Exploring CA State Space to Synthesize Cellular Automata with Specified Attractor Set*

Sukanta Das[1], Chandrama Shaw[1], and Biplab K. Sikdar[2]

[1] Dept. of Information Technology
{sukanta,chandrama}@it.becs.ac.in
[2] Dept. of Computer Science & Technology
biplab@cs.becs.ac.in
Bengal Engineering & Science University, Shibpur, West Bengal, India, 711103

Abstract. The cellular automaton (CA) with multiple attractors in its state space creates immense interest to devise solutions for pattern classification, pattern recognition, design of associative memory, query processing, etc. This work characterizes the CA state space to explore the essential properties of 1-dimensional nonlinear cellular automata with single cycle attractors. The characterization of pseudo-exhaustive bits (PE bits) is done to uniquely identify the attractor set of such a CA. Theoretical framework thus evolved provides means to synthesize a CA for a given attractor set with specified PE bits.

Keywords: Nonlinear cellular automata, attractor, MACA, PE bit, classifier.

1 Introduction

The concept of Cellular Automata (CA) was initiated in 1950s by von Neumann and Ulam [9]. Neumann's CA involved 5-neighborhood interactions among the cells with 29 states per cell. Researchers had tried to view rather simplified structure of CA with the target to characterize its behavior, essentially keeping the flavour as that of Neumann's model [1,3,8]. In early 1980s, Stephen Wolfram [10] studied a family of simple 3-neighborhood 1-dimensional cellular automata [7] with two states per cell. This structure attracted a large section of researchers working in diverse fields [2].

While characterizing the CA state space, the researchers identified a set of CA states towards which the neighboring states asymptotically approach in the course of dynamic evolution [11]. They referred this set of states as the attractor of CA state space forming a basin of attractions. The CA with multiple attractors in its state space were of primary interest [2,6]. The single cycle attractor is one where the number of states of an attractor is one [2]. Characterization of single cycle attractors of the linear/ additive CA has been reported in [2,5,6].

* This research work is supported by the Sponsored CA Research Projects, Bengal Engineering and Science University, Shibpur, West Bengal, India-711103.

H. Umeo et al. (Eds): ACRI 2008, LNCS 5191, pp. 152–159, 2008.

The applications of linear/ additive CA having multiple single cycle attractors are also investigated. However, the characterization of single cycle attractors in nonlinear CA state space is yet to be explored.

The above scenario motivates us to concentrate on the characterization of single cycle attractors in 1-dimensional cellular automata [10]. A theoretical framework has been developed to explore all the attractors of such a CA. This further enables identification of pseudo-exhaustive (PE) bits [2] to address an attractor and then synthesis of a CA for the given set of attractors & PE bits. The algorithms/ schemes are proposed for efficient synthesis of such desired CA.

The preliminaries of CA, relevant for this work, have been reported in the twin paper (*Characterization of Non-reachable States in Irreversible CA State Space*). In the next section, we report the proposed characterization.

2 Characterization of Single Cycle Attractors

This section detailed out the theoretical framework developed to explore the single cycle attractors of a given CA and the PE bits specifying the attractors. It is based on the analysis of RMTs [4] of each cell rule of the CA.

