Characterization of Single Cycle CA and its Application in Pattern Classification

Sukanta Das 1  Sukanya Mukherjee 2  Nazma Naskar 3

Department of Information Technology
Bengal Engineering and Science University, Shibpur
Howrah, West Bengal, India 711103

Biplab K. Sikdar 4

Department of Computer Science & Technology
Bengal Engineering and Science University, Shibpur
Howrah, West Bengal, India 711103

Abstract

The special class of irreversible cellular automaton (CA) with multiple attractors is of immense interest to the CA researchers. Characterization of such a CA is the necessity to devise CA based solutions for diverse applications. This work explores the essential properties of CA attractors towards characterization of the 1-dimensional cellular automata with point states (single length cycle attractors). The concept of Reachability Tree is introduced for such characterization. It enables identification of the pseudo-exhaustive bits (PE bits) of a CA defining its point states. A theoretical framework has been developed to devise schemes for synthesizing a single length cycle multiple attractor CA with the specific set of PE bits. It also results in a linear time solution while synthesizing a CA for the given set of attractors and its PE bits. The experimentation establishes that the proposed CA synthesis scheme is most effective in designing the efficient pattern classifiers for wide range of applications.

Keywords: Cellular automata, attractor, pseudo-exhaustive bit, reachability tree, pattern classifier.

1 Introduction

Introduction of Cellular Automaton (CA) is an important development in history to provide abstract model of concrete computers [20]. The concept of CA was initiated in the early 1950s by J. von Neumann and Stan Ulam [21]. Researchers had tried

1 Email: sukanta@it.becs.ac.in
2 Email: sukanyaiem@gmail.com
3 Email: nazma.preeti@yahoo.co.in
4 Email: biplab@cs.becs.ac.in
5 This research work is supported by the Sponsored Cellular Automata Research Projects, Bengal Engineering and Science University, Shibpur, India-711103.

1571-0661/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
to view simplified structure of CA amenable to characterization. The works that targeted structural simplification were [1,2,4,5].

In the early 1980s, Stephen Wolfram [26] studied in detail a family of simple 1-dimensional cellular automata that could simulate complex behaviors [15,22,23,24,25]. The CA structure was viewed as a discrete lattice of two-state per cell with 3-neighborhood dependency (self, left and right neighbors). This structure attracted a large section of researchers working in the diverse fields and a special class of 1-dimensional CA, called linear/additive CA, had gained the primary attention [3]. The theoretical framework developed in [3] targets characterization of non-uniform linear/additive CA.

While characterizing the CA state space, the researchers identified a set of CA states towards which neighboring states asymptotically approach in the course of dynamic evolution [27]. This set of states, referred to as the attractor of CA state space, forms a basin of attraction with its neighboring states. Such a CA with multiple attractors in its state space were of primary interest in applications like pattern recognition, pattern classification, design of associative memory, query processing etc. [3,11,12,13,14].

Characterization of a CA with multiple single length cycle attractors (point states) received special attention for cost effective solutions of real life applications. The issues related to identification of such attractors in linear/additive CA, and synthesis of single length cycle multiple attractor linear/additive CA were addressed in [3,12,14]. A graph based solution for such identification was also proposed [16,19]. However, characterization of single length cycle attractors as well as the synthesis of a CA with specified set of single length cycle attractors are yet to be explored.

In this context, we concentrate on the characterization of single length cycle attractors in a specific class of 1-dimensional nonlinear cellular automata. We explore the essential properties of CA attractors that enable such characterization. The introduction of Reachability tree provides the theoretical basis for identification of the attractors of a CA as well as its PE (pseudo-exhaustive) bits, defining the attractors. A theoretical framework has been developed that effectively been exploited to devise schemes for synthesizing a CA with the specific set of PE bits and having only single length cycle attractors. The proposed synthesis scheme is found to be effective while designing the CA based pattern classifier for standard applications.

The next section introduces the cellular automata preliminaries relevant for the current work. Section 3 introduces the concept of reachability tree and the theoretical basis of the proposed characterization of CA state space. A number of linear time algorithms/solutions, such as, computation of the number of attractors, identification of PE-bits, etc. are also reported in this section. Synthesis of a single length cycle multiple attractor CA with the specific set of PE bits is reported in Section 4. In Section 5, we report the design of a pattern classifier following the synthesis scheme devised in Section 4.
2 Preliminaries of Cellular Automata

A Cellular Automaton (CA) consists of a number of cells organized in the form of a lattice. It evolves in discrete space and time. Each cell of a CA stores a discrete variable at time $t$ that refers to the present state of the cell. The next state of the cell at $(t+1)$ is affected by its state and the states of its neighbors at time $t$. In the current work, we concentrate on the 3-neighborhood (self, left and right neighbors) CA, where a CA cell is having two states - 0 or 1. The next state of the $i^{th}$ cell of such a CA is

$$S_{i}^{t+1} = f_i(S_{i-1}^{t}, S_{i}^{t}, S_{i+1}^{t})$$

$f_i$ is the next state function; $S_{i-1}^{t}$, $S_{i}^{t}$ and $S_{i+1}^{t}$ are the present states of the left neighbor, self and right neighbor of the $i^{th}$ CA cell at time $t$.

The collection of states $S^{t}(S_{1}^{t}, S_{2}^{t}, \ldots, S_{n}^{t})$ of its cells at time $t$ is the present state of an $n$-cell CA and its next state is

$$S^{t+1} = (f_1(S_{0}^{t}, S_{1}^{t}, S_{2}^{t}), f_2(S_{1}^{t}, S_{2}^{t}, S_{3}^{t}), \ldots, f_n(S_{n-1}^{t}, S_{n}^{t}, S_{n+1}^{t}))$$

If $S_{0}^{t} = S_{n}^{t}$ and $S_{n+1}^{t} = S_{1}^{t}$, then the CA is a periodic boundary CA. On the other hand, if $S_{0}^{t} = S_{n+1}^{t} = 0$ (null), the CA is null boundary. Figure 1 shows the schematic diagram of a two-state 3-neighborhood null boundary CA. Each CA cell is implemented with a memory element and a combinational logic realizing the next state function ($f_i$). In the current work, we concentrate on null boundary CA.

