A Generic NSRTD Framework for Characterizing Cellular Automaton Rules in Constant (0/1) Boundary Conditions

Som Banerjee
Mamata Dalui
Department of Computer Science and Engineering
National Institute of Technology, Durgapur
West Bengal 713209, India

Suvadip Hazra
Department of Computer Science and Engineering
ITER, Siksha 'O' Anusandhan
Bhubaneshwar 751030, India

The cellular automaton (CA) is a rapidly emerging computational model that is widely used in different areas of simulations and scientific applications, as it can solve complex problems using simple rules. Cellular automata (CAs) are largely used in VLSI systems, fault detection, cryptography and other fields. In most of these applications, 2-state 3-neighborhood one-dimensional CAs are used. Some of the work on characterizing elementary CA (ECA) rules has been done, but all such work mainly focuses on a specific boundary condition (i.e., null, periodic boundary). This paper targets the design and development of a generic framework for the next state rule minterm transition diagram (NSRTD), referred to as the generic NSRTD (G-NSRTD), for constant (0/1) boundary conditions. By utilizing the developed G-NSRTD framework, which is reconfigurable based on the chosen boundary condition (constant-0/1), ECA rules can be characterized for both null and constant-1 boundary conditions. Hence, the current paper also focuses on characterizing all 256 ECA rules in constant (0/1) boundary conditions using the developed G-NSRTD framework, leading to the identification of all CA rules forming uniform single length cycle single-attractor CAs (SACAs), single length cycle two-attractor CAs (TACAs) and single length cycle multi-attractor CAs (MACAs) for arbitrary CA length n. Some of the relevant CA theories related to the characterization of ECA rules in constant (0/1) boundary conditions have been developed and reported accordingly.

Keywords: cellular automata; SACA; TACA; MACA; fixed-point attractor; NSRTD; constant (0/1) boundary CA

1. Introduction

John von Neumann first coined the term cellular automaton (CA) and introduced the concept [1] in the 1950s. Later, Stephen Wolfram

https://doi.org/10.25088/ComplexSystems.33.1.61

continued fundamental research [2] on cellular automata (CAs) and introduced different concepts. Since then, many researchers have been motivated to explore the structure and properties of CAs. CAs can be represented as a class of dynamic computation that can solve complex problems in a very simple way. CA models are represented by a series of cells in discrete time and space, where each cell is updated by using a set of rules and based upon the states of itself and its neighborhood. Many papers have been written about the 2-state 3-neighborhood one-dimensional CA because it is structurally simple to implement and forms basins of attraction [3], which is a potential modeling tool for solving various complex applications like pattern recognition [4, 5], authentication and cryptography [6, 7], fault detection [8], language recognition [9], cache coherence verification [10] and memory testing [11]. Researchers in [12] explored one-dimensional CAs and demonstrated how the asymptotic property of a CA changes with increased neighborhood size. The researchers in [3] identified a set of attractors during the characterization of one-dimensional 3-neighborhood CAs. The researchers in [6] developed a technique and identified attractors for linear or additive CAs. Researchers in [13] and [14] proposed a graphical approach for the verification of CA invertibility in linear time for periodic and null boundary CAs, respectively. Researchers in [11, 15] proposed the explicit rule vector graph and reachability tree, respectively, for identifying single length cycle attractors during the CA state transitions but failed to identify the existence of a multi-length cycle attractor. The researchers in [16] proposed the basic concept of the next state rule minterm transition diagram (NSRTD) and established it as an effective tool for the characterization of the single length cycle single-attractor CA (SACA) and single length cycle two-attractor CA (TACA) rules in the null-boundary condition. Researchers in [17] proposed a method for identifying SACA rules in the periodic boundary condition using the NSRTD. The authors in [18] used the concept of NSRTD and proposed an algorithm for synthesizing hybrid CA rules that form single length cycle two-attractor CAs (TACAs) from all the 15 uniform TACA rules as identified in [16]. However, the worst-case time complexity of the proposed design remains in the order of $O(15^n)$. The authors in [19] further overcame the drawbacks of [18] by introducing the concept of compatibility class relationship between rules R_i and R_{i+1}, using NSRTD to synthesize the hybrid CA rules that form TACAs in linear time. The researchers in [20] used the concept of compatibility class relationship between CA rules following the concept of NSRTD to synthesize nonuniform single length cycle single-attractor cellular automata (SACAs) for any arbitrary CA length n in linear time.

However, no work has been done so far to develop a generic framework of NSRTD that can be applied for characterizing rules in both

null boundary (constant-0) CAs and constant-1 boundary CAs. This motivates us to pursue research that can lead to the development of a generic NSRTD framework for constant (0/1) boundary conditions. The underlying theoretical basis for developing the generic NSRTD (G-NSRTD) framework has been formalized as algorithms and analyzed. Further, this paper focuses on characterizing all of the ECA rules using the developed G-NSRTD framework for constant (0/1) boundary conditions, leading to the identification of all CA rules forming uniform SACAs, TACAs and single length cycle multi-attractor CAs (MACAs) for any arbitrary length n in the null boundary condition as well as the constant-1 boundary condition.

The main contribution of this research is summarized in the following.

- Design and development of a G-NSRTD framework for constant (0/1) boundary conditions that can be used for characterizing CA rules for both null boundary and constant-1 boundary CAs.
- Characterizing all 256 elementary CA rules using the developed G-NSRTD framework and the identification of all the uniform SACA, TACA and MACA rules for arbitrary CA length n for both null boundary and constant-1 boundary conditions.
- Development of some relevant CA theories forming the basis for the proposed characterization of CA rules.

2. Cellular Automata Preliminaries

A CA is essentially an autonomous finite state machine where each cell is characterized by either of the two states 0 or 1. The next state (NS) of cell i of the CA at time step $t+1$ is $S_i^{t+1} = f_i(S_{i-1}^t, S_i^t, S_{i+1}^t)$, where S_i^t, S_{i+1}^t and S_{i-1}^t are the present state (PS) (at time t) of self, right and left neighbors of cell i, respectively, and f_i is known as the NS function. The collection of states represented by $S^t = (S_1^t, S_2^t, \ldots, S_n^t)$ represents the PS of the CA cells at time step t. Figure 1 depicts the NS function f_i. Here, a CA cell with state 0 is represented by □ and a CA cell with state 1 is represented by ■. The decimal representation of the eight outputs of f_i is known as rule R_i.

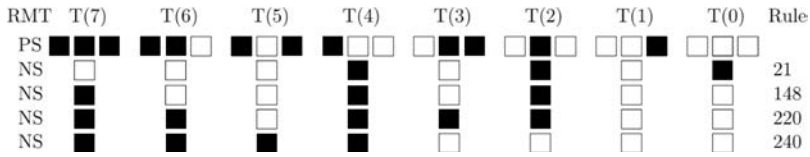

Figure 1. Rule minterms of the rules. PS: present state; NS: next state; RMT: rule minterm.

https://doi.org/10.25088/ComplexSystems.33.1.61

There can be $2^8 = 256$ possible rules in 2-state 3-neighborhood CAs. Four such rules—21, 148, 220, 240—are illustrated in Figure 1. The second row shows the possible $2^3 = 8$ combinations of the PS of cells $(i-1)$, i and $(i+1)$ at time t.

Following are some important definitions that are required for understanding the CA theory.

- *Rule vector*. The combination of rules $\langle R_1, R_2, \ldots, R_i, \ldots, R_n \rangle$ that configures the CA cells is known as the rule vector (RV).
- *Uniform and nonuniform CA*. A CA is said to be a uniform CA if $R_1 = R_2 = \cdots = R_n$ in an RV; otherwise, it is called a nonuniform or hybrid CA.
- *Constant-1 boundary CA*. A CA can be called a constant-1 boundary CA if its left (resp. right) neighbor of the first (resp. last) CA cell is permanently fixed to 1-state as depicted in Figure 2.

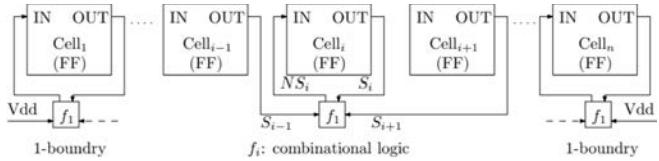

Figure 2. Constant-1 boundary CA.

