal{R}_0$ and $\mathcal{R}_{n-1}$ based on the value of $\mathtt{P}$

Category	$\mathcal{R}_0$	$\mathcal{R}_{n-1}$
Completely Dependent	5, 6, 9, 10	5, 20, 65, 80
Independent	3, 12	17, 68

of that cell. So, the information of cell $i-1$ and cell $i+1$ is never blocked by cell i. This information flow is *symmetrical* and *completely deterministic* in a sense that, an observer can always predict the next state of such a cell from the present configuration.

However, the rules belonging to the other classes are non-linear and not always dependent on the neighbors. For example, a rule can belong to the *partially dependent* class only when it has either $\mathtt{P_l} = 1$, $\mathtt{P_r} = 0.5$ or $\mathtt{P_l} = 0.5$, $\mathtt{P_r} = 1$. In this case, there is always a flow of information from either direction, but in the other direction, the chance of information flow is 50%, that is, on this direction information flow is non-deterministic. Similarly, a rule can be *weakly dependent* if for this rule $\mathtt{P_l} = 0.5$ and $\mathtt{P_r} = 0.5$. So, for 50% cases, this rule allows the flow of information from its neighbors, but for the other cases, it blocks the information flow. Therefore, rules belonging to this category also have non-deterministic information flow. But, in case of the rules from the *independent* category, information flow from at least one side is always blocked. This assured blocking of information makes the system deterministic and is not suitable for generating large length cycle(s).

As stated earlier, a reversible CA is expected to have large length cycle(s) only when most of the rules are dependent on the neighbors. Now, for the terminal rules, there are only two categories – *completely dependent* and *independent*. So, following the above logic, we have to choose $\mathcal{R}_0$ and $\mathcal{R}_{n-1}$ uniformly at random from the *completely dependent* class to fulfil our purpose. However, if all the rules of a reversible CA belong to the *completely dependent* category, the CA is linear. Such CAs are not desirable for security issues (already mentioned in the introduction of this chapter). So, our target is to synthesize non-linear reversible CAs. To accomplish this, we need to select at least some rules which are non-linear; these rules belong to either the *partially dependent* or the *weakly dependent* class. But, selecting more rules from these classes implies flow of information in the CA is sometimes blocked which is a hindrance for generating large length cycle. To trade off this scenario, we choose a Gaussian distribution for the intermediate $n-2$ rules such that maximum number of rules are selected from the category of *completely dependent*, some are chosen from the *partially*

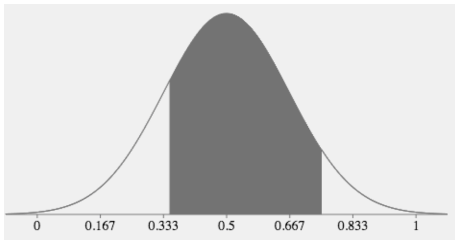

Figure 5.1: Normal distribution of a random variable with mean 0.5 and s.d=0.167

dependent category, and a few are picked up from the category of *weakly dependent* class. Therefore, rules from all these classes are allowed to participate in the synthesis process. This adds a flavor of non-deterministic asymmetric information flow in the configuration space of the synthesized CAs making the configurations more unpredictable, while allowing the CAs to be non-linear. Moreover, as most of the rules have high degree of dependence on both of their neighbors, it is expected that the corresponding CA has a cycle of large length.

An issue with this approach is that the category of the rules takes discrete values whereas the Gaussian distribution works only for continuous variables. Thus we introduce X, a continuous random variable, which follows normal distribution, in particular with mean $\frac{1}{2}$ and standard deviation $\frac{1}{6}$. The mean and standard deviation are so selected such that most values of X (99.74%) lies between 0 and 1. The probability distribution function of X is given in Figure 5.1.

Therefore, our process of selecting each of the $n-2$ rules such that Gaussian distribution is observed is as follows: Pick a random number X following this normal distribution. If X is less than 0.35, choose any rule from the category *partially dependent*. Otherwise, if X is between 0.35 and 0.75, select a rule from the *completely dependent* category; else, pick a rule from the category *weakly dependent*. Thus, probability that the selected rule belongs to the *partially dependent* category is given by

$$Prob[X \leq 0.35] = Prob[Z \leq \frac{0.35-0.5}{0.167}] = \Phi[-0.898] = 0.1841$$

where Z and Φ are the standardized normal variable and the cumulative distribution function of the standard normal distribution [361]. Similarly, probability that the selected rule belongs to the *completely dependent* category (shown by the shaded region in Figure 5.1) is given by

$$\begin{aligned} Prob[0.35 \leq X \leq 0.75] &= Prob[\frac{0.35-0.5}{0.167} \leq Z \leq \frac{0.75-0.5}{0.167}] \\ &= \Phi[1.497] - \Phi[-0.898] = 0.7491 \end{aligned}$$

Algorithm 8 Synthesis process of non-linear reversible CAs with large length cycles

```
    Input: $n$ (CA size)
    Output: $\mathcal{R} = \langle \mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{n-1} \rangle$
 1: Partition each class $v$ of Table 2.7a into different sets $\mathscr{P}^v_q$ as $\mathscr{P}^v_q = \{\mathcal{R} | \mathcal{R}$ belongs to class $v$ and $\texttt{P}(\mathcal{R}) = q\}$, for $q = 0.25,\ 0.5,\ 1$ and $v =$ I, II, III, IV, V, VI.
 2: Select $\mathcal{R}_0$ uniformly at random from $\{5, 10, 9, 6\}$ (see the second column of Table 5.4c). From $\mathcal{R}_0$, the class $q$ of $\mathcal{R}_1$ is identified using Table 2.7b.
 3: for ($i = 1$ to $n - 2$)    (to figure out $\mathcal{R}_2$ to $\mathcal{R}_{n-2}$) do
 4:    Generate a random number $X$ between 0 to 1.
 5:    if $X < 0.35$ then
 6:       if $\mathscr{P}^v_{0.5} \neq \phi$ then
 7:          Select $\mathcal{R}_i$ uniformly at random from $\mathscr{P}^v_{0.5}$.
 8:       else
 9:          Select $\mathcal{R}_i$ uniformly at random from $\mathscr{P}^v_1$.
10:       end if
11:    else if $X < 0.75$ then
12:       Select $\mathcal{R}_i$ uniformly at random from $\mathscr{P}^v_1$.
13:    else
14:       if $\mathscr{P}^v_{0.25} \neq \phi$ then
15:          Select $\mathcal{R}_i$ uniformly at random from $\mathscr{P}^v_{0.25}$.
16:       else
17:          Select $\mathcal{R}_i$ uniformly at random from $\mathscr{P}^v_1$.
18:       end if
19:    end if
20:    Determine class $q$ of $\mathcal{R}_{i+1}$ based on $\mathcal{R}_i$, (using Table 2.7a).
21: end for
22: Select $\mathcal{R}_{n-1}$ uniformly at random from the set {5, 20, 65, 80} (see the third column of Table 5.4c).
```

and finally, probability that the selected rule belongs to the *weakly dependent* category is given by

$$Prob[X \geq 0.75] = Prob[Z \geq \frac{0.75 - 0.5}{0.167}] = 1 - \Phi[1.497] = 0.0668.$$

To summarize the aforementioned synthesis process to obtain non-linear reversible CAs which are expected to have large length cycles, Algorithm 8 is applied.

Example 5.2 Let us consider a CA of size 4 which conforms to the above mentioned stochastic synthesis process. Let $\mathcal{R}_0$ be 9. Therefore, $\mathcal{R}_1$ is

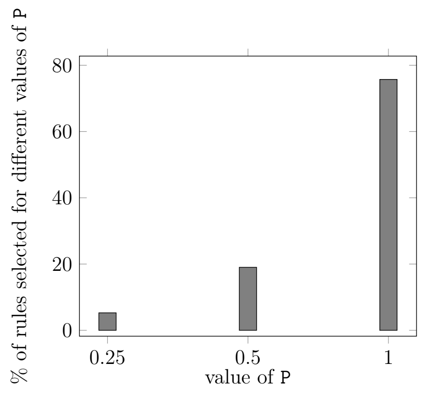

Figure 5.2: Distribution of rules from each category for 1000 large length cycle CAs of size 10

to be selected from class III (using Table 2.7b). To choose $\mathcal{R}_1$, generate X randomly. Let $X = 0.25$. Therefore, $\mathcal{R}_1$ should be selected uniformly at random from set $\mathscr{P}_{0.5}^{III}$. Note that $\mathscr{P}_{0.5}^{III} = \{23, 43, 77, 113, 142, 178, 212, 232, 27, 39, 78, 114, 141, 177, 216, 228\}$. Let $\mathcal{R}_1 = 113$. Next we compute the class of $\mathcal{R}_2$ using Table 2.7a. Here, $\mathcal{R}_2$ is the member of class IV. Again, X is generated randomly. Let $X = 0.95$. Therefore, $\mathcal{R}_2$ should be selected uniformly at random from set $\mathscr{P}_{0.25}^{IV}$. Here, $\mathscr{P}_{0.25}^{IV} = \varnothing$. Therefore, $\mathcal{R}_2$ must be selected uniformly at random from set $\mathscr{P}_{1}^{IV}$. Let $\mathcal{R}_2 = 90$. So, $\mathcal{R}_3$ is to be selected from class IV again. Here, we select $\mathcal{R}_3 = 20$ randomly from Table 2.7c. Therefore, the stochastically synthesized CA is $(9, 113, 90, 20)$.

5.2.3 Observations and Issues with this Approach

To validate our stochastic synthesis process, we have synthesized a large number of CAs for different n. In our experiments we have observed that, for any particular n, around 50% of the synthesized CAs have a cycle of length more than 2^{n-1}. For example, out of 2077 stochastically synthesized CAs generated in a sample experiment for $n = 10$, 1000 CAs have large length cycle(s). For these CAs, the percentage of rules selected for each category are shown in Figure 5.2. Here, 75.72% of the rules are from the *completely dependent* category, 19.01% from to the *partially dependent* class, while 5.27% belong to the *weakly dependent* class. Thus the values conform to the probabilities of selecting rules from the respective categories. We consider this percentage distribution in our endeavors of stochastically synthesizing non-linear reversible CAs which have large length cycle(s).

Table 5.5 shows some sample large length cycle CAs generated by this process for $n = 10$ to 14. Observe that, the largest cycle length of these CAs varies from

Table 5.5: Sample n-cell CAs having cycle of length more than 2^{n-1} where $n = 10$ to 14

CA size	Cycle length	CA
10	1023	$(10, 90, 105, 166, 105, 165, 105, 150, 150, 65)$
	827	$(9, 43, 105, 154, 105, 165, 150, 90, 150, 20)$
	1017	$(9, 90, 89, 150, 165, 150, 106, 90, 89, 80)$
	761	$(5, 90, 165, 180, 154, 165, 106, 165, 90, 80)$
	1003	$(10, 105, 90, 150, 57, 150, 90, 105, 150, 65)$
11	1771	$(6, 150, 135, 169, 105, 90, 105, 90, 150, 165, 65)$
	1959	$(5, 90, 149, 165, 105, 150, 165, 90, 150, 105, 80)$
	2003	$(5, 90, 166, 105, 150, 86, 105, 165, 150, 90, 80)$
	2044	$(10, 165, 54, 149, 105, 105, 165, 165, 165, 150, 65)$
12	3890	$(5, 150, 57, 89, 165, 154, 105, 105, 105, 150, 90, 20)$
	3961	$(5, 90, 58, 165, 90, 90, 165, 90, 165, 105, 165, 80)$
	3999	$(10, 105, 58, 105, 105, 165, 165, 150, 105, 101, 150, 65)$
	3939	$(10, 45, 58, 90, 165, 105, 166, 90, 165, 165, 90, 5)$
13	8021	$(6, 90, 177, 105, 150, 105, 150, 165, 150, 105, 150, 154, 5)$
	8143	$(10, 30, 149, 150, 150, 165, 90, 89, 105, 90, 90, 165, 20)$
	8161	$(9, 150, 30, 105, 90, 154, 165, 101, 150, 105, 105, 105, 65)$
	8179	$(5, 180, 101, 90, 150, 150, 105, 105, 165, 101, 165, 90, 5)$
	8191	$(9, 113, 165, 90, 150, 105, 105, 105, 150, 150, 90, 90, 65)$
14	16381	$(10, 135, 166, 165, 165, 165, 150, 105, 165, 105, 105, 165, 165, 5)$
	16293	$(6, 165, 101, 90, 150, 165, 165, 90, 90, 90, 154, 90, 90, 80)$
	16118	$(6, 113, 90, 105, 86, 165, 165, 150, 105, 154, 165, 105, 105, 65)$

2^{n-1} to $2^n - 1$. For example, the CA shown in the second row of Table 5.5 has a cycle of length $2^n - 1$, that is, maximal length. Hence, we can apply this scheme to generate CAs of size n which can form cycles of length as close as $2^n - 1$. We make the following observations from the sets of the CAs generated.

- Our CAs include both linear and non linear rules. There even exist some non-linear CAs which can generate the cycle of maximum length (first row of Table 5.5). Prior work has studied the use of primitive polynomial (which employ only linear CAs) for generation of cycles of maximal length. It is an interesting open problem to prove the existence of such non-linear CAs for any size.

- If we generate good number of CAs of a given n, it has been seen that for any prime number less than 2^n, there exists at least one CA which generates cycle of that length. In fact, the statement is true for any natural number less than 2^n. Our experiments show that the statement holds for CAs of size less than 30. Proving that cycle lengths of CAs can be used as natural number generator for any n is, again, a challenging open problem.

The stochastically generated reversible CAs produce large length cycles can be verified experimentally upto $n < 30$. But when n is large, to judge whether

Table 5.6: Number of unique configurations explored for some sample 51-cell CAs in 1 hour; experiments are performed on an Intel(R) i5 2.40GHz system

CA	Initial configuration	Explored length
(10,165,150,149,90,169, 90,86,90,169,150,150, 90,105,165,105,150,105, 105,165,105,105,105,105, 165,165,150,165,105,150, 90,150,90,150,150,90, 166,150,90,90,105,105, 150,165,90,165,105,90, 150,90,20)	11011000110010001010100011 0111110101001111110111111	2515293917
	01110010001101111010101000 0011100110100110011010000	2516267780
	10101101010011111000101011 1111100001100110000101101	2515924016
	00011100001101110011111011 0000010001001000101111100	2515601984
	00001100100000010101100011 1000110111001101010000111	2515611772
(9,150,150,99,90,165, 150,150,90,165,165,90, 165,90,105,106,165,169, 90,105,150,165,165,150, 105,165,165,105,105,90, 150,105,101,105,105,149, 105,165,90,90,165,105, 106,150,105,105,90,89, 150,165,65)	10001110001110101100101100 0111101001111000110011001	2509354513
	10110111000010000100101011 0010010100011011000010110	2510227312
	11110011011110110000100100 0011001101110111001101101	2509867813
	00100100100111101110001000 0101101000010111000001110	2509222003
	10110101110001001110000000 1010000100000010011000010	2508540804
(6,89,165,150,105,105, 106,105,165,165,90,150, 106,165,150,165,105,165, 165,150,165,165,105,105, 105,90,105,101,150,150, 154,165,90,90,106,165, 101,150,150,165,105,165, 150,166,105,105,169,150, 90,90,65)	01110000101110011001100101 1011110001001010100001000	2516410312
	01111000101000110100000100 1001000100100001111000110	2517302106
	01010000110100101000011111 0101100010010111111010010	2516687313
	10101000110100100010101011 1100011110001111011101011	2516617796
	11000011001111011000010100 1000111011011010101110000	2516463320
(9,90,105,30,169,90, 165,150,165,150,105,150, 165,165,165,90,165,150, 105,105,150,150,150,90, 165,150,105,165,90,150, 90,105,150,90,105,105, 150,86,105,105,166,105, 105,106,165,105,165,90, 150,165,20)	11010011010111010111110101 1000011001100111000101010	2510352727
	00010101111100101100011100 1100100100101011101100000	2509928244
	10100111010011010111000011 0101111111000010101110001	2509967091
	10111100110001111110011000 1010100110010100101011101	2509370458
	11010000100111111010101111 1001111101001111100011001	2508828137
(9,78,90,150,90,150, 101,105,90,90,165,165, 105,90,105,150,90,105, 150,105,90,105,105,165, 105,105,150,150,169,90, 89,150,90,150,86,165, 149,90,105,105,165,90, 165,105,150,150,105,165, 165,166,80)	01001101010100001101110010 0111110100001111011011000	2552170157
	11010111010111110101001111 1011100110011000011111110	2526258429
	11111010101100101011011100 0111011100001110000010011	2527739478
	10000101001111111100000001 1111001111010010001110101	2525248655
	10101010100011101010010001 0000100101111100111001011	2527589112

the given CA can produce large length cycle or not, we start with an arbitrary seed which is a sub-configuration of that CA and check when that seed will be revisited. Table 5.6 to Table 5.8 show some sample CAs of size 51, 65 and 129 generated using this scheme. In each table, the first column shows the CA. As, n is very large, we run these CAs for some arbitrary seeds for a fixed time and check whether the cycle is completed in the allotted time. The third column reports the number of unique configurations visited by the CAs for the seeds in the allotted time. It is observed that, cycle is not exhausted in the allotted time for any of the CAs with any n. This proves that, our stochastic process really generates CAs with large length cycle, and many of synthesized CAs have a cycle at least as large as 2^{n-1}.

Though Table 5.6 to Table 5.8 confirm that stochastically generated CAs produce large length cycles but we do not get the cycle length. Therefore, for $l \approx 2^{100}$, we can not figure out the corresponding CAs (as from the pool of all possible 100-cell large length cycle CAs, the proper CAs need to be figured out) or for a 100-cell CA, figuring out the CS of that CA is very time consuming, since we use brute force method if there is no right or left independent rule.

Table 5.5 shows that for some values of n, we can apply this scheme to generate CAs of size n which form cycles of length close to $2^n - 1$.

As a solution to this problem, we use the generic operator $\odot$ (see Section 4.3.1 of Chapter 4). The operator generates a CA with a cycle of length l from two CAs having cycles of lengths l_i^1 and l_j^2 where $l \geq l_i^1, l_j^2$.

5.3 Design of (Near) Optimal Cellular Automata

In this section, we extend the $\odot$ operator to concatenate multiple CAs using the theoretical framework established in Section 4.3.1 of Chapter 4.

Let $\mathcal{R}^{n_1}$, $\mathcal{R}^{n_2}$, $\cdots$ $\mathcal{R}^{n_p}$ be the CAs of sizes n_1, n_2, $\cdots$ n_p respectively. Therefore, R, a CA of size $n_1 + n_2 + \cdots + n_p$ can be generated by applying the $\odot$ operator $p-1$ times. We define $\mathcal{R}$ as follows:

$$\mathcal{R} = (\cdots((\mathcal{R}^{n_1} \odot \mathcal{R}^{\mathtt{n_2}}) \odot \mathcal{R}^{\mathtt{n_3}}) \cdots \odot \mathcal{R}^{\mathtt{n_p}}).$$

Suppose $l_{m_v}^v$ be the length of any cycle of CA $\mathcal{R}^{n_v}$, for $v = 1$ to p. Then by extending Corollary 4.3, it can be shown that the CA $\mathcal{R}$ generates some cycle(s) of length $\mathtt{lcm}(l_{m_1}^1, l_{m_2}^2, \cdots, l_{m_p}^p)$.

We now show how a cycle of large length can be generated using the above concept. Recall that we had used brute force method to compute the cycle structure of an n-cell CAs. The large running time for the brute force method compelled us to restrict n to a value at most nbase. In our case nbase $= 30$. In

Table 5.7: Number of unique configurations explored for some sample 65-cell CAs in 75 minutes; experiments are performed on an Intel(R) i5 2.40GHz system

CA	Initial configuration	Explored length
(6,150,150,89,150,90,90, 90,150,150,150,150,150,105, 165,105,90,165,150,90,105, 105,165,165,90,105,150,90, 90,90,90,150,150,105,90, 169,150,105,149,150,90,105, 86,165,89,150,105,169,90, 106,90,86,105,90,150,169, 165,105,165,165,90,105,105, 90,65)	00101011011111001001111001110010 0001110100011110111111111010011010	3085073588
	00110010101010111000010100010110 1011100001001111011000100101011 10	3085781422
	10110100101010000000110011001100 0110010110000000110001110111101 11	3085663848
	11000100100011101100011000100001 1011111100111010111000000110111 01	3085712119
	10001101010001010010000111000011 0100101100001001011011100101110 01	3085316105
(5,150,108,150,105,90,165, 165,150,150,105,90,150,154, 150,150,90,165,105,105,150, 105,150,165,165,90,105,150, 165,165,150,150,90,90,90, 150,150,105,105,165,105,150, 150,105,105,165,89,150,165, 150,105,105,90,90,165,165, 150,169,90,150,165,169,105, 90,65)	00111010101001110000110100010011 01001100110111110111010010101011 1	3093170059
	11001011000011000101000010101110 00000111011101000010111000011000 0	3093645131
	00001100010110001010110000100100 10000000000100110000000000100101 00	3093120691
	11001100110110110110111111010000 10011010111011010100111001011011 0	3093019603
	10011000010111101110001100101000 00111111010111001111101100110010 0	3093113356
(9,165,166,150,105,150,90, 86,90,105,165,165,150,90, 90,150,165,105,90,165,90, 90,150,165,165,150,105,105, 105,90,90,90,105,105,150, 149,90,154,165,90,90,165, 90,105,90,86,165,154,90, 150,90,169,105,150,101,105, 150,105,90,90,165,86,105, 90,65)	11001100011001100101111101011100 11000000110101100001110101110111 0	3083504477
	11001110111101101110110001110000 00011101010101100100010001110110 0	3084098682
	11101010001011001000101000011010 01011100110100111000010011001101 0	3083739808
	01001010000001101100011111001010 11001101001011010010111000100001 1	3084111777
	01011111010111001010000010010100 00011101101001011011011101000010 0	3083821399
(6,43,150,86,105,150,149, 105,90,150,150,150,165,90, 165,150,149,90,105,165,166, 105,90,105,149,165,149,165, 150,150,165,165,150,105,105, 165,90,105,150,105,165,150, 86,150,90,150,105,90,105, 165,150,150,165,90,90,150, 150,165,165,150,150,149,105, 90,20)	10000001100001010001000000001101 00110110110111111011011001010001 11	3083026957
	01111011100010001100101011000110 10000101100100111111101101101010 1	3083376882
	10100111101101110011101011110101 00101110111100100011000101001101 1	3082620243
	11001000110000000000010101011100 10100101001100100010111100011000 0	3082710343
	10000000111111110100100100010111 10100101111010110111100000101011 0	3082910844
(6,169,165,150,90,90,105, 150,165,150,105,165,90,149, 90,105,90,169,105,105,106, 150,150,105,105,165,165,105, 90,165,150,150,165,90,165, 150,105,150,90,105,90,105, 150,106,150,105,90,105,150, 86,105,165,165,105,165,90, 166,90,90,90,169,90,101, 150,20)	10011110100101001000101110001010 01110011101111101110000100110011 0	3082159180
	10010000101010111111101111010111 10110111111110001111100010101111 1	3082869155
	00100011100100011101001001010101 11111000101101111111001011110100 0	3082340966
	11100110001110011111110000110110 00001111100101000111000001100000 0	3082340966
	01110010000100010110110000001101 01101001001011110010110111100100 0	3083358915

this section, we remove this restriction on the values of n by application of the above mentioned theories. Here, we design a non-linear CA of size n ($n >$ `nbase`) for any length of cycle.

Table 5.8: Number of unique configurations explored for some sample 129-cell CAs in 150 minutes; experiments are performed on an Intel(R) i5 2.40GHz system

CA	Initial configuration	Explored length

(6,149,165,105,90,150,150,165,150,
169,105,150,105,90,169,165,90,90,
105,105,105,165,165,86,165,89,165,
	10011100000101101101111111100011110011011 00
	111110001100011110011111001111011100000 1000
	0111111001111001101010110110100011101101001
		4211949917

165,105,150,89,105,165,165,165,90,
90,105,165,90,106,90,149,105,90,
90,105,86,90,105,165,105,90,150,
	000101011111111000001110111110010000111 0010
	1011010101010101001001000101111010000010111
	1011110110100101010101101111001110000111101
		4212044602

105,165,105,150,105,105,105,90,149, 90,89,90,89,90,169,90,165,150, 105,165,105,90,105,106,165,105,90,	1111001110011111110000100010101011111100100 1110000110011011110101101001111000011000101 0001111001100101111101010010110110000000100	4212040372
86,105,105,150,105,165,165,150,90, 150,165,165,165,150,105,165,149,105, 105,86,90,86,105,105,105,150,150,	1100001001011000010100000000100001001001100 0100110011011010100101000011111101010001100 1101001001001011110010010110010100010010111	4211048173
90,90,169,150,105,86,105,165,165, 165,150,101,165,106,105,150,105,150, 165,90,65)	0101010010100010001011000010101010111110100 0010101111111001101111100011001001011000101 0011110101000011010001110100001000111001101	4212042280
(5,105,105,166,90,90,150,165,165, 165,150,90,150,165,90,105,105,105, 165,150,105,165,165,165,154,90,90,	1110000011100010111100100011101011011010010 0100111001000001101010001100110100110000000 0001000010000100010111011111111110010110100	4212008048
165,150,165,154,150,165,165,165,165, 105,165,150,165,105,166,165,106,165, 165,150,105,169,165,166,105,150,89,	1010111000000110011010011100110111111011010 0010001011011011101001110101101101111111000 1010110011101110010011100001101000110111010	4212289294

165,90,150,150,149,90,101,165,150,
150,165,90,105,149,90,105,90,86,105,
105,165,150,150,105,150,105,105,105,
	1011001010001000100000110010000101001010010
	1100100100000101101111011110101011010111 1011
	1100111001100101101010011100110000010110000
		4212277222

90,90,90,165,165,150,166,165,90, 165,105,105,90,90,105,150,165,90, 105,165,150,105,105,105,105,105,150,	0110011101110011001001111000100101100110001 1100101100010110000100100001000111101000101 1111111110100110010101000001010001110000000	4212993550
105,105,150,90,105,90,165,105,86, 105,105,150,165,150,90,150,150, 90,165,20)	0011110111011011110111010111101100010001110 0111100001111000101010000011001010000010101 0001100110100111110000011001101101011101111	4212279899
(6,150,150,105,150,45,202,165,90, 90,90,90,165,150,150,105,105,165, 105,165,150,101,105,150,86,90,154,	1100111101100101011101110011011101101011110 0001010111101100111011110111001001101100111 0011100111011101100000101101010011101011110	4213246977
90,165,105,150,90,150,150,150,105, 165,165,165,90,90,150,150,105,150, 90,165,89,165,86,90,150,165,90,	0010100010011100100001111111000101110111010 1011011010110001101001111110101010111011110 1100001000011100011001110101011111111101101	4213002097
105,150,165,165,105,150,90,90,165, 90,165,150,86,90,86,165,89,150, 165,150,165,90,154,90,90,150,150,	1111100101011101010100101001100011100110010 0001011001101100000111011000010010101101011 1011011101010010000100000101011110110010110	4213020271
150,165,150,165,165,150,165,105,90, 105,165,165,90,105,169,165,150,165, 90,165,149,105,150,165,105,105,89,	0110101001101001010000110101111100100000010 1000010011111110011110010101111010101101000 0110100000010111011101000111011001011011011	4213006379

150,105,150,150,150,165,150,105,165,
165,166,90,89,150,105,150,105,105,
90,150,65)
	1100111000001010000000101010010101110101111 1
	0110110101010100001110011000100111010000000
	0100001111101111111000011101101101101010101
		4213001971

(10,90,150,165,90,178,105,90,105, 90,105,165,150,105,106,165,86,165, 150,150,105,101,105,165,90,165,150,	0010000011001010010111100100101101011010110 1011001010110011100101111111100001110011001 1000100010101110000000110101100001001100101	4213181782
169,105,90,90,105,154,105,105,90, 165,105,165,166,165,90,105,105,105, 165,90,165,101,150,105,165,105,90,	1111100011100001101110111011001111111110000 0011010100100111111110001111111110011010000 0110101011010101010111100110110011000101001	4213271342
165,165,105,105,105,90,105,169,90, 150,150,105,165,150,105,154,105,150, 106,90,169,165,165,165,105,90,150,	0111111000101001110101101001000011100100111 0110100011011000100101100010101000011010111 0010100000100110100100111101001100110010011	4213970221
90,89,90,105,150,150,150,105,105, 90,150,90,150,90,150,150,150,90, 105,105,165,105,150,105,165,165,	0000100001110000101001110100111100010001110 0000101100010001110110111110100101111000011 1110010101010000110011111111100010001111001	4213976201
105,90,90,169,150,150,101,150,150, 90,90,165,165,105,165,166,105,165, 165,165,90,20)	0010100100011100010100101001000001010100101 0110100011010011111111001101000011010011110 0001111010011100100101111000101101110011001	4213279780

(9,165,165,177,150,90,149,150,165,
165,105,86,165,165,165,86,150,150,
101,165,165,150,165,165,105,105,165,
	1010010001010000000101101001011001000100000 1
	0111110101011000110110110010011000111111110
	0110101011001011010010011100011100110000000
		4212590101

169,165,165,105,90,150,105,165,150,
150,150,150,90,165,150,90,165,150,
150,105,90,90,150,89,105,150,105,
	1010010100100001101001100010000000100001001 1
	0111110100100110100101111100111010100010000
	0110110001010110111100000001000001100101111 0
		4212965120

165,169,150,165,150,105,90,86,165, 149,150,150,105,90,86,105,165,90, 165,150,90,90,105,165,105,150,105,	0010100000100100000011100111101101101100110 1101101101001011110011111101011000110110010 1110001000011110110110110001011111011101101	4212271020
169,165,166,165,89,165,165,90,165, 165,86,150,165,165,90,105,90,105, 105,165,150,105,150,90,166,150,150,	1101011011100101110001001100101010111110000 0101000101100101000010110000111101111100010 1010010011001001110100011111100011010101001	4212279922
90,150,150,166,90,89,165,106,150, 165,165,165,165,165,90,90,150,169, 105,90,65)	1000011111100010100011001101100010110001110 1111011000110011001010100101011001100111100 1010010100000001011010000111011011000011011	4212762707

5.3.1 Finding Candidate CAs

Suppose $\mathcal{R}$ be the CA formed by concatenation of the CAs $\mathcal{R}^{n_1}$, $\mathcal{R}^{n_2}$, $\cdots$ $\mathcal{R}^{n_p}$ of sizes n_1, n_2, $\cdots$, n_p respectively. Thus

$$\mathcal{R} = (\cdots((\mathcal{R}^{n_1} \odot \mathcal{R}^{\mathtt{n_2}}) \odot \mathcal{R}^{\mathtt{n_3}}) \cdots \odot \mathcal{R}^{\mathtt{n_p}}).$$

Each of the CAs on which the concatenation operation is performed, termed as *component* CA. Let $l^v_{m_v}$ be the length of the *longest* cycle of CA $\mathcal{R}^{n_v}$, for $v = 1$ to p. As stated before, the CA $\mathcal{R}$ will generate some cycle(s) of length $\texttt{lcm}(l^1_{m_1}, l^2_{m_2}, \cdots, l^p_{m_p})$. Thus for the CA $\mathcal{R}$ to generate a large cycle, we require each of the individual $l^v_{m_v}$ values to be large as well. Each component CA is thus constructed using the technique mentioned in Section 5.2. We hence restrict $n_v \leq \texttt{nbase}$ for all v; moreover the $\mathcal{R}^{n_v}$ is chosen with the constraint that $l^v_{m_v}$ is at least 2^{n_v-1}.

Our procedure thus concatenates component CAs iteratively. For an input l, we check the value of $\texttt{lcm}(l^1_{m_1}, l^2_{m_2}, \cdots, l^p_{m_p})$ after concatenation of p component CAs. If the value is less than $2^{\lfloor \log_2 l \rfloor}$, we concatenate another component CA, otherwise we stop. The final CA $\mathcal{R}$ is the *candidate* CA. In our experiments, we generate independently multiple candidate CAs of different sizes ($\geq \lfloor \log_2 l \rfloor + 1$). The *best* of the candidate CA is the one whose size is closest to $\lfloor \log_2 l \rfloor + 1$. This solution is referred as *current best* solution for our given problem.

As we select the component CAs using a non-deterministic process, for any v we may have to repeat the process for constructing $\mathcal{R}^{n_v}$ until $l^v_{m_v}$ becomes at least 2^{n_v-1}. This may require quite a number of iterations. Secondly, even though the individual $l^v_{m_v}$ values might be large, their $\texttt{lcm}$ values may not be. Therefore, we use a preprocessing step where we precompute the cycle structure of a set of CAs of size n_v ($n_v \leq \texttt{nbase}$ for every v). For each CA of this set, we store the rule vector along with the corresponding longest cycle length in a *database* if the maximum cycle length of the CA exceeds 2^{n_v-1}. Thus the database is the collection of component CAs that we consider subsequently. We will refer this database as *universe*.

Our process of determining a candidate CA thus involves selecting the component CAs from this pool of CAs in the universe. For a given cycle length l, first we select any CA size randomly between 3 and $\texttt{nbase}$. Let n_1 be the size of the first CA. Out of all the CAs of size n_1 from the universe, we choose a random one following Gaussian distribution, with more weights given to the one generating longer cycle. This CA forms the first component CA. The reason behind this way of selection is that it is more likely that the length of the longest cycle of the selected CA is close to $2^{n_1} - 1$. Using the same approach, we select p component CAs. By applying $\texttt{lcm}$ on the cycle lengths, if the output cycle length is larger than $2^{\lfloor \log_2 l \rfloor}$, we are done; else we proceed to get the next component CA. Thus finally, we obtain our desired CA which generates a cycle of length at least $2^{\lfloor \log_2 l \rfloor}$. By following Gaussian distribution on the length of cycles to select a component CA, we expect that the candidate CA can be achieved of size closer to $\lfloor \log_2 l \rfloor + 1$.

It is possible to formulate the aforementioned problem as a variant of the

Table 5.9: Experimental results for identifying the best candidate CAs of given length of cycles

Range of Cycle length l	size of current best solution	displacement percentage
$[2^{500}, 2^{501})$	509	1.6
$[2^{1000}, 2^{1001})$	1078	7.69
$[2^{2000}, 2^{2001})$	2310	15.44
$[2^{4000}, 2^{4001})$	5034	23.01
$[2^{6000}, 2^{6001})$	7726	28.74
$[2^{8000}, 2^{8001})$	10711	33.88
$[2^{10000}, 2^{10001})$	13631	36.3
$[2^{12000}, 2^{12001})$	15603	30.01
$[2^{14000}, 2^{14001})$	18212	30.01
$[2^{16000}, 2^{16001})$	20812	30.07
$[2^{18000}, 2^{18001})$	23425	30.01
$[2^{20000}, 2^{20001})$	26121	30.53

standard knapsack cover problem. For instance, each entry in the universe is associated with CA size (cost) and cycle length (value). Given an input l, the objective thus is to find the subset of the universe of minimum cost such that the `lcm` of the values of the selected entries in the subset is at least l. However, extending the dynamic programming based algorithm for knapsack cover problem to this specific version has its limitation due to increased time complexity.

A connection between knapsack cover problem and the problem of solving equations on discrete dynamical systems (DDS) has been shown in [362, 363]. Furthermore, it is demonstrated that multi-valued decision diagrams enhance the performance of the algorithm for the DDS problem [363]. It is an interesting future work to extend this algorithm for generating CAs with large length cycles.

5.3.2 Result and Analysis

We have done extensive experiments using the scheme for various input cycle lengths. For a particular input l, we consider 500 candidate solutions, each of which generates a cycle of length at least $2^{\lfloor \log_2 l \rfloor}$. In Table 5.9, for an input l (the first column), the second column of the table gives the size of the current best solution which generates a cycle of length at least $2^{\lfloor \log_2 l \rfloor}$. As an example, let us consider, for a given l (where $l \in [2^{4000}, 2^{4001})$), our procedure returns the current best solution of size 5034. Note that on contrary to the brute force method which work only for small input values, our scheme can handle input values as large as 2^{25000}.

Sometimes it may happen that the size of the current best CA is too large compared to the optimal CA. From Table 5.9, we can compare between the optimal CA size and the size of the current best solution which we obtain experimentally, for a given k. From the results, we observe that the displacement percentage is proportional to the optimal CA size. Therefore, when the input cycle length is very large, we may get the displacement close to $\approx \lfloor \log_2 l \rfloor + 1$. So, we are motivated to identify the reason behind it which can help us to reduce the displacement.

5.3.3 An Issue with the Given Approach

Our procedure of selecting the component CAs from the universe is a randomized process, thus even though the individual cycle length of the component CAs might be large, the value of their lowest common multiple may not be that large. The `lcm` of two numbers is as large as their product only if the numbers are mutually co-prime; in general, the `lcm` is large only if the numbers have only few factors in common. However, in our scheme, we have not explicitly checked co-primality among cycle lengths of the component CAs when we create the candidate CA - it might not be even feasible in practice to create a candidate CA, where the cycle lengths of the component CAs are constrained to be mutually co-prime.

We thus try to improve the candidate solution by applying some heuristic to reduce the displacement and produce near optimal solution. In this dissertation, we address this issue by employing a novel biologically inspired algorithm, which we discuss in the next section. We begin with a *family* of solutions, each created by concatenation of the component CAs as explained in this section - this forms our initial solution space. In the next section, we explain how to improve these solutions to produce a near optimal CA.

5.4 An Evolutionary Strategy for Getting (Near) Optimal CAs

We now present a mechanism for improving the solutions that we have obtained using the techniques discussed in the previous sections. Recall that we create a family of solutions where each solution is formed by concatenating different component CAs. This set of solutions forms the initial solution space or the population. Each solution of the population is guaranteed to generate a cycle of length more than $2^{\lfloor \log_2 l \rfloor}$, however, the size of a solution can be much larger than the optimal value of $\lfloor \log_2 l \rfloor + 1$. Our focus in this section is to ensure that the population gets evolved toward better solutions. Here we use an *evolutionary strategy* having the following basic steps.

- Determination of initial population: A family of candidate solutions is determined randomly.
- Evaluation of fitness score: Determine a fitness function to measure how *fit* the solution is for the given environment.
- Generation of offspring: Select *good parents* (solutions) and create offspring in biological metaphor by inheriting the *goodness* of the parents.

We follow some variant of a genetic algorithm - the notable change in our strategy with respect to genetic algorithm is that the offspring are not produced by the selected parents using crossover or mutation. Here, a *good* offspring is generated by the *significant* features of multiple parents. A solution (parent or off-spring) is termed as *good* if the corresponding CA size has less displacement. The *significant* feature of a solution is derived from the component CAs that concatenate to form the solution (candidate CA). This *significant* feature mainly contributes in our system to achieve *good* solutions with significantly less value of displacement. Next we describe the proposed strategy in detail.

5.4.1 Obtaining Good Solution using Significant Features

Our scheme is an iterative process, the population in each iteration is called a *generation*. Thus, the initial population corresponds to the first generation solutions. In each generation, the *fitness score* of every individual (solution) in the population is evaluated. In our case, the fitness score is the size of the CA, thus, lower the score value, better is the solution. The intuition is to select the more fit individuals, which we term *good* solutions, from the current population, and pass them on to the next generation. In addition, we *mix* the solutions in the current population in such a way that the components of the solutions are mixed. That means the solutions are broken into *elementary components* - the collection of all such components forms the *elementary component space*. In our case, an elementary component is a component CA and the number of factors of the cycle length of the elementary component forms the significant feature. We create the next generation by selecting the elements of elementary component space with a strategy which we discuss later. Therefore, a new solution is not restricted to be created by only some selected parents, rather it is formed by the elementary components of multiple parents. This approach brings diversity in the next generation since an offspring can inherit features of multiple parents. In addition, a portion of the next generation is created by selecting the component

CAs from the universe randomly and uniformly. This step is included to incur diversification - from these CAs we may achieve more significant features which are not the part of current generation.

We now give some more detail on how the $(i+1)$th generation is created from the ith generation. Let $\mathcal{M}_i$ denote the set of individuals (solutions) which form the ith generation population. We sort the solutions in $\mathcal{M}_i$ in ascending order of their fitness value. The following steps are then carried out to generate $\mathcal{M}_{i+1}$. Our steps ensure that the cardinality of sets $\mathcal{M}_i$ and $\mathcal{M}_{i+1}$ are same.

- *Selection of good parents from $\mathcal{M}_i$:*
  The first 30% solutions in $\mathcal{M}_i$ are passed on to $\mathcal{M}_{i+1}$. Here, the offspring are identical to the selected parents. As we select top 30% solutions from $\mathcal{M}_i$, therefore it is guaranteed that the best solution of $\mathcal{M}_i$ must be evolved to next generation. However, in order to obtain even better solutions (compared to the best solution of $\mathcal{M}_i$ in $(i+1)$th generation) in the next generation, we should mix the solutions of the current generation. The next two steps are used for diversification which leads to achieve more accurate solution.

- *Mixing the component CAs of $\mathcal{M}_i$ and allocation of weightage to them on significant feature:*
  50% of the solutions in $\mathcal{M}_{i+1}$ are generated by mixing the elementary components of the solutions of $\mathcal{M}_i$. Mixing is done in such a way that an offspring (solution) possesses the features of multiple parents of $\mathcal{M}_i$. Let $\mathtt{C_i}$ be the elementary component space of $\mathcal{M}_i$, which is the collection of all the component CAs that form the different solutions in $\mathcal{M}_i$. The feature of each component is determined by the number of factors of the cycle length of the component - a more significant feature implies less number of factors of the corresponding cycle length. Recall from the last section that the performance of our algorithm can be improved if we concatenate component CAs with co-prime cycle lengths to form candidate CAs. Thus intuitively, our mixing step picks component CAs (from the elementary component space) generating cycle lengths with less number of factors, and concatenates them to form candidate CAs, with the hope that the these selected component CAs will generate co-prime cycle lengths (or cycle lengths with very few factors in common).

  To select the component CAs for mixing, we partition the elementary component space into blocks depending on the number of factors of the cycle lengths of each component CA in it. The length of the cycle generated by each of component CAs in a particular block will have the same number of factors. A component CA of an off-spring is selected as follows: first, a block is selected randomly following Gaussian distribution with more weights given to the blocks representing less number of factors; once a block is selected, a component CA is chosen from the block randomly with uniform distribution. Once the component CAs are selected, an offspring is created by their concatenation in a similar manner as described in Section 5.3.

- Generation of solutions randomly:
  The final 20% solutions of $\mathcal{M}_{i+1}$ are created by selecting randomly the component CAs with uniform distribution from the universe. Note that this is not a refinement technique since these solutions are generated right away by combining the selected component CAs, and may or may not have small fitness values. The reason for inclusion of this step is that some component CAs which were missing in $\mathcal{M}_i$ may also be useful in the subsequent generations to produce good solutions. Thus by adding some impurity in $\mathtt{C_{i+1}}$, there is a chance of obtaining some component CAs which produce large cycle lengths with less number of factors. Note that the previous two steps ensured that $\mathtt{C_{i+1}}$ as the subset of $\mathtt{C_i}$, whereas this step adds new component CAs which brings diversity in $\mathtt{C_{i+1}}$.