The next state function (combinational logic) of $i^{th}$ CA cell can be expressed in the form of a truth table (Table 1). The decimal equivalent of the 8 outputs is called ‘Rule’ $R_i$ [22]. In a two-state 3-neighborhood CA, there can be a total of $2^8$ (256) rules. Three such rules 90, 150, and 75 are illustrated in Table 1. The first row of the table lists the possible $2^3$ (8) combinations of the present states of the $(i-1)^{th}$, $i^{th}$ and $(i+1)^{th}$ cells at time $t$. The last three rows indicate the next states of the $i^{th}$ cell at $(t+1)$ for different combinations of the present states of its neighbors, forming the rules 90, 150 and 75 respectively. Out of 256, 14 rules are called as linear/additive rules [3] that employs only XOR/XNOR logic.

Rule Min Term (RMT): From the view point of Switching Theory, a combination of the present states (as noted in the 1st row of Table 1) can be viewed as the Min Term of a 3-variable ($S_{i-1}^{t}, S_{i}^{t}, S_{i+1}^{t}$) switching function. Therefore, each column of the first row of Table 1 is referred to as Rule Min Term (RMT). The column 011 is the 3rd RMT. The next states corresponding to this RMT are 1 for Rule 90.
Table 1
Truth table for rule 90, 150 and 75

Present state : 111 110 101 100 011 010 001 000 Rule

(RMT)  (7) (6) (5) (4) (3) (2) (1) (0)

(i) Next State : 0 1 0 1 1 0 1 0 90
(ii) Next State : 1 0 0 1 0 1 1 0 150
(iii) Next State : 0 1 0 0 1 0 1 1 75

Note: RMT stands for Rule Min Term. The value 0/1 noted on 3rd/4th/5th row shows the output of the three variable switching function.

Fig. 2. State transitions of a reversible CA <105, 177, 170, 75>

and 75, and 0 for Rule 150. The characterization reported in this work is based on the analysis of RMTs of a CA rule.

Definition 2.1 The set of rules \( \mathcal{R} = < \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_n > \) that configures the cells of a CA is called the rule vector.

Definition 2.2 If \( \mathcal{R}_1 = \mathcal{R}_2 = \cdots = \mathcal{R}_n \), the CA is a uniform CA, otherwise it is non-uniform or hybrid CA.

Definition 2.3 If all the \( \mathcal{R}_i \)'s \( (i = 1, 2, \ldots, n) \) of a rule vector \( \mathcal{R} \) are linear/additive, the CA is referred to as Linear/Additive CA, otherwise the CA is a Nonlinear one.

The sequence of states generated (state transitions), during its evolution (with time), directs the CA behavior. The state transition diagram (Fig. 2 and Fig. 3) of a CA may contain cyclic and non-cyclic states (a state is called cyclic if it lies in a cycle; the states of Fig. 2) and based on this, the CA can be categorized either as reversible or irreversible CA.

In a reversible CA, each CA state repeats after certain number of time steps (Fig. 2). Therefore, all the states of a reversible CA are reachable from some other states, where each state has exactly one predecessor. On the other hand, in an irreversible CA (Fig. 3), there are some non-reachable states. Such states
are not reachable from any other state of the CA. Moreover, some states of the irreversible CA are having more than one predecessor [17,18]. The states 5 and 13 of Fig. 3 are the non-reachable states whereas 15 and 7 are having more than one predecessor. The non-reachable states of an irreversible CA form Garden of Eden. The cycles 7 \rightarrow 3 \rightarrow 11 \rightarrow 7 and 15 \rightarrow 15 of Fig. 3 are the attractors of CA <105,177,171,75>. The 15 is a single length cycle attractor (point state).

**Pseudo-Exhaustive (PE) bits:** A set of \( m \) bits can uniquely identifies \( 2^m \) attractors of an \( n \)-cell CA, where \( m \leq n \). These exhaustively appear in the set of \( 2^m \) attractors and called PE (Pseudo-Exhaustive) bits of the CA. In Fig. 4, there are four attractors – 2 (0010), 12 (1100), 13 (1101) and 3 (0011). The least significant two bits 10, 00, 01 and 11 of the attractors can uniquely identify those and called PE bits. The identification of PE bits of a CA reduces computation overhead as well as storage overhead while developing CA model for an application.

In this work, we concentrate only on the characterization of single length cycle attractor CA and its PE-bits. The following sections report such characterization.

### 3 Characterization of CA attractors

This section reports properties of CA attractors to explore the single length cycle attractors (point states) of an irreversible CA. The proposed characterization is
Fig. 6. RMTs of Rule 204

based on the analysis of *Reachability Tree*. The theoretical foundation thus evolved is then employed to identify the set of single cycle attractors in a *CA* as well as its *PE*-bits of a *CA*.

Since the next state of a single length cycle attractor is the attractor itself (attractor 15 of Fig. 3), there should be at least one *RMT* (Table 1) of each cell rule (*R*<sub>i</sub>) of the *CA* (*R*) for which the *CA* cell (*i*) does not change its state. For example, the *RMT* x0x (*x* = 0/1) of a rule (Fig. 5) is considered to find the next state of cell *i* when the current states of its left neighbor ((*i* − 1)<sup>th</sup> cell), self and right neighbor ((*i* + 1)<sup>th</sup> cell) are *x*, 0 and *x* respectively. It implies, if such an *RMT* is ‘0’, the state change of the cell (*i*) is 0 → 0 (Fig. 5). That is, for the rule *R*<sub>i</sub>, if the *RMT* 0 (000), 1 (001), 4 (100) or 5 (101) are 0, then the *CA* cell *i*, configured with *R*<sub>i</sub>, does not change its state. Similarly, if the *RMT*’s 2 (010), 3 (011), 6 (110) or 7 (111) are 1 in *R*<sub>i</sub>, the cell configured with *R*<sub>i</sub> can stick to its current state in the next time step. For rule 204 (Fig. 6), the *RMT*’s 0, 1, 4 & 5 are 0 and the *RMT*’s 2, 3, 6 & 7 are 1. It implies that if a *CA* cell is configured with rule 204, all *RMT*’s of the rule contribute towards formation of attractors of the *CA*.

**Property 1**: A rule *R*<sub>i</sub> can contribute to the formation of single length cycle attractor(s) if at least one of the *RMT*’s 0, 1, 4 or 5 is 0, or the *RMT*’s 2, 3, 6 or 7 is 1.

If any rule (*R*<sub>i</sub>) of the *CA* (*R*) does not obey **Property 1**, the *CA* can not have a single length cycle attractor. Therefore, examination of **Property 1** in the rules of *R*, configuring the cells, is a necessity for identification of single length cycle attractors (if any) of the *CA*.