- *Null boundary CA*. A CA can be called a null boundary CA if its left (resp. right) neighbor of the first (resp. last) CA cell is permanently fixed to 0-state.
- *RMT*. The combination of the present states (PSs) $(S_{i-1}^t, S_i^t, S_{i+1}^t)$ in a 3-neighborhood CA forms a rule minterm (RMT) as illustrated in row 2 of Figure 1. Rule minterms (RMTs) can be expressed by $T(x)$, where $x = \{0, 1, \ldots, 6, 7\}$; and $T(x) \in \{T\}$, where $T = \{T(0), T(1), \ldots, T(7)\}$. The RMT for a cell i at time t can be expressed as $T_i^t(x)$. For example, $T_3^t(2)$ refers to cell 3 RMT T(2) at time t. But, in all the pictorial representations, the RMT is expressed by using the decimal number format x, for example, $T_i^t(6)$ is expressed as 6.
- *0-RMT and 1-RMT*. The RMT for which the NS is 0 is called a 0-RMT, and the RMT for which the NS is 1 is called a 1-RMT.
- *Active and passive RMT*. If xyz is assumed as the PS RMT of a rule R, and if the state of y remains the same in the NS (i.e., 0(PS) → 0(NS) or 1(PS) → 1(NS)), it is known as a passive RMT. Similarly, if the state of y changes in the NS (i.e., 0(PS) → 1(NS) or 1(PS) → 0(NS)), it is known as an active RMT. The T(0) and T(2) in rules 148 and 220 (in Figure 1) are passive, whereas T(0) in rule 21 is active.
- *0-passive RMT and 1-passive RMT*. Assume that xyz is the PS RMT of a rule R, wherein $y = 0$ (resp. $y = 1$), then if the NS corresponding to xyz is 0 (resp. 1), RMT xyz is referred to as a 0-passive (resp. 1-passive) RMT.

- *RMT string*. RMT string (RS) can be expressed as a consecutive sequence of RMTs that appears in a CA state. It is expressed as $(T_1, T_2, \ldots, T_n)$ for a CA with n number of cells, where $T_i \in T$. In constant-1 boundary, a four-cell CA with state 0110 can be expressed by the RS $(T_1, T_2, T_3, T_4) = (T(5), T(3), T(6), T(5))$.
- *Attractor*. A set of states that form a cycle ($0 \to 0$ and $2 \to 3 \to 2$ of Figure 3(a)) is known as an attractor in a CA. An attractor that forms a self-loop ($0 \to 0$ of Figure 3(a)) is known as a single length cycle attractor or fixed-point attractor. A cycle that is formed by multiple states ($2 \to 3 \to 2$ of Figure 3(a)) is known as a multi-length cycle attractor.
- *Depth of a CA*. The longest path length (number of edges in the path) from a state to an attractor state in the state transition of a CA is referred to as the depth of the CA. Figure 3(a) shows a CA with a depth of three ($10 \to 7 \to 12 \to 0$).
- *Reversible and irreversible CAs*. If CA states during its state transitions form only cycles, then it is known as a reversible CA; otherwise, the CA is an irreversible CA (as depicted in Figures 3(a) and 3(b)).

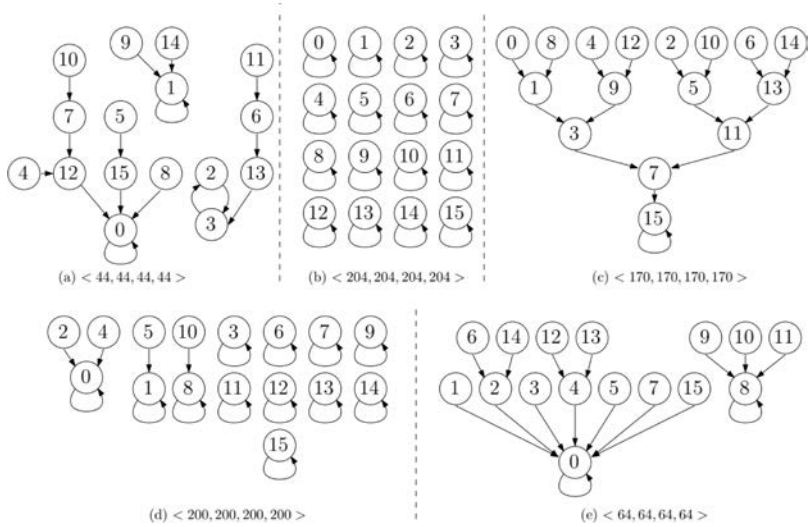

Figure 3. State transitions of SLCA in the constant-1 boundary condition.

- *SLCA*. If a CA during its state transition forms at least one single length cycle attractor, then it is known as a single length cycle attractor (SLCA) (as in Figures 3(a) and 3(b)). Multi-length attractors may or may not be present in an SLCA.
- *SACA*. If an SLCA forms a single connected component with only one self-loop and without any multi-length attractors during its state transition, then it is known as an SACA. A uniform four-cell CA with rule 170 (as shown in Figure 3(c)) forms an SACA, and rule 170 is an SACA rule.

https://doi.org/10.25088/ComplexSystems.33.1.61

- *MACA*. If an SLCA forms more than one connected component, each with one self-loop and without any multi-length attractors during its state transition, then it is known as an MACA. A uniform four-cell CA with rule 200 (as shown in Figure 3(d)) forms an MACA, and rule 200 is an MACA rule.
- *TACA*. A TACA is a special case of MACA that forms two connected components, each with one self-loop and without any multi-length attractors during its state transition. A uniform four-cell CA with rule 64 (as shown in Figure 3(e)) forms a TACA, and rule 64 is a TACA rule.

3. Next State Rule Minterm Transition Diagram

The NSRTD is a graph-based tool that was reported in [16] to identify the existence of single length cycle attractors as well as multi-length cycle attractors during the CA state transitions. Some of the definitions and terms that are relevant for proper understanding of the NSRTD theory include NCR, NSR, NSRS, NSRS graph (NSRS-G) and compatible classes of NSRSs.

During the CA state transition at time t, the RMTs T_i^t and T_{i+1}^t of two consecutive CA cells (cell i and cell $i+1$) are related via the next cell RMT (NCR) relationship, which is reported in Table 1. On the other hand, the RMT T_i^t of cell i at time t is updated to T_i^{t+1} at time $t+1$ during the CA state transition, and T_i^{t+1} is called the next state RMT (NSR) of T_i^t. The next state RMT sequence (NSRS) is the sequence of next state RMTs (NSRs) depending upon which the state of a CA cell changes during its state transition. In a CA cell, all the possible next state RMT sequences (NSRSs) can be represented by using a directed graph called the NSRS graph (NSRS-G), where the RMTs represent the vertex and the edge represents the NSRs. Further, the NSRSs between the two consecutive CA cells are said to be compatible if all the self-loops or multi-length cycles of cell $i+1$ follow the NCR relationship with self-loop(s) or multi-length cycle(s) of

RMT T_i (cell rule i)	NCR of RMT T_i (RMTs of cell rule $(i+1)$)
T(0)/T(4)	T(0), T(1)
T(1)/T(5)	T(2), T(3)
T(2)/T(6)	T(4), T(5)
T(3)/T(7)	T(6), T(7)

Table 1. NCR relationship between cell i and cell $i+1$.

cell i. The NSRTD is a two-dimensional directed graph where one dimension represents the NSR relationship of a CA cell, and the other represents the NCR relationship between the consecutive CA cells; that is, it is constructed with the compatible class of NSRSs (self-loop(s) and/or multi-length cycles(s)).

The details of NSRTD theory have been reported in [16]. The next section elaborates the development of a generic framework of NSRTD for the effective characterization of CA rules.

4. Proposed Work

In this section we develop a generic framework of NSRTD (G-NSRTD) for constant (0/1) boundary CAs. Further, we present the characterization of the class of SACA, TACA and MACA rules in constant (0/1) boundary conditions. SACAs that form a single graph with only one fixed point are used extensively in various applications, as reported in Section 1. TACAs (respectively, MACAs) form two fixed points (resp. specified number of fixed points) and are used largely in designing a two-class classifier (resp. multi-class classifier). Devising solutions to such problems requires characterizing SACA, TACA and MACA rules in different boundary conditions, which essentially demands the development of a G-NSRTD for different boundary conditions. In the current paper, we consider developing a framework of NSRTD for constant (0/1) boundary.

Finally, by utilizing the developed framework, we have identified all the SACA, TACA and MACA rules from among the 256 ECA rules (in both constant-0 (null) and constant-1 boundary conditions). Algorithms 1 through 5 formalize the construction of the proposed G-NSRTD framework.

Algorithm 1 accepts CA length n, a set of NCRs (Table 1) and a boundary condition $b = 0/1$. It outputs the sets of SACA rules ($\text{Rules}_{\text{SACA}}$), TACA rules ($\text{Rules}_{\text{TACA}}$) and MACA rules ($\text{Rules}_{\text{MACA}}$). The algorithm checks each CA rule (represented as K) from 0 to 255 to identify the sets of SACA, TACA and MACA rules, as shown in step 1. Step 2 identifies the set of NSRs of a cell represented as NSRs_i for CA cells $i = 1, 2, 3$ (intermediate cells), $n-1, n$ by calling Algorithm 2.

Algorithm 2 accepts CA length n, rule K, boundary condition b and gives NSRs_i as output. The algorithm checks for each CA cell $i = 1, 2, 3$ (intermediate cells), $n-1, n$ to identify the NSRs_i of each cell, as shown in step 1. Step 2 initializes the NSR structure (composed of L =left, M =middle, R =right values) based upon CA cell index i and the value of boundary condition b. There will be distinct

https://doi.org/10.25088/ComplexSystems.33.1.61

NSR structure initializations for cells 1, 2, $n-1$, n, as they are dependent on boundary condition b, and for cell 3 (to be replicated for all intermediate cells in the case of a uniform CA), the NSR structure initialization will be the same because it is independent of boundary condition b. After initialization, step 2 calls Algorithm 3 to compute the NSRs_i by passing the parameter values L, M, R, rule K, boundary condition b and CA cell index i.

