Characterization of Single Length Cycle Two-Attractor Cellular Automata Using Next-State Rule Minterm Transition Diagram

Suvadip Hazra
Mamata Dalui

Computer Science & Engineering, NIT Durgapur
Durgapur
West Bengal 713209, India

Cellular automata (CAs) are simple mathematical models that are effectively being used to analyze and understand the behavior of complex systems. Researchers from a wide range of fields are interested in CAs due to their potential for representing a variety of physical, natural and real-world phenomena. Three-neighborhood one-dimensional CAs, a special class of CAs, have been utilized to develop various applications in the field of very large-scale integration (VLSI) design, error-correcting codes, test pattern generation, cryptography and others. A thorough analysis of a three-neighborhood cellular automaton (CA) with two states per cell is presented in this paper. A graph-based tool called the next-state rule minterm transition diagram (NSRTD) is presented for analyzing the state transition behavior of CAs with fixed points. A linear time mechanism has been proposed for synthesizing a special class of irreversible CAs referred to as single length cycle two-attractor CAs (TACAs), having only two fixed points.

Keywords: cellular automata; TACA; NSRTD

1. Introduction

Cellular automata (CAs), a significant development in the history of computing, were first developed by John von Neumann in the 1950s to study self-reproducing automata [1]. Wolfram established that CAs could be used to describe complex natural events and subsequently set the foundation for a theory of CAs, which are defined as discrete dynamic systems in which local interactions among components cause global changes in space and time [2]. Due to the simple but sophisticated structure, three-neighborhood two-state CAs have recently proved to be an effective modeling tool [3–10]. Characterizing such a machine, on the other hand, is still under study. While analyzing the state space of a three-neighborhood cellular automaton (CA), the researchers identified a collection of states known as attractors, toward which adjacent states asymptotically approach during dynamic evaluation [11]. An attractor that has just one state is known as a fixed point [12]. The CA with fixed points is of utmost importance when developing schemes for various applications, particularly authentication and cryptography [12–15]. The verification of cache coherence in chip multiprocessors [16], pattern classification [17], diagnosis of malfunctioning nodes in wireless sensor networks [18], memory testing [19], language recognition [9] and parity problem-solving [10] are some other possible application domains. This paper focuses on the characterization of such a particular class of CAs in null boundary, which is based on an examination of individual rules and their capacity to produce fixed points. The next-state rule minterm transition diagram (NSRTD) [20] tool provides the theoretical basis for identifying such rules.

In [12, 21, 22], the researchers have developed techniques to identify attractors for linear/additive CAs. The researchers in [5, 23] have proposed a graph-based scheme for identifying attractors, wherein the de Bruijn diagram and its subdiagrams have been employed to study the CA and a quadratic time methodology is proposed to assess whether or not a linear CA is reversible. In addition, the rule vector graph (RVG), an alternative graph-based approach, is proposed for use in null-boundary CAs in [24]. It aims for a solution that takes linear time in order to test the invertibility of a CA. It is reported in [25] that a similar solution exists for periodic-boundary CAs. However, there is still much to discover about the characterization of CAs with fixed points outside the linear domain. This encourages us to develop a CA theory with the objective of identifying rules that may generate a specific number of fixed points in the null boundary. The class of irreversible CAs with only two fixed points can act as a two-class classifier; hence, it has wide applicability in various application domains.

The currently available tools, specifically the reachability tree (RT) for attractors [17] and the explicit rule vector graph (ERVG) [19], are efficient for identifying fixed points in the state transition of a CA. On the other hand, in their current form, none of these can be used to identify the existence of multilength cycle attractors in a CA. It is indispensable to identify the CA rules that have the potential to configure nonuniform CAs having only two fixed points for all lengths, referred to as single length cycle two-attractor CAs (TACAs). The characterization of rules for the synthesis of TACAs may be achieved through the use of the graph-based tool NSRTD. The NSRTD is able to investigate the presence of both multilength cycles and fixed points, making it a more generic method for characterizing a CA with fixed points and/or multilength cycles. It effectively identifies the single length cycle (fixed-point) attractor CA (SLCA) rules that result in the synthesis of arbitrary-length TACAs.

The major contributions of this paper are:

- Characterization of the CA rules that are the building blocks for synthesizing the desired TACAs.
- Devising a highly efficient linear time synthesis mechanism for nonuniform TACAs based on analysis of individual CA rules and their potential to form TACAs of arbitrary length.

2. Cellular Automata Preliminaries

A CA consists of a number of cells organized in the form of a lattice. It evolves in discrete space and time and can be viewed as an autonomous finite-state machine. Each cell of a CA stores a discrete variable at time t that refers to the present state (PS) of the cell (CA configuration at time t). The next state (NS) of the cell at $t+1$ (configuration at $t+1$) is affected by its PS and the PS values of its neighbors. In this paper, we concentrate on three-neighborhood CAs (self, left and right neighbors) having two states: 0 or 1. The NS of the ith cell of such a CA 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 PSs of itself and the left and right neighbors of the ith cell and f_i is the next state function. The set of states $S^t = (S_1^t, S_2^t, \ldots, S_n^t)$ of the cells at t is the PS of the CA. Therefore, the NS of an n –cell CA is

$$S^{t+1} = (f_1(S_0^t, S_1^t, S_2^t),\ f_2(S_1^t, S_2^t, S_3^t),\ \ldots,\ f_n(S_{n-1}^t, S_n^t, S_{n+1}^t)).$$

The f_i can be expressed in the form of a truth table (Table 1). The decimal equivalent of the eight outputs of f_i is called the "rule" R_i. In a two-state three-neighborhood CA, there can be 2^8 (256) rules. Eight such rules, 14, 15, 164, 192, 207, 240, 254 and 255, are illustrated in Table 1. The first row lists the possible 2^3 (8) combinations of PSs of the $(i-1)$th, ith and $(i+1)$th cells. The next seven rows indicate the NSs of the ith cell, forming the rules 14 $(f_i = S'_{i-1}S_{i+1} + S'_{i-1}S_i)$, 15 $(f_i = S'_{i-1})$, 164$(f_i = S_{i-1}S_{i+1} + S'_{i-1}S_iS'_{i+1})$, 192 $(f_i = S_{i-1}.S_i)$, 207 $(f_i = S'_{i-1} + S_i)$, 240 $(f_i = S_{i-1})$, 254 $(f_i = S_{i-1} + S_i + S_{i+1})$ and 255 $(f_i = 1)$. The following definitions are relevant for ease of understanding of the CA theory developed in this work.

PS	111	110	101	100	011	010	001	000	
RMT	T(7)	T(6)	T(5)	T(4)	T(3)	T(2)	T(1)	T(0)	Rule
NS	0	0	0	0	1	1	1	0	14
NS	0	0	0	0	1	1	1	1	15
NS	1	0	1	0	0	1	0	0	164
NS	1	1	0	0	0	0	0	0	192
NS	1	1	0	0	1	1	1	1	207
NS	1	1	1	1	0	0	0	0	240
NS	1	1	1	1	1	1	1	0	254
NS	1	1	1	1	1	1	1	1	255

Table 1. RMTs of the rules. PS: present state; NS: next state; RMT: rule minterm.

Definition 1. The set RV $= \langle R_1, R_2, \ldots, R_i, \ldots, R_n \rangle$ of rules that configures the cells of a CA is called the rule vector of the CA.

Definition 2. If $R_1 = R_2 = \cdots = R_n$ in RV $= \langle R_1, R_2, \ldots, R_i, \ldots, R_n \rangle$, then the CA is a uniform CA; otherwise it is a nonuniform CA.

Definition 3. A CA is a null-boundary CA if its left (respectively right) neighbor of the leftmost (respectively rightmost) terminal cell is permanently fixed to 0-state.

The n-cell CA of Figure 1 is a null-boundary CA. The left neighbor of its leftmost cell and the right neighbor of the rightmost cell are "null."