### 3.1 Reachability tree characterizing attractors

Reachability Tree, we proposed in [7,8,9], is a binary tree that represents the reachable states of a *CA*. Each node of the tree is constructed with *RMT*’s of a rule (Section 2). The left edge of a node is referred to as the 0-edge and the right edge is as 1-edge (Fig. 7). The number of levels of the reachability tree for an *n*−cell *CA* is (*n* + 1). Root node is at Level 0 and the leaf nodes are at Level *n*. The nodes at Level *i* are constructed from the *RMT*’s of (*i* + 1)<sup>th</sup> *CA* cell rule *R*<sub>i+1</sub>.

The number of leaf nodes in a reachability tree denotes the number of reachable states of the *CA* and a sequence of edges from the root to a leaf node, representing an *n*−bit binary string, is the reachable state [8]. The 0-edge and 1-edge represent 0 and 1 respectively.

The *RMT*’s of two consecutive cell rules *R*<sub>i</sub> and *R*<sub>i+1</sub> are related while the *CA* changes its state. Since the *CA* is in 3-neighborhood, the *RMT*’s are of 3-bit. So,
Table 2
Relationship between RMTs of cell $i$ and cell ($i + 1$) for next state computation

<table>
<thead>
<tr>
<th>$i^{th}$ rule</th>
<th>$(i + 1)^{th}$ rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0, 1</td>
</tr>
<tr>
<td>1</td>
<td>2, 3</td>
</tr>
<tr>
<td>2</td>
<td>4, 5</td>
</tr>
<tr>
<td>3</td>
<td>6, 7</td>
</tr>
<tr>
<td>4</td>
<td>0, 1</td>
</tr>
<tr>
<td>5</td>
<td>2, 3</td>
</tr>
<tr>
<td>6</td>
<td>4, 5</td>
</tr>
<tr>
<td>7</td>
<td>6, 7</td>
</tr>
</tbody>
</table>

Fig. 7. Reachability Tree for the CA $<8, 112, 44, 68>$

a three bit window can be considered to get the next state of the CA [8]. If the window for $i^{th}$ cell is $(b_{i-1}b_ib_{i+1})$, $b_i = 0/1$, then the window for $(i+1)^{th}$ cell is either $(b_ib_{i+1}0)$ or $(b_ib_{i+1}1)$. In other words, if the $i^{th}$ CA cell changes its state following the RMT $k$ (decimal equivalent of $b_{i-1}b_ib_{i+1}$) of rule $R_i$, then the $(i+1)^{th}$ cell can generate the next state based on the RMT $2k \mod 8 (b_ib_{i+1}0)$ or $(2k + 1) \mod 8 (b_ib_{i+1}1)$ of rule $R_{i+1}$. This relationship between the RMTs of $R_i$ and $R_{i+1}$, while computing the next state of a CA, is shown in Table 2.

Figure 7 is the reachability tree of a CA $<8, 112, 44, 68>$. The RMTs of the CA rules are noted in Table 3. The decimal numbers within a node at level $i$ represent the RMTs of the CA cell rule $R_{i+1}$ based on which the cell $(i + 1)$ can change its state. The RMTs of a rule for which we follow 0-edge or 1-edge are noted.
Table 3

RMTs of the CA \(< 8, 112, 44, 68 >\) cell rules

<table>
<thead>
<tr>
<th>RMT</th>
<th>111</th>
<th>110</th>
<th>101</th>
<th>100</th>
<th>011</th>
<th>010</th>
<th>001</th>
<th>000</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(7)</td>
<td>(6)</td>
<td>(5)</td>
<td>(4)</td>
<td>(3)</td>
<td>(2)</td>
<td>(1)</td>
<td>(0)</td>
<td></td>
</tr>
</tbody>
</table>

First cell  

| d   | d   | d   | d   | 1   | 0   | 0   | 0   | 8    |

Second cell  

| 0   | 1   | 1   | 1   | 0   | 0   | 0   | 0   | 112  |

Third cell  

| 0   | 0   | 1   | 0   | 1   | 1   | 0   | 0   | 44   |

Fourth cell  

| d   | d   | 0   | d   | 1   | d   | 0   | 68  |

Fig. 8. Reachability Tree for attractors

in the bracket. For example, the root node (level 0) of Fig. 7 is constructed from RMTs 0, 1, 2 and 3 as cell 1 (rule 00001000) can change its state following any one of the RMTs 0, 1, 2 and 3. As the state of left neighbor of cell 1 is always 0, the RMTs 4, 5, 6 & 7 are the don’t cares for cell 1. It is obvious from Fig. 7 that there are 12 possible sequences of edges in the tree. That is 12, out of 16, CA states are reachable and the rests are non-reachable.

A reachability tree identifies all the reachable states including attractors of a CA. The tree of Fig. 7 can also be modified to display only the attractors. Since all the RMTs of a cell rule can not contribute to generate attractors, such (insignificant) RMTs are removed from the reachability tree. For example, RMT 2 (010) is 0 implies that it is insignificant (it can not contribute to generate attractors).

The tree shown in Fig. 8 corresponds to the CA \(< 8, 112, 44, 68 >\). It is derived from Fig. 7 to point to the attractors only. The RMTs that have potential to form the attractors are utilized to construct the nodes of Fig. 8. It shows 5 (O, P, Q, R and Y), out of 12, reachable states are the attractors.
3.2 The attractor set

The modified reachability tree, shown in the earlier subsection, can be utilized to characterize the attractor set of a CA. To find the number of attractors of a CA, we need to scan the CA (R) from left to right and virtually form a reachability tree. The number of leaves in the tree denotes the number of attractors of the CA. A weight is associated with each subtree (Fig. 8) representing its capability of generating attractors.

**Algorithm 1 CalNoOfAttractors**

**Input:** < R₁, ⋯, Rᵢ, ⋯, Rₙ > (n-cell CA).

**Output:** Number of attractors of the CA.

1. **Step 1:** If any rule does not hold Property 1, report the number of attractor as 0 and return.

2. **Step 2:** Let S₀ and S₁ are two sets of RMTs, capable of generating attractors, of the first cell rule, where RMTs of S₀ are 0 and S₁ are 1.
   - If S₀ ≠ φ, set weight, the capability to generate attractor, for left subtree as $2^{n-1}$.
   - If S₁ ≠ φ, set weight for the right subtree as $2^{n-1}$.

