Linear Time Algorithm for Identifying the Invertibility of Null-Boundary Three Neighborhood Cellular Automata

Nirmalya S. Maiti
Soumyabrata Ghosh*
Shiladitya Munshi
P. Pal Chaudhuri
Cellular Automata Research Lab (CARL)
Alumnus Software Limited, Sector V, Kolkata, West Bengal
India 700091
**soumyabrata@alumnux.com*

This paper reports an algorithm to check for the invertibility of null-boundary three neighborhood cellular automata (CAs). While the best known result for checking invertibility is quadratic [1, 2], the complexity of the proposed algorithm is linear. An efficient data structure called a rule vector graph (RVG) is proposed to represent the global functionality of a cellular automaton (CA) by its rule vector (RV). The RVG of a null-boundary invertible CA preserves the specific characteristics that can be checked in linear time. These results are shown in the more general case of hybrid CAs. This paper also lists the elementary rules [3] that are invertible.

1. Introduction

Theory and applications of cellular automata (CAs) were initiated in [4] and carried forward by a large number of authors [3, 5-38]. By convention, a cellular automaton (CA) that employs the same rule for each of its cells is referred to as a uniform CA. However, a large number of authors [13, 14, 16-19, 21, 29, 30, 37, 38] have discussed the hybrid CA concept where different rules are employed in different cells. An n cell hybrid CA (Figure 1) is represented by its rule vector (RV) $\langle R_0, R_1, \ldots, R_i, \ldots, R_{n-1} \rangle$ where the same rule is not employed for each of the cells. For a uniform CA, $R_0 = R_1 = \cdots R_i = \cdots R_{n-1}$. In subsequent discussions, both hybrid and uniform CAs are referred to as simply CAs. For an invertible CA, its global map is invertible. Each of its states is reachable and there is no Garden of Eden (i.e., nonreachable) state. Hence, an invertible CA has exactly one predecessor for a given state. The need to model different real-life physical processes provides the major motivation for investigating the invertibility of CAs.

https://doi.org/10.25088/ComplexSystems.19.1.89

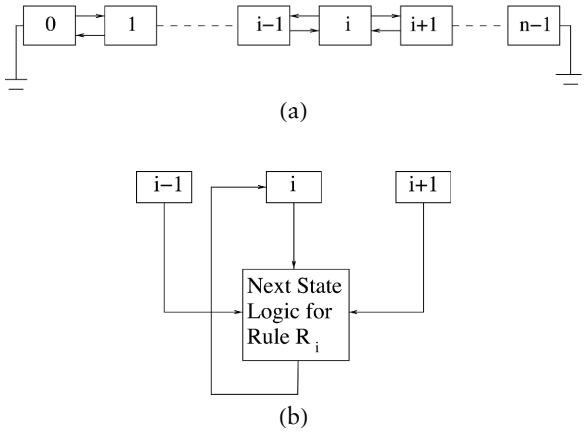


Figure 1. General structure of a CA employing RV $\langle R_0, R_1, \ldots R_i, \ldots R_{(n-1)} \rangle$ of an n cell CA. (a) An n cell null-boundary CA. (b) Rule R_i employed on cell i.

The invertibility issue of CAs was first addressed by Amoroso and Patt [31]. Subsequently, many theoretical works on invertible CAs (ICAs) are reported [1, 2, 9, 11]. Toffoli and Margolus [12] have represented the existence of ICAs that are computation and construction universal. Sutner [1, 2] has utilized the general network of a de Bruijn graph to represent a CA and to identify its invertibility in quadratic time. In this context, this paper reports a linear time algorithm for checking the invertibility of a null-boundary three neighborhood CA. While a de Bruijn graph is a general network with wide applications in different fields, the structure of a rule vector graph (RVG) has been specifically designed to represent CA characteristics. As a result, it has become possible to design a linear time algorithm for traversing a RVG to identify invertibility.

Generating the RVG of a CA from its RV is presented in Section 3, subsequent to introducing a few basic terminologies in Section 2. The linear time algorithm for checking the invertibility of a null-boundary CA is reported in Section 4. This section also lists the elementary rules [3] that are identified as invertible by the algorithm. Experimental results are reported in Section 5 in respect to growth of storage space and execution time for the algorithm. Unless mentioned otherwise, a CA in the rest of the paper refers to a three neighborhood null-boundary CA.

2. Basic Terminologies and Definitions

Generating the next state of a three neighborhood CA rule can be viewed as a three variable function with eight possible input patterns.

https://doi.org/10.25088/ComplexSystems.19.1.89

Borrowing from the concept of switching theory [39], such input patterns (Table 1) are referred to as rule minterms (RMTs). A few basic terminologies and definitions used in this paper are summarized in the rest of this section.

Next State Value b_i								
T(7)	T(6)	T(5)	T(4)	T(3)	T(2)	T(1)	T(0)	Rule Number
7	6	5	4	3	2	1	0	—
1	1	0	0	1	0	1	0	202
1	0	1	0	0	1	1	0	166
0	1	0	1	1	0	1	0	90
0	0	0	1	0	1	0	0	20
0	1	1	1	1	0	0	0	120

Table 1. RMT and CA rule. The left columns represent the next state value b_i of cell i for the present state values $\langle a_{i-1}\, a_i\, a_{i+1}\rangle$ of cells $(i-1)$, i, and $(i+1)$. The eight minterms $\langle a_{i-1}\, a_i\, a_{i+1}\rangle = 000$ to 111 are represented as T(0) to T(7) in the text and 0 to 7 in the figures. The decimal value of the 8-bit binary pattern in the left column, referred to as its rule number, is noted in the right column.

Definition 1. The ordered sequence of rules $\langle R_0\, R_1 \ldots R_i \ldots R_{n-1}\rangle$ of an n cell CA is referred to as its *rule vector* (RV) where rule R_i is employed on cell i (Figure 1). The RV of a uniform CA employs the same rule R for each cell $(R_0 = R_1 = \cdots R_i \cdots = R_{n-1} = R)$ and is represented by the RV $\langle R\, R \ldots R\rangle$.

Definition 2. The rule employed on a CA cell represents a local map, that is, the *local next state function* denoted as f. Thus, f_i represents the local next state function corresponding to the rule R_i employed on cell i.

Definition 3. The *global next state function* F is derived from the local next state functions as $\mathrm{F} = \langle f_0\, f_1 \ldots f_i \ldots f_{n-1}\rangle$ with the three variable Boolean function f_i employed on the current state $\langle a_{i-1}, a_i, a_{i+1}\rangle$ of cells $(i-1)$, i, and $(i+1)$, respectively.

Definition 4. The *global present and next states* are denoted by capital letters A and B, respectively. Thus $\mathrm{A} = \langle a_0, a_1 \ldots a_i \ldots a_{n-1}\rangle$ and $\mathrm{B} = \mathrm{F(A)} = \langle b_0, b_1 \ldots b_i \ldots b_{n-1}\rangle$. That is, B is the successor state of A and A is the predecessor state of B. Consequently, for the invertible CA, $\mathrm{F}'(\mathrm{B}) = \mathrm{A}$, where F' is the inverse of the global next state function F.

https://doi.org/10.25088/ComplexSystems.19.1.89

Both local and global states are noted simply as "state" in the rest of the paper and can be differentiated from the context. For example, as shown later in Figure 2(a), the state B = $\langle b_0\, b_1\, b_2\, b_3 \rangle = (0111) = 7$ of the four cell uniform CA $\langle 120\ 120\ 120\ 120 \rangle$ is the successor of state A = $\langle a_0\, a_1\, a_2\, a_3 \rangle = (0110) = 6$, where the present and next states of the first cell are $a_1 = 1$ and $b_1 = 1$.

Definition 5. For a CA having F as its next state function, a state B is a *nonreachable state* (NRS) if F′(B) = ϕ, where F′ is the inverse of F. That is, there does not exist any predecessor of state B. An NRS is also referred to as a Garden of Eden state.

Definition 6. The eight minterms (Table 1) of the three variable Boolean function f_i, corresponding to rule R_i employed on cell i, are referred to as the *rule minterms* (RMTs). The three Boolean variables are a_{i-1}, a_i, a_{i+1}, the current state values of cells $(i-1)$, i, and $(i+1)$, respectively, whereby the minterm m = $\langle a_{i-1}\, a_i\, a_{i+1} \rangle$. T(m) denotes a single RMT in the text and it is noted simply as m for clarity in the figures. The symbol {T} represents the set of all eight RMTs, whereby {T} = {T(0), T(1), T(2), T(3), T(4), T(5), T(6), T(7)} = {T(m)}. In general, a single RMT for cell i is also denoted as $T^i \in \{T\}$, where $T^i = \langle a_{i-1}, a_i, a_{i+1} \rangle$.

Definition 7. For a specific CA rule R_i, the next state value b_i of cell i is 0 for a subset of RMTs while for the other subset the value is 1. Hence, a CA rule divides the RMTs into two subsets referred to as *0-RMT* and *1-RMT*, respectively, which are denoted as $\{T_0^i\}$ and $\{T_1^i\}$ where $\{T_0^i\} \bigcap \{T_1^i\} = \phi$, $\{T_0^i\} \bigcup \{T_1^i\} = \{T\}$. For the rule 90 (Table 1) on cell i, $\{T_0^i\}$ = {T(7), T(5), T(2), T(0)}, $\{T_1^i\}$ = {T (6), T(4), T(3), T (1)}.