Using the above steps, we generate solutions over different generations. We repeat the next generation evolution process until one of the given conditions is satisfied:(1) The fitness value of the best solution in the current generation is same as the fitness value of the best solution in the previous generation.(2) The population has evolved for a considerable number of generations.

5.4.2 Experimental Observation

We now present the performance of our strategy of designing CAs with large cycles (see Table 5.10). In Table 5.10, the first column represents the length of the given cycle l, the second column of the table gives the size of the near optimal CA which is obtained using the aforementioned evolutionary strategy. The displacement δ is given by the difference of this value and $\lfloor \log_2 l \rfloor + 1$. The third column represents the percentage of displacement which is computed as $((\delta / \lfloor \log_2 l \rfloor + 1) * 100)$. As an example, let us consider, for a given l, we get $\lfloor \log_2 l \rfloor + 1 = 3001$. So, our aim is to find a CA which generates a cycle of length more than 2^{3000} (see first column and row eight of Table 5.10).

The near optimal solution for this instance is of size 3013 (second column of the same row) with displacement percentage of 0.4. From this table, we conclude that the displacement is very less compared to the size of the optimal CA. The relation between displacement percentage and optimal CA size is shown in Figure 5.3. We observe that our bio-inspired strategy drastically reduces the displacement of the solution. Although the displacement increases with l,

Table 5.10: Experimental results for near optimal CAs of cycle length k; experiments are performed on an Intel(R) i5 2.40GHz system

Range of Cycle length l	size of current best solution	displacement percentage	t (in seconds)
[274877906944, 549755813888]	39	0	0.59
$[2^{80}, 2^{81})$	81	0	1.42
$[2^{100}, 2^{101})$	101	0	1.80
$[2^{200}, 2^{201})$	201	0	3.92
$[2^{500}, 2^{501})$	501	0	14.30
$[2^{1000}, 2^{1001})$	1001	0	39.50
$[2^{2000}, 2^{2001})$	2006	0.25	137.93
$[2^{3000}, 2^{3001})$	3013	0.40	297.97
$[2^{4000}, 2^{4001})$	4018	0.42	553.01
$[2^{5000}, 2^{5001})$	5024	0.46	880.30
$[2^{6000}, 2^{6001})$	6029	0.47	1319.42
$[2^{7000}, 2^{7001})$	7036	0.50	1846.44
$[2^{8000}, 2^{8001})$	8043	0.52	2585.02
$[2^{9000}, 2^{9001})$	9051	0.56	3516.66
$[2^{10000}, 2^{10001})$	10058	0.57	4653.00
$[2^{11000}, 2^{11001})$	11064	0.57	6059.43
$[2^{12000}, 2^{12001})$	12072	0.59	8289.78
$[2^{13000}, 2^{13001})$	13082	0.62	11156.39
$[2^{14000}, 2^{14001})$	14089	0.63	15618.94
$[2^{15000}, 2^{15001})$	15096	0.63	21838.29
$[2^{16000}, 2^{16001})$	16108	0.67	30621.81
$[2^{17000}, 2^{17001})$	17121	0.71	42735.80
$[2^{18000}, 2^{18001})$	18132	0.73	60367.73
$[2^{19000}, 2^{19001})$	19143	0.75	83645.53
$[2^{20000}, 2^{20001})$	20157	0.78	118136.77

however, we observe that the rate of its growth decreases with increase in l. For $l < 2^{1000}$, our scheme outputs the optimal CA. In this table, we also add t (column four), the time taken to find the near optimal result using our strategy. Our scheme takes less than a minute to determine the near optimal solution even for $l = 2^{1000}$. In fact the running time is less than a second for $l = 2^{39}$. The relation between running time t (in minutes) and optimal CA size is shown in Figure 5.4. Although the running time increases with k, however, we observe that the rate of its growth decreases with increase in k.

The cellular automata which generate large length cycles are highly useful in computational processes like pseudo random number generator [222, 364, 365], cryptosystem [53, 250] etc. Now, the CAs which generate large length cycles by concatenating the small sized CAs are strictly reducible cellular automata and are *predictable* - from the current configuration, we can predict the future configurations. Therefore, these CAs are not suitable for randomness. Our next

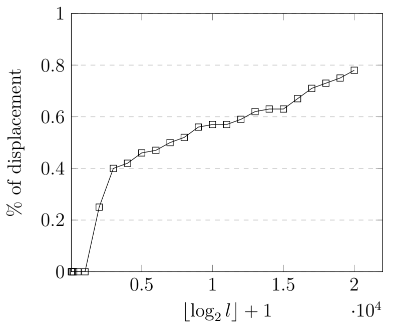

Figure 5.3: Relation between the displacement percentage and optimal CA size $\lfloor \log_2 l \rfloor + 1$

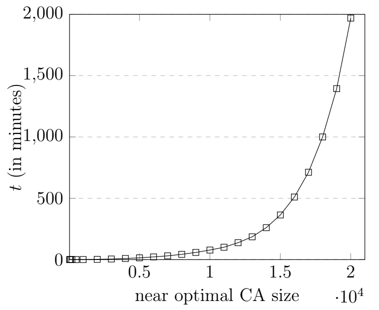

Figure 5.4: Relation between the running time t (in minutes) and near optimal CA size

choice is the component CAs (which are stochastically generated large length cycle CAs). A component CA can be either reducible or irreducible. In case of reducible CA, the rule vector sometimes contains right or left independent rules and the presence of such rule blocks the flow of information in one direction. For example, if a right independent rule is present at cell i of an n-cell CA (where $n > i$), then, such CA is left quasi-isomorphic and in a cycle of that CA, a sub-configuration of length $(i + 1)$ or a collection of sub-configurations of same length may reappear. Hence, the information is blocked up to cell i and the CA is predictable - both the conditions are not desirable for PRNG design even if the n-cell CA produces a cycle of length larger than 2^{n-1}. Analogously, the CA with left independent rule in the rule vector is not acceptable for PRNG design. It is also established that a CA can be reducible even if its rule vector does not contain any left or right independent rule. In such CA, the information flow has not been blocked. Obviously, in irreducible CA, there is no chance of information block. Therefore, for designing pseudo random number generator we use n-cell reversible CA which does not consider any independent rule in its rule vector to generate a cycle of length larger than 2^{n-1}.

5.5 Pseudo-random Number Generator: An Application

The previous sections illustrates the fact that, the stochastically synthesized CAs generated by the aforementioned scheme have cycle at least as large as 2^{n-1} (see Table 5.5 to Table 5.8). As very large period is an essential criterion of a good PRNG, in this section, we use some of these CAs as PRNGs. Here, to use a CA as a PRNG, we extract the numbers from the individual configurations of a CA. So, the period of a CA-based PRNG can be directly related with the

Table 5.11: Number of unique configurations visited by three sample CAs (Table 5.12) in a fixed time, where $n - w$ cells are fixed to $0^{n-w-1}1$ and w cells are set arbitrarily

CA	Initial configuration	Explored length	Allotted time
CA_{51}	1011111101101010111110001	2510266954	3600
	1001101101010000001010000	2510856214	3600
	0000111001010001011110110	2510568945	3600
	1010111000100010111100100	2509456258	3600
	1010100100100111110011001	2509489652	3600
CA_{65}	000000111011001100100111001100001	3093574812	4500
	011110001100100000100100110101111	3093236984	4500
	100101100101111001101110010000101	3092485632	4500
	011100101101011100011011111101000	3093458621	4500
	000000100101100111111101111101001	3092568975	4500
CA_{129}	101110011010111111111000011001100 110101111101000111011100110011101	4213258694	9000
	001110100011000010001111000011000 110100111110100000001011111001100	4213284562	9000
	111001011111111010111111101100100 000011100100011100001001001111101	4213225846	9000
	000001111111000010010111111000011 110010100000100111001101100111111	4213356845	9000
	000001101101001100001111111111001 111000000010110110101111000011110	4213302579	9000

largest cycle(s) in its cycle structure, provided, we can somehow choose the seed as one the configurations of the cycle(s).

First, we report a generalized scheme of using these CAs as *window*-based pseudo-random number generators (Section 5.5.1). This scheme offers some advantages like ease of hardware implementation, robustness and portability. Then, we have empirically tested these PRNGs for randomness using Diehard, TestU01 and NIST as the testbeds (Section 5.5.2). Finally, we compare our PRNGs with 27 popular PRNGs which are considered as *good* (Section 5.5.3). We claim that, our PRNGs can be the more preferable choice for most of the intended applications than all their kins.

However, presenting a large cycle is not sufficient for a CA to be a *good* source of randomness. For this, the evolution of the CA needs to be *unpredictable*, that is, one can not predict the future configurations of the CA from its present configuration, if the rule vector is not known. The preliminary conditions that must be satisfied are: the CA must be non-linear and its local rule must be balanced and dependent on the neighbors, such that, the configurations of the CA are (as far as possible) independent of each other. For these CAs, even the smallest addition of information to the initial configuration propagates in the CA and affects the future sequence of configurations. So, the system can not become unchangeable in a finite time. Nevertheless, the CAs synthesized in the

```
    int cellp[], celln[], window[]; //stores configurations and seed
    int srandCA(int w, int seed[]){//supply window & seed as user input
    n=w*2+1; //set the CA size
    for(i=0; i<w; i++) //initialize window
    cellp[i]=seed[i];
    for(i=w; i<n-1; i++) //initialize other values
    cellp[i]=0;
    cellp[n-1]=1;
    }

    int[] randCA(){     //Generate pseudo-random numbers
    celln[0]=Rule[0][2*(cellp[0])+1*(cellp[1])];
    for(i=1; i<n-1; i++)
    celln[i]=Rule[i][4*(cellp[i-1)+2*(cellp[i])+1*(cellp[i+1])];
    celln[n-1]=Rule[n-1][4*(cellp[n-2])+2*(cellp[n-1])];
    for(i=0; i<w; i++) //populate window from the first w cells
    window[i]=celln[i];
    for(i=0; i<n; i++) //update the next configuration
    cellp[i]=celln[i];
    return window; //return a w-bit number
    }

```

Listing 5.1: Generalized Code for PRNG based on Non-linear CA

previous section satisfy all these conditions. Therefore, these CAs can be candidates to be good PRNGs. The following subsection depicts a generalized scheme to implement PRNGs using these CAs.

5.5.1 Design of Proposed PRNGs

In a PRNG, every number in the range of its period is generated uniformly. However, if we use the CAs synthesized in Section 5.2, then, it may happen that all the possible configurations do not belong to the same cycle. So, taking the whole configuration of the CA as a number does not work. Further, if the size of the CA is large, we can not even (exhaustively) find an initial configuration corresponding to the longest cycles(s). In order to address these issues, instead of the whole n-cell configuration, a window of length w is taken from n $(w < n)$

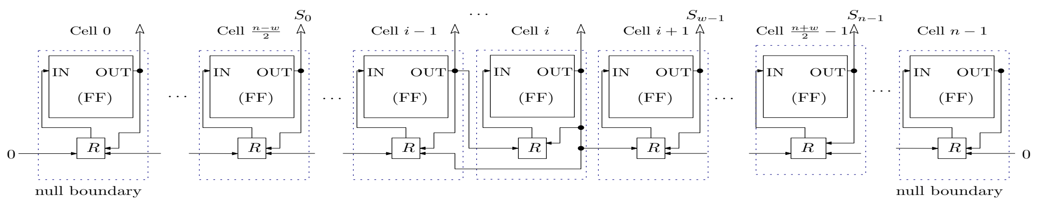

Figure 5.5: Hardware implementation of an n-cell CA-based PRNG

to generate the random number. The cell length n is to be large enough to contain all possible combinations of the window in the same cycle. The value of this window works as the seed of the PRNG which is taken as a user input.

To ensure that the cycle length of the CA is extremely large, we suggest to take the CA size (n) as an odd number at least twice the size of the window (w). For instance, to collect 24 bit numbers, we take $n = 51$; to get a 32 bit number, n is taken as 65; whereas, for $w = 64$, we choose $n = 129$. Further, for the initial configuration, we impose $n - w$ cells to be set to a fixed value. However, to find this fixed seed, we again conduct an experiment. We choose $0^{n-w-1}1$ as the fixed seed and set the remaining cells arbitrarily. Then we run the CA for some fixed time and check whether the cycle is exhausted by that time. Table 5.11 shows some sample results of this experiment for three CAs of sizes 51, 65 and 129. Here also, we observe that, for none of the seeds the cycle is completed in the allotted time. So, in our scheme we take $0^{n-w-1}1$ as the fixed seed. Although any programming language can be used to implement

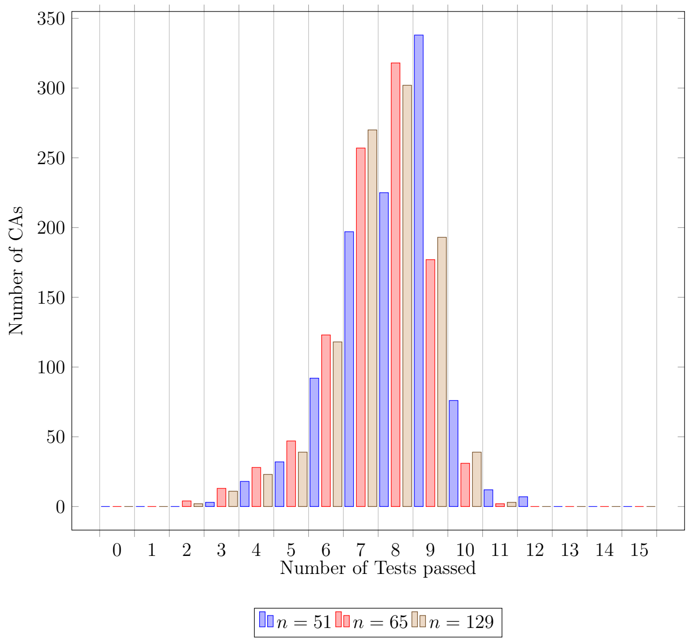

Figure 5.6: Results for 1000 random CAs with Diehard battery of Tests

this scheme, here we show a sample code snippet in *JAVA* (see Listing 5.1). In this code, the method *srandCA()* sets value the window as per the seed supplied by the user. It also sets the CA size accordingly. A candidate CA of that size is chosen arbitrarily following the synthesis algorithm of Section 5.2.2. The method *randCA()* generates a w-bit random number using this CA and returns that number.

This window-based PRNG scheme offers several advantages. For example, it is highly portable and robust. Further, use of 2-state 3-neighborhood CAs in the design has the added edge of ease in hardware implementation, which is difficult for non-binary CAs based PRNGs. To implement a cell i, we need a memory element (Flip-Flop) which stores its state and to find the next state of the cell, a combinational logic circuit implementing rule $\mathcal{R}_i$ is required [44]. For instance, Figure 5.5 represents a schematic hardware implementation of a CA based PRNG with n cells where $\mathcal{R}_0 = \mathcal{R}_1 = \mathcal{R}_{n-1} = R$. Here, the output sequence $S_0 \cdots S_{w-1}$ represents a random number generated by the PRNG. Further, this scheme offers the additional advantage of unpredictability; because, here repetition of a number does not imply completion of cycle. In the next subsection, we empirically verify the randomness quality of PRNGs designed by our approach.

5.5.2 Empirical Verification of the Proposed PRNGs

Empirical tests are used to detect patterns in the generated numbers of a PRNG for proving its non-randomness. In this case, it is checked whether the numbers are locally random – that is, instead of the whole period, randomness of the numbers are approximated over a minimum sequence length [366]. There are several statistical testbeds available in the literature. Here, we discuss results of three specific testbeds that are traditionally well-known to probe the non-randomness of a binary sequence generated by a PRNG, namely, Diehard battery of tests [367], battery *rabbit* of TestU01 library [368] and NIST Statistical Test Suite [369]. Each of Diehard and NIST testbeds contains 15 different tests, whereas, the battery *rabbit* of TestU01 library consists of 25 tests.

To test a sequence using Diehard and TestU01 library, a binary file of size 10-12 MB is supplied, whereas, for NIST statistical testbed, a 125 MB binary file is provided. A generator *passes* a test if p-value of the test is within 0.025 to 0.975 and within 0.001 to 0.999 for each of the tests of Diehard and TestU01 library respectively. However, for each statistical test of NIST (with the exception of the random excursion (variant) test), the generator passes that test if approximately 980 out of 1000 (615 out of 629 for the random excursion (variant) test) binary sequences have p-value > 0.01.

Table 5.12: Sample CAs for $n = 51$, 65 and 129

CA Name ↓	Rule vector
CA_{51}	(9, 90, 105, 30, 169, 90, 165, 150, 165, 150, 105, 150, 165, 165, 165, 90, 165, 150, 105, 105, 150, 150, 150, 90, 165, 150, 105, 165, 90, 150, 90, 105, 150, 90, 105, 105, 150, 86, 105, 105, 166, 105, 105, 106, 165, 105, 165, 90, 150, 165, 20)
CA_{65}	(5, 150, 108, 150, 105, 90, 165, 165, 150, 150, 105, 90, 150, 154, 150, 150, 90, 165, 105, 105, 150, 105, 150, 165, 165, 90, 105, 150, 165, 165, 150, 150, 90, 90, 90, 150, 150, 105, 105, 165, 105, 150, 150, 105, 105, 165, 89, 150, 165, 150, 105, 105, 90, 90, 165, 165, 150, 169, 90, 150, 165, 169, 105, 90, 65)
CA_{129}	(6, 150, 150, 105, 150, 45, 202, 165, 90, 90, 90, 90, 165, 150, 150, 105, 105, 165, 105, 165, 150, 101, 105, 150, 86, 90, 154, 90, 165, 105, 150, 90, 150, 150, 150, 105, 165, 165, 165, 90, 90, 150, 150, 105, 150, 90, 165, 89, 165, 86, 90, 150, 165, 90, 105, 150, 165, 165, 105, 150, 90, 90, 165, 90, 165, 150, 86, 90, 86, 165, 89, 150, 165, 150, 165, 90, 154, 90, 90, 150, 150, 150, 165, 150, 165, 165, 150, 165, 105, 90, 105, 165, 165, 90, 105, 169, 165, 150, 165, 90, 165, 149, 105, 150, 165, 105, 105, 89, 150, 105, 150, 150, 150, 165, 150, 105, 165, 165, 166, 90, 89, 150, 105, 150, 105, 105, 90, 150 65)

We have tested a large number of CAs based PRNGs, synthesized in Section 5.2.2, for different window sizes (accordingly n is varied) on these batteries of tests for many seeds. As, number of possible CAs for each n is huge, we conduct an experiment to get an idea of average tests passed by these CAs for variable n. Here, for each of CA sizes 51 ($w = 24$), 65 ($w = 32$) and 129 ($w = 64$), 1000 CAs are generated arbitrarily and tested using Diehard battery of test for arbitrary seeds. The result of this experiment is shown in Figure 5.6. We can observe that, most of these CAs pass at least 7 out of the 15 tests. However, if chosen wisely, our CAs based PRNGs can pass 13 tests of Diehard. Further, for each w, when n is chosen as suggested, each of the tests of battery *rabbit* of TestU01 library and of NIST statistical test-suite are passed by these PRNGs for any arbitrary seed. Table 5.12 shows three sample *good* CAs for each of these n. The test results of these CAs for TestU01 library and NIST test suite, where, a seed is taken arbitrarily, are shown in Table 5.13 and 5.14 respectively.

These tables prove that, CAs following our proposed scheme have good ran-

Table 5.13: The results of CA_{51}, CA_{65} and CA_{129} test with TestU01 library. Here, window sizes are 24, 32 and 64 respectively with seeds 1005843561, 1178288166 and 823902801 respectively

Test Name ↓	CA_{51}	CA_{65}	CA_{129}
MultinomialBitsOver	Pass	Pass	Pass
ClosePairsBitMatch (t=2 dimensions)	Pass	Pass	Pass
ClosePairsBitMatch (t=4 dimensions)	Pass	Pass	Pass
AppearanceSpacings	Pass	Pass	Pass
LinearComplexity	Pass	Pass	Pass
LempelZiv	Pass	Pass	Pass
Spectral (Fourier 1)	Pass	Pass	Pass
Spectral (Fourier 3)	Pass	Pass	Pass
LongestHeadRun	Pass	Pass	Pass
PeriodsInStrings	Pass	Pass	Pass
HammingWeight (with blocks of L=32 bits)	Pass	Pass	Pass
HammingCorrelation (with blocks of L=32 bits)	Pass	Pass	Pass
HammingCorrelation (with blocks of L=64 bits)	Pass	Pass	Pass
HammingCorrelation (with blocks of L=128 bits)	Pass	Pass	Pass
HammingIndependence (with blocks of L=16 bits)	Pass	Pass	Pass
HammingIndependence (with blocks of L=32 bits)	Pass	Pass	Pass
HammingIndependence (with blocks of L=64 bits)	Pass	Pass	Pass
AutoCorrelation (with a lag d=1)	Pass	Pass	Pass
AutoCorrelation (with a lag d=2)	Pass	Pass	Pass
Run	Pass	Pass	Pass
MatrixRank (with 32×32 matrices)	Pass	Pass	Pass
MatrixRank (with 320×320 matrices)	Pass	Pass	Pass
RandomWalk (with walks of length L=128)	Pass	Pass	Pass
RandomWalk (with walks of length L=1024)	Pass	Pass	Pass
RandomWalk (with walks of length L=10016)	Pass	Pass	Pass

domness quality and they may be used in different applications like, for example, in cryptography. Table 5.15 shows the performance of these CAs in Diehard battery of tests for the same seeds. Observe that, our CAs can pass most of the tests (except *overlapping permutations* and *parking lot* tests). In the next subsection, we compare these CAs with some existing *good* PRNGs (as presented in Table 5.16).

5.5.3 Comparison with Existing PRNGs

From the previous section, we can see that, *parking lot* and the *overlapping permutation* tests of Diehard are not passed by our CAs. But, is there any other existing PRNG which passes these tests? To investigate the randomness quality of the existing well-known *good* PRNGs, these PRNGs are tested on the same

Table 5.14: The results of CA_{51}, CA_{65} and CA_{129} with NIST test suite. Window sizes are 24, 32 and 64 respectively with seeds 1005843561, 1178288166 and 823902801 respectively

Test Name ↓	CA_{51}	CA_{65}	CA_{129}
Frequency	Pass	Pass	Pass
Block Frequency	Pass	Pass	Pass
Cumulative Sums	Pass	Pass	Pass
Runs	Pass	Pass	Pass
Longest Runs	Pass	Pass	Pass
Rank	Pass	Pass	Pass
Spectral	Pass	Pass	Pass
Non-overlapping Template Matchings	Pass	Pass	Pass
Overlapping Template Matchings	Pass	Pass	Pass
Universal Statistical	Pass	Pass	Pass
Approximate Entropy	Pass	Pass	Pass
Random Excursions	Pass	Pass	Pass
Random Excursion Variant	Pass	Pass	Pass
Serial	Pass	Pass	Pass
Linear Complexity	Pass	Pass	Pass

Table 5.15: The results of CA_{51}, CA_{65} and CA_{129} with Diehard battery tests. Window sizes are 24, 32 and 64 respectively with seeds 1005843561, 1178288166 and 823902801 respectively

Test Name ↓	CA_{51}	CA_{65}	CA_{129}
Birthday spacings	Pass	Pass	Pass
Overlapping permutations	Fail	Fail	Fail
Binary rank 31×31	Pass	Pass	Pass
Binary rank 32×32	Pass	Pass	Pass
Binary rank 6×8	Pass	Pass	Pass
Bitstream	Pass	Fail	Pass
Monkey Test OPSO	Pass	Fail	Pass
Monkey Test OQSO	Pass	Fail	Fail
Monkey Test DNA	Fail	Fail	Pass
Count the 1s (stream)	Pass	Pass	Pass
Count the 1s (specific bytes)	Pass	Pass	Fail
Parking lot	Fail	Fail	Fail
Minimum distance	Pass	Pass	Pass
3D spheres	Pass	Pass	Pass
Squeeze	Pass	Pass	Pass
Overlapping sums	Pass	Pass	Pass
Runs	Pass	Pass	Pass
Craps	Pass	Pass	Pass

platform with similar settings. The list of 27 PRNGs selected for comparison is shown below. In case of Rule 30 [222], a 101-cell periodic boundary CA is

Table 5.16: List of some good PRNGs

List of 26 good PRNGs	
Class of PRNGs	**Name of the PRNGs**
Linear Congruential Generators (LCGs)	Knuth's `MMIX` [366], `rand`, `lrand48` in GNU C Library [370], `MRG31k3p` [371] and `PCG-32` [372]
Linear Feedback Shift Registers (LFSRs)	`random` in GNU C Library [370], `Taus88` [373], `LFSR113`, `LFSR258`, `WELL512a` and `WELL1024a` [374], `xorshift32` [375], `xorshift64*`,`xorshift1024*M_8`, `xorshift128+` [376], `MT19937-32`, `MT19937-64` [67], `SMFT19937-32`, `SFMT19937-64` [68], `dSFMT19937-32` and `dSFMT19937-64` [69]
Cellular Automata (CAs)	Rule 30 with CA size 101 [222], Maximal Length CAs with 1 cell spacing ($\gamma = 1$) and no cell spacing ($\gamma = 0$) [61], non-linear 2-state CA [215], 3-state CA-based PRNG [52] and 10-state CA-based PRNG [52]

used where we have extracted the next state of the 51th cell for consecutive 32 configurations to form a single 32-bit integer.

Now, to compare our PRNGs with these PRNGs, we need to select some seeds. For ease of reference, we choose the seeds hard-coded in the standard implementation of the PRNGs, that is, 7, 1234, 12345 and 19650218. To maintain uniformity in testing, for each of these PRNGs along with our proposed PRNGs, a stream of binary sequence generated by the PRNG is tested with all these seeds. If a PRNG, like `LFSR113, LFSR258,` etc. requires more than one (non-zero) seed to initialize its components, we supply the same seed to all of its components.

Table 5.17 shows the summary of results of Diehard, TestU01 and NIST testbeds for all these PRNGs with the fixed seeds. In this table, for each PRNG, we show results in terms of the numbers of tests passed. As one proposed PRNG, result of CA_{51} which generates 24-bit output numbers is reported. It is noticed that, each of the existing PRNGs fails to pass the *overlapping permutation* and *parking lot* tests of Diehard (see Table 5.17). Further, it can be noted that, our proposed PRNGs have better performance than most of the existing PRNGs. Moreover, considering the strength of these PRNGs in terms of portability, large cycle length, robustness and ease of hardware implementation, they have potentiality to be the most popular choice for all intended applications.

One of the well-known general purpose PRNGs is *Mersenne Twister* and its variants [67–69] where the period length is a *Mersenne* prime. Now, similar cycles can also be generated by a maximal length CA of size n, where the

Table 5.17: Comparison of existing PRNGs with our proposed PRNG using Diehard, TestU01 and NIST for 4 fixed seeds.

Seeds →	7			1234			12345			19650218		
Name of the PRNGs ↓	Diehard	Testu01	NIST	Diehard	Testu01	NIST	Diehard	Testu01	NIST	Diehard	Testu01	NIST
MMIX	6	19	7	5	18	7	6	17	8	4	16	8
rand()	1	1	2	1	1	2	1	3	2	1	2	2
lrand48()	1	3	2	1	2	2	1	3	2	1	3	2
MRG31k3p	1	2	1	1	1	1	1	2	1	1	1	1
PCG-32	9	25	15	9	25	14	11	25	14	10	24	15
random()	1	1	1	1	1	1	1	3	1	1	2	1
Tauss88	11	21	15	9	23	15	11	23	15	11	23	14
LFSR113	5	6	1	11	23	14	9	23	15	7	23	14
LFSR258	0	0	1	0	5	2	1	5	2	1	5	2
WELL512a	9	23	15	10	23	14	10	23	15	8	23	15
WELL1024a	9	25	15	10	24	15	9	24	14	9	25	15
MT19937-32	10	25	13	9	25	13	9	25	14	9	25	15
MT19937-64	10	25	15	10	24	15	8	24	15	11	25	15
SFMT-32	10	25	15	9	25	15	10	25	15	9	25	15
SFMT-64	11	25	15	10	25	15	10	25	15	9	25	15
dSFMT-32	7	25	15	8	25	15	11	24	13	11	25	15
dSFMT-64	0	5	1	0	5	1	0	4	1	0	5	1
XORShift32	4	17	4	4	17	4	4	17	2	0	17	13
XORShift64	10	25	15	10	25	15	8	25	15	7	25	15
XORShift1024	7	20	6	9	21	15	7	20	15	8	20	15
XORShift128	9	25	14	9	25	14	10	24	15	10	25	15
Rule 30	11	25	15	10	25	15	9	25	15	8	25	15
Maximal Length CA ($\gamma = 0$)	2	12	10	0	12	11	1	12	11	1	12	11
Maximal Length CA ($\gamma = 1$)	4	17	14	3	16	14	3	17	14	4	15	14
Non-linear 2 state CA	6	11	4	8	10	2	5	12	3	5	12	4
3-state CA	3	12	6	3	12	6	3	11	5	2	11	4
10-state CA	9	25	15	10	25	15	11	25	15	11	25	15
Proposed CA $\mathbf{CA_{51}}$	**10**	**25**	**15**	**9**	**25**	**15**	**10**	**25**	**15**	**9**	**25**	**14**

corresponding cycle length $2^n - 1$ is prime, that is a Mersenne prime. Hence maximal length CAs can be used as a special category of prime generator - Mersenne prime generator. However, if the CA is not a maximal length, then the cycle length , if it is a prime, can not be a Mersenne prime, but some other primes. This observation steers us to think whether CA is capable of generating a cycle of arbitrary prime length. We also explore if there exists a collection of CAs of size n which can generate cycles of lengths of all possible primes in between 2^{n-1} and $2^n - 1$.

5.6 Cellular Automata as Prime Generator

Some natural phenomena exhibit the properties of prime number; for example, cicadas emerges after a prime period. But till today, it is unknown how the prime numbers are generated. The count of primes within a range of integers is empirically computable; however, there is no proven formula to calculate the exact number of primes between any two integers. It has already been seen in Chapter 4, the irreducible cellular automata produce *prime* length cycles. This motivates us to investigate a very challenging and hard problem of whether *an n-cell CA can generate a cycle of an arbitrary prime length l*. We are also motivated to explore that do there exist a collection of CAs of size n which can generate cycles of lengths of all possible primes in between any two integers.

A cycle of length l can be generated by CAs of various sizes, out of which the minimum (optimal) sized one is the maximal length CA of size $(\lfloor \log l \rfloor + 1)$. We therefore focus to explore the optimal CA of size $(\lfloor \log l \rfloor + 1)$ which has l length cycle, where l is *prime*. We call such cycle as *large length prime cycle* and the corresponding CA as *large prime length* CA. To study such large prime length CAs, we use here reversible and non uniform CAs. Moreover, it is already established that a maximal length CA can not be obtained from other CAs; its *cycle structure* can not be *decomposed* further. We call such type of CA as *irreducible* CA (See Chapter 4). Therefore, to explore large prime length CAs, more specifically, we need to use irreducible CAs - the backbone of any large prime length CA. As large length is also concerned, therefore, we need to figure out the distribution of rules based on the parameter `P` values for designing large prime length cellular automata. This chapter empirically presents that an n-cell irreducible CA can also be used as an arbitrary large prime generator. Our work also establishes that there exists an n-cell large prime length CA for *each* prime in between 2^{n-1} to $2^n - 1$. .

5.6.1 Insignificant Rules

It is already noted that one of the aspects of this research is to find a CA of size $O(\log l)$ for generating a cycle of length l where l is prime and $l \in [2^{n-1}, 2^n - 1]$. We call such CA as ***large prime length CA***(LPLCA). Moreover, we are also interested to study that CAs of size n can generate all possible cycles of prime length in the range of $[2^{n-1}, 2^n - 1]$. We also focus on some specific region of prime numbers which can be mapped to CA. To study all these problems, the CAs should have following properties.

- An n-cell large prime length CA must be *irreducible* CA.
- An n-cell large prime length CA must be *large cycle* CA.
- For an n-cell large prime length CA, the maximum cycle length must be *prime*.

In primary selection, we identified those rules out of 62 rules, which can never contribute in irreducible CAs. If we revisit, there are such 10 rules (204, 51, 102, 153, 60, 195, 15, 240, 85, 170), which are named as *independent*. Therefore, they never participate for generating CAs with large prime length cycles. These CA rules are *insignificant* for generating such irreducible CAs. The following corollary is useful for generating large prime length CAs.

Corollary 5.1 *An n-cell CA* $(\mathcal{R}_0, \mathcal{R}_1, \cdots \mathcal{R}_{i-1}, \cdots \mathcal{R}_{n-1})$ *never forms a cycle of prime length which is larger than* 2^{n-1} *if* $\mathcal{R}_i$ *(for any* i*) is selected from the set of right independent and left independent rules.*

Proof: Let us assume, without loss of generality that $\mathcal{R}_{i-1}$ is right independent (similar argument holds when $\mathcal{R}_{i-1}$ is left independent). The length l of any arbitrary cycle is a multiple of l_p, the length of some cycle of i-cell CA $\mathcal{R}^1_0, \mathcal{R}^1_1, \cdots \mathcal{R}^1_{i-1}$. So, l is prime iff l_p is prime and l equals l_p. However, $l_p \leq 2^{n-1}$. □

As our target is on *large* prime length CAs, the necessary condition is that the CA is *large cycle* CA. An n-cell CA is called as large cycle CA if the maximum cycle length of that CA is larger than 2^{n-1}. This characteristic is to be found in such CA where maximum number of CA rules have *higher degree of dependence on their neighbors* and the synthesis of such CAs are reported in Section 5.2.2. Though in the experimental results of the synthesis of large cycle CA (which is reported in Table 5.5), there is no independent rule which contributes for large cycle CA, it does not mean that no strictly reducible is large cycle CA.

Next, we try to figure out the contribution of the remaining 52 rules for our problem domain. Already such 52 rules are categorized into three categories - *completely dependent*, *partially dependent* and *weakly dependent*. We study how these categories are effective for generating large prime length CA. Firstly, we consider the scenario where all the rules are from completely dependent category only.

Observation 5.1 *If all the rules of an n-cell CA are selected from the completely dependent category, then such CA can generate the large prime length cycle only when* $2^n - 1$ *is prime.*

From the above observation, it is clear that for generating a large prime length cycle of an arbitrary size CA, the rules can not be selected only from completely dependent category. Thus corollary 5.1 and Observation 5.1 imply that no linear CA can generate large prime length cycle (other than cycles whose length are Mersenne prime). Next we study the behaviour of CAs where rules are selected not only from completely dependent category but also from partially dependent category.

Lemma 5.1 *Suppose the rules of an n-cell CA are selected from completely dependent and partially dependent categories. Then the number of rules selected from partially dependent category is at most* $n/2$*.*

Proof: We refer to Table 5.4c on generating an n-cell CA consisting of rules that are members of completely dependent and partially dependent categories. It can be verified that the maximum number of rules gets chosen from the partially dependent class if the rules are selected using the following manner. Suppose $\mathcal{R}_1$ is selected from class II and partially dependent category. Therefore, $\mathcal{R}_2$ must be selected from class I. The rules of class I are members of completely dependent and partially dependent categories. Let us take $\mathcal{R}_2$ from partially dependent category. The next rule $\mathcal{R}_3$ will be selected from class VI which only contains the rules of completely dependent category and the class of $\mathcal{R}_4$ is V. Class V contains rules from completely dependent and partially dependent categories where next class is again class VI. Thus the rules are selected such that $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_4, \mathcal{R}_6 \cdots \mathcal{R}_p$ are from partially dependent category, where $p = n-2$ when n is even or $n-3$ otherwise. $\square$

Next we study the nature of the CAs when the rules are selected from the completely dependent and the weakly dependent categories. The following lemma shows that there can not be too many rules from the weakly dependent category in a CA that is generated from rules selected only from the completely dependent and the weakly dependent categories.

Lemma 5.2 *Suppose the rules of an n-cell CA are selected from completely dependent and weakly dependent categories. Then the number of rules selected from weakly dependent category is at most one.*

Proof: We refer to Table 5.4c on generating an n-cell CA consisting of rules that are members of completely dependent and weakly dependent categories. We see that only class I and class III contain rules from weakly dependent categories. Let $\mathcal{R}_i$ be the first weakly dependent rule in the CA, where $1 \leq i \leq n-2$. If $\mathcal{R}_i$ is selected from class I or class III, then $\mathcal{R}_{i+1}$ must be selected from class IV or class V respectively. We further observe from Table 2.7a that if $\mathcal{R}_{i+1}$ is a completely dependent rule selected from class IV or V, then all the subsequent rules $\mathcal{R}_{i+2}$ to $\mathcal{R}_{n-1}$ are selected from completely dependent category only. Therefore, only $\mathcal{R}_i$ can be member of weakly dependent category in the CA. $\square$

Finally, we consider the scenario where the rules of the CA are selected from all the three categories - weakly dependent, partially dependent and completely dependent. The following lemma shows that the maximum count of weakly dependent rule in such CA is restricted to one.

Lemma 5.3 *Suppose the rules of an n-cell CA are selected from completely dependent, partially dependent and weakly dependent categories. Then the count of rules selected from only weakly dependent category is less than two.*

Proof: We refer to Table 5.4c on generating an n-cell CA consisting of rules that are members of completely dependent, partially dependent and weakly dependent categories. We see that only class I and class III contain rules from weakly dependent categories. Let $\mathcal{R}_i$ be the first weakly dependent rule in the CA, where $1 \leq i \leq n-2$. If $\mathcal{R}_i$ is selected from class I or class III, then $\mathcal{R}_{i+1}$ must be selected from class IV or class V respectively. We further observe from Table 2.7a that if $\mathcal{R}_{i+1}$ is a completely dependent rule or partially dependent rule selected from class IV or V, then all the subsequent rules $\mathcal{R}_{i+2}$ to $\mathcal{R}_{n-1}$ are selected only from either completely dependent category or partially dependent category as they are selected from classes IV, V and VI. No rule from classes IV, V and VI is member of weakly dependent category. It is also deduced from Table 2.7a that if any rule is selected from classes IV, V and VI, then next rule can never be selected from class I or class III. Therefore, there is no chance of getting more than one rule from weakly dependent category if $\mathcal{R}_i$ is a member of weakly dependent category in the CA. □

5.6.2 Empirical Facts

In this section, we study two related problems. First, we target to design CAs that generate large prime length cycles. The other problem is to figure out if for every prime number in the range of $[2^{n-1}, 2^n-1]$, there exists some n-cell CA that generates a cycle with length as that prime number. Both of these problems are addressed by designing CAs with rules from the completely dependent, partially dependent and partially dependent categories.

In our experiments, we follow sampling with replacement, where each element of the sample is drawn as follows. Each element (CA) of the sample is formed from rules selected from the given categories. If the rule at any cell position is a member of class IV or VI, then it is be selected from completely dependent category only. If the rule belongs to any other class, let z be the probability of selecting the rule from partially dependent category. Once the category of the rule is selected for a cell position, the rule is chosen uniformly at random among all the rules that belong to that category. If the rule is selected from class I or III, there may be a chance to select the rule from weakly dependent category and this is for only one time. An element of the sample is said to be *good* if the corresponding CA generates a large length prime cycle that is not of maximal length. The success rate (SR) is determined by the ratio of the count of good elements of sample to the total number elements in the sample that generate large length cycles. Here, we have taken a snapshot of our experiments where CA sizes vary from seven to fifteen. The sample size for a particular CA size is selected by making a run with an initial sample size of 10,000, and doubling the sample size in every run till the success rates do not significantly vary over two consecutive runs.

Our first set of experiments study the nature of generation of large prime

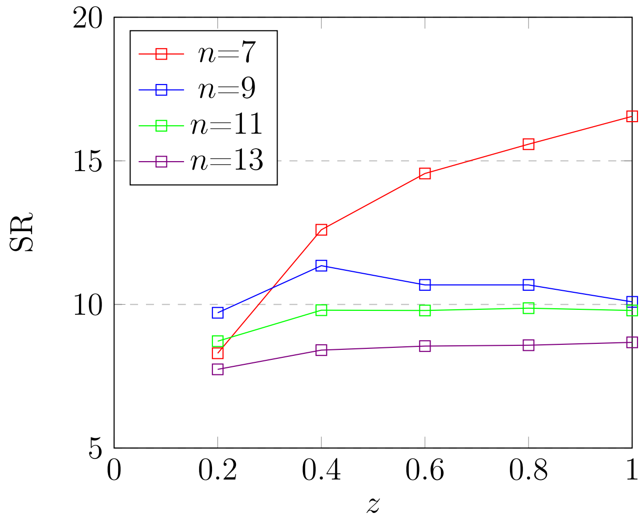

Figure 5.7: Distribution of # LPLCAs by considering rules from completely dependent and partially dependent categories

length cycles by varying the number of rules from the partially dependent category in the CAs. We have done experiments for different sizes of CA. In figure 5.7, for each CA size, we plot the success rate with varying z. We observe from the results that the success rate increases with increase in value of z. Recall that from Observation 5.1, a CA with rules from only completely dependent category forms a large prime length cycle that is of length Mersenne prime. Therefore, with selecting more rules from partially dependent category, there is a higher chance of getting a CA that generates large prime length cycle (which is not a maximal length cycle).

Since a CA can be composed of at most one rule from weakly dependent category, we vary the cell position of that weakly dependent rule to study the nature of the CA for generating large prime length cycles. When a CA is constructed by the rules from completely dependent and weakly dependent categories, we vary the cell position of that weakly dependent rule to study the nature of the CA for generating large prime length cycles. As before, we use sampling with replacement: the sample sizes determined as described above. Let i be the position where the rule from weakly dependent category is selected, which means that each element (CA) of the sample is formed from weakly dependent rule at cell position i and completely dependent rules at other positions. Once the category of the rule is selected for a cell position, the rule is chosen uniformly at random among all the rules that belong to that category. As before, we have done experiments for different sizes of CA from seven to fifteen. For each CA size, we have performed evaluations by varying i from 1 to $n-2$. In figure 5.8, for each CA size, we plot the success rate with varying i. We observe from the

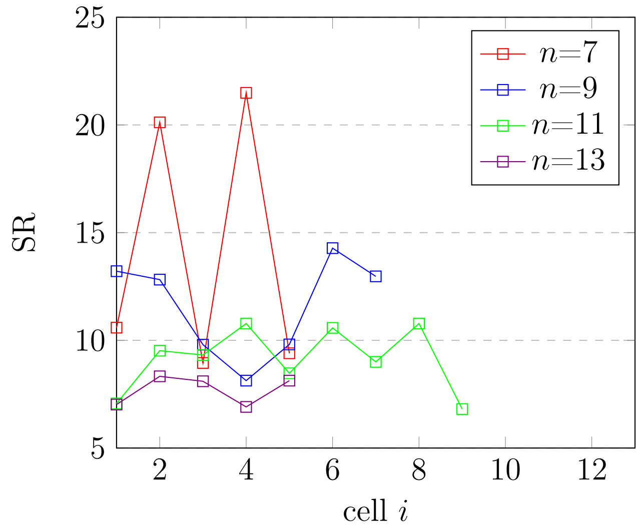

Figure 5.8: Distribution of # LPLCAs by considering rules from completely dependent and weakly dependent categories

results that the success rate is maximum if i equals $n-3$ (see figure 5.8).