3. **Step 3:** For i = 2 to n
   - For each set of RMTs {
     - Determine RMTs, capable of generating attractors, for the next level nodes of the reachability tree considering Table 2 and Rᵢ.
     - Distribute these RMTs into the sets S₀' and S₁' based on the next state values as 0 and 1 respectively.
     - If S₀' ≠ φ, set weight for new left subsubtree as half of the weight of subtree in consideration.
     - If S₁' ≠ φ, set weight for new right subsubtree as half of the weight of subtree in consideration.
   }
   - If there are duplicate sets (subsubtrees), consider only one, and assign its weight as the sum of all duplicate sets.

4. **Step 4:** Sum up the weights, calculated, and report it as the number of attractors of the CA.

**Complexity:** The complexity of the algorithm depends on n and the number of sets of RMTs. Since the number of RMTs is 8, maximum possible number of sets of RMTs is also fixed (≤ 8). Hence the complexity is $O(n)$.

**Example 3.1** This example illustrates the steps of Algorithm 1. Let us consider that the CA < 8, 112, 44, 68 > (Table 3) is the input to Algorithm 1. Each rule of the CA maintains Property 1 (Step 1) and $S₀ = \{0, 1\}$ & $S₁ = \{3\}$ (Step 2). Since $S₀ ≠ φ$ and $S₁ ≠ φ$, both the subtrees may generate attractors. The maximum possible number of attractors indicated by a subtree is $2^{4-1} = 8$ (weight of the subtree). In Step 3, the next nodes of reachability tree for attractors are determined. The nodes are $\{0, 1\}$ and $\{6\}$. For the first node, that is, for the first set, $S₀' = \{0, 1\}$
and \( S'_1 = \phi \). Therefore, the right subsubtree can not point to an attractor. The left subsubtree points the existence of attractors and weight of this left subsubtree is \( \frac{8}{2} = 4 \). On the other hand, the node \( \{6\} \) can generate only its right subsubtree \( (S'_0 = \phi \text{ and } S'_1 = \{6\}) \). Hence the weight for its right subsubtree is \( \frac{8}{2} = 4 \).

The process is continued until the last rule of the CA is encountered. In the next level, two nodes are identified – \( \{0, 1, 2, 3\} \) and \( \{4\} \). The first node is having two children, but the node \( \{4\} \) is having one and the weights (2, 2 and 2) are calculated accordingly. After processing of the last rule, 5 subtrees each with single node (weight = 1) are constructed. However, the two nodes O and Y of Fig. 8 are the same as both are derived from RMT 0 of the last rule (rule 68). These are replaced by a single one assigning weight = 2. Finally, the sum of weights \( 2 + 1 + 1 + 1 = 5 \) defines the number of attractors (Step 4).

3.3 Identification of PE bits

This subsection reports scheme to identify the \( K \) bit positions from a set of \( n \)-bit attractors, where \( K \leq n \), that can uniquely identify all the attractors. These \( K \) bits may act as the pseudo-exhaustive bits of the CA. The proposed scheme explores the modified reachability tree of a CA, defined in Section 3.1. The following example illustrates the scheme.

**Example 3.2** Consider the root node of a reachability tree shown in Fig. 8. It has two sub-trees. Left sub-tree contains 4 attractors (O, P, Q and R), whereas the right sub-tree contains only one (Y). It is obvious from the figure that the attractors starting with 0 are the part of left sub-tree, and the attractor (Y) starting with 1 is a part of the right sub-tree. Therefore, first bit (MSB) of the \( n \)-bit CA state is an identification bit. Similarly, the nodes D, H and I distinguish among the attractors of left sub-tree of root. Hence, the least significant two bits are also the identification bits. Therefore, the 3 bits (first, third and fourth), out of 4, can identify all the attractors of the CA. It can be noted that the least significant two (third and fourth) bits appear exhaustively in the attractor set. However, the 3 identification bits do not appear exhaustively in all the (5) attractors. Therefore, these 3 identification bits are not the PE bits. That is, the identification bits can not necessarily be the PE bits for a given attractor set.

**Theorem 3.3** \( m \) number of \( n \)-bit attractors can be identified by \( K \) bit positions, where \( K \leq n \) \& \( m \leq 2^K \) and there exists \( r \) sets of PE bit positions that are the subset of \( K \) with cardinality \( p_1, p_2, \ldots, p_r \), where \( 2^{p_1} + 2^{p_2} + \cdots + 2^{p_r} = m \).

**Proof.** Consider the reachability tree of a CA for \( m \) attractors. The first node, starting from the root, having both the left and right children, splits the set of attractors into two subsets. The bit corresponding to that node is the identification bit and can exhaustively identify two subtrees (subsets). Now, for each subtree, we can find another identification bit that splits the subtree into two subsubtrees and also can exhaustively identify the subsubtrees. This process is continued until we reach the leaves. Hence, each attractor can be identified by a set of identification
bits, and the maximum number of bits required to uniquely identify all the \( m \) attractors is \( K \), where \( K \leq n \).

Now, a number of attractors can be grouped in such a way that a subset of identification bits appear exhaustively to identify all the attractors of the group (that is, subset). Hence, the subset of identification bits is the \( PE \) bits for that particular subset of attractors. Let us consider, the number of such subsets of attractors is \( r \). Therefore, \( 2^{p_1} + 2^{p_2} + \cdots + 2^{p_r} = m \), where \( p_i (\leq K) \) is the cardinality of such \( i^{th} \) subset. Hence the proof.\[Q.E.D.\]

For example, 5 attractors of Example 3.2 can be identified by 2 sets of \( PE \) bits – the first bit and third & fourth bits. Therefore, \( p_1 = 1 \) and \( p_2 = 2 \).

**Corollary 3.4** \( 2^K \) number of attractors of an \( n \)-cell CA can be identified by \( K \) bit positions, where \( K \leq n \).

**Proof.** The corollary directly follows from Theorem 3.3 if there is one set of \( PE \) bit positions, that is, \( r = 1 \).\[Q.E.D.\]

We next propose the following algorithm to find the \( K \) such identification bits of a CA. The algorithm implicitly constructs the reachability tree for attractors of the CA. If a node, having both the children, is found, the bit corresponding to that node is marked as an identification bit.

