Greedy Strategies to Generate Maximal Length Cellular Automata

Dhawal Verma[1], Patatri Acharya[2], and Mentor: Sumit Adak[3]

[1]Indian Institute of Technology, Delhi, India
[2]Institute of Engineering and Management, Kolkata, India
[3]Technical University of Denmark, Denmark

September 2024

Abstract

This report provides an in-depth exploration of maximal length cellular automata (MLCA). It covers the fundamental principles of cellular automata (CAs), the specific characteristics of MLCA, and the mathematical foundations underlying their design. Key topics include characteristic polynomials, the analysis of maximality, boundary condition dependence, and the design of minimal cost MLCA. The report includes detailed examples, mathematical derivations, proofs, and diagrams to offer a comprehensive understanding of these complex systems.

Introduction

Cellular automata are a class of discrete, computational systems that consist of a grid of cells, each of which can be in one of a finite number of states. These cells update their states simultaneously based on a local rule that depends on the states of neighboring cells. Despite their simple rules, CAs can exhibit complex global behaviors, making them valuable for modeling natural phenomena and for computational applications.

Historical Context

The concept of cellular automaton was first introduced by John von Neumann and Stanislaw Ulam in the 1940s. They were interested in creating a self-replicating system, which led to the development of the first theoretical model of a CA. Stephen Wolfram's work in the 1980s popularized CAs, especially one-dimensional CAs, and demonstrated their capability to perform universal computation.

Importance and Applications of CAs

CAs are used in various fields such as biology (e.g., modeling cell growth), physics (e.g., simulating fluid dynamics), computer science (e.g., parallel computing and cryptography), and art (e.g., generating complex patterns). Their ability to model complex behaviors with simple rules makes them a versatile tool in both theoretical and practical contexts.

Basic Definitions

A cellular automaton is defined by a quadruple (L,S,M,R)

- L: A finite lattice of cells, typically in one or more dimensions.
- S: A finite set of states.
- M: A set of vectors defining the neighborhood of each cell.
- R: A local rule that determines the next state of a cell based on the states of its neighbors.

Key Components and Parameters

- Lattice Structure: The arrangement of cells can be one-dimensional, two-dimensional, or higher-dimensional.
- State Set: The possible states each cell can occupy, often represented as binary states (0 and 1).
- Neighborhood: The set of cells whose states influence a given cell's next state. Common neighborhoods include the von Neumann and Moore neighborhoods.
- Local Rule: The function that determines the next state of a cell based on the current states of its neighbors.

Elementary Cellular Automata (ECAs)

ECAs are the simplest form of CAs, consisting of a one-dimensional array of cells, each of which can be in one of two states (0 or 1). The next state of each cell depends on its current state and the state of its two immediate neighbors. There are 256 possible rules for ECAs, each corresponding to a unique local update rule.

Non-uniform or Hybrid CAs

Non-uniform CAs allow different cells to use different local rules. This flexibility makes them more suitable for modeling heterogeneous systems where different regions may follow different rules.

Examples and Applications:

- Biological Systems: Modeling cellular differentiation and development.
- Computational Systems: Designing fault-tolerant systems with varying local rules for robustness.

Global Properties and Hardware Implementations

Non-uniform CAs exhibit diverse global behaviors, from periodic and chaotic patterns to complex emergent phenomena. They can be implemented in hardware for parallel processing applications, leveraging their local interaction rules for efficient computation.

Maximal Length Cellular Automata

Maximal Length Cellular Automata (MLCA) are a specialized type of cellular automata (CA) that produce sequences cycling through all possible states except one in a single, uninterrupted loop. The significance of this property lies in its application to pseudorandom number generation, cryptography, and other fields where non-repetitive, long sequences are crucial.

A CA consists of a grid of cells, each of which can exist in one of a finite number of states. The state of each cell evolves over discrete time steps according to a local rule that depends on the states of its neighboring cells. The entire system updates simultaneously, creating a dynamic, evolving pattern. In an MLCA, the goal is to design the local rule in such a way that the CA explores the entire state space as efficiently as possible.