To study the nature of n-cell CA for generating large prime length cycle, we use different proportions of rules from completely, partially and weakly dependent. For every element of the sample, the category of the rule at each cell position is chosen as follows: (i) if the rule belongs to class IV or VI, the rule must belong to completely dependent category (ii) if the rule belongs to class II or V, the rule is selected with probability z from the partially dependent category and $1-z$ from the completely dependent category. It is already shown in Figure 5.7 that the success rate improves by increasing z. (iii) if the rule belongs to class I or III, it is selected with probability z from the partially dependent and $1-z$ from the completely dependent category. Once the cell position is $n-3$, the corresponding rule is selected from uniformly at random from weakly dependent category. Using Figure 5.8, it is deduced that the success rate is maximum when we place a weakly dependent rule only at cell position $n-3$. In Figure 5.9, for each CA size, we plot the success rate with varying z. We observe from the results that the success rate is maximum if z equals 0.8.

We next present the results for the problem where we figure out if all the prime numbers between 2^{n-1} and 2^n-1 can be generated by CAs of size n. In particular, we find the distinct primes $p_{expt}(n)$ in the range of $[2^{n-1}, 2^n-1]$ that can be generated as cycle lengths of different n-cell CAs. For any n, we study the relation between $p_{expt}(n)$, $p_{app}(n)$ and $p_{act}(n)$. In case $p_{expt}(n)$ exceeds $p_{act}(n)$, we can conclude that our scheme of designing CAs can be used as prime number generator, i.e., every prime number between 2^{n-1} and 2^n can be generated as a

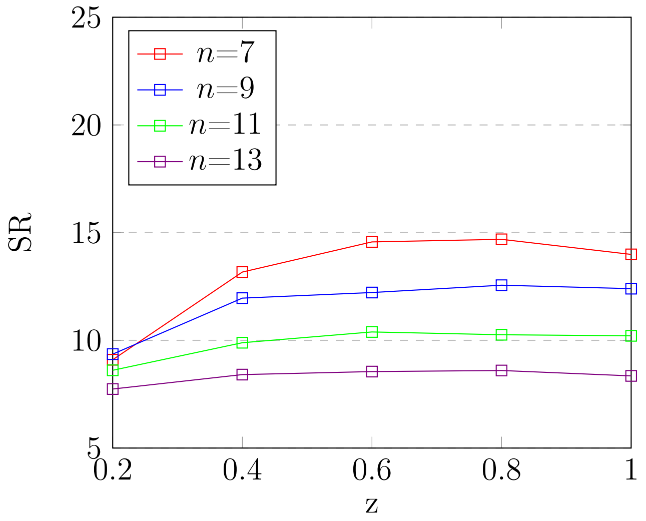

Figure 5.9: Distribution of # LPLCAs by considering rules from completely dependent, partially dependent and weakly dependent categories

cycle length of some n-cell CA. Since it is possible to determine $p_{act}(n)$ only for small n, we may relax the above criterion by checking if $p_{expt}(n)$ exceeds $p_{app}(n)$.

We study the design of n-cell CAs with rules from weakly dependent, partially dependent and completely dependent categories. Recall that in this sampling procedure, for every element of the sample, the category of the rule at each cell position is chosen as follows: (i) if the rule belongs to class IV or VI, the rule must belong to completely dependent category (ii) if the rule belongs to class II or V, the rule is selected with probability z from the partially dependent category and $1 - z$ from the completely dependent category. (iii) if the rule belongs to class I or III, it is selected with probability z from the partially dependent and $1-z$ from the completely dependent category. Once the cell position is $n-3$, the corresponding rule is selected from uniformly at random from weakly dependent category. For each CA size n, we determine $p_{expt}(n)$ with varying z (as shown in Table 5.18). Here, we have taken a snapshot of my experiments.

5.7 Summary

In this chapter, we have studied the problem of designing a near optimal cellular automaton which can produce a cycle of length l. We have shown that this is an optimization problem, and used an evolutionary approach to solve this problem. This chapter also introduced a notion of large length cycle CA and a stochastic approach is described for generating such CA. As an application of large length cycle CAs, we have reported a design of window-based PRNGs. It is observed

Table 5.18: Experimental results for determining the number of distinct prime cycle lengths generated by CAs with rules of completely dependent, partially dependent and weakly dependent categories

n	$z = 0.2$	$z = 0.4$	$z = 0.6$	$z = 0.8$	$z = 1$	$p_{app}(n)$	$p_{act}(n)$	$p_{act}(n)$
7	13	13	13	13	13	11	13	13
9	41	43	43	43	36	36	43	43
11	136	137	137	137	137	121	137	137
13	464	464	464	464	464	417	464	464

that, performance of our proposed PRNGs are better than many of the well-known PRNGs existing today. Further, considering the inherent advantages of our scheme, we claim that, these CAs-based PRNGs are good contenders to be the more suitable random number generators for different technologies. We have verified the randomness quality of these CAs as PRNGs by using Diehard battery of tests, battery rabbit of TestU01 library and NIST statistical test suite. The research reported in this chapter also shows that CA of size n can act as prime number generator for an arbitrary prime number in the range $[2^{n-1}, 2^n - 1]$. The design of optimal CA can be generalized for any d-state r-radius non-uniform CAs.

CHAPTER 6

Reversible Cellular Automata: A Natural Clustering Technique

In Chapters 2 to 5, we have studied the behaviour of finite reversible cellular automata (CAs) in terms of cycle generation like how many cycles a cellular automaton (CA) has, how many equal length cycles a CA can produce, what is the cycle structure of a CA, whether the cycle structure (CS) of a CA can be decomposed or not, for a given length of cycle figuring out the best possible CA, introduction of CA as prime number generator. Additionally, we focus on the practical usability of large length cycles CAs and we have achieved the goal by designing efficient pseudo random number generators (mentioned in Chapter 5).

In the cyclic space of reversible cellular automaton, those configurations are reachable from each other which are in a cycle. Hence, these configurations are *related* from reachability point of view. In the clustering problem, the related target objects are in same cluster. In case of CA, the objects (configurations) belong to the same cycle, can be considered as *close* to each other. Therefore, a reversible finite cellular automaton can be thought of as a *natural* clustering tool, where *closeness* between two configurations is measured as whether they are *reachable* from each other. Based on 'reachability', this dissertation diagnoses the properties in CAs which make the clustering *effective.* In our work, first we use non-uniform ECAs as the candidate CAs for the clustering technique. Next, we study uniform reversible cellular automata in clustering problem where the radius of the CA is more than *one.*

6.1 Introduction

Clustering [70, 377–379] is a well studied research topic where data objects are grouped together based on some similarity measuring metric. The aim of a clustering technique is to distribute the target objects (data records) among the clusters *effectively*. This metric also exploits the inherent anatomy of data objects for partitioning the *dissimilar* objects into separate clusters. There exists several important result about this clustering problem. On one hand, it has been shown theoretically that there are no efficient exact solutions for clustering methods. On the other hand, different heuristics have been proposed till date; these include genetic algorithm based modified gravitational search algorithm [380], the quantum chaotic cuckoo search algorithm [381], particle swarm optimization algorithm [382], bee colony technique [383], hybrid water cycle algorithm [384], even using stochastic cellular automata [349]. A cluster also establishes an intrinsic connection among the objects. This connectivity indicates us to think cellular automata (CAs) as natural clusters.

Any clustering technique is mainly driven by two influencing factors - less *intra-cluster* distance and *optimal* number of clusters. In this work also, our primary target is to diagnose such features in cellular automata. Therefore, one of the objectives of our basic model is to select such *significant* CA rules for generating CAs which are capable of connecting the configurations which maintain minimum possible change in their cells' state values (here, the CA configurations are used as the data objects). For ensuring that a CA has limited number of cycles, we use the framework of a related problem - CA with large length cycles (introduced in Section 5.2.2 in Chapter 5).

Since each object is represented by a finite set of features, finite CAs are suitable to represent such objects. Definitely, generating a CA for a fixed n maintaining less intra-cluster distance and limited number of clusters is a very challenging problem; moreover, distributing the target objects among the desired number of clusters is difficult to ensure. The inherent hardness of this problem motivates us to take an iterative strategy which distributes the target objects in m clusters. In the proposed algorithm, the clusters of level i are generated by *merging* a set of clusters of level $i-1$ which are *closely reachable* (which we define in Section 6.5). To measure the quality of clusters, some benchmark *internal* cluster validation indices [71] are used. This chapter presents the performance analysis of our proposed cycle based clustering algorithm on some real datasets taken from ML repository (`http://archive.ics.uci.edu/ml/index.php`). Finally, we compare our proposed algorithm with some traditional benchmark clustering algorithms like *partitional clustering*, *hierarchical clustering* [70, 71]. Our results indicate that, performance of our CA-based clustering technique is at least as good as the best known clustering algorithms existing today. This research also discovers that our CA-based clustering scheme owns such characteristic by which the arrangement of clusters remains *fixed* even if the ordering of the features of a given set of objects is changed. Such property is verified with the real datasets used in this work. In our CA-based clustering technique, we use *two* sets of CAs. First, we experiment using the non-uniform reversible CAs with radius (r) *one*. For further improvement through generation of a large variety of reversible CAs, we increase the radius of CA. However, increasing the radius to higher numbers results in generation of extensive number of reversible CAs. For our experiments, we set the radius value to *two* and we choose only uniform reversible CAs. Next, we take a tour on the clustering algorithms.

6.2 A Brief Note on Clustering

For assigning the target objects, traditionally, *hard* clustering [377, 378] technique is used where a target object is placed in a unique cluster. Let us consider a set of target objects $\mathbb{X} = \{\mathtt{X}_1, \mathtt{X}_2, \cdots, \mathtt{X}_w\}$; every object can be represented as $\mathtt{X} = (x^1 x^2 \cdots x^p)$ where x^j is said to be a feature. Let there exist non-empty m ($m < w$) clusters C_1, C_2, $\cdots$, C_m such that the following conditions are satisfied to place the target objects: (1) $\bigcup_{i=1}^{m} C_i = \mathbb{X}$ and (2) $C_i \cap C_j = \varnothing$, $i, = 1, \cdots, m$ and $i \neq j$. However, an object can be assigned to each cluster with a degree of membership. Such category is *fuzzy* clustering. To perform clustering effectively, a sequence of steps are followed: (i.) feature selection or extraction, (ii.) choosing or designing clustering algorithm and (iii.) cluster analysis. Using *feature selection* [377,385,386], optimal subset of useful features is found from the candidate features and such mechanism reduces the workload for designing the cluster. On the other hand, *feature extraction* generates novel features by some metamorphism of the original features. The next step of the clustering is to design functions on *proximity* measure and *grouping* of the target objects as an optimization problem. Proximity measure function quantifies the similarity or dissimilarity between a pair of objects based on their features. Some popular measures are Minkowski distance, Standardized Euclidean distance, Cosine distance, Pearson correlation distance, Mahalanobis distance, Jaccard similarity, Hamming similarity, City-block distance, Point symmetry distance, etc., [70,378]. Additionally, Hausdroff distance [387,388], edit or Levenshtein distance [378,389,390], mutual neighbor distance [391]. The objective of any clustering algorithm is to minimize the distance (given by the above measures) between objects that belong to the same cluster. Such distance is called as *intra-cluster* distance.

Traditionally, we use *partitional* and *hierarchical* algorithms. In hierarchical clustering, the objects maintain hierarchical relationship [430] and form groups

Table 6.1: A summary table on clustering techniques

Category of Clustering	Name of the Algorithms
Hierarchical	Single-linkage [392], complete-linkage [393], BIRCH [394], CURE [395], ROCK [396], Chameleon [397], DIANA [398], SOTA [399]
Partitional	k-means [400], k-medoids [401], PAM [398], CLARA [398], CLARANS [402], ISODATA [378, 403]
Density based	DBSCAN [404], DENCLUE [405], OPTICS [406], Mean shift [407]
Distribution based	DBCLASD [408], GMM [409]
Graph-theory based	CLICK [410], CAST [411], DTG [412], HCS [378]
Grid based	STING [413], CLIQUE [414], WaveCluster [415]
Model based	COBWEB [416], GMM [409], SOM [417], ART [418]
Fuzzy	FCM [419–421], FCS [422] and MM [423].
Modern	Kernel K-means [424], kernel SOM [425], kernel FCM [426], SVC [427], MMC [428], MKC [429]

(clusters) in a nested series of partitions [377] where the clusters are formed either in *agglomerative* manner (the clusters are merged which have less distance and initially, all w objects are in w distinct clusters) or they follow *divisive* approach (the objects get split into multiple clusters but at initial phase all objects are kept only in one cluster). Single-linkage [392] and complete-linkage [393] are two famous hierarchical clustering techniques. In case of single linkage technique, the distance between two clusters is defined as the *minimum* distance among all the pairs of objects in such two corresponding clusters. For complete-linkage, the cluster distance is considered as the maximum distance among the same pairs of objects. Besides single and complete linkage clustering, there are some advanced hierarchical clustering techniques like BIRCH [394], CURE [395], ROCK [396] and Chameleon [397] which can handle large datasets. All the algorithms mentioned on hierarchical clustering are examples of agglomerative clustering. Divisive Analysis (DIANA) [398] clustering uses the reverse approach of agglomerative. Generally, divisive method is computationally demanding [378] as $2^{w-1}-1$ possibilities are there for w target objects. DIANA uses some heuristic which makes it as a successful clustering tool for all kinds of urban roads with varying traffic loads [431]. The Self-Organizing Tree Algorithm (SOTA) [399] is a hierarchical divisive clustering technique which uses binary tree structure. This algorithm uses the combination of Kohonen self-organizing map [417] and the growing cell structures algorithm [432] as the backbone. This algorithm is capable of handling large volume datasets. In partitional clustering, the number of clusters is pre-defined and in this technique criterion function plays a key role [433]. Most popular criterion function is the squared error [378, 434]. One of the demanding algorithms based on squared error criterion is k-means technique. The principle of this algorithm is to update the *centre* of cluster and to minimize the within-class sum of squares (intra-cluster distance is minimized) [71, 377]. As k-means uses Euclidean distance, it has the tendency to generate hyperspherical cluster [378]; therefore, for clusters with other geometrical shapes can not be clustered properly. Several variants of k-means [400] have been designed.

k-medoids [401] is an improved version of k-means. Here, *medoid* is the most centrally located object in a cluster from which the average dissimilarities to all objects of that cluster is minimal. *Partitioning around medoids* (PAM) [398] is an example of k-medoid clustering technique which considers all possible objects in the cluster for figuring out the next medoid with minimum error sum of squares. Some other partitional algorithms are CLARA [398], CLARANS [402]. CLARA uses random sampling approach and applies PAM on samples for figuring out menoids [71]. CLARA is useful for large datasets. For getting the betterment on the performance of CLARA, one can use CLARANS which uses a sample of neighbors at every step of the search whereas CLARA uses a sample of nodes at the beginning. Another squared error based clustering technique is ISODATA [378, 403], which is used to adjust the count of clusters by merging and splitting the objects.

Besides hierarchical and partitional clustering, some other popular clustering methods are density based clustering (a collection data which together form high dense area in the dataspace, form a cluster) [435], clustering based on distribution (the collection of objects which follow same distribution in the dataspace form cluster if diverse distributions are occurred in the dataspace) [70,378], clustering based on graph theory (node of the graph is object and an edge is the relationship between two given objects) [70,378], grid based clustering [70,378] (the dataspace is considered as a grid structure), clustering based on model [70,378], etc. DBSCAN is the most popular density based clustering algorithm [404]. Other notable density based clustering algorithms are DENCLUE [405], OPTICS [406], Mean shift [407] (see [436–438] for more details). DBCLASD [408] and GMM [409] are the examples of distribution based clustering technique. The most popular graph theory based clustering algorithm is clustering identification via connectivity kernel (CLICK) [410]; it generates clusters based on minimum eight cut. Chameleon, cluster affinity search technique (CAST) [411], Delaunay triangulation graph (DTG) [412] and highly connected subgraph (HCS) are also based on graph theory approach [378]. Some examples of grid based clustering are STING [413], CLIQUE [414], WaveCluster [415] and fractal clustering [439]. In model based clustering, mainly statistical based learning and neural network based learning are followed. COBWEB [416] and GMM [409] follow statistical based learning whereas SOM [417] and ART [418] are the examples of neural network based clustering algorithms. Till now, we have discussed about hard clustering. Another category of clustering i.e. fuzzy clustering [440–442] distributes every pattern in all clusters with a degree of membership. Some of the notable fuzzy based algorithms are FCM [419–421], FCS [422] and MM [423]. Besides all the above mentioned clustering techniques, there are some modern clustering algorithms. One category is based on kernel; the examples are Kernel K-means [424], kernel SOM [425], kernel FCM [426], SVC [427], MMC [428] and

MKC [429]. Clustering can be done based on the change in biological population [443–449]. Clustering can be done based on quantum theory also [450, 451]. Table 6.1 represents a chart of algorithms based on the categories of clustering.

For any clustering system, the foundational aspects are the number of clusters and the quality of clusters [452] i.e. how *good* the clusters are? For validation of the clusters, there are several metrics [70, 71, 453]. Generally, for the validation cluster, two way assessment can be done: in one approach, the quality of the clustering judged by the intrinsic information of the objects [71] and this technique is referred as *internal* assessment; whereas, in *external* assessment, the recovered structure is compared to a priori structure [377]. In internal cluster validation, the datasets and the clusters are used as the input to measure the quality of the clustering. A special case of internal validation i.e., *stability measure* checks the consistency of a clustering by comparing the clustering keeping all data and removing each column [71, 454]. Biological validation measures how the clustering is capable of producing biologically meaningful clusters [71]. Some of the popular internal validation indices are *Davies-Bouldin* index [455], *Dunn* index [456], *Calinski-Harabasz* index [457], *silhouette* index [458], *connectivity* [459], *Xie-Beni* index [460].

Next, we introduce how the properties of reversible CAs make them effective for clustering.

6.3 Cellular Automata and Clustering

In hard clustering, one object is included in only one cluster. This may lead to think that a clustering technique can be as a bijective function $A : D \to D$, where D is a (finite) domain of objects to be clustered. Two elements $\mathtt{x}$ and $\mathtt{y}$ are *close* to each other with respect to A if $A^{l_1}(\mathtt{x}) = \mathtt{y}$, $A^{l_2}(\mathtt{y}) = \mathtt{x}$ for some $l_1, l_2 \in \mathbb{N}$. On the other hand, two elements $\mathtt{x}, \mathtt{z}$ are *not close* to each other if $A^{l}(\mathtt{x}) \neq \mathtt{z}$ (or, $A^{l'}(\mathtt{z}) \neq \mathtt{x}$) for all $l, l' \in \mathbb{N}$. Here, the function A measures *closeness* of objects in some sense to result in clustering of elements of D. Obviously, different function may result in different clustering of the same data. Therefore, for any given dataset, the real challenge lies in finding the appropriate bijective function from the huge set of possible bijections that can cluster it according to its requirement. A reversible CA can also act as a function that maintains bijective mapping among the configurations.

6.3.1 Intrinsic Properties of Reversible CA for Clustering

A cluster is a collection of alike objects where the *closeness* between any two target objects is measured by the feature based distances (intra-cluster) among them. In this work, we propose reversible finite cellular automata (CAs) as our clustering tool. Here, the global transition function of the cellular automaton (CA) distributes the configurations into a number of clusters where each cluster constitutes of a distinct cycle. Further, to be an effective clustering, the intra-cluster and inter-cluster distance have to be lower and higher respectively. In this case also, CA offers an integral solution. In a CA, the cells update their states by looking at the states of their neighbors guided by a local *rule*. Because of this *locality* property, any change of state at a local cell flows through the configurations affecting only the neighboring cell(s) at a time step. Hence, if two configurations have less number of state changes corresponding to their individual cells, then their images will also have the same property. So, if we take a reversible CA where two configurations $\mathtt{x}$ and $G(\mathtt{x})$ have less number of changes in states, then by our closeness metric, these configurations are close as they belong to the same cycle. Because of locality property, all their successors will also be close having less number of changes in states. Therefore, for such a reversible CA, number of changes in state for any two configurations inside the same cycle is lesser than that of two configurations from two distinct cycles. This intrinsic property of such CAs guarantees that intra-cluster and inter-cluster distances are lower and higher respectively, which is essential for good clustering. In this way, the inherent structure of reversible CAs works as a *natural* clustering technique (Section 6.3.3).

Recall that each object is represented by a finite set of features; so, finite CAs are suitable to represent such objects. Moreover, each feature value is to be mapped into a binary string as this work supports binary CAs. By appending all these feature values in binary form, a binary string of length n is formed which we call an *useful configuration*. For the same object, several useful configurations can be formed if the ordering of features is changed. This reordering of features for the given objects guide us to use different CAs to cluster the same set of objects depending on the useful configuration.

Ideally, a CA of size n distributes 2^n configurations among m cycles where m ranges from 2 to 2^n. To an extreme degree, a CA based clustering can attract all target objects in one cluster or distribute among m clusters where the count of target objects is m ($\leq 2^n$) – both of which are not desirable for good clustering. Therefore, a CA is said to be *effective* for clustering if the useful configurations (objects) can be distributed among *optimal* number of cycles (clusters). As we focus on clustering using CA, firstly, each target object needs to be mapped to a *configuration*. To do that, *data discretization* should be done *effectively*.

6.3.2 The Encoding Technique

Let $\mathbb{X} = \{\mathtt{X}_1, \mathtt{X}_2, \cdots, \mathtt{X}_w\}$ be the set of target objects which are to be distributed among m clusters and $A_1, A_2, \cdots, A_p$ are p distinct attributes (features) of the objects where each A_j represents a finite set of t values. Depending on these values, we call the corresponding attribute as *quantitative* or *qualitative*. As, we use binary CA as our tool, so, each object ($\mathtt{X}$) is converted into a binary string $\mathtt{x}$ (where $\mathtt{x} \in \{0,1\}^n$ and $n \in \mathbb{N}$). Now, let $\mathtt{X} = (x^1 x^2 \cdots x^p)$ where $x^j \in A_j$. To maintain less intra cluster distance based on the feature space, the configurations having more similarity in their features' space (attributes' values) should be reachable from each other. To measure such similarity, hamming distance [70] is taken in our consideration. Next, the role of hamming distance is shown for designing the encoded target objects.

If A_j is *quantitative* attribute, the range can be partitioned into v *closed* intervals such that A_j can be represented as an ordered (either *ascending* or *descending*) list of elements. As *frequency based* encoding [461] is used for each A_j, therefore, each interval can be encoded as a binary string of length $v-1$ to maintain a *minimum* hamming distance. Let the first interval be represented as a binary string such that 0^{v-1}; the next interval is represented by $0^{v-2}1$. Obviously, the hamming distance between 0^{v-1} and $0^{v-2}1$ is *one* and the values represented by first interval is the closest with the second interval with respect to other intervals. To maintain such feature in the representation of encoded objects, the encoding function $\mathtt{M}$ can be designed using the following manner:

$$\mathtt{M}(a) = 0^{v-i}1^{i-1} \tag{6.1}$$

where $a \in [a_{i_1}, a_{i_w}]$ and $[a_{i_1}, a_{i_w}]$ is the i^{th} interval of A_j. In this work, for each quantitative feature A_j, v is set to 3; so *two* bits are needed to encode any value from A_j. Using Equation 6.1, the intervals are represented as 00, 01 and 11.

By considering A_j as *qualitative* attribute, $\mathtt{M}(a)$ $(a \in A_j)$ can be redesigned in the following manner: Let $A_j = \{a_1, a_2, \cdots, a_{u_j}\}$, then each element of A_j is represented as a binary string of length u_j where there is only one 1 at some unique position. Let a be the i^{th} element of A_j which has value 1, then $\mathtt{M}(a)$ can be represented using Equation 6.2.

$$\mathtt{M}(a) = 0^{i-1}10^{u_j-i} \tag{6.2}$$

So, the hamming distance between any pair of elements for such qualitative A_j is always fixed i.e. *two*.

Using the encoding techniques of quantitative and qualitative attributes, a target object with p features (where p_{q_n} and p_{q_l} are respectively the count of quantitative and qualitative attributes [377]) is mapped to a configuration of

Table 6.2: Encoding a set of hypothetical books into CA configurations

Object ID	Continuous Attribute				Categorical Attribute		Encoded CA configuration
	Number of Pages	Encoding	Ratings	Encoding	Binding Type	Encoding	
1	300	01	9	11	Hard	01	011101
2	325	11	8	11	Soft	10	111110
3	40	00	9.5	11	Soft	10	001110
4	200	01	4	00	Hard	01	010001
5	129	01	4.5	01	Soft	10	010110
6	65	00	7	01	Hard	01	000101
7	319	11	6.8	01	Soft	10	110110
8	110	00	3	00	Soft	10	000010
9	400	11	2.6	00	Soft	10	110010
10	350	11	9.3	11	Soft	10	111110

n-cell CA. Here, n can be computed as $n = ((v-1) * p_{q_n}) + (u_1 + u_2 + \cdots + u_{p_{q_l}})$ where $u_i = |A_i|$ $(\forall i)$ and A_i is a qualitative attribute. This research assumes v as 3. In this way, $\mathbb{X}$ is mapped to a set of *useful configurations* for clustering which is a proper subset of C. Let $\mathscr{C}$ denote the set of useful configurations and $\mathtt{M} : \mathbb{X} \rightarrow \mathscr{C}$ such that $\mathtt{M}$ is surjective. Therefore, $|\mathscr{C}| < |\mathbb{X}|$ and actually, the clustering operation is performed on less number of objects. Moreover, the maximum count of $\mathscr{C}$ can also be computed. If all the attributes are *quantitative*, then, $n = (v-1) * p$; so, $|\mathscr{C}| \leq v^p$. Otherwise, $|\mathscr{C}| \leq (v^{p_{q_n}} * \prod_{i=1}^{p_{q_l}} u_i)$.

Example 6.1 Let us take a hypothetical set of *books*, where each book is identified by three attributes – *number of pages*, *ratings by reviewers* and *type of binding*. The first two attributes are *continuous*, whereas the last one is categorical. Table 6.2 shows the detailed encoding scheme for ten such objects into CA configurations. The categorical attribute values are encoded as 01 (hard) or 10 (soft). The continuous attribute values are divided into three sub-intervals to be represented by 00, 01 and 11 respectively. For example, values in *Ratings* are divided into sub-intervals $[2.6, 4]$, $[4.5, 7]$ and $[8, 9.5]$ depicted by 00, 01 and 11 respectively. Therefore, each object is mapped to a 6-bit string which can be shown as a configurations of a 6-cell CA.

The above example mainly depicts the data encoding technique. Let $\mathbb{X}$ be the set of target elements such that $C_i \cap C_j = \varnothing$ $(\forall i, j$ and $i \neq j)$. Now, each cluster can be formed using a unique cycle of a CA. Next we present how the inherent cycles structure of CAs connect the close objects naturally into the same cluster.

6.3.3 Natural Clusters: Cycles of Reversible CA

As reversible CA can also be represented as bijective mapping, therefore, the closeness property of clustering can be defined by the reachability property of

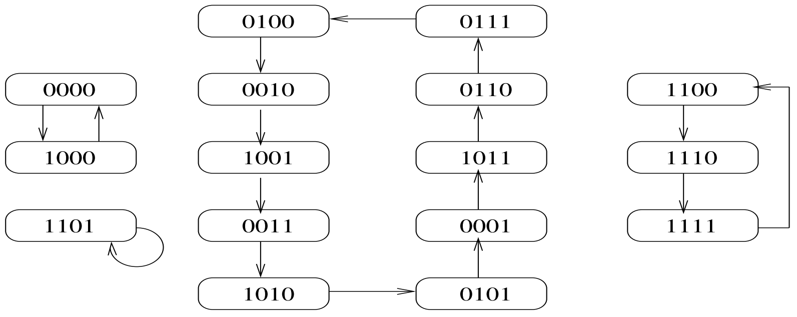

Figure 6.1: Transition diagram of the 4-cell reversible CA $\langle 9, 232, 90, 20\rangle$.

reversible CA.

Definition 6.1 *Two configurations* $\mathtt{x}$ *and* $\mathtt{y}$ *are said to be* close *if* $\mathtt{x}, \mathtt{y} \in C_i \subset \mathbb{X}$ *such that* $G_n^{l_1}(\mathtt{x}) = \mathtt{y}$ *and* $G_n^{l_2}(\mathtt{y}) = \mathtt{x}$ *for some* l_1, l_2. *Similarly, if* $\mathtt{z}$ *is* not reachable *from* $\mathtt{x}$ *then obviously,* $\mathtt{x}$ *is also* not reachable *from* $\mathtt{z}$ *and they are also* not close *to each other as* $\mathtt{x} \in C_i$ *and* $\mathtt{z} \in C_j$ *where* $i \neq j$.

In this way, a CA can place a set of close objects (configurations) in one cluster (cycle) which are not reachable from the remaining objects (configurations). Thus the cyclic spaces of CA can be used as natural clusters.

Example 6.2 Let us consider that the following data objects (configurations) are to be clustered: 1100, 1110, 0100, 0110 and 0101. As each object is represented by a 4-bit binary string, so a binary CA of size 4 can act as a clustering tool. Let a 4-bit CA $\langle 9, 232, 90, 20\rangle$ (see Figure 6.1) is taken for this purpose. This CA distributes the objects among *two* clusters where the first one represents $\{1100, 1110\}$ as 1100 is reachable from 1110 and 1110 is also reachable from 1100. Similarly, the second cluster is formed as $\{0100, 0110, 0101\}$.

However, by changing the CA, it may be found that the same data objects can be clustered in different arrangement – the distribution of the target objects as well as the number of clusters can be changed. For instance, let us take the CA $\langle 9, 23, 90, 20\rangle$ for clustering the same set of objects $\{1100, 1110, 0100, 0110, 0101\}$. The only change in the later CA with the former one is that $\mathcal{R}_1$ is changed to 23 from 232. Figure 6.2 explains how clustering is to be done using CA $\langle 9, 23, 90, 20\rangle$. In this CA, 0101 is separated from the remaining $2^n - 1$ configurations which form another cycle. Therefore, the target objects are distributed in *two* clusters in the following way: the first cluster connects 1100, 1110, 0100 and 0110 whereas the second one is formed of only 0101. This shows that, even though both CAs have minimum change in the rule vector form two clusters for

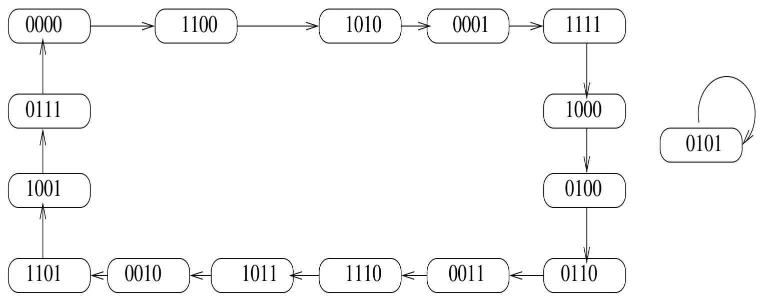

Figure 6.2: Transition diagram of the 4-cell reversible CA $\langle 9, 23, 90, 20\rangle$.

the given objects, but the arrangement is different. For the CA $\langle 9, 232, 90, 20\rangle$, 0100 and 0110 were in second cluster and it was separated from the other cluster consisting of $\{1100, 1110\}$; but now all these objects - 1100, 1110, 0100 and 0110 are in the same cluster using the CA $\langle 9, 23, 90, 20\rangle$. As a CA can be viewed as an unique bijective function, therefore, the connectivity among the configurations can be changed even if a small change is occurred (at least one participating rule is changed) in the given rule sequence of the CA. For example, let $\mathtt{x} = x_0x_1 \cdots x_{i-1}x_ix_{i+1} \cdots x_{n-1}$ be a configuration of an n-cell CA G_n such that $G_n(\mathtt{x}) = \mathtt{y_1}$. Let $\mathcal{R}_i$ be the rule at cell i of the given CA such that $\mathcal{R}_i(x_{i-1}x_ix_{i+1}) = x$. Let consider another CA of same size (n) G'_n and let $\mathcal{R}'_i$ be the i^{th} rule of G'_n. If $\mathcal{R}'_i(x_{i-1}x_ix_{i+1}) = x' \neq x$, then $\mathtt{y_1} \neq \mathtt{y_2}$ where $G'_n(\mathtt{x}) = \mathtt{y_2}$. Definitely, $\mathtt{y_1}$ is *reachable* from $\mathtt{x}$ in one step using G_n whereas $\mathtt{y_1}$ is not reachable from $\mathtt{x}$ in one step if we use G'_n. However, it may be reached from $\mathtt{x}$, if ${G_2}^k(\mathtt{x}) = \mathtt{y_1}$, for some $k \in \mathbb{N}$. That is, $\mathtt{x}$, $\mathtt{y_1}$ and $\mathtt{y_2}$ can be part of the same cyclic space. As all configurations inside the same cyclic space form the same cluster, similar clustering may be formed even if different CAs are used.

Let G_n and $G_{n'}$ are two CAs and $\mathbb{X}$ be the set of target objects such that $\mathbb{X} \subset C$. Let the clustering done by G_n and $G_{n'}$ are $\mathbb{X} = C_1^1 \cup C_2^1 \cup \cdots \cup C_m^1$ and $\mathbb{X} = C_1^2 \cup C_2^2 \cup \cdots \cup C_m^2$ respectively, where C_i^j and $C_{i'}^j$ are *disjoint* sets for any $i \neq i'$. Now, two clusters C_i^1 and C_j^2 are named as *identical clusters* if $C_i^1 = C_j^2$. If for each C_i^1 $(1 \leq i \leq m)$, there is an identical cluster C_j^2, then G_n and $G_{n'}$ are named *Equivalent Clustering CAs* with respect to the set of target objects $\mathbb{X}$.

Example 6.3 Let us consider that the following data objects (configurations) are to be clustered: 0100, 1001, 0001, 1011 and 0101. Let us take the 4-bit CA $\langle 9, 232, 90, 20\rangle$ as G_1 where $C_1^1 = \{0100, 1001, 0001, 1011\}$ and $C_2^1 = \{0101\}$. Using another 4-bit CA $\langle 9, 23, 90, 20\rangle$ as G_2, we can get the following clusters - $C_1^2 = \{0100, 1001, 0001, 1011\}$ and $C_2^2 = \{0101\}$. As, $C_1^1 = C_1^2$ and $C_2^1 = C_2^2$ therefore, each pair represents *identical* clusters and two CAs $\langle 9, 232, 90, 20\rangle$ and $\langle 9, 23, 90, 20\rangle$ are said to be *Equivalent Clustering CAs* with respect to the given set of data objects {0100, 1001, 0001, 1011, 0101}.

From the above discussion, it is clear that, same set of objects can be distributed among clusters in multiple ways – sometimes, in the similar arrangement among the clusters even if the different CAs are used; sometimes, in same number of clusters but different arrangements. It can also be noted that the objects can be distributed among different number of clusters using dissimilar arrangement. So, out of all those possibilities of arrangements, we need to figure out which is the most appropriate reversible CA for a given dataset. Such CA contributes to *effective* clusters. In the next section, the target is to figure out the *significant* CA rules for designing this *effective* clustering.

6.4 Clustering using Reversible CA with $r = 1$

This section considers cellular automata with radius *one*. For clustering, as reversible CA is required, therefore, only a few reversible uniform CAs are available which limits the search space. So, we select non-uniform reversible cellular automata for clustering problem. Obviously, all CAs may not produce good quality clusters for a particular dataset. A CA is *significant* for *effective* clustering if it maintains the following necessary conditions - (i) less intra-cluster distance in feature space and (ii) optimal number of clusters. So, our next challenge is to figure the arrangement of CA rules for designing the rule vectors which are appropriate for the clustering of a given dataset.

6.4.1 CA Rules that Maintain Minimum Intra-cluster Distance

For our present objective, we need to select such CA rules where the minimum changes occur during state transition. Ideally, it is desired that $f_i(x_{i-1}, x_i, x_{i+1}) = x_i$ for any value of i and any combination of $x_{i-1}x_ix_{i+1}$. The property of such n-cell CA is that each cell follows a special rule where *all* RMTs are *self replicating* - this is rule 204. However, this n-cell CA with only rule 204 is not *effective* as number of clusters is 2^n where each object forms an unique cluster. Therefore, it is desired to rank the rules based on the number of self replicating RMTs it possesses when used in designing the rule vector of a reversible CA. Our objective is to determine *significant* rules and the proportion of high ranked rules for designing clusters of objects that maintain less intra-cluster distances. To detect such rules, we rank each rule of Table 2.7a to 2.7c. This *rank* determines how a rule can act as an influencing factor for designing a cluster with more similar (less hamming distance) data.

Table 6.3: Ranking of CA rules based on number of self replicating RMTs

Rank	$\mathcal{R}_i$
1	12, 204, 68
2	92, 172, 197, 202, 108, 156, 198, 201, 77, 142, 212, 232, 78, 141, 216, 228
3	6, 9, 5, 10, 86, 89, 101, 106, 149, 154, 166, 169, 30, 45, 75, 120, 135, 180, 210, 225, 90, 105, 150, 165, 85, 170, 102, 153, 60, 195, 15, 240, 5, 20, 65, 80
4	53, 58, 83, 163, 54, 57, 99, 147, 23, 43, 113, 178, 27, 39, 114, 177
5	3, 51, 17

As balanced rules are used for reversible CAs, each rule follows even number of self replicating RMTs (0, 2, 4, 6 and 8). We rank these CA rules into *five* categories based on the contribution of self replicating RMTs (the contribution is measured by the ratio of the number of self replicating RMTs with total number of RMTs). Table 6.3 presents the rank of rules depending on the number of self replicating RMTs. Column I of Table 6.3 refers the rank of the corresponding rule. We can see that, rule 204 is ranked *first* but it is already mentioned that for an n-cell CA, if rule 204 is applied to every cell, then each target object belongs to an unique cycle. Also, that means, it distributes even similar objects into different clusters, none of which is not desirable for clustering. Moreover, rule 51 is the *least significant* rule for clustering as its rank is 5 – if rule 51 is applied at every cell of a CA of size n, every time cell changes its state which results that each cycle is formed of two configurations and they have no similarity in feature space. Therefore, we need to select rule 204 for as many cells as possible and just opposite strategy should be used for rule 51. However, CAs with only rules 204 and 51 are not *effective* for clustering. Hence, to design clusters of objects with less intra-cluster distance, we take the following strategy of choosing rules for synthesizing a reversible CA:

1. *Discard all rules with Rank* 4 *and* 5 *(that is, less than* 4 *self-replicating RMTs) from Table 2.7a.* `Seventeen` *rules (51, 53, 58, 83, 163, 54, 57, 99, 147, 23, 43, 113, 178, 27, 39, 114, 177) are discarded by this condition. Therefore, currently, the rule space is reduced to* 45.

2. *For the $n-2$ non-terminal cell positions (cell* 1 *to cell $n-2$), at most 50% rules with* rank 2 *(*6 *self-replicating RMTs) are to be selected.*

The following example explains how this can be effective for maintaining less intra-cluster distance.

Example 6.4 Let us consider a hypothetical dataset where each object is identified by three *quantitative* features A_1, A_2 and A_3. As $v = 3$, so n can be computed as 6. Let the total number of target objects counted as 50. After encoding, let those 50 objects are mapped to 10 useful configurations and let the configurations are: 111100, 111111, 111101, 011011, 011000, 011001, 001100, 001101, 001011 and 001001. Let a 6-bit CA $\langle 6, 232, 60, 197, 105, 17 \rangle$ is designed using the above strategy to perform clustering on those 10 configurations. Based on this CA, these ten configurations belong to *six* different cycles which results that the target objects are actually distributed over 6 clusters. The clustering is done in the given manner: cluster 1 contains 111100 and 111111 whereas cluster 2 keeps the configuration 111101; similarly, cluster 3 and cluster 4 present the configurations $\{011011, 011000\}$ and $\{011001\}$ respectively. Cluster 5 refers to $\{001100, 001101\}$ and configurations 001011 and 001001 are represented as cluster 6. Out of these six clusters, *four* of them stores two configurations in each cluster and each of the remaining *two* clusters considers only one configuration. We can observe that, each cluster includes configurations with where state values differ only at two positions - cell 5 and 6, that means, target objects have close values with respect to *feature 1* and *feature 2*. In this way, the less intra-cluster distance is maintained in the resultant clustering.

Using example 6.4, it is described that the above mentioned scheme can guarantee of distributing similar objects in the same cluster but it can not restrict the number of clusters (cycles). Next, we focus on the design of CA with limited number of cycles.

6.4.2 Designing CA with Optimal Number of Clusters

From the above discussion, it is obvious that, the consecutive configurations of a cycle maintain minimum distance in feature space if *more* significant rules are used in an n-cell CA. That is, the same cycle connects alike objects. However, it may increase the number of cycles. So, there is a trade-off between these two aspects of clustering technique – maintaining less number of cycles (clusters) and less intra-cluster distance among the objects, that is, configurations with less hamming distances are in the same cycle. In this section, we discuss a technique to generate CAs with limited number of cycles, that is, more configurations are to be placed on the same cycle. This requirement matches with an existing problem statement, *generation of large length cycle CA*, already studied in Section 5.2.2. For ease of understanding, we briefly recall the idea.