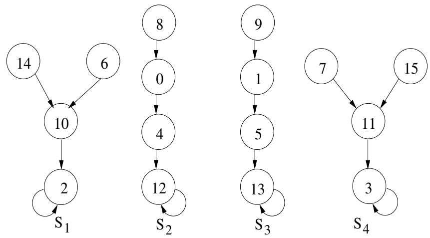

Fig. 1. State transitions of a CA with rule vector $< 10, 69, 204, 68 >$

2.1 Identification of Attractors of a CA

Since the next state (NS) of a single cycle attractor is the attractor itself (*Fig.1*), there should be at least one RMT (*Table 1*) of each cell rule ($\mathcal{R}_i$) for which the CA cell (i) does not change its state. For example, RMT $x0x$ ($x = 0/1$) is considered to find the NS of cell i when the current states of its left neighbor ($(i-1)^{th}$ cell), self and right neighbor ((($i+1)^{th}$) cell) are x, 0 and x respectively. It implies, if such RMT is '0', the state change of cell (i) is $0 \rightarrow 0$. That is, for the rule $\mathcal{R}_i$, if RMT 0 (000), 1 (001), 4 (100) or 5 (101) is 0, the CA cell i configured with $\mathcal{R}_i$ does not change its state. Similarly, if the RMT 2 (010), 3 (011), 6 (110) or 7 (111) is 1, it ensures a cell configured with $\mathcal{R}_i$ can stick to its current state. For example, all RMTs of rule 204 help formation of attractors (*Fig.2*).

Property 1: A rule $\mathcal{R}_i$ can contribute to the formation of single cycle attractor(s) if at least one of the RMTs 0, 1, 4 or 5 is 0, or the RMT 2, 3, 6 or 7 is 1.

RMTs	111 (7)	110 (6)	101 (5)	100 (4)	011 (3)	010 (2)	001 (1)	000 (0)	
	1	1	0	0	1	1	0	0	Rule for cell i

Fig. 2. *RMT*s of rule 204

If any rule of a CA does not maintain *Property 1*, the CA can not have single cycle attractors. The following recursive algorithm scans the rule vector $\mathcal{R} =< \mathcal{R}_1, \cdots, \mathcal{R}_i, \cdots, \mathcal{R}_n >$ of a CA and explores all of its attractors.

Algorithm 1: **FindAttractors** (*i*, *State*)
Input: n (CA size), CA rule vector.
Output: set of attractors.
A). if $i = 1$, *i.e*, for the first rule {
  if RMT 0 is 0 then
    Set first two bits of *State* as 00 and call *FindAttractors (i+1, State)*
  if RMT 1 is 0 then
    Set first two bits of *State* as 01 and call *FindAttractors (i+1, State)*
  if RMT 2 is 1 then
    Set first two bits of *State* as 10 and call *FindAttractors (i+1, State)*
  if RMT 3 is 1 then
    Set first two bits of *State* as 11 and call *FindAttractors (i+1, State)* }
else if $(1 < i < n)$, *i.e*, for an intermediate rule {
B). Set $(i+1)^{th}$ bit of *State* as 0, and
  Compute k – decimal equivalent of $(i-1)$, i, and $(i+1)^{th}$ bit sequence of *State*
  Check whether the RMT k of the i^{th} rule can stick to the i^{th} bit of *State*.
C). If yes, call *FindAttractors (i+1, State)*
D). Set $(i+1)^{th}$ bit of *State* as 1
  Compute k and check whether RMT k of i^{th} rule can stick to i^{th} bit of *State*.
  If yes, call *FindAttractors (i+1, State)* }
else
E). Compute k= decimal equivalent of the $(n-1)^{th}$ bit, n^{th} bit of *State* and 0
  If RMT k of last rule is the last bit of *State*, output the *state* as an attractor.

The argument *State* contains partially constructed attractor states. To find the exact form of an attractor, we need to run $FindAttractors(1, State)$.

Example 1. Consider a 4-cell CA with rule vector $< 10, 69, 204, 68 >$ (*Table 1*). To find attractors, we call *FindAttractors(1,State)*, where *State* (- - - -) is empty. Initially, the algorithm finds that $i = 1$ and RMT 0 for the first rule is 0. Hence if a state starts with 00, the first bit will stick to 0. That is, the state may be an attractor. Therefore, the first two bits of 4-bit *State* is filled up with 00, and the recursive algorithm *FindAttractors* ($i = 2$ and *State* (0 0 - -)) is called.

First recursive call (i = 2 and State = 00- -): According to the algorithm, $(i+1)^{th}$ bit -that is, the third bit of *State* is set to 0 (*Algorithm 1 Step B*). Hence, $k = 0$ (000), the $(i-1)^{th}$ bit of the $State = 0$, i^{th} bit $= 0$, and $(i+1)^{th}$ bit $= 0$.