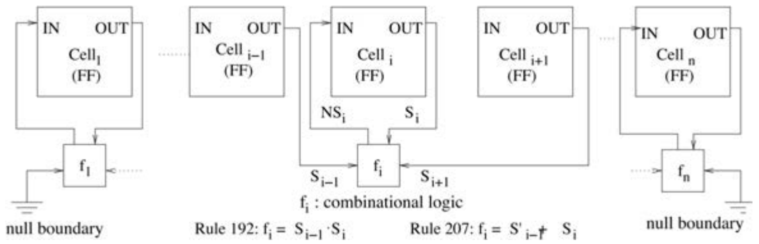

Figure 1. Null-boundary CA.

Definition 4. A combination of present states $(S_{i-1}^t, S_i^t, S_{i+1}^t)$, shown in the first row of Table 1, is referred to as the rule minterm (RMT).

The column 011 of Table 1 is the third RMT. The next states corresponding to this RMT are 1 for rules 14 and 15, and 0 for 192. An RMT is represented as T(m), $m = 0, 1, 2, 3, 4, 5, 6, 7$; and T($m$) $\in \{T\}$, where $T =$ T(0), T(1), …, T(7). The RMT for the i^{th} cell at time t is denoted as $T_i^t(m)$. For example, $T_1^t(0)$ denotes cell 1 RMT T(0) at time t. However, in the diagrams of this presentation, an RMT is represented only by the corresponding decimal number m; that is, $T_i^t(0)$ is represented as 0.

Definition 5. A rule divides the RMTs into two subsets, 0-RMT and 1-RMT, denoted as T / 0 and T / 1. The RMT for which the next state is 0 belongs to T / 0 and that having the next state as 1 belongs to T / 1. Hence, $T/0 \bigcap T/1 = \emptyset$ and $T/0 \bigcup T/1 = T$.

Definition 6. An RMT string (RS) is defined as a sequence of consecutive RMTs that appear in a state of the CA. It is represented as $(T_1, T_2, \ldots, T_n)$ for an n-cell CA, where $T_i \in T$. For example, the state 1011 of a four-cell null-boundary CA can be represented by the RS $(T_1, T_2, T_3, T_4) = (T(2), T(5), T(3), T(6))$.

Definition 7. An RMT x0y (respectively x1y), where x and y are the PSs of the left and right neighbors in a rule R is called passive (self-replicating) if it belongs to $T/0$ (respectively $T/1$). On the other hand, if an RMT x0y (respectively x1y) in R belongs to $T/1$ (respectively $T/0$), it is active (respectively, non-self-replicating). The RMT $T(0)(000) \in T/0$ and RMT $T(2)(010) \in T/1$ in rule 14 (Table 1). These two RMTs are passive. On the other hand, RMT $T(0)(000) \in T/1$ in rule 15 is active (Table 1).

Definition 8. A set of states of a CA can form a cycle ($0 \to 0$, $13 \to 13$, $11 \to 11$ and $15 \to 9 \to 15$ of Figure 2(a)) and is called an attractor. An attractor forms a basin with the states that lead to the attractor (($15 \to 9 \to 15$)-basin of Figure 2(a) contains three states including the attractor states nine and 15). The cycle $0 \to 0$ is a single length cycle attractor and called a fixed point (referred to as a fixed-point attractor in the rest of this paper). It forms the 10-attractor basin having only state 10.

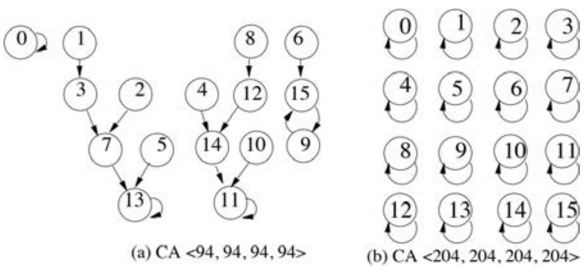

Figure 2. State transitions of SLCA.

Definition 9. The depth of a CA is the length of the longest path traversed from any state to an attractor during its state transitions. The depth of the CA shown in Figure 2(a) is three ($8 \to 12 \to 14 \to 11$).

Definition 10. A CA is reversible if its states form only cycles during its state transitions; otherwise, the CA is irreversible (Figure 2(a) and (b)).

Definition 11. Single length cycle attractor CAs (SLCAs) refer to the CAs that have at least one fixed-point attractor. Additional multilength cycle attractors may or may not be present in SLCAs.

The CA of Figure 2(a) has three fixed-point attractors (state 0, 13 and 11) and one multilength cycle attractor ($15 \to 9 \to 15$) and it is an SLCA. Figure 2(b) depicts the state transitions of a specific kind of SLCA with 2^n (n = length of CA) fixed-point attractors.

Definition 12. The SLCA producing a connected graph during its state transitions, that is, having only one fixed-point attractor and no multilength cycle, is called a single length cycle single-attractor CA (SACA).

An n-cell uniform CA with rule 112 creates a connected graph with a single fixed-point attractor for any value of n. The uniform CA with rule 112 is thus an SACA (Figure 3(a) for a four-cell CA) and rule 112 is an SACA rule.

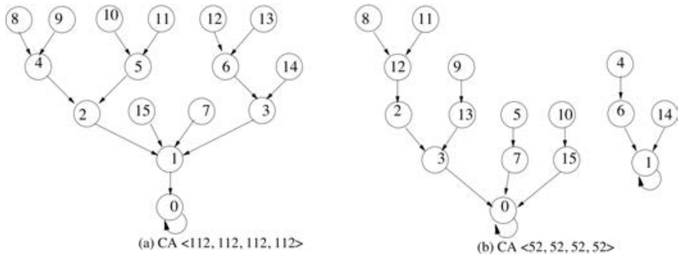

Figure 3. State transitions of the SACA and TACA.

Definition 13. During its state transition, if an SLCA forms exactly two connected components, each with one fixed-point attractor, then it is a single length cycle two-attractor CA (TACA).

A uniform CA with rule 52 creates precisely two connected components, each with one fixed-point attractor (as shown in Figure 3(b) for a four-cell CA). As a result, it is a TACA, thus rule 52 is a TACA rule.

Such a special class of irreversible CA having only two fixed points is our current interest.

3. Next-State Rule Minterm Transition Diagram

The graph-based tool referred to as the next-state RMT transition diagram (NSRTD), introduced in [20] for uniform CAs, has been employed to get more insight into the state transition behavior of SLCAs. We employ NSRTD to characterize the class of TACA rules. The NSRTD provides a generic methodology to determine the presence of fixed-point attractors as well as the multilength cycle attractors in a CA and, therefore, enables identification of rules that form only fixed-point attractors. The following terminologies are essential for describing the NSRTD.

Definition 14. If T_i^t is the RMT of the ith-cell rule, based on which the cell i changes its state at the tth time step, then the next cell RMTs (NCRs) of T_i^t are the RMTs $T_{i+1}^t = (2 \times T_i^t \bmod 8)$, $((2 \times T_i^t + 1) \bmod 8)$ of the $(i+1)$th-cell rule, based on which the $(i+1)$th cell can change its state at the tth time step.

For example, if the RMT of the ith-cell rule R_i is T(1), then its NCRs are T(2) ($= 2 \times \mathrm{T}(1) \bmod 8$) and T(3) ($= (2 \times \mathrm{T}(1) + 1) \bmod 8$). That is, if cell i changes its state on RMT $T_i(1)$ (T(1) of rule R_i) at time t, then the $(i+1)$th cell can change its state on RMTs $T_{i+1}(2)$ or $T_{i+1}(3)$ of the $(i+1)$th-cell rule R_{i+1} at time t. All such NCRs of the RMTs are shown in Table 2.

RMT T_i	NCR of RMT T_i
(ith-cell rule)	(RMTs of $(i+1)$th-cell rule)
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 2. Relationship between T_i and T_{i+1} (NCR).