A CA is expected to have a cycle of large length, if its rules are dependent on both of the left and right neighbors. To measure this dependence, a *parameter* ($\mathtt{P}$), called *degree of dependence on both the neighbors*, is defined which determines how much a cell is dependent on its neighbors for updating its state. For a rule $\mathcal{R}_i$, $\mathtt{P}(\mathcal{R}_i) = \mathtt{P_r}(\mathcal{R}_i) * \mathtt{P_l}(\mathcal{R}_i)$. Here, $\mathtt{P_r}(\mathcal{R}_i)$ (resp. $\mathtt{P_l}(\mathcal{R}_i)$) is the *degree*

Table 6.4: Categories of reversible CA rules on the parameter P

(a) Categories of reversible CA rules

Category	$\mathcal{R}_i$
Completely Dependent	90, 165, 150, 105
Partially Dependent	30, 45, 75, 120, 135, 180, 210, 225, 86, 89, 101, 106, 149, 154, 166, 169
Weakly Dependent	92,172, 197, 202, 108, 156, 198, 201, 77, 142, 212, 232, 78, 141, 216, 228,
	53, 58, 83, 163, 54, 57, 99, 147, 23, 43, 113, 178, 27, 39, 114, 177
Independent	51, 85, 170, 102, 153, 60, 195, 15, 240 204

(b) Categories of $\mathcal{R}_0$ and $\mathcal{R}_{n-1}$

Category	$\mathcal{R}_0$	$\mathcal{R}_{n-1}$
Completely Dependent	5, 6, 9, 10	5, 20, 65, 80
Independent	5 12	17 68

of right (resp. left) dependence, defined as the ratio of the number of combinations of values of x_i and x_{i-1} (resp. x_{i+1}) for which the next state function on x_i depends on x_{i-1} (resp. x_{i+1}). Evidently, $\mathtt{P}(\mathcal{R}_i)$ can take values 0, 0.25, 0.5 or 1. Based on these values, the rules of reversible CAs are classified into *four* categories – *completely dependent* ($\mathtt{P} = 1$), *partially dependent* ($\mathtt{P} = 0.5$), *weakly dependent* ($\mathtt{P} = 0.25$) and *independent* ($\mathtt{P} = 0$) (see Table 6.4). It is observed that in a CA with large length cycle(s), majority of the participating rules are from the *completely dependent* category, some are from the *partially dependent* category and a few are from the category of *weakly dependent*, whereas, none are from the *Independent* category. For more detailed discussion, please see Section 5.2.2.

Obviously, following the strategy mentioned in Section 6.4.1, sixteen rules from *weakly dependent* category and all rules (*ten* rules) from *independent* categories are rejected for clustering purpose as they produce more pair of configurations with high hamming distances. So, remaining 36 (for cell 1 to $n-2$) rules are *significant* for designing CA which results in *effective* clustering. These rules are: the sixteen rules of *weakly dependent* category with *rank* 2, whereas, the rules of *completely dependent* and *partially dependent* categories with *rank* 3. We can now proceed to our clustering technique in the next section.

6.4.3 Cycle based Clustering

As already discussed, the clusters are nothing but the cycles of CA – so, we call our CA based clustering technique as *cycle based clustering*. Such clustering does not only maintain the cycles of the configurations with lesser hamming distances, but also ensure that the number of cycles in the CA does not grow exponentially with CA size. Moreover, the scheme of generating significant CA also works with reduced rule set. Next, the steps of our proposed clustering technique using such a CA is stated as following:

- *Step 1:* Based on the types of attributes (equation 6.1 for quantitative and equation 6.2 for qualitative), the target objects of $\mathbb{X}$ are encoded into the set of n-bit useful configurations which is denoted as $\mathscr{C}$.
- *Step 2:* Randomly choose an arbitrary CA $\mathcal{R}$ of size n that maintains rules at all cells from a set of *rank 3* which are either *completely dependent* or *partially dependent* and at most *one* rule from *weakly dependent* category with *rank 2*.
- *Step 3:* Let the set of remaining objects (configurations) to be clustered is $\mathscr{C}'$. Initially, $\mathscr{C}' = \mathscr{C}$. Set $k = 1$.
- *Step 4:* Let C_k represent those configurations which are *close* to $\mathtt{x_i}$ such that $C_k \subset \mathscr{C}'$ and $\mathtt{x_i} \in \mathscr{C}'$. Set $C_k = C_k \cup \mathtt{x_i}$ and $\mathscr{C}' = \mathscr{C}' \setminus C_k$. Increment k by 1.
- *Step 5:* Continue the above step until all the configurations are clustered such that $\mathscr{C}' = \varnothing$ and $C_1 \cap C_2 \cap \cdots \cap C_m = \varnothing$, where m is the number of clusters.

Let us illustrate the process using an example.

Example 6.5 Let us consider `Iris` dataset from UCI Machine Learning repository (`http://archive.ics.uci.edu/ml/index.php`, see Table 6.5) where $\mathbb{X} = \{\mathtt{X_1}, \mathtt{X_2}, \cdots, \mathtt{X_{150}}\}$, that is $|\mathbb{X}| = 150$. There are `four` quantitative attributes (p_{q_n}); therefore, n can be computed as $n = (2 * p_{q_n}) = 2 * 4 = 8$ and using the encoding function (mentioned in equation 6.1), each target object is mapped to an useful configuration and ultimately, these 150 are represented by only 24 configurations. Here, $\mathscr{C}$ can be represented as $= \{\mathtt{x_1}, \mathtt{x_2}, \cdots, \mathtt{x_{24}}\}$, where, $\mathtt{x_1} = 00110000$, $\mathtt{x_2} = 10111011$, $\mathtt{x_3} = 00000000$, $\mathtt{x_4} = 11001110$, $\mathtt{x_5} = 10110000$, $\mathtt{x_6} = 10101111$, $\mathtt{x_7} = 00100000$, $\mathtt{x_8} = 11111011$, $\mathtt{x_9} = 11001010$, $\mathtt{x_{10}} = 10111111$, $\mathtt{x_{11}} = 10001011$, $\mathtt{x_{12}} = 10001110$, $\mathtt{x_{13}} = 00001010$, $\mathtt{x_{14}} = 10001010$, $\mathtt{x_{15}} = 11101010$, $\mathtt{x_{16}} = 11001111$, $\mathtt{x_{17}} = 10101010$, $\mathtt{x_{18}} = 11111111$, $\mathtt{x_{19}} = 11111010$,

$\mathtt{x_{20}} = 11101110$, $\mathtt{x_{21}} = 11101111$, $\mathtt{x_{22}} = 10101011$, $\mathtt{x_{23}} = 10001111$ and $\mathtt{x_{24}} = 00001011$.

Let us consider $\mathcal{R}$ as $\langle 10, 45, 156, 86, 90, 165, 150, 65\rangle$ by following the strategy mentioned in *Step 2*. Initially, $\mathscr{C}' = \mathscr{C}$ and let $m = 1$. Let $\mathtt{x_1} = 00110000 \in \mathscr{C}'$ be taken to figure out which remaining configurations from $\mathscr{C}'$ are *close* to $\mathtt{x_1}$. Out of 23 configurations, only configuration 00100000 is close to $\mathtt{x_1}$. Therefore, $C_1 = \{\mathtt{x_1}, \mathtt{x_7}\}$ and $\mathscr{C}'$ is updated to $\mathscr{C}' \setminus C_1$ such that $|\mathscr{C}'| = 22$. Next, $\mathtt{x_2} = 10111011$ is chosen from $\mathscr{C}'$; we can see that the configurations 00000000, 00001010, 10101111 and 10001011 are close to $\mathtt{x_2}$. Therefore, cluster C_2 consists of five configurations $\{\mathtt{x_2}, \mathtt{x_3}, \mathtt{x_{13}}, \mathtt{x_6}, \mathtt{x_{11}}\}$ and $\mathscr{C}'$ is updated to $\mathscr{C}' \setminus C_2$ such that $|\mathscr{C}'| = 17$. In the same way, the remaining 17 configurations of $\mathscr{C}'$ are distributed among two clusters such that one cluster is $C_3 = \{\mathtt{x_{19}}, \mathtt{x_9}, \mathtt{x_{15}}, \mathtt{x_{17}}, \mathtt{x_4}, \mathtt{x_{21}}, \mathtt{x_{23}}, \mathtt{x_{18}}, \mathtt{x_{24}}, \mathtt{x_{16}}, \mathtt{x_{12}}, \mathtt{x_{22}}, \mathtt{x_{10}}\}$ and another is $C_4 = \{\mathtt{x_5}, \mathtt{x_8}, \mathtt{x_{14}}, \mathtt{x_{20}}\}$. Therefore, these 150 target objects which are represented by 24 configurations are distributed among *four* clusters by our algorithm.

The above example confirms that our clustering technique is not only maintaining less intra-cluster distance, but is also capable of distributing the target objects among a limited number of clusters. Therefore, the cycle based clustering is an effective technique for clustering. However, sometimes, there is a requirement from the users to distribute the target objects among a *desired* number of clusters. To deal the issue, we re-sketch our proposed technique by introducing the use of multiple CAs; the revised technique is an *iterative* level-wise approach which uses multiple levels and uses *cycle based clustering* at every level.

6.5 Iterative Cycle based Clustering for Desired Number of Clusters

It is already established that CA can perform *effective* clustering using its cyclic space, but sometimes to find a CA that can distribute the objects into desired number of clusters is difficult. To solve this problem, we opt for level-wise iterative clustering technique where at every level the number of clusters are *reduced* to reach either the desired number of clusters or the *optimal* number of clusters for the given dataset. So, we re-formulate our technique for clustering using the *candidate* CAs. It is already reported that any arbitrary CA is not *acceptable* as candidate; the participating CAs must follow the conditions discussed in Section 6.4.1 and Section 6.4.2.

- `Property 1:` Participating CA of size n maintains rules at all cells from a subset of *rank 3* and at most *one* rule from *rank 2*.

Algorithm 9 Iterative Level-wise Cycle based Clustering.

```
Input: Target object set $\mathbb{X} = \{\mathtt{X}_1, \mathtt{X}_2, \cdots, \mathtt{X}_k\}$, quantitative and qualitative attributes, optimal number of clusters ($m$), auxiliary CA space
Output: The clusters $\{\mathtt{c_1}^v, \mathtt{c_2}^v, \cdots, \mathtt{c_m}^v\}$
1: Set $p_{q_n}$ ($p_{q_l}$ resp.) $\leftarrow$ number of quantitative (qualitative resp.) attributes; $u_i \leftarrow$ number of distinct elements in $i^{th}$ qualitative attribute; $n \leftarrow (2*p_{q_n}) + (\sum_{i=1}^{p_{q_l}} u_i)$.
2: for each $j = 1$ to $k$ do Encode $\mathtt{X}_j$ into an $n$-bit binary string. end for
3: Let $\mathtt{M}(\mathbb{X})$ be the set of encoded target objects $\{\mathtt{x}_1, \mathtt{x}_2, \cdots, \mathtt{x}_{k'}\}$.
4: Construct a set of $n$-cell CAs $\mathtt{R}$ from the given auxiliary CA space.
5: Set $m_0 \leftarrow k'$, $i \leftarrow 1$ and $\mathtt{z} \leftarrow 1$.
6: for $j = 1$ to $m_0$ do Set $\mathtt{c_j}^0 \leftarrow \{\mathtt{x_j}\}$ end for   # level 0 primary clusters.
7: while $(m_i \neq m_{i-1}) || (m_i \neq m)$ do
8:    Select $\mathcal{R} \in \mathtt{R}$ and Set $\mathtt{R} \leftarrow \mathtt{R} \setminus \{\mathcal{R}\}$   # auxiliary CA selected randomly at uniform without replacement
9:    Generate auxiliary clusters $C_1^i, C_2^i, \cdots, C_{m'}^i$ for the CA $\mathcal{R}$
10:   Initialize a matrix $A[a_{tj}]_{m' \times m_{i-1}}$ to 0.
11:   for $t = 1$ to $m'$ do
12:      for $j = 1$ to $m_{i-1}$ do Set $a_{tj} \leftarrow \mu(C_t^i, \mathtt{c_j}^{i-1})$. end for
13:   end for
14:   for each $j = 1$ to $m_{i-1}$ do   # For every primary cluster of previous level
15:      Let $a_{t'j}$ = maximum of $a_{tj}$ where $1 \leq t \leq m'$.
16:      Find the auxiliary cluster with maximum participation of $\mathtt{c_j}^{i-1}$.
17:      for $(j_1 = 1$ to $m_{i-1})\&\&(j_1 \neq j)$ do
18:         Find $a_{t'j'}$ = maximum of $a_{tj_1}$ such that $a_{t'j'} \neq 0$.
19:      end for
20:      if there exists some $a_{t'j'}$ then
21:         Set $\mathtt{c_z}^i \leftarrow \mathtt{c_j}^{i-1} \cup \mathtt{c_{j'}}^{i-1}$ and $\mathtt{z} \leftarrow \mathtt{z} + 1$.
22:         Mark $\mathtt{c_j}^{i-1}$ and $\mathtt{c_{j'}}^{i-1}$ as modified. Remove row $t'$ from $A$.
23:      end if
24:   end for
25:   for every unmodified clusters $\mathtt{c_y}^{i-1}$ do
26:      Set $\mathtt{c_z}^i \leftarrow \mathtt{c_y}^{i-1}$ and $\mathtt{z} \leftarrow \mathtt{z} + 1$.   # Move unmodified primary cluster of level $i-1$ to new primary cluster of level $i$ and update cluster number.
27:   end for
28:   Set $m_i \leftarrow z$ and $\mathtt{i} \leftarrow \mathtt{i} + 1$.
29: end while
30: Report $\mathtt{c_1}^i$, $\mathtt{c_2}^i$, $\cdots$, $\mathtt{c_{m_i}}^i$ as the final clusters at level $i$ and Exit
```

- `Property 2:` Our technique *converges* by *merging* the clusters. To do that, a hierarchy of levels has to be maintained.

- `Property 3`: Only *closely reachable* clusters of level $i-1$ are to be merged to generate the updated clusters of level i.

Let $\mathtt{M}(\mathbb{X})$ be the set of *encoded* target objects where $\mathtt{M}(\mathbb{X}) \subset C$ and $\mathbb{X} = \{\mathtt{X}_1, \mathtt{X}_2, \cdots, \mathtt{X}_\mathtt{k}\}$ is the set of target objects. For any particular ordering of the features, these encoded target objects are a set of *useful configurations* $\mathscr{C}$. Let $|\mathtt{M}(\mathbb{X})| = |\mathscr{C}| = k'$ where $k' \leq k$. At any level, a useful configuration $\mathtt{x} \in \mathscr{C}$ is member of a distinct cluster $\mathtt{c}$ (a set of encoded target objects) such that $|\bigcup c| = |\mathscr{C}|$.

Let m_i be the number of *primary* clusters at level i. For level 0, the primary clusters are ${\mathtt{c}_1}^0, {\mathtt{c}_2}^0, \cdots, {\mathtt{c}_{\mathtt{m}_0}}^0$, where each cluster is a singleton set. Therefore, $k' = m_0$. In general, for any level i, the primary clusters are ${\mathtt{c}_1}^i, {\mathtt{c}_2}^i, \cdots, {\mathtt{c}_{\mathtt{m}_\mathtt{i}}}^i$. To form these primary clusters of level i from level $i-1$, a CA of size n is selected *uniformly random without replacement* from a pool of candidate CAs maintaining `property 1`. This process is maintained at every level i. Such a CA is named as an *auxiliary* CA. This CA plays a major role in clustering. Firstly, it is needed to compute the number of *auxiliary clusters* of such a CA, in which the *target* configurations (k') *strictly belong to*.

Definition 6.2 *Let x be a useful configuration and $G : C \mapsto C$ be an auxiliary CA. If $x \in C_j$ where $C_j \subset C$ is a cyclic space of G, then x strictly belongs to the auxiliary cluster C_j.*

Let the useful configurations *strictly belong* to m' number of *auxiliary* clusters $C_1^i, C_2^i, \cdots, C_{m'}^i$ of level i. Our *second* step is to follow `property 2`, that is, *merging* the primary clusters ${\mathtt{c}_1}^{i-1}, {\mathtt{c}_2}^{i-1}, \cdots, {\mathtt{c}_{\mathtt{m}_0}}^{i-1}$ of level $i-1$ using these auxiliary clusters to get the resultant primary clusters of level i where $m_i \leq m_{i-1}$. However, these clusters can not be merged arbitrarily; a pair of primary clusters can be merged depending on their *degree of membership of participation.*

Definition 6.3 *Let ${\mathtt{c}_\mathtt{j}}^{i-1}$ be a primary cluster of level $i-1$ where $|{\mathtt{c}_\mathtt{j}}^{i-1}| = v_j$. Let C_t^i be an auxiliary cluster of level i. The degree of membership of participation of ${\mathtt{c}_\mathtt{j}}^{i-1}$ in C_t^i, denoted by $\mu(C_t^i, {\mathtt{c}_\mathtt{j}}^{i-1})$, is defined as the availability of configurations of ${\mathtt{c}_\mathtt{j}}^{i-1}$ in C_t^i. It is computed as v_j'/v_j where v_j' refers to the count of useful configurations from primary cluster ${\mathtt{c}_\mathtt{j}}^{i-1}$ in auxiliary cluster C_t^i.*

The configurations of ${\mathtt{c}_\mathtt{j}}^{i-1}$ can *strictly belong to* more than one auxiliary cluster. Similarly, C_t^i can possess useful configurations from different clusters of level $i-1$. Let ${\mathtt{c}_\mathtt{l}}^{i-1}$ and ${\mathtt{c}_\mathtt{j}}^{i-1}$ be two primary clusters of level $i-1$. These two clusters may be merged if they are necessarily *closely reachable* (`property 3`).

Definition 6.4 *Let ${\mathtt{c}_\mathtt{j}}^{i-1}$, ${\mathtt{c}_\mathtt{l}}^{i-1}$ and ${\mathtt{c}_\mathtt{s}}^{i-1}$ be the clusters whose members strictly belong to C_t^i. Now, clusters ${\mathtt{c}_\mathtt{j}}^{i-1}$ and ${\mathtt{c}_\mathtt{l}}^{i-1}$ are said to be* closely reachable *in C_t^i if $|(\mu(C_t^i, {\mathtt{c}_\mathtt{j}}^{i-1}) - \mu(C_t^i, {\mathtt{c}_\mathtt{l}}^{i-1}))| < |(\mu(C_t^i, {\mathtt{c}_\mathtt{j}}^{i-1}) - (\mu(C_t^i, {\mathtt{c}_\mathtt{s}}^{i-1}))|$.*

Therefore, for every C_t^i, we can get pairs of closely reachable clusters. The *degree of participation* plays a vital role for selecting the closely reachable clusters which are then merged. The pseudocode of our methodology is presented in Algorithm 9. Next, we discuss the algorithm in detail.

1. Let $\mathtt{c_1}^0, \mathtt{c_2}^0, \cdots, \mathtt{c_{m_0}}^0$ (resp. $\mathtt{c_1}^{i-1}, \mathtt{c_2}^{i-1}, \cdots, \mathtt{c_{m_{i-1}}}^{i-1}$) be the primary clusters of level 0 (resp. $i-1$) where the count of clusters is m_0 (resp. m_{i-1}). Also let $C_1^1, C_2^1, \cdots, C_{m'}^1$ (resp. $C_1^i, C_2^i, \cdots, C_{m'}^i$) be the auxiliary clusters of level 1 (resp. i) where the count of auxiliary clusters is m'. For all t, $1 \le t \le m'$, compute $\mu(C_t^1, \mathtt{c_j}^0)$ (resp. $\mu(C_t^i, \mathtt{c_j}^{i-1})$). For any given $\mathtt{c_j}^0$ (resp. $\mathtt{c_j}^{i-1}$), find the auxiliary cluster of level 1 (resp. i) in which it has *maximum* participation, that is, its *degree of participation* is maximum. Obviously, for some value of t, maximum participation of $\mathtt{c_j}^0$ (resp. $\mathtt{c_j}^{i-1}$) is in C_t^1 (resp. C_t^i).
2. Let $C_{t_1}^1$ (resp. $C_{t_1}^i$) be the auxiliary cluster having maximum configurations belonging from $\mathtt{c_j}^0$ (resp. $\mathtt{c_j}^{i-1}$). Therefore, $\mathtt{c_j}^0$ (resp. $\mathtt{c_j}^{i-1}$) can merge with some of the clusters which have also participated in $C_{t_1}^1$ (resp. $C_{t_1}^i$). However, only those clusters are to be merged with $\mathtt{c_j}^0$ (resp. $\mathtt{c_j}^{i-1}$), which are closely reachable to $\mathtt{c_j}^0$ (resp. $\mathtt{c_j}^{i-1}$). Hence, a new primary cluster $\mathtt{c_j}^1$ (resp. $\mathtt{c_j}^i$) is formed as $\mathtt{c_j}^i = \mathtt{c_j}^{i-1} \cup \mathtt{c_l}^{i-1}$ if and only if $|(\mu(C_{t_1}^i, \mathtt{c_j}^{i-1}) - \mu(C_{t_1}^i, \mathtt{c_l}^{i-1}))| < |(\mu(C_{t_1}^i, \mathtt{c_j}^{i-1}) - \mu(C_{t_1}^i, \mathtt{c_s}^{i-1}))|$, for any $s \ne l$ where $\mathtt{c_s}^{i-1}$ is another participating cluster in $C_{t_1}^i$ and $max\{\mu(C_t^i, \mathtt{c_j}^{i-1})_{(\forall t)}\} = \mu(C_{t_1}^i, \mathtt{c_j}^{i-1})$. Therefore, the newly generated cluster $\mathtt{c_j}^i$ constitutes of a set of useful configurations, out of which some strictly belongs to a cluster (cycle) of the auxiliary CA of level i.
3. If for any primary cluster $\mathtt{c_j}^0$ (resp. $\mathtt{c_j}^{i-1}$), there is no closely reachable primary cluster in all auxiliary clusters, then the new primary cluster of level i is $\mathtt{c_j}^i = \mathtt{c_j}^{i-1}$. Therefore, $m_i \le m_{i-1}$.
4. The algorithm stops when we reach the *optimal* number of clusters (m). The test of optimality is determined either by arriving at the user specified number of clusters or if $m_i = m_{i-1}$ after a fixed number of attempts.

Example 6.6 Let us consider the `Iris` dataset (`http://archive.ics.uci.edu/ml/index.php`, see Table 6.5) where $\mathbb{X} = \{\mathtt{X_1}, \mathtt{X_2}, \cdots, \mathtt{X_{150}}\}$ and each object has four quantitative (p_{q_n}) and no qualitative attributes (p_{q_l}). Hence, size of the CA for this data set is $n = 2 * 4 = 8$. Now, let the desired number of clusters (m) is `two`. Using the encoding technique, we get $\mathtt{M}(\mathbb{X}) = \{\mathtt{x_1}, \mathtt{x_2}, \cdots, \mathtt{x_{24}}\}$.

Initially, at level 0, there exist `twenty four` primary clusters such that $\mathtt{c_1}^0 = \{\mathtt{x_1}\}, \mathtt{c_2}^0 = \{\mathtt{x_2}\}, \cdots, \mathtt{c_{24}}^0 = \{\mathtt{x_{24}}\}$, where $m_0 = 24$. As $m_0 \ne m$, we select an auxiliary CA $\mathcal{R}$ from the set of candidate CAs (satisfying `Property 1`) uniformly random without replacement. Let $\mathcal{R} = \langle 9, 169, 150, 150, 165, 105, 165, 20 \rangle$. This CA generates `four` auxiliary clusters - $C_1^0, C_2^0, C_3^0, C_4^0$ (see Example 6.5).

Table 6.5: Description of real datasets used for the proposed CA based clustering technique.

Name	Number of p	Number of p_{q_n}	Number of p_{q_l}	Number of target objects	CA size n	Number of objects with missing terms
Iris	4	4	0	150	8	0
User Knowledge Modeling	5	5	0	403	10	0
BuddyMove	6	6	0	249	12	0
Seed	7	7	0	210	14	0
StoneFlakes	8	8	0	79	16	6
Wholesale Customers	8	6	2	440	16	0
Heart failure clinical records	12	12	0	299	19	0
Wholesale Customers	8	6	2	440	16	0

Next, we find the degree of participation of each $\mathtt{c_j}^0$ in these clusters. As we are at level 1, $\mu(C_1^1, \mathtt{c_{j_1}}^0) = 100\%$ ($\forall j_1 \in \{1, 3, 4, 22, 13, 14\}$), $\mu(C_2^1, \mathtt{c_{j_2}}^0) = 100\%$ ($\forall j_2 \in \{2, 5, 6, 7, 8, 9, 10, 11, 12, 15, 17, 19, 20, 21, 23, 24\}$), $\mu(C_3^1, \mathtt{c_{16}}^0) = 100\%$ and $\mu(C_4^1, \mathtt{c_{18}}^0) = 100\%$. So, we can merge the closely reachable primary clusters of level 0 to form the primary clusters of level 1. Here, auxiliary cluster C_1^1 has maximum (and equal) participation of primary clusters $\mathtt{c_1}^0$, $\mathtt{c_3}^0$, $\mathtt{c_4}^0$, $\mathtt{c_{22}}^0$, $\mathtt{c_{13}}^0$ and $\mathtt{c_{14}}^0$. Similarly, C_2^1 has maximum participation of $\mathtt{c_2}^0$, $\mathtt{c_5}^0$, $\mathtt{c_6}^0$, $\mathtt{c_7}^0$, $\mathtt{c_8}^0$, $\mathtt{c_9}^0$, $\mathtt{c_{10}}^0$, $\mathtt{c_{11}}^0$, $\mathtt{c_{12}}^0$, $\mathtt{c_{15}}^0$, $\mathtt{c_{17}}^0$, $\mathtt{c_{19}}^0$, $\mathtt{c_{20}}^0$, $\mathtt{c_{21}}^0$, $\mathtt{c_{23}}^0$ and $\mathtt{c_{24}}^0$. Therefore, the newly generated primary cluster of level 1 is $\mathtt{c_1}^1 = \mathtt{c_1}^0 \cup \mathtt{c_3}^0 \cup \mathtt{c_4}^0 \cup \mathtt{c_{22}}^0 \cup \mathtt{c_{13}}^0 \cup \mathtt{c_{14}}^0$. Similarly, $\mathtt{c_2}^1$ can be generated. For the remaining two auxiliary clusters, new primary clusters are formed as $\mathtt{c_3}^1 = \mathtt{c_{16}}^0$ and $\mathtt{c_4}^1 = \mathtt{c_{18}}^0$. As, the number of primary clusters at level 1 (m_1) is $4 \neq m$, we move from level 1 to level 2.

Let, at level 2, the selected auxiliary CA is $\langle 6, 232, 90, 90, 165, 90, 90, 20\rangle$. This CA generates `six` auxiliary clusters C_1^2, C_2^2, C_3^2, C_4^2, C_5^2 and C_6^2. Like level 1, here also, we compute the maximum participation of each primary cluster of level 1 in auxiliary cluster C_t^2, $(1 \leq t \leq 6)$. It is found that $\mu(C_1^2, \mathtt{c_1}^1) = 16\%$, $\mu(C_2^2, \mathtt{c_1}^1) = 33\%$, $\mu(C_2^2, \mathtt{c_2}^1) = 62\%$, $\mu(C_2^2, \mathtt{c_3}^1) = 100\%$, $\mu(C_3^2, \mathtt{c_1}^1) = 16\%$, $\mu(C_3^2, \mathtt{c_2}^1) = 12\%$, $\mu(C_3^2, \mathtt{c_4}^1) = 100\%$, $\mu(C_4^2, \mathtt{c_1}^1) = 33\%$, $\mu(C_5^2, \mathtt{c_2}^1) = 18\%$ and $\mu(C_6^2, \mathtt{c_2}^1) = 6\%$. Hence, we can merge $\mathtt{c_2}^1$ and $\mathtt{c_3}^1$ with respect to the closeness in the auxiliary cluster C_2^2. Similarly, $\mathtt{c_1}^1$ and $\mathtt{c_4}^1$ can be merged with respect C_3^2. Hence, the newly generated primary clusters of level 2 are $\mathtt{c_1}^2 = \mathtt{c_2}^1 \cup \mathtt{c_3}^1 = \{2, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 17, 19, 20, 21, 23, 24\}$ and $\mathtt{c_2}^2 = \mathtt{c_1}^1 \cup \mathtt{c_4}^1 = \{1, 3, 4, 22, 13, 14, 18\}$. Therefore, at this level, $m_2 = 2$. As desired number of clusters is already achieved, the algorithm exits. $\square$

Our algorithm generates the requirement based clusters, also it gives the direction of optimal count of the clusters. Our technique uses v auxiliary CAs if the optimal number of clusters is achieved at level v. It is also to be noted that optimal number of clusters can be achieved using a set of v_1 CAs, v_2 CAs and so on. Now, our interest to figure out that set of bijective functions (CAs) which gives the *best* quality clusters among those possible set of CAs. We next report the experimental results with some *real* datasets.

6.6 Experimental Results using Auxiliary CAs with $r = 1$

This section reports the performance of our proposed iterative cycle based clustering algorithm on some real datasets (`http://archive.ics.uci.edu/ml/index.php`). To check the quality of the generated clusters using our algorithm, we use some benchmarks validation indices like *silhouette score*, *dunn index* and *connectivity* [70,462] - the set of bijective functions (auxiliary CAs) with the *least* score in *connectivity* and *highest* score in *dunn index* and *silhouette* generates the *best* quality clusters. Next, we introduce the real datasets used in this research.

We use *seven* datasets *Iris*, *User Knowledge Modeling*, *BuddyMove*, *Seed*, *StoneFlakes*, *Heart failure clinical records* and *Wholesale Customers*. Table 6.5 reports the details about these datasets. Here column I depicts the names of the datasets, whereas columns II, III and IV denotes the total number of attributes (p), number of quantitative attributes (p_{q_n}) and the count of qualitative attributes (p_{q_l}) respectively. Column V shows the number of target objects to be distributed among m clusters using an n-cell CA. Here, n (see column VI for reference) is computed based on p_{q_n} and p_{q_l} ($n > p$) (see Section 6.7). Column VII refers to the count of objects with missing terms for each dataset.

For analyzing the performance of our iterative clustering technique, it is needed to check first whether the target objects are distributed among desired number clusters; thereafter, the quality of cluster is to be evaluated based on the validation indices scores which can be computed using the package *clValid* in *R* (detail description of the package is reported in [71]). In this research, the desired number of cluster is taken as 2. Next section reports the results of these experiments taking a fixed ordering of features for each of the datasets.

6.6.1 Experiment on Real Datasets

Table 6.6 presents the effectiveness of our technique. This effectiveness is ensured based on validation indices scores and if each of the datasets are distributed into same desired number of clusters, that is *two*.

In Table 6.6, columns III refers the optimal scores of the validation indices -

Table 6.6: Performance of our clustering algorithm on available datasets where desired number of clusters (m) is *two*.

Dataset	Optimal Score	# Levels	Auxiliary CA
Iris	Silhouette = 0.6867, Dunn= 0.3389, Connectivity= 0.0000	2	$CA_0 : \langle 10, 75, 166, 105, 105, 166, 150, 20\rangle$ $CA_1 : \langle 6, 166, 165, 154, 105, 165, 165, 65\rangle$
User Knowledge Modeling	Silhouette= 0.186714,	3	$CA_0 : \langle 6, 166, 105, 165, 165, 150, 154, 150, 165, 65\rangle$ $CA1 : \langle 10, 180, 169, 165, 90, 90, 165, 165, 149, 80\rangle$ $CA2 : \langle 5, 180, 165, 225, 86, 105, 90, 105, 154, 80\rangle$
	Dunn= 0.1508426, Connectivity= 3.667857	4	$CA_0 : \langle 6, 169, 90, 169, 90, 169, 150, 150, 90, 80\rangle$ $CA_1 : \langle 9, 228, 154, 105, 105, 89, 90, 106, 105, 65\rangle$ $CA_2 : \langle 9, 105, 135, 172, 90, 165, 90, 90, 90, 20\rangle$ $CA_3 : \langle 5, 45, 89, 90, 169, 165, 101, 150, 90, 20\rangle$
Buddymove	Silhouette= 0.4763, Dunn= 0.3146, Connectivity= 2.9289	4	$CA_0 : \langle 10, 135, 197, 150, 169, 105, 105, 101, 105, 105, 86, 80\rangle$ $CA_1 : \langle 9, 150, 30, 198, 154, 150, 165, 165, 90, 105, 105, 86, 80\rangle$ $CA_2 : \langle 9, 166, 105, 150, 101, 150, 90, 90, 105, 86, 105, 20\rangle$ $CA_3 : \langle 6, 105, 30, 202, 90, 150, 105, 105, 105, 166, 105, 65\rangle$
Seed	Silhouette= 0.528,	2	$CA_0 : \langle 9, 228, 169, 165, 101, 90, 90, 90, 154, 150, 150, 105, 105, 20\rangle$ $CA_1 : \langle 5, 105, 89, 105, 105, 105, 150, 150, 150, 150, 165, 165, 165, 20\rangle$
	Dunn= 0.09, Connectivity= 3.91	4	$CA_0 : \langle 5, 120, 197, 150, 86, 165, 106, 165, 166, 105, 90, 105, 166, 5\rangle$ $CA_1 : \langle 5, 210, 165, 180, 202, 165, 150, 101, 90, 90, 105, 165, 165, 65\rangle$ $CA_2 : \langle 10, 120, 90, 105, 172, 150, 86, 150, 105, 154, 150, 165, 165, 65\rangle$ $CA_3 : \langle 5, 120, 172, 165, 150, 149, 150, 150, 169, 150, 105, 166, 105, 20\rangle$
Stone-Flakes	Silhouette= 0.376,	3	$CA_0 : \langle 6, 86, 165, 149, 165, 89, 165, 90, 90, 101, 105, 90, 150, 169, 150, 65\rangle$ $CA_1 : \langle 10, 135, 154, 165, 165, 150, 165, 90, 105, 106, 165, 154, 150, 150, 154, 5\rangle$ $CA_2 : \langle 10, 105, 166, 150, 150, 149, 150, 150, 165, 165, 154, 150, 90, 105, 86, 80\rangle$
	Dunn= 0.1805, Connectivity= 3.5123	4	$CA_0 : \langle 6, 166, 105, 105, 89, 90, 165, 165, 166, 150, 150, 150, 90, 106, 150, 20\rangle$ $CA_1 : \langle 6, 166, 150, 165, 165, 105, 105, 90, 169, 105, 90, 150, 89, 105, 165, 65\rangle$ $CA_2 : \langle 5, 210, 92, 105, 149, 90, 150, 105, 90, 165, 105, 106, 105, 165, 165, 65\rangle$ $CA_3 : \langle 5, 165, 92, 150, 165, 90, 169, 105, 105, 89, 105, 90, 150, 89, 150, 20\rangle$
Wholesale Customer	Silhouette= 0.5257,	3	$CA_0 : \langle 10, 45, 149, 105, 165, 90, 150, 101, 90, 149, 165, 106, 90, 150, 150, 65\rangle$ $CA_1 : \langle 9, 78, 165, 105, 90, 165, 105, 90, 90, 86, 105, 105, 154, 150, 90, 65\rangle$ $CA_2 : \langle 9, 154, 165, 150, 90, 150, 150, 90, 150, 86, 105, 90, 150, 86, 150, 20\rangle$
	Connectivity= 3.7329,	2	$CA_0 : \langle 6, 149, 150, 90, 150, 105, 90, 90, 165, 150, 90, 169, 165, 90, 105, 20\rangle$ $CA_1 : \langle 5, 210, 150, 228, 86, 105, 90, 105, 89, 90, 105, 105, 165, 90, 90, 20\rangle$
	Dunn= 0.0508	3	$CA_0 : \langle 6, 216, 149, 150, 150, 169, 150, 150, 106, 105, 90, 90, 90, 150, 169, 5\rangle$ $CA_1 : \langle 6, 89, 150, 165, 90, 90, 165, 90, 150, 101, 150, 105, 89, 90, 90, 5\rangle$ $CA_2 : \langle 10, 150, 197, 165, 90, 165, 90, 150, 154, 165, 106, 105, 165, 105, 101, 5\rangle$
Heart failure clinical records	Silhouette = 0.6885, Dunn= 0.0945,	2	$CA_0 : \langle 9, 154, 90, 166, 105, 105, 154, 105, 150, 106, 150, 90, 150, 154, 90, 86, 165, 90, 5\rangle$ $CA_2 : \langle 9, 212, 165, 105, 166, 165, 101, 105, 150, 169, 150, 150, 105, 150, 90, 90, 105, 154, 80\rangle$
	Connectivity= 3.1718	2	$CA_0 : \langle 5, 135, 166, 165, 165, 105, 165, 90, 90, 90, 165, 105, 89, 105, 150, 165, 150, 90, 20\rangle$ $CA_1 : \langle 9, 149, 150, 105, 169, 90, 89, 150, 90, 150, 106, 90, 89, 90, 166, 165, 86, 105, 20\rangle$

connectivity, *silhouette score* and *Dunn index*. The next two columns (IV and V) represent the number of levels used to get the optimal results for each dataset and the corresponding ordered list of auxiliary CAs where any CA$_i$ is exclusively used for a particular level i. For example, in case of `Iris` dataset, the optimal scores for all three validation indices are achieved while clustering is completed in level 2. Moreover, same set of auxiliary CAs are used for all three cases. For

Table 6.7: Performance on reordering of features of Iris dataset where $m = 2$; for any feature ordering (first column), there exist auxiliary CAs (given by second column) that generate the set of clusters with same scores of validation indices (Silhouette = 0.6867, Dunn = 0.3389, Connectivity = 0.0000) as given in Table 6.6.

Order of features	Set of Auxiliary CAs	# Levels
$\langle A_1, A_2, A_3, A_4\rangle$	$CA_0 : \langle 5, 165, 101, 165, 89, 165, 106, 80\rangle$	1
$\langle A_1, A_2, A_4, A_3\rangle$	$CA_0 : \langle 9, 90, 90, 101, 150, 105, 106, 80\rangle$; $CA_1 : \langle 9, 141, 86, 90, 150, 105, 165, 20\rangle$	2
$\langle A_1, A_3, A_2, A_4\rangle$	$CA_0 : \langle 10, 105, 156, 86, 105, 150, 166, 80\rangle$; $CA_1 : \langle 10, 180, 169, 165, 86, 150, 90, 65\rangle$	2
$\langle A_1, A_4, A_2, A_3\rangle$	$CA_0 : \langle 10, 165, 101, 105, 90, 105, 106, 80\rangle$; $CA_1 : \langle 5, 90, 105, 149, 150, 165, 90, 20\rangle$	2
$\langle A_2, A_1, A_3, A_4\rangle$	$CA_0 : \langle 10, 105, 198, 150, 90, 106, 150, 20\rangle$; $CA_1 : \langle 9, 166, 90, 169, 105, 105, 166, 80\rangle$	2
$\langle A_2, A_1, A_4, A_3\rangle$	$CA_0 : \langle 6, 86, 90, 165, 150, 105, 166, 80\rangle$; $CA_1 : \langle 9, 150, 120, 198, 165, 165, 106, 80\rangle$; $CA_2 : \langle 10, 90, 169, 165, 169, 150, 90, 65\rangle$	3
$\langle A_2, A_3, A_1, A_4\rangle$	$CA_0 : \langle 10, 75, 166, 90, 166, 90, 106, 80\rangle$; $CA_1 : \langle 9, 89, 90, 165, 90, 106, 150, 20\rangle$; $CA_2 : \langle 6, 78, 150, 165, 86, 90, 89, 80\rangle$	3
$\langle A_2, A_4, A_1, A_3\rangle$	$CA_0 : \langle 5, 135, 105, 90, 142, 150, 106, 80\rangle$; $CA_1 : \langle 9, 86, 165, 86, 165, 105, 105, 65\rangle$	2
$\langle A_3, A_1, A_2, A_4\rangle$	$CA_0 : \langle 10, 150, 172, 90, 165, 105, 169, 5\rangle$; $CA_1 : \langle 10, 180, 90, 165, 150, 101, 105, 65\rangle$; $CA_2 : \langle 6, 106, 150, 150, 106, 90, 86, 80\rangle$; $CA_3 : \langle 10, 30, 172, 105, 105, 105, 165, 65\rangle$	4
$\langle A_3, A_1, A_4, A_2\rangle$	$CA_0 : \langle 6, 216, 150, 90, 169, 105, 90, 65\rangle$; $CA_1 : \langle 10, 150, 156, 89, 165, 150, 105, 65\rangle$; $CA_2 : \langle 6, 212, 90, 90, 150, 106, 105, 65\rangle$	3
$\langle A_3, A_2, A_1, A_4\rangle$	$CA_0 : \langle 6, 105, 120, 105, 101, 90, 106, 80\rangle$; $CA_1 : \langle 6, 169, 90, 150, 105, 150, 166, 80\rangle$; $CA_2 : \langle 9, 232, 90, 105, 165, 150, 90, 65\rangle$	3
$\langle A_3, A_2, A_4, A_1\rangle$	$CA_0 : \langle 6, 105, 120, 105, 101, 90, 106, 80\rangle$; $CA_1 : \langle 6, 169, 90, 150, 105, 150, 166, 80\rangle$; $CA_2 : \langle 9, 232, 90, 105, 165, 150, 90, 65\rangle$	3
$\langle A_3, A_4, A_1, A_2\rangle$	$CA_0 : \langle 10, 30, 92, 105, 105, 105, 165, 65\rangle$; $CA_1 : \langle 6, 216, 105, 105, 165, 105, 105, 80\rangle$; $CA_2 : \langle 10, 120, 198, 169, 165, 105, 150, 65\rangle$	3
$\langle A_3, A_4, A_2, A_1\rangle$	$CA_0 : \langle 10, 150, 108, 105, 165, 166, 105, 65\rangle$; $CA_1 : \langle 6, 165, 165, 78, 106, 165, 90, 80\rangle$	2
$\langle A_4, A_1, A_2, A_3\rangle$	$CA_0 : \langle 6, 90, 212, 150, 106, 90, 89, 5\rangle$; $CA_1 : \langle 10, 180, 149, 105, 150, 86, 105, 65\rangle$; $CA_2 : \langle 5, 105, 154, 165, 101, 165, 166, 80\rangle$	3
$\langle A_4, A_1, A_3, A_2\rangle$	$CA_0 : \langle 10, 30, 172, 105, 166, 165, 101, 80\rangle$; $CA_1 : \langle 9, 228, 150, 150, 105, 101, 150, 65\rangle$	2
$\langle A_4, A_2, A_1, A_3\rangle$	$CA_0 : \langle 6, 228, 86, 105, 150, 149, 105, 65\rangle$; $CA_1 : \langle 6, 90, 232, 90, 150, 154, 150, 65\rangle$; CA_2: $(10, 120, 86, 90, 150, 165, 169, 5)$;$CA_3$: $(9, 78, 166, 105, 105, 86, 150, 65)$	4
$\langle A_4, A_2, A_3, A_1\rangle$	$CA_0 : \langle 6, 212, 150, 105, 90, 150, 105, 20\rangle$; $CA_1 : \langle 10, 210, 149, 90, 89, 105, 90, 20\rangle$	2
$\langle A_4, A_3, A_2, A_1\rangle$	CA_0: $(6, 216, 105, 165, 90, 150, 165, 20)$	1
$\langle A_4, A_3, A_1, A_2\rangle$	$CA_0 : \langle 10, 120, 166, 150, 150, 165, 105, 20\rangle$; $CA_1 : \langle 10, 120, 166, 150, 150, 165, 105, 20\rangle$	2

`Buddymove` dataset also, the best scores are found for all three indices using same

Table 6.8: Performance of other datasets on different ordering of features where $m = 2$

Dataset	Order of features	Set of Auxiliary CAs	Level Level	Scores of indices
Buddy -move	$\langle A_1, A_2, A_3, A_4, A_5, A_6\rangle$	CA_0: $\langle 10, 210, 156, 106, 90, 106, 150, 150, 105, 165, 86, 80\rangle$ CA_1: $\langle 10, 165, 201, 154, 165, 169, 90, 154, 150, 165, 165, 20\rangle$ CA_2: $\langle 5, 135, 86, 105, 105, 165, 90, 89, 150, 150, 105, 5\rangle$ CA_3: $\langle 5, 30, 89, 90, 105, 165, 86, 165, 101, 165, 166, 80\rangle$ CA_4: $\langle 9, 166, 165, 169, 105, 90, 150, 89, 150, 105, 149\rangle$	5	Silhouette= 0.4763, Dunn= 0.3146, Connectivity= 2.9289
	$\langle A_2, A_4, A_3, A_1, A_6, A_5\rangle$	CA_0: $\langle 9, 86, 105, 90, 105, 149, 165, 86, 105, 90, 165, 65\rangle$	1	
	$\langle A_3, A_4, A_1, A_2, A_5, A_6\rangle$	CA_0: $\langle 5, 135, 172, 150, 154, 105, 150, 90, 150, 105, 89, 80\rangle$ CA_1: $\langle 6, 90, 105, 120, 92, 90, 165, 90, 150, 106, 105, 65\rangle$ CA_2: $\langle 9, 149, 90, 101, 90, 106, 90, 90, 105, 105, 105, 80\rangle$	3	
Stone-Flakes	$\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\rangle$	CA_0 : $\langle 6, 86, 165, 149, 165, 89, 165, 90, 90, 101, 105, 90, 150, 169, 150, 65\rangle$ CA_1: $\langle 10, 135, 154, 165, 165, 150, 165, 90, 105, 106, 165, 154, 150, 150, 154, 5\rangle$ CA_2: $\langle 5, 30, 166, 165, 165, 150, 90, 65\rangle$	3	Silhouette= 0.376
	$\langle A_4, A_5, A_6, A_7, A_8, A_1, A_2, A_3\rangle$	CA_0: $\langle 6, 142, 105, 101, 165, 166, 90, 169, 105, 105, 166, 90, 89, 90, 86, 80\rangle$ CA_1: $\langle 6, 89, 105, 165, 90, 105, 154, 165, 150, 165, 169, 165, 89, 150, 165, 65\rangle$	2	Silhouette= 0.3377
	$\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\rangle$	CA_0: $\langle 6, 166, 105, 105, 89, 90, 165, 165, 166, 150, 150, 150, 90, 106, 150, 20\rangle$ CA_1: $\langle 6, 166, 150, 165, 165, 105, 105, 90, 169, 105, 90, 150, 89, 105, 165, 65\rangle$ CA_2: $\langle 5, 210, 92, 105, 149, 90, 150, 105, 90, 165, 105, 106, 105, 165, 165, 65\rangle$ CA_3: $\langle 5, 165, 92, 150, 165, 90, 169, 105, 105, 89, 105, 90, 150, 89, 150, 20\rangle$	4	Dunn= 0.1805, Connectivity= 3.5123
	$\langle A_4, A_5, A_6, A_7, A_8, A_1, A_2, A_3\rangle$	CA_0: $\langle 9, 166, 150, 165, 105, 149, 165, 90, 90, 150, 150, 165, 165, 105, 86, 5\rangle$	1	
Seed	$\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7\rangle$	CA_0 : $\langle 9, 228, 169, 165, 101, 90, 90, 90, 154, 150, 150, 105, 105, 20\rangle$ CA_1 : $\langle 5, 105, 89, 105, 105, 105, 150, 150, 150, 150, 165, 165, 165, 20\rangle$	2	Silhouette= 0.5288
	$\langle A_1, A_2, A_5, A_6, A_7, A_3, A_4\rangle$	CA_0 : $\langle 10, 225, 198, 149, 150, 90, 105, 89, 150, 90, 90, 90, 165, 65\rangle$ CA_1 : $\langle 6, 77, 165, 90, 150, 106, 150, 90, 150, 150, 150, 90, 90, 65\rangle$ CA_2 : $\langle 9, 169, 90, 106, 150, 105, 150, 105, 105, 89, 105, 105, 154, 80\rangle$ CA_3 : $\langle 9, 101, 105, 150, 149, 105, 165, 105, 169, 105, 165, 165, 90, 65\rangle$	4	Silhouette= 0.5282
	$\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7\rangle$	CA_0 : $\langle 5, 120, 197, 150, 86, 165, 106, 165, 166, 105, 90, 105, 166, 5\rangle$ CA_1 : $\langle 5, 210, 165, 180, 202, 165, 150, 101, 90, 90, 105, 165, 165, 65\rangle$ CA_2 : $\langle 10, 120, 90, 105, 172, 150, 86, 150, 105, 154, 150, 165, 165, 65\rangle$ CA_3 : $\langle 5, 120, 172, 165, 150, 149, 150, 150, 169, 150, 105, 166, 105, 20\rangle$	4	Dunn= 0.09, Connectivity = 3.91
	$\langle A_1, A_2, A_5, A_6, A_7, A_3, A_4\rangle$	CA_0 : $\langle 5, 75, 154, 150, 90, 105, 106, 150, 165, 90, 105, 166, 150, 65\rangle$ CA_1 : $\langle 9, 86, 105, 105, 166, 165, 154, 90, 166, 150, 90, 165, 165, 65\rangle$ CA_2 : $\langle 6, 86, 150, 105, 89, 90, 101, 90, 105, 90, 105, 150, 165, 65\rangle$	3	Dunn= 0.08, Connectivity = 3.76

set of auxiliary CAs. For other datasets, the same set of CAs has been used to produce the *best* quality clusters with respect to *two* validation indices.