Table 1. *RMT*s of the *CA* < 10, 69, 204, 68 > cell rules

RMT	111 (7)	110 (6)	101 (5)	100 (4)	011 (3)	010 (2)	001 (1)	000 (0)	Rule
First cell	*d*	*d*	*d*	*d*	1	0	1	0	10
Second cell	0	1	0	0	0	1	0	1	69
Third cell	1	1	0	0	1	1	0	0	204
Fourth cell	*d*	1	*d*	0	*d*	1	*d*	0	68

RMT stands for rule min term. *d* denotes don't care bit. It can either be 0/1.

However, the i^{th} (second) bit of the *State* -that is, 0 and the *RMT* x (*RMT* 0) of i^{th} rule (69, *Table 1*) is (1) not the same. Then, execute Step D, and the second bit of *State* is set to 1. Now the *State* is 001-, and $k = 1$. The *RMT*k (1) of the second rule (69) is 0 -that is, the second bit value of *State*, and so *FindAttractors* is called with $i = 3$ and *State* (001-).

Second recursive call (i = 3 and State = 001-): Since $i < 4$, as like the 1^{st} recursive call, the next bit (fourth bit) of *State* is set to 0. Therefore, the final *State* is 0010, and $k = 2$ (010). The *RMT* 2 of third rule, however, is 1 -that is, the third bit value of the *State*. So, *FindAttractors* is called with $i = 4$ and *State* = 0010.

Third recursive call (i = 4 and State = 0010): Here it is checked whether the last bit of *State* can be justified by the last cell rule to recognize the *State* as an attractor. Now, $k = 4$ (100) and *RMT* 4 of the last rule can generate the fourth bit of the *State*. Therefore, the *State* (0010) is a single cycle attractor (2).

Now $i = 3$. The fourth bit of *State* is set to 1. Hence the *State* is 0011, and $k = 3$ (011). The *RMT* 3 of third rule is again 1 -that is, the third bit of the *State*. So, the algorithm is further called with $i = 4$ and *State* = 0011.

Fourth recursive call (i = 4 and State = 0011): As like the third recursive call, it checks whether the new last bit of *State* can be justified by the last cell rule. Here $k = 6$ (110) and the *RMT* 6 of last rule is 1. Hence, the *State* 0011 (3) is also an attractor.

If the execution of algorithm is continued, we can identify another two attractors - 1100 (12) and 1101 (13) (*Fig.1*).

To identify all the attractors of a *CA*, we may not repeat *FindAttractors*. An attractor *A* can be derived once *B* and *C* are identified as the attractors.

Theorem 1. *If* $B = b_1 b_2 \cdots b_{i-1} b_i b_{i+1} \cdots b_n$ *and* $C = c_1 c_2 \cdots c_{i-1} c_i c_{i+1} \cdots c_n$ *are two attractors of an* $n-$*cell CA and* $b_{i-1} = c_{i-1}$, $b_i = c_i$ & $b_{i+1} \bigoplus c_{i+1} = 1$, *then* $B' = b_1 b_2 \cdots b_{i-1} b_i c_{i+1} \cdots c_n$ *and* $C' = c_1 c_2 \cdots c_{i-1} c_i b_{i+1} \cdots b_n$ *are also the attractors of the CA [4].*

Example 2. Let $B = 010101$ and $C = 000111$ be the two attractors of a 6-cell *CA*. For the attractors *B* and *C*, the third and fourth bits are the same ($b_{i-1} = c_{i-1}$ and $b_i = c_i$). Whereas the fifth bits (b_{i+1} and c_{i+1}) are different. Therefore, the derived attractors are $B' = 010111$ and $C' = 000101$.