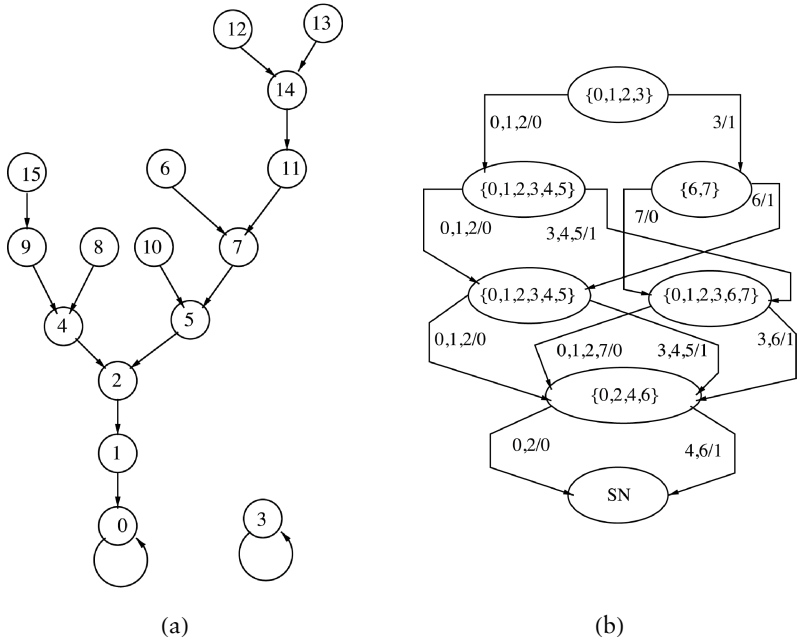
Note that in general, with reference to the next state b_i of cell i, a RMT subset containing q_i' number of RMTs is denoted as $\left\{T_{b_i}^i\right\} = \left\{T_{b_i,1}^i, T_{b_i,2}^i, \ldots T_{b_i,q_i}^i \ldots T_{b_i,q_i'}^i\right\}$, $T_{b_i,q_i}^i \in \left\{T_{b_i}^i\right\}$, $b_i = \{0, 1\}$, $q_i = 1, 2 \ldots q_i'$, $q_i' = \left|\left\{T_{b_i}^i\right\}\right|$ = cardinality of the subset. However, a RMT T_{b_i,q_i}^i is simply noted as $T^i \in \left\{T_{b_i}^i\right\}$ where reference to the next state b_i is not specified. T^i represents a single RMT while a subset of its RMTs denoted as $\{T^i\}$, as elaborated next, is also referred to as $\{V^i\}$.

https://doi.org/10.25088/ComplexSystems.19.1.89

Definition 8. A subset of RMTs $\{T^i\} \subseteq \{\mathrm{T}\}$ of cell i without reference to the next state value b_i is denoted as $\{V^i\}$. The x_i subsets are denoted as $\left\{V^i_{x_i}\right\}$, where $x_i = 1, 2, \ldots x'_i$, $\bigcup_{x_i=1}^{x'_i} \left\{V^i_{x_i}\right\} = \{\mathrm{T}\}$ and x'_i is the number of subsets formed out of the set of RMTs {T}. The symbol {V} representing a subset of RMTs is used to denote a node in the RVG introduced next.

3. Rule Vector Graph

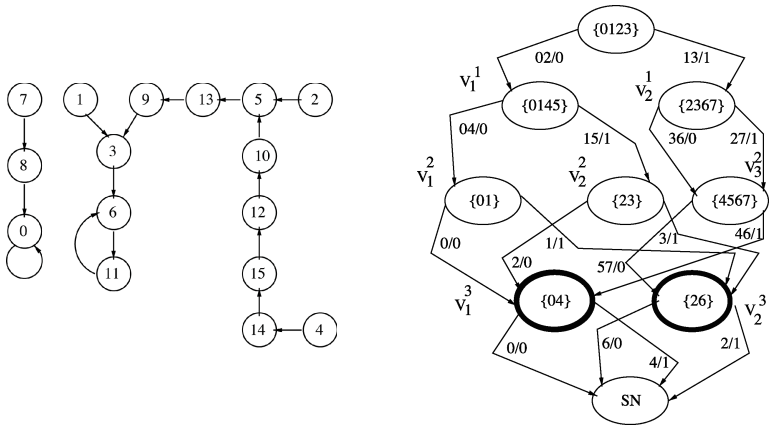
This section presents an efficient data structure called the rule vector graph (RVG) (Figure 2) designed to represent CA characteristics. The preliminary RVG concept was reported in [34-36]. The RVG of an n cell CA denoted by its RV $\langle R_0, R_1 \ldots R_i \ldots R_{n-1}\rangle$ consists of n levels marked as 0 to $(n-1)$; the level i refers to the rule R_i. Each level has a set of input and output nodes connected by directed edges. Figure 2(a) shows the state transition graph (STG) while Figure 2(b) shows the RVG of a uniform CA ⟨120 120 120 120⟩. Figure 3(b) illustrates a RVG for the hybrid CA ⟨202 166 90 20⟩.



(a) (b)

Figure 2. (a) STG and (b) RVG of a four cell CA employing uniform rule 120 represented by its RV ⟨120 120 120 120⟩.

https://doi.org/10.25088/ComplexSystems.19.1.89



(a) (b)

Figure 3. (a) STG and (b) RVG of a CA with RV ⟨202 166 90 20⟩. The binary bit string of a state is denoted by its decimal value. A RMT T(m) is denoted as m (m = 0 to 7) and SN is the sink node. Because of the null boundary, as per Step 3 of Algorithm 2, only even valued RMTs can exist on the third level input nodes V_1^3 and V_2^3.

Definition 9. A *node* represents a subset of RMTs. An output node of level i is derived from its input node through RMT transition (Table 2). The output node of level i is the input node of level $(i+1)$ corresponding to the rule R_{i+1}.

Definition 10. The *edge* of a level represents the RMT transition from input to output node. Let $T^i = \mathrm{T(m)} = \langle a_{i-1}\, a_i\, a_{i+1} \rangle = \langle 1\,0\,1 \rangle$ be a RMT of an input node. Consequently, T^{i+1} (a RMT of an output node) can be derived from T^i as $T^{i+1} = \langle a_i\, a_{i+1}\, a_{i+2} \rangle = \langle 0\,1\,0 \rangle = 2$ and $\langle 0\,1\,1 \rangle = 3$ by deleting a_{i-1} and appending 0 and 1 as a_{i+2}. Table 2 shows this RMT transition for all eight RMTs.

$T^i \langle a_{i-1}\, a_i\, a_{i+1} \rangle$	$T^{i+1} \langle a_i\, a_{i+1}\, a_{i+2} \rangle$	
T(0) (000) and T(4) (100)	T(0) = 000 (0)	T(1) = 001 (1)
T(1) (001) and T(5) (101)	T(2) = 010 (2)	T(3) = 011 (3)
T(2) (010) and T(6) (110)	T(4) = 100 (4)	T(5) = 101 (5)
T(3) (011) and T(7) (111)	T(6) = 110 (6)	T(7) = 111 (7)

Table 2. RMT transition. The left column refers to the RMTs of cell i while the right column refers to the corresponding RMTs of cell $(i+1)$.

https://doi.org/10.25088/ComplexSystems.19.1.89

Definition 11. The *0-edge* and *1-edge* refer to the edges from an input node of level i corresponding to the 0-RMT and 1-RMT for the rule R_i employed on cell i. The b_i-edge ($b_i \in \{0, 1\}$) is assigned the *edge weight* $\left\{T^i_{b_i}\right\}/b_i$, where $\left\{T^i_{b_i}\right\}$ represents the set of RMTs for rule R_i for which the next state value is b_i.

3.1 Generating Rule Vector Graphs for Level *i*

As per the rule R_i, the RMTs of an input node can be divided into two groups referred to as 0-RMT $\left\{T^i_0\right\}$ and 1-RMT $\left\{T^i_1\right\}$ (Definition 7). Two edges marked as 0-edge and 1-edge are derived from an input node corresponding to 0-RMT and 1-RMT respectively.

Each RMT $T^i \in \left\{T^i_{b_i}\right\}$ specifies the decimal value corresponding to the three bit string $\langle a_{i-1}\, a_i\, a_{i+1}\rangle$. An output node $\left\{V^{i+1}_{x_{i+1}}\right\}$ is derived from the RMT subset $\left\{T^i_{b_i}\right\}$. A RMT $T^{i+1} \in \left\{V^{i+1}_{x_{i+1}}\right\}$ is generated, as noted in Table 2, by deleting a_{i-1} from $T^i = \langle a_{i-1}\, a_i\, a_{i+1}\rangle$ and appending 0 and 1 bits as a_{i+2}, whereby $T^{i+1} = \langle a_i\, a_{i+1}\, a_{i+2}\rangle$. The algorithm employed to generate the RVG for level i for rule R_i follows. Such a graph is referred to as the i^{th} *rule vector subgraph* RVS(i).

Algorithm 1: Generate RVS (i) for rule R_i.

We employ constructs such as Draw Edge, Assign Edge Weight, Iterate, Attach Tag, Generate RMTs, Derive Output Node, and Node Merging.

Input: The rule R_i and the input nodes $\left\{V^i_{x_i}\right\}(x_i = 1, 2, \ldots x'_i, x'_i =$ number of input nodes) that are output nodes of level $(i - 1)$ derived for rule R_{i-1}.

Output: The i^{th} level RVG with its edges and output nodes that serve as input nodes for level $(i + 1)$.

Step 1:

(a) Draw Edge: b_i-edge ($b_i \in \{0, 1\}$) for rule R_i out of each input node.

(b) Assign Edge Weight: $\left\{T^i_{b_i}\right\}/b_i \rightarrow b_i$-edge, ($b_i \in \{0, 1\}$).

Step 2: Iterate Steps 3 through 6 for each edge.

Step 3:

(a) Attach Tag: "SLS Tag" $\rightarrow$ edge with weight $\left\{T^i_{b_i}\right\}/b_i$ if $b_i = a_i$, where $T^i = \langle a_{i-1}\, a_i\, a_{i+1}\rangle$, $\left(T^i \in \left\{T^i_{b_i}\right\}\right)$.