**Algorithm 2 FindIdentificationBits**

**Input:** \(< \mathcal{R}_1, \cdots, \mathcal{R}_i, \cdots, \mathcal{R}_n > \) (\( n \)-cell CA).

**Output:** Identification bits.

**Step 1:** If any rule does not hold Property 1, return.

**Step 2:** Suppose \( S_0 \) and \( S_1 \) are the two sets of RMT’s of first rule that can contribute to the formation of attractors, where RMT’s of \( S_0 \) are 0 and \( S_1 \) are 1.

If \( S_0 \neq \phi \neq S_1 \), mark the first bit.

**Step 3:** For \( i = 2 \) to \( n \)

For each set of RMT’s

Determine RMT’s that contribute to the formation of attractors for next level nodes of the reachability tree (for attractors) using Table 2 and \( \mathcal{R}_i \).

Distribute these RMT’s into \( S_0^i \) and \( S_1^i \) based on the next state values 0 and 1.

If \( S_0^i \neq \phi \neq S_1^i \), mark the \( i^{th} \) bit.

**Step 4:** Report the marked bits as identification bits.

**Complexity:** The complexity of the Algorithm 2 is dependent on \( n \) and the number of sets of RMT’s. Since the number of RMT’s is 8, the maximum possible number of sets of RMT’s is also fixed \(( \leq 8) \). Hence the complexity is \( O(n) \).

**Example 3.5** This example illustrates the execution steps of Algorithm 2. Consider the CA \(< 8,112,44,68 > \), noted in Table 3. Since all the rules maintain Property 1, single length cycle attractor(s) may exist for the CA (Step 1). Here, \( S_0 = \{0, 1\} \) and \( S_2 = \{3\} \). Hence the first bit (MSB) is marked as an identification
bit (Step 2). However, for $S_0$, $S'_1$ and for $S_1$, $S'_0$ are empty. Therefore, the second MSB can not be an identification bit (Step 3). Similarly, it can be found that third and fourth bits are the identification bits.

In the next subsection, Algorithm 2 is modified to find the pseudo-exhaustive bits of a CA, if any, to identify all the attractors.

### 3.4 CA Synthesis for specified PE-bits

This subsection proposes a synthesis scheme for multiple attractor CA, based on the theoretical framework reported in Section 3.3. The following algorithm describes the proposed synthesis scheme.

**Algorithm 3 GeneralizedMACASynwithPE**

**Input:** $n$ (CA size), $K$ (PE bits).

**Output:** CA (Rule vector).

**Step 1:** Randomly identify $K$ bits that is treated as the PE bits.

**Step 2:** Suppose $S_0$ and $S_1$ are the two sets of RMTs of first cell, where RMTs of $S_0$ are 0 and the RMTs of $S_1$ are 1 if the RMTs contribute to form attractors.

If the first bit is the identified bit, then randomly set RMTs such that $S_0 \neq \phi \neq S_1$.

Otherwise, set the RMTs so that $S_0 \neq \phi (S_1 \neq \phi)$ but $S_1 = \phi (S_0 = \phi)$.

**Step 3:** For $i = 2$ to $n$

For each set of RMTs

Determine RMTs for the next level nodes of reachability tree following Table 2.

Distribute these RMTs into $S'_0$ and $S'_1$, where RMTs of $S'_0$ are 0 and $S'_1$ are 1 if the RMTs are selected for generating the attractors.

If the $i$th bit is an identified bit, then randomly set RMTs such that $S'_0 \neq \phi \neq S'_1$.

Otherwise, set the RMTs such that $S'_0 \neq \phi (S'_1 \neq \phi)$ but $S'_1 = \phi (S'_0 = \phi)$.

**Step 4:** Set the unfilled RMTs, if any, for each cell rule so that no extra bit can be considered as PE bit.

**Step 5:** Report the CA with $K$ PE bits.

**Complexity:** Algorithm 3 uses a main loop in Step 3 that depends on $n$. The maximum number of sets is constant. That is, the execution time of Algorithm 2 is dependent only on $n$ and the number of sets of RMTs. Therefore, the complexity of the above algorithm is clearly $O(n)$.

Although Algorithm 3 targets synthesis of a CA having single length cycle attractors, the CA synthesized from Algorithm 3 may have also multi length cycle attractors. The scheme that ensures synthesis of a CA having only single length cycle attractors is reported next.
4 Synthesis of single cycle attractor CA

The earlier section reports characterization of the CA attractors and its PE bits. This section further characterizes the CA targeting synthesis of a CA having only single length cycle attractors with specified PE-bits. To facilitate the characterization, we next introduce the concept of RMT sequence (RS).

**Definition 4.1** The edge traversed in the reachability tree of an n-cell CA to reach a reachable state is derived from a sequence of RMTs \( <x_1x_2\cdots x_n> \). It is RMT sequence or RS for the reachable state.

For example, consider the 4-cell CA \(<8,112,44,68>\) of Table 3. The corresponding reachability tree is shown in Fig. 7. The RS \(<3640>\) derives the state 1100, where 3, 6, 4 and 0 are the RMTs corresponding to \( R_1(8), R_2(112), R_3(44), \) and \( R_4(68) \) respectively. If a state is reachable from more than one, say 3 states, then 3 RSs points to the reachable state. A non-reachable state, on the other hand, can not associate an RS.

The two RSs, associated with a reachable state & its next state, are related. To find the relationship, we divide the 8 RMTs into two sets - RMT0 and RMT1, where \( RMT_0 = \{0, 1, 4, 5\} \) and \( RMT_1 = \{2, 3, 6, 7\} \). The RMTs of RMT0 are 0 and RMTs of RMT1 are 1 while the RMTs contribute to form single length cycle attractors (Property 1 of Section 3). Suppose \(<x_1x_2\cdots x_n>\) is an RMT sequence for an n-cell CA states. That is, \( x_i \) is an RMT of \( R_i \). Now, consider RMT \( x_i \) does not follow Property 1 and \( x_i \in RMT_0 \). That is, next state for RMT \( x_i \) is 1. The RMT \( x_{i-1} \) & RMT \( x_{i+1} \) can be 0/1. Let us consider \(<y_1y_2\cdots y_n>\) be the RS for next state. Therefore, \( y_i \) can be 010, 011, 110 or 111 – that is, 2, 3, 6 or 7. Similarly, if RMT \( x_i \) follows Property 1, the possible \( y_i \) is 0, 1, 2 or 3. These are noted in Table 4. Now if \( x_i \in RMT_1 \), with similar logic we get Table 5. Therefore, utilizing Table 4 and Table 5, the next RS \((RS_{t+1})\) of a given RS \((RS_t)\) can be determined.

