

Research Article

Characterization of the Evolution of Nonlinear Uniform Cellular Automata in the Light of Deviant States

**Pabitra Pal Choudhury,¹ Sudhakar Sahoo,²
and Mithun Chakraborty³**

¹ *Applied Statistics Unit, Indian Statistical Institute, Kolkata 700108, India*

² *Department of Computer Science, Institute of Mathematics and Applications, Andharua, Bhubaneswar 751003, India*

³ *Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA*

Correspondence should be addressed to Sudhakar Sahoo, sudhakar.sahoo@gmail.com

Received 4 December 2010; Accepted 21 February 2011

Academic Editor: Marco Squassina

Copyright © 2011 Pabitra Pal Choudhury et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Dynamics of a nonlinear cellular automaton (CA) is, in general asymmetric, irregular, and unpredictable as opposed to that of a linear CA, which is highly systematic and tractable, primarily due to the presence of a matrix handle. In this paper, we present a novel technique of studying the properties of the State Transition Diagram of a nonlinear uniform one-dimensional cellular automaton in terms of its deviation from a suggested linear model. We have considered mainly elementary cellular automata with neighborhood of size three, and, in order to facilitate our analysis, we have classified the Boolean functions of three variables on the basis of number and position(s) of bit mismatch with linear rules. The concept of deviant and nondeviant states is introduced, and hence an algorithm is proposed for deducing the State Transition Diagram of a nonlinear CA rule from that of its nearest linear rule. A parameter called the proportion of deviant states is introduced, and its dependence on the length of the CA is studied for a particular class of nonlinear rules.

1. Introduction

The study of Boolean functions by G. Boole finds its application in various fields like electronics, computer hardware and software and is the base of digital electronics. On the other hand, the concept of Cellular Automata (CA) introduced by von Neumann [1] is a suitable tool for Complex Systems. CA rules have many real-life applications in almost all areas of science like physics, chemistry, mathematics, biology, engineering, and finance. A connection can be made between CA rules in different dimensions with n -variable Boolean functions [2, 3]. Out of 2^{2^n} Boolean functions of n variables, only 2^n are linear and the rest are

nonlinear [4]. In this way we get linear CAs and nonlinear CAs [2, 5, 6]. Likewise, we have *uniform* or *hybrid* CA, abbreviated as UCA and HCA, respectively, according to whether or not the same rule is applied to all the cells of the CA.

The *dynamic behavior* of any CA is visualized and studied in terms of either its *space-time pattern* or its *basin-of-attraction* field [7]. The latter is essentially a *graph*, which may or may not consist of disjoint subgraphs, and is commonly referred to as the *State Transition Diagram* or, in short, the STD of the CA. In the past, several attempts have been made [7–12] to study qualitatively and quantitatively the characteristics of UCA STDs in general (linear or nonlinear) in terms of parameters such as Z-parameter, λ -ratio, and λ -parameter. However, all these are techniques of *absolute characterization* of an STD (or, equivalently, a CA rule), that is, their main objective is to capture the graphical features of an STD as it *is* and they are not based on the comparison of a given STD with some known standard STD. Any linear UCA STD may be taken as a standard for comparison because all its essential features bear simple and well-known relationships with the fundamental properties such as rank, nullity, and determinant of the *state transition matrix* or *transformation matrix*, denoted by T , of the corresponding linear CA rule. With this in mind, we have made an attempt at the *relative characterization* of a particular set of nonlinear UCA STDs by first identifying the *nearest linear rule* of each such nonlinear rule, then considering the STD of the said nearest linear CA rule as a *linear model* for the nonlinear STD concerned and finally determining the nature and extent of departure of this nonlinear STD from the said linear model. But, first of all, we cluster the Boolean rules themselves into *classes* in order to separate out those rules that are readily amenable to the above analysis.

The remainder of this paper is organized in the following manner. In Section 2, some preliminary discussions on both Boolean functions and Cellular Automata are presented. In Section 3, some theoretical results are obtained using Hamming Distance (H.D) between Boolean functions. In [5, 6] Boolean functions are classified and subclassified according to their degree of nonlinearity and also the position of bit mismatch. Some of these ideas are also included in this section. Using these ideas we introduce the concepts of deviant and nondeviant states in Section 4. Finally Section 5 concludes the paper.

2. Basic Concepts

2.1. Boolean Functions: Their Representations, Naming Conventions, and Types

A *Boolean function* or *rule* $f(y_1, y_2, \dots, y_p)$ of p independent binary variables is defined as a mapping from $\{0, 1\}^p$ to $\{0, 1\}$. Any Boolean function can be represented either by a Truth Table or by one of several alternative algebraic forms such as D.N.F. (“Disjunctive Normal Form”), C.N.F. (“Conjunctive Normal Form”), and A.N.F. (“Algebraic Normal Form”). A Boolean rule is often identified with the output column of its Truth Table, which is a binary string of length 2^p for p independent variables. The decimal equivalent of this binary string, with the output of the first row being taken as the *least significant bit*, is called the *Wolfram’s number* (W) of the rule, and the rule is referred to as Rule W or \mathbf{f}_W . For example, the two-variable function $f(y_1, y_2)$ satisfying $f(0, 0) = 0$, $f(0, 1) = 1$, $f(1, 0) = 1$, $f(1, 1) = 1$ is considered identical to the bitstring **1110** and is designated as Rule 14 of two variables or $\mathbf{f}_{14}(y_1, y_2)$. Another naming scheme of Boolean functions is based on their Algebraic Normal Form which consists of AND and XOR operations only [4]. For one variable y_1 , the *complete algebraic normal form* is $(y_1 \oplus 1)$ which is also the A.N.F. of $f_1(y_1) = \bar{y}_1$. Proceeding recursively,