Corollary 1. *The derived attractors (B' and C') are same as the original (B and C) if $b_k = c_k$, $\forall k$, where $1 \leq k < i - 1$.*

Proof. If $b_k = c_k$, $\forall k$ where $1 \leq k < i-1$, then we can write $B' = b_1 b_2 \cdots b_{i-1} b_i c_{i+1} \cdots c_n = c_1 c_2 \cdots c_{i-2} b_{i-1} b_i c_{i+1} \cdots c_n = c_1 c_2 \cdots c_{i-2} c_{i-1} c_i c_{i+1} \cdots c_n = C$. Similarly, it can be shown that $C' = B$. Hence the proof.

Corollary 2. *An $n-$cell CA synthesized from 2 arbitrary attractors, can have maximum 2^{m+1} attractors, where $m = \lfloor \frac{n-1}{3} \rfloor$.*

Proof. Let us consider a CA is synthesized from the attractors B and C. For one set of $(i-1,\, i,\, i+1)$, maintaining *Theorem 1*, the number of attractors is doubled. That is, two pairs (2^2) of attractors are there. If there is another such set of $(i-1,\, i,\, i+1)$, each pair of attractors derives another pair of attractors. Hence, for two such sets of $(i-1,\, i,\, i+1)$, number of attractors is 2^{2+1}.

However, it is obvious that if $i = 2$ (*Theorem 1*), then $B = B'$ and $C = C'$. That is, no new attractor is derived. Therefore, excluding the left most bit, maximum number of possible $(i-1,\, i,\, i+1)$ set is $\lfloor \frac{n-1}{3} \rfloor$. Therefore, maximum number of possible attractors is 2^{m+1}, where $m = \lfloor \frac{n-1}{3} \rfloor$. Hence the proof.

2.2 Extraction of *PE*-Bits

A number of works [2,5] have been reported, where $MACA$ (multiple attractor cellular automata) is considered to classify the set of data. The CA of *Fig.1* can be employed to classify the patterns of two classes. Class I is represented by S_1 & S_2 with attractors 2 (0010) & 12 (1100). Whereas class II (S_3 & S_4) is represented by the attractors 13 (1101) & 3 (0011). To find the class of a pattern p (1010), the CA is to be run for same time steps considering p as the seed. Finally, the CA settles to an attractor A (2). Hence p (1010) belongs to the class of A - that is, class I.

In *Fig.1*, there are 4 attractors. We need 4 places to store the class information. The places can be identified by the least significant 2 bits of the attractors. That is, for an input pattern, if the CA settles to the attractor 12 (1100), then we have to search the place 00. These two least significant bit positions are the pseudo-exhaustive (PE) bits of the CA.

The scheme to identify the PE bits of a linear $MACA$ has been reported in [2]. However, extraction of PE-bit positions in nonlinear CA is yet to be addressed. We next report characterization of $MACA$ that guides identification of pseudo-exhaustive bits of the attractors of an $MACA$.

Theorem 2. *2^k attractors, derived from 2 attractors, can uniquely be identified by k bit positions.*

Proof. Consider, a CA is synthesized from two given attractors B and C. According to *Theorem 1*, for one set of $(i-1,\, i,\, i+1)$, the number of attractors is doubled. For these four attractors, $(i+1)^{th}$ and $(i-2)^{th}$ bits (or any bit from 1 to $(n-2)$) are unique. Therefore, the four attractors can be identified by these two bits. If there is another such set $(i-1, i, i+1)$, each of the attractor pairs derives another pair of attractors. An additional bit -that is, last bit of the set, is required to identify the attractors. Hence, for two such sets of $(i-1, i, i+1)$, total number of attractors are 2^{2+1} and 3 bits are sufficient to identify the attractors. Therefore, if there are $(k-1)$ sets of $(i-1, i, i+1)$, then we can find 2^k number of attractors that can be identified by k bits. Hence the proof.

Example 3. The 4 attractors (010101, 000111, 010111 & 000101) of *Example 2*, derived from 010101 and 000111, can uniquely be identified by the second and fifth bits (from left) of the attractors.

Theorem 2 states that if we can construct two attractors that can derive in total 2^k attractors (*Theorem 1*), and then if a CA is synthesized considering those 2 attractors, the CA will must have 2^k attractors with k PE-bits.

