Computers and Mathematics with Applications 57 (2009) 1337–1351

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Investigation of the global dynamics of cellular automata using Boolean derivatives

Pabitra Pal Choudhury [a,*], Sudhakar Sahoo [b], Mithun Chakraborty [c], Subir Kumar Bhandari [d], Amita Pal [d]

[a] *Applied Statistics Unit, Indian Statistical Institute, Kolkata, 700108, India*

[b] *Department of CSEA, Silicon Inst. of Tech., Silicon Hills, Patia, Bhubaneswar-751024, India*

[c] *Department of Electronics and Telecommunication Engg., Jadavpur University, Kolkata-700032, India*

[d] *Bayesian and Interdisciplinary Research Unit, Indian Statistical Institute, Kolkata, 700108, India*

ARTICLE INFO

Article history:
Received 30 May 2008
Received in revised form 21 October 2008
Accepted 8 November 2008

Keywords:
Boolean functions
Linear and affine functions
Wolfram's naming scheme
Algebraic normal form
Error function
Jacobian matrix
State transition diagram
Modified Jacobian matrix

ABSTRACT

Global dynamics of a non-linear Cellular Automaton (CA), is, in general irregular, asymmetric and unpredictable as opposed to that of a linear CA, which is highly systematic and tractable. In this paper, efforts have been made to systematize non-linear CA evolutions in the light of Boolean derivatives and Jacobian matrices. A few new theorems on Hamming Distance between Boolean functions as well as on Jacobian matrices of cellular automata are proposed and proved. Moreover, a classification of Boolean functions based on the nature of deviation from linearity has been suggested with a view to grouping them together to classes/subclasses such that the members of a class/subclass satisfy certain similar properties. Next, an error vector, which cannot be captured by the Jacobian matrix, is identified and systematically classified. This leads us to the concept of modified Jacobian matrix whereby a quasi-affine representation of a non-linear cellular automaton is introduced.

1. Introduction

Cellular Automata (CA) introduced by von Neumann [1] is a new kind of science [2] to handle Complexity Theory in various disciplines. CA rules have enumerable applications in almost all area of science like Physics, Chemistry, Mathematics, Biology, Engineering, and Finance etc. These applications and further applications in future, needs an in-depth study of CA rules so that one can efficiently use these rules to get some interesting results.

Cellular Automata rules in different dimensions can be realized on using Boolean functions of n variables. Out of 2^{2^n} number of Boolean functions we have 2^n are linear and the rest are non-linear. This way we get linear CA and non-linear CA [3].

The State Transition Diagrams (STDs) of all linear CA are symmetric in representation and have a linear handle. A single matrix can represent a linear function for any input string [4,5]. The non-linear functions on the other hand are non-uniform and asymmetrical in representation in the state transition diagrams. No single matrix can represent a non-linear function for any input string. This paper classifies the set of all 2^{2^n} CA rules into different groups, based on their Jacobian matrix or modified Jacobian matrix (newly introduced in this paper) so that the rules in a group posses the similar STDs.

* Corresponding author.
E-mail addresses: pabitrapalchoudhury@gmail.com (P.P. Choudhury), sudhakar.sahoo@gmail.com (S. Sahoo), mithun.chakra108@gmail.com (M. Chakraborty), subir@isical.ac.in (S.K. Bhandari), pamita@isical.ac.in (A. Pal).

0898-1221/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2008.11.012

In Section 2, some preliminary discussions on both Boolean functions and Cellular Automata are discussed. In Section 3, some theoretical results are obtained using Hamming Distance (H.D.) between Boolean functions. Further, several results are proved in Section 4 on using Boolean derivatives which is the main thrust of this paper. In Section 5, Boolean functions are classified and sub-classified according to their degree of non-linearity and also the position of bit-mismatch. The importance of Jacobian matrix in the context of the evolution of CA is shown in Section 6. In Section 7 a concept of modified Jacobian matrix is introduced for any even-numbered rules and Section 8 concludes the paper.

2. Literature review

2.1. Algebraic representation and nomenclature of Boolean functions

Let $\mathbb{B}$ denote the set {0,1}; then $(\mathbb{B}, \oplus, \bullet)$ is the well-known *Galois field modulo* 2 or $GF(2)$ where "$\oplus$" denotes *addition modulo* 2 (logical Exclusive-OR) and "$\bullet$" denotes *multiplication modulo* 2 (logical AND).

Any function or rule (mapping) $f : \mathbb{B}^n \to \mathbb{B}$ is called a Boolean function of n binary variables, which may be written as $f(X)$ where $X = (x_1, x_2, \ldots, x_n)$, is the *input vector*, $x_i \in \mathbb{B} \forall i = 1, 2, \ldots, n$. The number of all possible Boolean functions of n variables is 2^{2^n}.

Any Boolean function is uniquely described by its *Truth Table* and may be identified with the string of bits in the output column of its Truth Table from the bottom upwards; e.g. if, for a two-variable Boolean rule f, $f(0, 0) = 1, f(0, 1) = 0$, $f(1, 0) = 0$ and $f(1, 1) = 1$, then the rule may be denoted by 1001 or by its decimal equivalent 9 (Rule 9), the latter being the *label* or *number* of the rule according to *Wolfram's naming scheme* [3].

Algebraic Normal Form (A.N.F.) also known as Ring Sum Expansion (RSE) [6] of a Boolean function is an algebraic representation of the function in terms of XOR and AND operations only. The *generalized A.N.F.* for one variable a is given by $a \oplus 1$, that for two variables (a, b) is $[(a \oplus 1).b] \oplus (a \oplus 1) = ab \oplus b \oplus a \oplus 1$, that for three variables (a, b, c) is $[(ab \oplus b \oplus a \oplus 1).c] \oplus (ab \oplus b \oplus a \oplus 1) = abc \oplus bc \oplus ca \oplus c \oplus ab \oplus b \oplus a \oplus 1$ and so on. Any A.N.F. of a given number of variables may be generated by taking one or more of the product-terms in the corresponding *generalized A.N.F.* If each term included is replaced by '1' and each term excluded by '0', then we shall get a bit-string whose decimal equivalent is the number of the rule concerned according to the *A.N.F. naming scheme*; e.g. $f(a, b) = ab \oplus b$ has A.N.F. number equal to decimal $(1100) = 12$.

Throughout this paper, Boolean functions have always been represented by their algebraic normal forms but have been referred to by their Wolfram numbers. Rule $\boldsymbol{W}$ may be sometimes denoted as $f_{\boldsymbol{W}}$. For example, for the case of 3 binary variables, Rule 150 or $f_{150}(a, b, c)$ denotes $f(a, b, c) = a \oplus b \oplus c$.

If Wolfram's number of a rule is even (odd), its A.N.F number is also even (odd), hence, without loss of generality, a rule may be referred to as "even-numbered" rule or "odd-numbered", as the case may be. Thus Rule 120 (A.N.F. no. = 66) is an *even* rule while Rule 37 (A.N.F. no. = 147) is an *odd* rule.

2.2. Types of Boolean functions

The functions $f(X) = 0$ (Rule 0) and $f(X) = 1$ (Rule $2^n - 1$) are *constant functions*; all other functions may be called *proper functions*.

A Boolean function of algebraic degree at most unity is called an *affine Boolean function*, the general form for n variables being

$$f_{\text{affine}}(X) = k_n x_n \oplus k_{n-1} x_{n-1} \oplus \cdots \oplus k_2 x_2 \oplus k_1 x_1 \oplus k_0 \quad \text{where } k_i \in \mathbb{B} \ \forall\ i \in \{0, 1, 2, \ldots.., n\}.$$

If the constant term of an affine function is zero, i.e. $\mathbf{k}_0 = 0$, then the function is called a *linear Boolean function*. The general A.N.F. of a linear rule must be an *even*-numbered rule. All rules other than the linear ones are called *non-linear rules*.

If $\mathbf{k}_0 = 1$, then the affine function has the form $f_{\text{affine}}(X) = f_{\text{linear}}(X) \oplus 1$ which is the Boolean complement of some linear rule. Thus, the affine Boolean functions of any number of variables are either linear rules or their complements.

For n binary variables, the total number of affine Boolean functions is 2^{n+1} while that of linear rules is 2^n. For instance, the 16 affine Boolean functions of 3 variables are **0, 60, 90, 102, 150, 170, 204, 240, 15, 51, 85, 105, 153, 165, 195, 255**, of which the first 8 are linear rules.

A Boolean function which possesses an equal number of '1's and '0's in the output column of its Truth Table is called a *balanced Boolean function*. It is known that all linear rules are balanced.

2.3. Terminology and notation pertaining to one-dimensional cellular automata

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

Table 2.3.1
Notation used in the paper.