/ The edge with the SLS Tag is referred to as the potential self loop state edge (PSLSE) and can be used to identify all self loop states (SLSs). */*

https://doi.org/10.25088/ComplexSystems.19.1.89

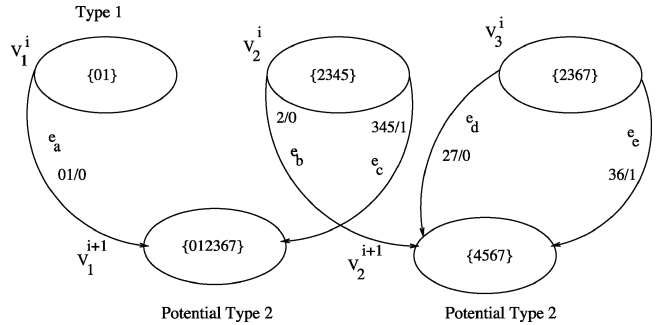


Figure 4. RVS for rule 120. A RMT $T(m)$ is simply denoted as m and the punctuation symbol "," between two RMTs is removed for clarity. {012367} and {4567} are Potential Type 2 nodes with reference to edges e_a and e_b, respectively, and {0, 1} is a Type 1 node due to the missing 1-edge.

(b) Attach Tag: "Type 1" → Input node, if either the 0- or 1-edge outgoing from the node is missing.

Step 4: Generate RMTs: $\{T^{i+1}\}$ for each $T^i \in \{T^i_{b_i}\}$, in the output node as per Table 2.

Step 5: Derive Output Node: $\{V^{i+1}_{x_{i+1}}\}$, $x_{i+1} = 1, 2, \ldots$, where $\bigcup\{T^{i+1}\} = \{V^{i+1}_{x_{i+1}}\}$.

Step 6:

(a) Node Merging: Merge V' with V if $V' \subseteq V$, where V and V' are the output nodes generated in Step 3.

(b) Attach Tag: "Potential Type 2" → Merged node V, if $V' \subset V$.

Stop.

Step by step execution of Algorithm 1:

We now give an example to illustrate the execution of Algorithm 1 for rule 120, which has the binary string 01111000 (Table 1). Figure 4 shows the RVS(i) for rule $R_i = 120$. There are three input nodes: $V_1^i = \{\text{T}(0), \text{T}(1)\}$, $V_2^i = \{\text{T}(2), \text{T}(3), \text{T}(4), \text{T}(5)\}$, and $V_3^i = \{\text{T}(2), \text{T}(3), \text{T}(6), \text{T}(7)\}$. As per Step 1, there is only one edge with weight {T(0), T(1)}/0 out of input node V_1^i. In the first iteration (Step 2) this edge is marked as Type 1 as per Step 3(b) because the outgoing 1-edge is missing from this input node. As per Steps 4 and 5, the {T(0), T(1)}/0 edge (e_a) generates the output node {T(0), T(1), T(2), T(3)} (not shown explicitly in Figure 4) since it gets merged in Step 6(a). In subsequent iterations, from $\{V_2^i\} = \{\text{T}(2), \text{T}(3), \text{T}(4), \text{T}(5)\}$ two edges e_b and e_c are drawn with the edge weights {T(2)}/0 and {T(3), T(4), T(5)}/1, respectively. e_c gen-

https://doi.org/10.25088/ComplexSystems.19.1.89

erates the output node $V_1^{i+1} = \{T(0), T(1), T(2), T(3), T(6), T(7)\}$. Since the output node out of $e_a\,\{T(0), T(1), T(2), T(3)\} \in V_1^{i+1}$, it is merged with V_1^{i+1}. The merging is represented by e_a entering into $V_1^{i+1} = \{T(0), T(1), T(2), T(3), T(6), T(7)\}$. Similarly, edge e_b with weight $\{T(2)\}/0$ generates $\{T(4), T(5)\}$, which gets merged with $V_2^{i+1} = \{T(4), T(5), T(6), T(7)\}$ generated out of e_d from V_3^i. Hence, as per Step 6(b), V_1^{i+1} and V_2^{i+1} are marked as Potential Type 2 nodes. The edge e_e coming out from node V_3^i generates $V_5^{i+1} = \{T(4), T(5), T(6), T(7)\}$ that gets merged with V_2^{i+1}.

A few terminologies relevant for subsequent discussions are introduced next.

Definition 12. The CA state can be expressed as a *RMT string* $\langle T^0\, T^1 \ldots T^i \ldots T^{n-1}\rangle$ where $T^i \in \{T\} = \{T(0)\, T(1)\, T(2)\, T(3)\, T(4)\, T(5)\, T(6)\, T(7)\}$, and $T^i = \langle a_{i-1}\, a_i\, a_{i+1}\rangle$. Thus T^i denotes the decimal value of the bit string $\langle a_{i-1}\, a_i\, a_{i+1}\rangle$, where a_{i-1}, a_i, and a_{i+1} represent the current state of the cells $(i-1)$, i, and $(i+1)$, respectively. For a null-boundary CA, the left neighbor of the cell 0 and the right neighbor of the cell $(n-1)$ of a CA are assumed to be null (0). Consequently, $T^0 = \langle 0\, a_0\, a_1\rangle$, and $T^{n-1} = \langle a_{n-2}\, a_{n-1}\, 0\rangle$. For example, the state of a four bit string 1010 of a four cell can be denoted as the RMT string $\langle T^0\, T^1\, T^2\, T^3\rangle = \langle T(2)\, T(5)\, T(2)\, T(4)\rangle$ since $010 = 2$ for the leftmost cell (cell 0), $101 = 5$ for cell 1, $010 = 2$ for cell 2, and finally $100 = 4$ for the rightmost cell (cell 3). Conversely, a CA state expressed as a RMT string, say, $\langle T(1)\, T(3)\, T(6)\, T(4)\rangle$, can be converted to a binary string $\langle 0\, 1\, 1\, 0\rangle = 6$.

Definition 13. A pair of RMTs T^i and T^{i+1} in a RMT string $\langle \ldots T^{i-1}\, T^i\, T^{i+1} \ldots\rangle$ (where $T^i = \langle a_{i-1}\, a_i\, a_{i+1}\rangle$ and $T^{i+1} = \langle a'_i\, a'_{i+1}\, a'_{i+2}\rangle$) are *compatible* if (*i*) $a_i = a'_i$ and (*ii*) $a_{i+1} = a'_{i+1}$. The pair T^i and T^{i+1} is referred to as *incompatible* if these are not compatible.

Definition 14. A RMT string $\langle T^0 \ldots T^{i-1}\, T^i\, T^{i+1} \ldots T^{n-1}\rangle$ representing the state of a CA is a *valid RMT string* if each pair T^i and T^{i+1} ($i = 0$ to $(n-2)$) is a compatible RMT pair.

Definition 15. For a null-boundary CA, the leftmost cell (cell 0) can have RMTs T(0), T(1), T(2), T(3). Consequently, the input node for level 0, $\{T(0), T(1), T(2), T(3)\}$ is referred to as the *root node* (RN). The edges outgoing from the RN can have only the RMTs T(0), T(1), T(2), T(3). For a null-boundary n cell CA, cell $(n-1)$ can have the value $\{T(0), T(2), T(4), T(6)\}$. Consequently, input nodes and

https://doi.org/10.25088/ComplexSystems.19.1.89

the edges on level $(n-1)$ can have only the RMTs T(0), T(2), T(4), T(6) where m is even for the RMT T(m). The output node of level $(n-1)$ is marked as the *sink node* (SN).

Definition 16. Path, subpath, and parallel subpath.

(a) Path in a RVG. A path in a RVG is a sequence of edges from the RN to the SN. The level i edge of a RVG is marked with the weight $\left\{T^i_{b_i}\right\}/b_i$ where $b_i \in \{0, 1\}$, $\left\{T^i_{b_i}\right\} = \left\{T^i_{b_i 1}, T^i_{b_i 2} \ldots T^i_{b_i q_i} \ldots T^i_{b_i q'_i}\right\}$, and $T^i_{b_i q_i} \in \left\{T^i_{b_i}\right\} \subseteq \{\mathrm{T}\} =$ {T(7), T(6), T(5), T(4), T(3), T(2), T(1), T(0)}, $q_i = 1, 2 \ldots q'_i$ and q'_i is the cardinality of RMTs in the set $\left\{T^i_{b_i}\right\}$. Thus, a path in the RVG of an n cell CA can be denoted as a set of sequential weighted edges $\left\{\left\{T^0_{b_0}\right\}/b_0, \left\{T^1_{b_1}\right\}/b_1, \ldots \left\{T^i_{b_i}\right\}/b_i, \ldots \left\{T^{n-1}_{b_{n-1}}\right\}/b_{n-1}\right\} = \left\{\left\{T^i_{b_i}\right\}/b_i\right\}$ for $i = 0$ to $(n-1)$, where the RMT $T^i \in \left\{T^i_{b_i}\right\}$ is compatible with $T^{i+1} \in \left\{T^{i+1}_{b_{i+1}}\right\}$.