Algorithm 3 accepts left value L, middle value M, right value R, rule K, boundary condition b, CA cell index i as input and generates NSRs for cell i as output. Step 1 creates $L\,|\,M\,|\,R$ sets based on all the combinations of present states (PSs) of L, M and R sets. Step 2 computes the NS of M from all the $L\,|\,M\,|\,R$ sets created in step 1 based on CA cell index i. For the first cell of the CA, Algorithm 4 is called two times by passing the parameter values PSbits, Lyz and yzR, respectively, and rule K. Then the two NSbits returned from Algorithm 4 are concatenated to create the NS represented by $y'z'$. For the second, third (intermediate cells), and $n-1$ CA cells, Algorithm 4 is called three times by passing the parameter values PSbits (Lxy, xyz and yzR, respectively) and rule K. Then the three NSbits returned from Algorithm 4 are concatenated to create the NS represented by $x'y'z'$. For the last cell of the CA, Algorithm 4 is called two times by passing the parameter values PSbits Lxy and xyR, respectively, and rule K. Then the two NSbits returned from Algorithm 4 are concatenated to create the NS represented by $x'y'$.

Algorithm 4 accepts rule K and PSbits as input and generates NSbit as output. Step 1 converts rule K into an eight-bit binary number and stores the value of each bit in bit-7, bit-6, bit-5, bit-4, bit-3, bit-2, bit-1 and bit-0, respectively. Step 2 checks the value of PSbits and assigns NSbit accordingly. The value of NSbit corresponding to the current PSbit (111/110/101/100/011/010/001/000) will be set with the value stored in bit-7/bit-6/bit-5/bit-4/bit-3/bit-2/bit-1/bit-0, accordingly. At the end, step 3 returns the calculated NSbit.

Step 3 of Algorithm 3 creates the NSR by using the transition (represented by $\rightarrow$) from PS to NS of M for all combinations of L, M and R calculated in step 2 and stores it in the NSR set represented by NSRs. For the first and last cells, the NSR is computed using the formula $\{(b\times 4) + \text{decimal}(yz) \rightarrow (b\times 4) + \text{decimal}(y'z')\}$ and $\{(\text{decimal}(xy)\times 2) + b \rightarrow (\text{decimal}(x'y')\times 2) + b\}$, respectively. Similarly, for CA cells 2, 3 (intermediate cells), and $n-1$, the NSR is computed using the formula $\{\text{decimal}(xyz) \rightarrow \text{decimal}(x'y'z')\}$.

In step 3 of Algorithm 1, the NSRS-G for each cell (represented by NSRS-G_i) is computed by calling Algorithm 5. Algorithm 5 accepts the NSRs as input and generates the NSRS-G as output. Step 1 of Algorithm 5 merges all the NSRs of a CA cell i to create the NSRS-G_i

for CA cell i and returns the computed NSRS-G$_i$ in step 2. For a uniform CA, the NSRs and NSRS-G of the intermediate cells (cell 3, ... , cell $(n-2)$) will be the same. Now, the NSRS-G$_i$ are examined to determine the presence of cycles and cycle length. Steps 4(a) and 4(b) find the self-loops and multi-length cycles represented by SL$_i$ and ML$_i$, respectively, from the NSRS-G$_i$. Now, in step 4(c), the SL$_i$ and ML$_i$ for each NSRS-G$_i$ are grouped together in a set, called the cycle set CYC$_i$. Further, in step 5, if any of the NSRS-G$_i$ has no self-loop (SL$_i = \phi$), the algorithm checks for the next CA rule. If all the NSRS-G$_i$ are made up only of self-loop(s) (i.e., ML$_i = \phi$), it jumps to step 6. Otherwise, the presence of a multi-length cycle is confirmed, and it jumps to step 8.

In step 6, each unique RMT sequence represented by S_u is identified by examining the NCR compatibility (using NCR Table 1) between self-loops SL$_i$ of two consecutive CA cells for each n number of CA cells. The unique RMT sequences S_us are represented by graphs in NSRTD. In step 7, the algorithm identifies a rule K as SACA (if $S_u(s) = 1$), TACA (if $S_u(s) = 2$) and MACA (if $S_u(s) > 2$) and classifies the rule K in a set of SACA (Rules$_{\text{SACA}}$), TACA (Rules$_{\text{TACA}}$) and MACA (Rules$_{\text{MACA}}$) rules, respectively. Otherwise, the algorithm checks for the next CA rule.

In steps 8 and 9, each unique RMT sequence represented by S_u is identified by checking the compatible NSRS class $C_{i(i+1)}$ between cycles (self-loop(s) along with the multi-length cycle(s)) of two consecutive CA cells for each n number of CA cells. In step 10, the algorithm checks for the next CA rule if any of the S_u is made up of a cycle length more than one, that is, if there are any multi-length cycle(s). The algorithm identifies a rule K as SACA (if $S_u(s) = 1$), TACA (if $S_u(s) = 2$) and MACA (if $S_u(s) > 2$) and adds the rule K in the sets of SACA (Rules$_{\text{SACA}}$), TACA (Rules$_{\text{TACA}}$) and MACA (Rules$_{\text{MACA}}$) rules, respectively. Otherwise, the algorithm checks for the next CA rule. At the end, the algorithm returns a set of SACA rules (Rules$_{\text{SACA}}$), a set of TACA rules (Rules$_{\text{TACA}}$) and a set of MACA rules (Rules$_{\text{MACA}}$) based on the selected boundary condition.

Algorithm 1. Compute_NSRTD_CB.

Input: CA length (n), set of NCRs, Boundary condition ($b = 0 / 1$)
Output: Rules$_{\text{SACA}}$, Rules$_{\text{TACA}}$, Rules$_{\text{MACA}}$

Start

1. **for** $K = 0$ to 255 where K represents the CA rule, **do**

2. NSRs$_i$ = call Identify_NSR (CA length (n), Rule K, Boundary condition (b)); for $i = 1, 2, 3, n-1, n$
   /* For uniform CAs, NSRs for all the intermediate cells will be the same for $i = 3$ to $(n-2)$*/

3. NSRS-G_i = call Compute_NSRS-G(NSRs$_i$); for $i = 1, 2, 3, n-1, n$
/* For uniform CAs, NSRS-G for all the intermediate cells will be the same for $i = 3$ to $(n-2)$*/
4. (a) Identify self-loops $SL_i = SL_{ij}$, for $i = 1, 2, 3, n-1, n$ and $j = 1, 2, \ldots, p$, where p is the number of self-loops in NSRS-G_i
(b) Identify multi-length cycle $ML_i = ML_{ik}$, for $i = 1, 2, 3, n-1, n$ and $k = 1, 2, \ldots, q$, where q is the number of multi-length cycle in NSRS-G_i
/* For uniform CA, SL_i and ML_i for all the intermediate cells will be the same for $i = 3$ to $(n-2)$*/
(c) $CYC_i = \{SL_i \bigcup ML_i\}$
5. **if** any NSRS-G_i has no self-loops ($SL_i = \phi$), for $i = 1, 2, 3, n-1, n$, **then**
Check for $(K+1)^{\text{th}}$ CA rule
else if all NSRS-G_i has only self-loops ($ML_i = \phi$), for $i = 1, 2, 3, n-1, n$ **then**
Goto step 6
else
Goto step 8
6. Generate each RMT sequence S_u as
$S_u = S_{1u} \rightarrow S_{2u} \rightarrow \cdots \rightarrow S_{(n-1)u} \rightarrow S_{nu}$;
where $S_{iu} \in SL_i$ and also $\in \{\text{NCR of } S_{(i-1)u} \forall_{i \neq 1}\}$, for $i = 1, 2, \ldots, n-1, n$.
7. **if** (number of $S_u == 1$) **then**
$\text{Rules}_{\text{SACA}} \leftarrow K$
else if (number of $S_u == 2$) **then**
$\text{Rules}_{\text{TACA}} \leftarrow K$
else if (number of $S_u > 2$) **then**
$\text{Rules}_{\text{MACA}} \leftarrow K$
else
Check for $(K+1)^{\text{th}}$ CA rule
8. Find compatibility class between
CYC_i and $CYC_{i+1} = C_{i(i+1)} = \{CYC_{ia}, CYC_{(i+1)b} \forall_{a,b}\}$; for $i = 1, 2, \ldots, n-1$
9. Generate each RMT sequence S_u as
$S_u = S_{1u} \rightarrow S_{2u} \rightarrow \cdots \rightarrow S_{(n-1)u} \rightarrow S_{nu}$;
where $(S_{iu}, S_{(i+1)u}) \in C_{i(i+1)}$, for $i = 1, 2, \ldots, n-1$
10. **if** there exists an S_{iu} without a self-loop in S_u **then**
Check for the $(K+1)^{\text{th}}$ CA rule
else if (number of $S_u == 1$) **then**
$\text{Rules}_{\text{SACA}} \leftarrow K$
else if (number of $S_u == 2$) **then**
$\text{Rules}_{\text{TACA}} \leftarrow K$
else if (number of $S_u > 2$) **then**
$\text{Rules}_{\text{MACA}} \leftarrow K$

else
Check for the $(K + 1)^{th}$ CA rule
end
Stop

Algorithm 2. Identify_NSR.
Input: CA length(n), Rule K, Boundary condition (b)
Output: $NSRs_i$