the complete A.N.F. of two variables y_1, y_2 is $[\{y_2(y_1 \oplus 1)\} \oplus (y_1 \oplus 1)] = (y_2y_1 \oplus y_2 \oplus y_1 \oplus 1) \equiv f_1(y_1, y_2) = \bar{y}_1\bar{y}_2$, and that of three variables is given by $[\{y_3(y_1y_2 \oplus y_2 \oplus y_1 \oplus 1)\} \oplus (y_1y_2 \oplus y_2 \oplus y_1 \oplus 1)] = (y_1y_2y_3 \oplus y_2y_3 \oplus y_3y_1 \oplus y_3 \oplus y_1y_2 \oplus y_2 \oplus y_1 \oplus 1) \equiv f_1(y_1, y_2, y_3) = \bar{y}_1\bar{y}_2\bar{y}_3$ and so on. In general, there are 2^p product terms in the complete A.N.F. of p -variables. The A.N.F. of any p -bit Boolean function can be generated by excluding one or more terms from the complete A.N.F. of p variables. If we denote presence of a term by 1 and absence by 0, we get a new binary string (of length 2^p) corresponding to every Boolean rule and the decimal equivalent of this bitstring is called the A.N.F. number of the rule concerned, for example, consider the function $f_{102}(y_1, y_2, y_3) = \bar{y}_1\bar{y}_2y_3 + \bar{y}_1y_2\bar{y}_3 + y_1\bar{y}_2y_3 + y_1y_2\bar{y}_3 = y_2 \oplus y_3$; comparing with the complete A.N.F. of three variables, the binary string corresponding to the A.N.F. of f_{102} is found to be **00010100**, so its A.N.F. number must be 20. Throughout this paper, Boolean functions have always been represented by their A.N.F. but have been referred to by their Wolfram numbers [2]. It is also worthwhile to mention here that if the Wolfram's number of a rule is even (odd), its ANF number is also even (odd), hence, without loss of generality, a rule may be referred to as "even-numbered" or "odd-numbered," as the case may be. Thus Rule 10 is an *even* rule while Rule 57 is an *odd* rule.

The generalized A.N.F. of a Boolean function of p variables y_1, y_2, \dots, y_p is given by

$$f(y_1, y_2, \dots, y_p) = a_0 \oplus \left(\sum_{i=1}^p a_i y_i \right) \oplus \left(\sum_{\substack{i=1 \\ i < j}}^p \sum_{j=1}^p a_{ij} y_i y_j \right) \oplus \left(\sum_{\substack{i=1 \\ i < j \\ j < k}}^p \sum_{j=1}^p \sum_{k=1}^p a_{ijk} y_i y_j y_k \right) \oplus \dots \oplus a_{123\dots p} y_1 y_2 \dots y_p, \tag{2.1}$$

where each of the coefficients $a_0, a_1, a_2, \dots, a_{12}, a_{13}, \dots, a_{123}, \dots, a_{123\dots p}$ may be either 0 or 1.

The number of variables in the highest product term with non-zero coefficient in the A.N.F. of a Boolean function is the algebraic degree of the function. A function of degree at most one is called an *affine function*. An affine function with the constant term equal to zero is called a *linear function*; all other functions are nonlinear functions. For p binary variables, there are 2^{p+1} affine functions of which 2^p are linear and the remaining 2^p are *logical complements* of these linear rules (they are nonlinear as each has its constant term equal to unity). For example, $f_0, f_{60}, f_{90}, f_{102}, f_{150}, f_{170}, f_{204}, f_{240}$ are the 8 linear functions of 3 variables; the remaining 248 functions are called nonlinear functions out of which the 8 functions $f_{255}, f_{195}, f_{165}, f_{153}, f_{105}, f_{85}, f_{51}, f_{15}$ are nonlinear affine functions.

2.2. Hamming Distance and Degree of Nonlinearity of a Boolean Rule

The *Hamming distance* (abbreviated as H.D. throughout this paper) between any two bit sequences of equal length is defined as the number of positions at which the bits differ in the two sequences.

The H.D. between two Boolean functions of p variables is defined as the H.D. between the binary sequences formed by the output columns of their Truth Tables (i.e., the p -bit binary equivalents of the rule numbers according to Wolfram's labeling convention). For example, let us take two Boolean functions of three variables namely Rule 34 and Rule 225. Their 8-bit binary representations are **00100010** and **11100001**, respectively. Clearly, these two strings differ from each other at 4 bit positions. Hence, the H.D. of Rule 34 from Rule 225 is 4. Equivalently, the H.D. between two rules f_1 and f_2 is given by the *weight* of the *sum mod 2* (XOR) of these two rules; in other words, it is the number of "1"s in the output column of the Truth Table of $f_1 \oplus f_2$.

The *degree of nonlinearity* of a p -variable Boolean function f is defined as the *minimum H.D.* of f from the set of all affine functions of p variables.

2.3. Terminology and Notation Pertaining to One-Dimensional Cellular Automata

In this paper, we will restrict ourselves to the study of a one-dimensional, binary cellular automaton (CA) of n cells (i.e., n bits) x_1, x_2, \dots, x_n , with local architecture [7]. The *global state* or simply *state* of a CA at any time instant t is represented as a vector $X^t = (x_1^t, x_2^t, \dots, x_n^t)$ where x_i^t denotes the bit in the i th cell x_i at time instant t . However, instead of expressing a state as a bitstring, we will frequently represent it by the decimal equivalent of the n -bit string with x_1 as the Most Significant Bit; for example, for a 4-bit CA, the state *1011* may be referred to as state **11** ($=1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3$).