3 Synthesis of CA with Single Cycle Attractors

The theoretical framework reported in the earlier section enables synthesis of a CA $(\mathcal{R})$ for a given set of attractors. The following algorithm *SynMACA* reports the synthesis of such a CA.

Algorithm 2: **SynMACA**
Input: set of attractors.
Output: CA (rule vector).
For each of the attractors {
for i^{th} CA cell $(1 \leq i \leq n)$
S1: Set RMT k as the i^{th} bit of the attractor, where k is the decimal equivalent of the sequence of $(i-1)$, i, and $(i+1)$ bits of the attractor, assuming 0^{th} and $(n+1)^{th}$ bits are 0. }
S2: Set the unfilled RMTs such that those RMTs as a whole can not contribute to generate single cycle attractors. Output the CA rule vector.

The avoidance of RMTs that can generate single cycle attractors (S2, *Algorithm 2*), can be realized following the designed next algorithm. It synthesizes a CA that does not have any single cycle attractor.

Algorithm 3: **CAWithoutSingleCycle**
Input: n (CA size)
Output: CA
Step 1: Randomly synthesize an n-cell CA.
Step 2: If the CA is having no single cycle, goto *Step 6*.
Step 3: Select a CA cell arbitrarily as the victim.
Step 4: Identify the RMTs of victim cell rule that generates attractors.
Step 5: Replace the value of each such RMT by its complement.
Step 6: Report the final CA.

Example 4. This example illustrates the execution steps of *Algorithm 2*. Consider the three 4-bit single cycle attractors - 0010 (2), 0101 (5) and 1111 (15). The algorithm scans each attractor from left to right and sets the RMTs accordingly. Since the CA is a 3-neighborhood CA, a 3-bit window can be considered that slides from left to right, assuming the left of the leftmost bit and right the rightmost bit are 0. Hence while scanning attractor 2, it sets RMT 0 (000) of first rule as 0, RMT 1 (001) of 2^{nd} rule as 0, RMT 2 (010) of third rule as 1 and 0 to RMT 4 (100) of the 4^{th} rule. Similarly, the attractors 3 and 15 can be considered to fill up the RMTs of cell rules. However, a number of RMTs remain unfilled. The unfilled RMTs are set in such a way that those can not contribute to produce attractors. Here for simplicity, we set each unfilled RMT abc by b', where b' is the complement of b (underlined RMTs of *Table 2*). The CA synthesized from *Algorithm 3*, for a given set of attractors S, may have additional attractors called the spurious attractors that are not belong to S. For example, *Algorithm 3* outputs a $CA < 14, 145, 191, 65 >$ for the attractor set - {0010 (2), 1011 (11) and 1111 (15)}. The attractor set of the CA contains two spurious attractors– 1010 (10) and 0011 (3) (*Theorem 1*).

Table 2. Formation of $CA < 8, 181, 151, 69 >$

RMT	111 (7)	110 (6)	101 (5)	100 (4)	011 (3)	010 (2)	001 (1)	000 (0)	Rule
First cell	d	d	d	d	$\underline{1}$	0	$\underline{0}$	$\underline{0}$	8
Second cell	$\underline{1}$	0	1	1	0	$\underline{1}$	$\underline{0}$	1	181
Third cell	$\underline{1}$	0	$\underline{0}$	1	0	$\underline{1}$	1	1	151
Fourth cell	d	$\underline{1}$	d	$\underline{0}$	d	$\underline{1}$	d	1	69

The synthesis of an n-cell CA with k ($k \leq 1 + \lfloor \frac{n-1}{3} \rfloor$, *Corollary 2*) pseudo-exhaustive bits is described in the following algorithm. It exploits *Theorem 1* and constructs an n-bit attractor (A) randomly. Then a new one (say B), based on the A, is formed such that while synthesizing an MACA with the attractors A & B, a number (2^k-2) of spurious attractors are generated.