Definition 15. If T_i^t (respectively T_i^{t+1}) is the RMT of the ith-cell rule R_i, based on which cell i changes its state at time t (respectively at time $t+1$), then T_i^{t+1} is the next-state RMT (NSR) of T_i^t for cell i.

Definition 16. The sequence of NSRs based on which the ith cell changes its state during the state transitions of the CA is called a next-state RMT sequence (NSRS). That is, an NSRS for the ith cell (denoted as NSRS_i) is $T_i^0, T_i^1, \ldots, T_i^t, T_i^{t+1}, \ldots$ where T_i^{t+1} is the NSR of T_i^t.

Definition 17. For the i^{th}-cell rule R_i of a CA, all possible NSRSs can be represented by a directed graph G(V,E), called an NSRS graph (NSRS-G), where all the $\mathrm{T}_i^t \in V$ and $e_i\left(\mathrm{T}_i^t, \mathrm{T}_i^{t+1}\right) \in E$, if and only if T_i^{t+1} is the NSR of T_i^t.

Definition 18. The NCR, NSR, NSRS and NSRS-G of a CA enable construction of the set of directed graphs, called NSRTD, as a two-dimensional arrangement of nodes. Each node is an RMT and each graph in NSRTD represents an attractor of the CA. The edge $\left(\mathrm{T}_i^t, \mathrm{T}_i^{t+1}\right) \in E$, such that T_i^{t+1} is the NSR of T_i^t and also the edge $(\mathrm{T}_i^t, \mathrm{T}_{i+1}^t) \in E$, such that T_{i+1}^t is the NCR of T_i^t. Further, the nodes T_i^t $(\forall\, t)$ form an NSRS for the i^{th} cell.

Definition 19. Two NSRSs defined for the i^{th} and $(i+1)^{\text{th}}$ CA cell (NSRS_i and NSRS_{i+1}) are called compatible if the j^{th} NSR of the two are related in the sense that the j^{th} NSR of NSRS_{i+1} is the NCR (Table 2) of the j^{th} NSR of NSRS_i.

The following subsections explain the NSR, NSRS, NSRS-G, compatibility class of NSRSs and the construction of NSRTD.

3.1 Identification of Next-State Rule Minterms

If xyz is the RMT of rule R_i, based on which the i^{th} cell changes its state, and l and r are the PSs of cell $(i-2)$ and cell $(i+2)$, respectively, then the NSR of the cell can be $d_l d d_r$, where the RMT xyz of R_i is d, d_l = RMT lxy of the $(i-1)^{\text{th}}$-cell rule R_{i-1}, and d_r = RMT yzr of the $(i+1)^{\text{th}}$-cell rule R_{i+1}. For uniform CAs with rule R, the d_l, d and d_r are the RMTs lxy, xyz and yzr of R. In a null-boundary n-cell (cell_1, cell_2, …cell_i, …$\text{cell}_{(n-1)}$, cell_n) CA, the following conditions are satisfied.

1. For the first cell ($i = 1$): $d_l = 0$, $x = 0$, $r = 0/1$.
2. For the second cell ($i = 2$): $l = 0$, $r = 0/1$.
3. For the intermediate cells ($i = 3$, …$(n-2)$): $l = r = 0/1$.
4. For the second to last cell ($i = n-1$): $l = 0/1$, $r = 0$.
5. For the last cell ($i = n$): $l = 0/1$, $z = 0$, $d_r = 0$.

The derivation of NSRs for an n-cell null-boundary uniform CA with rule 58 is shown in Figure 4.

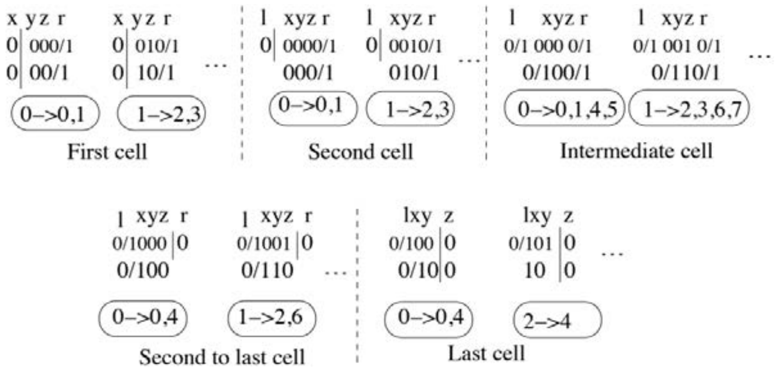

Figure 4. Identification of NSRs for a uniform CA with rule 58.

First Cell		Second Cell		Intermediate Cells	
1-10 RMT (1)	NSR (2)	RMT (3)	NSR (4)	RMT (5)	NSR (6)
T(0)	T(0), T(1)	T(0)	T(0), T(1)	T(0)	T(0), T(1), T(4), T(5)
T(1)	T(2), T(3)	T(1)	T(2), T(3)	T(1)	T(2), T(3), T(6), T(7)
T(2)	T(1)	T(2)	T(5)	T(2)	T(5)
T(3)	T(2)	T(3)	T(6)	T(3)	T(6)
		T(4)	T(2), T(3)	T(4)	T(2), T(3)
		T(5)	T(2), T(3)	T(5)	T(2), T(3)
		T(6)	T(5)	T(6)	T(1), T(5)
		T(7)	T(4)	T(7)	T(0), T(4)

Second to Last Cell		Last Cell	
RMT (7)	NSR (8)	RMT (9)	NSR (10)
T(0)	T(0), T(4)	T(0)	T(0), T(4)
T(1)	T(2), T(6)	T(2)	T(4)
T(2)	T(5)	T(4)	T(2)
T(3)	T(6)	T(6)	T(0), T(4)
T(4)	T(2)		
T(5)	T(2)		
T(6)	T(1), T(5)		
T(7)	T(0), T(4)		

Table 3. NSRs of the cells of a uniform CA with rule 58.

3.2 Construction of a Next-State Rule Minterm Sequence Graph

The NSRS-G for a CA is the union of all possible NSRSs corresponding to that cell. For example, for the first cell of the uniform CA with rule 58, NSR of RMT T(1) is T(2), and NSR of T(2) is T(1). Therefore, $T^t(1) \to T^{t+1}(2) \to T^{t+2}(1)$ forms one NSRS for the first cell (Figure 5(i)). The NSR of T(3) is T(2). Further, the NSRs of T(0) are T(0) itself and T(1) (columns 1 and 2 of Table 3). Therefore, the other NSRSs are $T^t(0) \to T^{t+1}(0) \to T^{t+2}(0) \to \ldots$ and $T^t(1) \to T^{t+1}(3) \to T^{t+2}(2) \to T^{t+3}(1) \to \ldots$ and others as shown in Figure 5(i). The union of all such NSRSs for the first cell is the NSRS-G (NSRS-G_1) for the first cell (Figure 5(iia)). The NSRS-G_2, NSRS-G_I, NSRS-$G_{(n-1)}$ and NSRS-G_n for the second cell, intermediate cells, second to