The bit in the i th cell at the "next" time instant $t + 1$ is given by a *local mapping* denoted by f^i , say, which takes as its argument a vector of the bits (in proper order) at time instant t in the cells of a certain predefined *neighborhood* (of size p , say) of the i th cell. Thus, the size of the neighborhood is taken to be the same for each cell and may also be called the "number of variables" (which f^i takes as inputs).

Null Boundary (NB)

The left neighbor of x_1 and the right neighbor of x_n are taken as 0 each.

Periodic Boundary (PB)

x_n is taken as the left neighbor of x_1 and x_1 as the right neighbor of x_n .

A CA may be represented as a string of the rules applied to the cells in proper order, along with a specification of the boundary conditions. For example, $\langle \mathbf{103, 234, 90, 0} \rangle \text{NB}$ refers to the CA (x_1, x_2, x_3, x_4) , where $x_1^{t+1} = f_{103}(0, x_1^t, x_2^t)$, $x_2^{t+1} = f_{234}(x_1^t, x_2^t, x_3^t)$, $x_3^{t+1} = f_{90}(x_2^t, x_3^t, x_4^t)$, $x_4^{t+1} = f_0(x_3^t, x_4^t, 0)$.

If the "present state" of an n -bit CA (at time t) is X_t , its "next state" (at time $t + 1$), denoted by X^{t+1} , is in general given by the *global mapping* $F(X^t) = (f^1(lb^t, x_1^t, x_2^t), f^2(x_1^t, x_2^t, x_3^t), \dots, f^n(x_{n-1}^t, x_n^t, rb^t))$, where lb and rb denote, respectively, the left boundary of x_1 and right boundary of x_n .

If the rule applied to each cell of a CA is a linear Boolean function, the CA will be called a *Linear Cellular Automaton*, otherwise a *Nonlinear Cellular Automaton*, for example, $\langle 0, 60, 60, 204 \rangle \text{NB}$ is a linear CA while $\langle 31, 31, 31, 31 \rangle \text{NB}$ and $\langle 60, 90, 87, 123 \rangle \text{PB}$ are nonlinear CAs.

If the same Boolean function (rule) determines the "next" bit in each cell of a CA, the CA will be called a *Uniform Cellular Automaton (UCA)*, otherwise it will be called a *Hybrid*

Table 1: H.D.s of three-variable Boolean Functions from linear rules.

Rule no.	H.D. from Rule nos.							
	0	60	90	102	150	170	204	240
0	0	4	4	4	4	4	4	4
1	1	5	5	5	5	5	5	5
2	1	5	3	3	3	3	5	5
3	2	6	4	4	4	4	6	6
4	1	3	5	3	3	5	3	5
5	2	4	6	4	4	6	4	6
6	2	4	4	2	2	4	4	6
7	3	5	5	3	3	5	5	7
8	1	3	3	5	5	3	3	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
248	5	3	3	5	5	3	3	1
249	6	4	4	6	6	4	4	2
250	6	4	2	4	4	2	4	2
251	7	5	3	5	5	3	5	3
252	6	2	4	4	4	4	2	2
253	7	3	5	5	5	5	3	3
254	7	3	3	3	3	3	3	3
255	8	4	4	4	4	4	4	4

Cellular Automaton (HCA), for example, $\langle 135, 135, 135, 135 \rangle$ PB is a UCA, $\langle 0, 60, 72, 72 \rangle$ NB is a HCA.

For a UCA, the Boolean function applied to each cell will be called the *rule of the CA*. So for a UCA, we can obviously drop the superscript “*i*” from the local mapping f^i and simply denote it as f . for example. for the 4-bit CA $\langle 230, 230, 230, 230 \rangle$ PB, the rule of the CA is Rule 230 and the CA will be called the “Rule 230 CA” of 4 bits with periodic boundary conditions. Henceforth, we will use the following notation UCAnNB, UCAnPB, HCAnNB, and HCAnPB for an n -bit CA. For our purpose, we will be mostly interested in *elementary CA* defined by Wolfram [2] to be one-dimensional binary CA with a symmetrical neighborhood of size $p = 3$ for each cell so that $x_i^{t+1} = f^i(x_{i-1}^t, x_i^t, x_{i+1}^t)$, $i = 2, 3, \dots, n - 1$.

3. Studies on the Hamming Distances between Boolean Functions and Their Classification

In our paper [5], we proposed and proved a few theorems (and their corollaries) on the H.D.s between Boolean functions of any number of variables n . Next, we presented a chart of the H.D.s of all the 256 Boolean rules of three variables from the set of the 8 linear rules of three variables and, based on these observations, suggested a classification of three-variable Boolean functions. Here, we have a quick recapitulation.

3.1. Theorems on the H.D.s between Boolean Functions

Theorem 3.1. *If the H.D. of an n - variable Boolean function f from another rule g is m , then the H.D. of the complement of f from the same rule g is $(2^n - m)$.*

Corollary 3.2. *For any nonlinear rule of n variables, there exists at least one affine rule of n variables such that the H.D. between the two is smaller than or equal to 2^{n-1} .*

3.2. Classification of Boolean Rules of Three Variables Based on H.D.s from the Set of Linear Rules

In Table 1, we provide an incomplete chart of the H.D.s of three-variable Boolean functions from the set of linear rules and hence enlist a few interesting observations thereof.