Algorithm 4: **SynMACAwithPE**
Input: n (length of $MACA$), k (number of PE-bits)
Output: non-linear $MACA$ (rule vector) with k pseudo-exhaustive bit positions
Step 1: Randomly synthesize an n-bit attractor A.
Step 2: Arbitrarily identify k bit positions on A such that an identified bit can have at least one identified bit either of its left or right at a distance not less than 3 bit.
Step 3: Synthesize an n-bit attractor B following the rules -

(i) Identified bits of A & B should be complement to each other.

(ii) If i^{th} and j^{th} bits are two consecutive identified bits and $|i - j| \geq 3$, $\forall i, j$, the bits starting from $(j + 1)^{th}$ to $(i - 1)^{th}$ positions of A & B are the same.

(iii) Randomly fill up the other bits of B such that the non-identified bits can not behave like an identified bit - that is, (i) and (ii) are denied.

Step 4: Synthesize CA that includes the attractors A and B (*Algorithm 3*).

Example 5. Let us consider synthesis of an $n = 4$ cell $MACA$ with $k = 2$ PE bits. Assume $A = 0101$ is randomly selected as an attractor. The 1^{st} and

4^{th} bits are identified as PE. The attractor $B = 1100$ is synthesized from A following *Step 3* of *Algorithm 4*. If we run *Algorithm 3*, a $CA < 9, 119, 3, 20 >$ is synthesized. It is having 4 single cycle attractors (0100, 0101, 1100, and 1101) that can be identified by the 1^{st} and 4^{th} bits (00, 01, 10, and 11) only.

The earlier discussion points to the fact that for a given attractor set [the PE-bits], we can synthesize an MACA following *Algorithm 3* [*Algorithm 4*]. However, the performance of *Algorithm 4* is limited by the number of PE-bits (k) expected.

4 Conclusion

This paper reports a detail characterization of single cycle attractors in the CA state space. Pseudo-exhaustive (PE) bits to identify the single cycle attractors are identified. A theoretical framework is proposed to synthesize a CA with specified PE bits and the attractor set.

References

1. Banks, E.R.: Information Processing and Transmission in Cellular Automata. PhD thesis, MIT (1971)
2. Pal Chaudhuri, P., Roy Chowdhury, D., Nandi, S., Chatterjee, S.: Additive Cellular Automata - Theory and Applications, vol. 1. IEEE Computer Society Press, California (1997)
3. Codd, E.F.: Cellular Automata. Academic Press Inc., London (1968)
4. Das, S., Das(Bit), S., Sikdar, B.K.: Non-linear Cellular Automata Based Design of Query Processor for Mobile Network. In: IEEE SMC conference, Hawaii, vol. 3, pp. 2751–2756 (October 2005)
5. Ganguly, N.: Cellular Automata Evolution: Theory and Applications in Pattern Recognition and Classification. PhD thesis, Bengal Engineering College (a Deemed University), India (2004)
6. Maji, P., Shaw, C., Ganguly, N., Sikdar, B.K., Pal Chaudhuri, P.: Theory and Application of Cellular Automata For Pattern Classification. Special issue of Fundamenta Informaticae on Cellular Automata 58, 321–354 (2003)
7. Martin, O., Odlyzko, A.M., Wolfram, S.: Algebraic Properties of Cellular Automata. Comm. Math. Phys. 93, 219–258 (1984)
8. Moore, E.F.: Machine models of self reproduction. In: Burks, A.W. (ed.) Essays on Cellular Automata, University of Illinois Press, Urbana (1970)
9. von Neumann, J.: The theory of self-reproducing Automata. In: Burks, A.W. (ed.), Univ. of Illinois Press, Urbana (1966)
10. Wolfram, S.: Statistical mechanics of cellular automata. Rev. Mod. Phys. 55(3), 601–644 (1983)
11. Wuensche, A., Lesser, M.J.: The Global Dynamics of Cellular Automata; An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata. Santa Fe Institute Studies in the Science of Complexity. Addison Wesley, Reading (1992)