The design of an MLCA is intimately connected to the concept of primitive polynomials. A polynomial is termed primitive if its roots generate all non-zero elements of a finite field, implying that the associated CA can cycle through all possible states. For an n-cell CA, the goal is to find a primitive polynomial of degree n over the finite field F2, the field of binary numbers. This polynomial ensures that the CA achieves a cycle length of $2^n - 1$, which is the maximum length possible for an n-cell system.

A common example of MLCA design is the 3-cell CA using the polynomial *p(x)*=$x^3 + x + 1$ This CA cycles through all seven possible non-zero states, demonstrating the principle of maximality. Extending this to a 4-cell CA, the polynomial *p(x)*=$x^4 + x + 1$ generates a 15-state sequence, cycling through all non-zero states.

The practical implications of MLCA are far-reaching. In cryptographic systems, the ability to generate long, non-repetitive sequences is vital for ensuring security. Similarly, in pseudorandom number generation, MLCA ensures that the numbers generated do not exhibit repetitive patterns, thus enhancing randomness and unpredictability.

To design an MLCA, one must carefully select a characteristic polynomial that is both irreducible and primitive. The irreducibility ensures that the polynomial cannot be factored into lower-degree polynomials, maintaining the complexity of the CA's state transitions. Primitivity guarantees that the polynomial generates a maximal length sequence.

In summary, Maximal Length Cellular Automata are powerful tools in fields that require long, non-repetitive sequences. Their design relies on the careful selection of primitive polynomials, ensuring that the CA covers the entire state space in a single cycle. This property makes them invaluable in cryptographic applications and pseudorandom number generation.

Characteristic Polynomial

The characteristic polynomial of a cellular automaton (CA) is a mathematical construct that encapsulates the CA's update rule. It plays a crucial role in determining the behavior of the CA, particularly in the context of maximal length cellular automata (MLCA), where the goal is to achieve the longest possible cycle length.

In a linear CA, where the update rule is linear in nature, the characteristic polynomial is derived from the transition matrix that represents the state transitions of the CA. For an n-cell CA, this matrix is $n \times n$, and each entry in the matrix represents the influence of one cell's state on another. The characteristic polynomial *p(x)* is then defined as the determinant of xI-M, where I is the identity matrix and M is the transition matrix.

The characteristic polynomial's properties, particularly irreducibility and primitivity, determine whether the CA can achieve maximal length cycles. An irreducible polynomial is one that cannot be factored into polynomials of lower degrees over the given field. This property ensures that the CA's state transitions are complex and cannot be reduced to simpler patterns. For instance, the polynomial $x^3 + x + 1$ is irreducible over F2, ensuring that a CA governed by this polynomial exhibits complex state transitions.

Primitivity is a stronger condition than irreducibility. A primitive polynomial not only is irreducible but also generates the entire multiplicative group of the finite field. This means that the roots of the polynomial produce all non-zero elements of the field, which, in the context of a CA, translates to the CA cycling through all possible states except one. The polynomial $x^3 + x + 1$, which is both irreducible and primitive, is an example of a characteristic polynomial that leads to maximal length cycles in a 3-cell CA.

The role of the characteristic polynomial in CA behavior is profound. If the polynomial is reducible, the CA's state space can be decomposed into smaller cycles, leading to non-maximal cycles. If the polynomial is irreducible but not primitive, the CA may exhibit complex behavior but will still not achieve maximal length. Only when the polynomial is both irreducible and primitive does the CA achieve the desired maximal length cycle.

Characteristic polynomials also facilitate the analysis and design of CAs. By examining the roots and factors of the polynomial, one can predict the CA's behavior and adjust the design to meet specific requirements, such as maximal length or specific cycle structures.

In practical applications, especially in cryptography and pseudorandom number generation, the characteristic polynomial's properties are leveraged to design CAs that produce long, non-repetitive sequences. These sequences are critical for ensuring security and unpredictability, making the characteristic polynomial a central tool in the design of secure and efficient CAs.