Uniform/ Hybrid	Boundary conditions	
	Null	Periodic
Uniform	**UCA*n*NB**	**UCA*n*PB**
Hybrid	**HCA*n*NB**	**HCA*n*PB**

The bit in the ith 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 pre-defined *neighborhood* (of size p, say) of the ith 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. e.g. $\langle$**103, 234, 90, 0**$\rangle$***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* $F(X^t) = (f^1(lb^t, x_1^t, x_2^t), f^2(x_1^t, x_2^t, x_3^t), \ldots, f^n(x_{n-1}^t, x_n^t, rb^t))$, where *lb* and *rb* denote respectively the left boundary of x_1 and the 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 **non-linear Cellular Automaton**, e.g. $\langle 0, 60, 60, 204\rangle$NB is a linear CA while $\langle 31, 31, 31, 31\rangle$NB and $\langle 60, 90, 87, 123\rangle$PB are non-linear 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 Cellular Automaton (HCA)**, e.g. $\langle 135, 135, 135, 135\rangle$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. e.g. 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 shall use the following notation, presented in Table 2.3.1 for an n-bit CA:

For our purpose, we shall be mostly interested in *elementary CA* defined by Wolfram [3] to be one-dimensional binary CA with a symmetrical neighborhood of size $p = 3$ for each cell so that $x_i^{t+1} = f^i(x_{i-1}^t, x_i^t, x_{i+1}^t)$, $i = 2, 3, \ldots, n-1$.

2.4. Boolean derivatives and Jacobian matrix

The *first-order partial Boolean derivative* [4] of a Boolean function $f(x_1, x_2, \ldots, x_n)$ with respect to x_j, $j = 1, 2, \ldots, n$ is defined as $\partial f/\partial x_j = f(x_1, x_2, \ldots, x_j, \ldots, x_n) \oplus f(x_1, x_2, \ldots, \bar{x}_j, \ldots, x_n)$ where $\bar{x}_j$ is the Boolean complement of x_j.

The *gradient* of a Boolean function $f(x_1, x_2, \ldots, x_n)$, denoted by grad(f) is defined as the vector of the n first-order partial Boolean derivatives of the function with respect to the n input variables *in the proper order*, i.e. grad$(f) = [\partial f/\partial x_1 \partial f/\partial x_2 \ldots \partial f/\partial x_n]$.

The Jacobian matrix of an n-bit one-dimensional CA is defined as an $n \times n$ binary matrix, denoted by J, whose (i, j)th entry is $J_{ij} = \partial f^i/\partial x_j^t$ $\forall i \in \{1, 2, 3, \ldots, n\}, \forall j \in \{1, 2, 3, \ldots, n\}$. Under the assumption $p = 3$, the Jacobian matrix is a tri-diagonal matrix, except for the two *off-diagonal corner elements* in the periodic-boundary case.

In [7] the authors derived both Taylor series and MacLaurins series expansion of any Boolean function. Using MacLaurins series expansion, various constants appearing in A.N.F of any Boolean function have been seen as different partial order derivatives. For example, if the A.N.F coding is (a_3, a_2, a_1, a_0) for a rule $f(a, b)$ then $a_0 = f(0, 0)$, $a_1 = (\partial f/\partial a)_{(0,0)}$, $a_2 = (\partial f/\partial b)_{(0,0)}$, $a_3 = \left(\partial^2 f/\partial a \partial b\right)_{(0,0)}$. The connection between damage spreading and Jacobian was introduced in [8,9].

3. Studies on the H.D.s between Boolean functions

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 n binary variables is defined as the H.D. between the n-bit binary equivalents of the rule numbers according to Wolfram's labeling convention [3]. For example, let us take two Boolean functions of three variables viz. Rule 34 and Rule 225. Their 8-bit binary representations are **00**100**10** and **11**100**01** 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 of these two rules (viz. $f_1 \oplus f_2$), the *weight* of a Boolean function being defined as the number of '1's in the output column of its Truth Table. It is worthwhile to mention here that the *minimum H.D.* between a Boolean function f and the set of all affine functions is called the *degree of non-linearity* of f.

Theorem 3.1. *The H.D. between any two different affine Boolean functions of n variables is 2^{n-1} except when they are the complements of each other, in which case the H.D. between them is 2^n.*

Proof. The H.D. between any n-variable Boolean rule and its complementary rule is 2^n because they differ from each other in every bit position and their length is 2^n.

Rule 0 is the only linear rule with weight equal to 0; as each of the remaining $(2^n - 1)$ linear rules is *balanced*, its weight is $2^n/2 = 2^{n-1}$. Again, the complement of the null function i.e. Rule $(2^n - 1)$ is the only non-linear affine rule of weight equal to 2^n. Each of the remaining $(2^n - 1)$ non-linear affine functions, being the complement of some linear rule, also has its weight equal to $2^n - 2^{n-1} = 2^{n-1}$. So the weight of each proper affine function of n binary variables is 2^{n-1}.

Thus, the H.D. between the null function and each proper affine function is 2^{n-1} because the former has all 2^n outputs in its Truth Table equal to 0 while the latter has exactly 2^{n-1} outputs equal to 1.

Similarly, the H.D. between the identity function and each proper affine function is 2^{n-1} because the former has all 2^n outputs equal to '1' whereas the latter has exactly 2^{n-1} outputs equal to '0'.

Finally, the sum mod 2 of two distinct proper affine functions, which are not complements of each other, is evidently another proper affine rule, hence its weight is also 2^{n-1}.

This proves our theorem. □

Corollary 1. *H.D. between any two linear Boolean rules of n variables is 2^{n-1}.*

Theorem 3.2. *If H.D. of a non-linear rule of n variables from one of the balanced rules is even (odd), then that from any other balanced rule is also even (odd).*

Proof. The weight of every balanced rule is 2^{n-1}. From any non-linear rule if one wants to construct all possible balanced rules having weight 2^{n-1} then this needs some bit position to be changed or flipped and the number of flipping operations on a non-linear Boolean function f to get a balanced Boolean function g which is same as the H.D. between two rules f and g. To get 2^{n-1} number of 1's from any non-linear rule f requires flipping some 1's to 0's and also in some other places from 0 to 1. Assume that the weight of f is x, then three cases arises:

Case 1: $x > 2^{n-1}$

Here f contains $(x - 2^{n-1})$ extra 1's and to construct a balanced function, $(x - 2^{n-1})$ number of flipping operations is required to change these extra 1's to 0's and without changing any 0's to 1's in f. Therefore the H.D. between f and all these balanced functions is $(x - 2^{n-1})$. This value is either an even number or an odd number.

Another possibility is to get 2^{n-1} number of 1's from f is to change y number of 0's to 1's in f becoming the weight is $(x + y)$ and $(x + y - 2^{n-1})$ extra 1's are required to flip them to 0's to obtain a balanced Boolean function. In this case the H.D. is $y + (x + y - 2^{n-1}) = (x - 2^{n-1}) + 2y$. If $(x - 2^{n-1})$ is even (or odd) then $(x - 2^{n-1}) + 2y$ is also even (or odd).

Two other cases, Case 2: $x = 2^{n-1}$ and Case 3: $x < 2^{n-1}$ can be proved similarly. Hence proved. □

Corollary 2. *Except Rule 0 and Rule 2^{n-1}, if H.D. of a non-linear rule of n variables from one of the linear/affine rules is even (odd), that from any other linear/affine rule is also even (odd).*

Theorem 3.3. *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 $(2^n - m)$.*

Proof. We are given that $m =$ H.D. between f and $g =$ weight of $(f \oplus g)$. Now the complement of f is $\bar{f} = f \oplus 1$. Thus, H.D. between $\bar{f}$ and $g =$ weight of $(\bar{f} \oplus g) =$ weight of $[(f \oplus 1) \oplus g] =$ weight of $[(f \oplus g) \oplus 1] =$ weight of $\overline{(f \oplus g)} = 2^n -$ weight of $(f \oplus g) = 2^n - m$. □

Corollary 3. *For any non-linear 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}.*

Proof. Consider an arbitrary non-linear rule f and any affine rule g. If the H.D. between f and g is less than or equal to 2^{n-1} then there is nothing to prove; otherwise g^c which is an affine rule will serve the purpose on using the relation:

$\text{H.D.}(f, g) = 2^n - \text{H.D.}(f, g^c)$. □

4. Theorems on Boolean functions and Jacobian matrices

Theorem 4.1. *The generalized algebraic normal form of any Boolean function f of p independent variables $y_1, y_2, y_3, \ldots, y_p$ may be written as*

$$f(Y) = k_0 \oplus \sum_{i=1}^{p} k_i y_i \oplus \sum_{\substack{i=1 \\ i<j}}^{p} \sum_{j=1}^{p} k_{ij} y_i y_j \oplus \sum_{\substack{i=1 \\ i<j<k}}^{p} \sum_{j=1}^{p} \sum_{k=1}^{p} k_{ijk} y_i y_j y_k \oplus \cdots \oplus k_{123\ldots p} y_1 y_2 \ldots y_p$$

where $\sum$ denotes continued "addition modulo 2", each of the coefficients $k_0, k_1, k_2, \ldots, k_{12}, k_{13}, \ldots, k_{123}, \ldots, k_{123\ldots p}$ may be either 0 or 1 and $Y = (y_1, y_2, y_3, \ldots, y_p)$ which is a $1 \times p$ row-vector.