**Definition 4.2** Two RMTs are **equivalent** if both result in the same set of RMTs for the next level of Reachability Tree.

For example, the RMTs 0 and 4 are equivalent as both result in the same set of effective RMTs \( 000=0, 001=1 \) (Table 2) for the next level of Reachability Tree. Similarly, the RMTs 1 & 5, 2 & 6, and 3 & 7 are equivalent.

### 4.1 Multi length cycle

The motivation of this section is to design a CA having only single length cycle attractors. The following theorem identifies the causes of the multi length cycle formation.

**Theorem 4.3** A set of states of an n-cell CA belong to a cycle of length \( l \), where \( l \geq 2 \), if the RMTs \( r_1, r_2 \) of \( R_i \) do not follow Property 1 and \( r_1 \in RMT_0 \) & \( r_2 \in RMT_1 \), then either \( r'_1 \in RMT_0 \) & \( r'_2 \in RMT_1 \), or \( r'_1, r'_2 \in RMT_0/RMT_1 \), where \( r'_1, r'_2 \) are RMTs of \( R_{i+1} \) and \( r'_1 (r'_2) \) is derived from \( r_1 (r_2) \).
The following example illustrates how a CA forms multi length cycle (of length three) during its state transition.

**Example 4.4** Let us consider a 4-cell $CA < 5, 73, 200, 80 >$ of Fig. 9. The $RMT$s of the CA cell rules are noted in Table 6. For the first cell ($R_1 = 5$), $RMT$s 0 and 3 do not maintain Property 1 and are from different sets ($RMT_0$ & $RMT_1$). Among
their successive $RMT$s for the next level, $RMT$s 0 and 7 are taken into consideration to form multi length cycle. The $RMT$s 0 and 7, belong to the different sets, do not follow Property 1, whereas $RMT$s 1 and 6 follow Property 1. For $R_2$ (73), $RMT$s 0, 7 and 2 do not follow Property 1, where $RMT$ 7, 2 $\in$ $RMT_1$ and $RMT$ 0 $\notin$ $RMT_0$. Now, if $RMT$s 0 and 7 are taken, their successive $RMT$s follow Property 1 but these are from the different sets ($RMT_0$ & $RMT_1$). It violets the rule to form a multi length cycle. When $RMT$s 0 and 2 are considered, their successive $RMT$s follow Property 1 which are from the same set ($RMT$ 0, 2 $\in$ $RMT_0$) and contribute to the formation of multi length cycle. Therefore, $RMT$ 1 is considered for $R_1$. $RMT$s 4 & 0 of $R_3$ and $RMT$ 0 of $R_4$ follow Property 1, which are the successive $RMT$s at second and third level respectively of the $RMT$s 2, 6 and 0. Now if we consider $RMT$ sequence $RS_1 < 3640 >$, then other than $RMT$ 3, all follow Property 1. When $RS_2$ is computed, except $RMT$ 2 at second position, all follow Property 1 where the other $RMT$s of the sequence are 1, 4 and 0 at first, third and fourth positions respectively. The $RS_3$ deduces $< 0000 >$, where $RMT$s at first and second positions do not hold Property 1 but at third and fourth positions Property 1 is followed which repeat $RS_1$. Thus cycle $[3640(0010) \rightarrow 1240(0000) \rightarrow 0000(1100) \rightarrow 3640(0010)]$ is formed of length three.

The next subsection provides the theoretical framework for designing a CA with only single length cycles.
4.2 Single length cycle

The following theorems guide to identify the CA having only single length cycle attractors with specified PE-bits.

**Theorem 4.5** For an \( n \)-cell CA, if \( i^{th} \) bit is the PE-bit, then RMTs 0, 1, 2, 3 or RMTs 4, 5, 6, 7 or all eight RMTs of \( R_i \) follow Property 1 depending on the RMTs, that follow Property 1, of \( R_{i-1} \).

**Proof.** Let us consider an \( n \)-cell CA. If the first bit is the PE-bit, then 0-edge and 1-edge of the of the reachability tree at level 0 must have the RMTs following Property 1. Therefore, RMTs 0, 1, 2, 3 follow Property 1 at \( R_1 \) when first and second bits are the PE-bits. If the second bit is not a PE-bit, then at \( R_1 \) RMTs 0 or 1 or both have to follow Property 1 and that is same for RMTs 2, 3 at a time.

For cell \( i \), if \( i^{th} \) bit is the PE-bit, then the RMTs of \( R_i \) are selected in such a way that it must follow Property 1 to represent the significance as PE-bit, RMTs of \( R_{i-1} \) are considered. When at \( i^{th} \) level, RMTs 0, 1, 2, 3 follow Property 1 then Property 1 is followed by RMTs 0, 1 or 4, 5 or 0, 1, 4, 5 at level \( (i-1) \). In the same way, RMTs 4, 5, 6, 7 are computed at rule \( R_i \). RMTs 0, 1, 2, 3 all follow Property 1 to avoid the multi length cycle formation. The same is done for RMTs 4, 5, 6, 7. For the RMTs, which do not hold Property 1 and are from the same set at level \( (i-1) \), the successive RMTs must select (0/1) randomly. Hence the proof. \( \Box \)

**Theorem 4.6** For an \( n \)-cell CA, if \( i^{th} \) bit is not the PE-bit, then \( R_i \) is constructed in such a manner that

(i) when RMTs 0,1,4,5 follow Property 1, the RMTs 2,3,6,7 do not follow Property 1 or vice versa depending on which RMTs are selected to maintain Property 1 at \( R_{i-1} \), where \( (i+1)^{th} \) bit is the PE-bit

(ii) only two equivalent RMTs follow Property 1 and other six RMTs do not follow depending on which RMTs follow Property 1 at \( R_{i-1} \), where \( (i+1)^{th} \) bit is not the PE-bit.

**Proof.** Let us consider an \( n \)-cell CA. If the first bit is not the PE-bit, then either RMTs 0, 1 or RMTs 2, 3 follow Property 1, where the next bit is the PE-bit. As when the next bit is the PE-bit then either the RMT 0, 1, 2, 3 or the RMT 4, 5, 6, 7 are to be followed to restrict multi cycle formation. Therefore, the first rule must follow RMT either 0, 1 or 2, 3. If the second bit is not the PE-bit, only one RMT among 0, 1, 2, 3 follows Property 1 at the first rule.