Observations

- (1) H.D. between any two linear rules of 3 variables is $4 = 2^{3-1}$.
- (2) For any nonlinear rule of 3 variables, there exists at least one linear rule of 3 variables such that the H.D. between the two is smaller than or equal to 4.
- (3) If H.D. of a nonlinear rule of 3 variables from one of the 8 linear rules is even (odd), that from any other linear rule is also even (odd); for example, Rules 3 and 255 shown in Table 1.
- (4) If the H.D. of a nonlinear rule of 3 variables from a linear rule of 3 variables is m , say, then the H.D. of the complement of the said nonlinear rule from that linear rule is $(8 - m)$; for example, Rule 2 and Rule 253.
- (5) If the H.D. of a nonlinear rule of 3 variables from a linear rule of 3 variables is 8, then its H.D. from every other linear rule is 4. There are 8 such nonlinear rules, each of which is, evidently, the complement of one of the 8 linear rules.
- (6) In case the H.D. of a nonlinear rule of 3 variables from a linear rule is even and the said nonlinear rule is not the complement of a linear rule, we have either only one or exactly three linear rule(s) at an H.D. of 2 from the nonlinear rule under consideration. In that case, we have exactly three or only one linear rule(s), respectively, each at an H.D. of 6 from the nonlinear rule under consideration. The H.D. of the nonlinear rule in question from each of the remaining four linear rules is 4.

A Boolean rule of 3 variables is said to belong to Class m if m is the minimum possible H.D. of the nonlinear rule from any linear rule of 3 variables, that is, there exists at least one linear rule such that the H.D. of the rule under consideration from this linear rule is m and, if m' is the H.D. of the said rule from any other linear rule, then m' is larger than or equal to m . The classification is presented in Table 2.

Comments

- (1) The rules in Class 0 and Class 4 are the affine Boolean functions of 3 variables, Class 0 rules being linear rules and Class 4 rules being the logical complements of the linear rules. The degree of nonlinearity of each of these rules is 0.
- (2) Each rule in Class 3 has its complement in Class 1 (in the order in which the rules are arranged in Table 3, the i th rule in Class 3 is the complement of the $(65 - i)$ th rule in Class 1, $(i = 1, 2, \dots, 64)$; the degree of nonlinearity of each of the Class 1 and Class 3 rules is 1.
- (3) In the order in which the rules are presented in Table 3, the i th rule in Class 2 is the complement of the $(113 - i)$ th rule in Class 2, $i = 1, 2, \dots, 56$; the degree of nonlinearity of each Class 2 rule is 2.

Table 2: Classification of three-variable Boolean rules.

Name of the class	Rules in the class	Number of rules in the class
Class 0	0,60,90,102,150,170,204,240	8
Class 1	1,2,4,8,16,22,26,28,32,38,42,44,52,56,61,62,64,70,74,76,82,88,91,94,98,100,103,110,112,118,122,124,128,134,138,140,146,148,151,158,162,168,171,174,176,182,186,188,196,200,205,206,208,214,218,220,224,230,234,236,241,242,244,248	64
Class 2	3,5,6,9,10,12,17,18,20,23,24,27,29,30,33,34,36,39,40,43,45,46,48,53,54,57,58,63,65,66,68,71,72,75,77,78,80,83,86,89,92,95,96,99,101,106,108,111,113,114,116,119,120,123,125,126,129,130,132,135,136,139,141,142,144,147,149,154,156,159,160,163,166,169,172,175,177,178,180,183,184,187,189,190,192,197,198,201,202,207,209,210,212,215,216,219,221,222,225,226,228,231,232,235,237,238,243,245,246, 249,250,252	112
Class 3	7,11,13,14,19,21,25,31,35,37,41,47,49,50,55,59,67,69,73,79,81,84,87,93,97,104,107,109,115,117,121,127,131,133,137,143,145,152,155,157,161,164,167,173,179,181,185,191,193,194,199,203,211,213,217,223,227,229,233,239,247,251,253,254	64
Class 4	15,51,85,105,153,165,195,255	8

Table 3: Classification of three-variable Boolean rules.

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
22	0	0	0	1	0	1	1	0
150	1	0	0	1	0	1	1	0

3.3. Subclassification of the Classes of Three-Variable Boolean Rules Based on Position of Bit- Mismatch with the Nearest Linear Rule

Each Class 1 rule has exactly one linear rule at an H.D. of 1 from itself; that linear rule will be called its *nearest linear rule*. We express the Wolfram’s number of every Class 1 rule in its 8-bit binary form and compare it with the binary equivalent of the nearest linear rule. If mismatch occurs at bit position 2^q , $q = 0, 1, 2, \dots, 7$, the rule is said to belong to Subclass q of Class 1, denoted by $1 : q$; for example, nearest linear rule of Rule 22 is 150 as shown in Table 3.

Therefore, Rule 22 belongs to Subclass 7 of Class 1. Thus, there are 8 subclasses of Class 1, as shown in Table 4.

We observed that each Class 3 rule has exactly *three* nearest linear rules; so, it is not possible to subclassify them by the method adopted for Class 1 rules for there remains a confusion as to which of the three nearest linear rules to choose. But, since each rule in Class 3 has its complement in Class 1, the 64 rules in Class 3 can be subclassified into 8 subclasses of 8 rules each in the following manner.

Table 4: Subclassification of Class 1 Rules.

Subclass of Class 1	Wolfram's numbers of the rules included in the subclass
1:7	22, 42, 76, 112, 128, 188, 218, 230
1:6	26, 38, 64, 124, 140, 176, 214, 234
1:5	28, 32, 70, 122, 138, 182, 208, 236
1:4	16, 44, 74, 118, 134, 186, 220, 224
1:3	8, 52, 82, 110, 158, 162, 196, 248
1:2	4, 56, 94, 98, 146, 174, 200, 244
1:1	2, 62, 88, 100, 148, 172, 206, 242
1:0	1, 61, 91, 103, 151, 171, 205, 241

Table 5: Subclassification of Class 3 Rules.