Such a function $f(Y)$ satisfies the relation

$$f(Y) = \text{grad}\,(f)\,.Y^{\mathrm{T}}.$$

[where '.' denotes usual matrix multiplication modulo 2, Y^{T} is the transpose of Y, i.e. Y written as a $p \times 1$ column-vector] if and only if

$k_0 = k_{12} = k_{13} = \cdots = k_{1234} = \cdots = 0$. *In other words, all the Boolean functions having product-terms containing only an odd number of literals in the A.N.F satisfies the above relation.*

Proof. Throughout this proof, the product-terms such as y_1y_2, $y_1y_2y_3$ etc. (including '1' and the single-literal terms like y_1, y_2 etc.), appearing in the A.N.F. of any Boolean function, will be referred to as *elementary functions*.

An *elementary function* will be called *odd* or *even* according to the number of literals in the function is odd or even. Thus, $y_1, y_1y_2y_3$ etc. are *odd elementary functions* while $y_1y_2, y_1y_2y_3y_4$ etc. are *even elementary functions.*

The elementary function $f(Y) = 1$ is also considered an even elementary function as number of literals in it is 0, 0 being considered an even number.

Evidently, given the number of variables, the set of odd elementary functions and the set of even elementary functions are mutually exclusive and their union gives the set of all possible elementary functions.

In proving this theorem, we shall take the help of the following two results:

(I) $\underbrace{a \oplus a \oplus \cdots \oplus a}_{k \text{ times}} = \begin{cases} a & \text{if } k \text{ is an odd number,} \\ 0 & \text{if } k \text{ is an even number.} \end{cases} \quad \forall a \in \mathbb{B}$

This is obvious from the very definition of addition modulo 2 and its associativity property.

(II) The *gradient* operator is distributive over addition modulo 2, i.e. $\text{grad}\,(\phi \oplus \psi) = \text{grad}\,(\phi) \oplus \text{grad}\,(\psi)$ for any two Boolean functions $\phi : \mathbb{B}^n \to \mathbb{B}, \quad \psi : \mathbb{B}^n \to \mathbb{B}$.

This follows directly from the *additivity* property [4] of Boolean derivatives, viz.

$\partial\,(\phi \oplus \psi)\,/\partial x_j = \left(\partial\phi/\partial x_j\right) \oplus \left(\partial\psi/\partial x_j\right) \quad \forall\, j, j = 1, 2, \ldots, n.$

To prove the sufficiency:

For the function $f(Y) = 0$, which certainly does not contain any even elementary function, $\text{grad}(f) = 0$ identically so that the relation $f(Y) = \text{grad}(f).Y^{\mathrm{T}}$ is trivially true.

Irrespective of the number of variables (p), let us take an odd elementary function of the form $f_1(Y) = y_i$ where $i \in \{1, 2, 3, \ldots, p\}$. Clearly,

$$\partial f_1/\partial y_r = \begin{cases} 1 & \text{if } r = i, \\ 0 & \text{otherwise.} \end{cases} \quad r \in \{1, 2, 3, \ldots, p\}\,.$$

Thus, all entries of the vector $\text{grad}(f_1)$ are '0's except the ith one which is a '1'.

$$\therefore\ \text{grad}\,(f_1)\,.Y^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & \cdots & 1 & \cdots & 0 \end{bmatrix}.\begin{bmatrix} y_1 & y_2 & \cdots & y_i & \cdots & y_p \end{bmatrix}^{\mathrm{T}}$$
$$= 0 \oplus 0 \oplus \cdots \oplus y_i \oplus \cdots \oplus 0 = y_i = f_1(Y).$$

Now, let us take another odd elementary function of the form $f_3(Y) = y_iy_jy_k$ where $i, j, k \in \{1, 2, 3, \ldots, p\}$, $i \neq j \neq k$.

Again, evidently,

$$\partial f_3/\partial y_r = \begin{cases} y_jy_k & \text{if } r = i, \\ y_ky_i & \text{if } r = j, \\ y_iy_j & \text{if } r = k, \\ 0 & \text{otherwise.} \end{cases} \quad r \in \{1, 2, \ldots, p\}\,.$$

So, in this case, we shall obtain

$$\begin{aligned} \text{grad}\,(f_3) &= y_i.\,(\partial f_3/\partial y_i) \oplus y_j.\,\left(\partial f_3/\partial y_j\right) \oplus y_k.\,(\partial f_3/\partial y_k)\,, \quad \text{all other terms vanish,} \\ &= y_i.y_jy_k \oplus y_j.y_ky_i \oplus y_k.y_iy_j = \left(y_iy_jy_k \oplus y_iy_jy_k\right) \oplus y_iy_jy_k \\ &= 0 \oplus y_iy_jy_k = y_iy_jy_k \\ &= f_3(Y). \end{aligned}$$

Proceeding in this manner, it can be shown that, for any odd elementary function $f(Y)$, the expression $\text{grad}(f).Y^{\mathrm{T}}$ reduces to a continued addition modulo 2 of an odd number of terms, each of which is identical to $f(Y)$ and, hence, the result is nothing but $f(Y)$ [in accordance with (I)].

Moreover, it clearly follows from (II) that the relation $f(Y) = \text{grad}(f).Y^{\mathrm{T}}$ is also satisfied by a linear combination of any number of odd elementary functions; e.g.

Suppose, $f(Y) = y_1 \oplus y_1y_2y_3$. Say, $f_1(Y) = y_1, \qquad f_1(Y) = y_1y_2y_3$.

Then, $f(Y) = f_1(Y) \oplus f_3\,(Y)$.

$$\therefore \text{ grad } (f) = \text{grad}\,(f_1 \oplus f_3) = \text{ grad } (f_1) \oplus \text{grad}\,(f_3)\,.$$

$$\begin{aligned}\therefore \text{ grad } (f)\,.Y^{\mathrm{T}} &= (\text{ grad } (f_1) \oplus \text{grad}\,(f_3))\,.Y^{\mathrm{T}}, \quad \text{by (II)}\\ &= \text{grad}\,(f_1)\,.Y^{\mathrm{T}} \oplus \text{grad}\,(f_3)\,.Y^{\mathrm{T}}\\ &= f_1(Y) \oplus f_3(Y), \quad \text{since both are odd elementary functions}\\ &= f(Y).\end{aligned}$$

To prove the necessity:

For the even elementary function $f(Y) = 1$, $\text{grad}(F) = 0$ so that $\text{grad}(F).Y^{\mathrm{T}} = 0 \neq F(Y)$.

Now, let us consider an even elementary function of the form

$f_2(Y) = y_i y_j$ where $i, j \in \{1, 2, 3, \ldots, p\}$, $i \neq j$.

Thus,

$$\partial f_2/\partial y_r = \begin{cases} y_j & \text{if } r = i,\\ y_i & \text{if } r = j, \qquad r \in \{1, 2, 3, \ldots, p\}\,.\\ 0 & \text{otherwise..}\end{cases}$$

$\therefore$ In this case,

$$\begin{aligned}\text{grad}\,(f_2) &= y_i.\,(\partial f_2/\partial y_i) \oplus y_j.\,\left(\partial f_2/\partial y_j\right), \quad \text{all other terms vanish,}\\ &= y_i.y_j \oplus y_j.y_i = y_i y_j \oplus y_i y_j = 0 \neq f_2(Y).\end{aligned}$$

Proceeding similarly, it can be shown that, for any even elementary function $f(Y)$, the expression $\text{grad}(f).Y^{\mathrm{T}}$ reduces to a continued addition modulo 2 of an even number of terms, each of which is identical to $f(Y)$ and, hence, the result is 0 [according to (I)].

Again, it follows from (II) that the relation $\text{grad}(f).Y^{\mathrm{T}} = 0$ *does not hold* for any linear combination of even elementary functions e.g. for

$$f(Y) = y_1y_2 \oplus y_2y_3, \qquad f(Y) = y_1y_2 \oplus y_1y_2y_3y_4 \text{ etc.}$$

Finally, we conclude that, for a rule which is a linear combination of any number of elementary functions (odd or even), the bitwise product modulo 2 of the gradient of a Boolean function with the input vector retains or "filters out" only the odd elementary functions (if any) from the rule under consideration and discards the even elementary functions (if any).