For cell \( i \), if the \( i^{th} \) bit is not the PE-bit then either RMTs 0, 1, 4, 5 or RMTs 2, 3, 6, 7 follow Property 1 when \( (i+1)^{th} \) bit is the PE-bit. The rule \( R_i \) is constructed in such a way that the RMTs from the same set either follow Property 1 or do not follow to prevent multi length cycle formation depending on the RMTs. If the \( (i+1)^{th} \) bit is not the PE-bit, then at \( R_i \), the RMTs are constructed in such a manner that only two equivalent RMTs follow Property 1. The \( R_i \) is constructed in such a manner that the RMTs from the same set do not follow Property 1 while for other sets, two equivalent RMTs only follow Property 1 to prevent multi length cycle formation depending on the RMTs satisfying Property 1 at \( R_{i-1} \). Hence the
proof.

The formal algorithm to synthesize a CA having only single length cycle attractors with specified PE bits is presented in the next subsection.

4.3 Synthesis of CA with single length cycles

The following algorithm takes CA size \( n \) and number of PE bits \( K \) as input, and outputs an \( n \)-cell CA (rule vector) that contains only single length cycle attractors during its state transitions.

**Algorithm 4 SingleCycleCAwithPE**

**Input:** \( n \) (CA size), \( K \) (PE bits).

**Output:** CA \( (R = < R_1, R_2, \ldots, R_n >) \).

Step 1: Randomly identify \( K \) bits as the PE bits.

Step 2: For each cell, the RMTs are distributed into the following sets \( RMT_0 = \{0, 1, 4, 5\} \) and \( RMT_1 = \{2, 3, 6, 7\} \).

Step 3: (a) If first and second bits are PE-bits:

- Set \( R_1 \) & \( R_2 \) in such a way that all RMTs of both rules follow Property 1.

- (b) If first bit is PE bit, but second is not:

  - RMTs, randomly selected from \( RMT_0 \) & \( RMT_1 \), of \( R_1 \) are set to follow Property 1. The RMTs of \( R_2 \), derived from the selected RMTs of \( R_1 \) using Table 2, are also set to follow Property 1. If other RMTs of \( R_1 \) are set to disobey Property 1, the RMTs of \( R_2 \) are set to disobey Property 1 when the RMTs of \( R_1 \) & \( R_2 \) are from the same set (\( RMT_0 \) & \( RMT_1 \)), and obey Property 1 when from different sets.

- (c) If second bit is PE-bit (while first bit is not a PE-bit):

  - RMTs of \( R_1 \) from either \( RMT_0 \) or \( RMT_1 \) only hold Property 1. For \( R_2 \), the RMTs which follow Property 1 for \( R_1 \), their successive RMTs follow Property 1 and other RMTs selected randomly.

  Only one RMT from \( RMT_0 \) follows Property 1 and \( RMT_1 \) does not follow (or vice versa) for \( R_1 \) and their successive RMTs do follow Property 1 for the second rule. The RMTs which do not follow Property 1 and are from different sets for the first cell, their successive RMTs are selected in such a manner that RMTs from the same set do not follow Property 1. Property 1 is followed when taken from different sets.

Step 4: For \( i = 3 \) to \( n \)

- (a) If \( i^{th} \) and \( (i+1)^{th} \) bits are the PE-bit,

  then RMTs 0,1,2,3 or 4,5,6,7 or eight RMTs follow Property 1 at \( R_i \).

- Otherwise, RMTs 0,1,2,3 or 4,5,6,7 have to follow Property 1. Others are taken as 0/1.

- (b) If \( i^{th} \) bit is not the PE-bit but \( (i+1)^{th} \) bit is the PE-bit,

  then for \( R_i \), when RMTs from \( RMT_0 \) follow Property 1, \( RMT_1 \) does not follow (or vice versa).

- (c) Otherwise, all the RMTs from \( RMT_0 \) do not follow Property 1 and only two equivalent RMTs from \( RMT_1 \) follow Property 1 (or vice versa).

Step 5: Report the CA with \( k \) PE bits.
Complexity: Algorithm 4 executes a loop in Step 3 that depends on \( n \) (CA size). The complexity of the above algorithm, therefore, is \( O(n) \).

Example 4.7 This example illustrates the execution steps of Algorithm 4. Let us consider a 6-cell CA where the number of \( PE \)-bits is three and these are selected randomly at bit positions 2, 5 and 6 (Fig. 10). As the first bit is not the \( PE \)-bit but the second bit is the \( PE \)-bit, the \( RMT \)s of \( RMT_0 \) (or \( RMT_1 \)) follow Property 1, and so \( RMT \)s of \( RMT_1 \) (\( RMT_0 \)) can’t. Here \( RMT_0 \) follows Property 1 at cell 1. The successive \( RMT \)s also follow Property 1. For \( R_2 \), the rest of the \( RMT \)s are set randomly as \( RMT_1 \) does not follow Property 1, so there is no possibility to form multi length cycle (Theorem 4.3).

Both the 3\(^{rd} \) and 4\(^{th} \) bit positions are not the \( PE \)-bit. The third rule is constructed in such a way that the members of \( RMT_1 \) do not hold Property 1 and the equivalent \( RMT \)s 0 and 4 follow Property 1 while 1 and 5 can not. Now, the 4\(^{th} \) bit is not the \( PE \)-bit but the next is the \( PE \)-bit. So, the fourth cell is represented in a manner such that \( RMT \) 0 and 4 follow Property 1 at second level, \( RMT_0 \) have to follow Property 1 for \( R_4 \) and \( RMT_1 \) does not hold Property 1 to restrict multi length cycle. As the 5\(^{th} \) bit and 6\(^{th} \) bit are the \( PE \)-bits, \( R_5 \) is constructed in such a way that the \( RMT \)s 0, 1, 2, 3 follow Property 1 and others are selected randomly (either 0 or 1) to restrict multi length cycle formation. Then the next rule follows Property 1 at \( RMT \)s 0, 2, 4, 6. Hence the 6-cell CA is \(< 0, 76, 34, 0, 220, 68 >\).

The CA structure, synthesized in this section, can effectively be utilized for designing a pattern classifier. Next section reports such a design.

5 Design of CA based classifier

An \( n \)-cell CA with \( k \) point states can be viewed as \( k \) class natural classifier. For example, the CA of Fig. 4 can act as a 4 class classifier where each attractor basin \((S_1/S_2/S_3/S_4)\) represents a class.
In this work, we target the design of a 2-class classifier. Suppose the patterns of $S_1$, $S_2$ & $S_3$ belong to class 1 and the patterns of $S_4$ are from class 2 (Fig. 4). Then the $CA$ of Figure 4 can also act as the 2-class classifier, where attractors 2, 12 and 13 identify class 1 and the attractor 3 corresponds to class 2.