In conclusion, the characteristic polynomial is a foundational element in the study and design of cellular automata. Its properties of irreducibility and primitivity are essential for achieving maximal length cycles, making it a critical tool in applications requiring long, non-repetitive sequences.

Analysis of Maximality

Maximality in cellular automata (CA) refers to the property of a CA where it cycles through all possible states except one in a single, uninterrupted loop. This property is essential in applications like pseudorandom number generation and cryptography, where long, non-repetitive sequences are crucial. The analysis of maximality involves determining the conditions under which a CA can achieve this property and understanding the mathematical foundations that govern it.

To analyze whether a given CA is maximal length, we begin by examining its characteristic polynomial. The characteristic polynomial encapsulates the CA's update rule and is derived from the transition matrix representing the state transitions of the CA. For an n-cell CA, this polynomial is a degree-n polynomial over the finite field F2, the field of binary numbers.

For a CA to be maximal length, its characteristic polynomial must satisfy two key conditions: irreducibility and primitivity. An irreducible polynomial cannot be factored into lower-degree polynomials, ensuring that the CA's state transitions are not reducible to simpler patterns. This irreducibility guarantees that the CA's behavior is sufficiently complex. However, irreducibility alone is not enough to ensure maximality.

Primitivity is a stronger condition. A primitive polynomial is one that not only is irreducible but also generates the entire multiplicative group of the finite field. This means that the polynomial's root generates all non-zero elements of the field, which in the context of a CA, translates to the CA cycling through all possible states (except the all-zero state) in a single cycle. For instance, in a 3-cell CA, the polynomial $x^3 + x + 1$ is both irreducible and primitive, ensuring that the CA achieves a cycle length of $2^3 - 1 = 7$, which is the maximum length possible for a 3-cell system.

The analysis of maximality involves verifying these conditions for the characteristic polynomial. The polynomial can be tested for irreducibility using factorization algorithms, and its primitivity can be confirmed by checking whether its root generates all non-zero elements of the field.

Mathematical proofs also play a crucial role in the analysis of maximality. For example, it can be proven that a CA governed by a primitive polynomial of degree n over F2 will exhibit a cycle length of $2^n - 1$. This result provides a clear criterion for maximality: the polynomial must be both irreducible and primitive.

In practical terms, the analysis of maximality is critical for the design of CAs in applications requiring long sequences without repetition. For example, in cryptography, a CA with a maximal length cycle can be used to generate pseudorandom numbers that are difficult to predict, enhancing security. Similarly, in simulations, maximal length CAs ensure that the system explores all possible states, providing comprehensive coverage of the state space.

In summary, the analysis of maximality in cellular automata involves verifying the irreducibility and primitivity of the characteristic polynomial. These conditions ensure that the CA achieves a maximal length cycle, making it suitable for applications requiring long, non-repetitive sequences.

Boundary Condition Dependence

Boundary conditions in cellular automata (CA) significantly influence the behavior of the system. These conditions determine how the states at the edges of the CA grid evolve over time, impacting the overall dynamics and the sequences generated by the CA. Understanding the dependence of CA behavior on boundary conditions is crucial, particularly in the context of maximal length cellular automata (MLCA), where the goal is to achieve the longest possible cycle length.

There are several types of boundary conditions commonly used in CAs, each with distinct effects on the system's behavior:

1. **Fixed Boundary Conditions**: In fixed boundary conditions, the states at the boundaries of the CA grid are held constant throughout the evolution of the system. These fixed states can introduce symmetry or fixed points into the CA, potentially reducing the system's overall complexity and cycle length. For example, in a binary CA, if the boundary states are fixed at 0, the CA may not explore the full state space, as the fixed boundaries can constrain the state transitions.

2. **Periodic Boundary Conditions**: Periodic boundary conditions treat the CA grid as a loop, where the cells at one edge of the grid are adjacent to the cells at the opposite edge. This "wrap-around" effect allows the CA to behave as if it is on a toroidal surface, eliminating edge effects. Periodic boundary conditions often preserve the complexity of the CA's behavior, enabling the system to explore a larger portion of the state space. In the context of MLCA, periodic boundaries are often preferred

because they maximize the length of the generated sequences by avoiding the constraints introduced by fixed boundaries.