For example, let $f(Y) = y_1y_2y_3 \oplus y_2y_3 \oplus y_1 \oplus 1$.

$$\begin{aligned}\text{Then, grad}(f).Y^{\mathrm{T}} &= \text{grad}(y_1y_2y_3).Y^{\mathrm{T}} \oplus \text{grad}(y_2y_3).Y^{\mathrm{T}} \oplus \text{grad}(y_1).Y^{\mathrm{T}} \oplus \text{grad}(1).Y^{\mathrm{T}}\\ &= y_1y_2y_3 \oplus 0 \oplus y_1 \oplus 0 = y_1y_2y_3 \oplus y_1\\ &\neq f(Y).\end{aligned}$$

This establishes the proposition, which may be re-stated as:

The relation $f(\boldsymbol{Y}) = \textbf{grad}\,(\boldsymbol{f}).\boldsymbol{Y}^{\mathbf{T}}$ is satisfied if and only if $f(Y)$ is a linear combination of odd elementary functions. □

Corollaries:

(i) A linear rule of p variables has the general form

$$f_L(Y) = k_1y_1 \oplus k_2y_2 \oplus \cdots \oplus k_py_p, \quad k_i \in \mathbb{B}\ \forall\ i \in \{1, 2, \ldots, p\}\,,$$

and, since $f_1(Y) = y_i$ is an odd elementary function $\forall i \in \{1, 2, \ldots, p\}$, any *linear Boolean function* $f_L(Y)$ satisfies the relation $f_L(Y) = \text{grad}\,(f_L).Y^{\mathrm{T}}$.

(ii) For p variables, the number of elementary functions is $N = 2^p$; the number of odd elementary functions is $N_o = 2^{p-1}$;

The number of even elementary functions is $N_e = 2^{p-1}$. The number of Boolean functions of p variables which satisfy the relation $f(Y) = \text{grad}(f).Y^{\mathrm{T}}$ is

$$M_p = {}^{N_o}\mathbf{C}_0 + {}^{N_o}\mathbf{C}_1 + {}^{N_o}\mathbf{C}_2 + \cdots + {}^{N_o}\mathbf{C}_{N_o} = 2^{N_o} = 2^{2^{p-1}} = \sqrt{N_p};\ \text{of which only } 2^p \text{ are linear rules.}$$

e.g. for 3 variables, the number $M_3 = 2^{2^{3-1}} = 2^4 = 16$, of which $2^3 = 8$ are linear rules.

(iii) If a function $f(Y)$ is a linear combination of even elementary functions only, then we must have $\text{grad}\,(f).Y^{\mathrm{T}} = 0$ identically. Clearly, the number of Boolean functions of p variables satisfying the relation $\text{grad}(f).Y^{\mathrm{T}} \equiv 0$ is also

$$M_p = 2^{2^{p-1}}.$$

(iv) In general, we can write any Boolean function $f(Y)$ as

$$f(Y) = \textbf{grad}\,(f)\,.Y^{\mathbf{T}} \oplus \mathcal{E}(Y)$$

where $\mathcal{E}(Y)$ may be called the *error function* or *error part* of $f(Y)$ and is the continued addition modulo 2 of the even elementary terms present in $f(Y)$.

e.g. For $f(Y) = y_1y_2y_3 \oplus y_2y_3 \oplus y_1 \oplus 1$,

$\mathcal{E}(Y) = y_2y_3 \oplus 1$.

Theorem 4.2. *The Jacobian matrices of two UCAs of the same size, with the same boundary conditions but with different rules, are identical if and only if the rule of one of the CAs is the Boolean complement of that of the other CA.*

Proof. Let us denote the two rules of the two UCAs in question as f and g and their Jacobian matrices as J_1 and J_2. Thus, according to our convention stated in Section 2.2 (with $p = 3$), $f^i = f(x^t_{i-1}, x^t_i, x^t_{i+1})$; $g^i = g(x^t_{i-1}, x^t_i, x^t_{i+1})$ for each $i = 2, 3, \ldots, n-1$. For $i = 1$ and for $i = n$, x^t_{i-1} and x^t_{i+1} are to be replaced by the relevant *left neighbor* and *right neighbor* respectively.

It is clear, from the definition of $\boldsymbol{J}$, that $\boldsymbol{J}_1 \equiv \boldsymbol{J}_2$ *if and only if*

$$\partial f^i/\partial x^t_j = \partial g^i/\partial x^t_j \quad \forall\, i \in \{1, 2, \ldots, n\}\,, \forall j \in \{1, 2, \ldots, n\}\,.$$

Obviously, if x_j does *not belong to the neighborhood* of x_i, then $\partial f^i/\partial x^t_j = \partial g^i/\partial x^t_j = 0$ or, in other words, f^i and g^i are independent of each such x_j.

So, to prove the proposition in question, it suffices to show that:

For any two *non-identical* Boolean functions f and g of p independent variables $y_1, y_2, \ldots, y_p$,

$$\partial f/\partial y^t_j = \partial g/\partial y^t_j \quad \forall\, j \in \{1, 2, 3, \ldots, p\}$$

if and only if

$g = \bar{f}$ i.e. f and g are Boolean complements of each other (in our case, $p = 3$).

To prove the sufficiency:

Let $g = \bar{f}$. Then, we can write g as $g = f \oplus 1$.

$$\therefore\ \partial g/\partial x_j = \partial(f \oplus 1)/\partial y_j = \partial f/\partial y_j \oplus \partial(1)/\partial y_j = \partial f/\partial y_j \oplus 0 = \partial f/\partial y_j \quad \forall\, j \in \{1, 2, \ldots, n\}\,.$$

To prove the necessity:

Suppose, there exist two rules f and g such that $\partial f/\partial y^t_j = \partial g/\partial y^t_j = \alpha_j$, say, $\forall j \in \{1, 2, 3, \ldots, p\}$, where $\alpha_j \in \mathbb{B}$.

Let us define a function $h(y_1, y_2, \ldots, y_p)$ of the p independent variables $y_1, y_2, \ldots, y_p$ as $h = f \oplus g$.

Then,

$$\partial h/\partial y_j = \partial(f \oplus g)/\partial y_j = (\partial f/\partial y_j) \oplus (\partial g/\partial y_j) = \alpha_j \oplus \alpha_j = 0 \quad \forall\, j \in \{1, 2, \ldots, p\}.$$

Thus, $h(y_1, y_2, \ldots, y_p)$ is independent of all the p input variables, which implies that h must be a constant function, either 0 or 1.

If $h = 0, f \oplus g = 0$ which implies that $g = f$. If $h = 1, f \oplus g = 1$ which implies that $g = f \oplus 1 = \bar{f}$. This completes the proof. □

Corollaries:

All the corollaries to Theorem 4.2 are stated in terms of CA with $p = 3$ (Section 2.2), although they are fairly general.

(i) Irrespective of the number of bits in the CA, it is possible to have $2^{2^3} = 256$ different UCAs with either type of boundary condition (null/periodic) – since there are 256 different Boolean functions of 3 variables – but there are only $256/2 = 128$ distinct Jacobian matrices (for given boundary conditions), each characterizing a pair of Boolean functions which are logical complements of each other; e.g. the complement of Rule 30 is Rule 225($= 255 - 30$), hence each of the UCAs $\langle 30, 30, 30, 30\rangle$PB and $\langle 225, 225, 225, 225\rangle$PB has the Jacobian matrix

$$J_{30}|_{\text{PB}} = J_{225}|_{\text{PB}} = \begin{bmatrix} \bar{x}_2 & \bar{x}_1 & 0 & 1 \\ 1 & \bar{x}_3 & \bar{x}_2 & 0 \\ 0 & 1 & \bar{x}_4 & \bar{x}_3 \\ \bar{x}_4 & 0 & 1 & \bar{x}_1 \end{bmatrix}.$$

(ii) Let us now consider the HCA $\langle 225, 30, 30, 225\rangle$PB. As $\partial f_{225}/\partial x_i = \partial f_{30}/\partial x_i \forall i \in \{1, 2, 3\}$ ($\because \partial \bar{f}/\partial x_j = \partial f/\partial x_j \forall j$, as established in the proof of Theorem 4.2), it is clear that this HCA will have the same Jacobian matrix $\boldsymbol{J_{30}}|_{\text{PB}}$ shown in corollary (i). Thus, in general, we can say that if we are given an $n \times n$ Jacobian matrix, which *resembles* that of a UCA of n cells, the matrix may actually belong to any one of 2^n different CAs, of which only 2 are uniform and the rest are hybrid. In this context, "*resemblance* to the Jacobian matrix of a UCA" means that the vector formed by the diagonal element of each row, along with its two neighbors, *in the correct order*, is *essentially* the same for all the rows (e.g. in $\boldsymbol{J_{30}}|_{\text{PB}}$ considered in corollary (i), the relevant vectors are $[1\ \bar{x}_2\ \bar{x}_1]$, $[1\ \bar{x}_3\ \bar{x}_2]$, $[1\ \bar{x}_4\ \bar{x}_3]$, $[1\ \bar{x}_1\ \bar{x}_4]$ which are of the general form $[1\ \overline{rb_i}\ \bar{x}_i]$, $i = 1, 2, 3, 4$, $\overline{rb_i}$ being the right neighbor of $\bar{x}_i$).