Algorithm 4 that synthesizes $CA$ having only single length cycles and a specified set of $PE$ bits, can be utilized to design an $n$-bit classifier. The primary metric for evaluating classifier performance is the classification efficiency. It is measured as

$$\text{efficiency} = \frac{\text{Number of patterns properly classified}}{\text{Total number of patterns}} \times 100\%$$

In the proposed design, we generate 100 such $CA$ using Algorithm 4 and then compute their classification efficiency. The $CA$ with highest efficiency is considered as the desired classifier. One major advantage of the classifier, designed out of Algorithm 4, is – it reduces the memory overhead for storing the classifier. While storing $n$-bit attractors to identify a class, only $PE$ bits of the attractors are to be considered.

5.1 Experimental setup

The performance analysis of the pattern classifier, based on nonlinear single length cycle $CA$, is evaluated on the basis of datasets available in http://www.ics.uci.edu/~mlearn/MLRepository.html, summarized in Table 7. All the datasets taken into consideration have two classes. While columns I and II of Table 7 represent the dataset and its domain, the columns III and IV depict the number of categorical and continuous attributes in the dataset. Column V and column VI represent the number of examples (tuples) of the dataset and the experimental set up respectively.

To handle such real data, the dataset is suitably modified to fit the input characteristics of the proposed pattern classifier. Each categorical attribute is converted into binary form as per the Thermometer Code [10]. For continuous-valued attribute, it is transformed into a categorical attribute by calculating the Mean and Standard deviations for all instances of an attribute.

For large datasets, a test set is used to estimate the classification accuracy. The classifier is constructed considering the patterns in the training set and next its performance is evaluated based on the test set. For small datasets m-fold cross validation process is needed where the total dataset is divided into $m$ subsets each containing approximately same number of records. For each subset, a classifier is constructed from the remaining $(m-1)$ subsets.

5.2 Performance analysis

To design an $n$-cell classifier, the cell rules are generated using Algorithm 4. For the experimentation of a dataset, a number of $CA$ are synthesized and their performance as classifier are evaluated. The $CA$ with the highest performance is treated as the desired classifier. The performance of proposed classifier is compared to that of existing classification schemes [14]. Table 8 reports the comparison results. Column
### Table 7
Description of Datasets and Experimental Setup

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Domain</th>
<th>No of Attributes</th>
<th>No. of example</th>
<th>Experimental setup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Cate</td>
<td>Conti</td>
<td></td>
</tr>
<tr>
<td>monk1</td>
<td>Monk’s Problem</td>
<td>6</td>
<td>0</td>
<td>556</td>
</tr>
<tr>
<td>monk2</td>
<td>Monk’s Problem</td>
<td>6</td>
<td>0</td>
<td>601</td>
</tr>
<tr>
<td>monk3</td>
<td>Monk’s Problem</td>
<td>6</td>
<td>0</td>
<td>554</td>
</tr>
<tr>
<td>vote</td>
<td>Voting Records</td>
<td>32</td>
<td>0</td>
<td>435</td>
</tr>
<tr>
<td>Spect</td>
<td>Heart Disease</td>
<td>22</td>
<td>0</td>
<td>267</td>
</tr>
<tr>
<td>Pima</td>
<td>Diabetis Disease</td>
<td>8</td>
<td>0</td>
<td>768</td>
</tr>
<tr>
<td>Haber -man</td>
<td>Survival Records</td>
<td>3</td>
<td>0</td>
<td>306</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>Endgame Records</td>
<td>9</td>
<td>0</td>
<td>958</td>
</tr>
</tbody>
</table>

I shows the name of dataset while Column II depicts the name of the scheme. The efficiencies of known algorithms are noted in Column III. The efficiency of our design is reported in the last column.

It can be observed from Table 8 that the reported classifier is as efficient as the existing designs. Moreover, the proposed classifier reduces the memory overhead significantly. During the design, we set the maximum number of $PE$ bits for each dataset as the 10% of total number of bits. Therefore, the classifier saves 90% of memory by storing only the $PE$ bits.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Algorithm</th>
<th>Efficiency (in %)</th>
<th>Efficiency (in %) of proposed scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>monk1</td>
<td>Bayesian</td>
<td>99.9</td>
<td>91.93</td>
</tr>
<tr>
<td></td>
<td>C4.5</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCC</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MTSC</td>
<td>98.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>monk2</td>
<td>Bayesian</td>
<td>69.4</td>
<td>75.73</td>
</tr>
<tr>
<td></td>
<td>C4.5</td>
<td>66.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCC</td>
<td>78.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MTSC</td>
<td>77.32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>75.16</td>
<td></td>
</tr>
<tr>
<td>monk3</td>
<td>Bayesian</td>
<td>92.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C4.5</td>
<td>96.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCC</td>
<td>76.58</td>
<td>95.08</td>
</tr>
<tr>
<td></td>
<td>MTSC</td>
<td>97.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>98.10</td>
<td></td>
</tr>
<tr>
<td>Haberman</td>
<td></td>
<td></td>
<td>73.49</td>
</tr>
<tr>
<td>Pima-indian</td>
<td></td>
<td></td>
<td>81.54</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>Sparse grid</td>
<td>98.33</td>
<td>82.63</td>
</tr>
<tr>
<td></td>
<td>ASVM</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSVM</td>
<td>93.33</td>
<td></td>
</tr>
<tr>
<td>Vote</td>
<td>Bayesian</td>
<td>92.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C4.5</td>
<td>94.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCC</td>
<td>95.88</td>
<td>97.0</td>
</tr>
<tr>
<td></td>
<td>MTSC</td>
<td>95.91</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MLP</td>
<td>90.87</td>
<td></td>
</tr>
<tr>
<td>Spect Heart</td>
<td></td>
<td></td>
<td>91.97</td>
</tr>
</tbody>
</table>
6 Conclusion

This paper reports a detail characterization of single length cycle attractors in CA state space. Pseudo-exhaustive (PE) bits to identify the single length cycle attractors of a CA are identified. A theoretical framework has been proposed to synthesize a CA with the specified PE bits for a given set of attractors. The synthesized CA is effectively utilized to design efficient pattern classifier.

References