Start
1. **for** $i = 1, 2, 3, n - 1, n$ where n represents CA length(n) **do**
2. **if** ($i == 1$) **then**
$L = b$; $M = \{00, 01, 10, 11\}$; $R = \{0, 1\}$;
$NSRs_i$ = call Compute_NSR(L, M, R, K, b, i);
/* For the first cell, M is represented as yz */
else if ($i == 2$) **then**
$L = b$; $M = \{000, 001, 010, 011, 100, 101, 110, 111\}$; $R = \{0, 1\}$;
$NSRs_i$ = call Compute_NSR(L, M, R, K, b, i);
/* For the second cell, M is represented as xyz */
else if ($i == 3$) **then**
$L = \{0, 1\}$; $M = \{000, 001, 010, 011, 100, 101, 110, 111\}$; $R = \{0, 1\}$;
$NSRs_i$ = call Compute_NSR(L, M, R, K, b, i);
/* For all the intermediate cells, M is represented as xyz; and the NSR_i of all the intermediate cells will be the same for uniform CA */
else if ($i == n - 1$) **then**
$L = \{0, 1\}$; $M = \{000, 001, 010, 011, 100, 101, 110, 111\}$; $R = b$;
$NSRs_i$ = call Compute_NSR(L, M, R, K, b, i);
/* For the second to last cell, M is represented as xyz */
else
$L = \{0, 1\}$; $M = \{00, 01, 10, 11\}$; $R = b$;
$NSRs_i$ = call Compute_NSR(L, M, R, K, b, i);
/* For the last cell, M is represented as xy */
end
3. return $NSRs_i$
Stop

Algorithm 3. Compute_NSR
Input: Left Value (L), Middle Value (M), Right Value (R), Rule K, Boundary condition (b), CA cell index (i)
Output: NSRs

Start
1. For all combinations of the present state (PS) of L, M, R sets, create a $L \mid M \mid R$ set
/* For the 1^{st}, 2^{nd}, 3^{rd} (intermediate), $n - 1^{th}$, n^{th} cells, the PS of M is represented as yz, xyz, xyz, xyz, xy, respectively */
2. Calculate the next state (NS) of M from the $L \mid M \mid R$ sets created in step 1 according to cell index i

https://doi.org/10.25088/ComplexSystems.33.1.61

/* For the 1st, 2nd, 3rd (intermediate), $n-1^{th}$, n^{th} cells, the NS of M is represented as $y'z'$, $x'y'z'$, $x'y'z'$, $x'y'z'$, $x'y'$, respectively */
if (i == 1) **then**
$y'z'$ = call compute_NSbit(K, Lyz); || call compute_NSbit(K, yzR);
/* For the first cell, the PS of M is represented as yz and the NS of M is represented as $y'z'$ */
else if (i == 2 || i == 3 || $i == n-1$) **then**
$x'y'z'$ = call compute_NSbit(K, Lxy); || call compute_NSbit(K, xyz); || call compute_NSbit(K, yzR);
/* For the 2nd, 3rd, $n-1^{th}$ cells, the PS of M is represented as xyz and the NS of M is represented as $x'y'z'$ */
else
$x'y'$ = call compute_NSbit(K, Lxy); || call compute_NSbit(K, xyR);
/* For the last cell, the PS of M is represented as xy and the NS of M is represented as $x'y'$ */

3. Create NSR (PS to NS transition of M) for all combinations in the M set and store it in the NSR set represented as NSRs
if (i == 1) **then**
NSRs $\leftarrow (b\times 4)$ + Decimal(yz) $\rightarrow (b\times 4)$ + Decimal($y'z'$)
/* For the first cell */
else if (i == 2 || i == 3 || $i == n-1$) **then**
NSRs $\leftarrow$ Decimal(xyz) $\rightarrow$ Decimal($x'y'z'$)
/* For the 2nd, 3rd (intermediate), $n-1^{th}$ cells*/
else
NSRs $\leftarrow$ (Decimal(xy)$\times 2$) + $b \rightarrow$ (Decimal($x'y'$)$\times 2$) + b
/* For the last cell */

4. return NSRs
Stop

Algorithm 4. Compute_NSbit
Input: Rule K, PSbits
Output: NSbit

Start

1. Convert Rule K into 8-bit binary format and represent the 8 bit as bit-7, bit-6, bit-5, bit-4, bit-3, bit-2, bit-1, bit-0, respectively.
2. **if** (PSbits == 000) **then**
NSbit $\leftarrow$ bit-0
else if (PSbits == 001) **then**
NSbit $\leftarrow$ bit-1
else if (PSbits == 010) **then**
NSbit $\leftarrow$ bit-2
else if (PSbits == 011) **then**
NSbit $\leftarrow$ bit-3
else if (PSbits == 100) **then**
NSbit $\leftarrow$ bit-4
else if (PSbits == 101) **then**
NSbit $\leftarrow$ bit-5

else if (PSbits == 110) **then**
NSbit ← bit-6
else
NSbit ← bit-7

3. return NSbit
Stop

Algorithm 5. Compute_NSRS-G.
Input: NSRs
Output: NSRS-G

Start
1. Merge all the NSRs to create a graph called NSRS-G
2. return NSRS-G
Stop

4.1 Complexity Analysis

Step 1 of Algorithm 1 runs for all the CA rules 0 through 255 and takes constant time. Step 2 finds NSRs for the first, second, intermediate, second to last and last cells by calling Algorithm 2.

Step 1 of Algorithm 2 runs for CA cell index $i = 1, 2, 3, n-1, n$. Step 2 assigns L, M and R values for each CA cell in constant time and calls Algorithm 3.

Step 1 of Algorithm 3 creates all possible combinations of $L \mid M \mid R$ sets. There can be a maximum of eight possible combinations of $L \mid M \mid R$ sets in a CA cell, and hence, it also takes constant time. Step 2 computes the NS of M from $L \mid M \mid R$ sets by calling Algorithm 4.

Step 1 of Algorithm 4 converts rule K into an eight-bit binary number, and step 2 assigns the value of NSbit according to the value of PSbits. So, Algorithm 4 runs in constant time. Since Algorithm 4 takes constant time, so also step 2 of Algorithm 3 takes constant time. Step 3 of Algorithm 3 creates the NSR for all the combinations of PS to NS transitions of M sets.

In a CA cell, the maximum number of RMT nodes (PS of M) can be eight, and each RMT node can have a maximum of four out-degree edges called NSR (NS of M). Therefore, the overall construction of the NSRs_i in step 2 of Algorithm 1 also takes constant time. Step 3 computes the NSRS-G for the first, second, intermediate, second to last and last cells by calling Algorithm 5, which merges all the NSRs of a CA cell i to create the NSRS-G_i in constant time. Step 4 finds all possible cycles (self-loop(s) and multi-length cycle(s)) from the NSRS-G, which is finite and independent of CA length n taking constant time. Step 5 finds the number of self-loop(s) from the set of cycles, and it takes constant time. Step 6 finds each RMT sequence S

in $O(n)$ time complexity, where n is the CA length. Step 7 finds the total number of RMT sequences S (total number of graphs in NSRTD) and it is dependent on the rule number. Only in the case of rule 204, the total number of RMT sequences S is 2^n, making it the worst-case time complexity with $O(2^n)$. But for all other rules, the total number of RMT sequences S is $\ll 2^n$, making the average-case time complexity $O(S)$. Step 8 finds the compatibility class between two CA cells and depends upon the total number of cycles in the NSRS-G, so it is independent of n. Finding each RMT sequence S in step 9 will be the same as in step 6 and will take $O(n)$ time complexity. Similarly, counting the total number of RMT sequences S in step 10 will be the same as in step 7 and will take the worst-case time complexity of $O(2^n)$ (only in the case of rule 204) and an average time complexity of $O(S)$.

4.2 Examples

Example 1. Consider a run of Algorithm 1 (a single iteration) with CA length $n = 5$ and boundary condition $b = 1$ for CA rule 251 (Step 1). Step 2 generates the NSRs (a set of NSRs for each CA cell) for the first, second, intermediate, second to last and last cells (represented by NSRs_i for CA cell index $i = 1, 2, 3, n-1, n$, respectively) by calling Algorithm 2.

Algorithm 2 initializes the values of the L, M and R sets according to CA cell index i and calls Algorithm 3. Since constant-1 is the chosen boundary condition, the value of L will be $L = b = 1$ for the first and second cells, and the value of R will be $R = b = 1$ for the second to last and last cells, respectively.