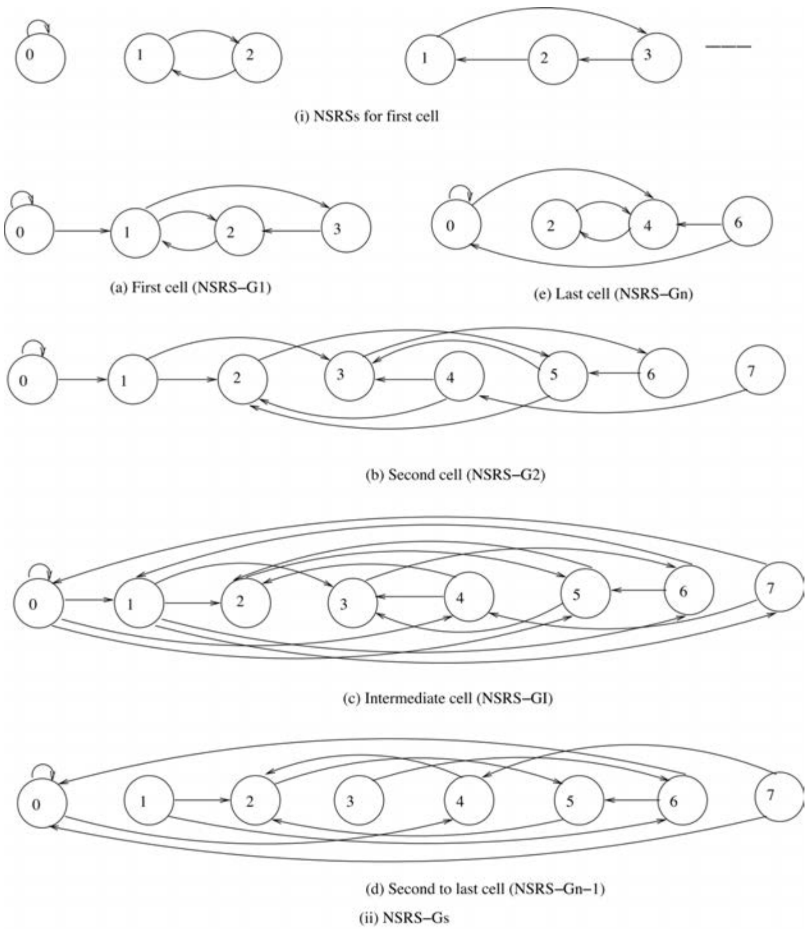
(i) NSRSs for first cell
(a) First cell (NSRS–G1)
(e) Last cell (NSRS–Gn)
(b) Second cell (NSRS–G2)
(c) Intermediate cell (NSRS–GI)
(d) Second to last cell (NSRS–Gn−1)
(ii) NSRS–Gs

Figure 5. NSRSG for uniform CA with rule 58.

last cell and the last cell of the CA are shown in Figure 5 (iib), (iic), (iid) and (iie), respectively. Since the first (leftmost) cell of the CA can change state only on RMTs T(0), T(1), T(2) and T(3) and the last (rightmost) cell can only change state on RMTs T(0), T(2), T(4) and T(6), the NSRS-G_1 and NSRS-G_n are composed of four nodes. However, NSRS-G_2, NSRS-G_I and NSRS-$G_{(n-1)}$ have eight nodes, as the second cell, intermediate cells and the second to last cell can change state on any of the eight RMTs.

3.3 Construction of a Next-State Rule Minterm Transition Diagram

The NSRTD of a uniform CA with rule 58 is shown in Figure 6. It has two graphs, namely G_1 and G_2, where G_1 corresponds to a fixed-point attractor and G_2 corresponds to multilength cycle attractors.

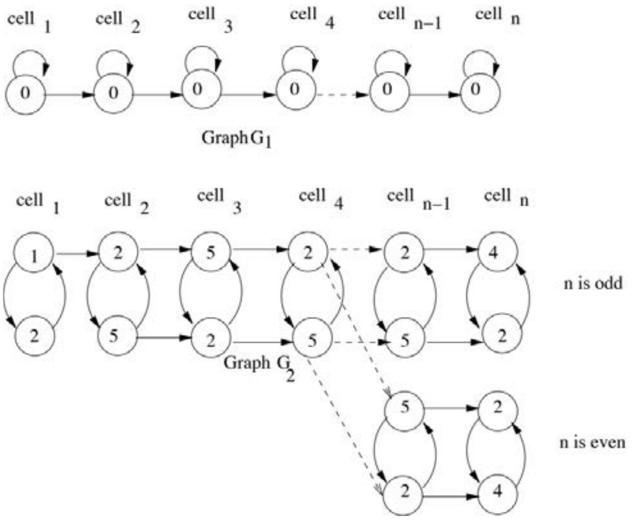

Figure 6. NSRTD for uniform CA with rule 58.

Given a rule R, length of CA n and the NCRs of the eight RMTs (Table 2), the NSRs for different cells (first, second, intermediate, second to last and last) are identified and the NSRS-Gs are formed. The cycles LPs in NSRS-Gs are then found. From the set of cycles LP_1, LP_2, LP_I, $LP_{(n-1)}$, LP_n for the first, second, intermediate, second to last and last cell, respectively, the compatibility classes of the cycle (Definition 19) are computed. Then the sequence of RMT $P_u = P_{1u} \to P_{2u} \to P_{3u} \to P_{4u} \to \ldots P_{(n-2)u} \to P_{(n-1)u} \to P_{nu}$ representing a graph in the NSRTD is extracted. The number of such n-length paths P_u corresponds to the number of attractors of the CA. If all the NSRSs P_{iu}, for $i = 1$ to n, in a P_u is a single length cycle, then P_u corresponds to a fixed point; otherwise it is a multilength cycle attractor.

Extensive experimentation with 256 CA rules in three-neighborhood null-boundary conditions has resulted in the identification of uniform TACA rules. Fifteen such rules are listed in Table 4. A CA configured with any of these rules forms a uniform TACA for all lengths. Depending upon the design requirement for the specific application, hybrid (nonuniform) TACA synthesis may be required. Hence, the next section elaborates the proposed methodology for synthesis of a nonuniform TACA for a given length.

Group	Rule for TACA
3	38, 52
4	46, 106, 116, 120, 166, 180, 235, 249
5	174, 239, 244, 253, 254

Table 4. TACA rules.

4. Proposed Methodology

We now describe in detail the proposed synthesis mechanisms for a TACA. A basic framework for synthesizing a nonuniform TACA has been reported in [26], which ensures an exponential time solution in all three cases (best, average and worst). In this paper, we attempt to devise a linear time synthesis mechanism for a TACA. The first approach ensures a linear time solution in the best case, but it takes exponential time in the worst case. Hence, we further develop a more efficient synthesis mechanism capable of ensuring a linear time solution in all cases (best, average and worst). However, by doing some nominal modifications, both the methodologies can be extended to synthesize CAs with any number of fixed point(s). A TACA, which has only two fixed points, can be useful in a variety of applications, especially when differentiating faulty states from nonfaulty ones. To come up with a solution to this kind of problem, the required TACA must be synthesized. Utilizing all 256 rules in the synthesis of such a TACA results in exponential complexity. To minimize the search space, we examine hybridization of 15 uniform TACA rules as identified in [16] and reported in Table 4 to construct a nonuniform TACA. However, the worst-case time complexity to synthesize a nonuniform/hybrid TACA of length n is still $O(15^n)$.

Algorithms 1 and 2 are provided for the synthesis of a scalable TACA structure that utilizes the 15 uniform TACA rules in a three-neighborhood null boundary.

Algorithm 1. Count cycles in a CA.