(b) Path representing a CA state. The set of sequential edges corresponding to a valid RMT string $\left\langle T^0_{b_0 q_0}, T^1_{b_1 q_1}, \ldots T^i_{b_i q_i}, \ldots T^{n-1}_{b_{n-1} q_{n-1}}\right\rangle$ $\left(T^i_{b_i q_i} \in \left\{T^i_{b_i}\right\}\right)$ on the path from the RN to the SN represents a CA state. The global state $\mathrm{A} = \langle a_0, a_1, \ldots a_i, \ldots a_{n-1}\rangle$ can be derived by converting the RMT string to its binary counterpart (Definition 12). Thus, $\mathrm{A} = \left\langle T^0_{b_0 q_0}, T^1_{b_1 q_1}, \ldots T^i_{b_i q_i}, \ldots T^{n-1}_{b_{n-1} q_{n-1}}\right\rangle = \langle a_0\, a_1 \ldots a_i \ldots a_{n-1}\rangle$. Its next state $\mathrm{B} = \langle b_0, b_1, \ldots b_i, \ldots b_{n-1}\rangle$ can be derived from the path $\left\{\left\{T^i_{b_i}\right\}/b_i\right\}$. Consequently, forward traversal of the path $\left\{\left\{T^i_{b_i}\right\}/b_i\right\}$ for a state $\mathrm{A} = \langle a_0\, a_1, \ldots a_i, \ldots a_{n-1}\rangle = \langle T^0\, T^1 \ldots T^i \ldots T^{n-1}\rangle$ generates its next state $\mathrm{B} = \langle b_0\, b_1, \ldots b_i \ldots b_{n-1}\rangle$. Similarly, backward traversal with B generates its previous state(s) as $\langle T^0\, T^1 \ldots T^i \ldots T^{n-1}\rangle$. The state B is a NRS if the path between the root and sink nodes cannot be established due to missing edges or incompatible RMT pairs T^i and T^{i+1}.

(c) Subpath. A subpath refers to a section of a path $\left\langle\left\{T^{i-1}_{b_{i-1}}\right\}/b_{i-1} \left\{T^i_{b_i}\right\}/b_i \left\{T^{i+1}_{b_{i+1}}\right\}/b_{i+1} \ldots \left\{T^j_{b_j}\right\}/b_j\right\rangle$ from level $(i-1)$ to level j of a RVG that starts with a RMT $T^{i-1} \in \left\{T^{i-1}_{b_{i-1}}\right\}$.

(d) Parallel subpath. A subpath $\left\langle\left\{(T')^{i+1}_{b_{i+1}}\right\}/b_{i+1} \left\{(T')^{i+2}_{b_{i+2}}\right\}/b_{i+2} \ldots\right\rangle$ is parallel to the subpath $\left\langle\left\{T^{i+1}_{b_{i+1}}\right\}/b_{i+1} \left\{T^{i+2}_{b_{i+2}}\right\}/b_{i+2} \ldots\right\rangle$ if both have

https://doi.org/10.25088/ComplexSystems.19.1.89

identical next state values b_{i+1}, $b_{i+2} \ldots b_{n-1}$ at each level from level $(i+1)$ to level $(n-1)$ of a RVG.

With reference to the node merging noted in Step 6 of Algorithm 1, the following definitions are formally introduced after defining a path and subpath in a RVG.

Definition 17. Let V and V' be a pair of output nodes generated by Algorithm 1, where each node covers a subset of RMTs. The node V' gets merged with the node V if $V' \subseteq V$. The resulting node V is referred to as a *merged node* (Figure 5).

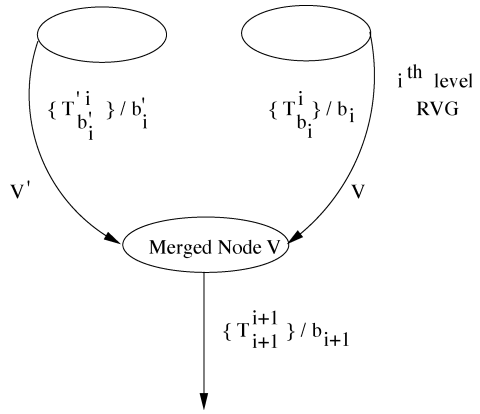


Figure 5. Node merging.

Definition 18. A node V is marked with a Type 1 tag if there is a missing 0/1-edge outgoing from the node. The node V_1^i is a Type 1 node in Figure 4.

Definition 19. A Type 2 node satisfies the following two conditions.

Condition (i). A merged node V (Figure 5) is marked with a Potential Type 2 tag if the merging has occurred for two nodes V' and V $(V' \subset V)$, where V' is generated out of RMTs $\left\{(T')^i_{b'_i}\right\}$ of the edge having a weight of $\left\{(T')^i_{b'_i}\right\}\Big/ b'_i$ and V is generated out of RMTs $\left\{T^i_{b_i}\right\}$ of the edge having a weight of $\left\{T^i_{b_i}\right\}\Big/ b_i$, $(b_i \in \{0, 1\}, b'_i \in \{0, 1\})$ where $\left|\left\{T^i_{b_i}\right\}\right| > \left|\left\{(T')^i_{b'_i}\right\}\right|$. A Potential Type 2 node is noted with reference to the edge having a weight of $\left\{(T')^i_{b'_i}\right\}\Big/ b'_i$ that has a smaller number of RMTs. In Figure 4, the node {T(4), T(5), T(6), T(7)} is a Potential Type 2 node with reference to the edge e_b having weight T(2)/0 that

https://doi.org/10.25088/ComplexSystems.19.1.89

has fewer RMTs than the other incoming edges with weights {T(2), T(7)}/0 and {T(3),T(6)}/1.

Condition (ii). A Potential Type 2 node V (at output level i) is marked as a Type 2 node if (1) a subpath (Definition 16(c)) can be identified from the node to the SN starting with a RMT $T^{i+1} \in \{V - V'\}$; and (2) no parallel subpath (Definition 16(d)) exists starting with a RMT $T'^{i+1} \in V'$.

Condition (ii) of Definition 19 regarding Type 2 nodes is illustrated later in Section 3.2 Figure 6(b).

3.2 Generating Rule Vector Graphs

The algorithm for generating RVGs, presented next, employs Algorithm 1 for each of the n levels for an n cell null-boundary CA.

Algorithm 2: Generate RVG.

Input: The RV $\langle R_0\, R_1 \ldots R_i \ldots R_{n-1} \rangle$ of an n cell CA.

Output: The RVG.

Step 0:

(a) Generate Root Node: Mark {T(0), T(1), T(2), T(3)} as the RN (the input node for level 0).

(b) Generate RVS(0): Execute Algorithm 1 to generate the level 0 RVS for rule R_0.

Step 1: Iterate Step 2 for $i = 1$ to $(n - 2)$.

Step 2:

(a) Mark Input Node(i): Output Node($i - 1$)→Input Node(i).

(b) Generate RVS(i): Execute Algorithm 1 for rule R_i and mark "Type 1" and "Potential Type 2" nodes.

(c) Check Status: Condition (ii) of "Type 2" node (Definition 19) at each level of the RVG until the SN is reached.

Step 3:

(a) Delete odd valued RMTs from the level $(n - 1)$ input nodes.

(b) Assign Edge Weight: As per rule $R_{(n-1)}$ to $(n - 1)$ level edges (from input node $(n - 1)$ to the output node marked as SN).

(c) Mark "Potential Type 2" node as "Type 2" if Condition (ii) is true for the node.

Stop.

Step by step execution of Algorithm 2:

Figure 6(b) illustrates the execution of Algorithm 2 for the four cell null-boundary CA with RV ⟨12 162 166 80⟩. As per Step 0, the input node {T(0), T(1), T(2), T(3)} is the RN and two edges are drawn for Rule 12 as {T(0), T(1)} / 0 and {T(2), T(3)} / 1 generating the level 0 output nodes as {T(0), T(1), T(2), T(3)} and {T(4), T(5), T(6), T(7)}. These two nodes, as per Step 2(a), are the input nodes of the first level.

https://doi.org/10.25088/ComplexSystems.19.1.89

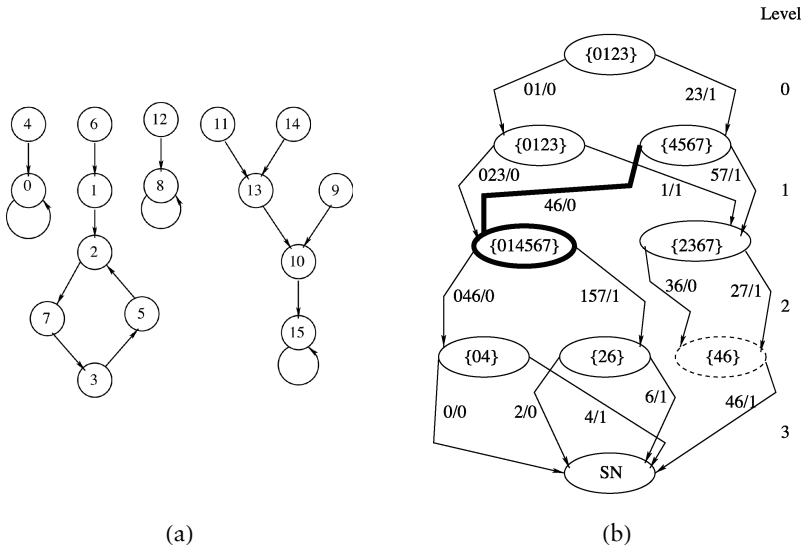


Figure 6. (a) STG and (b) RVG for a four cell CA with RV ⟨12 162 166 80⟩.

In the next two iterations, the first and second levels of the RVG for rules 162 and 166 are generated.

Odd valued RMTs are deleted, as per Step 3(a), from the third level input nodes to generate the nodes {T(0), T(4)}, {T(2), T(6)}, {T(4), T(6)}. Edge weights are next assigned as per Step 3(b) for $R_3 = 80$ to the edges input to the SN.

The Type 1 node {T(4), T(6)} (level three input node) and the Potential Type 2 nodes {T(0), T(1), T(4), T(5), T(6), T(7)} and {T(2), T(3), T(6), T(7)} (level two input nodes) are identified as per Step 2(b) while executing Algorithm 1. The first node is marked by a bold outline in Figure 6(b) with reference to the edge having weight {T(4), T(6)}/0 (also marked with a bold line).