Next, we show that our algorithm can produce similar clusters even if the ordering of features is changed – such claim is verified by the extensive experimentation on the real datasets used here.

6.6.2 Experimentation on Datasets after Reordering of Features

Every target object is represented by its p feature values if it owns p distinct features $A_1, A_2, \cdots, A_p$. Generally, an object is *independent* of ordering of its features. But as CA is used, therefore, the mapping of such target object to the useful configuration can be changed if the ordering of its features is altered.

However, if we can get two set of auxiliary CAs $\mathscr{R}^1$ and $\mathscr{R}^2$ which clusters the given sets of target objects in similar fashion irrespective of the ordering of elements, then we name the set of auxiliary CAs $\mathscr{R}^1$ and $\mathscr{R}^2$ as *Equivalent Auxiliary CA spaces.* Table 6.7 and 6.8 show that our iterative level-wise clustering algorithm also supports reordering of features.

Let us first take `Iris` dataset where each object is represented by the attributes A_1, A_2, A_3 and A_4. As we use CA, each object is mapped to a configuration. Till now, the experimentation is done on this dataset using the feature ordering as $\langle A_1, A_2, A_3, A_4\rangle$. However, actually by changing the ordering of the features, we can get, $4! = 24$ such distinct combinations. Some of them are $\langle A_2, A_3, A_1, A_4\rangle$, $\langle A_3, A_4, A_2, A_1\rangle$, etc. Evidently, when ordering of features is changed, the corresponding useful configuration is also changed. Therefore, another set of CAs can be used for clustering that set of useful configurations. Let, here also, the desired number of clusters is 2. Table 6.7 reports the result of this experiment on all possible ordering. From Table 6.7, it can be concluded that the count of clusters and the arrangement of clusters remain same even after altering the order of the features. Table 6.7 also shows that there exist several different sets of auxiliary CAs which contribute to same clusters. For instance, $\mathscr{R}^1 = \{\langle 5, 165, 101, 165, 89, 165, 106, 80\rangle\}$, $\mathscr{R}^2 = \{\langle 9, 90, 90, 101, 150, 105, 106, 80\rangle, \langle 9, 141, 86, 90, 150, 105, 165, 20\rangle\}$ and $\mathscr{R}^3 = \{\langle 10, 150, 108, 105, 165, 166, 105, 65\rangle, \langle 6, 165, 165, 78, 106, 165, 90, 80\rangle\}$ are *three* sets of auxiliary CAs for the given feature orders $\langle A_1, A_2, A_3, A_4\rangle$, $\langle A_1, A_2, A_4, A_3\rangle$ and $\langle A_3, A_4, A_2, A_1\rangle$ respectively. These $\mathscr{R}^1$, $\mathscr{R}^2$ and $\mathscr{R}^3$ forms equivalent clustering of the objects; hence these are *Equivalent auxiliary CA spaces.* Moreover, it is also observed that each of them produces the *best* clusters with same Silhouette score, Dunn index and Connectivity value.

However, it is not always necessary that, exactly same cluster will be formed for every possible reordering. They can be almost similar, which can be indicated by nearly same score in the validation indices (see result of Seed dataset on Table 6.8). For example, in Table 6.8, for the Buddymove dataset, each of three distinct ordering of features can be clustered using equivalent auxiliary CA spaces giving exactly same value on the three validation indices. Whereas, for StoneFlakes dataset, equivalent auxiliary CA spaces can be found with respect to Dunn index and Connectivity for the two orderings $\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\rangle$ and $\langle A_4, A_5, A_6, A_7, A_8, A_1, A_2, A_3\rangle$. Now, we can compare our algorithm with some existing well-known techniques.

6.6.3 Comparison with Existing Clustering Techniques

To judge the efficiency of our iterative clustering model, we need to compare the results of the validation indices achieved by our technique with the exist-

Table 6.9: Comparison of clustering techniques based on internal validation indices for each of the available datasets of Table 6.5. Here, $m = 2$ for all entries except those marked with * for which $m = 6$.

Dataset	Algorithm	Connectivity	Dunn Index	Silhouette Score
Iris	Hierarchical	0.0000	0.3389	0.6867
	K-means	6.1536	0.0765	0.6810
	DIANA	6.1536	0.0765	0.6810
	PAM	3.9623	0.0811	0.6858
	SOTA	11.5016	0.0349	0.6569
	CA-based Clustering	**0.0000**	**0.3389**	**0.6867**
User Knowledge Modeling	Hierarchical	3.0540	0.1970	0.2589
	K-means	50.3321	0.0943	0.2063
	DIANA	68.7329	0.0297	0.2066
	PAM	102.8329	0.0537	0.1934
	SOTA	64.1060	0.0321	0.2042
	CA-based Clustering	**3.6678**	**0.1508**	**0.1867**
Buddymove	Hierarchical	2.9290	0.3146	0.4764
	K-means	35.2032	0.0485	0.3079
	DIANA	34.3694	0.0658	0.3020
	PAM	38.3802	0.0485	0.3055
	SOTA	44.4111	0.0518	0.3134
	CA-based Clustering	**2.9289**	**0.3146**	**0.4763**
Heart failure clinical records	Hierarchical	5.7536	0.1506	0.7862
	K-means	1.8635	0.0079	0.5829
	DIANA	1.8635	0.0079	0.5829
	PAM	3.5020	0.0036	0.4630
	SOTA	5.0972	0.0035	0.5302
	CA-based Clustering	**3.1718**	**0.0945**	**0.6885**
StoneFlakes	Hierarchical	2.9290	0.3176*	0.2937
	K-means	4.9488	0.1735	0.4875
	DIANA	4.9488	0.1735	0.4875
	PAM	7.9762	0.1693	0.4827
	SOTA	4.9488	0.1735	0.4875
	CA-based Clustering	**3.5123**	**0.1805**	**0.376**
Seed	Hierarchical	8.7861	0.1065*	0.5248
	K-means	21.3698	0.0548	0.5229
	DIANA	19.1714	0.0544	0.5218
	PAM	20.6762	0.0404	0.5175
	SOTA	16.0179	0.0361	0.5049
	CA-based Clustering	**3.91**	**0.09**	**0.528**
Wholesale Customers	Hierarchical	2.9290	0.3853	0.7957
	K-means	30.0881	0.0178	0.5115
	DIANA	27.1242	0.0313	0.5806
	PAM	41.6647	0.0167	0.3819
	SOTA	43.8266	0.0061	0.3699
	CA-based Clustering	**3.7329**	**0.0508**	**0.5257**

ing benchmark clustering algorithms. Here, we use five benchmark clustering algorithms – *K-means* (centroid based clustering) [71], *hierarchical* (agglomerative hierarchical clustering) [71], *DIANA* (divisive hierarchical clustering) [71], *PAM (Partitioning around medoids)* (centroid based clustering) [71] and *SOTA (Self-organizing tree algorithm)* (unsupervised network with a divisive hierarchical clustering) [71] using the implementation in **R** [71]. Table 6.9 reports result of this comparison. From the experiments, it can be noted that, among the existing algorithms, for the StoneFlakes dataset, optimal silhouette score is found by `K-means, DIANA, SOTA`, whereas, the best connectivity score for Heart failure clinical records dataset is achieved by using `K-means` and `DIANA` algorithms. However, by overall performance on all datasets, the *hierarchical* algorithm forms clusters most *effectively*. Further, Table 6.9 also shows that, the validation indices scores obtained by our algorithm can compete with the best scores obtained by the benchmarks algorithms. Hence, our algorithm is one of the best algorithm existing today for clustering any kind of dataset.

Till now as an auxiliary CA we consider non-uniform ECAs. Next, we want to use the beauty of classical CAs by increasing the radius of 1-dimensional CAs.

6.7 Cellular Automata with radius $r > 1$ and Clustering

Section 6.4 has introduced the notion of Cycle-based Clustering. A smarter approach has been proposed in Section 6.5 and the experimental results using popular datasets and the performance analysis of the proposed algorithm (based on the cyclic spaces of CAs) have been reported in Section 6.6. In our proposed clustering technique, we use multiple levels and merging of the clusters is based on closeness metric (*closely reachable*). We worked out on non-uniform reversible ECAs where the radius (r) is 1. In such case, we consider one continuous attribute and at most one categorical attribute due to the design of our encoding scheme (that is, each continuous attribute is encoded into 2 bits and if there are at most three values for a categorical attribute). Now, we want to increase the radius of the CA. The reason is that increase in radius produces more number of possible auxiliary CAs which may help us to get better result using the same algorithm on the same dataset. Another key point of increasing the radius of CA is to consider the auxiliary CAs are dependent of more than one attribute which may affect in the results. We are attracted to see performance of our algorithm on reversible CAs with higher radius (more than one). For higher radius, we expect an extensive number of uniform CAs. So, for our work, we consider only uniform reversible CAs with radius more than one.

In this work, we use one-dimensional n-cell CAs under null boundary condition, where each cell changes its state depending on the present states of itself and its r number of consecutive neighbors at left and right sides. Next we report some terminologies in CAs which are applicable for any given radius r. Then, it introduces how fundamental structure of an n-cell reversible CA can be used for clustering any set of target objects represented in bits.

Table 6.10: Some $r = 2$ CA rules. RMT is represented in decimal format

RMT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Rule Number
	0	0	1	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0	1	1	0	0	869020620
R	1	0	1	1	0	1	1	0	1	0	0	1	0	1	0	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	3063191190
	1	0	1	1	0	0	0	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	2966606546

6.7.1 Definitions

This work uses $D = 1$, $\mathscr{L} = \{0, 1, \cdots, n-1\}$, $\mathcal{S} = \{0, 1\}$, $R : \mathcal{S}^{2r+1} \to \mathcal{S}$ and for each cell, its missing neighbors are treated as 0 to constitute null boundary condition. An input $\langle s_{-r}, \cdots, s_0, \cdots, s_r \rangle$ to R is called as *Rule Min Term (RMT)*. Each RMT is represented by its equivalent decimal number $\mathtt{r} = 2^{2r} \cdot s_{-r} + 2^{2r-1} \cdot s_{-r+1} + \cdots + 2^r \cdot s_0 + \cdots + 2 \cdot s_{r-1} + s_r$, whereas $R(s_{-r}, \cdots, s_0, \cdots, s_r)$ is presented by $R[\mathtt{r}]$; $-r \leq j \leq r$ is the neighborhood index. The number of possible RMTs for a CA is 2^{2r+1} (see Table 6.10). We represent a rule by the string "$R[\mathtt{2^{2r+1}-1}] \cdots R[\mathtt{1}]R[\mathtt{0}]$" or its decimal equivalent, also called rule number. For ease of reference, we shall indicate a rule by its rule number only.

The present states of all cells at a given time is called the *configuration* of a CA. Thus, evolution of a CA is determined by a *global transition function* $G : \mathcal{C} \to \mathcal{C}$, where $\mathcal{C} = \{0, 1\}^n$ represents the configuration space, that is the set of all configurations of an n-cell CA. Hence, if the next configuration of $\mathtt{x} = (x_i)_{\forall i \in n}$ is $\mathtt{y} = (y_i)_{\forall i \in n}$, then $\mathtt{y} = G(\mathtt{x}) = G(x_0 x_1 \cdots x_{n-1}) = (R(x_{i-r}, \cdots, x_i, \cdots, x_{i+r}))_{0 \leq i \leq n-1}$, where $\mathtt{x}, \mathtt{y} \in \mathcal{C}$ and x_i, y_i are the present and next state values of cell i respectively. Here, $\langle x_{i-r}, \cdots, x_i, \cdots, x_{i+r} \rangle$ is the RMT for cell i and $R(x_{i-r}, \cdots, x_i, \cdots, x_{i+r}) = y_i$. Hence, a configuration can also be represented as an *RMT sequence.*

Definition 1 *A set of RMTs are called* m-equivalent *or* $\mathcal{E}^m$*, if the values of all neighbors of them are invariant except the* m^{th} *neighbor* ($-r \leq m \leq r$)*. Mathematically,* $\mathcal{E}_i^m = \{a_{-r} a_{-r+1} \cdots a_{m-1} \mathtt{x} a_{m+1} \cdots a_r \in \mathcal{S}^{2r+1} \mid \mathtt{x} \in \mathcal{S}\}$*. Here,* i *is the decimal equivalent of* $a_{-r} a_{-r+1} \cdots a_{m-1} a_{m+1} \cdots a_r$*.*

Obviously, for each m, $-r \leq m \leq r$, there are 2^{2r} number of such equivalent sets indexed from 0 to $2^{2r} - 1$. Table 6.11 shows the relationship among the RMTs of binary CAs with $r = 2$. An interesting relation is followed in the RMT sequence: if $\mathtt{r} \in \mathcal{E}_i^{-r}$, then the next RMT in the sequence $\mathtt{s} \in \mathcal{E}_i^r$ $(0 \leq i \leq 2^{2r} - 1)$.

Definition 2 *An RMT* $r = 2^{2r} \cdot x_{i-r} + 2^{2r-1} \cdot x_{i-r+1} + \cdots + 2^r \cdot x_i + 2^{r-1} \cdot x_{i+1} + \cdots + x_{i+r}$ *is said to be self-replicating, if* $R(x_{i-r}, \cdots, x_i, \cdots, x_{i+r}) = x_i$*, where* R *is the rule of the CA.*

If all RMTs of a configuration $\mathtt{x}$ is self-replicating, then its next configuration $\mathtt{y}$ is *identical* to it and they form *a cycle of length one.* Such a cycle is called

Table 6.11: Relations among the different Equivalent RMTs for $r = 2$ CAs

#Set	$\mathcal{E}_j^{-2}$		$\mathcal{E}_j^{-1}$		$\mathcal{E}_j^{0}$		$\mathcal{E}_j^{1}$		$\mathcal{E}_j^{2}$	
(j)	**RMTs**	**Decimal**	**RMTs**	**Decimal**	**RMTs**	**Decimal**	**RMTs**	**Decimal**	**RMTs**	**Decimal**
0	00000, 10000	0, 16	00000, 01000	0, 8	00000, 00100	0, 4	00000, 00010	0, 2	00000, 00001	0, 1
1	00001, 10001	1, 17	00001, 01001	1, 9	00001, 00101	1, 5	00001, 00011	1, 3	00010, 00011	2, 3
2	00010, 10010	2, 18	00010, 01010	2, 10	00010, 00110	2, 6	00100, 00110	4, 6	00100, 00101	4, 5
3	00011, 10011	3, 19	00011, 01011	3, 11	00011, 00111	3, 7	00101, 00111	5, 7	00110, 00111	6, 7
4	00100, 10100	4, 20	00100, 01100	4, 12	01000, 01100	8, 12	01000, 01010	8, 10	01000, 01001	8, 9
5	00101, 10101	5, 21	00101, 01101	5, 13	01001, 01101	9, 13	01001, 01011	9, 11	01010, 01011	10, 11
6	00110, 10110	6, 22	00110, 01110	6, 14	01010, 01110	10, 14	01100, 01110	12, 14	01100, 01101	12, 13
7	00111, 10111	7, 23	00111, 01111	7,15	01011, 01111	11, 15	01101, 01111	13, 15	01110, 01111	14, 15
8	01000, 11000	8, 24	10000, 11000	16, 24	10000, 10100	16, 20	10000, 10010	16, 18	10000, 10001	16, 17
9	01001, 11001	9, 25	10001, 11001	17, 25	10001, 10101	17, 21	10001, 10011	17, 19	10010, 10011	18, 19
10	01010, 11010	10, 26	10010, 11010	18, 26	10010, 10110	18, 22	10100, 10110	20, 22	10100, 10101	20, 21
11	01011, 11011	11, 27	10011, 11011	19, 27	10011, 10111	19, 23	10101, 10111	21, 23	10110, 10111	22, 23
12	01100, 11100	12, 28	10100, 11100	20, 28	11000, 11100	24, 28	11000, 11010	24, 26	11000, 11001	24, 25
13	01101, 11101	13, 29	10101, 11101	21, 29	11001, 11101	25, 29	11001, 11011	25, 27	11010, 11011	26, 27
14	01110, 11110	14, 30	10110, 11110	22, 30	11010, 11110	26, 30	11100, 11110	28, 30	11100, 11101	28, 29
15	01111, 11111	15, 31	10111, 11111	23, 31	11011, 11111	27, 31	11101, 11111	29, 31	11110, 11111	30, 31

a *fixed-point attractor*. For instance, let a CA has 0 at RMT `0`, that is, RMT `0` is self-replicating, then there exists a fixed point at 0^n, for any cell length n. Thus, the number of self replicating RMTs for a rule plays a major role in cycle formation, that is, in the number of cycles and lengths of cycles. Next section describes a synthesis scheme to find reversible CAs for our purpose.

6.7.2 Reversibility

In this section, we extend an established discrete mathematical tool, named *Reachability tree*, for an n-cell 2-state r-radius CA under null-boundary condition ($n \geq 2r+1$). Reachability tree was initially proposed for binary CAs [45, 46], and later generalized for d-state CAs under *periodic-boundary* condition [52]. It depicts all reachable configurations of an n-cell CA and helps to efficiently decide whether a given n-cell CA is reversible. For an r-radius binary CA under null-boundary condition, the reachability tree can be defined as following –

Definition 3 *Reachability tree of an n-cell r-radius 2-state CA is a rooted and edge-labeled binary tree with $(n+1)$ levels where each node $N_{i.j}$ $(0 \leq i \leq n,\ 0 \leq j \leq 2^i - 1)$ contains a set of RMTs. We denote the edges between $N_{i.j}$ $(0 \leq i \leq n-1,\ 0 \leq j \leq 2^i - 1)$ and its two possible children as $E_{i.2j} = (N_{i.j}, N_{i+1.2j}, l_{i.2j})$ and $E_{i.2j+1} = (N_{i.j}, N_{i+1.2j+1}, l_{i.2j+1})$ where $l_{i.2j+x}$ is the label of the edge and $x \in \{0,1\}$. Like nodes, the labels also contain a set of RMTs. Following are the relations that exist in the tree:*

1. *[For root] $N_{0.0} = \bigcup_k \mathcal{E}_k^r$ where $0 \leq k \leq 2^r - 1$ and $\mathcal{E}_k^r$ is the k^{th}* r-equivalent *RMT set (see Definition 1).*

2. *$\forall \mathtt{s} \in N_{i.j}$, $\mathtt{s}$ is included in $E_{i.2j}$ (resp. $E_{i.2j+1}$), if $R[\mathtt{s}] = 0$ (resp. 1), where R is the rule of the CA.*

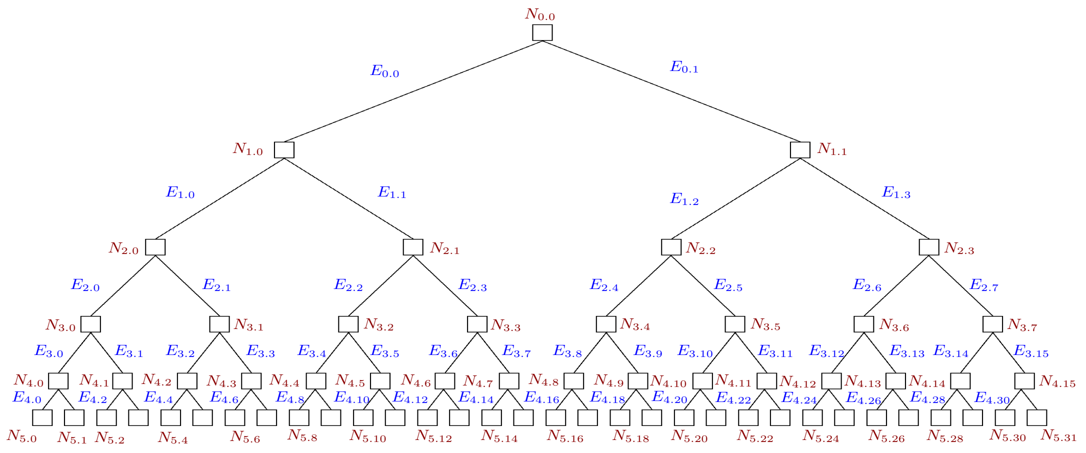

Figure 6.3: Reachability tree for 5-cell 2-state $r = 2$ CA 3063191190

3. *$\forall$s, if s $\in E_{i.2j}$ (resp. $E_{i.2j+1}$), then the RMTs of $\mathcal{E}_p^r = \{2\mathtt{s}, 2\mathtt{s}+1\} \subseteq N_{i+1.2j}$ (resp. $N_{i+1.2j+1}$), where $0 \leq i \leq n-1$, $0 \leq j \leq 2^i - 1$ and* $\mathtt{s} \equiv p \pmod{2^{2r}}$.

4. *[For level $n - \iota$, $1 \leq \iota \leq r$] $N_{n-\iota.j} = \{\mathtt{y} \mid$ if $\mathtt{s} \in E_{n-\iota-1.j}$, then $\mathtt{y} \in \{2\mathtt{s}, 2\mathtt{s}+1\} \cap \{0, 2^{r-\iota+1}, \cdots, (2^{r+\iota}-1)2^{r-\iota+1}\}\}$ where $0 \leq j \leq 2^{n-\iota} - 1$.*

5. *[For level n] $N_{n.j} = \varnothing$, where $0 \leq j \leq 2^n - 1$.*

It can be observed that, the nodes of level $n - \iota$, $1 \leq \iota \leq r$, are different from other intermediate nodes (Points 4 of Definition 3). At level $n-\iota$, $1 \leq \iota \leq r$, only some specific RMTs can be present, these RMTs are called *valid* RMTs for this level. For example, at level $n - \iota$, $1 \leq \iota \leq r$, only the RMTs which are divisible by $2^{r-\iota+1}$ can be valid. In a reachability tree, the root is at level 0 and the leaves are at level n. A sequence of edges from root to leaf, $\langle E_{0.j_1}, E_{1.j_2}, ..., E_{n-1.j_n}\rangle$, where $0 \leq j_i \leq 2^i - 1, 1 \leq i \leq n$, constitutes an RMT sequence, where each edge represents the state of a cell. If an edge (resp. node) has no RMT, then it is non-reachable edge (resp. node) and represents a non-reachable configuration.

Example 6.7 The reachability tree for the 2-state $r = 2$ CA 3063191190 (10110110100101001001011010010110) with 5-cells is shown in Figure 6.3. The details of the nodes and edges of it is shown in Table 6.12. This tree contains no non-reachable nodes or edges, so the tree is complete. One reachable configuration 11010 is represented by the sequence $\langle E_{0.1}, E_{1.3}, E_{2.6}, E_{3.13}, E_{4.26}\rangle$.

Table 6.12: Reachability Tree for CA 2-state $r = 2$ CA 3063191190 (Figure 6.3) with $n = 5$

Level	Nodes	Edges
0	$N_{0.0} = \{0,1,2,3,4,5,6,7\}$	NA
1	$N_{1.0} = \{0,1,6,7,10,11,12,13\}$	$E_{0.0} = \{0,3,5,6\}$
	$N_{1.1} = \{2,3,4,5,8,9,14,15\}$	$E_{0.1} = \{1,2,4,7\}$
2	$N_{2.0} = \{0,1,12,13,22,23,26,27\}$	$E_{1.0} = \{0,6,11,13\}$
	$N_{2.1} = \{2,3,14,15,20,21,24,25\}$	$E_{1.1} = \{1,7,10,12\}$
	$N_{2.2} = \{6,7,10,11,16,17,28,29\}$	$E_{1.2} = \{3,5,8,14\}$
	$N_{2.3} = \{4,5,8,9,18,19,30,31\}$	$E_{1.3} = \{2,4,9,15\}$
3	$N_{3.0} = \{0,1,12,13,22,23,26,27\} = N_{2.0}$	$E_{2.0} = \{0,13,22,27\}$
	$N_{3.1} = \{2,3,14,15,20,21,24,25\} = N_{2.1}$	$E_{2.1} = \{1,12,23,26\}$
	$N_{3.2} = \{6,7,10,11,16,17,28,29\} = N_{2.2}$	$E_{2.2} = \{3,14,21,24\}$
	$N_{3.3} = \{4,5,8,9,18,19,30,31\} = N_{2.3}$	$E_{2.3} = \{2,15,21,25\}$
	$N_{3.4} = \{0,1,2,3,12,13,22,23\}$	$E_{2.4} = \{6,11,16,17\}$
	$N_{3.5} = \{14,15,20,21,24,25,26,27\}$	$E_{2.5} = \{7,10,28,29\}$
	$N_{3.6} = \{6,7,10,11,16,17,28,29\} = N_{3.2}$	$E_{2.6} = \{5,8,19,30\}$
	$N_{3.7} = \{4,5,8,9,18,19,30,31\} = N_{3.3}$	$E_{2.7} = \{4,9,18,31\}$
4	$N_{4.0} = \{0,1,12,13,22,23,26,27\} = N_{3.0}$	$E_{3.0} = \{0,13,22,27\}$
	$N_{4.1} = \{2,3,14,15,20,21,24,25\} = N_{3.1}$	$E_{3.1} = \{1,12,23,26\}$
	$N_{4.2} = \{6,7,10,11,16,17,28,29\} = N_{3.2}$	$E_{3.2} = \{3,14,21,24\}$
	$N_{4.3} = \{4,5,8,9,18,19,30,31\} = N_{3.3}$	$E_{3.3} = \{2,15,21,25\}$
	$N_{4.4} = \{0,1,2,3,12,13,22,23\} = N_{3.4}$	$E_{3.4} = \{6,11,16,17\}$
	$N_{4.5} = \{14,15,20,21,24,25,26,27\} = N_{3.5}$	$E_{3.5} = \{7,10,28,29\}$
	$N_{4.6} = \{6,7,10,11,16,17,28,29\} = N_{4.2}$	$E_{3.6} = \{5,8,19,30\}$
	$N_{4.7} = \{4,5,8,9,18,19,30,31\} = N_{4.3}$	$E_{3.7} = \{4,9,18,31\}$
	$N_{4.8} = \{0,1,6,7,12,13,26,27\}$	$E_{3.8} = \{0,3,13,22\}$
	$N_{4.9} = \{2,3,4,5,14,15,24,25\}$	$E_{3.9} = \{1,2,12,23\}$
	$N_{4.10} = \{10,11,16,17,22,23,28,29\}$	$E_{3.10} = \{14,21,24,27\}$
	$N_{4.11} = \{8,9,18,19,20,21,30,31\}$	$E_{3.11} = \{15,20,25,26\}$
	$N_{4.12} = \{0,1,2,3,12,13,22,23\} = N_{4.4}$	$E_{3.12} = \{6,11,16,17\}$
	$N_{4.13} = \{14,15,20,21,24,25,26,27\} = N_{4.5}$	$E_{3.13} = \{7,10,28,29\}$
	$N_{4.14} = \{6,7,10,11,16,17,28,29\} = N_{4.6}$	$E_{3.14} = \{5,8,19,30\}$
	$N_{4.15} = \{4,5,8,9,18,19,30,31\} = N_{4.7}$	$E_{3.15} = \{4,9,18,31\}$
5	$N_{5.0} = \{0,1,12,13,22,23,26,27\} = N_{4.0}$	$E_{4.0} = \{0,13,22,27\}$
	$N_{5.1} = \{2,3,14,15,20,21,24,25\} = N_{4.1}$	$E_{4.1} = \{1,12,23,26\}$
	$N_{5.2} = \{6,7,10,11,16,17,28,29\} = N_{4.2}$	$E_{4.2} = \{3,14,21,24\}$
	$N_{5.3} = \{4,5,8,9,18,19,30,31\} = N_{4.3}$	$E_{4.3} = \{2,15,21,25\}$
	$N_{5.4} = \{0,1,2,3,12,13,22,23\} = N_{4.4}$	$E_{4.4} = \{6,11,16,17\}$
	$N_{5.5} = \{14,15,20,21,24,25,26,27\} = N_{4.5}$	$E_{4.5} = \{7,10,28,29\}$
	$N_{5.6} = \{6,7,10,11,16,17,28,29\} = N_{5.2}$	$E_{4.6} = \{5,8,19,30\}$
	$N_{5.7} = \{4,5,8,9,18,19,30,31\} = N_{5.3}$	$E_{4.7} = \{4,9,18,31\}$
	$N_{5.8} = \{0,1,6,7,12,13,26,27\} = N_{4.8}$	$E_{4.8} = \{0,3,13,22\}$
	$N_{5.9} = \{2,3,4,5,14,15,24,25\} = N_{4.9}$	$E_{4.9} = \{1,2,12,23\}$
	$N_{5.10} = \{10,11,16,17,22,23,28,29\} = N_{4.10}$	$E_{4.10} = \{14,21,24,27\}$
	$N_{5.11} = \{8,9,18,19,20,21,30,31\} = N_{4.11}$	$E_{4.11} = \{15,20,25,26\}$
	$N_{5.12} = \{0,1,2,3,12,13,22,23\} = N_{5.4}$	$E_{4.12} = \{6,11,16,17\}$
	$N_{5.13} = \{14,15,20,21,24,25,26,27\} = N_{5.5}$	$E_{5.13} = \{7,10,28,29\}$
	$N_{5.14} = \{6,7,10,11,16,17,28,29\} = N_{5.6}$	$E_{4.14} = \{5,8,19,30\}$
	$N_{5.15} = \{4,5,8,9,18,19,30,31\} = N_{5.7}$	$E_{4.15} = \{4,9,18,31\}$
	$N_{5.16} = \{0,1,12,13,22,23,26,27\} = N_{5.0}$	$E_{4.16} = \{0,13,22,27\}$
	$N_{5.17} = \{2,3,14,15,20,21,24,25\} = N_{5.1}$	$E_{4.17} = \{1,12,23,26\}$
	$N_{5.18} = \{6,7,10,11,16,17,28,29\} = N_{5.2}$	$E_{4.18} = \{3,14,21,24\}$
	$N_{5.19} = \{4,5,8,9,18,19,30,31\} = N_{5.3}$	$E_{4.19} = \{2,15,21,25\}$
	$N_{5.20} = \{0,1,2,3,12,13,22,23\} = N_{5.4}$	$E_{4.20} = \{6,11,16,17\}$
	$N_{5.21} = \{14,15,20,21,24,25,26,27\} = N_{5.5}$	$E_{4.21} = \{7,10,28,29\}$
	$N_{5.22} = \{6,7,10,11,16,17,28,29\} = N_{5.2}$	$E_{4.22} = \{5,8,19,30\}$
	$N_{5.23} = \{4,5,8,9,18,19,30,31\} = N_{5.3}$	$E_{4.23} = \{4,9,18,31\}$
	$N_{5.24} = \{0,1,6,7,12,13,26,27\} = N_{5.8}$	$E_{4.24} = \{0,3,13,22\}$

Continued on next page

Table 6.12 – *Continued from previous page*

Level	**Nodes**	**Edges**
	$N_{5.25} = \{2,3,4,5,14,15,24,25\} = N_{5.9}$	$E_{4.25} = \{1,2,12,23\}$
	$N_{5.26} = \{10,11,16,17,22,23,28,29\} = N_{4.10}$	$E_{4.26} = \{14,21,24,27\}$
	$N_{5.27} = \{8,9,18,19,20,21,30,31\} = N_{5.11}$	$E_{4.27} = \{15,20,25,26\}$
	$N_{5.28} = \{0,1,2,3,12,13,22,23\} = N_{5.4}$	$E_{4.28} = \{6,11,16,17\}$
	$N_{5.29} = \{14,15,20,21,24,25,26,27\} = N_{5.5}$	$E_{5.29} = \{7,10,28,29\}$
	$N_{5.30} = \{6,7,10,11,16,17,28,29\} = N_{5.6}$	$E_{4.30} = \{5,8,19,30\}$
	$N_{5.31} = \{4,5,8,9,18,19,30,31\} = N_{5.7}$	$E_{4.31} = \{4,9,18,31\}$

□

One can observe that, although the number of levels in the tree grows with CA size n, there are only a few unique nodes at each level. For instance, in the reachability tree of Example 6.7, out of 32 nodes of level 5, there are only 10 unique nodes (see Table 6.12). Similarly at level 4 and 3, there are 10 and 6 unique nodes respectively. Hence, using *horizontal compression* at each level, it is sufficient to work with only the unique nodes at that level.

Moreover, it can be noticed that, as the root of the tree contains only 2^{r+1} RMTs, until level r, all RMTs can not exist in the tree. For example, at level 1, only the RMTs 0 to $2^{r+2}-1$ can be present. However, from level r to $n-r-1$, all 2^{2r+1} RMTs are present in each level. Whereas, from level $n-r$ onwards, for each level $n-\iota$, $1 \leq \iota \leq r$, every RMT that is divisible by $2^{r+1-\iota}$ is present in that level. This is because, reachability tree represents all possible RMT-sequences (or, configurations) of an n-cell CA. To generate all possible RMT sequences, participation of all RMTs is required. Hence, in the tree, the RMTs exist in this way.

Definition 4 *A rule $R : S^{2r+1} \rightarrow S$ is called **balanced** if it contains 2^{2r} number of RMTs corresponding to both the states 0 and 1 of that CA; otherwise it is an **unbalanced** rule.*

For example, for $r = 2$, the CA rule 3063191190 (3rd row of Table 6.10) is balanced, but rule 2966606546 (4th row of Table 6.10) is unbalanced.

Definition 5 *A set of RMTs is called **balanced** if number of RMTs corresponding to state 0 in the set is equal to the number of RMTs corresponding to state 1 in it; otherwise it is **unbalanced**. A node is called **balanced** if the set of all RMTs of the node is balanced; otherwise it is **unbalanced**.*

For example, the RMT set $\{\bigcup_k \mathcal{E}_k^r, 0 \leq k \leq 2^r\}$ are balanced for both the rules 3063191190 and 2966606546 of Table 6.10, where only the first rule is balanced. In Example 6.7, every node of Table 6.12 is balanced.

6.7.2.1 Characterization of Reversible CA

This section explores reachability tree when it represents an n-cell reversible CA. Following theorems report the observations which are similar to [45, 52].

Lemma 6.1 *The reachability tree of a reversible CA of length n $(n \geq 2r+1)$ is complete.*

Proof: Since all the configurations of a reversible CA are reachable, the number of leaves in the reachability tree of an n-cell reversible CA is 2^n (that is, the number of configurations). Therefore, the tree is complete. □

Hence, we can identify whether an n-cell CA is reversible by constructing the reachability tree for it. If the tree is complete, that is, there is no non-reachable edge in the reachability tree, then the CA is reversible. Following theorem further characterizes the reachability tree for a reversible CA.

Theorem 6.1 : *The reachability tree of an n-cell CA $(n \geq 2r+1)$ is complete if and only if each node contains*

1. *exactly 2^ι RMTs, where $1 \leq \iota \leq r$, if the node is at level $n-\iota$; that is, $\mid N_{n-\iota.j} \mid = 2^\iota$, for any j;*
2. *exactly 2^{r+1} RMTs, if the node is at level i where $0 \leq i \leq n-r-1$; that is, $\mid N_{i.j} \mid = 2^{r+1}$, for any j.*

Proof: In a reachability tree, the root $N_{0.0}$ contains 2^{r+1} number of RMTs (see Point 1 of Definition 3). Now, these RMTs are to be distributed into two edges. Let us imagine that, edge $E_{0.0}$ has less than 2^r RMTs whereas, the other edge $E_{0.1}$ has greater than 2^r RMTs. That means, the node $N_{1.0}$ has less than 2^{r+1} RMTs, whereas, node $N_{1.1}$ has greater than 2^{r+1} RMTs. So, the tree is complete for level 1. Let us consider that, the distribution of RMTs is similar for the next levels up to the level $n-r-1$. Hence, up to this level also, the tree is complete having no non-reachable edge/node.

However, at the next level, that is, at level $n-r$, only $\frac{1}{2}$ of the RMTs of each node are valid. So, number of RMTs at some node $N_{n-r.j}$ is less than 2^r. At this level, the tree may not have any non-reachable edge. Now, consider that this situation continues for each level $n-\iota$, where $1 \leq \iota \leq r$. At level $n-\iota$, only $\frac{1}{2^{r+1-\iota}}$ of the possible RMTs are valid (see Point 4 of Definition 3). In the best case, the tree may remain complete until level $n-1$. But at level $n-1$, after discarding the invalid RMTs, there will be one node at the subtree of $N_{1.0}$ which has only one RMT making the tree incomplete at level n. Hence, to be complete, the tree needs to maintain distribution of RMTs as specified in Point 1 and 2.

On the other hand, if the tree fulfils the distribution of RMTs at each node of each level as mentioned in Point 1 and 2 of the theorem, then, there is no non-reachable edge at any level even after crossing out of the RMTs. That is, the CA is complete. Hence, the proof. □

Corollary 6.1 : *In a reachability tree of an n-cell reversible CA ($n \geq 2r + 1$),*

1. *the label $l_{n-\iota.j}$, for any j, contains only $2^{\iota-1}$ RMTs, where $1 \leq \iota \leq r$; that is, $\mid E_{n-\iota.j} \mid = 2^{\iota-1}$;*
2. *each other label $l_{i.j}$ contains exactly 2^r RMTs, where $0 \leq i \leq n - r - 1$; that is, $\mid E_{i.j} \mid = 2^r$.*

Proof: This can be directly followed from Theorem 6.1. □

Corollary 6.2 : *The nodes of the reachability tree of an n-cell ($n \geq 2r + 1$) reversible CA are balanced.*

Proof: If for any level, one node is unbalanced, then at the next level the children nodes will violate the reversibility condition of Theorem 6.1. Hence, the proof follows. □

Corollary 6.3 : *A finite CA of length $n(n \geq 2r + 1)$ with unbalanced rule is irreversible.*

Proof: If the rule is unbalanced, then it has unequal number of RMTs corresponding to each state. At best, the root may be balanced and this may continue until level r. But from level r onwards, all RMTs are generated in each level of the tree. So, there exists one node at level r, which is unbalanced. Hence, by Corollary 6.2, the CA is irreversible. □

However, not all CAs with balanced rules are reversible, as, the corresponding reachability trees need to satisfy the conditions of Theorem 6.1 to be complete. In the next section, we illustrate way to synthesize reversible n-cell CAs using reachability tree.