For example, $\langle 30, 30, 30, 30\rangle$PB, $\langle 225, 225, 225, 225\rangle$PB, $\langle 30, 225, 30, 30\rangle$PB, $\langle 30, 30, 225, 225\rangle$PB, $\langle 225, 225, 30, 225\rangle$PB are some of the $2^4 = 16$ 4-bit CAs characterized by $\boldsymbol{J_{30}}|_{\text{PB}}$.

(iii) For a linear CA, whether uniform or hybrid, the Jacobian matrix is a unique **constant binary matrix** [4,5] but *the converse is not true*. This follows from corollaries (i) and (ii) because the complement of each linear rule is itself necessarily a *non-linear* rule. e.g. The UCA $\langle 60, 60, 60, 60\rangle$NB, where Rule 60 is a linear rule, is characterized by the Jacobian matrix

$$\boldsymbol{J_{60}}|_{\text{NB}} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Table 5.1.1
Hamming distances of three-variable Boolean functions from the 8 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
…	…	…	…	…	…	…	…	…
255	**8**	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Table 5.1.2
Classification of three-variable Boolean functions on the basis of deviation from linearity.

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

(i) 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 non-linearity of each of these rules is **0**.
(ii) Each rule in **CLASS 3** has its complement in **CLASS 1** (in the order in which the rules are arranged in Table 5.1.2, the ith rule in Class 3 is the complement of the $(65-i)$th rule in Class 1, $i = 1, 2, \ldots, 64$); the degree of non-linearity of each of the **CLASS 1** and **CLASS 3** rules is **1**.
(iii) In the order in which the rules are presented in Table 5.1.2, the ith rule in Class 2 is the complement of the $(113-i)$th rule in **CLASS 2**, $i = 1, 2, \ldots, 56$; the degree of non-linearity of each **CLASS 2** rule is **2**.

The complement of Rule 60 is Rule 195 $(=255-60)$; so, the non-linear UCA $\langle 195, 195, 195, 195\rangle$NB gives the same Jacobian matrix $\boldsymbol{J_{195}}|_{\mathbf{NB}} = \boldsymbol{J_{60}}|_{\mathbf{NB}}$. However, as the set of *affine functions* comprises the linear rules and their complements, we conclude that *if the Jacobian matrix of a UCA/HCA is constant, all the rules involved must be affine Boolean rules*.

5. Boolean functions of three variables

5.1. Classification of Boolean rules of 3 variables based on H.D.s from the set of linear rules

We proceed by drawing up a table of H.D.s of all Boolean functions of 3 variables from the 8 linear rules, as shown in Table 5.1.1:

All observations made are consistent with the theorems in Section 3.

A Boolean rule of 3 variables is said to belong to **Class** $\boldsymbol{m}$ if $\boldsymbol{m}$ is the minimum possible H.D. of the non-linear rule from any linear rule of 3 variables, i.e. there exists at least one linear rule such that the H.D. of the rule under consideration from this linear rule is $\boldsymbol{m}$ and, if $\boldsymbol{m}'$ is the H.D. of the said rule from any other linear rule, then $\boldsymbol{m}'$ is larger than or equal to $\boldsymbol{m}$. The classification is presented in Table 5.1.2.

5.2. Sub-classification of the classes of three-variable Boolean rules based on position of bit-mismatch with nearest linear rule

Each **CLASS 1** rule has exactly one linear rule at a H.D. of 1 from itself; that linear rule will be called its *nearest linear rule*. We express 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, $q = 0, 1, 2, \ldots, 7$, the rule is said to belong to Subclass q of **CLASS 1**, denoted by $1 : q$; e.g. Nearest linear rule of Rule 22 is 150.

	$\mathbf{2^7}$	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

Table 5.2.1
Sub-classification of **CLASS 1**.

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 5.2.2
Sub-classification of **CLASS 3**.

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

Therefore, Rule 22 belongs to Subclass 7 of **CLASS 1**. Thus, there are 8 subclasses of **CLASS 1**, as shown in Table 5.2.1.

Table 5.2.2 shows that, each **CLASS 3** rule has exactly *three* nearest linear rules; so, it is not possible to sub-classify 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 sub-classified into 8 subclasses of 8 rules each in the following manner:

If the complement of a **CLASS 3** rule belongs to Subclass $\boldsymbol{q}$ of Class 1, then that **CLASS 3** rule is said to belong to Subclass $\boldsymbol{q}^*$ of **CLASS 3**, denoted by $\mathbf{3} : \boldsymbol{q}^*$.

For the **CLASS 2** rules, we observe that:

(i) Each of the 56 even-numbered rules in **CLASS 2** is the complement of one of the 56 odd-numbered rules in **CLASS 2**.

(ii) Every odd rule in **CLASS 2** is at a Hamming Distance of 2 from exactly one linear rule (and at a 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 a H.D. of 2 from exactly three linear rules and at a H.D. of 6 from exactly one linear rule and, hence, for an even **CLASS 2** rule, the nearest linear rule *is not unique*. e.g. Rule 6 (Table 5.1.1)

(iii) 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 sub-classification 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 sub classification could be done in a manner similar to that in which the **CLASS 3** rules have been sub-classified. e.g. Nearest linear rule of Rule 3 is 0.

	2^7	2^6	2^5	2^4	2^3	2^2	$\mathbf{2^1}$	$\mathbf{2^0}$
3	0	0	0	0	0	0	**1**	**1**
0	0	0	0	0	0	0	**0**	**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 $\mathbf{2 : 1^*}$ as shown in Table 5.2.3.

5.3. Error function for three binary variables

The general A.N.F. of a function $f(y_1, y_2, y_3)$ of three variables y_1, y_2, y_3 is

$$f(y_1, y_2, y_3) = k_{123}y_1y_2y_3 \oplus k_{23}y_2y_3 \oplus k_{31}y_3y_1 \oplus k_3y_3 \oplus k_{12}y_1y_2 \oplus k_2y_2 \oplus k_1y_1 \oplus k_0$$

where $k_{123}, k_{12}, k_{23}, k_{31}, k_3, k_2, k_1, k_0 \in \mathbb{B}$.

Hence, from corollary (iv) of Theorem 4.1, the general form of the *error function* is

$$\mathcal{E}(y_1, y_2, y_3) = k_{12}y_1y_2 \oplus k_{23}y_2y_3 \oplus k_{31}y_3y_1 \oplus k_0.$$

Table 5.2.3
Sub-classification of **CLASS 2**.

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

Thus, we have divided the 256 rules of 3 variables into 32 classes/subclasses of 8 rules each. All the 8 rules in each such class/subclass have a number of similar properties.

Table 5.3.1
Algebraic expressions of error parts of all 256 three-variable Boolean functions.

Row no.	Form of $\mathcal{E}(Y)$	Number of rules with the specified $\mathcal{E}(Y)$	Classes /subclasses to which these rules belong
1	0	16	**0, 1 : 7**
2	y_1y_2	16	**1 : 6, 2 : 1***
3	y_3y_1	16	**1 : 5, 2 : 2***
4	$y_1y_2 \oplus y_3y_1$	16	**1 : 4, 2 : 3***
5	y_2y_3	16	**1 : 3, 2 : 4***
6	$y_1y_2 \oplus y_2y_3$	16	**1 : 2, 2 : 5***
7	$y_2y_3 \oplus y_3y_1$	16	**1 : 1, 2 : 6***
8	$y_1y_2 \oplus y_2y_3 \oplus y_3y_1$	16	**3 : 0*, 2 : 7***
9	1	16	**3 : 7*, 4**
10	$y_1y_2 \oplus 1$	16	**3 : 6*, 2 : 1**
11	$y_3y_1 \oplus 1$	16	**3 : 5*, 2 : 2**
12	$y_1y_2 \oplus y_3y_1 \oplus 1$	16	**3 : 4*, 2 : 3**
13	$y_2y_3 \oplus 1$	16	**3 : 3*, 2 : 4**
14	$y_1y_2 \oplus y_2y_3 \oplus 1$	16	**3 : 2*, 2 : 5**
15	$y_2y_3 \oplus y_3y_1 \oplus 1$	16	**3 : 1*, 2 : 6**
16	$y_1y_2 \oplus y_2y_3 \oplus y_3y_1 \oplus 1$	16	**1 : 0, 2 : 7**

(i) It is evident that each rule in Row 1 of Table 5.3.1 does not contain any even elementary function so that the entire function is covered by $\text{grad}(F).Y^{\mathrm{T}}$.
(ii) All the rules occurring in the first 8 rows of Table 5.3.1 are even-numbered while all the rules in the last 8 rows are, obviously, odd-numbered (indicated by the absence and presence of the term '1' respectively).
(iii) In general, for p variables, it can be easily shown that there will be M_p such groups, each of M_p rules, such that all the rules in a group will be characterized by the same error function $\mathcal{E}(Y)$.