The subpath starting from the Potential Type 2 to the SN is checked in Step 2(c) until the SN is reached. The second level input node {T(0), T(1), T(4), T(5), T(6), T(7)} satisfies both Conditions (i) and (ii) of Type 2 nodes (Definition 19) where $V = \{T(0), T(1), T(4), T(5), T(6), T(7)\}$, $V' = \{T(0), T(1), T(4), T(5)\}$, and $\{V - V'\} = \{T(6), T(7)\}$. This is true since the subpath ⟨T(6)/0 T(4)/1⟩ starting with RMT $T(6) \in \{V - V'\}$ from the Potential Type 2 node to the SN through levels two and three is a unique subpath with no parallel counterpart through these two levels. As a result, the state 9 ⟨1 0 0 1⟩ through the Potential Type 2 node {T(0), T(1), T(4), T(5), T(6), T(7)} (with reference to the bold line edge {T(4), T(6)}/0) and the path ⟨{T(2), T(3)}/1, {T(4), T(6)}/0, {T(0), T(4), T(6)}/0, {T(4)}/1⟩ is a NRS. Similarly, the subpath ⟨T(7)/1 T(6)/1⟩ starting with RMT $T(7) \in \{V - V'\}$ (of a Potential

https://doi.org/10.25088/ComplexSystems.19.1.89

Type 2 node) is another unique subpath resulting in the state 11 ⟨1 0 1 1⟩ as a NRS.

However, Condition (ii) is not satisfied for the Potential Type 2 node {T(2), T(3), T(6), T(7)}, where $V = \{T(2), T(3), T(6), T(7)\}$, $V' = \{T(2), T(3)\}$, and $\{V - V'\} = \{T(6), T(7)\}$. This is true since there exists the following parallel subpaths through the second level edges {T(3) T(6)}/0 and {T(2) T(7)}/1:

(a) Subpath ⟨T(3)/0 T(6)/1⟩ starting with RMT $T(3) \in V'$ is parallel to subpath ⟨T(6)/0 T(4)/1⟩ that starts with RMT $T(6) \in \{V - V'\}$. As a result, there exists the predecessor state $3 = \langle 0\,0\,1\,1\rangle = \langle T(0)\,T(1)\,T(3)\,T(6)\rangle$ for the state $5 = \langle 0\,1\,0\,1\rangle$ on the path ⟨{T(0) T(1)}/0, {T(1)}/1, {T(3), T(6)}/0, {T(4), T(6)/1}⟩ from the RN to the SN through the Potential Type 2 node {T(2), T(3), T(6), T(7)}.

(b) Similarly, the subpath ⟨T(2)/1 T(4)/1⟩ starting with RMT $T(2) \in V'$ is parallel to the subpath ⟨T(7)/1 T(6)/1⟩ that starts with RMT $T(7) \in \{V - V'\}$. As a result, there exists the predecessor state $2 = \langle 0\,0\,1\,0\rangle = \langle T(0)\,T(1)\,T(2)\,T(4)\rangle$ for the state $7 = \langle 0\,1\,1\,1\rangle$ on the path ⟨{T(0) T(1)}/0, {T(1)}/1, {T(2), T(7)}/1, {T(4), T(6)}/1⟩ from the RN to the SN through the Potential Type 2 node {T(2), T(3), T(6), T(7)}.

Hence, the Potential Type 2 node {T(2), T(3), T(6), T(7)} is not a Type 2 node.

The input node in the third level {T(4), T(6)} is a Type 1 node marked with a broken outline because the 0-edge is missing for this node.

3.3 Time and Space Complexity of Generating Rule Vector Graphs for Null-Boundary Cellular Automata

The time complexity of Algorithm 2 for generating a RVG for a null-boundary CA is clearly linear with each rule R_i of RV $\langle R_0\,R_1 \ldots R_i \ldots R_{n-1}\rangle$ being processed once to generate the level i RVG ($i = 0, 1, 2, \ldots (n-1)$). The space complexity, as shown in Lemma 1, is also linear because of node merging (Definition 17).

Lemma 1. The maximum number of output nodes at any level of a RVG is six.

Proof. The set of output nodes at any level are derived as per Table 2 from the RMTs specified in the set $\left\{T^i_{b_i}\right\}$ noted on the i^{th} level b_i-edge ($b_i \in \{0, 1\}$) having edge-weight $\left\{T^i_{b_i}\right\}/b_i$. The RMTs in the output nodes, as shown in Table 2, always appear in pairs (one even and one odd). The four pairs are (T(0), T(1)), (T(2), T(3)), (T(4), T(5)), (T(6), T(7)). Hence, due to node merging (Definition 17), the maximum number of possible output nodes are $4_{C_2} = 6$, which are {T(0), T(1), T(2), T(3)}, {T(4), T(5), T(6), T(7)}, {T(0), T(1), T(4), T(5)},

https://doi.org/10.25088/ComplexSystems.19.1.89

{T(2), T(3), T(6), T(7)}, {T(0), T(1), T(6), T(7)}, and {T(2), T(3), T(4),T(5)}. □

Figure 7 displays the RVG of a five cell null-boundary CA (with RV ⟨12 202 166 90 20⟩) having six input nodes at the fourth level. The odd valued RMTs are deleted as per Step 3(a) of Algorithm 2.

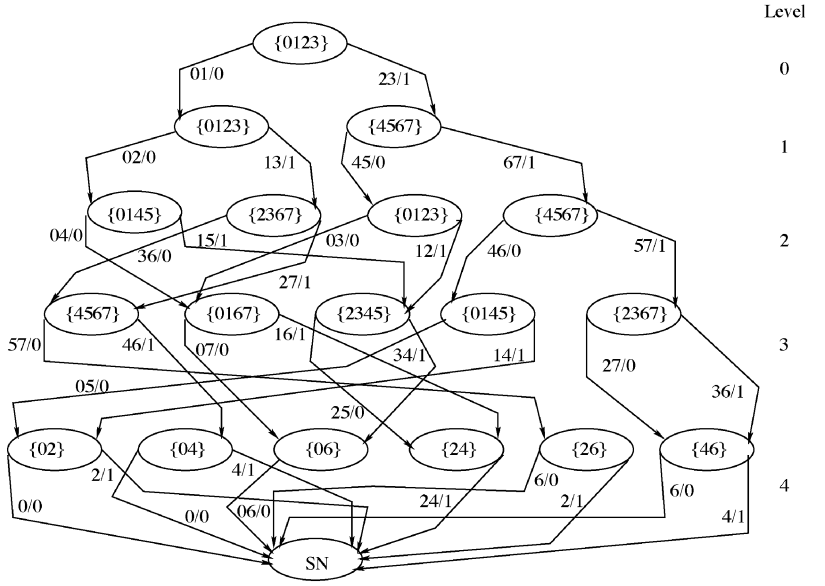


Figure 7. An illustration of six nodes in a level with a RVG for a five cell CA with RV ⟨12 202 166 90 20⟩.

4. Linear Time Algorithm for Identifying Invertibility

This section reports a linear time algorithm to identify the invertibility of a null-boundary three neighborhood CA. Theorems 1 and 2 establish the fact that the presence of Type 1 or 2 nodes in the RVG of a CA makes it noninvertible.

Theorem 1. If the RVG of a null-boundary CA has a Type 1 node, it is a noninvertible CA.

Proof. As discussed in Definition 18, a Type 1 node is identified by a missing outgoing b_i-edge ($b_i \in \{0, 1\}$) (Definition 11). Hence the path $\left\{\left\{T^i_{b_i}\right\}/b_i\right\}$ for the state $\langle b_0\, b_1, \ldots b_i, \ldots b_{n-1}\rangle$ from the RN to the SN cannot be established through the Type 1 node due to a missing edge. Consequently, such a state is nonreachable. The presence of a NRS

https://doi.org/10.25088/ComplexSystems.19.1.89

having no pre-image makes the CA noninvertible. Hence, the CA having a Type 1 node in its RVG is noninvertible. □

Figure 8(b) illustrates the RVG of a noninvertible CA. The input node {T(2) T(3)} of level 1 is a Type 1 node, as it does not have an outgoing 1-edge. Hence, all the states corresponding to nonexistent paths passing through the missing 1-edge are nonreachable. For example, the path for state B = $\langle b_0\, b_1\, b_2\, b_3 \rangle = \langle 0\ 1\ 0\ 0 \rangle$ does not exist and consequently the state 4 is nonreachable. Similarly, three other states 0101(5), 0110(6), and 0111(7) with $b_1 = 1$ are nonreachable (Figure 8(a)). Hence, the CA ⟨13 112 196 64⟩ is noninvertible.

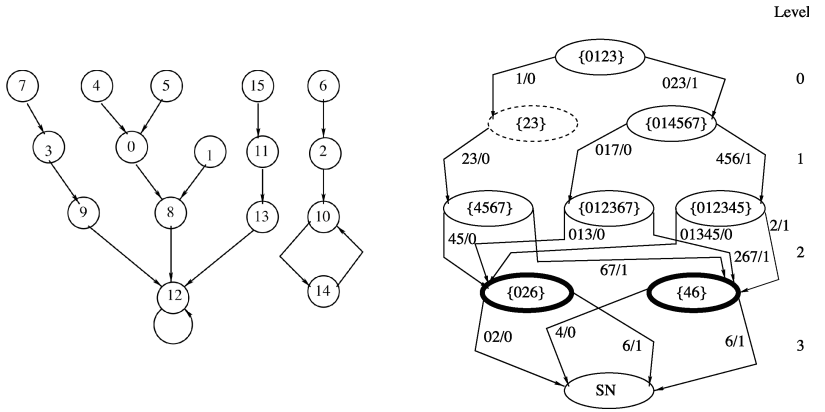


Figure 8. (a) STG and (b) RVG for a four cell CA with RV ⟨13 112 196 64⟩. {23} is a Type 1 node, as marked with a broken outline. {46} is a Type 2 node (marked with a bold line) with reference to edge 2/1 (also marked with a bold line).