3. **Null Boundary Conditions**: In null boundary conditions, the states at the boundaries are set to zero, effectively introducing a fixed boundary at 0. Like fixed boundaries, null boundaries can constrain the CA's behavior, leading to shorter cycles and less complex state transitions. However, in some cases, null boundaries can be useful for simplifying the analysis of the CA or for specific applications where edge effects need to be controlled.

The impact of boundary conditions on CA behavior can be profound. For example, in a CA with periodic boundary conditions, the system can achieve a maximal length cycle, exploring all possible states except one in a single cycle. In contrast, the same CA with fixed or null boundary conditions may exhibit shorter cycles, as the fixed boundaries can introduce repeating patterns or fixed points that limit the exploration of the state space.

The choice of boundary conditions is particularly important in the design of MLCA. To achieve maximal length, the CA's characteristic polynomial must be both irreducible and primitive, and the boundary conditions must not introduce constraints that prevent the system from exploring the entire state space. Periodic boundaries are often used in MLCA design because they allow the CA to behave as a closed system, preserving the maximality of the cycle length.

In summary, boundary conditions play a critical role in determining the behavior of cellular automata. Fixed, periodic, and null boundaries each have distinct effects on the system's dynamics, particularly in the context of maximal length cellular automata. Periodic boundaries are often preferred for MLCA because they maximize the length of the generated sequences by avoiding the constraints introduced by fixed or null boundaries.

Minimal Cost Maximal Length CA

Designing a minimal cost maximal length cellular automaton (CA) involves optimizing the CA's structure and update rules to achieve maximal length cycles with the least computational complexity and resource requirements. This optimization is crucial in applications where computational efficiency and resource constraints are important, such as in cryptography, parallel computing, and embedded systems.

A minimal cost maximal length CA is defined as a CA that achieves the longest possible cycle length (i.e., it cycles through all possible states except one) while minimizing the number of cells, the complexity of the update rule, and the computational resources required for implementation. The challenge in designing such a CA lies in balancing these competing factors: achieving maximal length cycles typically requires complex update rules, but minimizing cost requires simplifying these rules and reducing the number of cells.

Results

Given a rule vector configuration which can exhibit maximal length CA for a range of lattice sizes, it be modified by swapping the rules positions corresponding to the cells. All cells following rule 90(and rule 150) in first series will now follow rule 150(and rule 90) in its complementary series. A pair of rule vector which are complement of each other can be studied to create a strong strategy of generating primitive polynomials with greedy algorithms, reducing the search space significantly. A pair of rule vectors and its complement generally shows a pattern of its correlation. We have studied two such pairs: i) [90,90,150]* and [150,90,90]* upto 2000 lattices size ii) [90,150,90*,150] and [150,90,150*,90] upto 1000 lattices size and stumbled on these observations. First, we found strong correlations between the pairs for particular lattices size in term of showing the maximality. Second the number of observed maximal length CA up to an evaluation size are nearby for each pair. For example in pair 1, constituting series 1 and series 2, the number of MLCAs observed are 8 for both. Further, for lattice sizes of n = 3,72, 291, 459, 891, 903, 1323 both series produce MLCAs. Again in pair 2, for same lattice length size, 14 MLCAs are common in both series 3 and 4, namely for n = 7, 8, 31, 37, 42, 61, 108, 127, 337, 391, 589, 649, 751, 879. We tried to first study [150,90,90*] and its complement, but we got only 4 MLCAs upto evaluation of 4000 lattice length size, which show there are some rule vectors which is very sparse generator of primitive polynomials.