As each of these 4 coefficients may take on one of the two values 0 or 1, there are $2^4 = 16$ different forms of error functions possible for the three-variable case.

Again, for $f(y_1, y_2, y_3)$, the general form of $\text{grad}(f).Y^{\mathrm{T}}$ is $k_{123}y_1y_2y_3 \oplus k_1y_1 \oplus k_2y_2 \oplus k_3y_3$; there are again $2^4 = 16$ different forms of $\text{grad}(f).Y^{\mathrm{T}}$

So, given a particular form of the *error function* (say, $y_1y_2 \oplus y_2y_3 \oplus 1$), there are 16 different Boolean rules (e.g. $y_1y_2y_3 \oplus y_1y_2 \oplus y_2y_3 \oplus 1$, $y_1y_2 \oplus y_2y_3 \oplus y_2 \oplus y_1 \oplus 1$ etc.) all of which possess the specified *error part*, by corollary (iv) of Theorem 4.1.

Thus, the 256 Boolean rules of 3 variables may be divided into $256/16 = 16$ groups, each of the 16 functions characterized by the same *error function*. We also find that each such 16-member group contains two subclasses (or sometimes one class and one subclass) of our classification based on H.D. from linear rules. This is elaborated in Table 5.3.1.

6. Importance of the Jacobian matrix in the context of the evolution of a CA

6.1. The state transition diagram of a CA

A cellular automaton is uniquely specified if (i) the number of cells, (ii) the boundary conditions, (iii) the definition of *neighborhood* of each cell *and* (iv) the rule applied to each cell are specified.

Suppose we have a Rule 170 UCA4NB. If at an instant t, the state of the CA is **11** $\equiv 1011$, that at $t + 1$ will be **6** $\equiv 0110$. This transition may be represented by drawing an *arrow* from the state **11** to its *successor* state **6** i.e. $11 \rightarrow 6$.

Similarly, the successor of state **6** is **12**, that of **15** is **14** and so on. Thus, 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 S.T.D.) of the CA. In other words, the S.T.D. 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 S.T.D. of the UCA considered is shown in Fig. 6.1.

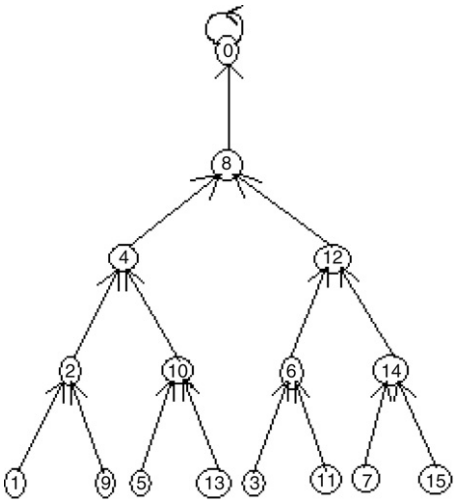

Fig. 6.1. S.T.D. of ⟨170, 170, 170, 170⟩NB.

6.2. The Jacobian matrices of linear CAs

For any linear CA, as already stated, the Jacobian matrix is identically equal to a matrix of '0's and '1's, irrespective of the present state.

Moreover, for a linear CA, the following relation holds for any instant

$$t : (X^{t+1})^{\mathrm{T}} = [F(X^t)]^{\mathrm{T}} = J.(X^t)^{\mathrm{T}}$$

where $(X^k)^{\mathrm{T}}$ denotes the transpose of the $1 \times n$ row-vector X^k, $k = t, t+1$.

Henceforth, for the sake of convenience, the superscript '$^{\mathrm{T}}$' will be dropped, whenever this does not cause any ambiguity, and the symbol X^t will often be taken to represent the $n \times 1$ column-vector $\begin{bmatrix} x_1^t & x_2^t & \cdots & x_n^t \end{bmatrix}^{\mathrm{T}}$; similarly for $X^{t+1}, F(X^t)$.

Thus, $X^{t+1} = F(X^t) = J.X^t$ for a linear CA.

Furthermore, for a linear CA, we cannot only obtain the successor of each state by simply multiplying its Jacobian matrix with the present state (instead of applying the local mappings to individual cells) but can also deduce all the properties of the state transition diagram directly from the algebraic properties (such as rank, nullity, determinant etc.) of the said Jacobian matrix which is a constant binary matrix; thus the Jacobian matrix acts as a *linear handle* for the linear CAs. As such, the STDs of linear CAs are predictable and symmetric in structure.

6.3. The Jacobian matrices of non-linear CAs

For a non-linear CA, the Jacobian matrix cannot act as linear handle because:

(a) $X^{t+1} \neq J.X^t$ in general.
(b) J is, in general, itself a function of X^t so that its matrix properties change depending on the present state X^t of the CA.

However, in analogy with the case of a linear CA, we may define the $n \times 1$ column-vector $J.X^t$ as the *predicted successor* of the present state X^t of a non-linear CA whereas $X^{t+1} = F(X^t)$ may be called the *actual successor* of X^t.

We have studied extensively the Jacobian matrices of all the 256 types of 4-bit UCA (with either type of boundary conditions) and, while studying the efficacy of the Jacobian matrix in predicting the evolution of a non-linear UCA, we made the following interesting experimental observations:

Observation (i) For each of exactly 8 non-linear rules viz. Rules 22, 42, 76, 112, 128, 188, 218, 230, we found that the relation $X^{t+1} = J.X^t$ holds good, or, in other words, the Jacobian matrix predicts the entire evolution of the system correctly, just as the J of a linear UCA does. An *important difference* with the linear UCA case, however, is that J itself depends on the present state of the CA in each of the said cases.

Observation (ii) For each of exactly 16 non-linear rules viz. 15, 25, 37, 51, 67, 85, 105, 127, 143, 153, 165, 179, 195, 213, 233, 255, it was found that the *predicted successor* of each state differs from the *actual successor* in every bit; an example is presented in Table 6.3.1:

Observation (iii) For each of exactly 16 non-linear rules viz. 0, 23, 40, 63, 72, 119, 136, 159, 160, 183, 192, 95, 96, 215, 232, 255, $J.X^t = 0$ for each X^t, where $0 = [0\ 0\ 0\ 0]^{\mathrm{T}}$, i.e. the Jacobian matrix predicts state 0 as the successor of each state but we know that this is true only for a Rule 0 UCA.

Table 6.3.1
Prediction of Jacobian matrix vis-à-vis actual evolution of a Rule 67 UCA4NB.

Present state X^t	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Predicted successor $J.X^t$	0000	0001	0011	0010	0110	0111	0101	0110	1100	1101	1111	1110	1010	1011	1101	1110
Actual successor $X^{t+1} = F(X^t)$	1111	1110	1100	1101	1001	1000	1010	1001	0011	0010	0000	0001	0101	0100	0010	0001

Table 6.3.2
Instances of correct prediction by Jacobian matrix for Class 1 rules.

Subclass of Class1	Present states (X^t) for which $X^{t+1} = J.X^t$, i.e. the Jacobian matrix predicts next state correctly
1 : 7	All 16 states
1 : 6	0, 1, 2, 4, 5, 8, 9, 10
1 : 5	0, 1, 2, 3, 4, 6, 8, 9, 12
1 : 4	0, 1, 2, 4, 8, 9
1 : 3	0, 1, 2, 4, 5, 8, 9, 10
1 : 2	0, 1, 2, 4, 5, 8, 9, 10
1 : 1	0, 1, 2, 4, 8, 9
1 : 0	15

Table 6.3.3
Difference in prediction by Jacobian matrix and actual CA evolution for two rules in 1:3.