A path in a RVG (Definition 16(b)) identifies a state A and its successor B. Lemma 2 specifies the condition for which a state is nonreachable, even though there is no missing 0- or 1-edge from a node in the RVG.

Lemma 2. A NRS exists in the STG of a CA if no valid path exists in the RVG of the CA.

Proof. For each reachable state of a CA there exists a path $\left\langle \left\{T^0_{b_0}\right\} \middle/ b_0 \left\{T^1_{b_1}\right\} \middle/ b_1 \ldots \left\{T^i_{b_i}\right\} \middle/ b_i \left\{T^{i+1}_{b_{i+1}}\right\} \middle/ b_{i+1} \ldots \left\{T^{n-1}_{b_{n-1}}\right\} \middle/ b_{n-1} \right\rangle$, where each $T^{i+1} \in \left\{T^{i+1}_{b_{i+1}}\right\}$ is compatible with $T^i \in \left\{T^i_{b_i}\right\}$. Consequently, there exists a predecessor state A $\langle a_0\, a_1 \ldots a_i\, a_{i+1} \ldots a_{n-1} \rangle = \langle T^0\, T^1 \ldots T^i\, T^{i+1} \ldots T^{n-1} \rangle$ of the state $\langle b_0\, b_1 \ldots b_i\, b_{i+1} \ldots b_{n-1} \rangle$ = B. On the other hand, if T^i and T^{i+1} are incompatible (Definition 13)

https://doi.org/10.25088/ComplexSystems.19.1.89

then the path is invalid. Consequently, there exists no predecessor for the state $\langle b_0\, b_1 \ldots b_i\, b_{i+1} \ldots b_{n-1}\rangle$ and it is not reachable from any other state. □

Theorem 2. If the RVG of a null-boundary CA has a Type 2 node, it is a noninvertible CA.

Proof. As per Condition (i) (Definition 19), a Type 2 node V is generated due to merging output nodes V and V' of level i (Figure 5) where $V' \subset V$. The node V' is generated out of $\left\{(T')^i_{b'_i}\right\}$ while V is generated out of $\left\{T^i_{b_i}\right\}$ as per RMT transition of Table 2. Since $V' \subset V$, $\left|\left\{(T')^i_{b'_i}\right\}\right| < \left|\left\{T^i_{b_i}\right\}\right|$. Consequently, at least two RMTs exist in $(V - V')$.

Let an edge from level $(i+1)$ be denoted as $\left\{T^{i+1}_{b_{i+1}}, (T')^{i+1}_{b_{i+1}}\right\}\Big/b_{i+1}$, where $T^{i+1}_{b_{i+1}} \in (V - V')$, while $(T')^{i+1}_{b_{i+1}} \in V'$ and $(T')^{i+1}_{b_{i+1}} \notin (V - V')$.

Condition (ii) (Definition 19) ensures the following:

(a) The presence of a subpath $\left\langle T^{i+1}_{b_{i+1}}\Big/b_{i+1}\; T^{i+2}_{b_{i+2}}\Big/b_{i+2} \ldots\right\rangle$ (Definition 16(c)) from a Potential Type 2 node to the SN.

(b) That there is no subpath $\left\langle (T')^{i+1}_{b_{i+1}}\Big/b_{i+1}\; T^{i+2}_{b_{i+2}}\Big/b_{i+2} \ldots\right\rangle$ parallel to (a) (Definition 16(d)).

If no parallel subpath exists, the path $\left\langle \left\{T^0_{b_0}\right\}\Big/b_0 \left\{T^1_{b_1}\right\}\Big/b_1 \ldots \left\{(T')^i_{b'_i}\right\}\Big/b'_i\; \left\{T^{i+1}_{b_{i+1}}\right\}\Big/b_{i+1} \ldots \left\{T^{n-1}_{b_{n-1}}\right\}\Big/b_{n-1}\right\rangle$ is invalid since the RMT pair $(T')^i_{b'_i}$ and $T^{i+1}_{b_{i+1}}$ is incompatible (Definition 13). This is true since $T^{i+1}_{b_{i+1}}$ is not generated out of $(T')^i_{b'_i}$ through RMT transition of Table 2. As per Lemma 2, the path $\left\langle \left\{T^0_{b_0}\right\}\Big/b_0 \left\{T^1_{b_1}\right\}\Big/b_1 \ldots \left\{(T')^i_{b'_i}\right\}\Big/b'_i \left\{T^{i+1}_{b_{i+1}}\right\}\Big/b_{i+1} \ldots \left\{T^{n-1}_{b_{n-1}}\right\}\Big/b_{n-1}\right\rangle$ through a Potential Type 2 node is invalid and no predecessor exists for the state $\langle b_0\, b_1 \ldots b'_i\, b_{i+1} \ldots b_{n-1}\rangle$. So the state is a NRS and the CA is marked as noninvertible.

On the other hand, if a parallel subpath (Definition 16(d)) exists, we can always identify a valid path $\left\langle \left\{T^0_{b_0}\right\}\Big/b_0 \left\{T^1_{b_1}\right\}\Big/b_1 \ldots \left\{(T')^i_{b'_i}\right\}\Big/b'_i \left\{(T')^{i+1}_{b_{i+1}}\right\}\Big/b_{i+1} \ldots \left\{T^{n-1}_{b_{n-1}}\right\}\Big/b_{n-1}\right\rangle$ from the RN to the SN. This is true since $\left\{(T')^{i+1}_{b_{i+1}}\right\}\Big/b_{i+1} \in V'$, RMTs of V' are generated from $\left\{(T')^i_{b'_i}\right\}$, the node pair T'^i and $T'^{(i+1)}$ is compatible where

https://doi.org/10.25088/ComplexSystems.19.1.89

$T'^{i} \in \left\{(T')^{i}_{b'_i}\right\}$, and $T'^{(i+1)} \in \left\{(T')^{i+1}_{b_{i+1}}\right\}$. As a result there exists a predecessor $\langle b_0\, b_1 \ldots b'_i\, b'_{i+1} \ldots b_{n-1}\rangle$. □

The instances of absence and presence of a parallel subpath from a Potential Type 2 node to the SN have been illustrated in Section 3.2 (Figure 6) with reference to the Potential Type 2 nodes {T(0), T(1), T(4), T(5), T(6), T(7)} and {T(2), T(3), T(6), T(7)}. The first Potential Type 2 node generates a NRS. On the other hand, the presence of a valid path through the second Potential Type 2 node prohibits the generation of a NRS.

Figure 8 illustrates another case of generating a NRS due to the absence of a parallel subpath through a Potential Type 2 node.

The state 1111(15) in Figure 8(a) is a NRS because from the Potential Type 2 node {T(4) T(6)} there is a unique subpath ⟨T(6)/1⟩ to the SN without any parallel subpath (Definition 16(d)). Hence, the node {T(4) T(6)} is a Type 2 node. As a result, the pre-image for the state $15 = \langle 1\,1\,1\,1\rangle$ through the Type 2 node {T(4) T(6)} (input node of level three) does not exist due to the incompatible RMT pair T(2) and T(6) on the path {{T(0) T(2) T(3)} / 1 {T(4) T(5) T(6)} / 1 {T(2)} / 1 {T(6)}/1} from the RN to the SN.

Theorem 3. Necessary and sufficient conditions for the RVG of an invertible CA are that no Type 1 or 2 nodes exist.

Proof. *Necessity*: The presence of a Type 1 or 2 node, as established in the proof of Theorems 1 and 2, makes the CA noninvertible.

Sufficiency: The RVG, as per Algorithm 2, is generated with compatible RMT pairs on incoming and outgoing edges of any node. RMT T^{i+1} on the outgoing edge with weight $\left\{T^{i+1}_{b_{i+1}}\right\}\Big/b_{i+1}$ is generated, as per Table 2, out of $T^{i} \in \left\{T^{i}_{b_i}\right\}$ on the incoming edge with weight $\left\{T^{i}_{b_i}\right\}\Big/b_i$. Consequently, if there is no Type 1 or 2 node, then for each state $\mathrm{B} = \langle b_0\, b_1 \ldots b_i \ldots b_{n-1}\rangle$ there exists a path $\left\langle\left\{T^{0}_{b_0}\right\}\Big/b_0\, \left\{T^{1}_{b_1}\right\}\Big/b_1 \ldots \left\{T^{i}_{b_i}\right\}\Big/b_i \ldots \left\{T^{n-1}_{b_{n-1}}\right\}\Big/b_{n-1}\right\rangle$ generating its previous state $\mathrm{A} = \langle a_0\, a_1 \ldots a_i \ldots a_{n-1}\rangle = \langle T^0\, T^1 \ldots T^i \ldots T^{n-1}\rangle$, where $T^{i} \in \left\{T^{i}_{b_i}\right\}$ and the RMT string is a valid one with only one compatible RMT pair T^i and T^{i+1} ($i = 0$ to $(n-2)$). Hence, each state B has a predecessor A. Since each state has one successor state and there is no NRS, there exists only one pre-image of each state. Hence, the CA is an invertible one. □

A noninvertible CA, as established in Theorem 3, can be identified by the presence of either Type 1 or 2 nodes. Algorithm 3 formalizes the identification procedure.

https://doi.org/10.25088/ComplexSystems.19.1.89

Algorithm 3: Check Invertibility

Input: An n cell CA, $\langle R_0, R_1, \ldots R_i, \ldots R_{n-1} \rangle$.

Output: Identification of Invertibility.