Input: Rule R, NCRs for RMTs T(0), T(1), T(2), … , T(7) forming the sets NCR_0, NCR_1, …NCR_7.
Output: Cycle in NRSR-Gs.

```
1: Start;
2: Find NSRs for the first, second, intermediate, second to last and last
   cells for rule $R$.
3: Construct NSRS-Gs for the first cell (NSRS-$G_1$), second cell (NSRS-
   $G_2$), intermediate cells (NSRS-$G_I$), second to last cell (NSRS-$G_{(n-1)}$)
   and last cell (NSRS-$G_n$);
   NSRS-G = {NSRS-$G_1$, NSRS-$G_2$, NSRS-$G_I$, NSRS-$G_{(n-1)}$, NSRS-$G_n$}.
4: Find cycles (simple cycles) in the set NSRS-G
   /*these include self-loops as well as multilength cycles*/
   $LP_1 = LP_{11}, LP_{12}$, …. for the first cell; /*cycles in NSRS-$G_1$*/
   $LP_2 = LP_{21}, LP_{22}$, …. for the second cell; /*cycles in NSRS-$G_2$*/
   $LP_I = LP_{I1}, LP_{I2}$, …. for intermediate cells; /*cycles in NSRS-$G_I$*/
   $LP_{(n-1)} = LP_{(n-1)1}, LP_{(n-1)2}$, …. for the second to last cell; /*cycles in
   NSRS-$G_{(n-1)}$*/
   $LP_n = LP_{n1}, LP_{n2}$, …. for the last cell; /*cycles in NSRS-$G_n$*/
   $LP = \{LP_1, LP_2, LP_I, LP_{(n-1)}, LP_n\}$;
5: Stop.
```

Algorithm 1 (as in [26]) computes all the single length as well as multilength cycles in all NSRSs for the first, second, intermediate, second to last and last cells. To do so, Algorithm 1 accepts rule R and NCRs of the eight RMTs (Table 2) as input. It first finds the NSRs of the first, second, intermediate, second to last and last cells. After that, it constructs NSRS-Gs for the first, second, intermediate, second to last and last cells. Finally, it computes all the cycles, which are utilized in Algorithm 2.

Algorithm 2 takes into account the length of the CA n, the NCRs of the eight RMTs (Table 2), and the cycles in the NSRS-Gs, which might be single length or multilength. To choose the first cell rule, Algorithm 2 selects one rule from Table 4 randomly, as depicted in step 1 of Algorithm 2. In step 3, Algorithm 2 chooses a rule R_p randomly from all the TACA rules and computes the compatibility classes C_{12} between LP_1 (R_1) and LP_2 (R_p). It assigns R_p as the second cell rule only if C_{12} has exactly two pairs of self-loop NSRSs and no pair of multilength NSRSs. If that condition is not satisfied, it further checks for another TACA rule. Following the same approach, Algorithm 2 chooses the third cell rule randomly from all TACA rules. If the length of the CA is more than five, it will continue to select intermediate cell rules $R_{(I+1)}$ for lengths three to $n-3$ by consulting the compatibility class $C_{I(I+1)}$ between LP_I (R_I) and LP_I $(R_{(I+1)})$ as discussed in steps 19, 20 and 21 of Algorithm 2. Next, it selects a TACA rule randomly and includes that rule as the second to last cell rule only if the compatibility class $C_{I(n-1)}$ has exactly two pairs of self-loop NSRSs and no multilength NSRSs. Finally, by following the same approach, Algorithm 2 chooses the last cell rule randomly, as shown in steps 35, 36 and 37 of Algorithm 2.

Algorithm 2. Construct a hybrid TACA.

Input: Length of CA n, NCRs, LPs (as computed by Algorithm 1).

Output: The rule vector RV of TACA $RV = \langle R_1, R_2, \ldots R_n \rangle$.

```
1:  Select rule $R_1$ randomly from Table 4;
2:  for $p = 1$ to 15
3:  Choose a TACA rule $R_p$ randomly and compute compatibility classes
    $C_{12} = \{(LP_{1i}, LP_{2j})\}$ where $C_{12}$ is the compatibility class of NSRSs
    between $LP_1$ $(R_1)$ and $LP_2$ $(R_p)$;
4:  if $C_{12}$ has exactly two pairs of self-loop NSRSs and no pair of
    multilength NSRSs then
5:      Assign $R_2 = R_p$;
6:      break;
7:  end if
8:  end for
9:  for $p = 1$ to 15
10: Choose a TACA rule $R_p$ randomly and compute compatibility classes
    $C_{2I} = \{(LP_{2k}, LP_{Il})\}$ where $C_{2I}$ is the compatibility class of NSRSs
    between $LP_2$ $(R_2)$ and $LP_I$ $(R_p)$;
11: if $C_{2I}$ has exactly two pairs of self-loop NSRSs and no pair of
    multilength NSRSs then
12:     Assign $R_3 = R_p$;
13:      break;
14: end if
15: end for
16: if $n > 5$ then
17:     for $i = 3$ to $(n - 3)$
18:     for $p = 1$ to 15
```

```
19:         Choose a TACA rule $R_p$ randomly and compute compatibility
            classes $C_{I(I+1)} = \{(LP_{Im}, LP_{(I+1)o}\}$ where $C_{I(I+1)}$ is the compatibility
            class of NSRSs between $LP_I$ $(R_I)$ and $LP_{I+1}$ $(R_p)$;
20:         **if** $C_{I(I+1)}$ has exactly two pairs of self-loop NSRSs and no pair of
            multilength NSRSs **then**
21:             Assign $R_{I+1} = R_p$;
22:             break;
23:         **end if**
24:     **end for**
25:     **end for**
26: **end if**
27: **for** $p = 1$ to 15
28: Choose a TACA rule $R_p$ randomly and compute compatibility classes
    $C_{I(n-1)} = \{(LP_{Iq}, LP_{(n-1)r})\}$ where $C_{I(n-1)}$ is the compatibility class of
    NSRSs between $LP_I$ $(R_I)$ and $LP_{(n-1)}$ $(R_p)$;
29: **if** $C_{I(n-1)}$ has exactly two pairs of self-loop NSRSs and no pair of
    multilength NSRSs **then**
30:     Assign $R_{n-1} = R_p$;
31:     break;
32: **end if**
33: **end for**
34: **for** $p = 1$ to 15
35: Choose a TACA rule $R_p$ randomly and compute compatibility classes
    $C_{(n-1)n} = \{(LP_{(n-1)s}, LP_{nt})\}$ where $C_{(n-1)n}$ is the compatibility class of
    NSRSs between $LP_{(n-1)}$ $(R_{(n-1)})$ and $LP_n$ $(R_p)$;
36: **if** $C_{(n-1)n}$ has exactly two pairs of self-loop NSRSs and no pair of
    multilength NSRSs **then**
37:     Assign $R_n = R_p$;
38:     break;
39: **end if**
40: **end for**
41: Output: $\langle R_1, R_2, \ldots, R_n \rangle$
```

Example 1. Here, to illustrate the proposed synthesis mechanism following Algorithm 2, we consider the synthesis of an n-length $(n = 6)$ nonuniform TACA. In step 1 of Algorithm 2, it chooses the first cell rule $R_1 = 166$ randomly from Table 4. Next in step 2, Algorithm 2 selects rule 38 randomly and computes the compatibility class C_{12} between $LP_1(166)$ and $LP_2(38)$. As the compatibility class $C_{12} = \{(T_1(0), T_2(0)), (T_1(2), T_2(4))\}$ has exactly two pairs of self-loop NSRSs, Algorithm 2 does not further check for any other TACA rules and assigns rule 38 as the second cell rule. Following the same procedure, rule 46 is chosen as the third cell rule R_3 as the compatibility class between $LP_2(38)$ and $LP_I(46)$: $C_{23} = \{(T_2(0), T_3(0)), (T_2(4), T_3(0))\}$ has exactly two pairs of self-loop NSRSs. Again, to choose the fourth cell rule, it assigns rule 174 as the compatibility class between $LP_I(46)$ and $LP_I(174)$: $C_{34} = \{(T_3(0), T_4(0)), (T_3(4), T_4(0))\}$ has exactly two pairs of self-loop NSRSs. Similarly, rule 106 is selected as the fifth cell rule as the compatibility class between $LP_I(174)$ and $LP_{(n-1)}(106)$: $C_{45} = \{(T_4(0), T_5(0)), (T_4(4), T_5(0))\}$ has exactly two pairs of self-loop NSRSs. Finally, in step 34, Algorithm 2 chooses rule 180 randomly as the compatibility class between $LP_{(n-1)}(106)$ and $LP_n(180)$: $C_{56} = \{(T_5(0), T_6(0)), (T_5(4), T_6(0))\}$ has exactly two pairs of self-loop NSRSs, as shown in Figure 7. The NSRTD of a hybrid CA with rule vector $\langle 166, 38, 46, 174, 106, 180\rangle$ is shown in Figure 8. From Figure 8, we can observe that there exist only two paths with self-loop NSRSs, that is, $(T_1(0), T_2(0), T_3(0), T_4(0), T_5(0), T_6(0))$ and $(T_1(2), T_2(4), T_3(0), T_4(0), T_5(0), T_6(0))$, and no path with multi-length cycle(s). So, the six-cell CA $\langle 166, 38, 46, 174, 106, 180\rangle$ thus constructed is a TACA.