6.7.2.2 Analysis and Synthesis of Reversible CA

Reachability tree helps to identify whether a CA is reversible for a size $n \geq 2r+1$. If the reachability tree of an n-cell CA is complete, that means, the CA is reversible for that cell length (see Lemma 6.1). Further, the tree needs to satisfy the following two conditions (Theorem 6.1) –

1. Every node of the reachability tree is to be balanced.

2. Each node of level $n - \iota$, $1 \leq \iota \leq r$, is to contain 2^ι RMTs, whereas all other nodes are to be made with 2^{r+1} RMTs.

Furthermore, the CA rule itself has to be balanced (Corollary 6.3). Therefore, for an n-cell CA with a balanced rule, we can construct the (horizontally compressed) reachability tree and check whether the tree violates any of the aforementioned conditions at any level. If it does not violate these conditions, then, the tree will have $n+1$ levels with only a small number of unique nodes at each level. This is because, the nodes (up to level $n-r-1$) are made with partitioning $\bigcup\limits_k \mathcal{E}_k^r$, $0 \leq k \leq 2^{2r}$, into sets of cardinality 2^r, whereas, for the remaining levels, some of the RMTs of these nodes are invalid and discarded. Also, all RMTs of $\mathcal{E}_k^r$, for any k, exist together in a node of level l, $0 \leq l \leq n - r - 1$. Therefore, for any specific rule, only a fixed number of such partitioning is possible. Hence, maximum number of unique nodes at each level is constant. In fact, in the algorithm to test reversibility, we need to store the unique nodes of previous level only to calculate the nodes of current level. So, space complexity of the reversibility testing algorithm is $O(1)$ and time complexity is $O(n)$.

However, it is not required to exhaustively test all possible balanced rules for reversibility. Using Definition 3, Theorem 6.1 and Corollary 6.3, we can further characterize some properties that the RMTs of a reversible rule need to satisfy. These properties are associated with the presence of specific set of RMTs at the first and last r number of levels of the corresponding reachability tree. For example, the RMT set $\{0, \cdots, 2^{r+1}-1\}$ has to be balanced, otherwise the root is unbalanced. Similarly, for the nodes of level 1 to be balanced, the set of RMTs $\{2^{r+1}, \cdots, 2^{r+2}-1\}$ are to be balanced. In this way, for level r, the RMTs $\{2^{2r}, \cdots, 2^{2r+1}-1\}$ are to be balanced. Consequently, for the level $n-r$, the set of even RMTs needs to be balanced, as well as, for any level $n-\iota$, $1 \leq \iota \leq r$, the set of RMTs divisible by $2^{r+1-\iota}$ are to be balanced. Hence, the properties of a reversible CA rule can be summarized as the following:

Property 6.1 *The 2^{2r+1} RMTs of a rule can be divided into r sets $\{0, \cdots, 2^{r+1} - 1\}$, $\{2^{r+1}, \cdots, 2^{r+2}-1\}$, $\cdots$, $\{2^{2r}, \cdots, 2^{2r+1}-1\}$, where each of these sets is to be balanced.*

Property 6.2 *The set of RMTs $\{x \mid x \equiv 0 \pmod{2^{r+1-\iota}}\}$ is to be balanced, where $1 \leq \iota \leq r$.*

Moreover, an n-cell CA is irreversible, if any configuration has more than one predecessor. The following further helps to identify if a CA is irreversible.

Theorem 6.2 : *An n-cell CA is irreversible if anyone of the following conditions is satisfied:*

1. *(for $n \leq 2r$) each of the RMTs of the set $\{0, 2^{i+r-n+1}(\sum\limits_{j} 2^{n-j})\}$ for all i, j $(0 \leq i \leq n-1,\ 1 \leq j \leq n)$ has same next state value.*
2. *(for $n > 2r$) each of the RMTs from the set $\{0, (2^{r+1}-1), (2^{r+2}-1), \cdots, (2^{2r+1}-1), (2^{2r+1-2}), \cdots (2^{2r+1}-2^{r})\}$ have same next state value.*

Proof: For any $n \in \mathbb{N}$, if any of these conditions is satisfied, then the RMT sequences corresponding to the two configurations 0^n and 1^n have same next state value. So, in the CA many to one mapping exists and it is irreversible. This completes the proof. □

For example, if $n = 3$ and $r = 2$, then if the RMTs $\{0, 7, 14, 28\}$ have same next state value, then in the CA, the RMT sequences $\langle 0, 0, 0\rangle$ and $\langle 7, 14, 28\rangle$ (configuration 000 and 111 respectively) are mapped to the same next RMT sequence. Hence, the CA is irreversible. Similarly, for $r = 2$ and $n = 6$, if the RMTs $\{0, 7, 15, 31, 30, 28\}$ have same next state value, that implies the RMT sequences $\langle 0, 0, 0, 0, 0, 0\rangle$ and $\langle 7, 15, 31, 31, 30, 28\rangle$ have same next configuration making the CA irreversible. Observe that, for any $n > 2r$, the set of RMTs to check according to Theorem 6.2 remains the same.

So, we can synthesize CA rules which satisfy Theorem 6.2 and properties 6.1, 6.2. For such a rule, we can test its reversibility using reachability tree to see if any of the conditions of Theorem 6.1 and Corollary 6.3 is violated. If the tree is complete for the CA size n, we can declare the CA as reversible for n. As a concrete example, Table 6.13 of Appendix shows the list of reversible $r = 2$ CAs for any n generated using our scheme; similarly, Table 6.14 refers a collection of reversible $r = 2$ CAs for some specific CA sizes. In the next section we discuss how to find appropriate r-radius n-cell reversible CAs to implement clustering using those CAs.

6.8 Cycle based Clustering using Reversible CAs with $r > 1$

Now we discuss how to identify *appropriate* reversible CAs to cluster these data.

Using the scheme of Section 6.7.2.2, we can generate reversible CAs that can cluster a given set of objects encoded as configurations. However, any arbitrary reversible CA can not produce *effective* clusters. To get such clusters, we need get those collection of CAs which can produce less number of cycles and the

Table 6.13: The reversible CA rules with radius two for an arbitrary n.

252645135	252645195	252645336	252645360	252648975	252652815
252656655	252656880	252691215	252691440	252695055	252698895
252702735	252702960	253678110	254611245	254618925	255594255
255652080	256577295	256577355	259190799	259526415	260509455
260960271	263458575	263458635	264441615	264499440	265482450
267390735	267390795	267390960	267416079	267422991	267448335
267448359	267448560	505290270	505336350	509222490	517136850
517140690	521018910	568251090	570174960	570217200	695036265
755961615	755969295	756011535	756019215	756994590	757927725
757931565	757935405	757939245	764265357	764269197	764273037
764276877	921712188	1010580540	1016918172	1016918172	1511938590
1231776105	1259293455	1259293515	1259293560	1259293680	1263225615
1263225675	1267157835	1267157880	1271089995	1752787305	1768515945
1799965515	1831415085	1921479288	1924428408	2018211960	2018212020
2018212080	2019194970	2021161080	2072267535	2072267595	2076199695
2076199755	2076199920	2218767375	2218767540	2218767600	2222699700
2222699760	2273806215	2275772325	2276755215	2276755275	2276755335
2370538887	2373488007	2463552210	2495001780	2526451350	2542179990
2777130375	2779096485	2783028705	3023877300	3027809415	3027809460
3031741620	3031741680	3035673615	3035673735	3035673780	3035673840
3063191190	3530690418	3530698098	3530701938	3537028050	3537031890
3537035730	3537039570	3537972705	3538948080	3538955760	3538998000
3539005680	3599931030	3724750095	3724792335	3726716205	3773948385
3777826605	3777830445	3785744805	3789630945	3789677025	4027518735
4027518936	4027518960	4027544304	4027551216	4027576335	4027576500
4027576560	4029484845	4030467855	4030525680	4031508660	4031508720
4034007024	4034457840	4035440880	4035776496	4038389940	4038390000
4039315215	4039373040	4040348370	4040356050	4041289185	4042264335
4042264560	4042268400	4042272240	4042275855	4042276080	4042310415
4042310640	4042314480	4042318320	4042321935	4042321959	4042322100
4042322160	1515870810	3284386755	3373255107	1517836920	3278049123
3530694258					

configurations inside any cycle have to be more similar to each other, whereas, the configurations of two different cycles are dissimilar. These two key points have already been addressed for CA with radius one; but previously the number of feature was restricted to one. Now, our target is that a candidate CA rule has to be designed such that it is not dependent only on a single feature. Obviously, any arbitrary reversible CA may not satisfy these criteria. So, our requirement is to choose only those CAs as auxiliary CAs for clustering which satisfy these conditions.

To incorporate the condition, that is, the CA is not dependent only on a

Table 6.14: The reversible $r = 2$ CAs for some specific n. In column II of this table j is a natural number.

Rule	Reversible for n other than
859032780	$\{3j+1, 3j+2, 3j+4, 3j+5\}$
869020620	$\{2j+3\}$
869059635	$\{7j+1, 7j+5, 7j+6, 7j+8\}$
2573624985	$\{6j+1, 6j+4, 6j+3, 6j+5, 6j+7\}$
1010615235	$\{7j+3, 7j+5, 7j+6\}$
1019428035	$\{3j+2, 6j+2, 3j+5\}$
1019462460	$\{4j+2, 4j+3, 4j+6\}$
1010615235	$\{7j+3, 7j+5, 7j+6\}$
1431677610	$\{4j+2, 4j+1, 4j+3, 4j+6, 4j+5, 4j+7\}$
1437226410	$\{3j+2, 3j+5, 3j+4\}$
1437248085	$\{15j+2, 15j+1, 15j+9, 15j+10, 15j+13, 15j+12, 15j+14, 15j+17, 15j+16\}$
1515890085	$\{6j+4, 6j+3, 6j+5\}$
3275539260	$\{3j+2, 6j+2, 3j+5\}$
1520786085	$\{7j+3, 7j+6, 7j+5\}$
1520805210	$\{7j+4, 7j+6, 7j+5\}$
1718000025	$\{15j+1, 15j+2, 15j+10, 15j+9, 15j+13, 15j+12, 15j+14, 15j+16, 15j+17\}$
1721329305	$\{7j+5, 7j+6, 7j+8\}$
1721342310	$\{6j+1, 6j+4, 6j+3, 6j+5, 6j+7\}$
1768527510	$\{7j+5, 7j+4, 7j+6\}$
1771465110	$\{4j+3, 4j+2, 8j+3, 8j+2, 4j+7, 4j+6\}$
1771476585	$\{5j+3, 5j+2, 5j+4, 5j+8, 5j+7\}$
2779077210	$\{6j+4, 6j+3, 6j+5\}$
2857719210	$\{15j+2, 15j+1, 15j+9, 15j+10, 15j+13, 15j+12, 15j+14, 15j+17, 15j+16\}$
2857740885	$\{3j+2, 3j+5, 3j+4, 6j+2\}$
2863289685	$\{4j+2, 4j+1, 4j+3, 4j+6, 4j+5, 4j+7\}$
1019428035	$\{3j+2, 6j+2, 3j+5\}$
1019462460	$\{4j+2, 4j+3, 4j+6\}$
2573637990	$\{7j+6, 7j+5, 7j+8\}$
2576967270	$\{15j+1, 15j+2, 15j+10, 15j+9, 15j+13, 15j+12, 15j+14, 15j+16, 15j+17\}$
2774162085	$\{7j+5, 7j+4, 7j+6\}$
2774181210	$\{7j+5, 7j+3, 7j+6\}$
2523490710	$\{5j+3, 5j+2, 5j+4, 5j+8, 5j+7\}$
2523502185	$\{4j+3, 4j+2, 8j+3, 8j+2, 4j+7, 4j+6\}$
2526439785	$\{7j+5, 7j+4, 7j+6\}$
3275504835	$\{4j+2, 4j+3, 4j+6\}$
3284352060	$\{7j+3, 7j+5, 7j+6\}$
3425907660	$\{7j+1, 7j+5, 7j+6, 7j+8\}$
3425946675	$\{2j+3\}$
3435934515	$\{3j+1, 3j+2, 3j+4, 3j+5\}$

single feature, we choose the *neighborhood* vector of the CAs in such a way that, the *local rules* are always dependent on at least two features. So, the global transition function works considering all features collectively. However, not all r-radius CAs are actually dependent on r neighbors at each side. This is because, in general any function of k arguments can be extended to another equivalent function of $k+1$ arguments. As an example, consider, $f(x_1, x_2 \cdots, x_k)$ be a function such that $f : \{0,1\}^k \longrightarrow \{0,1\}$. We can always design another function $f'(x_1, x_2 \cdots, x_k, x_{k+1})$ where $f' : \{0,1\}^{k+1} \longrightarrow \{0,1\}$ such that, f' is equivalent to f:

$$f'(x_1, x_2 \cdots, x_k, x_{k+1}) = \begin{cases} 0 & \text{if } f(x_1, x_2 \cdots, x_k) = 0 \\ 1 & \text{if } f(x_1, x_2 \cdots, x_k) = 1 \end{cases}$$

That means, if we consider x_{k+1} at the last position, for both the tuples $\langle x_1, x_2 \cdots, x_k, 0\rangle$ and $\langle x_1, x_2 \cdots, x_k, 1\rangle$, the function f' returns the same value which is equal to $f(x_1, x_2 \cdots, x_k)$. Here, x_{k+1} is a *redundant* argument. Obviously, by varying the position of this *redundant* argument x_{k+1}, $k+1$ number of such functions can be generated which are equivalent to f.

Nevertheless, recall that, the local rule of a CA is also a function $R : \{0,1\}^{2r+1} \longrightarrow \{0,1\}$. Therefore, among the rule-space of r-radius n-cells CAs, there are many CA rules which are equivalent to functions of lesser number of arguments (neighbors). For these CAs, depending on the neighborhood index of the *redundant* neighbor k ($-r \le k \le r$) in the neighborhood vector, the RMTs $\mathtt{x_{-r}x_{-r+1}\cdots x_{k-1}0x_{k+1}\cdots x_r}$ and $\mathtt{x_{-r}x_{-r+1}\cdots x_{k-1}1x_{k+1}\cdots x_r}$ have same next state value with respect to R, where $\mathtt{x_i} \in \{0,1\}$. Now, according to Definition 1, these RMTs are *k-equivalent* or part of the set $\mathcal{E}_i^k$, $0 \le i \le 2^{2r}-1$. Hence, by observing these equivalent sets, we can identify whether a neighbor index k is redundant in the local rule of the CA as mentioned in the following property:

Property 6.3 *For any* $\mathtt{x}, \mathtt{y} \in \mathcal{E}_i^k$, $0 \le i \le 2^{2r}-1$, *if* $R[\mathtt{x}] = R[\mathtt{y}]$, *then, the corresponding CA rule is independent of the* k^{th} *neighbor in the neighborhood vector, where* $k \in \{-r, -r+1, \cdots, r-1, r\}$.

The choice of r has implications on the performance of our clustering technique. If r is too large, then although more number of reversible rules can be identified, but the complexity and running time of our technique increases due to increased size of the candidate CA. On the other hand, if $r = 1$, we would not be able to deal with categorical attributes. However, our scheme can be modified to include dependency of more number of features by increasing the radius as larger than r, the number of bits required to encode any feature. Nevertheless, our requirement is the corresponding r-radius candidate CA rules are dependent on at least two features, each encoded using r bits. To ensure that we make the following restriction: We select only those r-radius CA rules which can not be represented as equivalent to functions of less than r-radius. So, all rules which are dependent on at maximum $r-1$ radius, that is, maximum $r-1$ neighbors at any side, are discarded from the rule space. That means, in the candidate CAs, none of the neighborhood indices k, $-r+1 \le k \le r-1$, can be redundant, as well as, the CAs are also dependent on the r^{th} or $-r^{th}$ neighbor. Hence, we can formalize our first strategy for rule selection as the following:

Table 6.15: CA rule-space satisfying Condition 6 for $r = 2$ with different CA sizes (n)

n	#ReversibleRules	#Rules without Condition 6	#Rules without Condition 7	#Rules in $\mathscr{R}_n^2$
8	210	32	40	140
10	200	34	40	128
12	204	34	40	132
14	204	32	40	134
16	210	32	40	138
19	194	32	40	124
20	204	32	40	134

Condition 6 *Choose only those n-cell reversible CAs, for which the following two conditions are satisfied –*

1. *For each k, $-r + 1 \leq k \leq r - 1$, there exists i, $0 \leq i \leq 2^{2r} - 1$, such that $R[\mathtt{x}] \neq R[\mathtt{y}]$, where $\mathtt{x}, \mathtt{y} \in \mathcal{E}_i^k$. That is, for each k, the CA is dependent on its k^{th} neighbor, where $-r + 1 \leq k \leq r - 1$.*
2. *There exists j, $0 \leq j \leq 2^{2r} - 1$, such that, $R[\mathtt{z}] \neq R[\mathtt{w}]$, where $\mathtt{z}, \mathtt{w} \in \mathcal{E}_j^r$ or $\mathtt{z}, \mathtt{w} \in \mathcal{E}_j^{-r}$. That is, the CA is also dependent on its r^{th} or $-r^{th}$ neighbor.*

For example, consider a hypothetical case, where $r = 1$ and $n = 4$. There are 10 reversible *Elementary* CAs (ECAs) for 4-cells – 00110011(51), 00111100(60), 01011010(90), 01100110(102), 01101001(105), 10010110(150), 10011001(153), 10100101(165), 11000011(195) and 11001100(204). However, among them, next state functions of rules 51 and 204 are independent of the left and right neighbor, whereas, rules 60, 102, 165 and 195 are independent of neighbors at one of the directions. Hence, we discard these six rules from the rule-space. Similarly, if we take $r = 2$, then among the CA rules of Table 6.13 and Table 6.14, there are only 2 rules which do not satisfy Point 2 of Condition 6 whereas, the number of rules that violates Point 1 of Condition 6 varies with n. Table 6.15 gives the number of $r = 2$ CAs satisfying this strategy for some specific CA sizes, $n \in \{8, 10, 12, 14, 16, 19, 20\}$.

Nonetheless, any arbitrary CA satisfying Condition 6 may not maintain less intra-cluster and high inter-cluster distance with limited number of cycles. For this, we further probe into the RMTs of the CAs as discussed next.

In Section 6.7.1 we have discussed that self-replicating RMTs play a major role in preserving the number of cycles and the type of configurations placed inside the same cycle. This is because, presence of self-replicating RMTs in a configuration ensures that there is less number of changes in the state for any two consecutive configuration. So, if we consider *hamming distance* as our metric to measure distance, then this implies configurations with lesser hamming distances are placed in the same cycle maintaining less intra-cluster distance. However, if all RMTs of the configuration are self-replicating, then the configuration is a *fixed-point attractor* resulting a cycle of length 1. So, if all RMTs of a rule are *self replicating*, then, for this CA, each of the 2^n configurations are fixed-points, that is, the number of cycles is 2^n which is not desirable for clustering. Hence, the count of self-replicating RMTs for any configuration has to be restrained – we can not take a CA as candidate where all RMTs are self-replicating. Moreover, if a CA has no self-replicating RMTs, then also, it is not *effective* as clustering tool. Because, firstly, the CA is dependent only on its 0^{th} neighbor, which is not allowed as per Condition 6. Secondly, for any configuration of this CA, the next configuration is its complement (flipping 0 to 1 and vice versa); so, the hamming distance between any two configuration is maximum, that is, n. Further, such a CA also has exponential number of cycles, i.e., it has 2^{n-1} cycles where each cycle is of length 2. Hence, as a trade-off, we take the following strategy to choose the appropriate CAs:

Condition 7 *Select those CAs as candidates which have* moderate *number of self-replicating RMTs.*

Here, by the term *moderate* number of self-replicating RMTs, we mean to take those CAs which are not only satisfying Condition 6, but are also more dependent on their neighbors. That is, we discard those CAs which are almost $r-1$ radius CAs. One can observe that, the CAs with no self-replicating RMTs and all RMTs as self-replicating can not satisfy Condition 6. Therefore, to filter out other rules which have possibility to generate ineffective clustering, the *acceptable range* of self-replicating RMTs can be set according to the given set of data. For example, while working with CAs with radius $r = 2$, we set the acceptable range as [18.75%, 87.5%] of the number of possible RMTs. We discard those rules which have 6.25%, 12.5%, 18.75% self replicating RMTs – these rules are less dependent on their neighbors. Same is for the rules with 87.5%, 93.75% self replicating RMTs. These rules, which are not covered in Condition 6, are screened out using Condition 7. Table 6.15 shows the number of rules for $r = 2$ CAs which violates this condition, From this table, we can see that the total number of *effective* reversible radius 2 CAs that can satisfy both Condition 6 and 7 are between 125 to 150.

Therefore, using these two conditions, we can ensure that the CAs selected as candidates do not have exponential number of cycles, maintain lesser intra-cluster distance and invariably keep larger inter-cluster distance. Let us name, the list of *candidate* r-radius n-cell CA rules for clustering as $\mathscr{R}^r_n$. So, $\bigcup \mathscr{R}^r_n = \mathscr{R}^r$, where $\mathscr{R}^r$ is the set of candidate r-radius CAs for all $n \in \mathbb{N}$. We can now proceed to our cycle based clustering technique in the next section.

Table 6.16: Description of real datasets used for our proposed clustering algorithm

Name	# of p	# of p_{q_n}	# of p_{q_l}	# of target objects	CA size (n)	# of objects with missing terms
Iris	4	4	0	150	8	0
User Knowledge Modeling	5	5	0	403	10	0
BuddyMove	6	6	0	249	12	0
Seed	7	7	0	210	14	0
StoneFlakes	8	8	0	79	16	6
Heart failure clinical records	12	12	0	299	19	0
Travel Reviews	10	10	0	980	20	0

6.9 Experimental Results using Auxiliary CAs with $r > 1$

This section aims to analyze the performance of our Algorithm 9 in *effectively* distributing objects of some benchmark datasets among m clusters, where m is the desired number of clusters. We have tested our algorithm considering CAs of radius (r) *two*. That is, each continuous attribute is encoded into 2 bits and each categorical attribute can have only two values. Further, the candidate CAs with $r = 2$ will be dependent on at least two features. Already, it has been shown that CAs with $r = 1$ shows dependency on at most one feature. So, $r = 2$ is taken as the choice for implementing our clustering algorithm where number of candidate CAs is reasonably good for random selections of auxiliary CAs.

Here we use *seven* datasets `Iris`, `User Knowledge Modeling`, `BuddyMove`, `Seed`, `StoneFlakes`, `Heart failure clinical records`, `Travel Reviews`. These are collected from `http://archive.ics.uci.edu/ml/index.php` where each of them have only *quantitative* attributes. Previously, we also use all six same datasets other than `Travel Reviews`. Table 6.16 introduces the meaningful information for each dataset.

To analyze the performance of our algorithm, two important factors are considered - (i) whether the data objects can be distributed to the desired number of clusters and (ii) the quality of the resultant clusters. Similar to the case of non uniform ECAs, the former one is computed by Algorithm 9 whereas the later one is measured by some popular benchmark validation metrics [70, 462] – *connectivity* [459], *silhouette score* [458] and *Dunn index* [456]. We have observed that, for each dataset, out of 1000 runs, at least 993 times the target objects are distributed among the *desired* number of *quality* clusters, which is taken as 2. Table 6.17 presents the performance of our scheme based on the validation metrics.

Table 6.17: Performance of our clustering algorithm on the available datasets (see Table 6.5)

Dataset	m	Optimal Score	level	CA

Iris	2	Silhouette=0.6867351,
		Dunn=0.3389087,
		Connectivity=0.0000
			5	CA_0: 3538955760
				CA_1: 1924428408
				CA_2: 3530701938
				CA_3: 2222699760
				CA_4: 2276755275

User Knowledge Modeling	2	Silhouette=0.1928821	6	CA_0: 2542179990 CA_1: 1520805210 CA_2: 2523490710 CA_3: 2573637990 CA_4: 3023877300 CA_5: 3530698098

		Dunn=0.1744164,
		Connectivity=3.178968
			10	CA_0: 570174960
				CA_1: 267416079
				CA_2: 3530690418
				CA_3: 3023877300
				CA_4: 2495001780
				CA_5: 2573637990
				CA_6: 764273037
				CA_7: 4034007024
				CA_8: 4035776496
				CA_9: 1016918172

Buddymove	2	Silhouette=0.3015836	5	CA_0: 252691440 CA_1: 3031741680 CA_2: 1771465110 CA_3: 3537972705 CA_4: 1267157835

		Dunn=0.1221073,
		Connectivity=3.825794
			5	CA_0: 757931565
				CA_1: 1259293680
				CA_2: 3035673780
				CA_3: 2072267535
				CA_4: 4034007024

Seed	2	Silhouette=0.5263088	7	CA_0: 568251090 CA_1: 4041289185 CA_2: 3599931030 CA_3: 4040348370 CA_4: 1752787305 CA_5: 570174960 CA_6: 4042321959

		Dunn=0.07032726,
		Connectivity=3.603571
			7	CA_0: 3063191190
				CA_1: 4040348370
				CA_2: 3538955760
				CA_3: 1016918172
				CA_4: 4042268400
				CA_5: 3035673780
				CA_6: 764276877

	2	Silhouette =0.3791465	8	CA_0: 1511938590 CA_1: 4042272240 CA_2: 1520805210 CA_3: 1010615235 CA_4: 2523502185 CA_5: 4031508720 CA_6: 4027551216 CA_7: 4035776496

Continued on next page

Table 6.17 – *Continued from previous page*

Dataset	m	Optimal Score	level	CA
StoneFlakes		Dunn=3.61627	6	CA_0: 3284352060 CA_1: 517136850 CA_2: 259190799 CA_3: 3530690418 CA_4: 3035673780 CA_5: 2542179990
		Connectivity=0.1665206	4	CA_0: 2783028705 CA_1: 2523502185 CA_2: 2275772325 CA_3: 4030467855
Heart failure clinical records	2	Silhouette=0.7147869, Dunn=0.01213804	5	CA_0: 4031508660 CA_1: 2072267595 CA_2: 4027551216 CA_3: 570217200 CA_4: 4027544304
		Connectivity=4.428968	4	CA_0: 757931565 CA_1: 509222490 CA_2: 1515890085 CA_3: 4027544304
Travel Reviews	2	Silhouette=0.3406666, Dunn=0.129968	6	CA_0: 263458635 CA_1: 263458635 CA_2: 4027551216 CA_3: 695036265 CA_4: 3777826605 CA_5: 764269197
		Connectivity=3.295635	9	CA_0: 1271089995 CA_1: 2463552210 CA_2: 267416079 CA_3: 3027809415 CA_4: 757931565 CA_5: 4031508660 CA_6: 4042272240 CA_7: 3537035730 CA_8: 570217200

Here, columns III refers the optimal scores of the validation indices - *connectivity*, *silhouette score* and *Dunn index*. The next two columns (IV and V) represent the number of levels used to get the optimal results for each dataset and the corresponding ordered list of auxiliary CAs where any CA$_i$ is exclusively used for a particular level i. For example, in case of `Iris` dataset, the optimal scores for all three validation indices are achieved while clustering is completed in level 5. Moreover, same set of auxiliary CAs are used for all three cases. However, for other datasets like `Travel Reviews, Heart failure clinical records` the same set of auxiliary CAs give the optimal scores for silhouette score and dunn index; whereas for `User Knowledge Modeling, BuddyMove, Seed` datasets the best performance is observed for both connectivity and dunn index using the same set of auxiliary CAs.

To verify the performance of our algorithm, we compare the optimal scores of the validation indices already obtained (see Table 6.17) with some benchmark

Table 6.18: Comparison of clustering techniques based on internal validation indices for each of the available datasets of Table 6.16

dataset1	Algorithm	Connectivity		Dunn Index		Silhouette Score	
		Score	m	Score	m	Score	m
Iris	Hierarchical	0.0000	2	0.3389	2	0.6867	2
	K-means	6.1536	2	0.0765	2	0.6810	2
	DIANA	6.1536	2	0.0765	2	0.6810	2
	PAM	3.9623	2	0.0811	2	0.6858	2
	SOTA	11.5016	2	0.0349	2	0.6569	2
	CA-based Clustering	**0.0000**	**2**	**0.3389**	**2**	**0.6867**	**2**
User Knowledge Modeling	Hierarchical	3.0540	2	0.1970	2	0.2589	2
	K-means	50.3321	2	0.0943	2	0.2063	2
	DIANA	68.7329	2	0.0297	2	0.2066	2
	PAM	102.8329	2	0.0537	2	0.1934	2
	SOTA	64.1060	2	0.0321	2	0.2042	2
	CA-based Clustering	**3.1789**	**2**	**0.1744**	**2**	**0.1928**	**2**
Buddymove	Hierarchical	2.9290	2	0.3146	2	0.4764	2
	K-means	35.2032	2	0.0485	2	0.3079	2
	DIANA	34.3694	2	0.0658	2	0.3020	2
	PAM	38.3802	2	0.0485	2	0.3055	2
	SOTA	44.4111	2	0.0518	2	0.3134	2
	CA-based Clustering	**3.8257**	**2**	**0.1221**	**2**	**0.3015**	**2**
Seed	Hierarchical	8.7861	2	0.1065	6	0.5248	2
	K-means	21.3698	2	0.0548	2	0.5229	2
	DIANA	19.1714	2	0.0544	2	0.5218	2
	PAM	20.6762	2	0.0404	2	0.5175	2
	SOTA	16.0179	2	0.0361	2	0.5049	2
	CA-based Clustering	**3.6035**	**2**	**0.0703**	**2**	**0.5263**	**2**
StoneFlakes	Hierarchical	2.9290	2	0.3176	6	0.2937	2
	K-means	4.9488	2	0.1735	2	0.4875	2
	DIANA	4.9488	2	0.1735	2	0.4875	2
	PAM	7.9762	2	0.1693	2	0.4827	2
	SOTA	4.9488	2	0.1735	2	0.4875	2
	CA-based Clustering	**3.6162**	**2**	**0.1665**	**2**	**0.3791**	**2**
Heart failure clinical records	Hierarchical	5.7536	2	0.1506	2	0.7862	2
	K-means	1.8635	2	0.0079	2	0.5829	2
	DIANA	1.8635	2	0.0079	2	0.5829	2
	PAM	3.5020	2	0.0036	2	0.4630	2
	SOTA	5.0972	2	0.0035	2	0.5302	2
	CA-based Clustering	**4.4289**	**2**	**0.0121**	**2**	**0.7147**	**2**
Travel Reviews	Hierarchical	2.9290	2	0.3175	2	0.4107	2
	K-means	135.5111	2	0.0408	2	0.3009	2
	DIANA	120.0984	2	0.0691	2	0.2953	2
	PAM	151.6171	2	0.0689	2	0.2838	2
	SOTA	151.5698	2	0.0648	2	0.2949	2
	CA-based Clustering	**3.2956**	**2**	**0.1299**	**2**	**0.3406**	**2**

clustering algorithms. For this, the same datasets are also tested by *K-means* (centroid based clustering) [71], *hierarchical* (agglomerative hierarchical clustering) [71], *DIANA* (divisive hierarchical clustering) [71], *PAM (Partitioning around medoids)* (centroid based clustering) [71] and *SOTA (Self-organizing tree algorithm)* (unsupervised network with a divisive hierarchical clustering) [71] – a set of popular clustering algorithms. For each algorithm, the number of desired clusters is taken as *two* and the scores of the concerned validation indices are computed. Here also, the implementation of these existing algorithms are taken from [71]. Table 6.18 reports result of this comparison. From the experiments, it can be noted that, among the existing algorithms, for the StoneFlakes dataset, optimal silhouette score is found by `K-means, DIANA, SOTA`, whereas, the best connectivity score for Heart failure clinical records dataset is achieved by using `K-means` and `DIANA` algorithms. However, by overall performance on all datasets, the *hierarchical* algorithm forms clusters most *effectively*. Further, Table 6.18 also shows that, the validation indices scores obtained by our algorithm can compete with the best scores obtained by the bench marks algorithms. And, sometimes, our algorithm even performs better than the best existing result, see as reference, the connectivity and silhouette scores for `Seed` dataset using Algorithm 9 are optimal scores which are better than that of the existing best scores achieved by *hierarchical* algorithm. Hence, our clustering algorithm can successfully compete with any of the best clustering algorithms existing today.

6.10 Summary

This work has reported a CA based clustering technique where we have used the bijective functions of reversible CAs to distribute the target objects among clusters. Here, we use non-uniform reversible ECAs and uniform reversible CA where $r = 2$. In a reversible CA, the configuration space is divided into a number of cyclic spaces where the configurations inside a cyclic space are reachable from each other. This reachability is exploited as our closeness metric to distribute the configurations of each cyclic space as a distinct cluster. In this way, reversible CA can aptly work as a natural clustering tool, where the target objects are encoded as configurations of a CA. However, any arbitrary reversible CA may not work as effective clustering technique for a given dataset. So, we have identified some properties which are to be satisfied by a CA to be a good candidate for clustering. Such a CA can effectively cluster the target objects ensuring less intra-cluster and more inter-cluster distance. Nevertheless, sometimes the application may need a specific number of clusters whose corresponding bijective function may not be represented as a CA. To deal with this, we have developed a level-wise iterative clustering algorithm, where instead of a single reversible CA, we choose one reversible CA per level where each cluster of one level is treated as an object in the next level. We have tested our algorithm on some standard datasets based on three well-known benchmark validation indices. It can be observed that, performance of our CA based iterative clustering technique is at par with the best result obtained by the existing algorithm. Our algorithm has shown that even if after reordering of features, the CAs are capable clustering the same dataset in the same manner, i.e., same set of objects form a cluster if the order of features are changed However, some more analysis can be done to characterize the candidate CAs based on the nature of real data. In future, this work can be extended addressing this aspect of raw datasets.

CHAPTER 7

Conclusion and Future Work

This chapter outlines a summary on the major contributions of this research that have been provided in the different chapters of the dissertation. It also summarizes some issues that can be addressed in furtherance of the research in cellular automata.

7.1 Key Contributions

This dissertation has studied the cyclic behaviour of finite 1-dimensional three neighborhood null boundary binary cellular automata. Specifically, we have focused on four different aspects namely the cycle structure of a cellular automaton, reducibility of cellular automata, large length cycle cellular automata and application of reversible CAs in clustering.

An archive of cellular automata has been represented in Chapter 2 starting from its birth to the potentiality of cellular automata in machine learning. This chapter also covers a brief survey on reversibility of CA with an emphasis on finite (non-uniform) reversible cellular automata.

In Chapter 3, we have framed the detection of cycles as a sub-problem of reachability problem, and a few conditions have been inferred for detecting efficiently if a given configuration is acyclic. For evaluating the cycle structures, we have devised some efficient strategies that reduce the running time and space complexity. Based on empirical results, we have shown that our first strategy (Algorithm 1, page 55) performs well for a wide range of irreversible CAs, but can be computationally demanding for reversible CAs. This motivates us to introduce the notion of partial cycle structure and blocks in CA and to detect existence of isomorphism within a block of a CA. Our second strategy (Algorithm 2, page 81) applies the aforementioned concepts to evaluate the cycle structure for a reversible CA. Although the worst case complexity is still exponential, Algorithm 2 can evaluate efficiently the CSs for many reversible CAs in practice.

The concept of reducibility in CAs has been introduced in Chapter 4. We have classified the finite reversible CA family into reducible and irreducible cellular automata. Reducible cellular automata are again classified into strictly reducible CAs and weakly reducible CAs. Whether a given reversible CA is reducible or irreducible can be determined by the decomposition property of CS. In Chapter 4, a collection of lemmas, corollary and propositions have been reported to study and explain the reducibility aspects.

In Chapter 5 we have dealt with the problem of finding finite CA with (near) optimal size which can produce a cycle of a given length l - the reverse problem of figuring out the cycle structure of an n-cell CA. In this chapter, first we have introduced the parameters that determine the degree of dependence of the CA rules on their neighboring cells. We have also observed that more dependency on neighbors for the rules makes the CA capable of generating cycle length even larger than 2^{n-1}, where n is the size of the CA. This method of generating a CA based on the dependency of neighbors of the rules, and checking its cycle structure is, however, not scalable for large n. To overcome this issue, we have concatenated small sized CAs (for example, an $n_1 << n$-cell CA that can produce a cycle of length at least 2^{n_1-1}) and applied an evolutionary strategy using some divergent of genetic algorithm. The notable change in our strategy with respect to the genetic algorithm is that the creation of an offspring is influenced by multiple parents, and the offspring owns the significant features of the selected parents. Our strategy is able to determine the near optimal solution for cycle of length as large as 2^{1000} in less than a minute.

As an application of n-cell CAs which can produce cycles of lengths at least 2^{n-1}, we have reported a design of Pseudo Random Number Generators. We have verified the randomness quality of these CAs as PRNGs by using Diehard battery of tests, battery rabbit of TestU01 library and NIST statistical test suite. It is observed that the performance of our proposed PRNGs are better than many of the well-known PRNGs existing today. Based on the empirical results, in this chapter it has also been shown that CAs of size n can act as prime number generators for an arbitrary prime number in the range $[2^{n-1}, 2^n - 1]$.

In the last contributing chapter, Chapter 6, we have introduced the cycle based clustering technique by using reversible cellular automata (CAs). Based on reachability, this research aims to find out the properties in CAs which make the clustering effective. To this effect, firstly we have figured out the CA rules that contribute in maintaining minimum intra-cluster distance. Our CA is then designed with such rules to ensure limited number of cycles in the configuration-space. An iterative strategy has been introduced which works level by level; at any level it generates the desired number of clusters on demand by merging the objects of closely reachable clusters of the previous level using a unique auxiliary CA. We have tested our algorithm on some standard datasets based on three well-known benchmark validation indices. It has been observed that performance of our CA based iterative clustering technique is at par with the best result obtained by the existing algorithm. We have also shown that the arrangement of clusters in our scheme remains fixed even if the ordering of the features of a given set of objects is modified.

7.2 Future Extension

Following are some interesting problems for extending our work.

- In Chapter 3, we reported Algorithm 2 for evaluating the cycle structures of reversible cellular automata; for a collection of reversible cellular automata, Algorithm 2 performs efficiently. Devising an algorithm with better time complexity for evaluating the CS of an arbitrary reversible CA can be an interesting future work.

- The theories developed in Chapter 4 contribute to identification of reducible and irreducible cellular automata. These theories work for a wide range of reversible cellular automata, but may not be effective to decide whether an arbitrary reversible cellular automata is reducible or irreducible. Figuring out a generic scheme in this aspect can be an interesting research to pursue. In addition, this research can be extended for irreversible cellular automata.

- We showed a scheme for designing a near optimal cellular automaton for a given cycle length l for ECAs. Extending this work for any d-state r-radius can be a challenging problem for further research.

- This research introduced the notion of CAs with large length prime cycles such that n-cell non-uniform elementary cellular automata can generate cycles of prime lengths in between 2^{n-1} to $2^n - 1$. We have figured out rules that are effective for designing CAs for this purpose; nevertheless, there is no deterministic approach for designing CAs as prime generators. Development of a more formalized theory for prime number generation using CAs can be considered as an interesting problem. A few more interesting questions in this area are: (i) can we find all possible prime numbers between 2^{n-1} to $2^n - 1$ using n-cell uniform cellular automata by extending either radius or number of states? (ii) can we design an $(n+1)$-cell large length prime generator cellular automaton using an n-cell large length prime generator CA?

- This dissertation showed the capability of reversible cellular automata as an efficient clustering tool. Though our cycle based clustering technique performs well for real datasets, the following areas are yet to be explored.
    - Minimizing the complexity for identifying auxiliary clusters.
    - Characterization of auxiliary CAs based on the dataset to be clustered.
    - Use of CAs for clustering on large dimensional data.

  Study of all these aspects can be crucial for establishing the efficacy of cellular automata as a generic clustering tool.

- Finally, the study of cyclic behaviour of cellular automata can be extended for more general setting of CAs such as d-state CAs, periodic boundary CAs, CAs with increased neighborhood.

Author's Statement

1. Sukanya Mukherjee, Sumit Adak, Kamalika Bhattacharjee, and Sukanta Das. Non-uniform nonlinear cellular automata with large cycles and their application in pseudo-random number generation. *International Journal of Modern Physics C*, 32(07): 2150091, 2021.

2. Sukanya Mukherjee, Kamalika Bhattacharjee, and Sukanta Das. Reversible Cellular Automata: A Natural Clustering Technique. *Journal of Cellular Automata*, 16(1-2): 1–38, 2021.

3. Sukanya Mukherjee, Kamalika Bhattacharjee, and Sukanta Das. Clustering Using Cyclic Spaces of Reversible Cellular Automata. *Complex Systems*, 30(2): 205–237, 2021.

4. Sumit Adak, Sukanya Mukherjee, and Sukanta Das. Reachability problem in non-uniform cellular automata. *Information Sciences*, 543: 72–84, 2021.

5. Sukanya Mukherjee, Kamalika Bhattacharjee, and Sukanta Das. Cycle Based Clustering Using Reversible Cellular Automata. *In Proceedings of Twenty-sixth International Workshop on Cellular Automata and Discrete Complex Systems, AUTOMATA*, pages 29–42. Springer International Publishing, 2020.

6. Sumit Adak, Sukanya Mukherjee, and Sukanta Das. Do there exist non-linear maximal length cellular automata? a study. *In Proceedings of Thirteenth International Conference on Cellular Automata for Research and Industry, ACRI*, pages 289–297. Springer International Publishing, 2018.

7. Sukanya Mukherjee, and M. Nazma Naskar. Cycle structure of non-uniform cellular automata. *In Proceedings of Fifth International Conference on Emerging Applications of Information Technology (EAIT).* pages 1–4. IEEE, 2018.

Bibliography

[1] H. B. Hollinger and M. Zenzen, *The Nature of Irreversibility: A Study of Its Dynamics and Physical Origins.* Springer Science & Business Media, 2012, vol. 28.

[2] D. Helbing, "Traffic and Related Self-Driven Many-Particle Systems," *Reviews of Modern Physics*, vol. 73, no. 4, p. 1067, 2001.

[3] R. Livi, G. Martinez-Mekler, and S. Ruffo, "Periodic Orbits and Long Transients in Coupled Map Lattices," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1-3, pp. 452–460, 1990.

[4] M. Creutz, "Deterministic Ising Dynamics," *Annals of Physics*, vol. 167, no. 1, pp. 62–72, 1986.