Step 0: Execute Algorithm 2 to generate the RVG of the CA along with identifying Type 1 and 2 nodes, if there are any.

Step 1: Iterate Step 2 for each level ($i = 1$ to $(n-1)$) of the RVG.

Step 2: Check node type for the presence of any Type 1 or 2 nodes.

Step 3: Mark the CA as noninvertible if a Type 1 or 2 node exists.

Otherwise, mark the CA as invertible.

Stop.

Step by step execution of Algorithm 3:

An illustration of Algorithm 3 is shown in Figure 9 for the four cell CA ⟨6 240 60 65⟩. The RVG of the CA (Figure 9(b)) is drawn by executing Algorithm 2. Step by step execution of Algorithm 2 has been illustrated in Section 3.2 (Figure 6(b)). Step 2(b) of Algorithm 2 calls Algorithm 1. Step 3(b) of Algorithm 1 identifies a Type 1 node, while a Potential Type 2 node is identified in Step 6(b).

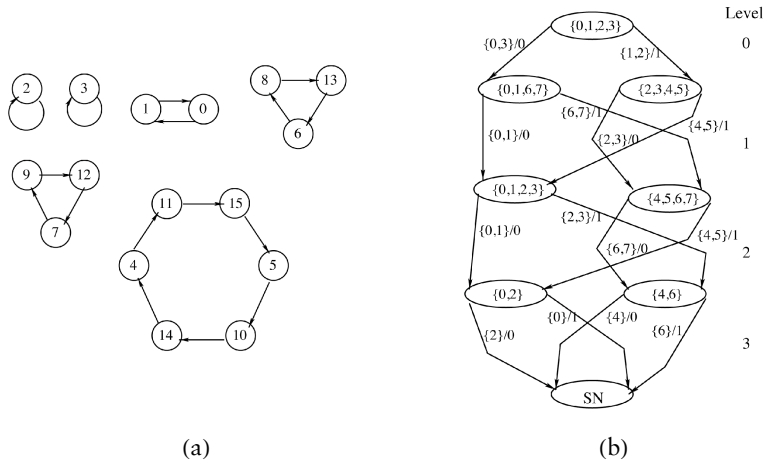


Figure 9. An illustration of Algorithm 3 with a four cell CA ⟨6 240 60 65⟩. (a) The STG and (b) the RVG.

In each iteration of Step 2(c) of Algorithm 2 at each level of the RVG, the status of Condition (ii) of the Type 2 node (Definition 19) is checked. This checking is implemented for each Potential Type 2 node until the SN is reached. Finally, on the last iteration at level $(n-1)$ (Step 3(c) of Algorithm 2), the Potential Type 2 node is marked as a Type 2 node if Condition (ii) is found to be true for a subpath from the Potential Type 2 node to the SN. Thus, in each iteration step of Algorithm 2, the presence of a Type 1 node, if it exists,

https://doi.org/10.25088/ComplexSystems.19.1.89

gets detected. On the other hand, marking a Type 2 node waits for traversal through the RVG until the SN is reached.

Step 2 of Algorithm 3 scans through each level to detect the presence of a Type 1 or 2 node. No Type 1 or 2 node exists in the RVG of Figure 9(b). Hence, it is an invertible CA. Its state transition behavior, as shown in Figure 9(a), consists of six cycles.

The absence of a Type 1 and 2 node, in general, imparts the following characteristic on the RVG: each node at each level i ($i = 0$ to $(n-2)$) of an n cell CA has four RMTs, while for $i = n-1$, the number of RMTs is two.

4.1 Invertible Uniform Cellular Automata and Elementary Rules

In a uniform CA, the same rule is employed for each cell. All 256 CA rules, as per [3], can be divided into 88 groups of elementary rules. Each elementary rule group has been derived as follows. Each group consists of four rules marked as (r_1, r_2, r_3, r_4); the first rule r_1 is the conventional rule number, the second rule r_2 is obtained by interchanging bits 1 and 0, the third rule r_3 is obtained by interchanging the left and right neighbors, while rule r_4 is derived by applying both operations. A few elementary rules are listed in Table 3 as per the formulation noted in [3]. Out of the 88 groups of elementary rules, six groups generate invertible n bit CAs (Table 3). Figures 10, 11, and 12 illustrate the results of Algorithm 3 for $n = 4$, 5, and 6 cell uniform CAs with rule 105. The RVGs shown in Figures 10(b), 11(b), and 12(b) are drawn as per Algorithm 2. The four and six cell CAs, as displayed by the STGs shown in Figures 10(a) and 12(a), are invertible since there is no Type 1 or 2 node in any level of the corresponding RVGs. On the other hand, the five cell CA, as shown in Figure 11(a), is not invertible due to the presence of Type 1 nodes (marked by bold lines in Figure 11(b)) at the fourth level.

Elementary Rule Group	Size for which the CA is Invertible
(51, 51, 51, 51)	for all values of n
(60, 195, 102, 153)	for all values of n
(90, 165, 90, 165)	for even values of n
(105, 105, 105, 105)	for all values of n excepting $n = 2 + 3y (y = 0, 1, 2, 3 \ldots)$
(150, 150, 150, 150)	for all values of n excepting $n = 2 + 3y (y = 0, 1, 2, 3 \ldots)$
(204, 204, 204, 204)	for all values of n

Table 3. Invertible elementary group rules.

https://doi.org/10.25088/ComplexSystems.19.1.89

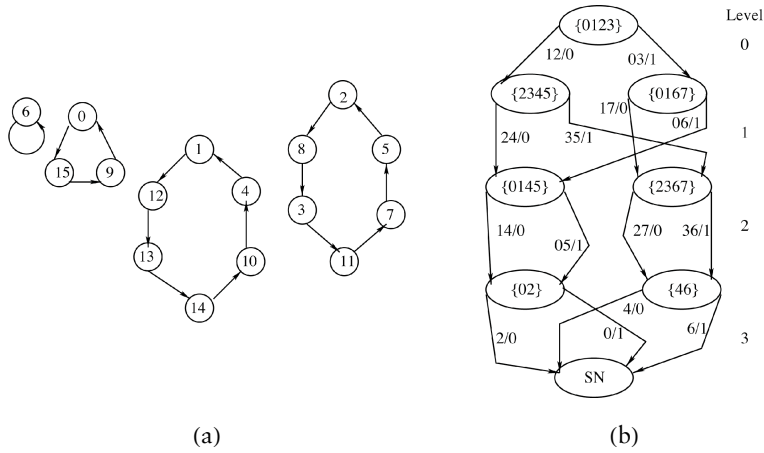


Figure 10. An illustration of Algorithm 3 with a four cell uniform CA with rule ⟨105 105 105 105⟩. (a) The STG and (b) the RVG.

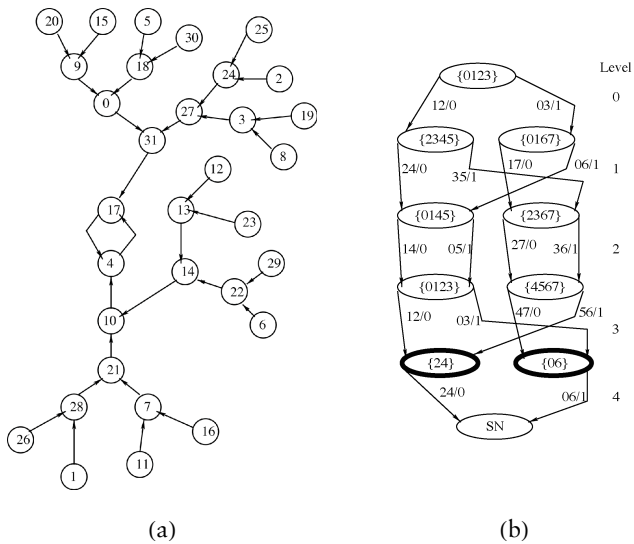


Figure 11. An illustration of Algorithm 3 with a five cell uniform CA with rule 105. (a) The STG and (b) the RVG. Type 1 nodes are marked with bold lines.

Table 3 displays all the associated rules (51, 60, 195, 102, 153, 90, 165, 105, 150, 204) of different groups and the value of n for which the rule generates invertible CAs. Table 3 has been derived by applying Algorithm 3 on the RVG of an n cell uniform CA. A formal proof of correctness of these results can be derived from the structure of their RVG.

https://doi.org/10.25088/ComplexSystems.19.1.89

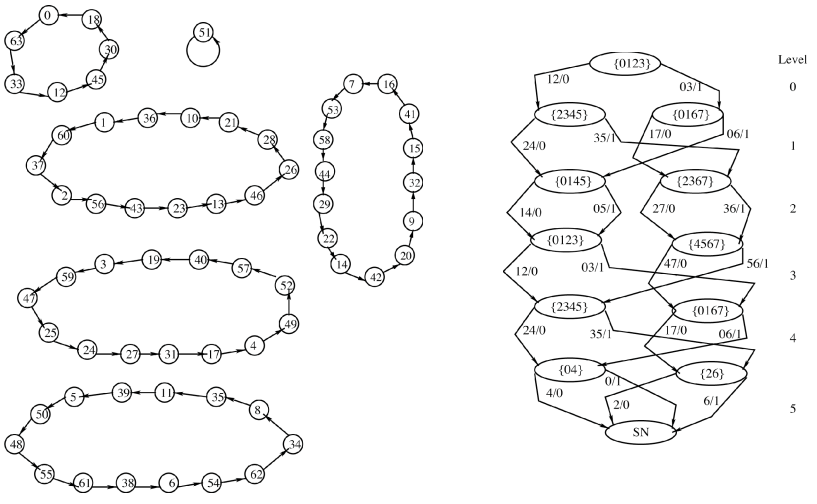


Figure 12. An illustration of Algorithm 3 with a six cell uniform CA with rule 105. (a) The STG and (b) the RVG.