Analysis: In the first step of Algorithm 2, a rule is picked up at random from among the 15 TACA rules as listed in Table 4, and this step takes a constant amount of time. Even though the selection of rule R_2 has 15 possibilities, it may happen that after very few search attempts, we can get a candidate rule for which the compatibility class C_{12} contains precisely two pairs of self-loop NSRSs and no NSRSs with multilength. Once a suitable candidate rule is obtained, the searching process for other alternative candidates stops. In the same way, we choose the rules for all the cells. For each cell, we will get a suitable rule after a certain number of search attempts, which is much less than 15. Suppose we find the candidate rule R_I for the I^{th} cell (for any value of $I = 2, 3, \ldots, n$) with at most P search attempts, such that the compatibility class $C_{(I-1)I}$ contains precisely two pairs of self-loop NSRSs and no NSRSs of multilength. So, the amount of time it takes to choose rules for a CA with n cells is P^n. Although, in the worst case, we may need to search all the 15 rules and hence, the worst-case time complexity is 15^n. The algorithm proposed in [26] has the time complexities 15^n for all the cases (best, average and

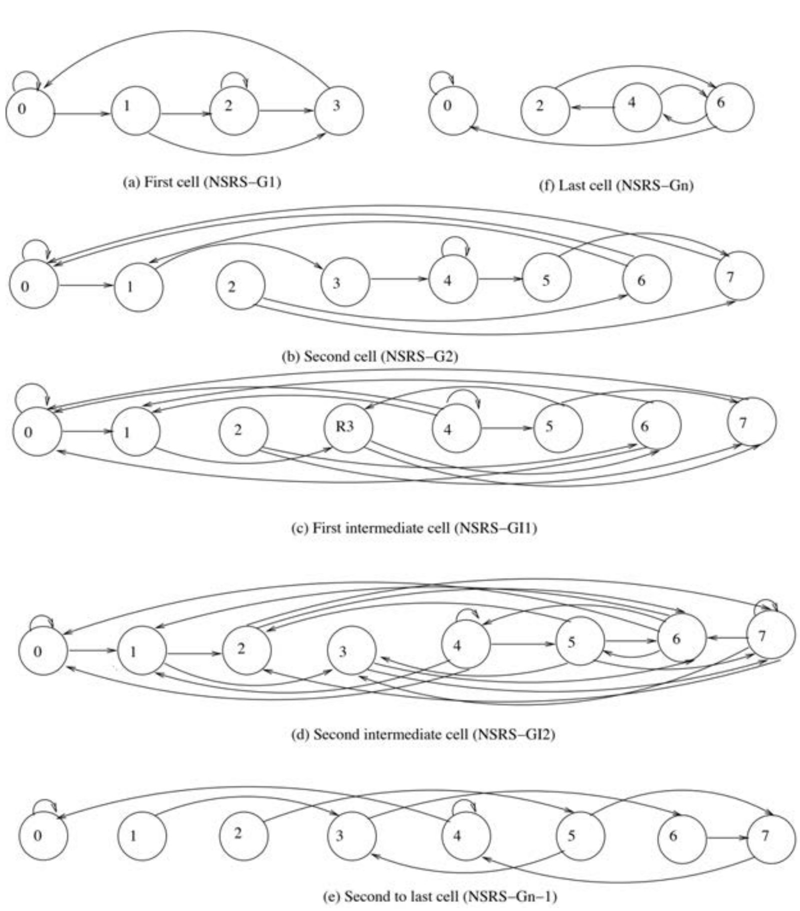
(a) First cell (NSRS–G1)
(f) Last cell (NSRS–Gn)
(b) Second cell (NSRS–G2)
(c) First intermediate cell (NSRS–GI1)
(d) Second intermediate cell (NSRS–GI2)
(e) Second to last cell (NSRS–Gn–1)

Figure 7. NSRS-Gs for the hybrid CA with rule vector ⟨166, 38, 46, 174, 106, 180⟩.

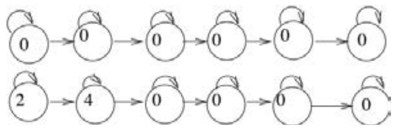

Figure 8. NSRTD for the hybrid CA with rule vector ⟨166, 38, 46, 174, 106, 180⟩.

worst), as for all the cases it computes a candidate set for a cell containing all the possible candidate rules for that cell and later, only one rule is picked up from the set to form the rule vector. However, in the case of Algorithm 2, we can achieve average case time complexity of P^n where $P \ll 15$ and the best case time complexity is n. However, to synthesize a nonuniform TACA, if we consider all 256 CA rules instead of only 15 rules in Table 4, Algorithm 2 may take $O(256^n)$ time in the worst case. Figure 9 shows the synthesis time of the nonuniform TACA with rule vector $\langle 180, 52, 116, 244, 52\rangle$ considering different lengths. Here, the first, second, second to last and last cell rules are 108, 52, 244 and 52, respectively, and the intermediate cell rule 116 is repeated as we increase the length of the CA from 5 to 16. The experimental results show that the synthesis time increases linearly with an increase in the length of the CA. Although we have minimized the complexity of constructing a hybrid TACA, our aim is to design an algorithm that can construct a hybrid TACA in linear time.

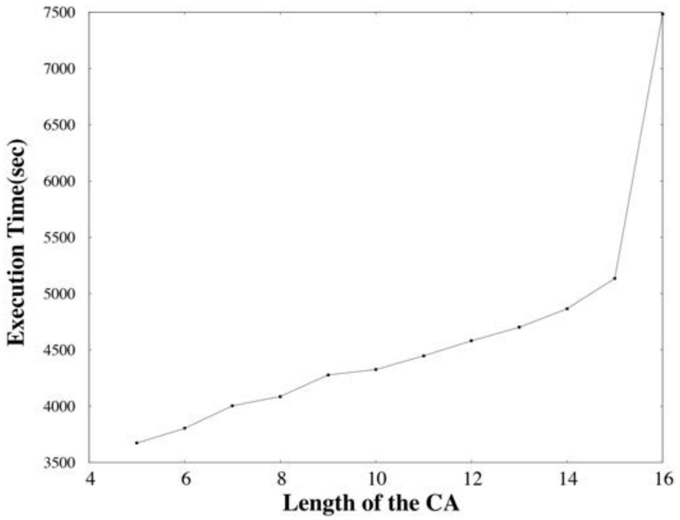

Figure 9. Synthesis time of the nonuniform TACA for varying length.

4.1 Synthesis of Single Length Cycle Two-Attractor Cellular Automata in Linear Time

The previous section reports the synthesis of a TACA wherein Algorithm 2 takes exponential time. In this section, we attempt to devise an algorithm for synthesizing a TACA in linear time. Here also, we have considered the uniform TACA rules as reported in Table 4, for synthesizing a nonuniform TACA in linear time. However, the methodology developed can also be utilized for synthesizing CAs with any specified number of fixed points.

To realize our objective, we have considered the NSRS-Gs for the first, second, intermediate, second to last and last cells for each of the 15 TACA rules. From the obtained NSRS-Gs, we have computed the compatibility classes between two consecutive cells by analyzing the cycle structure of each of the 15 TACA rules. Finally, by analyzing the compatibility classes between two consecutive cells, all the TACA rules have been organized in four different tables: Tables 5–8. The candidate rules for the first intermediate cell from a chosen second cell rule are listed in Table 6. Table 7 shows the relationship between the last intermediate cell rule and the second to last cell rule. Finally, Table 8 lists the class relationship between the second to last cell rule and last cell rule.