Subclass of Class 3	Wolfram's numbers of the rules included in the subclass
3:7*	233,213,179,143,127,67,37,25
3:6*	229,217,191,131,115,79,41,21
3:5*	227,223,185,133,117,73,47,19
3:4*	239,211,181,137,121,69,35,31
3:3*	247,203,173,145,97,93,59,7
3:2*	251,199,161,157,109,81,55,11
3:1*	253,193,167,155,107,83,49,13
3:0*	254,194,164,152,104,84,50,14

If the complement of a Class 3 rule belongs to Subclass q of Class 1, then that Class 3 rule is said to belong to Subclass q^* of Class 3, denoted by $3:q^*$. The details are presented in Table 5.

For Class 2 rules, we observe the following.

- (1) Each of the 56 even-numbered rules in Class 2 is the complement of one of the 56 odd-numbered rules in Class 2.
- (2) Every odd rule in Class 2 is at Hamming distance of 2 from exactly one linear rule (and at an H.D. 6 from exactly three linear rules); this single linear rule may be called the *nearest linear rule* of the odd-numbered rule concerned; naturally, each even rule in Class 2 is at an H.D. of 2 from exactly three linear rules and at an H.D. of 6 from exactly one linear rule, and, hence, for an even Class 2 rule, *the* nearest linear rule *is not unique*, for example, rule 6 (Table 2).
- (3) As a linear rule is necessarily even numbered, the binary representation of any odd rule in Class 2 will definitely differ from that of its nearest linear rule at the bit position 2^0 (i.e., at the LSB) which is always 1 for an odd rule and 0 for an even rule. The bit position of the second mismatch will naturally not be the same for all odd-numbered rules.

Thus, the subclassification of the odd-numbered Class 2 rules could be based on the aforesaid bit-*position of the second mismatch* with the *nearest linear rule* and, the even-numbered Class 2 rules being the complements of these odd rules, their subclassification could be done in a manner similar to that in which the Class 3 rules have been sub-classified. For example.

Table 6: Subclassification of Class 2 Rules.

Subclass of Class 2	Wolfram's numbers of the rules included in the subclass
2:7	23,43,77,113,129,189,219,231
2:6	27,39,65,125,141,177,215,235
2:5	29,33,71,123,139,183,209,237
2:4	17,45,75,119,135,187,221,225
2:3	9,53,83,111,159,163,197,249
2:2	5,57,95,99,147,175,201,245
2:1	3,63,89,101,149,169,207,243
2:7*	232,212,178,142,126,66,36,24
2:6*	228,216,190,130,114,78,40,20
2:5*	226,222,184,132,116,72,46,18
2:4*	238,210,180,136,120,68,34,30
2:3*	246,202,172,144,96,92,58,6
2:2*	250,198,160,156,108,80,54,10
2:1*	252,192,166,154,106,86,48,12

The nearest linear rule of Rule 3 is 0. Therefore, Rule 3 belongs to Subclass 1 of Class 2, denoted by 2:1 and complement of Rule 3 = $255 - 3 = 252$. Thus, Rule 252 belongs to Subclass 1* of Class 2, denoted by 2:1* as shown in Table 6.

4. Characterization of Nonlinear UCA STDs in Terms of Deviation from Linearity

4.1. The State Transition Diagram of a UCA

The evolution of a CA can be completely described by a diagram in which each state is connected to its successor by a properly *directed line segment*. This diagram is called the State Transition Diagram (abbreviated as STD) of the CA. In other words, the STD of a CA is essentially a directed graph where each node represents one of the states of the CA and the *edges* signify transitions from one state to another. The STD of the UCA Rule 170 NB is shown in Figure 1.

As already mentioned the structure of a linear CA STD is very well behaved and has been exhaustively studied. But, unfortunately, even for a moderate number of input variables, the linear rules constitute practically a microscopic fraction of the great multitude of Boolean functions. The nonlinear rules are, however, at various degrees of nonlinearity. So the idea that those nonlinear rules that are *not so distant from linear rules* may be studied in terms of their similarity (or, equivalently, dissimilarity) with their closest linear rules appears quite appealing and viable, and we have tried to work on it.

Before we proceed, we would like to draw attention to the biological analogy of the relationship between a rule table and the corresponding UCA STD that has been suggested in the literature [7]. Any Boolean rule (or, more correctly, the output column of a Boolean rule table) is comparable to a *genotype* of an organism so that a change (complementation) of one or more bits of the rule may be viewed as a *genetic mutation* in this context. Consequently, the change in the topology of the corresponding CA STD brought about by such a bit complementation is analogous to the change in *morphology* introduced by genetic mutation.

Table 7: Comparison of Truth Tables of Rules 150 and 22.

Row	a	b	c	Rule 150	Rule 22
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	0
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	0

As such, the new rule (offspring) obtained from another rule (parent) is called a *mutant* of the latter. Thus, in our classification, Class 1 rules may be regarded as one-bit mutants of linear rules, Class 2 rules as two-bit mutants of linear, rules and so on. Of these, the one-bit mutants are of utmost importance because, quite understandably, they can be expected to exhibit a greater similarity with linear rules than any other rule. This is found to be true in practice, as will be elaborated in the subsequent sections.