[5] C. A. Pérez-Delgado, M. Mosca, P. Cappellaro, and D. G. Cory, "Single Spin Measurement Using Cellular Automata Techniques," *Physical Review Letters*, vol. 97, no. 10, p. 100501, 2006.

[6] J. Greenberg, B. Hassard, and S. Hastings, "Pattern Formation and Periodic Structures in Systems Modeled by Reaction-Diffusion Equations," *Bulletin of the American Mathematical Society*, vol. 84, no. 6, pp. 1296–1327, 1978.

[7] J. M. Greenberg, C. Greene, and S. Hastings, "A Combinatorial Problem Arising in the Study of Reaction-Diffusion Equations," *SIAM Journal of Algebra and Discrete Mathematics*, vol. 1, pp. 34–42, 1980.

[8] J. R. Weimar, "Cellular Automata for Reaction-Diffusion Systems," *Parallel Computing*, vol. 23, no. 11, pp. 1699–1715, 1997.

[9] R. Monaco, *Discrete Kinetic Theory, Lattice Gas Dynamics and Foundations of Hydrodynamics*. World Scientific, 1989.

[10] U. Frisch, B. Hasslacher, and Y. Pomeau, "Lattice-Gas Automata for the Navier-Stokes Equation," *Physical Review Letters*, vol. 56, no. 14, p. 1505, 1986.

[11] G. D. et al., *Lattice Gas Methods for Partial Differential equations*. New York: Addison-Wesley, 1990.

[12] G. Doolen, *Lattice Gas Methods for PDE's, Theory, Applications and Hardware*. North-Holland, 1991.

[13] H. A. Gutowitz, "Maps of Recent Cellular Automata and Lattice Gas Automata Literature," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1-3, pp. 477–479, 1990.

[14] S. Wolfram, "Cellular Automaton Fluids 1: Basic Theory," *Journal of Statistical Physics*, vol. 45, no. 3, pp. 471–526, 1986.

[15] L. N. de Castro, "Fundamentals of Natural Computing: An Overview," *Physics of Life Reviews*, vol. 4, no. 1, pp. 1–36, 2007.

[16] L. Kari and G. Rozenberg, "The Many Facets of Natural Computing," *Communications of the ACM*, vol. 51, no. 10, pp. 72–83, 2008.

[17] F. Celada and P. E. Seiden, "A Computer Model of Cellular Interactions in the Immune System," *Immunology Today*, vol. 13, no. 2, pp. 56–62, 1992.

[18] D. Stauffer and G. Weisbuch, "High-Dimensional Simulation of the Shape-Space Model for the Immune System," *Physica A: Statistical Mechanics and its Applications*, vol. 180, no. 1-2, pp. 42–52, 1992.

[19] D. G. Mallet and L. G. De Pillis, "A Cellular Automata Model of Tumor–Immune System Interactions," *Journal of Theoretical Biology*, vol. 239, no. 3, pp. 334–350, 2006.

[20] R. M. Zorzenon Dos Santos, "Using Cellular Automata to Learn About the Immune System," *International Journal of Modern Physics C*, vol. 9, no. 06, pp. 793–799, 1998.

[21] L. Schulman and P. Seiden, "Statistical Mechanics of a Dynamical System Based on Conway's Game of Life," *Journal of Statistical Physics*, vol. 19, no. 3, pp. 293–314, 1978.

[22] B. A. McKinney, D. M. Reif, M. D. Ritchie, and J. H. Moore, "Machine Learning for Detecting Gene-Gene Interactions," *Applied Bioinformatics*, vol. 5, no. 2, pp. 77–88, 2006.

[23] B. Schonfisch and M. Kinder, "A Fish Migration Model," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*, 2002, pp. 210–219.

[24] S. Bandini and G. Pavesi, "Simulation of Vegetable Populations Dynamics Based on Cellular Automata," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*. Springer, 2002, pp. 202–209.

[25] S. Bandini, S. Manzoni, G. Mauri, and S. Redaelli, "Emergent Pattern Interpretation in Vegetable Population Dynamics," *Journal of Cellular Automata*, vol. 2, no. 2, 2007.

[26] I. Karafyllidis and A. Thanailakis, "A Model for Predicting Forest Fire Spreading Using Cellular Automata," *Ecological Modelling*, vol. 99, no. 1, pp. 87–97, 1997.

[27] L. H. Encinas, S. H. White, A. M. Del Rey, and G. R. Sánchez, "Modelling Forest Fire Spread Using Hexagonal Cellular Automata," *Applied Mathematical Modelling*, vol. 31, no. 6, pp. 1213–1227, 2007.

[28] A. Alexandridis, D. Vakalis, C. I. Siettos, and G. V. Bafas, "A Cellular Automata Model for Forest Fire Spread Prediction: The Case of The Wildfire That Swept Through Spetses Island in 1990," *Applied Mathematics and Computation*, vol. 204, no. 1, pp. 191–201, 2008.

[29] S. H. White, A. M. del Rey, and G. R. Sánchez, "A Cellular Automata Model for Predicting Fire Spread," *WIT Transactions on Ecology and the Environment*, vol. 81, pp. 69–78, 2005.

[30] L. Russo, P. Russo, D. Vakalisc, and C. Siettos, "Detecting Weak points of Wildland Fire Spread: A Cellular Automata Model Risk Assessment Simulation Approach," *Chemical Engineering Transactions*, vol. 36, pp. 253–258, 2014.

[31] B. Chopard, A. Dupuis, and P. Luthi, "A Cellular Automata Model for Urban Traffic and Its Application to the City of Geneva," *Proceedings of Traffic and Granular*, 1997.

[32] S. Deep and A. Saklani, "Urban Sprawl Modeling using Cellular Automata," *The Egyptian Journal of Remote Sensing and Space Science*, vol. 17, no. 2, pp. 179–187, 2014.

[33] W. Knospe, L. Santen, A. Schadschneider, and M. Schreckenberg, "Empirical Test for Cellular Automaton Models of Traffic Flow," *Physical Review E*, vol. 70, no. 1, p. 016115, 2004.

[34] G. C. Sirakoulis, "Cellular Automata for Crowd Dynamics," in *International Conference on Implementation and Application of Automata*. Springer, 2014, pp. 58–69.

[35] D. Pal and M. Chunchu, "Cellular Automata Cell Structure for Modeling Heterogeneous Traffic," *European Transport Trasporti Europei*, vol. 45, pp. 50–63, 2010.

[36] S. Bandini, F. Rubagotti, G. Vizzari, and K. Shimura, "A Cellular Automata Based Model for Pedestrian and Group Dynamics: Motivations and First Experiments," in *International Conference on Parallel Computing Technologies*. Springer, 2011, pp. 125–139.

[37] L. Crociani, A. Gorrini, K. Nishinari, and S. Bandini, "A CA-Based Model of Dyads in Pedestrian Crowds: The Case of Counter Flow," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*. Springer, 2016, pp. 355–364.

[38] K. Nagel and M. Schreckenberg, "A Cellular Automaton Model for Freeway Traffic," *Journal de Physique I*, vol. 2, no. 12, pp. 2221–2229, 1992.

[39] B. Chopard, P. O. Luthi, and P.-A. Queloz, "Cellular Automata Model of Car Traffic in a Two-Dimensional Street Network," *Journal of Physics A: Mathematical and General*, vol. 29, no. 10, p. 2325, 1996.

[40] M. Kanai, K. Nishinari, and T. Tokihiro, "Stochastic Cellular-Automaton Model for Traffic Flow," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*. Springer, 2006, pp. 538–547.

[41] H. Hartman and P. Tamayo, "Reversible Cellular Automata and Chemical Turbulence," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1, pp. 293 – 306, 1990.

[42] G. A. Hedlund, "Endomorphisms and Automorphisms of the Shift Dynamical System," *Mathematical Systems Theory*, vol. 3, no. 4, pp. 320–375, 1969.

[43] D. Richardson, "Tessellations with Local Transformations," *Journal of Computer and System Sciences*, vol. 6, no. 5, pp. 373–388, 1972.

[44] P. Pal Chaudhuri, D. Roy Chowdhury, S. Nandi, and S. Chatterjee, *Additive Cellular Automata – Theory and Applications.* IEEE Computer Society Press, USA, ISBN 0-8186-7717-1, 1997, vol. 1.

[45] S. Das, "Theory and Applications of Nonlinear Cellular Automata In VLSI Design," Ph.D. dissertation, Bengal Engineering and Science University, Shibpur, India, 2007.

[46] S. Das and B. K. Sikdar, "Characterization of 1-d Periodic Boundary Reversible CA," *Electronic Notes in Theoretical Computer Science*, vol. 252, pp. 205–227, 2009.

[47] A. Martın del Rey and G. Rodrıguez Sınchez, "Reversibility of Linear Cellular Automata," *Applied Mathematics and Computation*, vol. 217, no. 21, pp. 8360–8366, 2011.

[48] Z. Cinkir, H. Akin, and I. Siap, "Reversibilty of 1D Cellular Automata with Periodic Boundary over Finite Fields $\mathbb{Z}_p$," *Journal of Statistical Physics*, vol. 143, pp. 807–823, 2011.

[49] S. Inokuchi, K. Honda, H. Y. Lee, T. Sato, Y. Mizoguchi, and Y. Kawahara, "On Reversible Cellular Automata with Finite Cell Array," in *International Conference on Unconventional Computation.* Springer, 2005, pp. 130–141.

[50] S. Ghosh, N. S. Maiti, P. Pal Chaudhuri, and B. K. Sikdar, "On Invertible Three Neighborhood Null-Boundary Uniform Cellular Automata," *Complex Systems*, vol. 20, pp. 47–65, 2011.

[51] N. S. Maiti, S. Ghosh, S. Munshi, and P. Pal Chaudhuri, "Linear Time Algorithm for Identifying the Invertibility of Null-Boundary Three Neighborhood Cellular Automata," *Complex Systems*, vol. 19, no. 1, pp. 89–113, 2010.

[52] K. Bhattacharjee, "Cellular Automata : Reversibility, Semi-reversibility and Randomness," Ph.D. dissertation, Indian Institute of Engineering Science and Technology, Shibpur, 2019.

[53] S. Nandi, B. K. Kar, and P. P. Chaudhuri, "Theory and Applications of Cellular Automata in Cryptography," *IEEE Transactions on Computers*, vol. 43, no. 12, pp. 1346–1357, 1994.

[54] J. Kari, "Universal Pattern Generation by Cellular Automata," *Theoretical Computer Science*, vol. 429, pp. 180 – 184, 2012.

[55] K. Sutner, "On the Computational Complexity of Finite Cellular Automata," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 87–97, 1995.

[56] A. Clementi and R. Impagliazzo, "The Reachability Problem for Finite Cellular Automata," *Information Processing Letters*, vol. 53, no. 1, pp. 27–31, 1995.

[57] K. Cattell and S. Zhang, "Minimal Cost One-Dimensional Linear Hybrid Cellular Automata of Degree through 500," *Journal of Electronic Testing: Theory and Applications*, vol. 6, pp. 255–258, 1995.

[58] K. Cattell and J. C. Muzio, "Analysis of One-Dimensional Linear Hybrid Cellular Automata over GF(q) Configuration: Theory and Application," *IEEE Transactions on Computers*, vol. 45, no. 7, pp. 782–792, 1996.

[59] K. Cattell, S. Zhang, M. Serra, and J. C. Muzio, "2-by-n Hybrid Cellular Automata with Regular Configuration: Theory and Application," *IEEE Transactions on Computers*, vol. 48, no. 3, pp. 285–295, 1999.

[60] N. Ganguly, B. K. Sikdar, and P. Pal Chaudhuri, "Exploring Cycle Structures of Additive Cellular Automata," *Fundamenta Informaticae - Membrane Computing*, vol. 2, pp. 137–154, 1987.

[61] P. D. Hortensius, R. D. Mcleod, W. Pries, D. M. Miller, and H. C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-in Self-Test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 8, pp. 842–859, 1989.

[62] P. Tsalides, T. York, and A. Thanailakis, "Pseudorandom Number Generators for VLSI Systems Based on Linear Cellular Automata," *IEE Proceedings E (Computers and Digital Techniques)*, vol. 138, no. 4, pp. 241–249, 1991.

[63] S. Tezuka and M. Fushimi, "A Method of Designing Cellular Automata as Pseudorandom Number Generators for Built-in Self-Test for VLSI," *Finite Fields : Theory, Applications and Algorithms, Contemporary Mathematics AMS*, vol. 168, pp. 363–367, 1994.

[64] N. Naskar, "Characterization and Synthesis of Non-uniform Cellular Automata with Point State Attractors," Ph.D. dissertation, Indian Institute of Engineering Science and Technology, Shibpur, India, 2015.

[65] K. Cattell and J. C. Muzio, "Synthesis of One-Dimensional Linear Hybrid Cellular Automata," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 3, pp. 325–335, 1996.

[66] M. Živković, "Generation of Primitive Binary Polynomials," *Filomat*, vol. 9, no. 3, pp. 961–965, 1995.

[67] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, 1998.

[68] M. Saito and M. Matsumoto, "SIMD-Oriented Fast Mersenne Twister: A 128-bit Pseudorandom Number Generator," in *Seventh International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing.* Springer, 2008, pp. 607–622.

[69] ——, "A PRNG Specialized in Double Precision Floating Point Numbers Using an Affine Transition," in *Eighth International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing.* Springer, 2009, pp. 589–602.

[70] D. Xu and Y. A. Tian, "A Comprehensive Survey of Clustering Algorithms," *Annals of Data Science*, vol. 2, pp. 165–193, 2015.

[71] G. Brock, V. Pihur, S. Datta, and S. Datta, "clValid: An R Package for Cluster Validation," *Journal of Statistical Software*, vol. 25, no. 4, pp. 1–22, 2008.

[72] J. von Neumann, *The Theory of Self-Reproducing Automata, A. W. Burks ed.* Univ. of Illinois Press, Urbana and London, 1966.

[73] J. Thatcher, "Universality in Von Neumann Cellular Model," in *Tech. Report 03105-30-T, ORA, University of Michigan*, 1964.

[74] M. Arbib, "Simple Self-Reproducing Universal Automata," *Information and Control*, vol. 9, no. 2, pp. 177–189, 1966.

[75] E. F. Codd, *Cellular Automata.* Academic Press Inc., 1968.

[76] E. R. Banks, "Information Processing and Transmission in Cellular Automata," Ph.D. dissertation, MIT, 1971.

[77] E. F. Moore, "Machine Models of Self Reproduction," in *Essays on Cellular Automata*, A. W. Burks, Ed. Urbana: University of Illinois Press, 1970.

[78] T. Toffoli and N. Margolus, *Cellular Automata Machines: A New Environment for Modeling.* The MIT Press, 1987.

[79] A. R. Smith(III), “Cellular Automata Complexity Trade-Offs,” *Information and Control*, vol. 18, no. 5, pp. 466–482, 1971.

[80] K. Morita, M. Margenstern, and K. Imai, “Universality of Reversible Hexagonal Cellular Automata,” *RAIRO - Theoretical Informatics and Applications*, vol. 33, no. 6, pp. 535–550, 1999.

[81] I. Siap, H. Akin, and S. Uğuz, “Structure and Reversibility of 2D Hexagonal Cellular Automata,” *Computers & Mathematics with Applications*, vol. 62, no. 11, pp. 4161–4169, 2011.

[82] K. Imai and K. Morita, “A Computation-Universal Two-Dimensional 8-State Triangular Reversible Cellular Automaton,” *Theoretical Computer Science*, vol. 231, no. 2, pp. 181–191, 2000.

[83] K. Morita, “Universality of 8-State Reversible and Conservative Triangular Partitioned Cellular Automata,” in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*, 2016, pp. 45–54.

[84] M. Margenstern and K. Morita, “A Polynomial Solution for 3-SAT in the Space of Cellular Automata in the Hyperbolic Plane,” *Journal of Universal Computer Science*, vol. 5, no. 9, pp. 563–573, 1999.

[85] ——, “NP Problems Are Tractable in the Space of Cellular Automata in the Hyperbolic Plane,” *Theoretical Computer Science*, vol. 259, no. 1-2, pp. 99–128, 2001.

[86] N. H. Packard and S. Wolfram, “Two-Dimensional Cellular Automata,” *Journal of Statistical Physics*, vol. 38, no. 5, pp. 901–946, 1985.

[87] J. Kari, “Reversibility of 2D Cellular Automata is Undecidable,” *Physica D*, vol. 45, pp. 379–385, 1990.

[88] B. Durand, “Global Properties of 2D Cellular Automata: Some Complexity Results,” in *Mathematical Foundations of Computer Science*, 1993, pp. 433–441.

[89] G. M. B. de Oliveira and S. R. C. Siqueira, “Parameter Characterization of Two-Dimensional Cellular Automata Rule Space,” *Physica D: Nonlinear Phenomena*, vol. 217, no. 1, pp. 1–6, 2006.

[90] C.-A. Gandin and M. Rappaz, “A 3D Cellular Automaton Algorithm for the Prediction of Dendritic Grain Growth,” *Acta Materialia*, vol. 45, no. 5, pp. 2187–2195, 1997.

[91] P. Sarkar and R. Barua, "Multi-Dimensional σ-Automata, π-Polynomial and Generalized s-Matrices," *Theoretical Computer Science*, vol. 197, no. 1-2, pp. 111–138, 1998.

[92] D. B. Miller and E. Fredkin, "Two-State, Reversible, Universal Cellular Automata in Three Dimensions," in *Proceedings of The Second Conference on Computing Frontiers*, 2005, p. 45–51.

[93] Y. Mo, B. Ren, W. Yang, and J. Shuai, "The 3-Dimensional Cellular Automata for HIV Infection," *Physica A: Statistical Mechanics and its Applications*, vol. 399, pp. 31–39, 2014.

[94] A. Dennunzio, E. Formenti, and M. Weiss, "Multidimensional Cellular Automata: Closing Property, Quasi-Expansivity, and (UN)decidability Issues," *Theoretical Computer Science*, vol. 516, pp. 40–59, 2014.

[95] S. Amoroso and Y. N. Patt, "Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tesselation Structures," *Journal of Computer and System Sciences*, vol. 6, pp. 448–464, 1972.

[96] K. Sutner, "De Bruijin Graphs and Linear Cellular Automata," *Complex Systems*, vol. 5, no. 1, pp. 19–30, 1991.

[97] W. Pries, A. Thanailakis, and H. C. Card, "Group Properties of Cellular Automata and VLSI Applications," *IEEE Transactions on Computers*, vol. 35, no. 12, pp. 1013–1024, 1986.

[98] B. K. Sikdar, N. Ganguly, and P. Pal Chaudhuri, "Design of Hierarchical Cellular Automata for On-Chip Test Pattern Generator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 12, pp. 1530–1533, 2002.

[99] J. R. Jump and J. S. Kirtane, "On the Interconnection Structure of Cellular Networks," *Information Control*, vol. 24, pp. 74–91, 1974.

[100] N. Boccara and H. Fukś, "Cellular Automaton Rules Conserving the Number of Active Sites," *Journal of Physics A: Mathematical and General*, vol. 31, no. 28, p. 6007, 1998.

[101] C. Dyer, "One-Way Bounded Cellular Automata," *Information Control*, vol. 44, pp. 261–281, 1980.

[102] S. Wolfram, *A New Kind of Science.* Wolfram Media, Inc., 2002.

[103] ——, *Cellular Automata and Complexity — Collected Papers.* Addison Wesley, 1994.

[104] ——, *Theory and Applications of Cellular Automata.* World Scientific, 1986.

[105] T. Vollmar, "Cellular Space and Parallel Algorithms : An Introductory Survey," *Parallel Computation-Parallel Mathematics*, pp. 49–58, 1977.

[106] P. Sarkar, "A Brief History of Cellular Automata," *ACM Computing Systems*, vol. 32, no. 1, pp. 80–107, 2000.

[107] M. Mitchell, J. P. Crutchfield, and R. Das, "Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work," in *Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96)*, 1996.

[108] M. Mitchell, "Computation in Cellular Automata: A Selected Review," in *Non-Standard Computation.* Wiley-VCH Verlag GmbH & Co. KGaA, 1998, pp. 95–140.

[109] M. Cook, "Universality in Elementary Cellular Automata," *Complex Systems*, vol. 15, no. 1, pp. 1–40, 2004.

[110] N. Ganguly, B. K. Sikdar, A. Deutsch, G. Canright, and P. Pal Chaudhuri, "A Survey on Cellular Automata," B.E. college, Tech. Rep., 2003.

[111] W. Li, "Power Spectra of Regular Languages and Cellular Automata," *Complex Systems*, vol. 1, pp. 107–130, 1987.

[112] W. Li and N. Packard, "The Structure of the Elementary Cellular Automata Rule Space," *Complex Systems*, vol. 4, no. 3, pp. 281–297, 1990.

[113] S. Ninagawa and A. Adamatzky, "Classifying Elementary Cellular Automata using Compressibility, Diversity and Sensitivity Measures," *International Journal of Modern Physics C*, vol. 25, no. 03, p. 1350098, 2014.

[114] J. Kari, "Theory of Cellular Automata: A Survey," *Theoretical Computer Science*, vol. 334, no. 1-3, pp. 3–33, 2005.

[115] K. Bhattacharjee, N. Naskar, S. Roy, and S. Das, "A Survey of Cellular Automata: Types, Dynamics, Non-uniformity and Applications," *Natural Computing*, vol. 19, no. 2, pp. 433–461, 2020.

[116] E. R. Banks, "Universality in Cellular Automata," in *Proceedings of IEEE Conference Record of Eleventh Annual Symposium on Switching and Automata Theory*, 1970, pp. 194–215.

[117] J. C. Dubacq, "How To Simulate Turing Machines by Invertible One-Dimensional Cellular Automata," *International Journal of Foundations Computer Science*, vol. 6, no. 4, pp. 395–402, 1995.

[118] K. Morita and M. Harao, "Computation Universality of One Dimensional Reversible (Injective) Cellular Automata," *IEICE Transactions*, vol. 72, no. 6, pp. 758–762, 1989.

[119] J. Albert and K. Culik II, "A Simple Universal Cellular Automaton and Its One-Way and Totalistic Version," *Complex Systems*, vol. 1, pp. 1–16, 1987.

[120] M. Gardner, "On Cellular Automata Self-Reproduction, the Garden of Eden and the Game of 'Life'," *Scientific American*, vol. 224, no. 2, pp. 112–117, 1971.

[121] J. O. Durand-Lose, "About the Universality of the Billiard Ball Model," in *Proceedings International Colloquium Universal Machines and Computations*, 1998, pp. 118–132.

[122] C. Lindgren and M. Nordahl, "Universal Computation in Simple One Dimensional Cellular Automata," *Complex Systems*, vol. 4, pp. 299–318, 1990.

[123] S. Wolfram, "Universality and Complexity in Cellular Automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 1–35, 1984.

[124] K. Culik, L. P. Hard, and S. Yu, "Computation Theoretic Aspects of Cellular Automata," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1-3, pp. 357–378, 1990.

[125] N. Ollinger and G. Richard, "Four States Are Enough!" *Theoretical Computer Science*, vol. 412, pp. 22–32, 2011.

[126] S. Di Gregorio and G. Trautteur, "On Reversibility in Cellular Automata," *Journal of Computer and System Sciences*, vol. 11, no. 3, pp. 382–391, 1975.

[127] T. Toffoli, "Computation and Construction Universality of Reversible Cellular Automata," *Journal of Computer and System Sciences*, vol. 15, no. 2, pp. 213–231, 1977.

[128] K. Morita, "Reversible Simulation of One-Dimensional Irreversible Cellular Automata," *Theoretical Computer Science*, vol. 148, no. 1, pp. 157–163, 1995.

[129] A. Maruoka and M. Kimura, "Injectivity and Surjectivity of Parallel Maps for Cellular Automata," *Journal of Computer and System Sciences*, vol. 18, no. 1, pp. 47–64, 1979.

[130] K. Culik, "On Invertible Cellular Automata," *Complex Systems*, vol. 1, no. 6, pp. 1035–1044, 1987.

[131] J. M. G. Soto, "Computation of Explicit Preimages in One-Dimensional Cellular Automata Applying the de bruijn Diagram," *Journal of Cellular Automata*, vol. 3, no. 3, pp. 219–230, 2008.

[132] J. Kari, "Reversible Cellular Automata," in *Proceedings of International Conference on Developments in Language Theory.* Springer Berlin Heidelberg, 2005, pp. 57–68.

[133] ——, "Reversibility and Surjectivity Problems of Cellular Automata," *Journal of Computer and System Sciences*, vol. 48, no. 1, pp. 149–182, 1994.

[134] K. Morita, "Reversible Computing and Cellular Automata - A Survey," *Theoretical Computer Science*, vol. 395, no. 1, pp. 101 – 131, 2008.

[135] H. Moraal, "Graph-Theoretical Characterization of Invertible Cellular Automata," *Physica D: Nonlinear Phenomena*, vol. 141, no. 1–2, pp. 1–18, 2000.

[136] J. C. S. T. Mora, G. J. Martínez, and H. V. McIntosh, "The Inverse Behavior of a Reversible One-Dimensional Cellular Automaton Obtained by a Single Welch Diagram," *Journal of Cellular Automata*, vol. 1, no. 1, pp. 25–39, 2006.

[137] J. Myhill, "The Converse of Moore's Garden-of-Eden Theorem," *Proceedings of the American Mathematical Society*, vol. 14, no. 4, pp. 685–686, 1963.

[138] S. Amoroso, G. Cooper, and Y. N. Patt, "Some Clarifications of the Concept of a Garden-of-Eden Configuration," *Journal of Computer and System Sciences*, vol. 10, no. 1, pp. 77–82, 1975.

[139] M. Margenstern, "About the Garden of Eden Theorems for Cellular Automata in the Hyperbolic Plane," *Electronic Notes in Theoretical Computer Science*, vol. 252, pp. 93–102, 2009.

[140] T. Sato and N. Honda, "Certain Relations Between Properties of Maps of Tessellation Automata," *Journal of Computer and System Sciences*, vol. 15, no. 2, pp. 121–145, 1977.

[141] A. Maruoka and M. Kimura, "Condition for Injectivity of Global Maps for Tessellation Automata," *Information and Control*, vol. 32, no. 2, pp. 158–162, 1976.

[142] T. Toffoli and N. H. Margolus, "Invertible Cellular Automata: A Review," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1-3, pp. 229–253, 1990.

[143] A. Machì and F. Mignosi, "Garden of Eden Configurations for Cellular Automata on Cayley Graphs of Groups," *SIAM Journal on Discrete Mathematics*, vol. 6, no. 1, pp. 44–56, 1993.

[144] T. G. Ceccherini-Silberstein, A. Machi, and F. Scarabotti, "Amenable Groups and Cellular Automata," *Annales de l'institut Fourier*, vol. 49, no. 2, pp. 673–685, 1999.

[145] S. Capobianco, "Surjunctivity for Cellular Automata in Besicovitch Spaces," *Journal of Cellular Automata*, vol. 4, no. 2, pp. 89–98, 2009.

[146] E. Fredkin and T. Toffoli, "Conservative Logic," *International Journal of Theoretical Physics*, vol. 21, no. 3, pp. 219–253, 1982.

[147] M. Pivato, "Conservation Laws in Cellular Automata," *Nonlinearity*, vol. 15, no. 6, p. 1781, 2002.

[148] N. Boccara and H. Fukś, "Number-Conserving Cellular Automaton Rules," *Fundamenta Informaticae*, vol. 52, no. 1-3, pp. 1–13, 2002.

[149] H. Fukś, "Probabilistic Cellular Automata with Conserved Quantities," *Nonlinearity*, vol. 17, no. 1, p. 159, 2003.

[150] A. Moreira, "Universality and Decidability of Number-Conserving Cellular Automata," *Theoretical Computer Science*, vol. 292, no. 3, pp. 711–721, 2003.

[151] E. Formenti and A. Grange, "Number Conserving Cellular Automata II: Dynamics," *Theoretical Computer Science*, vol. 304, no. 1-3, pp. 269–290, 2003.

[152] B. Durand, E. Formenti, and Z. Róka, "Number-Conserving Cellular Automata I: Decidability," *Theoretical Computer Science*, vol. 299, no. 1-3, pp. 523–535, 2003.

[153] Morita, Kenichi and Imai, Katsunobu, "Number-Conserving Reversible Cellular Automata and their computation-universality," *RAIRO - Theoretical Informatics and Applications*, vol. 35, no. 3, pp. 239–258, 2001.

[154] J. Matsukidaira and K. Nishinari, "Euler-Lagrange Correspondence of Cellular Automaton for Traffic-Flow Models," *Physical Review Letters*, vol. 90, no. 8, p. 088701, 2003.

[155] B. S. Kerner, "Three-Phase Traffic Theory and Highway Capacity," *Physica A: Statistical Mechanics and its Applications*, vol. 333, pp. 379–440, 2004.

[156] K. Nagel, "Particle Hopping Models and Traffic Flow Theory," *Physical Review E*, vol. 53, no. 5, p. 4655, 1996.

[157] M. Fukui and Y. Ishibashi, "Traffic Flow in 1D Cellular Automaton Model Including Cars Moving with High Speed," *Journal of the Physical Society of Japan*, vol. 65, no. 6, pp. 1868–1870, 1996.

[158] S. Das, "Cellular Automata Based Traffic Model That Allows the Cars to Move with a Small Velocity During Congestion," *Chaos, Solitons & Fractals*, vol. 44, no. 4-5, pp. 185–190, 2011.

[159] S. Das, M. Saha, and B. K. Sikdar, "A Cellular Automata Based Model for Traffic in Congested City," in *IEEE International Conference on Systems, Man and Cybernetics.* IEEE, 2009, pp. 2397–2402.

[160] T. Kohyama, "Cellular Automata with Particle Conservation," *Progress of Theoretical Physics*, vol. 81, no. 1, pp. 47–59, 1989.

[161] ——, "Cluster Growth in Particle-Conserving Cellular Automata," *Journal of Statistical Physics*, vol. 63, no. 3, pp. 637–651, 1991.

[162] N. Margolus, "Physics-Like Models of Computation," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 81–95, 1984.

[163] T. Hattori and S. Takesue, "Additive Conserved Quantities in Discrete-Time Lattice Dynamical Systems," *Physica D: Nonlinear Phenomena*, vol. 49, no. 3, pp. 295 – 322, 1991.

[164] A. Dennunzio, E. Formenti, and J. Provillard, "Three Research Directions in Non-uniform Cellular Automata," *Theoretical Computer Science*, vol. 559, pp. 73–90, 2014.

[165] S. Das, "Characterization of Non-uniform Number Conserving Cellular Automata," in *Proceedings of International workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA).* Springer, 2011, pp. 17–28.

[166] R. Hazari and S. Das, “Number Conservation Property of Elementary Cellular Automata Under Asynchronous Update,” *Complex Systems*, vol. 23, no. 2, pp. 177–195, 2014.

[167] ——, “Number Conservation Property of Binary Cellular Automata Under α-Asynchronous Update,” *Journal of Cellular Automata*, vol. 13, no. 3, pp. 247–265, 2018.

[168] ——, “Rule 136 and Its Equivalents Are Liberal, But Become Conservative in their Conjugal Life,” *International Journal of Modern Physics C*, vol. 29, no. 06, p. 1850040, 2018.

[169] M. Mitchell, P. T. Hraber, and J. P. Crutchfield, “Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations,” *Complex Systems*, vol. 7, pp. 89–130, 1993.

[170] G. Cattaneo, E. Formenti, L. Margara, and G. Mauri, “On the Dynamical Behavior of Chaotic Cellular Automata,” *Theoretical Computer Science*, vol. 217, no. 1, pp. 31–51, 1999.

[171] G. Cattaneo, M. Finelli, and L. Margara, “Investigating Topological Chaos by Elementary Cellular Automata Dynamics,” *Theoretical Computer Science*, vol. 244, no. 1 - 2, pp. 219 – 241, 2000.

[172] L. Acerbi, A. Dennunzio, and E. Formenti, “Conservation of Some Dynamical Properties for Operations on Cellular Automata,” *Theoretical Computer Science*, vol. 410, no. 38-40, pp. 3685–3693, 2009.

[173] M. Eisele, “Long-Range Correlations in Chaotic Cellular Automata,” *Physica D: Nonlinear Phenomena*, vol. 48, no. 2-3, pp. 295–310, 1991.

[174] G. Cattaneo, E. Formenti, L. Margara, and G. Mauri, “On the Dynamical Behavior of Chaotic Cellular Automata,” *Theoretical Computer Science*, vol. 217, no. 1, pp. 31–51, 1999.

[175] M. Land and R. K. Belew, “No Perfect Two-State Cellular Automata for Density Classification Exists,” *Physical Review Letters*, vol. 74, no. 25, p. 5148, 1995.

[176] J. Kari and B. Le Gloannec, “Modified Traffic Cellular Automaton for the Density Classification Task,” *Fundamenta Informaticae*, vol. 116, no. 1-4, pp. 141–156, 2012.

[177] F. J. Morales, J. Crutchfield, and M. Mitchell, “Evolving Two-Dimensional Cellular Automata to Perform Density Classification: A Report on Work in Progress,” *Parallel Computing*, vol. 27, no. 5, pp. 571–585, 2001.

[178] N. S. Maiti, S. Munshi, and P. Pal Chaudhuri, "An Analytical Formulation for Cellular Automata (CA) Based Solution of Density Classification Task (DCT)," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*, 2006, pp. 147–156.

[179] H. Fuks-acute, "Solution of the Density Classification Problem with Two Cellular Automata Rules," *Physical Review E*, vol. 55, no. 3, p. R2081, 1997.

[180] P. P. de Oliveira, "Conceptual Connections Around Density Determination in Cellular Automata," in *International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA)*. Springer, 2013, pp. 1–14.

[181] K. Noguchi, "Simple 8-State Minimal Time Solution to the Firing Squad Synchronization Problem," *Theoretical Computer Science*, vol. 314, no. 3, pp. 303 – 334, 2004.

[182] H. Umeo, M. Hisaoka, and T. Sogabe, "A Survey on Optimum-Time Firing Squad Synchronization Algorithms for One-Dimensional Cellular Automata," *International Journal of Unconventional Computing*, vol. 1, no. 4, pp. 403–426, 2005.

[183] L. Manzoni and H. Umeo, "The Firing Squad Synchronization Problem on CA with Multiple Updating Cycles," *Theoretical Computer Science*, vol. 559, pp. 108 – 117, 2014.

[184] K. Culik and S. Dube, "An Efficient Solution to the Firing Mob Problem," *Theoretical Computer Science*, vol. 91, pp. 57–69, 1991.

[185] P. Banda, J. Caughman, and J. Pospichal, "Configuration Symmetry and Performance Upper Bound of One-Dimensional Cellular Automata for the Leader Election Problem," *Journal of Cellular Automata*, vol. 10, no. 1-2, pp. 1–21, 2015.

[186] R. Vollmar, "On Two Modified Problems of Synchronization in Cellular Automata," *Acta Cybernetica*, vol. 3, no. 4, pp. 293–300, 1977.

[187] A. Kundu, A. Pal, T. Sarkar, M. Banerjee, S. Guha, and D. Mukhopadhyay, "Comparative Study on Null Boundary and Periodic Boundary 3-Neighborhood Multiple Attractor Cellular Automata for Classification," in *Third International Conference on Digital Information Management (ICDM)*, 2008, pp. 204–209.

[188] J. Jin and Z. H. Wu, "A Secret Image Sharing Based on Neighborhood Configurations of 2-D Cellular Automata," *Optics & Laser Technology*, vol. 44, no. 3, pp. 538 – 548, 2012.

[189] S. Nandi and P. Pal Chaudhuri, "Analysis of Periodic and Intermediate Boundary 90/150 Cellular Automata," *IEEE Transactions on Computers*, vol. 45, no. 1, pp. 1–12, 1996.

[190] S. Cheybani, J. Kertész, and M. Schreckenberg, "Stochastic Boundary Conditions in the Deterministic Nagel-Schreckenberg Traffic Model," *Physical Review E*, vol. 63, p. 016107, 2000.

[191] K. Nakamura, "Asynchronous Cellular Automata and Their Computational Ability," *Systems, Computers, Controls*, vol. 5, no. 5, pp. 58–66, 1974.

[192] C. Marr and M. T. Hütt, "Outer-Totalistic Cellular Automata on Graphs," *Physics Letters A*, vol. 373, no. 5, pp. 546 – 549, 2009.

[193] P. D. Hortensius, R. D. McLeod, and H. C. Card, "Parallel Random Number Generation for VLSI Systems Using Cellular Automata," *IEEE Transactions on Computers*, vol. 38, no. 10, pp. 1466–1473, 1989.

[194] A. K. Das, A. Sanyal, and P. Pal Chaudhuri, "On Characterization of Cellular Automata with Matrix Algebra," *Information Sciences*, vol. 61, no. 3, pp. 251–277, 1992.

[195] M. Serra, T. Slater, J. Muzio, and D. Miller, "The Analysis of One-Dimensional Linear Cellular Automata and Their Aliasing Properties," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 7, pp. 767–778, 1990.

[196] G. Cattaneo, A. Dennunzio, E. Formenti, and J. Provillard, "Non-uniform Cellular Automata," in *Proceedings of Third International Conference on Language and Automata Theory and Applications, (LATA)*, 2009, pp. 302–313.

[197] A. Dennunzio, E. Formenti, and J. Provillard, "Non-uniform Cellular Automata: Classes, Dynamics, and Decidability," *Information and Computation*, vol. 215, pp. 32–46, 2012.

[198] V. Salo, "Realization Problems for Non-uniform Cellular Automata," *Theoretical Computer Science*, vol. 559, pp. 91–107, 2014.

[199] A. Lindenmayer, "Mathematical Models for Cellular Interactions in Development I. Filaments with One-Sided Inputs," *Journal of Theoretical Biology*, vol. 18, no. 3, pp. 280–299, 1968.

[200] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. Springer Science & Business Media, 2012.

[201] U. Pesavento, "An Implementation of von Neumann's Self-Reproducing Machine," *Artificial Life*, vol. 2, no. 4, pp. 337–354, 1995.

[202] J. H. Holland, *Studies of the Spontaneous Emergence of Self-Replicating Systems using Cellular Automata and Formal Grammars.* University of Michigan. Department of Computer and Communication Science, 1976.

[203] G. B. Ermentrout and L. Edelstein-Keshet, "Cellular Automata Approaches to Biological Modeling," *Journal of Theoretical Biology*, vol. 160, pp. 97–133, 1993.

[204] L. Reshodko and Z. Drska, "Biological Systems of Cellular Organization and their Computer Models," *Journal of Theoretical Biology*, vol. 69, no. 4, pp. 563–579, 1977.

[205] D. Green, "Cellular Automata Models in Biology," *Mathematical and Computer Modelling*, vol. 13, no. 6, pp. 69–74, 1990.

[206] T. Toffoli, "Cellular Automata as an Alternative to (Rather Than an Approximation of) Differential Equations in Modeling Physics," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 117–127, 1984.

[207] F. B. Manning, "An Approach to Highly Integrated, Computer-Maintained Cellular Arrays," *IEEE Transactions on Computers*, vol. 26, no. 06, pp. 536–552, 1977.

[208] A. Atrubin, "A One-Dimensional Real-Time Iterative Multiplier," *IEEE Transactions on Electronic Computers*, no. 3, pp. 394–399, 1965.

[209] S. N. Cole, "Real-Time Computation by n-Dimensional Iterative Arrays of Finite-State Machines," in *Seventh Annual Symposium on Switching and Automata Theory (swat 1966).* IEEE, 1966, pp. 53–77.

[210] P. C. Fischer, "Generation of Primes by a One-Dimensional Real-Time Iterative Array," *Journal of the ACM (JACM)*, vol. 12, no. 3, pp. 388–394, 1965.

[211] H. Nishio, "Real Time Sorting of Binary Numbers by One-Dimensional Cellular Automata," Technical Report, Kyoto University, Tech. Rep., 1981.

[212] H. Nishio and Y. Kobuchi, "Fault Tolerant Cellular Space," *Journal of Compututer and System Sciences*, vol. 11, pp. 150–170, 1975.

[213] S. C. Benjamin and N. F. Johnson, "A Possible Nanometer-Scale Computing Device Based on an Adding Cellular Automaton," *Applied Physics Letters*, vol. 70, no. 17, pp. 2321–2323, 1997.

[214] S. Wolfram, "Origins of Randomness in Physical Systems," *Physical Review Letters*, vol. 55, no. 5, p. 449, 1985.

[215] S. Das and B. K. Sikdar, "A Scalable Test Structure for Multicore Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 1, pp. 127–137, 2009.

[216] M. Matsumoto, "Simple Cellular Automata as Pseudorandom m-Sequence generators for Built-In Self-Test," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 31–42, 1998.

[217] S. Wolfram, "Cryptography With Cellular Automata," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1985, pp. 429–432.

[218] S. Das and D. Roy Chowdhury, "CAR30: A New Scalable Stream Cipher With rule 30," *Cryptography and Communications*, vol. 5, no. 2, pp. 137–162, 2013.

[219] E. Formenti, K. Imai, B. Martin, and J.-B. Yunès, "Advances on Random Sequence Generation by Uniform Cellular Automata," in *Computing with New Resources: Essays Dedicated to Jozef Gruska on the Occasion of His eighth Birthday*. Springer, 2014, pp. 56–70.

[220] A. Leporati and L. Mariot, "Cryptographic Properties of Bipermutive Cellular Automata Rules," *Journal of Cellular Automata*, vol. 9, no. 5-6, pp. 437–475, 2014.

[221] S. Das and D. Roy Chowdhury, "Cryptographically Suitable Maximum Length Cellular Automata," *Journal of Cellular Automata*, vol. 6, no. 6, pp. 439–459, 2011.

[222] S. Wolfram, "Random Sequence Generation by Cellular Automata," *Advances in Applied Mathematics*, vol. 7, no. 2, pp. 123–169, 1986.

[223] R. Alonso-Sanz and L. Bull, "Elementary Cellular Automata with Minimal Memory and Random Number Generation," *Complex Systems*, vol. 18, no. 2, pp. 195 – 213, 2009.

[224] M. Tomassini, M. Sipper, and M. Perrenoud, "On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata," *IEEE Transactions on Computers*, vol. 49, no. 10, pp. 1146–1151, 2000.

[225] A. Compagner and A. Hoogland, "Maximum-Length Sequences, Cellular Automata, and Random Numbers," *Journal of Computational Physics*, vol. 71, no. 2, pp. 391 – 428, 1987.

[226] M. Sipper and M. Tomassini, "Generating Parallel Random Number Generators by Cellular Programming," *International Journal of Modern Physics C*, vol. 7, no. 02, pp. 181–190, 1996.

[227] S.-U. Guan and S. K. Tan, "Pseudorandom Number Generation with Self-Programmable Cellular Automata," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 7, pp. 1095–1101, 2004.

[228] Q. Wang, S. Yu, W. Ding, and M. Leng, "Generating High-Quality Random Numbers by Cellular Automata with PSO," in *Fourth International Conference on Natural Computation*, vol. 7. IEEE, 2008, pp. 430–433.