Present state X^t	Rule 162 UCA4NB		Rule 110 UCA4NB	
	$J.X^t$	$F(X^t)$	$J.X^t$	$F(X^t)$
0000	0000	0000	0000	0000
0001	0010	0010	0011	0011
0010	0100	0100	0110	0110
0011	01<u>1</u>0	01<u>0</u>0	01<u>1</u>1	01<u>0</u>1
0100	1000	1000	1100	1100
0101	1010	1010	1111	1111
0110	1<u>1</u>00	1<u>0</u>00	1<u>1</u>10	1<u>0</u>10
0111	1<u>1</u>00	1<u>0</u>10	1<u>1</u>01	1<u>0</u>11
1000	0000	0000	1000	1000
1001	0010	0010	1011	1011
1010	0100	0100	1110	1110
1011	01<u>1</u>0	01<u>0</u>0	11<u>1</u>1	11<u>0</u>1
1100	<u>1</u>000	<u>0</u>000	<u>1</u>100	<u>0</u>100
1101	<u>1</u>010	<u>0</u>010	<u>1</u>111	<u>0</u>111
1110	<u>10</u>00	<u>01</u>00	<u>10</u>10	<u>01</u>10
1111	<u>10</u>00	<u>01</u>10	<u>10</u>01	<u>01</u>11

Observation (iv) For each of the remaining UCAs, the *predicted successors* of some of the states are identical to the corresponding *actual successors* while for other states, mismatch occurs between the predicted and actual successors in one, two, three or even four bits. In the course of our study, we came across certain interesting patterns. For example, for all the 8 rules in *any subclass*, the Jacobian matrix predicts the successor correctly for the same set of present states; besides, for each case of incorrect prediction, bit-mismatch between the actual and predicted successors occurs at the same bit(s) for all the rules. In short, the Jacobian matrix gives *similar predictions* for all rules in a particular subclass. This is elucidated with examples presented in Tables 6.3.2 and 6.3.3 for a UCA4NB. Table 6.3.3 shows the prediction of Jacobian matrix vis-à-vis actual evolution of two UCA4NB with rules in 1:3 where the positions of bit-mismatch are underlined.

For a mathematical justification of these observations, we turn to Theorem 4.1 and its corollaries, particularly the concept of the *error function*:

Evidently, for any n-bit CA, J is an $n \times n$ matrix while X^t is an $n \times 1$ column-vector so that $J.X^t$ is an $n \times 1$ column-vector whose ith element is given by

$$\begin{aligned}
(J.X^t)_i &= [i\text{th row of } J.X^t].X^t = \left[\partial f^i/\partial x_1^t \ldots \partial f^i/\partial x_{i-1}^t\ \partial f^i/\partial x_i^t\ \partial f^i/\partial x_{i+1}^t \ldots \partial f^i/\partial x_n^t\right].\left[x_1^t \ldots x_{i-1}^t\ x_i^t\ x_{i+1}^t \ldots x_n^t\right]^{\mathrm{T}} \\
&= \left[0\,0 \ldots \partial f^i/\partial x_{i-1}^t\ \partial f^i/\partial x_i^t\ \partial f^i/\partial x_{i+1}^t \ldots 0\right].\left[x_1^t \ldots x_{i-1}^t\ x_i^t\ x_{i+1}^t \ldots x_n^t\right]^{\mathrm{T}} \\
&\quad \left\{\because f^i \text{ is independent of } x_j^t \text{ for } j \in \{1, 2, \ldots, n\} \text{ but } j \neq i-1, i, i+1\right\} \\
&= x_{i-1}^t.\left(\partial f^i/\partial x_{i-1}^t\right) \oplus x_i^t.\left(\partial f^i/\partial x_i^t\right) \oplus x_{i+1}^t.\left(\partial f^i/\partial x_{i+1}^t\right),\ \{\because \text{ all other terms vanish}\} \\
&= \mathrm{grad}(f^i).Y_i^t
\end{aligned}$$

where $Y_i^t = (x_{i-1}^t, x_i^t, x_{i+1}^t)$ represents the pre-defined neighborhood of the ith cell, $i = 2, 3, \ldots, n-1$; for $i = 1$ and $i = n$, x_{i-1} and x_{i+1} are to be replaced by the relevant *lb* and *rb* respectively.

Table 6.3.4
Error vector for Rule 162 UCA4NB.

X^t	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
$\mathbf{E}(X^t)$	0000	0000	0000	0010	0000	0000	0100	0110	0000	0000	0000	0010	1000	1000	1100	1110
$J.X^t$	0000	0010	0100	0110	1000	1010	1100	1100	0000	0010	0100	0110	1000	1010	1000	1000
$J.X^t \oplus \mathbf{E}(X^t)$	0000	0010	0100	0100	1000	1010	1000	1010	0000	0010	0100	0100	0000	0010	0100	0110

The fourth row of Table 6.3.4 agrees exactly with the third column of Table 6.3.3, as expected.

Now, from corollary (iv) of theorem I, we see that the bit in the ith cell of an n-cell CA at the "next" time-instant is given by $x_i^{t+1} = f^i(Y_i^t) = \text{grad}(f^i).Y_i^t \oplus \mathcal{E}^i(Y_i^t)$, where $\mathcal{E}^i(Y_i^t)$ denotes the *error part* of $f^i(Y_i^t)$.

$$\therefore x_i^{t+1} = (J.X^t)_i \oplus \mathcal{E}^i(Y_i^t) \quad \forall i \in \{1, 2, \ldots, n\}.$$

Thus, in vector-matrix notation, we can write the *global state* of the CA at time $t+1$ as $X^{t+1} = J.X^t \oplus \mathbf{E}(X^t)$ where $E(X^t)$ is an n-element column-vector whose ith element is given by $\mathcal{E}^i(Y_i^t)$. For example, for Rule 208 UCA4PB, $\mathbf{E}(X^{\mathbf{t}}) = \left[x_1^t x_4^t \quad x_3^t x_1^t \quad x_4^t x_2^t \quad x_1^t x_3^t\right]^{\mathrm{T}}$, from Tables 5.2.1 and 5.3.1; for Rule 123 UCA4NB,

$$\mathbf{E}(X^{\mathbf{t}}) = \begin{bmatrix} x_1^t x_2^t \oplus 1 \\ x_1^t x_2^t \oplus x_2^t x_3^t \oplus 1 \\ x_2^t x_3^t \oplus x_3^t x_4^t \oplus 1 \\ x_3^t x_4^t \oplus 1 \end{bmatrix}.$$

Even for an HCA, we could similarly deduce the algebraic expression for the error vector. Naturally it is the error vector, which completely accounts for the discrepancies between X^{t+1} and $J.X^t$, as observed by us.

We are now in a position to explain all our **observations**:

(i) Table 5.2.1 shows that all the 8 rules 22, 42, 76,112, 128, 188, 218 and 230 belong to 1 : 7. Again, from Row **1** of Table 5.3.1, we notice that for a UCA based on any of these 8 rules, $\mathcal{E}^i(Y_i^t) = 0$ identically. So, $\mathbf{E}(X^t) = [0\,0\,0\,0]^{\mathrm{T}}$ for each X^t of each such UCA. This clearly justifies **observation (i)**.

(ii) Rules 15, 51, 85, 105, 153, 165, 195 and 255 belong to **CLASS 4** while rules 25, 37, 67, 127, 143, 179, 213 and 233 belong to 3: 7*. Thus, Row **9** of Table 5.3.1 shows that for a UCA with any of these rules, $\mathcal{E}^i(Y_i^t) = 1$ identically so that $\mathbf{E}(X^t) = [1\,1\,1\,1]^{\mathrm{T}}$ for each X^t. This implies that in order to obtain the actual successor of any state, we have to toggle each bit of the predicted successor. This explains **observation (ii)**.

(iii) The A.N.F. of each of the rules 0, 23, 40, 63, 72, 119, 136, 159, 160, 183, 192, 95, 96, 215, 232 and 255 does not include any *odd elementary function*; e.g. $f_{119}(a, b, c) = bc \oplus 1, f_{40}(a, b, c) = bc \oplus ca$ etc. So, in each of these cases, the entire function appears as the error function, i.e. $J.X^t = [0\,0\,0\,0]^{\mathrm{T}}$ identically in accordance with Corollary (iii) of Theorem 4.1. This establishes **observation (iii)**.

(iv) Now that we have expressed the error vector as a function of the present state, we can easily obtain its Truth Table representation; if for a certain X^t, the ith element of $\mathbf{E}(X^t)$ is 1, the predicted successor of X^t must differ from its actual successor at the ith bit. Moreover, *as all UCAs with rules belonging to the same subclass have identical error vectors* (for given boundary conditions), the observations presented in Tables 6.3.2 and 6.3.3 can also be justified. e.g. for a Rule 162 UCA4NB, $\mathbf{E}(X^{\mathbf{t}}) = \left[x_1^t x_2^t \quad x_2^t x_3^t \quad x_3^t x_4^t \quad 0\right]^{\mathrm{T}}$. The Truth Table of $\mathbf{E}(X^t)$ is given by the first two rows of Table 6.3.4:

7. The modified Jacobian matrix

In Section 6.3, we have established that, in general, for any CA, we can write $X^{t+1} = \boldsymbol{J}.X^t \oplus \mathbf{E}(X^t)$, all symbols having meanings explained in earlier sections. From Table 5.3.1, we notice that for any arbitrary three-variable function $f(y_1, y_2, y_3)$, the general form of the *error part* $\mathcal{E}(y_1, y_2, y_3)$ is $\mathcal{E}(y_1, y_2, y_3) = k_{12}y_1y_2 \oplus k_{23}y_2y_3 \oplus k_{31}y_3y_1 \oplus k_0$ where each of the coefficients $k_{12}, k_{23}, k_{31}, k_0$ may be either 0 or 1.