4.2. The Concept of Deviant States

If we compare the STD of a UCA with a Class 1 nonlinear rule (all subclasses) or an odd-numbered nonlinear rule from Class 2 with that of a same-sized UCA with its nearest linear rule, we observe that a considerable number of states have identical successors in both the STDs while the successors of other states differ in the two STDs. For a given size (n) of CA, we define a state of a nonlinear UCA, to which a Class 1 rule or an odd Class 2 rule has been applied, as a *deviant state* for that nonlinear rule if the successor of the state under consideration in the S.T.D. of the nonlinear UCA taken is different from its successor in the STD of the nearest linear rule UCA (of the same size). For example, Rule 128 belongs to Class 1:7 and its nearest linear rule is 240. Comparison of their STDs for a $UCA4NB$ reveals that only the states **7, 14, 15** have different successors in the two STDs, so for Rule 128 $UCA4NB$, the deviant states are **7, 14, 15**.

It is evident from this discussion that the deviant states have been defined with respect to the *nearest linear rule* only (and not just any linear rule) because we can expect maximum similarity of the STD of a nonlinear rule with the STD of its nearest linear rule. For this same reason, deviant states have been defined for the above subclasses only; for other subclasses, the nearest linear rule is not uniquely defined.

4.3. An Algorithm for Deducing the Set of Deviant States

For a Class 1 rule, the deviant states for a given size of CA can be easily determined and, hence, the STD of such a nonlinear rule, say f , can be easily deduced from that of its nearest linear rule f_L by Algorithm 1.

Illustration

Let us take Rule 22 $UCA4NB$. Rule 22 belongs to Class 1:7 with Rule 150 as its nearest linear rule. The deduction of the deviant states and their successors and, hence, the derivation of

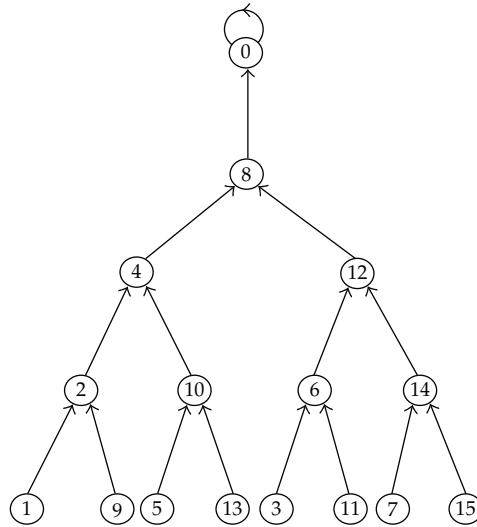


Figure 1: S.T.D. of $\langle 170, 170, 170, 170 \rangle$ NB.

Table 8: Determination of deviant states and their successors for Rule 22 UCA4NB.

Deviant states		Successor in Rule 150 UCA4NB		Successor in Rule 22 UCA4NB	
Binary	Decimal	Decimal	Binary	Decimal	Binary
0 1 1 1	7	10	1 0 1 0	1 0 0 0	8
1 1 1 0	14	5	0 1 0 1	0 0 0 1	1
1 1 1 1	15	6	0 1 1 0	0 0 0 0	0

the STD of $\langle 22, 22, 22, 22 \rangle$ NB from that of $\langle 150, 150, 150, 150 \rangle$ NB are shown with the help of Tables 7 and 8 and Figures 2 and 3.

4.4. Comments on Deviant States

- (1) Since the sequence to be searched may occur more than once in the same state and, in case of multiple occurrences, the repeating sequences may or may not overlap, it is clear that the task of identifying the deviant states and deducing their successors by the bit-toggling method becomes increasingly difficult with increasing length of CA.
- (2) Determination of the deviant states for odd Class 2 rules can be similarly accomplished by first locating the two positions of bit-mismatch and then searching for *at least one* of the two corresponding 3-bit combinations in the CA state. We could also proceed in two steps by considering one mismatch at a time. Obviously, the task is more difficult than that for the Class 1 case.
- (3) The set of deviant states for a given f depends strongly on the number of bits in the CA as well as on the boundary conditions (null/periodic); for example, for $\langle 22, 22, 22, 22 \rangle$ NB, the set of deviant states is $\{7, 14, 15\}$ whereas that for $\langle 22, 22, 22, 22 \rangle$ PB is $\{7, 11, 13, 14, 15\}$.

- (1) Locate the position of bit-mismatch in the Truth Table output column between f and f_L (which is identical to the subclass number of f).
- (2) Write down the 3-bit input combination, say $a^*b^*c^*$ corresponding to that row of the Truth Table where the said mismatch occurs (which is nothing but the 3-bit binary representation of the subclass number of f); evidently, it is only for this input sequence $a^*b^*c^*$ that f and f_L give different (i.e., complementary) outputs, their output for every other 3-bit input combination being identical. Consequently, a state of a three-neighborhood UCA of any length n will give different successors when acted upon by f and f_L if and only if the state contains the sequence $a^*b^*c^*$.
- (3) Search the space of all 2^n possible states of the UCA for states each of which contains the sequence (neighborhood) $a^*b^*c^*$ at least once. However, while looking for the neighborhood, $a^*b^*c^*$ we must properly take into account the *boundary values* for the two terminal bits of the CA.
- (4) In each deviant state, mark the position of that bit which, along with its immediate left and right neighbours, forms the sequence $a^*b^*c^*$. As the same sequence may occur more than once in state, we may get more than one such bit-positions in a state. Let these marked positions be p_1, p_2 and so forth.
- (5) In the STD of the nearest linear rule, locate the successors of the *deviant states*; in the binary representation of each such successor state, mark the bit-positions p_1, p_2 and so forth. (which were marked in the binary representation of each deviant state in step (4)) and toggle the bits at these positions; the resulting bit string will give us the successor of the deviant state in the STD of the nonlinear rule in question. Accordingly, make necessary changes in the diagram.
- (6) Leave the successors of the nondeviant states unchanged.