Further, step 1 of Algorithm 3 creates $L\,|\,M\,|\,R$ sets using all the possible combinations that can be made from the L, M and R values initialized by step 2 of Algorithm 2. There will be 4, 8, 8, 8, 4 combinations of $L\,|\,M\,|\,R$ sets for the first, second, intermediate, second to last and last cells, respectively, and we define it as the PS of M sets. For the first cell, the possible $L\,|\,M\,|\,R$ sets will be $1\,|\,00\,|\,0/1$, $1\,|\,01\,|\,0/1$, $1\,|\,10\,|\,0/1$, $1\,|\,11\,|\,0/1$. For the second cell, the possible $L\,|\,M\,|\,R$ sets will be $1\,|\,000\,|\,0/1$, $1\,|\,001\,|\,0/1$, $1\,|\,010\,|\,0/1$, $1\,|\,011\,|\,0/1$, $1\,|\,100\,|\,0/1$, $1\,|\,101\,|\,0/1$, $1\,|\,110\,|\,0/1$, $1\,|\,111\,|\,0/1$. For intermediate cells, the possible $L\,|\,M\,|\,R$ sets will be $0/1\,|\,000\,|\,0/1$, $0/1\,|\,001\,|\,0/1$, $0/1\,|\,010\,|\,0/1$, $0/1\,|\,011\,|\,0/1$, $0/1\,|\,100\,|\,0/1$, $0/1\,|\,101\,|\,0/1$, $0/1\,|\,110\,|\,0/1$, $0/1\,|\,111\,|\,0/1$. For the second to last cell, the possible $L\,|\,M\,|\,R$ sets will be $0/1\,|\,000\,|\,1$, $0/1\,|\,001\,|\,1$, $0/1\,|\,010\,|\,1$, $0/1\,|\,011\,|\,1$, $0/1\,|\,100\,|\,1$, $0/1\,|\,101\,|\,1$, $0/1\,|\,110\,|\,1$, $0/1\,|\,111\,|\,1$. For the last cell, the possible $L\,|\,M\,|\,R$ sets will be $0/1\,|\,00\,|\,1$, $0/1\,|\,01\,|\,1$, $0/1\,|\,10\,|\,1$, $0/1\,|\,11\,|\,1$.

Step 2 of Algorithm 3 computes the NS of M from the $L\,|\,M\,|\,R$ created in step 1 by calling Algorithm 4 depending upon CA cell index i.

For the first cell, Algorithm 4 is called two times for each $L \mid M \mid R$ set with parameter values Lyz and yzR as PSbits along with rule K. Algorithm 4 checks the value of PSbits and assigns a specific bit from the eight-bit binary representation of rule K to create NSbit. Now, the two NSbits are concatenated to create the NS of M represented as $y'z'$. So, for the first cell, the present states (PSs) of $M(yz)$ are 00, 01, 10, 11, and the computed next states (NSs) of $M(y'z')$ are 11, (10/11), 11, 11, respectively. In the case of the second, intermediate and second to last cells, the same process is repeated, but Algorithm 4 is called three times for each $L \mid M \mid R$ set with parameter values Lxy, xyz and yzR as PSbits along with rule K. For the second cell, the PSs of $M(xyz)$ are 000, 001, 010, 011, 100, 101, 110, 111, and the computed NSs of $M(x'y'z')$ are 111, (110/111), 101, 111, 111, (110/111), 111, 111, respectively. For the intermediate cells, the PSs of $M(xyz)$ are 000, 001, 010, 011, 100, 101, 110, 111, and the computed NSs of $M(x'y'z')$ are 111, (110/111), 101, 111, (011/111), (010/011/110/111), 111, 111, respectively. For the second to last cell, the PSs of $M(xyz)$ are 000, 001, 010, 011, 100, 101, 110, 111, and the computed NSs of $M(x'y'z')$ are 111, 111, 101, 111, (011/111), (011/111), 111, 111, respectively. For the last cell, the PSs of $M(xy)$ are 00, 01, 10, 11, and the computed NSs of $M(x'y')$ are 11, 11, (01/11), 11, respectively.

Further, step 3 of Algorithm 3 creates the NSR for each of the PS to NS transitions of M computed in step 2 according to CA cell index i and stores it in the NSR set represented as NSRs. For the first and last cells, the NSR is computed using the formula $\{(b\times 4) + \text{decimal}(yz) \rightarrow (b\times 4) + \text{decimal}(y'z')\}$ and $\{(\text{decimal}(xy)\times 2) + b \rightarrow (\text{decimal}(x'y')\times 2) + b\}$, respectively, whereas for the second, intermediate and second to last cells, the NSR is computed using the formula $\{\text{decimal}(xyz) \rightarrow \text{decimal}(x'y'z')\}$. So the NSRs for the first cell are $4 \rightarrow 7$; $5 \rightarrow 6, 7$; $6 \rightarrow 7$; $7 \rightarrow 7$. The NSRs for the second cell are $0 \rightarrow 7$; $1 \rightarrow 6, 7$; $2 \rightarrow 5$; $3 \rightarrow 7$; $4 \rightarrow 7$; $5 \rightarrow 6, 7$; $6 \rightarrow 7$; $7 \rightarrow 7$. The NSRs for the intermediate cells are $0 \rightarrow 7$; $1 \rightarrow 6, 7$; $2 \rightarrow 5$; $3 \rightarrow 7$; $4 \rightarrow 3, 7$; $5 \rightarrow 2, 3, 6, 7$; $6 \rightarrow 7$; $7 \rightarrow 7$. The NSRs for the second to last cell are $0 \rightarrow 7$; $1 \rightarrow 7$; $2 \rightarrow 5$; $3 \rightarrow 7$; $4 \rightarrow 3, 7$; $5 \rightarrow 3, 7$; $6 \rightarrow 7$; $7 \rightarrow 7$. The NSRs for the last cell are $1 \rightarrow 7$; $3 \rightarrow 7$; $5 \rightarrow 3, 7$; $7 \rightarrow 7$. This completes the NSR computation steps.

Further, step 3 of Algorithm 1 generates the NSRS-G for the first, second, intermediate, second to last and last cells (represented by NSRS-G$_i$ for CA cell index $i = 1, 2, 3, n-1, n$, respectively) by calling Algorithm 5. Step 1 of Algorithm 5 merges all the NSRs to create the NSRS-G and returns the computed NSRS-G in step 2. The generated graphs for the first, second, intermediate, second to last and last cells are shown in Figure 4(a)–(e), respectively.

https://doi.org/10.25088/ComplexSystems.33.1.61

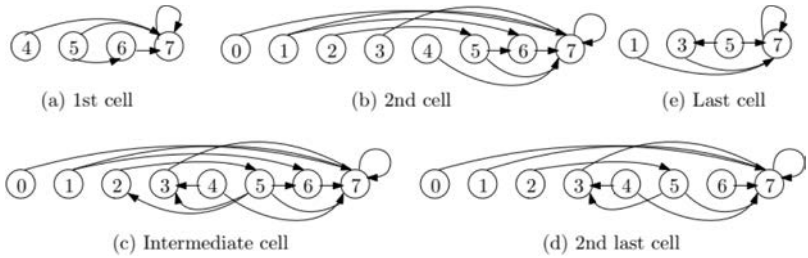

Figure 4. NSRS-G for uniform CA using rule 251.

From the graphs shown in Figure 4, we can identify that there are self-loops ((T(7), T(7)) in the first, second, intermediate, second to last and last cells) using step 4(a) and multi-length cycles ((T(2), T(5), T(2)) in all the intermediate cells) using step 4(b). In step 5, since the NSRS-G$_i$ are made up of self-loops and multi-length cycles, it jumps to step 8. In step 8, the compatibility classes $C_{i(i+1)}$ are identified as C_{12} = ((T(7), T(7)), (T(7), T(7))), C_{23} = ((T(7), T(7)), (T(7), T(7))), $C_{3(n-1)}$ = ((T(7), T(7)), (T(7), T(7))), $C_{(n-1)n}$ = ((T(7), T(7)), (T(7), T(7))). Then, step 9 finds the unique RMT sequences S_us by examining the compatibility between cycles (self-loops along with the multi-length cycle) of two consecutive CA cells. Here, S_u = (T(7), T(7)) → (T(7), T(7)) → (T(7), T(7)) → (T(7), T(7)) → (T(7), T(7)). The NSRTD S_u for CA rule 251 is shown in Figure 5. In step 10, since the number of S_u identified is 1 and it is made up of only self-loops (no multi-length cycle), rule 251 is identified as an SACA rule, and it is included in the set Rules$_{\text{SACA}}$.

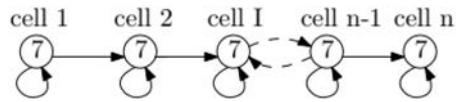

Figure 5. NSRTD for uniform CA using rule 251.

Example 2. Consider a run of Algorithm 1 (a single iteration) with CA length $n = 5$ and boundary condition $b = 0$ for CA rule 253 (step 1). Step 2 generates the NSRs (a set of NSRs for each CA cell) for the first, second, intermediate, second to last and last cells (represented by NSRs$_i$ for CA cell index $i = 1, 2, 3, n-1, n$, respectively) by calling Algorithm 2.