[229] S.-U. Guan and S. Zhang, "An Evolutionary Approach to the Design of Controllable Cellular Automata Structure for Random Number Generation," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 1, pp. 23–36, 2003.

[230] K. Bhattacharjee, D. Paul, and S. Das, "Pseudo-random Number Generation Using A 3-State Cellular Automaton," *International Journal of Modern Physics C*, vol. 28, no. 06, p. 1750078, 2017.

[231] K. Bhattacharjee and S. Das, "A List of Tri-State Cellular Automata Which Are Potential Pseudo-Random Number Generators," *International Journal of Modern Physics C*, vol. 29, no. 09, p. 1850088, 2018.

[232] ——, "Random Number Generation Using Decimal Cellular Automata," *Communications in Nonlinear Science and Numerical Simulation*, vol. 78, p. 104878, 2019.

[233] S. Das, A. Kundu, S. Sen, B. K. Sikdar, and P. Pal Chaudhuri, "Non-Linear Celluar Automata Based PRPG Design (Without Prohibited Pattern Set) In Linear Time Complexity," in *Proceedings of Asian Test Symposium*, 2003, pp. 78–83.

[234] A. K. Das, "Additive Cellular Automata : Theory and Application as a Built-in Self-test Structure," Ph.D. dissertation, IIT, Kharagpur, India, 1990.

[235] D. RoyChowdhury, "Theory and Applications of Additive Cellular Automata for Reliable and Testable VLSI Circuit Design," Ph.D. dissertation, IIT, Kharagpur, India, 1992.

[236] P. Tsalides, "Cellular Automata-Based Built-in Self-Test Structures for VLSI Systems," *Electronics Letters*, vol. 26, no. 17, pp. 1350–1352, 1990.

[237] R. Chakraborty and D. R. Chowdhury, “A Novel Seed Selection Algorithm for Test Time Reduction in BIST,” in *2009 Asian Test Symposium.* IEEE, 2009, pp. 15–20.

[238] M. Khare and A. Albicki, “Cellular Automata Used for Test Pattern Generation,” in *International Conference on Computer Design*, 1987, pp. 56–59.

[239] A. K. Das and P. P. Chaudhuri, “An Efficient On-Chip Deterministic Test Pattern Generation Scheme,” *Microprocessing and Microprogramming*, vol. 26, no. 3, pp. 195–204, 1989.

[240] N. Ganguly, B. K. Sikdar, and P. Pal Chaudhuri, “Design of an On-Chip Test Pattern Generator Without Prohibitited Pattern Set (PPS),” in *Proceedings of ASP-DAC/VLSI Design 2002, India*, 2002, pp. 689–694.

[241] N. Ganguly, A. S. Nandi, S. Das, B. K. Sikdar, and P. Pal Chaudhuri, “An Evolutionary Design of Pseudo-Random Test Pattern Generator Without Prohibited Pattern Set (PPS),” in *Proceedings of Asian Test Symposium*, 2002, pp. 260–265.

[242] A. K. Das and P. P. Chaudhuri, “Vector Space Theoretic Analysis of Additive Cellular Automata and Its Application for Pseudoexhaustive Test Pattern Generation,” *IEEE Transactions on Computers*, vol. 42, no. 3, pp. 340–352, 1993.

[243] S. Das, N. Ganguly, B. K. Sikdar, and P. Pal Chaudhuri, “Design of a Universal BIST (UBIST) Structure,” in *Proceedings of Sixteenth International Conference on VLSI Design*, January 2003, pp. 161–166.

[244] A. R. Khan, P. P. Choudhury, K. Dihidar, S. Mitra, and P. Sarkar, “VLSI Architecture of a Cellular Automata Machine,” *Computers & Mathematics with Applications*, vol. 33, no. 5, pp. 79–94, 1997.

[245] D. Mukhopadhyay, P. Joshi, and D. Roy Chowdhury, “VLSI Architecture of a Cellular Automata Based One-Way Function,” *Journal of Computers*, vol. 3, no. 5, pp. 46–53, 2008.

[246] S. Nandi, C. Rambabu, and P. P. Chaudhari, “A VLSI Architecture for Cellular Automata Based Reed-Solomon Decoder,” in *Proceedings Fourth International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN’99).* IEEE, 1999, pp. 158–165.

[247] V. A. Mardiris, G. C. Sirakoulis, and I. G. Karafyllidis, “Automated design architecture for 1-d cellular automata using quantum cellular automata,” *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2476–2489, 2014.

[248] F. Bao, "Cryptanalysis of a New Cellular Automata Cryptosystem," in *Australasian Conference on Information Security and Privacy.* Springer, 2003, pp. 416–427.

[249] J. Sung, D. Hong, and S. Hong, "Cryptanalysis of an Involutional Block Cipher Using Cellular Automata," *Information Processing Letters*, vol. 104, no. 5, pp. 183–185, 2007.

[250] P. Rio Piedras, "Cellular Automaton Public-key Cryptosystem," *Complex Systems*, vol. 1, pp. 51–57, 1987.

[251] S. Das and D. R. Chowdhury, "CASTREAM: A New Stream Cipher Suitable for Both Hardware and Software," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI).* Springer, 2012, pp. 601–610.

[252] D. Mukhopadhyay and D. Roy chowdhury, "Theory of a Class of Complemented Group Cellular Automata and its Application to Cryptography," *Journal of Cellular Automata*, vol. 2, no. 3, pp. 243–271, 2007.

[253] J. Kari, "Cryptosystems Based on Reversible Cellular Automata," *Manuscript*, 1992.

[254] B. Sethi and S. Das, "On the Use of Asynchronous Cellular Automata in Symmetric-Key Cryptography," in *International Symposium on Security in Computing and Communication.* Springer, 2016, pp. 30–41.

[255] M. Seredynski and P. Bouvry, "Block Cipher Based on Reversible Cellular Automata," *New Generation Computing*, vol. 23, no. 3, pp. 245–258, 2005.

[256] Z. Chuanwu, P. Qicong, and L. Yubo, "Encryption Based on Reversible Cellular Automata," in *International Conference on Communications, Circuits and Systems and West Sino Expositions*, vol. 2. IEEE, 2002, pp. 1223–1226.

[257] A. Clarridge and K. Salomaa, "A Cryptosystem Based on the Composition of Reversible Cellular Automata," in *Proceedings of Third International Conference on Language and Automata Theory and Applications, LATA*. Springer, 2009, pp. 314–325.

[258] F. Seredynski, P. Bouvry, and A. Y. Zomaya, "Cellular Automata Computations and Secret Key Cryptography," *Parallel Computing*, vol. 30, no. 5-6, pp. 753–766, 2004.

[259] A. Wuensche, “Cellular Automata Encryption: The Reverse Algorithm, z-Parameter and Chain-Rules,” *Parallel Processing Letters*, vol. 19, no. 02, pp. 283–297, 2009.

[260] S. Das and D. R. Chowdhury, “Generating Cryptographically Suitable Non-linear Maximum Length Cellular Automata,” in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI).* Springer, 2010, pp. 241–250.

[261] X. Wang and D. Luan, “A Novel Image Encryption Algorithm Using Chaos and Reversible Cellular Automata,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 11, pp. 3075–3085, 2013.

[262] K. M. Faraoun, “A Genetic Strategy to Design Cellular Automata Based Block Ciphers,” *Expert Systems with Applications*, vol. 41, no. 17, pp. 7958–7967, 2014.

[263] D. Roy Chowdhury, S. Basu, I. Gupta, and P. Pal Chaudhuri, “Design of CAECC - Cellular Automata Based Error Correcting Code,” *IEEE Transactions on Computers*, vol. 43, no. 6, pp. 759–764, 1994.

[264] K. Paul and D. Roy Chowdhury, “Application of GF(2^p) CA in Burst Error Correcting Codes,” in *Proceedings of International Conference of VLSI Design*, January 2000, pp. 562–567.

[265] P. D. Hortensius, R. D. McLeod, and H. C. Card, “Cellular Automata-Based Signature Analysis for Built-in Self-Test,” *IEEE Transactions on Computers*, vol. 39, no. 10, pp. 1273–1283, 1990.

[266] A. K. Das, D. Saha, A. Roy Chowdhury, S. Misra, and P. Pal Chaudhuri, “Signature analyzer based on additive cellular automata,” in *Digest of Papers. Fault-Tolerant Computing: Twentieth International Symposium.* IEEE Computer Society, 1990, pp. 265–272.

[267] S. Misra, “Theory and Application of Additive Cellular Automata for Easily Testable VLSI Circuit Design,” Ph.D. dissertation, IIT, Kharagpur, India, 1992.

[268] O. Lafe, “Data Compression and Encryption Using Cellular Automata Transforms,” *Engineering Applications of Artificial Intelligence*, vol. 10, no. 6, pp. 581–591, 1997.

[269] S. Bhattacharjee, J. Bhattacharya, and P. Pal Chaudhuri, “An Efficient Data Compression Hardware based on Cellular Automata,” in *Proceedings of Data Compression Conference (DCC95)*, 1995, p. 472.

[270] A. R. Khan, P. P. Choudhury, K. Dihidar, and R. Verma, "Text Compression Using Two-Dimensional Cellular Automata," *Computers & Mathematics with Applications*, vol. 37, no. 6, pp. 115–127, 1999.

[271] C. Shaw, B. K. Sikdar, and N. C. Maiti, "CA Based Document Compression Technology," in *Proceedings of Eleventh International Conference on Neural Information Processing*, 2004, pp. 679–685.

[272] K. Paul, S. Pal Chaudhuri, R. Ghosal, B. K. Sikdar, and D. Roy Chowdhury, "GF(2^p) CA Based Vector Quantization for Fast Encoding of Still Images," in *Proceedings of Thirteenth International Conference on VLSI Design*. IEEE, 2000, pp. 140–143.

[273] K. Paul, D. Roy Chowdhury, and P. Pal Chaudhuri, "Cellular Automata Based Transform Coding for Image Compression," in *Proceedings of International Conference on High Performance Computing (HiPC)*. Springer, 1999, pp. 269–273.

[274] ——, "Scalable Pipelined Micro-Architecture for Wavelet Transform," in *Proceedings of International Conference on VLSI Design*, January 2000, pp. 144–147.

[275] K. Paul, P. Dutta, D. Roy Chowdhury, P. K. Nandi, and P. Pal Chaudhuri, "A VLSI Arcitecture for On-Line Image Decompression using GF(2^8) Cellular Automata," in *Proceedings of International Conference on VLSI Design*, January 1999, pp. 532–537.

[276] R. Ye and H. Li, "A Novel Image Scrambling and Watermarking Scheme Based on Cellular Automata," in *Proceedings of International Symposium on Electronic Commerce and Security*, 2008, pp. 938–941.

[277] O. E. Lafe, "Method and Apparatus for Video Compression Using Sequential Frame Cellular Automata Transforms," 2002, US Patent 6,456,744.

[278] C. Shaw, D. Chatterjee, P. Maji, S. Sen, and P. Pal Chaudhuri, "A Pipeline Architecture For Encompression (Encryption + Compression) Technology," in *Proceedings of Sixteenth International Conference on VLSI Design*. IEEE, 2003, pp. 277–282.

[279] C. Shaw, P. Maji, S. Saha, B. K. Sikdar, S. Roy, and P. Pal Chaudhuri, "Cellular Automata Based Encompression Technology for Voice Data," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*, 2004, pp. 258–267.

[280] J. Li, Z. Chen, and T. Qin, "Using Cellular Automata to Model Evolutionary Dynamics of Social Network," in *Eleventh International Symposium on Operations Research and its Applications in Engineering, Technology and Management (ISORA 2013).* IET, 2013, pp. 1–6.

[281] K. Małecki, J. Jankowski, and M. Rokita, "Application of Graph Cellular Automata in Social Network Based Recommender System," in *International Conference on Computational Collective Intelligence.* Springer, 2013, pp. 21–29.

[282] R. Hunt, E. Mendi, and C. Bayrak, "Using Cellular Automata to Model Social Networking Behavior," in *Twelfth International Symposium on Computational Intelligence and Informatics (CINTI).* IEEE, 2011, pp. 287–290.

[283] R. A. Zimbres and P. P. B. de Oliveira, "Dynamics of Quality Perception in a Social Network: A Cellular Automaton Based Model in Aesthetics Services," *Electronic Notes in Theoretical Computer Science*, vol. 252, pp. 157 – 180, 2009.

[284] V. Dabbaghian, V. K. Mago, T. Wu, C. Fritz, and A. Alimadad, "Social Interactions of Eating Behaviour Among High School Students: A Cellular Automata Approach," *BMC Medical Research Methodology*, vol. 12, no. 1, p. 155, 2012.

[285] R. M. Z. dos Santos and S. Coutinho, "Dynamics of HIV Infection : A Cellular Automata Approach," *Physical Review Letters*, vol. 87, p. 168102, 2001.

[286] J. Moreira and A. Deutsch, "Cellular Automaton Models of Tumor Development: A Critical Review," *Advances in Complex Systems*, vol. 05, no. 02n03, pp. 247–267, 2002.

[287] J. H. Moore and L. W. Hahn, "A Cellular Automata-based Pattern Recognition Approach for Identifying Gene-Gene and Gene-Environment Interactions," *American Journal of Human Genetics*, vol. 67, no. 52, 2000.

[288] ——, "Multilocus Pattern Recognition using One-dimensional Cellular Automata and Parallel Genetic Algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001.

[289] C. Burks and D. Farmer, "Towards Modeling DNA Sequences As Automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 157–167, 1984.

[290] S. A. Smith, R. C. Watt, and S. R. Hameroff, "Cellular Automata in Cytoskeletal Lattices," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 168–174, 1984.

[291] D. A. Young, "A Local Activator-Inhibitor Model of Vertebrate Skin Patterns," *Mathematical Biosciences*, vol. 72, no. 1, pp. 51–58, 1984.

[292] S. Mitra, S. Das, P. Pal Chaudhuri, and S. Nandi, "Architecture of a VLSI Chip for Modeling Amino Acid Sequence in Proteins," in *Proceedings of Ninth International Conference on VLSI Design*, 1996, pp. 316–317.

[293] S. Ghosh, T. Bachhar, N. S. Maiti, I. Mitra, and P. Pal Chaudhuri, "Theory and Application of Equal Length Cycle Cellular Automata (ELCCA) for enzyme classification," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*, 2010, pp. 46–57.

[294] X. Xiao, S. Shao, Z. Huang, and K. Chou, "Using Pseudo Amino Acid Composition to Predict Protein Structural Classes: Approached with Complexity Measure Factor," *Journal of Comutational Chemistry*, vol. 27, no. 4, pp. 478–482, 2006.

[295] J. Santos, P. Villot, and M. Diéguez, "Protein Folding with Cellular Automata in the 3D HP Model," in *Proceedings of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation*, 2013, pp. 1595–1602.

[296] S. Ghosh, N. Laskar, S. Mahapatra, and P. Pal Chaudhuri, "Probabilistic Cellular Automata Model for Identification of CpG island in DNA String," in *Proceedings of Indian International Conference on Artificial Intelligence*, 2007, pp. 1490–1509.

[297] S. Ghosh, N. S. Maiti, and P. Pal Chaudhuri, "Theory and Application of Restricted Five Neighborhood Cellular Automata (R5NCA) for Protein Structure Prediction," in *Proceedings of International Conference on Cellular Automata for Research and Industry, (ACRI)*, G. C. Sirakoulis and S. Bandini, Eds. Springer, 2012, pp. 360–369.

[298] S. Ghosh, N. S. Maiti, and P. Pal Chaudhuri, "Cellular Automata Model for Protein Structure Synthesis (PSS)," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*, 2014, pp. 268–277.

[299] R. Raghavan, "Cellular Automata in Pattern Recognition," *Information Sciences*, vol. 70, no. 1-2, pp. 145–177, 1993.

[300] E. Jen, "Invariant Strings and Pattern-Recognizing Properties of One-Dimensional Cellular Automata," *Journal of Statistical Physics*, vol. 43, no. 1, pp. 243–265, 1986.

[301] R. Sommerhalder and S. C. van Westrhenen, "Parallel Language Recognition in Constant Time by Cellular Automata," *Acta Informatica*, vol. 19, no. 4, pp. 397–407, 1983.

[302] A. R. Smith III, "Real-Time Language Recognition by One-Dimensional Cellular Automata," *Journal of Computer and System Sciences*, vol. 6, no. 3, pp. 233–253, 1972.

[303] M. Mahajan, "Studies in Language Classes Defined by Different Time-Varying Cellular Automata," Ph.D. dissertation, Indian Institute of Technology, Madras, 1992.

[304] M. Kutrib and A. Malcher, "Fast Reversible Language Recognition Using Cellular Automata," *Information and Computation*, vol. 206, no. 9-10, pp. 1142–1151, 2008.

[305] O. H. Ibarra, M. A. Palis, and S. M. Kim, "Fast Parallel Language Recognition by Cellular Automata," *Theoretical Computer Science*, vol. 41, pp. 231–246, 1985.

[306] P. Tzionas, P. Tsalides, and A. Thanailakis, "Design and VLSI Implementation of a Pattern Classifier Using Pseudo 2D Cellular Automata," *IEE Proceedings G-Circuits, Devices and Systems*, vol. 139, no. 6, pp. 661–668, 1992.

[307] P. G. Tzionas, P. G. Tsalides, and A. Thanailakis, "A New, Cellular Automaton-Based, Nearest Neighbor Pattern Classifier and Its VLSI Implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 3, pp. 343–353, 1994.

[308] C. Walker, "Attractor Dominance Patterns in Sparsely Connected Boolean Nets," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1-3, pp. 441–451, 1990.

[309] M. Chady and R. Poli, "Evolution of Cellular Automaton Based Associative Memories," *Technical Report no. CSRP-97-15*, May 1997.

[310] K. Morita and S. Ueno, "Parallel Generation and Parsing of Array Languages Using Reversible Cellular Automata," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 8, no. 2, pp. 543–561, 1994.

[311] S. Chattopadhyay, S. Adhikari, S. Sengupta, and M. Pal, "Highly Regular, Modular, and Cascadable Design of Cellular Automata-Based Pattern Classifier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 6, pp. 724–735, 2000.

[312] N. Ganguly, "Cellular Automata Evolution : Theory and Applications in Pattern Recognition and Classification," Ph.D. dissertation, Bengal Engineering College (a Deemed University), India, 2004.

[313] N. Ganguly, B. K. Sikdar, J. Deb, D. Halder, and P. Pal Chaudhuri, "Hashing Through Cellular Automata," in *Proceedings of Eighth International Conference of Advance Computing and Communication, India*, 2000.

[314] *Evolving Cellular Automata Based Associative Memory For Pattern Recognition*, 2001.

[315] D. Roy Chowdhury, I. S. Gupta, and P. Pal Chaudhuri, "A low cost high capacity associative memory design using Cellular automata," in *Proceedings of VLSI Design*, January 1992, pp. 157–160.

[316] N. Ganguly, P. Maji, B. Sikdar, and P. Pal Chaudhuri, "Design and Characterization of Cellular Automata Based Associative Memory for Pattern Recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 672–678, 2004.

[317] P. Maji, "Cellular Automata Evolution for Pattern Recognition," Ph.D. dissertation, Jadavpur University, Kolkata, India, 2005.

[318] N. Ganguly, P. Maji, B. K. Sikdar, and P. Pal Chaudhuri, "Generalized Multiple Attractor Cellular Automata (GMACA) Model for Associative Memory," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 16, no. 7, pp. 781–796, 2002.

[319] P. Maji and P. P. Chaudhuri, "RBFFCA: A Hybrid Pattern Classifier Using Radial Basis Function and Fuzzy Cellular Automata," *Fundamenta Informaticae*, vol. 78, no. 3, pp. 369–396, 2007.

[320] ——, "Fuzzy Cellular Automata for Modeling Pattern Classifier," *IEICE Transactions on Information and Systems*, vol. 88, no. 4, pp. 691–702, 2005.

[321] S. Das, S. Mukherjee, N. Naskar, and B. K. Sikdar, "Characterization of Single Cycle CA and Its Application in Pattern Classification," *Electronic Notes in Theoretical Computer Science*, vol. 252, pp. 181–203, 2009.

[322] S. Adak, N. Naskar, P. Maji, and S. Das, "On Synthesis of Non-uniform Cellular Automata Having Only Point Attractors," *Journal of Cellular Automata*, vol. 12, no. 1-2, pp. 81–100, 2016.

[323] B. Sethi, S. Roy, and S. Das, "Asynchronous Cellular Automata and Pattern Classification," *Complexity*, vol. 21, no. S1, pp. 370–386, 2016.

[324] M. Espínola, J. A. Piedra-Fernández, R. Ayala, L. Iribarne, and J. Z. Wang, "Contextual and Hierarchical Classification of Satellite Images Based on Cellular Automata," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 2, pp. 795–809, 2014.

[325] J. Ponkaew, S. Wongthanavasu, and C. Lursinsap, "A Nonlinear Classifier using an Evolution of Cellular Automata," in *International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*. IEEE, 2011, pp. 1–5.

[326] M. Fang, W. Niu, and X. S. Zhang, "Research on the Classifier with the Tree Frame Based on Multiple Attractor Cellular Automaton," in *International Conference on Natural Computation.* IEEE, 2012, pp. 1079–1083.

[327] S. Wongthanavasu and J. Ponkaew, "A Cellular Automata-Based Learning Method for Classification," *Expert Systems with Applications*, vol. 49, pp. 99–111, 2016.

[328] P. Maji, R. Nandi, and P. P. Chaudhuri, "Application of Fuzzy Cellular Automata (FCA) for Modeling Tree-Structured Pattern Classifier," in *IICAI*, 2003, pp. 1220–1233.

[329] P. L. Rosin, "Training Cellular Automata for Image Processing," *IEEE Transactions on Image Proceedings*, vol. 15, no. 7, pp. 2076–2087, 2006.

[330] O. Kazar and S. Slatnia, "Evolutionary Cellular Automata for Image Segmentation and Noise Filtering Using Genetic Algorithms," *Journal of Applied Computer Science & Mathematics*, vol. 5, no. 10, pp. 33–40, 2011.

[331] A. R.Khan, "Replacement of some Graphics Routines with the Help of 2D Cellular Automata Algorithms for Faster Graphics Operations," Ph.D. dissertation, University of Kashmir, 1998.

[332] P. L. Rosin, "Image Processing Using 3-State Cellular Automata," *Computer Vision and Image Understanding*, vol. 114, no. 7, pp. 790–802, 2010.

[333] S. Wongthanavasu and R. Sadananda, "A CA-based Edge Operator and Its Performance Evaluation," *Journal of Visual Communication and Image Representation*, vol. 14, no. 2, pp. 83–96, 2003.

[334] S. Sadeghi, A. Rezvanian, and E. Kamrani, "An Efficient Method for Impulse Noise Reduction From Images Using Fuzzy Cellular Automata," *AEU-International Journal of Electronics and Communications*, vol. 66, no. 9, pp. 772 – 779, 2012.

[335] A. Scarioni and J. A. Moreno, "Border Detection in Digital Images with A Simple Cellular Automata Rule," in *Cellular Automata: Research towards Industry.* Springer, 1998, pp. 146–156.

[336] D. Ziou, S. Tabbone *et al.*, "Edge Detection Techniques - An Overview," *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, vol. 8, pp. 537–559, 1998.

[337] C.-L. Chang, Y.-J. Zhang, and Y.-Y. Gdong, "Cellular Automata for Edge Detection of Images," in *Proceedings of International Conference on Machine Learning and Cybernetics*, vol. 6. IEEE, 2004, pp. 3830–3834.

[338] P. J. Selvapeter and W. Hordijk, "Cellular Automata for Image Noise Filtering," in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC).* IEEE, 2009, pp. 193–197.

[339] F. Qadir, M. Peer, and K. Khan, "An Effective Image Noise Filtering Algorithm Using Cellular Automata," in *International Conference on Computer Communication and Informatics.* IEEE, 2012, pp. 1–5.

[340] X. Gu and Y. Sun, "Image Analysis of Ceramic Burning Based on Cellular Automata," *EURASIP Journal on Image and Video Processing*, vol. 2018, no. 1, pp. 1–11, 2018.

[341] N. R. da Silva, J. M. Baetens, M. W. da Silva Oliveira, B. De Baets, and O. M. Bruno, "Classification of Cellular Automata Through Texture Analysis," *Information Sciences*, vol. 370, pp. 33–49, 2016.

[342] S. Leguizamón, M. Espínola, R. Ayala, L. Iribarne, and M. Menenti, "Characterization of Texture in Images by Using A Cellular Automata Approach," in *World Summit on Knowledge Society.* Springer, 2010, pp. 522–533.

[343] N. R. da Silva, P. Van der Weeën, B. De Baets, and O. M. Bruno, "Improved Texture Image Classification Through the Use of A Corrosion-Inspired Cellular Automaton," *Neurocomputing*, vol. 149, pp. 1560–1572, 2015.

[344] X. Xu, L. Chen, and P. He, "Ant Clustering Embedded in Cellular Automata," in *European Conference on Artificial Life.* Springer, 2005, pp. 562–571.

[345] ——, "A Novel Ant Clustering Algorithm Based on Cellular Automata," *Web Intelligence and Agent Systems: An International Journal*, vol. 5, no. 1, pp. 1–14, 2007.

[346] A. V. Moere, J. J. Clayden, and A. Dong, "Data Clustering and Visualization Using Cellular Automata Ants," in *Australasian Joint Conference on Artificial Intelligence.* Springer, 2006, pp. 826–836.

[347] D. Shuai, Y. Dong, and Q. Shuai, "A New Data Clustering Approach: Generalized Cellular Automata," *Information Systems*, vol. 32, no. 7, pp. 968–977, 2007.

[348] J. de Lope and D. Maravall, "Data Clustering Using a Linear Cellular Automata-Based Algorithm," *Neurocomputing*, vol. 114, pp. 86–91, 2013.

[349] E. B. Dündar and E. E. Korkmaz, "Data Clustering with Stochastic Cellular Automata," *Intelligent Data Analysis*, vol. 22, no. 4, pp. 735–750, 2018.

[350] S. Saha, P. Maji, N. Ganguly, S. Roy, and P. P. Chaudhuri, "Cellular Automata Based Model for Pattern Clustering," in *Proceedings of International Conference on Advances in Pattern Recognition*, 2003, pp. 122–126.

[351] S. Ghosh, T. Bachhar, N. S. Maiti, I. Mitra, and P. P. Chaudhuri, "Theory and Application of Equal Length Cycle Cellular Automata (ELCCA) for Enzyme Classification," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI)*, 2010, p. 46–57.

[352] S. Das, B. K. Sikdar, and P. Pal Chaudhuri, "Characterization of Reachable/Nonreachable Cellular Automata States," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI).* Springer, 2004, pp. 813–822.

[353] S. Das and B. K. Sikdar, "Classification of ca rules targeting synthesis of reversible cellular automata," in *Proceedings of International Conference on Cellular Automata for Research and Industry (ACRI).* Springer, 2006, pp. 68–77.

[354] R. Dow, "Additive Cellular Automata and Global Injectivity," *Physica D: Nonlinear Phenomena*, vol. 110, no. 1–2, pp. 67 – 91, 1997.

[355] A. Nobe and F. Yura, "On Reversibility of Cellular Automata with Periodic Boundary Conditions," *Journal of Physics A: Mathematical and General*, vol. 37, no. 22, p. 5789, 2004.

[356] T. Sato, K. Honda, H. Y. Lee, and Y. Kawahara, "A Classification of Triplet Local Rules with Inverse Image Sequences," in *Natural Computing*. Springer, 2009, pp. 167–178.

[357] K. Bhattacharjee and S. Das, "Reversibility of d-State Finite Cellular Automata," *Journal of Cellular Automata*, vol. 11, no. 2-3, pp. 213–245, 2016.

[358] O. Martin, A. M. Odlyzko, and S. Wolfram, "Algebraic Properties of Cellular Automata," *Communications in Mathematical Physics*, vol. 93, no. 2, pp. 219–258, 1984.

[359] K. Paul, "Theory and Application of $GF(2^p)$ Cellular Automata," Ph.D. dissertation, Bengal Engineering College (a Deemed University), India, 2002.

[360] T. Hansen and G. Mullen, "Primitive Polynomials Over Finite Fields," *Mathematics of Computation*, vol. 59, no. 200, pp. 639–643, 1992.

[361] C. Forbes, M. Evans, N. Hastings, and B. Peacock, *Statistical Distribution*. Wiley Publication, 2010.

[362] A. Dennunzio, E. Formenti, L. Margara, V. Montmirail, and S. Riva, "Solving Equations on Discrete Dynamical Systems," in *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*. Springer, 2019, pp. 119–132.

[363] E. Formenti, J.-C. Régin, and S. Riva, "MDDs Boost Equation Solving on Discrete Dynamical Systems," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2021, pp. 196–213.

[364] W. Meier and O. Staffelbach, "Analysis of Pseudo Random Sequences Generated by Cellular Automata," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1991, pp. 186–199.

[365] M. Tomassini, M. Sipper, M. Zolla, and M. Perrenoud, "Generating High-Quality Random Numbers in Parallel by Cellular Automata," *Future Generation Computer Systems*, vol. 16, no. 2-3, pp. 291–305, 1999.

[366] D. E. Knuth, *The Art of Computer Programming – Seminumerical Algorithms*, 3rd ed. Pearson Education, 2000, vol. 2.

[367] G. Marsaglia, "DIEHARD: A Battery of Tests of Randomness," in *http://stat.fsu.edu/˜ geo/diehard.html*, 1996.

[368] P. L'ecuyer and R. Simard, "TestU01: AC Library for Empirical Testing of Random Number Generators," *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 4, pp. 1–40, 2007.

[369] J. Soto, "Statistical Testing of Random Number Generators," in *Proceedings of the Twenty Second National Information Systems Security Conference*, 1999.

[370] R. McGrath *et al.*, "GNU C Library," https://www.gnu.org/software/libc/manual/html_node/Pseudo_002dRandom-Numbers.html#index-pseudo_002drandom-numbers.

[371] P. L'Ecuyer and R. Touzin, "Fast Combined Multiple Recursive Generators with Multipliers of the Form $a = \pm 2^q \pm 2^r$," in *Proceedings of the Thirty Second Conference on Winter Simulation*, 2000, pp. 683–689.

[372] M. E. O'Neill, "PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation," *ACM Transactions on Mathematical Software*, 2014.

[373] P. L'ecuyer, "Maximally Equidistributed Combined Tausworthe Generators," *Mathematics of Computation*, vol. 65, no. 213, pp. 203–213, 1996.

[374] P. L'Ecuyer, "Random Number Generators," 2017, http://www-labs.iro.umontreal.ca/~simul/rng/.

[375] G. Marsaglia, "Xorshift RNGs," *Journal of Statistical Software*, vol. 8, no. 14, pp. 1–6, 2003.

[376] S. Vigna, "An Experimental Exploration of Marsaglia's Xorshift Generators, Scrambled," *ACM Transactions on Mathematical Software*, vol. 42, no. 4, pp. 1–23, 2016.

[377] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[378] R. Xu and D. Wunsch, "Survey of Clustering Algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[379] H. Frigui and R. Krishnapuram, "A Robust Competitive Clustering Algorithm with Applications in Computer Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 450–465, 1999.

[380] X. Han, L. Quan, X. Xiong, M. Almeter, J. Xiang, and Y. Lan, "A Novel Data Clustering Algorithm Based on Modified Gravitational Search Algorithm," *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 1–7, 2017.

[381] S. I. Boushaki, N. Kamel, and O. Bendjeghaba, "A New Quantum Chaotic Cuckoo Search Algorithm for Data Clustering," *Expert Systems with Applications*, vol. 96, p. 358–372, 2018.

[382] M. Alswaitti, M. Albughdadi, and N. A. M. Isa, "Density-Based Particle Swarm Optimization Algorithm forData Clustering," *Expert Systems with Applications*, vol. 91, pp. 170–186, 2018.

[383] G. S. Rao and P. Sudhakar, "A New Adaptive Artificial Bee Colony (AABC) Technique in Cellular Automata Data Clustering," in *Smart Intelligent Computing and Applications*, 2020, pp. 1–12.

[384] H. Taib and A. Bahreininejad, "Data Clustering using Hybrid Water Cycle Algorithm and A Local Pattern Search Method," *Advances in Engineering Software*, vol. 153, p. 102961, 2021.

[385] C. M. Bishop *et al.*, *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[386] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[387] M.-P. Dubuisson and A. K. Jain, "A Modified Hausdorff Distance for Object Matching," in *Proceedings of Twelfeth International Conference on Pattern Recognition*, vol. 1. IEEE, 1994, pp. 566–568.

[388] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing Images Using the Hausdorff Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[389] D. Gusfield, "Algorithms on Stings, Trees, and Sequences: Computer Science and Computational Biology," *ACM Sigact News*, vol. 28, no. 4, pp. 41–60, 1997.

[390] D. Sankoff, "Time Warps, String Edits, and Macromolecules," *The Theory and Practice of Sequence Comparison, Reading*, 1983.

[391] K. C. Gowda and G. Krishna, "Agglomerative Clustering Using the Concept of Mutual Nearest Neighbourhood," *Pattern Recognition*, vol. 10, no. 2, pp. 105–112, 1978.

[392] R. R. Sokal, "Numerical Taxonomy," *Scientific American*, vol. 215, no. 6, pp. 106–117, 1966.

[393] B. King, "Step-Wise Clustering Procedures," *Journal of the American Statistical Association*, vol. 62, no. 317, pp. 86–101, 1967.

[394] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," *ACM Sigmod Record*, vol. 25, no. 2, pp. 103–114, 1996.

[395] S. Guha, R. Rastogi, and K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," *ACM Sigmod record*, vol. 27, no. 2, pp. 73–84, 1998.

[396] ——, "ROCK: A Robust Clustering Algorithm for Categorical Attributes," *Information Systems*, vol. 25, no. 5, pp. 345–366, 2000.

[397] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical Clustering using Dynamic Modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[398] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, 2009, vol. 344.

[399] J. Dopazo and J. M. Carazo, "Phylogenetic Reconstruction Using an Unsupervised Growing Neural Network That Adopts the Topology of A Phylogenetic Tree," *Journal of Molecular Evolution*, vol. 44, no. 2, pp. 226–233, 1997.

[400] M. R. Anderberg, *Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks.* Academic press, 2014, vol. 19.

[401] H.-S. Park and C.-H. Jun, "A Simple and Fast Algorithm for k-Medoids Clustering," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[402] R. T. Ng and J. Han, "CLARANS: A Method for Clustering Objects for Spatial Data Mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1003–1016, 2002.

[403] G. H. Ball and D. J. Hall, "A Clustering Technique for Summarizing Multivariate Data," *Behavioral Science*, vol. 12, no. 2, pp. 153–155, 1967.

[404] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[405] A. Hinneburg, D. A. Keim *et al.*, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," in *KDD*, vol. 98, 1998, pp. 58–65.

[406] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering Points To Identify the Clustering Structure," *ACM Sigmod Record*, vol. 28, no. 2, pp. 49–60, 1999.

[407] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[408] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A Distribution-Based Clustering Algorithm for Mining in Large SpatialDdatabases," in *Proceedings Fourteenth International Conference on Data Engineering.* IEEE, 1998, pp. 324–331.

[409] C. E. Rasmussen *et al.*, "The Infinite Gaussian Mixture Model," in *NIPS*, vol. 12. Citeseer, 1999, pp. 554–560.

[410] R. Sharan and R. Shamir, "CLICK: A Clustering Algorithm with Applications to Gene Expression Analysis," in *Proc Int Conf Intell Syst Mol Biol*, vol. 8, no. 307, 2000, p. 16.

[411] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering Gene Expression Patterns," *Journal of Computational Biology*, vol. 6, no. 3-4, pp. 281–297, 1999.

[412] V. Estivill-Castro and I. Lee, "Amoeba: Hierarchical Clustering Based on Spatial Proximity using Delaunay Diagram," in *Proceedings of the Ninth International Symposium on Spatial Data Handling. Beijing, China*, 2000, pp. 1–16.

[413] W. Wang, J. Yang, R. Muntz *et al.*, "STING: A Statistical Information Grid Approach to Spatial Data Mining," in *VLDB*, vol. 97. Citeseer, 1997, pp. 186–195.

[414] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, 1998, pp. 94–105.

[415] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A Multi-Resolution Clustering Approach for Very Large Lpatial Databases," in *VLDB*, vol. 98, 1998, pp. 428–439.

[416] D. H. Fisher, "Knowledge Acquisition via Incremental Conceptual Clustering," *Machine Learning*, vol. 2, no. 2, pp. 139–172, 1987.

[417] T. Kohonen, "The Self-Organizing Map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[418] G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for A Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, no. 1, pp. 54–115, 1987.

[419] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Slusters," *Journal of Cybernatics*, vol. 3, pp. 32–57, 1973.

[420] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* Springer Science & Business Media, 2013.

[421] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: Tthe Fuzzy *c*-Means Clustering Algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[422] R. N. Dave and K. Bhaswan, "Adaptive Fuzzy *c*-Shells Clustering and Detection of Ellipses," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 643–662, 1992.

[423] R. R. Yager and D. P. Filev, "Approximate Clustering via the Mountain Method," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 8, pp. 1279–1284, 1994.

[424] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear Component Analysis As a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[425] D. MacDonald and C. Fyfe, "The Kernel Self-Organising Map," in *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No. 00TH8516)*, vol. 1. IEEE, 2000, pp. 317–320.

[426] Z.-d. Wu, W.-x. Xie, and J.-p. Yu, "Fuzzy *c*-Means Clustering Algorithm Based on Kernel Method," in *Proceedings Fifth International Conference on Computational Intelligence and Multimedia Applications. ICCIMA 2003.* IEEE, 2003, pp. 49–54.

[427] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support Vector Clustering," *Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2001.

[428] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans, "Maximum Margin Clustering," *Advances in Neural Information Processing Systems*, vol. 17, pp. 1537–1544, 2004.

[429] B. Zhao, J. T. Kwok, and C. Zhang, "Multiple Kernel Clustering," in *Proceedings of the 2009 SIAM International Conference on Data Mining.* SIAM, 2009, pp. 638–649.

[430] S. C. Johnson, "Hierarchical Clustering Schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[431] A. K. Patnaik, P. K. Bhuyan, and K. K. Rao, "Divisive Analysis (DIANA) of Hierarchical Clustering and GPS Data for Level of Service Criteria of Urban Streets," *Alexandria Engineering Journal*, vol. 55, no. 1, pp. 407–418, 2016.

[432] B. Fritzke, "Growing Cell Structures—A Self-Organizing Network for Unsupervised and Supervised Learning," *Neural Networks*, vol. 7, no. 9, pp. 1441–1460, 1994.

[433] P. Hansen and B. Jaumard, "Cluster Analysis and Mathematical Programming," *Mathematical Programming*, vol. 79, no. 1, pp. 191–215, 1997.

[434] J. MacQueen *et al.*, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[435] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-Based Clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[436] E. Januzaj, H.-P. Kriegel, and M. Pfeifle, "Scalable Density-Based Distributed Clustering," in *European Conference on Principles of Data Mining and Knowledge Discovery.* Springer, 2004, pp. 231–244.

[437] H.-P. Kriegel and M. Pfeifle, "Density-Based Clustering of Uncertain Data," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005, pp. 672–677.

[438] Y. Chen and L. Tu, "Density-Based Clustering for Real-Time Stream Data," in *Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 133–142.

[439] D. Barbará and P. Chen, "Using the Fractal Dimension to Cluster Datasets," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 260–264.

[440] M.-S. Yang, "A Survey of Fuzzy Clustering," *Mathematical and Computer Modelling*, vol. 18, no. 11, pp. 1–16, 1993.

[441] A. Baraldi and P. Blonda, "A Survey of Fuzzy Clustering Algorithms for Pattern Recognition. I," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 6, pp. 778–785, 1999.

[442] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image recognition.* John Wiley & Sons, 1999.

[443] J. Handl and B. Meyer, "Ant-based and Swarm-based Clustering," *Swarm Intelligence*, vol. 1, no. 2, pp. 95–113, 2007.

[444] A. Abraham, S. Das, and S. Roy, "Swarm Intelligence Algorithms for Data Clustering," in *Soft Computing for Knowledge Discovery and Data Mining.* Springer, 2008, pp. 279–313.

[445] D. Van der Merwe and A. P. Engelbrecht, "Data clustering using Particle Swarm Optimization," in *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, vol. 1. IEEE, 2003, pp. 215–220.

[446] D. Karaboga and C. Ozturk, "A Novel Clustering Approach: Artificial Bee Colony (ABC) Algorithm," *Applied Soft Computing*, vol. 11, no. 1, pp. 652–657, 2011.

[447] E. D. Lumer and B. Faieta, "Diversity and Adaptation in Populations of Clustering Ants," in *Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3: From Animals to Animats 3*, 1994, pp. 501–508.

[448] D. Karaboga and B. Akay, "A Survey: Algorithms Simulating Bee Swarm Intelligence," *Artificial Intelligence Review*, vol. 31, no. 1-4, p. 61, 2009.

[449] R. Xu, J. Xu, and D. C. Wunsch, "A Comparison Study of Validity Indices on Swarm-Intelligence-Based Clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1243–1256, 2012.

[450] D. Horn and A. Gottlieb, "Algorithm for Data Clustering in Pattern Recognition Problems Based on Quantum Mechanics," *Physical Review Letters*, vol. 88, no. 1, p. 018702, 2001.

[451] ——, "The Method of Quantum Clustering," in *Nips*, vol. 14, 2001, pp. 769–776.

[452] G. W. Milligan and M. C. Cooper, "An Examination of Procedures for Determining the Number of Clusters in a Data set," *Psychometrika*, vol. 50, no. 2, pp. 159–179, 1985.

[453] U. Maulik and S. Bandyopadhyay, "Performance Evaluation of Some Clustering Algorithms and Validity Indices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1650–1654, 2002.

[454] S. Datta and S. Datta, "Comparisons and Validation of Statistical Clustering Techniques for Microarray Gene Expression Data," *Bioinformatics*, vol. 19, no. 4, pp. 459–466, 2003.

[455] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 224–227, 1979.

[456] J. C. Dunn, "Well Separated Clusters and Fuzzy Partitions," *Journal on Cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.

[457] T. Caliński and J. Harabasz, "A Dendrite Method for Cluster Analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[458] J. C. Dunn, "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[459] J. Handl, J. Knowles, and D. B. Kell, "Computational Cluster Validation in Post-Genomic Data Analysis," *Bioinformatics*, vol. 21, no. 15, pp. 3201–3212, 2005.

[460] X. L. Xie and G. Beni, "A Validity Measure for Fuzzy Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 841–847, 1991.

[461] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and Unsupervised Discretization of Continuous Features," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 194–202.

[462] V. Estivill-Castro, "Why so many clustering algorithms – A Position Paper," *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 1, pp. 65–75, 2002.