Algorithm 1

- (4) For given CA length and boundary conditions, all the rules in a subclass have identical *deviant states* because bit-mismatch with the respective nearest linear rules occurs at the same bit-position but the set of *deviant states*, in general, varies from one subclass to another.
- (5) For the periodic boundary case, the bits in any state may be supposed to be arranged in a ring or circle so that it is somewhat less difficult to identify the deviant states because such states will occur in *rotationally equivalent sets* [7]; for example, for $\langle 22, 22, 22, 22 \rangle \text{PB}$, $(7)_{10} \equiv (01\bar{1}1)_2$ is clearly a deviant state and $(14)_{10} \equiv (1\bar{1}10)_2$, $(13)_{10} \equiv (\bar{1}101)_2$, $(11)_{10} \equiv (10\bar{1}1)_2$ are rotationally equivalent to 7, so we can immediately declare them deviant states. It is also interesting to note that it is sufficient to deduce the successor of any one state from this set by the bit-toggling method because their successors are also rotationally equivalent and hence occupy equivalent positions in the STD.

4.5. The Significance of the Proportion of Deviant States in an STD

Let us define a quantity P_{ds} , called the proportion of deviant states for an n -bit nonlinear UCA, as $P_{ds} = N_{ds}/2^n$, where N_{ds} = number of deviant states of the UCA considered and 2^n is the total number of states of the UCA.

The rationale underlying the introduction of the concept of deviant states is that the behavior of the deviant states embodies the effect of the bit-mismatch (es) between f and f_L (or, in other words, the nature and extent of nonlinearity of f) on the topology of the STD of the UCA of f —if, for the type of nonlinear UCA considered (Class 1 and Class 2 odd), we start from a seed which is a nondeviant state, the system evolution follows the path of the nearest

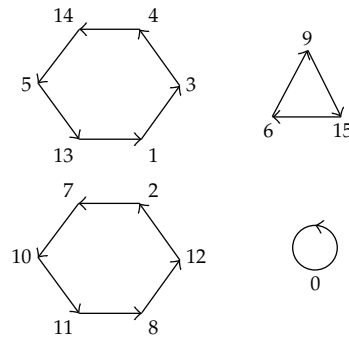


Figure 2: STD of $\langle 150, 150, 150, 150 \rangle$ NB.

linear rule until and unless a deviant state is encountered; when a deviant state is actually reached, the evolution deviates from the linear path, reaches some state, and henceforward resumes the linear path till the next deviant state is reached. As such, the *proportion of deviant states* in a CA may provide us with *some measure of the deviation from linearity* of a nonlinear rule (with respect to CA state transitions). This discussion reveals that it is not just the number of bit-mismatches (i.e., degree of nonlinearity) but also the *actual positions of bit-mismatch*, which determine how much a nonlinear UCA STD differs both qualitatively and quantitatively from the closest linear STD. With this in mind, we investigated the dependence of P_{ds} on the length of the CA for all Class 1 subclasses for the periodic boundary (PB) case, and the results are presented in Table 9 and Figure 4.

Comments on Figure 4

- (1) For any given length of CA (PB), $P_{ds}(1:7) = P_{ds}(1:0)$ as the neighborhoods **111** and **000** exhibit *complementary symmetry* [7], $P_{ds}(1:5) = P_{ds}(1:2)$ as the neighborhoods **101** and **010** also exhibit *complementary symmetry*, $P_{ds}(1:6) = P_{ds}(1:3) = P_{ds}(1:4) = P_{ds}(1:1)$ as the neighborhoods **110** and **011** exhibit *reflection symmetry* [7], **011** and **100** show *complementary symmetry*, and **100** and **001** show *reflection symmetry* again.
- (2) As the length of CA (PB) increases, the proportion of deviant states, say P_{ds} , tends towards unity for any subclass of Class 1, which means that the effect of nonlinearity becomes more pronounced but following inequality is always maintained: $P_{ds}(1:7, 1:0) < P_{ds}(1:5, 1:2) < P_{ds}(1:6, 1:3, 1:4, 1:1)$ with the difference between the last two becoming significantly smaller than that between the first two.

5. Conclusion and Future Research Directions

In this paper, we have identified a group of elementary nonlinear uniform cellular automata (Class 1 and Class 2 odd), which can be studied in terms of deviation from linearity with relative ease, and have suggested a new way of looking at their STDs by introducing the concept of deviant states. Of the remaining Boolean rules, Class 3 is the complement of Class 1 whereas even Class 2 rules are complements of odd Class 2 rules. Hence the last two sets of Boolean rules (Class 3 and Class 2 even) may be expected to bear the same relationship with nonlinear affine rules (Class 4) as do the aforesaid two sets (Class 1 and Class 2 odd) with

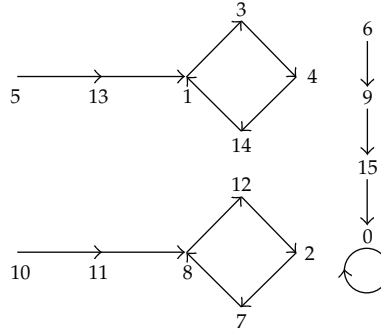


Figure 3: STD of $(22,22,22,22)NB$.