Algorithm 2 initializes the values of the L, M and R sets according to CA cell index i and calls Algorithm 3. Since null is the chosen boundary condition, the value of L will be $L = b = 0$ for the first and second cells, and the value of R will be $R = b = 0$ for the second to last and last cells, respectively.

Further, step 1 of Algorithm 3 creates $L\,|\,M\,|\,R$ sets using all the possible combinations that can be made from the L, M and R values initialized by step 2 of Algorithm 2. There will be 4, 8, 8, 8, 4 combinations of $L\,|\,M\,|\,R$ sets for the first, second, intermediate, second to last and last cells, respectively, and we define it as the PS of M sets. For the first cell, the possible $L\,|\,M\,|\,R$ sets will be $0\,|\,00\,|\,0/1$, $0\,|\,01\,|\,0/1$, $0\,|\,10\,|\,0/1$, $0\,|\,11\,|\,0/1$. For the second cell, the possible $L\,|\,M\,|\,R$ sets will be $0\,|\,000\,|\,0/1$, $0\,|\,001\,|\,0/1$, $0\,|\,010\,|\,0/1$, $0\,|\,011\,|\,0/1$, $0\,|\,100\,|\,0/1$, $0\,|\,101\,|\,0/1$, $0\,|\,110\,|\,0/1$, $0\,|\,111\,|\,0/1$. For the intermediate cells, the possible $L\,|\,M\,|\,R$ sets will be $0/1\,|\,000\,|\,0/1$, $0/1\,|\,001\,|\,0/1$, $0/1\,|\,010\,|\,0/1$, $0/1\,|\,011\,|\,0/1$, $0/1\,|\,100\,|\,0/1$, $0/1\,|\,101\,|\,0/1$, $0/1\,|\,110\,|\,0/1$, $0/1\,|\,111\,|\,0/1$. For the second to last cell, the possible $L\,|\,M\,|\,R$ sets will be $0/1\,|\,000\,|\,0$, $0/1\,|\,001\,|\,0$, $0/1\,|\,010\,|\,0$, $0/1\,|\,011\,|\,0$, $0/1\,|\,100\,|\,0$, $0/1\,|\,101\,|\,0$, $0/1\,|\,110\,|\,0$, $0/1\,|\,111\,|\,0$. For the last cell, the possible $L\,|\,M\,|\,R$ sets will be $0/1\,|\,00\,|\,0$, $0/1\,|\,01\,|\,0$, $0/1\,|\,10\,|\,0$, $0/1\,|\,11\,|\,0$.

Step 2 of Algorithm 3 computes the NS of M from the $L\,|\,M\,|\,R$ created in step 1 by calling Algorithm 4 depending upon CA cell index i. For the first cell, Algorithm 4 is called two times for each $L\,|\,M\,|\,R$ set with parameter values Lyz and yzR as PSbits along with rule K. Algorithm 4 checks the value of PSbits and assigns a specific bit from the eight-bit binary representation of rule K to create NSbit. Now, the two NSbits are concatenated to create the NS of M represented as $y'z'$. So, for the first cell, the PSs of $M(yz)$ are 00, 01, 10, 11, and the computed NSs of $M(y'z')$ are (10/11), 01, 11, 11, respectively. In the second, intermediate and second to last cells, the same process is repeated, but Algorithm 4 is called three times for each $L\,|\,M\,|\,R$ set with parameter values Lxy, xyz and yzR as PSbits along with rule K. For the second cell, the PSs of $M(xyz)$ are 000, 001, 010, 011, 100, 101, 110, 111, and the computed NSs of $M(x'y'z')$ are (110/111), 101, 011, 011, (110/111), 111, 111, 111, respectively. For the intermediate cells, the PSs of $M(xyz)$ are 000, 001, 010, 011, 100, 101, 110, 111, and the computed NSs of $M(x'y'z')$ are (110/111), 101, (011/111), (011/111), (110/111), 111, 111, 111, respectively. For the second to last cell, the PSs of $M(xyz)$ are 000, 001, 010, 011, 100, 101, 110, 111, and the computed NSs of $M(x'y'z')$ are 111, 101, (011/111), (011/111), 111, 111, 111, 111, respectively. For the last cell, the PSs of $M(xy)$ are 00, 01, 10, 11 and the computed NSs of $M(x'y')$ are 11, (01/11), 11, 11, respectively.

Further, step 3 of Algorithm 3 creates the NSR for each of the PS to NS transitions of M computed in step 2 according to CA cell index i and stores it in the NSR set represented as NSRs. For the first cell and last cells, the NSR is computed using the formula $\{(b\times 4) + \text{decimal}(yz) \rightarrow (b\times 4) + \text{decimal}(y'z')\}$ and $\{(\text{decimal}(xy)\times 2) +$

$b \rightarrow (\text{decimal}(x'y') \times 2) + b\}$, respectively, whereas for the second, intermediate and second to last cells, the NSR is computed using the formula $\{\text{decimal}(xyz) \rightarrow \text{decimal}(x'y'z')\}$. So the NSRs for the first cell are $0 \rightarrow 2, 3$; $1 \rightarrow 1$; $2 \rightarrow 3$; $3 \rightarrow 3$. The NSRs for the second cell are $0 \rightarrow 6, 7$; $1 \rightarrow 5$; $2 \rightarrow 3$; $3 \rightarrow 3$; $4 \rightarrow 6, 7$; $5 \rightarrow 7$; $6 \rightarrow 7$; $7 \rightarrow 7$. The NSRs for the intermediate cells are $0 \rightarrow 6, 7$; $1 \rightarrow 5$; $2 \rightarrow 3, 7$; $3 \rightarrow 3, 7$; $4 \rightarrow 6, 7$; $5 \rightarrow 7$; $6 \rightarrow 7$; $7 \rightarrow 7$. The NSRs for the second to last cell are $0 \rightarrow 7$; $1 \rightarrow 5$; $2 \rightarrow 3, 7$; $3 \rightarrow 3, 7$; $4 \rightarrow 7$; $5 \rightarrow 7$; $6 \rightarrow 7$; $7 \rightarrow 7$. The NSRs for the last cell are $0 \rightarrow 6$; $2 \rightarrow 2, 6$; $4 \rightarrow 6$; $6 \rightarrow 6$. This completes the NSR computation steps.

Further, step 3 of Algorithm 1 generates the NSRS-G for the first, second, intermediate, second to last and last cells (represented by NSRS-G_i for CA cell index $i = 1, 2, 3, n-1, n$, respectively) by calling Algorithm 5. Step 1 of Algorithm 5 merges all the NSRs to create the NSRS-G and returns the computed NSRS-G in step 2. The generated graphs for the first, second, intermediate, second to last and last cells are shown in Figure 6(a)–(e), respectively.

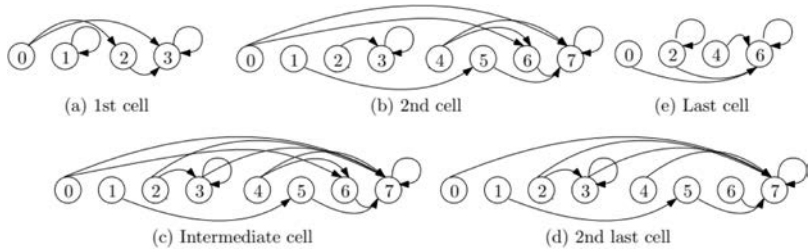

Figure 6. NSRS-G for uniform CA using rule 253.

From the graphs shown in Figure 6, we can identify that there are self-loops ((T(1), T(1)) and (T(3), T(3)) in the first cell; (T(3), T(3)) and (T(7), T(7)) in the second, intermediate, and second to last cells, respectively; and (T(2), T(2)) and (T(6), T(6)) in the last cell) using step 4(a) and no multi-length cycles in any cells using step 4(b). In step 5, since the NSRS-G_i are made up of only self-loops, it jumps to step 6. Then, step 6 finds the RMT sequences S_us by examining the NCR relationship between self-loops of two consecutive CA cells. For rule 253, there exist two RMT sequences S_1 = (T(1), T(1)) → (T(3), T(3)) → (T(7), T(7)) → (T(7), T(7)) → (T(6), T(6)) and S_2 = (T(3), T(3)) → (T(7), T(7)) → (T(7), T(7)) → (T(7), T(7)) → (T(6), T(6)). The NSRTDs S_us for CA rule 253 are shown in graphs G_1 and G_2 in Figure 7. In step 10, since there are two S_us, and as they are made up of only self-loops (no multi-length cycle), rule 253 is identified as a TACA rule, and it is included in the set $\text{Rules}_{\text{TACA}}$.

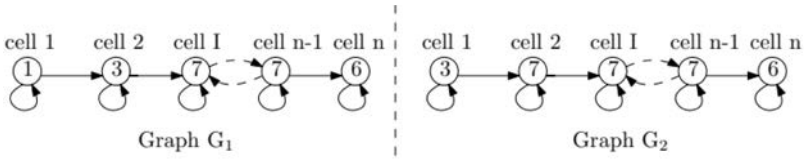

Figure 7. NSRTD for uniform CA using rule 253.