Tables are constructed by following the methodology.

- All the 15 TACA rules have been divided into eight different groups based on their cycle structure. For example, all the rules in group I (38, 46, 166, 174) have the same self-loop NSRSs in the first cell ($T_1(0)$, $T_1(1)$), second cell ($T_2(0)$, $T_2(4)$), intermediate cell ($T_I(0)$, $T_I(4)$), second to last cell ($T_{(n-1)}(0)$, $T_{(n-1)}(4)$) and last cell ($T_n(0)$, $T_n(4)$).
- Table 5 is constructed by checking whether the compatibility class between LP_1 (R_1) and LP_2 (R_2): C_{12} has exactly two pairs of self-loop NSRSs and no pair of multilength NSRSs. For example, if the first cell rule is selected from group I, then the second cell rule can be chosen from either group I or II as listed in column 3 of Table 5. Any rule from group I or II can be selected for the second cell rule because the compatibility class between LP_1 (any rule from group I) and LP_2 (any rule from group I) $C_{12} = \{(T_1(0), T_2(0)), (T_1(2), T_2(4))\}$ has exactly two pairs of self-loop NSRSs and the compatibility class between LP_1 (any rule from group I) and LP_2 (any rule from group II) $C_{12} = \{(T_1(0), T_2(0)), (T_1(0), T_2(1))\}$ has exactly two pairs of self-loop NSRSs.

Group	Rules for R_1	Class for R_2
I	38, 46, 166, 174	I, II
II	52, 116, 180, 244	II, IV, V, VI, VIII
III	106	II, III, IV, V, VI, VIII
IV	120	II, IV
V	235	V, VI
VI	239	V, VI
VII	249, 253	VII
VIII	254	III, V, VI, VIII

Table 5. Class relationship between R_1 and R_2.

- Table 6 is constructed by checking whether the compatibility class between LP_2 (R_2) and LP_I (R_I): C_{2I} has exactly two pairs of self-loop NSRSs and no pair of multilength NSRSs.

Rules for R_2	Class for First Intermediate cell
I	I
II	II,
III	I, II, III
IV	II, IV
V	V, VI
VI	V, VI
VII	VI
VIII	VIII

Table 6. Class relationship between R_2 and R_i.

- Table 7 is constructed by checking whether the compatibility class between LP_I (R_I) and $\mathrm{LP}_{(n-1)}$ ($R_{(n-1)}$): $\mathrm{C}_{I(n-1)}$ has exactly two pairs of self-loop NSRSs and no pair of multilength NSRSs.

Rules for Last Intermediate Cell	Class for R_{n-1}
I	I, III
II	II
III	I, III
IV	II, IV
V	V, VI
VI	V, VI
VII	VII
VIII	I, VIII

Table 7. Class relationship between R_i and R_{n-1}.

- Table 8 is constructed by checking whether the compatibility class between $\mathrm{LP}_{(n-1)}$ ($R_{(n-1)}$) and LP_n (R_n): $\mathrm{C}_{(n-1)n}$ has exactly two pairs of self-loop NSRSs and no pair of multilength NSRSs.

Rules for R_{n-1}	Class for R_n
I	I, II, III, IV
II	II
III	I, II, III
IV	IV, VIII
V	V, VI
VI	V, VI
VII	III, VI, VII, VIII
VIII	I, VIII

Table 8. Class relationship between R_{n-1} and R_n.

Class relationship between R_1 and R_2 (Table 5): In Table 5, we have classified all 15 TACA rules into eight groups according to a self-loop NSRS in each cell. The first column of Table 5 shows all the eight groups, and the second column represents the members in the groups that can be chosen as the first cell rule. The third column of Table 5 shows the groups for second cell rule R_2 when R_1 is selected as per column 2. The main criteria for selecting rule R_2 is that it must be compatible with the chosen rule R_1. That is, the self-loop NSRS of rule R_2 must be the NCR of R_1. For example, if rule 38 is chosen as the first cell rule from Table 5, then the second cell rule R_2 can be selected either from group I or group II. Like all other rules from group I, 46 is also compatible with rule 38 as the self-loop NSRS on RMT T(0) of rule 46 is the NCR of the self-loop NSRS on RMT T(0) of rule 38 and the self-loop NSRS on RMT T(4) of rule 46 is the NCR of the self-loop NSRS on T(2) of rule 38. That is, the compatibility class between R_1 and R_2 has exactly two members.

Class relationship between R_2 and R_i (Table 6): The first intermediate cell rule, that is, R_3, can be selected from Table 6. Following a similar strategy, rule R_3 is selected by checking the compatibility class between R_2 and R_3. For example, if rule 106 is chosen for the second cell rule R_2, we can then choose any of the rules from groups I, II and III as a candidate rule for R_3. Like all other rules from group II, 116 is also compatible with rule 106 as the self-loop NSRS on RMT T(0) of rule 116 is the NCR of the self-loop NSRS on RMT T(0) of rule 106, and the self-loop NSRS on RMT T(1) of rule 116 is the NCR of the self-loop NSRS on T(0) of rule 106. That is, the compatibility class between R_2 and R_3 has exactly two members. While selecting the rules for intermediate cells R_i, additional constraints need to be considered. The rules for intermediate cell R_i and R_{i+1} must belong to the same group; that is, if rule 52 (member of group II) is selected as R_i, then the next rule R_{i+1} must also belong to the same group, that is, group II.

Class relationship between R_i and R_{n-1} (Table 7): The second to last cell rule can be selected from Table 7. The first column of Table 7 lists the class of the final intermediate cell rule, and the second column lists the class of candidate rules for the second to last cell rule R_{n-1}. For example, if rule 120 is chosen for R_i then any member of groups II or IV can be selected as the second to last cell rule. So, rule 180 can be selected as the second to last cell rule as the self-loop NSRS on RMT T(0) of rule 180 is the NCR of the self-loop NSRS on RMT T(0) of rule 120, and the self-loop NSRS on RMT T(1) of rule 180 is the NCR of the self-loop NSRS on T(2) of rule 120. That is, the compatibility class between R_i and R_{n-1} has exactly two members.

Class relationship between R_{n-1} and R_n (Table 8): Based on the group defined in Table 5, the last cell rule can be selected from

Table 8. For example, rule 249, which is a member of group VIII, is chosen as the second to last cell, (i.e., R_{n-1}); then any rule from groups III, VI, VII and VIII can be chosen as the last cell rule R_n. So, rule 254 can be selected as the last cell rule as the self-loop NSRS on RMT T(6) of rule 254 is the NCR of the self-loop NSRS on RMT T(3) of rule 249, and the self-loop NSRS on RMT T(6) of rule 254 is the NCR of the self-loop NSRS on T(7) of rule 249. That is, the compatibility class between R_{n-1} and R_n has exactly two members.

Algorithm 3 formalizes the steps for the synthesis of an n-cell TACA.

Algorithm 3. Construct a hybrid TACA in linear time.

```
Input: length of TACA ($n$), Tables 5–8
Output: The CA rule vector RV $= \langle R_1, R_2, \ldots, R_n \rangle$
 1: Start;
 2: Choose rule $R_1$ from Table 5 randomly from any row (group I to
    VIII);
 3: Determine the class of $R_2$ ($cl_2$) from the appropriate row of Table 5;
 4: $R_2$ must be chosen from the appropriate row of $cl_2$ of Table 5;
 5: Determine the class of $R_3$ ($cl_3$) from the appropriate row of Table 6;
 6: $R_3$ must be chosen from the appropriate row of $cl_3$ of Table 6;
 7: **if** $n > 5$ **then**
 8:     **for** $i = 3$ to $(n - 3)$ **do**
 9:         $cl_{i+1}=cl_i$;
10:         Choose $R_{i+1}$ from the same group as rule $R_i$;
11:     **end for**
12: **end if**
13: Determine the class of $R_{n-1}$ ($cl_{n-1}$) from the appropriate row of
    Table 7;
14: $R_{n-1}$ must be chosen from the appropriate row of $cl_{n-1}$ of Table 7;
15: Determine the class of $R_n$ ($cl_n$) from the appropriate row of Table 8;
16: $R_n$ must be chosen from the appropriate row of $cl_n$ of Table 8;
17: Output: $\langle R_1, R_2, \ldots, R_n \rangle$
18: Stop.
```

Complexity: Algorithm 3 selects rules R_1, R_2, R_3, R_{n-1} and R_n from Tables 5–8, respectively. Each rule selection takes O(1) time. If the length of the CA is greater than five, then the loop is iterated for $n - 5$ times and each iteration takes constant time. Therefore, the running time of Algorithm 3 is $O(n)$.