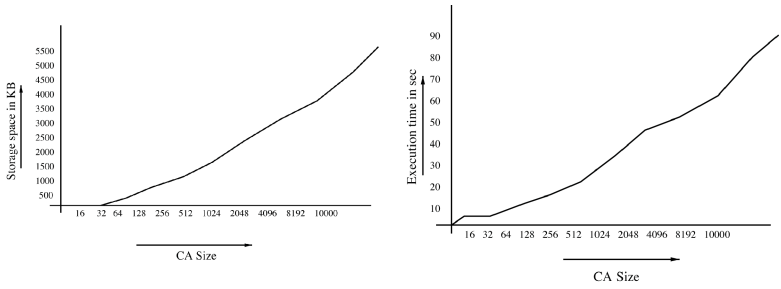
5. Experimental Results

Algorithm 3 has been coded using the C language on the Fedora 7 platform and run on an IBM Xeon server with various lengths of CAs ranging up to 10 000 cells. The CAs are chosen arbitrarily with an equal percentage of uniform and hybrid CAs. Figure 13 displays the linear growth of storage space and execution time, as confirmed in Section 3.3.

6. Conclusion

This paper reports a linear time algorithm for identifying null-boundary three neighborhood invertible cellular automata (CAs). An efficient data structure called the rule vector graph (RVG) of a cellular automaton (CA) is reported. The RVG of a CA can be derived from its rule vector (RV). Linear time traversal of a RVG identifies whether the CA is invertible or not. This result presents a significant improvement over quadratic time complexity reported earlier based on the general network of de Bruijn graphs.

https://doi.org/10.25088/ComplexSystems.19.1.89



(a) (b)

Figure 13. Experimental results. (a) Shows the growth in storage space and (b) shows the growth of execution time.

References

[1] K. Sutner, "Additive Automata on Graphs," *Complex Systems*, **2**(6), 1988 pp. 649-661.

[2] K. Sutner, "De Bruijn Graphs and Linear Cellular Automata," *Complex Systems*, **5**(1), 1991 pp. 19-30.

[3] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[4] J. Von Neumann, *The Theory of Self-Reproducing Automata* (A. W. Burks, ed.), Urbana, IL: University of Illinois Press, 1966.

[5] S. Wolfram, *Theory and Application of Cellular Automata*, Singapore: World Scientific Publishing Company, 1986.

[6] M. Sipper, "Co-evolving Non-Uniform Cellular Automata to Perform Computations," *Physica D*, **92**(3-4), 1996 pp. 193-208.

[7] N. Ganguly, "Cellular Automata Evolution: Theory and Applications in Pattern Recognition and Classification," Ph.D. Thesis, CST Department, BECDU, India, 2003.

[8] K. Culik, L. P. Hurd, and S. Yu, "Computation Theoretic Aspects of Cellular Automata," *Physica D*, **45**(1-3), 1990 pp. 357-378.

[9] M. Mitchell, P. T. Hraber, and J. P. Crutchfield, "Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations," *Complex Systems*, **7**(2), 1993 pp. 89-130.

[10] K. Sutner, "Linear Cellular Automata and Fischer Automata," *Parallel Computing*, **23**(11), 1997 pp. 1613-1634.

[11] T. Toffoli, "CAM: A High-Performance Cellular-Automaton Machine," *Physica D*, **10**(1-2), 1984 pp. 195-204.

[12] T. Toffoli and N. H. Margolus, "Invertible Cellular Automata: A Review," *Physica D*, **45**(1-3), 1990 pp. 229-253.

https://doi.org/10.25088/ComplexSystems.19.1.89

[13] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chatterjee, *Additive Cellular Automata, Theory and Applications Volume 1*, Los Alamitos, CA: Wiley-IEEE Computer Society Press, 1997.

[14] P. Maji, C. Shaw, N. Ganguly, B. K. Sikdar, and P. P. Chaudhuri, "Theory and Application of Cellular Automata for Pattern Classification," *Fundamenta Informaticae*, **58**(3-4), 2003 pp. 321-354.

[15] N. Ganguly, P. Maji, B. K. Sikdar, and P. P. Chaudhuri, "Design and Characterization of Cellular Automata Based Associative Memory for Pattern Recognition," *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, **34**(1), 2004 pp. 672-679.

[16] K. Cattell and J. C. Muzio, "Synthesis of One-Dimensional Linear Hybrid Cellular Automata," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **15**(3), 1996 pp. 325-335.

[17] S. Chakraborty, D. R. Chowdhury, and P. P. Chaudhuri, "Theory and Application of Nongroup Cellular Automata for Synthesis of Easily Testable Finite State Machines," *IEEE Transactions on Computers*, **45**(7), 1996 pp. 769-781.

[18] D. R. Chowdhury, S. Basu, I. S. Gupta, and P. P. Chaudhuri, "Design of CAECC—Cellular Automata Based Error Correcting Code," *IEEE Transactions on Computers*, **43**(6), 1994 pp. 759-764.

[19] D. R. Chowdhury, I. S. Gupta, and P. P. Chaudhuri, "CA-Based Byte Error-Correcting Code," *IEEE Transactions on Computers*, **44**(3), 1995 pp. 371-382.

[20] A. K. Das and P. P. Chaudhuri, "Vector Space Theoretic Analysis of Additive Cellular Automata and Its Application for Pseudoexhaustive Test Pattern Generation," *IEEE Transactions on Computers*, **42**(3), 1993 pp. 340-352.

[21] K. Cattell and J. C. Muzio, "Analysis of One-Dimensional Linear Hybrid Cellular Automata over GF(q)," *IEEE Transactions on Computers*, **45**(7), 1996 pp. 782-792.

[22] N. Ganguly, P. Maji, B. K. Sikdar, and P. P. Chaudhuri, "Generalized Multiple Attractor Cellular Automata (GMACA) Model for Associative Memory," *International Journal of Pattern Recognition and Artificial Intelligence,* **16**(7), 2002 pp. 781-795.

[23] P. Maji, N. Ganguly, and P. P. Chaudhuri, "Error Correcting Capability of Cellular Automata Based Associative Memory," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, **33**(4), 2003 pp. 466-480.

[24] K. Paul, D. R. Chowdhury, and P. P. Chaudhuri, "Theory of Extended Linear Machines," *IEEE Transactions on Computers*, **51**(9), 2002 pp. 1106-1110.

[25] M. Serra, T. Slater, J. C. Muzio, and D. M. Miller, "The Analysis of One-Dimensional Linear Cellular Automata and Their Aliasing Probabilities," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **9**(7), 1990 pp. 767-778.

[26] B. K. Sikdar, N. Ganguly, and P. P. Chaudhuri, "Design of Hierarchical Cellular Automata for On-Chip Test Pattern Generator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **21**(12), 2002 pp. 1530-1539.

https://doi.org/10.25088/ComplexSystems.19.1.89

[27] B. K. Sikdar, N. Ganguly, and P. P. Chaudhuri, "Fault Diagnosis of VLSI Circuits with Cellular Automata Based Pattern Classifier," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **24**(7), 2005 pp. 1115-1131.

[28] K. Cattell and J. C. Muzio, "Analysis of One-Dimensional Linear Hybrid Cellular Automata Over GF(q)," *IEEE Transactions on Computers*, **45**(7), 1996 pp. 782-792.

[29] K. Cattell, S. Zhang, M. Serra, and J. C. Muzio, "2xn Hybrid Cellular Automata with Regular Configuration: Theory and Application," *IEEE Transactions on Computers*, **48**(3), 1999 pp. 285-295.

[30] M. Serra."Hybrid Cellular Automata." (Jan 8, 2009) http://webhome.cs.uvic.ca/~mserra/CA.html.

[31] S. Amoroso and Y. N. Patt, "Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures," *Journal of Computer and System Sciences*, **6**(5), 1972 pp. 448-464.

[32] J. Kari, "Theory of Cellular Automata: A Survey," *Theoretical Computer Science*, **334**(1-3), 2005 pp. 3-33.

[33] J. Kari, "Reversibility of 2D Cellular Automata is Undecidable," in *Cellular Automata: Theory and Experiment*, (H. Gutowitz, ed.), Cambridge, MA: MIT Press, 1991 pp. 379-385.

[34] S. Das, B. K. Sikdar, and P. P. Chaudhuri, "Characterization of Reachable/Nonreachable Cellular Automata States," in *Lecture Notes in Computer Science*, Berlin: Springer, 2004 pp. 813-822.

[35] N. S. Maiti, S. Munshi, and P. P. Chaudhuri, "An Analytical Formulation for Cellular Automata (CA) Based Solution of Density Classification Task (DCT)," in *Seventh International Conference on Cellular Automata for Research and Industry (ACRI) 2006*, Perpignan, France (S. El Yacoubi, ed.), New York: Springer, 2006 pp. 147-156.

[36] S. Munshi, N. S. Maiti, D. Ray, D. R. Chowdhury, and P. P. Chaudhuri, "An Analytical Framework for Characterizing Restricted Two Dimensional Cellular Automata Evolution," in *Third Indian International Conference On Artificial Intelligence (IICAI 2007),* Pune, India (B. Prasad, ed.), 2007 pp. 1383-1402.

[37] K. Cattell and J. C. Muzio, "Synthesis of One-Dimensional Linear Hybrid Cellular Automata," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **15**(3), 1996 pp. 325-335.

[38] S. Dormann and A. Deutsch, "Modeling of Self-Organized Avascular Tumor Growth with Hybrid Cellular Automata," *In Silico Biology*, **2**(3), 2002 pp. 393-406.

[39] Z. Kohavi, *Switching and Finite Automata Theory, 2nd ed.*, New York: McGraw-Hill, 1978.

https://doi.org/10.25088/ComplexSystems.19.1.89