5. Results and Discussion

With extensive experimentation by using the concepts of NSRTD described in Section 3 and Algorithms 1 through 5 proposed in this paper, we have analyzed all the CA rules from 0 to 255 with different CA lengths n. Finally, we have identified all the rules that form SACA, TACA and MACA for any arbitrary CA length $n \geq 5$.

The uniform SACA rules for constant-1 boundary and null boundary conditions are reported in columns 2 and 3 of Table 2.

Group	Uniform SACA Rule in Constant-1 Boundary	Uniform SACA Rule in Null Boundary [16]	CA Rules That Form SACA in Both the Boundary Conditions
(1)	(2)	(3)	(4)
2	187, 243	34, 48	-
3	171, 185, 186, 191, 227, 241, 242, 247, 251	2, 16, 32, 42, 56, 98, 112, 162, 176	-
4	0, 15, 85, 170, 175, 184, 189, 190, 226, 231, 235, 240, 245, 246, 249, 250, 255	0, 10, 15, 24, 40, 66, 80, 85, 96, 130, 144, 160, 170, 184, 226, 240, 255	0, 15, 85, 170, 184, 226, 240, 255
5	14, 84, 174, 188, 230, 234, 239, 244, 248, 253, 254	8, 64, 128, 138, 143, 152, 168, 194, 208, 213, 224	-
6	238, 252	136, 192	-

Table 2. Uniform SACA rules in constant (0/1) boundary conditions.

Theorem 1. If any of the NSRS-G_i for a rule R do not have at least one self-loop, then that rule cannot be considered as a candidate for SACA, TACA or MACA.

Proof. If all the NSRs of a NSRS-G_i for a rule R form a cycle (or cycles) without a self-loop, then there will be no compatibility

class of self-loops between NSRS – $G_{(i-1)}$ – NSRS – G_i and NSRS – G_i – NSRS – $G_{(i+1)}$. So, no path in NSRTD with only self-loops will be created that corresponds to a single length cycle attractor. Hence, the proof. ■

Conjecture 1. In a SACA rule under the constant-1 boundary condition, either of the RMTs T(7) or T(5) must be a passive RMT.

Conjecture 2. The set of SACA rules is different in null boundary and constant-1 boundary conditions if and only if the number of 0-passive RMTs does not equal the number of 1-passive RMTs.

Conjecture 3. Some of the group 4 SACA rules are common in null boundary and constant-1 boundary conditions (as shown in Table 2) if and only if the number of 0-passive RMTs equals the number of 1-passive RMTs.

Theorem 2. Rules 0 and 255 are both SACA rules in constant (0/1) boundary conditions.

Proof. For a uniform CA configured with rule 0, all the states in the state transition diagram will generate an all-0s next state, forming a single graph where all the states point toward state-0 (CA state with all 0) with a self-loop on state-0. Similarly, for a uniform CA configured with rule 255, all the states in the state transition diagram point toward state x (where $x = 2^{\text{CA length}} - 1$) with a self-loop on state x. So, the state transition diagrams formed for the uniform CA with rule 0 as well as the uniform CA with rule 255 satisfy the necessary conditions (following the definition of SACA defined in Section 2) for an SACA rule in constant (0/1) boundary conditions. ■

Theorem 3. Rule 204 forms MACA in constant (0/1) boundary conditions.

Proof. Rule 204 is the only ECA rule that has eight passive RMTs. As all the RMTs are passive, so each state in the state transition diagram will give the same state during the next state transition (self-replicating state), forming a fixed-point attractor only. As all the states in the state transition diagram form fixed-point attractors only (no multi-length cycle), and as it is independent of boundary condition, so rule 204 is an MACA rule in constant (0/1) boundary conditions. ■

Theorem 4. A uniform CA configured with rule 204 gives the maximum number of fixed-point attractors among the set of rules in constant (0/1) boundary conditions, and the number of fixed-point attractors increases exponentially with an increase in CA length.

Proof. As all the states in the state transition diagram form fixed-point attractors (directly following the proof of Theorem 3), rule 204 forms the maximum number of fixed-point attractors among all the

256 ECA rules in 3-neighborhood CAs. Further, the number of states (fixed-point attractors in the state transition diagram of a uniform CA configured with rule 204) in the state transition diagram increases exponentially with an increase in CA length. So, for a uniform CA configured with rule 204, the number of fixed-point attractors increases exponentially with an increase in CA length in constant (0/1) boundary conditions. ■

Conjecture 4. SACA rules in null and constant-1 boundary conditions belong only to groups 2, 3, 4, 5 and 6.

The uniform TACA rules for constant-1 boundary and null boundary conditions are reported in columns 2 and 3 of Table 3.

Group (1)	Uniform TACA Rule in Constant-1 Boundary (2)	Uniform TACA Rule in Null Boundary [16] (3)
3	155, 211	38, 52
4	40, 96, 139, 154, 169, 209, 210, 225	46, 106, 116, 120, 166, 180, 235, 249
5	8, 64, 128, 138, 208	174, 239, 244, 253, 254

Table 3. Uniform TACA rules in constant (0/1) boundary conditions.

Conjecture 5. All TACA rules in constant (0/1) boundary conditions belong only to groups 3, 4 and 5.

Conjecture 6. The RMT T(2) will always be an active RMT for all TACA rules under constant-1 boundary condition.

Conjecture 7. In a TACA rule under constant-1 boundary condition, at least one of the RMTs between T(5) and T(4), T(1) and T(5), T(0) and T(7) must be a passive RMT.

The uniform MACA rules in constant-1 boundary and null boundary conditions are reported in columns 2 and 3 of Table 4.

Figures 8 and 9 show the number of fixed-point attractors (NSRTD) formed for an MACA rule with varying CA length n for constant-1 and null boundary conditions, respectively.

Conjecture 8. All the MACA rules (as shown in Table 4) form multiple numbers of fixed-point attractors for any arbitrary CA length ($n \geq 5$), and the number of such fixed-point attractors increases monotonically with an increase in CA length.

Theorem 5. Some of the MACA rules show a small linear increase in the number of fixed-point attractors, whereas some of the MACA rules show a sharp exponential increase in the number of fixed-point attractors with an increase in CA length.

https://doi.org/10.25088/ComplexSystems.33.1.61

Proof. This proof depends upon two factors:

1. Number of passive RMT(s).
2. Compatible class of NSRSs between second cell – intermediate cell, intermediate cell – intermediate cell, intermediate cell – second to last cell.

Group	Uniform MACA Rule in Constant-1 Boundary	Uniform MACA Rule in Null Boundary [21]	CA Rules That Form MACA in Both the Boundary Conditions
(1)	(2)	(3)	(4)
4	159, 215, 219	6, 20, 36	-
5	4, 13, 69, 79, 93, 143, 158, 168, 173, 203, 213, 214, 217, 218, 223, 224, 229, 233	4, 13, 14, 44, 69, 74, 79, 84, 88, 93, 100, 104, 134, 148, 164, 223, 234, 248	4, 13, 69, 79, 93, 223
6	12, 68, 72, 77, 78, 92, 132, 136, 141, 142, 172, 192, 197, 202, 207, 212, 216, 221, 222, 228, 232, 237	12, 68, 72, 77, 78, 92, 132, 141, 142, 172, 197, 202, 207, 212, 216, 221, 222, 228, 232, 237, 238, 252	12, 68, 72, 77, 78, 92, 132, 141, 142, 172, 197, 202, 207, 212, 216, 221, 222, 228, 232, 237
7	76, 140, 196, 200, 205, 206, 220, 236	76, 140, 196, 200, 205, 206, 220, 236	76, 140, 196, 200, 205, 206, 220, 236
8	204	204	204

Table 4. Uniform MACA rules in constant (0/1) boundary conditions.

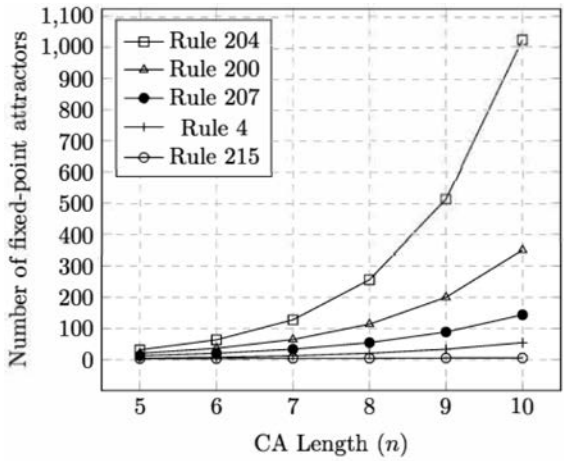

Figure 8. Number of fixed-point attractors for MACA rule under constant-1 boundary with varying CA length n.