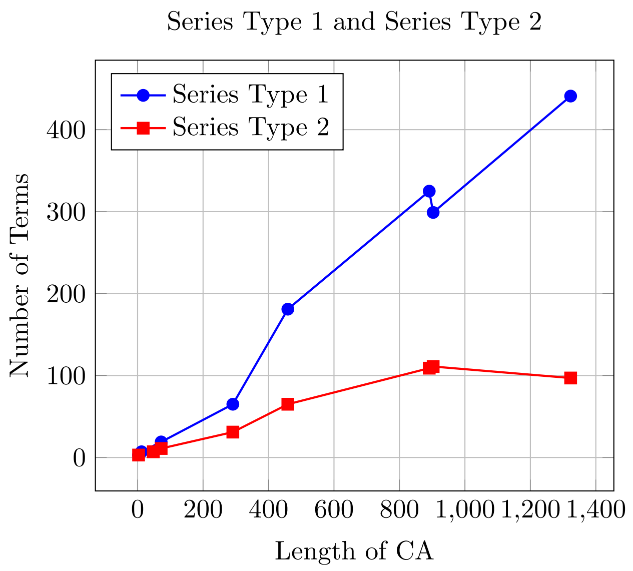

Figure 1: Plot of Series Type 1 and Series Type 2

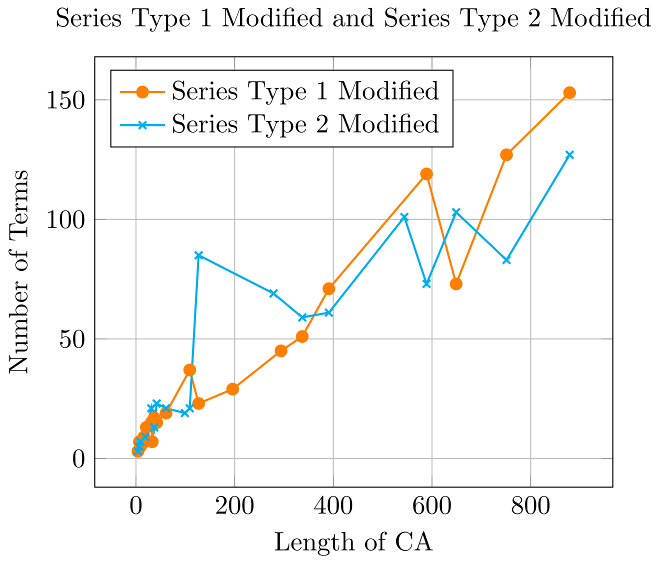

Figure 2: Plot of Series Type 1 Modified and Series Type 2 Modified

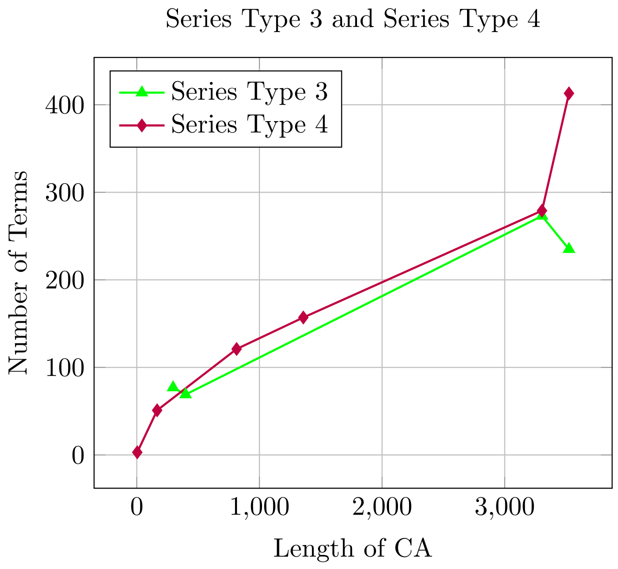

Figure 3: Plot of Series Type 3 and Series Type 4

Discussion

As division is a very costly execution, the determination of primitivity of a polynomial act as a bottleneck. So, for some lattices size, it was taking more than 3 hours to determine the primitivity even with the fastest algorithm. The correlations between lattice size suggest a pattern in the its distributions. A greedy algorithm can be constructed based on the pattern, which could be able to generate maximal length CA rule vector and thus primitive polynomial. Our future plan is to find a pattern in the rule vector configurations and the reason for such strong correlation.