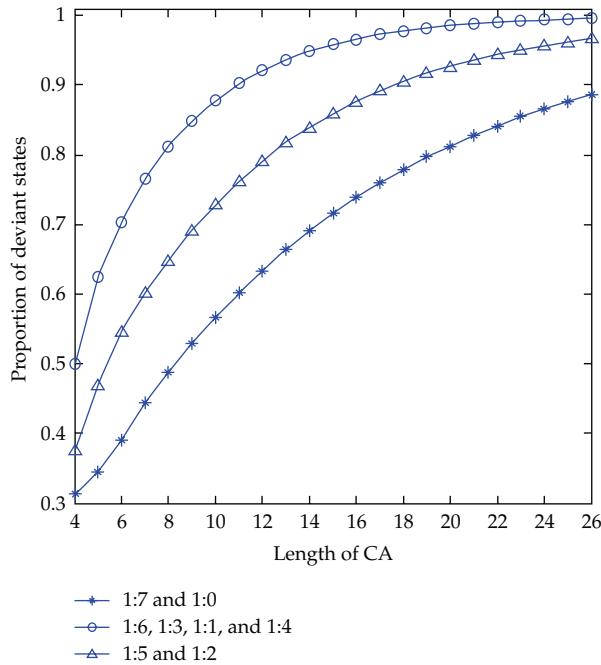


Figure 4: Plot of proportion of deviant states versus length of CA (periodic bound) for various subclasses of Class 1.

the linear rules. As such, it would apparently be wiser to classify Boolean functions on the basis of distance from the set of affine rules (i.e., degree of nonlinearity) rather than the set of linear rules and, in that case, the entire set of affine rules (linear rules and their complements) must be taken as the standard. We are exploring this possibility. Although the idea is still in an incipient state, it appears quite promising.

Another of our current efforts is aimed at deducing a possible formula for the number of deviant states in terms of the length n of the CA for a given nonlinear rule and given boundary conditions.

Moreover, we have considered only 3-neighborhood 1D UCA. We are investigating how far this idea could be carried forward to the cases with larger neighborhoods. From

Table 9: Table showing the variation in the number of deviant states for a 1:7 rule with CA length.

No. of cells in the CA (n)	No. of states (2^n)	No. of deviant states
4	16	5
5	32	11
6	64	25
7	128	57
8	256	125
9	512	271
10	1024	581
11	2048	1233
12	4096	2597
13	8192	5435
14	16384	11313
15	32768	23441
16	65536	48381
17	131072	99519
18	262144	204109
19	524288	417545
20	1048576	852245
21	2097152	1736043
22	4194304	3530121
23	8388608	7166985
24	16777216	14530301
25	33554432	29421711
26	67108864	59507605

the study presented in this paper, it appears that even if the neighborhood size (number of independent variables) increases, we can still possibly isolate a set of *mutants* of linear rules, which can be characterized in terms of deviant states. It is germane to mention here that, as already pointed out in [7], one-bit mutation amounts to $(1/2^3 \times 100)\% = 12.5\%$ change for a 3-variable function, to $(1/2^5 \times 100)\% = 3.125\%$ for a 5-variable function, and so on. In other words, the effect of mutation of a CA rule on a CA evolution (and thus the deviation of the CA STD from linearity) decreases as the size of the neighborhood increases. So, for a given length of CA and a given nonlinear rule, the proportion of deviant states must decrease with increasing size of neighborhood.

References

- [1] J. von Neumann, *The Theory of Self-Reproducing Automata*, A. W. Burks, Ed., Univ. of Illinois Press, Urbana, Ill, USA, 1966.
- [2] S. Wolfram, Ed., *Theory and Applications of Cellular Automata*, vol. 1 of *Advanced Series on Complex Systems*, World Scientific, Singapore, 1986.
- [3] S. Wolfram, *A New Kind of Science*, Wolfram Media, Champaign, Ill, USA, 2002.
- [4] I. Wegener, *The Complexity of Boolean Functions*, Wiley-Teubner Series in Computer Science, John Wiley & Sons, Chichester, UK, 1987.
- [5] P. P. Choudhury, S. Sahoo, M. Chakraborty, S. K. Bhandari, and A. Pal, "Investigation of the global dynamics of cellular automata using Boolean derivatives," *Computers & Mathematics with Applications*, vol. 57, no. 8, pp. 1337–1351, 2009.

- [6] S. Sahoo, P. P. Choudhury, M. Chakraborty, and B. K. Nayak, "Characterization of any non-linear Boolean function using a set of linear operators," *Journal of Orissa Mathematical Society*, vol. 29, no. 1-2, pp. 111–133, 2010.
- [7] A. Wuensche and M. Lesser, *The Global Dynamics of Cellular Automata, An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*, vol. 1 of *Santa Fe Institute Studies in the Sciences of Complexity*, Addison Wesley, 1992.
- [8] C. Moore, "Predicting nonlinear cellular automata quickly by decomposing them into linear ones," *Physica D*, vol. 111, no. 1–4, pp. 27–41, 1998.
- [9] S. Takakaju, "Non-linear cellular automata with group structure," *IEIC Technical Report*, vol. 99, no. 33, pp. 37–44, 1999.
- [10] R. Smolensky, "Algebraic methods in the theory of lower bounds for boolean circuit complexity," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, pp. 77–82, 1987.
- [11] B. H. Voorhees, *Computational Analysis of One-Dimensional Cellular Automata*, vol. 15 of *World Scientific Series on Nonlinear Science. Series A: Monographs and Treatises*, World Scientific, River Edge, NJ, USA, 1996.
- [12] B. Voorhees, "Predecessors of cellular automata states. II. Pre-images of finite sequences," *Physica D*, vol. 73, no. 1-2, pp. 136–151, 1994.