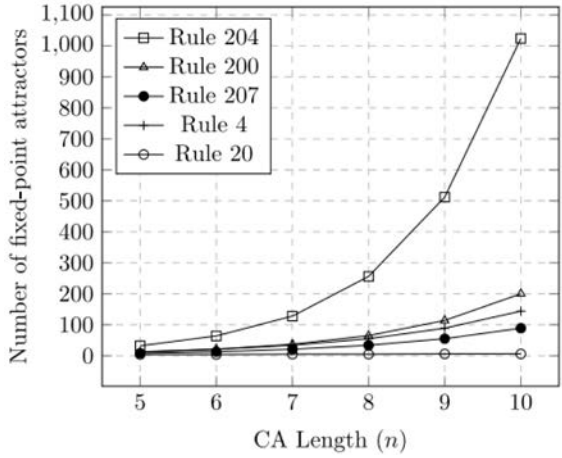

Figure 9. Number of fixed-point attractors for MACA rule under null boundary with varying CA length n.

A rule R with a larger number of passive RMTs tends to form a larger number of self-loops in an NSRS-G. Now, with a larger number of self-loops in the NSRS-G, the probability of getting a compatible class of NSRSs increases, thereby increasing the number of graphs in NSRTD (or fixed-point attractor). Again, with an increase in CA length (by replicating intermediate cells), the combination with compatible NSRSs class (between second cell – intermediate cell, intermediate cell – intermediate cell, intermediate cell – second to last cell) increases further, thereby increasing the number of graphs in NSRTD (or fixed-point attractor).

Figures 8 and 9 show the case for a uniform CA configured with rule 204 (the number of passive RMTs = 8), which forms self-loops in all NSRSs and has the maximum number of compatible classes possible between two consecutive CA cells. So, in the case of a CA with rule 204, there is a sharp exponential rise in the number of fixed-point attractors with an increase in CA length. On the other hand, a CA with rule 207 (number of passive RMTs = 6) shows a small linear rise in the number of fixed-point attractors with an increase in CA length, as it forms very few self-loop NSRSs and has a smaller number of self-loop NSRSs in the compatible class between two consecutive CA cells. ■

6. Conclusion

The evolution of the cellular automaton (CA) was started with von Neumann's work and progressed through Wolframs's model to the current research trends. Different forms of cellular automata

(CAs) have been proposed for different applications. CAs yielding fixed-point attractor(s) have proven their importance in cryptography, fault detection and many other useful applications that necessitate the characterization of CAs with fixed points. Some of the previous papers in the literature have addressed the characterization of elementary CA (ECA) rules using the next state rule minterm transition diagram (NSRTD) but have focused on only a specific boundary condition. Hence, this paper explores the concept of NSRTD for developing a generic NSRTD (G-NSRTD) framework, which is reconfigurable depending upon the user-chosen boundary condition (i.e., constant(0/1)). The developed G-NSRTD can be effectively used for characterizing CA rules in null boundary as well as constant-1 boundary conditions. Further, we have identified all uniform single length cycle single-attractor CA (SACA), single length cycle two-attractor CA (TACA) and single length cycle multi-attractor CA (MACA) rules in null boundary and constant-1 boundary conditions. Analysis of the SACA, TACA and MACA rules in different boundary conditions leads to the development of relevant CA theories, which are essential for characterizing the class of CA rules with fixed-point attractor(s). As a future extension of the proposed G-NSRTD framework, we may consider incorporating more boundary conditions and extended neighborhood configurations like the five-neighborhood (von Neumann neighborhood).

References

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

[2] S. Wolfram, *Cellular Automata and Complexity: Collected Papers*, Reading, MA: Addison-Wesley Publishing Company, 1994.

[3] A. Wuensche and M. Lesser, *Global Dynamics of Cellular Automata: An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*, Reading, MA: Addison-Wesley, 1992.

[4] N. Ganguly, "Cellular Automata Evolution: Theory and Applications in Pattern Recognition and Classification," Ph.D. thesis, Bengal Engineering College (DU), Shibpur, Howrah, W. B. India, 2004.

[5] P. Maji, "Cellular Automata Evolution for Pattern Recognition," Ph.D. thesis, Jadavpur University, Kolkata, India, 2005.

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

[7] P. Dasgupta, S. Chattopadhyay and I. Sengupta, "An ASIC for Cellular Automata Based Message Authentication," in *VLSI Design 2000, Wireless and Digital Imaging in the Millennium, Proceedings of 13th International Conference on VLSI Design*, Calcutta, India, Los Alamitos, CA: IEEE Computer Society, 2000 pp. 538–541. doi:10.1109/ICVD.2000.812663.

[8] S. Das, N. N. Naskar, S. Mukherjee, M. Dalui and B. K. Sikdar, "Characterization of CA rules for SACA Targeting Detection of Faulty Nodes in WSN," in *Proceedings of the 9th International Conference on Cellular Automata for Research and Industry (ACRI'10)*, Ascoli Piceno, Italy (S. Bandini, S. Manzoni and H. Umeo, eds.), Berlin, Heidelberg: Springer, 2010 pp. 300–311.

[9] N. Bacquey, "Complexity Classes on Spatially Periodic Cellular Automata," in *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2014 pp. 112–124. doi:10.4230/LIPIcs.STACS.2014.112.

[10] M. Dalui and B. K. Sikdar, "Design of Directory Based Cache Coherence Protocol Verification Logic in CMPs around TACA," in *International Conference on High Performance Computing & Simulation (HPCS 2013)* Helsinki, Finland (W. W. Smari and V. Zeljkovic, eds.), Piscataway, NJ: IEEE, 2013 pp. 318–325. doi:10.1109/HPCSim.2013.6641433.

[11] M. Saha, M. Dalui and B. K. Sikdar, "A Cellular Automata Based Highly Accurate Memory Test Hardware Realizing March C^-," *Microelectronics Journal, Elsevier*, **52**, 2016 pp. 91–103. doi:10.1016/j.mejo.2016.03.009.

[12] H. Zenil and E. Villarreal-Zapata, "Asymptotic Behavior and Ratios of Complexity in Cellular Automata," *International Journal of Bifurcation and Chaos,* **23**(9), 2013 pp. 135–159. doi:10.1142/S0218127413501599.

[13] N. S. Maitii, S. Ghosh, B. K. Sikdar and P. P. Chaudhuri, "Rule Vector Graph (RVG) to Design Linear Time Algorithm for Identifying the Invertibility of Periodic-Boundary Three Neighborhood Cellular Automata," *Journal of Cellular Automata*, 7(4), 2012 pp. 335–362. www.oldcitypublishing.com/journals/jca-home/jca-issue-contents/jca-volume-7-number-4-2012/jca-7-4-p-335-362.

[14] N. S. Maitii, S. Ghosh, S. Munshi and P. P. Chaudhuri, "Linear Time Algorithm for Identifying the Invertibility of Null-Boundary Three Neighborhood Cellular Automata," *Complex Systems*, **19**(1), 2010 pp. 89–113. doi:10.25088/ComplexSystems.19.1.89.

[15] S. Das, S. Mukherjee, N. Naskar and B. K. Sikdar, "Characterization of Single Cycle CA and Its Application in Pattern Classification," *Electronic Notes in Theoretical Computer Science*, **252**, 2009 pp. 181–203. doi:10.1016/j.entcs.2009.09.021.

[16] M. Dalui, B. Chakraborty, N. Das and B. K. Sikdar, "NSRT Diagram for Identification of SACA and TACA Rules in Null-Boundary," *International Journal of Modern Physics C*, **33**(6), 2022 2250071. doi:10.1142/S0129183122500711.

[17] B. Das, M. Saha, N. Das and B. K. Sikdar, "Identification of Periodic Boundary SACA Rules Exploring NSRT Diagram," in *15th International Conference on Cellular Automata for Research and Industry (ACRI 2022)*, Geneva, Switzerland (B. Chopard, S. Bandini, A. Dennunzio and M. A. Haddad, eds.), Cham: Springer, 2022 pp. 29–39. doi:10.1007/978-3-031-14926-9_3.

[18] S. Hazra and M. Dalui, "Synthesis of Single Length Cycle Two Attractor CA Using NSRT Diagram," in *Proceedings of First Asian Symposium on Cellular Automata Technology (ASCAT 2022)*, Kolkata, India (S. Das and G. J. Martinez, eds.), Singapore: Springer, 2022 pp. 235–246. doi:10.1007/978-981-19-0542-1_17.

[19] S. Hazra and M. Dalui, "Characterization of Single Length Cycle Two-Attractor Cellular Automata Using Next-State Rule Minterm Transition Diagram," *Complex Systems*, **31**(4), 2022 pp. 363–388. doi:10.25088/ComplexSystems.31.4.363.

[20] B. Chakraborty, M. Dalui and B. K. Sikdar, "Synthesis of Scalable Single Length Cycle, Single Attractor Cellular Automata in Linear Time," *Complex Systems*, **30**(3), 2021 pp. 415–439. doi:10.25088/ComplexSystems.30.3.415.

[21] S. Hazra, "Theory and Applications of Cellular Automata for Detection and Mitigation of Hardware Trojan Attacks Targeting Cache Performance in a Many Core System," Ph.D. thesis, Computer Science and Engineering, National Institute of Technology, Durgapur, India, 2021. hdl.handle.net/10603/524200.