For any ***even-numbered rule***, denoted by $f_{even}(y_1, y_2, y_3)$, we must necessarily have $k_0 = 0$; in such cases, the error part is

$$\mathcal{E}_{even}(y_1, y_2, y_3) = k_{12}y_1y_2 \oplus k_{23}y_2y_3 \oplus k_{31}y_3y_1$$

and for ***even-numbered rules*** only, we can rewrite the general *error part*, by a small trick, as

$$\mathcal{E}_{even}(y_1, y_2, y_3) = \left[k_{31}y_3 \quad k_{12}y_1 \quad k_{23}y_2\right].\left[y_1 \quad y_2 \quad y_3\right]^{\mathrm{T}} = \boldsymbol{\eta}(f_{even}).Y^{\mathrm{T}}$$

where $\boldsymbol{\eta}(f_{even}) \equiv \left[k_{31}y_3 \quad k_{12}y_1 \quad k_{23}y_2\right]$. Table 7.1 follows directly from Table 5.3.1.

Evidently, this enables us to write $\mathbf{E}(X^t) = \boldsymbol{M}.X^t$ where $\boldsymbol{M}$ is an $n \times n$ tri-diagonal matrix whose ith row is determined by $\boldsymbol{\eta}(f^i)$, for *even-numbered rules only*. This $\boldsymbol{M}$ may be called the *Modifying matrix*. For such rules,

$$X^{t+1} = J.X^t \oplus \mathbf{E}(X^t) = J.X^t \oplus \boldsymbol{M}.X^t = (J \oplus M).X^t = J_M.X^t.$$

Table 7.1
Forms of η-function for different subclasses of even three-variable rules.

Row no.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Classes /subclasses to which the even rules belong	**0, 1 : 7**	**1 : 6, 2 : 1***	**1 : 5, 2 : 2***	**1 : 4, 2 : 3***	**1 : 3, 2 : 4***	**1 : 2, 2 : 5***	**1 : 1, 2 : 6***	**3 : 0*, 2 : 7***
Form of $\boldsymbol{\eta}(f_{even})$	$[0\ \ 0\ \ 0]$	$[0\ \ y_1\ \ 0]$	$[y_3\ \ 0\ \ 0]$	$[y_3\ \ y_1\ \ 0]$	$[0\ \ 0\ \ y_2]$	$[0\ \ y_1\ \ y_2]$	$[y_3\ \ 0\ \ y_2]$	$[y_3\ \ y_1\ \ y_2]$

Here, J_M may be referred to as the *modified Jacobian matrix*, which is another $n \times n$ tri-diagonal matrix whose ith row is determined by $\text{grad}(f^i) \oplus \eta(f^i)$. For an odd-numbered rule $f_{odd}(y_1, y_2, y_3)$,

$$\begin{aligned}\mathcal{E}_{odd}(y_1, y_2, y_3) &= k_{12}y_1y_2 \oplus k_{23}y_2y_3 \oplus k_{31}y_3y_1 \oplus 1 = \begin{bmatrix} k_{31}y_3 & k_{12}y_1 & k_{23}y_2 \end{bmatrix}.\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}^{\mathrm{T}} \oplus 1 \\ &= \eta\left(\bar{f}_{odd}\right).Y^{\mathrm{T}} \oplus 1\end{aligned}$$

where $\eta\left(\bar{f}_{odd}\right)$ is the Boolean complement of f_{odd}, which is definitely an even rule. Thus for a UCA with an odd rule, $E(X^t) = \boldsymbol{M}.X^t \oplus 1_n$ where 1_n denotes the vector $[\underbrace{1\,1\,1\ldots1}_{n \text{ entries}}]^{\mathrm{T}}$. In general, for any CA (uniform/hybrid), we can write $X^{t+1} = J_M.X^t \oplus C_n$ where C_n is an n-element vector with constant entries (0 or 1).

8. Conclusion and future research directions

This paper characterizes the STDs of one-dimensional CA rules using calculus in digital domain. The study can hopefully be easily extended to arbitrary n-variable Boolean functions. Further we have introduced new ideas on H.D.between two CA rules. Particularly H.D. in a fixed bit position(s) is a new measure to classify Boolean function to study the STD characteristics.

Our current research endeavor focuses on the extraction of useful information on CA properties from the newly introduced modified Jacobian matrix and we have already obtained a few noteworthy results in that direction. For given UCA, we have computed the value of J_M corresponding to each of the input strings and noticed some interesting patterns. There exist subsets of the state space such that the values of J_M for the states in that subset are linear transformations of each other and thus retain some important algebraic properties (rank, determinant etc.); hence, a single constant matrix may be used to represent that particular subset. For example let us consider the Rule 218 UCA4PB, for which $J_M \equiv J$ (as Rule 218 belongs to subclass 1:7). In the case of this UCA, the J_M-matrix for each of the states 0, 1, 2, 4, 8 works out to be $\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$ which is identical to J-matrix of Rule 90 UCA4PB and may thus be written as [90, 90, 90, 90]. It is also interesting to note that Rule 90 is the nearest linear rule of Rule 218 and also that the states 1, 2, 4, 8 show remarkable similarity in behavior in the STD of ⟨218, 218, 218, 218⟩PB. Moreover, for the states 3, 6, 9, 12 (which are again similar in some respects), J_M works out to be [90, 90, 170, 240], [90, 170, 240, 90], [240, 90, 90, 170], [170, 240, 90, 90], respectively, which are clearly obtainable from each other by a simple pre-multiplication with an appropriate permutation matrix (linear operator). Similarly, J_M-values are [150, 90, 150, 90] and [90, 150, 90, 150] respectively for states 5 and 10, [150, 170, 204, 240], [240, 150, 170, 204], [204, 240, 150, 170], [170, 204, 240, 150] respectively for the states 7, 11, 13, and 14, and finally [204, 204, 204, 204] for state 15. Similar such systematic observations are made for other UCAs of other Class 1 rules and also for higher CA lengths—these observations are not likely to be coincidental.

We can thus say that, for non-linear rules, we shall have a set of binary matrices (along with the constant vector C_n) for a given CA rather than a single matrix as in the affine case; thus a non-linear UCA becomes equivalent to a dynamic hybrid linear CA (by *dynamic*, we mean that the rule set applied to the CA can change with time). We are investigating how much light on the CA evolutions (space–time pattern or STD) can be shed by the algebraic properties of this set of binary matrices characteristic of a UCA, for the set of all non-linear rules or any easily recognizable subset thereof. We are concentrating on even non-linear rules only because C_n is identically $[0\ 0\ \ldots\ 0]^{\mathrm{T}}$, giving us a quasi-linear representation of these rules.

Acknowledgements

The authors would like to thank the two anonymous referees for providing helpful comments and valuable criticisms on the original version of the manuscript, which have greatly improved the presentation of this paper.

References

[1] J. von Neumann, in: A.W. Burks (Ed.), The Theory of Self-Reproducing Automata, Univ. of Illinois Press, Urbana, London, 1966.
[2] S. Wolfram, A New Kind of Science, Wolfram publisher, 2002.
[3] S. Wolfram, Theory and Application of Cellular Automata, World Scientific, 1986.
[4] G.Y. Vichniac, Boolean derivatives on cellular automata, Physica D (Nonlinear Phenomena) 45 (1990) 63–74.
[5] K. Dihidar, P.P. Choudhury, Matrix algebraic formulae concerning some special rules of two-dimensional cellular automata, Internat. J. Inform. Sci. 165 (2004) 91–101.

[6] I. Wegener, The complexity of Boolean Functions, Wiley, New York, 1987.
[7] F. Bagnoli, Boolean derivatives and computation of cellular automata, Internat. J. Modern Phys. C 3 (1992) 307.
[8] F. Bagnoli, R. Rechtman, S. Ruffo, Damage spreading and lyapunov exponents in cellular automata, Phys. Lett. A 172 (1992) 34–38.
[9] F. Bagnoli, R. Rechtman, Synchronization and maximum lyapunov exponent in cellular automata, Phys. Rev. E 59 (1999) R1307.