Example 2. Consider synthesis of a five-length hybrid TACA using Algorithm 3. At step 1, rule R_1 is randomly chosen from Table 5. For example, rule 52 is selected for the first cell rule. As the first cell rule 52 belongs to group II, the second cell rule R_2 must belong to any one of the groups II, IV, V, VI or VIII. Suppose rule 235 (from group V) is selected as R_2. From Table 6, we can observe that the next cell rule must belong to either group V or VI. Now, rule 239 (member of group VI) is selected as R_3. Similarly, the next cell rule R_4 must belong to either group V or VI as listed in Table 7. So rule 239 is selected, as it is a member of group VI. Finally, rule 235 is selected as the last cell rule. The candidate rule for the last cell must belong to either group V or VI, and 235 is selected as it is a member of group V. From Figure 10, we can observe that there exist only two paths with self-loop NSRSs, that is, ($T_1(1)$, $T_2(3)$, $T_3(7)$, $T_4(6)$, $T_5(4)$) and ($T_1(1)$, $T_2(3)$, $T_3(7)$, $T_7(7)$, $T_5(6)$) in the NSRTD and no additional path with multilength NSRSs. So the resulting hybrid CA ⟨52, 235, 239, 239, 235⟩ is a TACA.

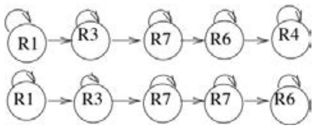

Figure 10. NSRTD for hybrid CA with rule vector ⟨52, 235, 239, 239, 235⟩.

5. Conclusion

Cellular automata (CAs) have progressed since von Neumann's early days through Wolfram's elementary form, and eventually to the current research trends using this straightforward yet attractive model. Along the way, different forms of CAs targeting various applications have been developed. In this paper, we have discussed the next-state rule minterm transition diagram (NSRTD) in order to gain a deeper understanding of how CAs with fixed points behave during state transitions, and an algorithm for synthesizing a nonuniform single length cycle two-attractor CA (TACA) in linear time is proposed.

References

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

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

[3] E. F. Moore, "Machine Models of Self Reproduction," in *Proceedings of Symposia in Applied Mathematics*, Vol. 14, (R. E. Bellman, ed.), Providence, RI: American Mathematical Society, 1962 pp. 17–33.

[4] D. A. Rosenblueth and C. Gershenson, "A Model of City Traffic Based on Elementary Cellular Automata," *Complex Systems*, **19**(4), 2011 pp. 305–322. doi:10.25088/ComplexSystems.19.4.305.

[5] K. Sutner, "De Bruijn Graphs and Linear Cellular Automata," *Complex Systems*, **5**(1), 1991 pp. 19–30. complex-systems.com/pdf/05-1-3.pdf.

[6] T. Yang, "Morphology Analysis of Urban Traffic Based on Multilevel Cellular Automata Networks Dynamics," in *Proceedings of 6th International Congress on Image and Signal Processing (CISP)*, Hangzhou, China, Piscataway, NJ: IEEE 2013 pp. 980–984. doi:10.1109/cisp.2013.6745307.

[7] L. Zaloudek and L. Sekanina, "Increasing Fault-Tolerance in Cellular Automata-Based Systems," in *Unconventional Computation (UC'11)*, Turku, Finland (C. S. Calude, J. Kari, I. Petre and G. Rozenberg, eds.), Berlin, Heidelberg: Springer-Verlag, 2011 pp. 234–245. doi:10.1007/978-3-642-21341-0_26.

[8] T. Toffoli and N. Margolus, *Cellular Automata Machines: A New Environment for Modeling*, Cambridge, MA: MIT Press, 1987.

[9] N. Bacquey, "Complexity Classes on Spatially Periodic Cellular Automata," in *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, Lyon, France (E. W. Mayr and N. Portier, eds.), Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014 pp. 112–124.

[10] H. Betel, P. P. B. de Oliveira and P. Flocchini, "On the Parity Problem in One-Dimensional Cellular Automata," *Electronic Proceedings in Theoretical Computer Science*, **90**, 2012 pp. 110–126. doi:10.4204/eptcs.90.9.

[11] A. Wuensche and M. Lesser, *The Global Dynamics of Cellular Automata*, Santa Fe Institute Studies in the Sciences of Complexity, Reading, MA: Addison-Wesley, 1992.

[12] P. Pal Chaudhuri, D. R. Chowdhury, S. Nandi and S. Chattopadhyay, *Additive Cellular Automata: Theory and Applications*, Los Alamitos, CA: IEEE Computer Society Press, 1997.

[13] P. Dasgupta, S. Chattopadhyay and I. Sengupta, "An ASIC for Cellular Automata Based Message Authentication," in *Proceedings of the 13th International Conference on VLSI Design (VLSID '00)*, Calcutta, India, Los Alamitos, CA: IEEE Computer Society, 1999 pp. 538–541. doi:10.1109/ICVD.2000.812663.

[14] H. Gutowitz, “Cryptography with Dynamical Systems,” *Cellular Automata and Cooperative Systems* (N. Boccara, E. Goles, S. Martinez and P. Picco, eds.), Dordrecht: Springer, 1993, pp. 237–274. doi:10.1007/978-94-011-1691-6_21.

[15] M. Mukherjee, N. Ganguly, A. Chaudhuri and P. Pal Chaudhuri, “Cellular Automata Based Authentication (CAA),” in *Cellular Automata (ACRI 2002)*, Geneva, Switzerland (S. Bandini, B. Chopard and M. Tomassini, eds.), Berlin, Heidelberg: Springer, 2002 pp. 259–269. doi:10.1007/3-540-45830-1_25.

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

[17] 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.

[18] 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 *Cellular Automata (ACRI 2010)*, Ascoli Piceno, Italy (S. Bandini, S. Manzoni, H. Umeo and G. Vizzari, eds.), Berlin, Heidelberg: Springer, 2010 pp. 300–311. doi:10.1007/978-3-642-15979-4_32.

[19] M. Saha, M. Dalui and B. 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.

[20] M. Dalui, “Theory and Application of Cellular Automata for CMPs Cache System Protocol Design and Verification,” Ph.D. thesis, IIEST Shibpur, West Bengal, India, 2014.

[21] N. Ganguly, “Cellular Automata Evolution: Theory and Applications in Pattern Recognition and Classification,” Ph.D. thesis, Bengal Engineering College (DU), 2004.

[22] P. Maji, “Cellular Automata Evolution for Pattern Recognition,” Ph.D. thesis, Bengal Engineering College (DU), 2004.

[23] H. V. McIntosh, *Linear Cellular Automata via de Bruijn Diagrams*, preprint, 1991.

[24] N. S. Maiti, S. Ghosh, B. K. Sikdar and P. Pal 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.

[25] N. S. Maiti, S. Ghosh, S. Munshi and P. Pal 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.

[26] 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)*, Howrah, India (S. Das and G. J. Martinez, eds.), Singapore: Springer, 2022 pp. 235–246. doi:10.1007/978-981-19-0542-1_17.