Modeling of asynchronous cellular automata with fixed-point attractors for pattern classification

Biswanath Sethi*    Sukanta Das
Department of Information Technology
Bengal Engineering and Science University, Shibpur, Howrah, West Bengal, India-711103.
Email: sethi.biswanath@gmail.com, sukanta@it.becs.ac.in
* Corresponding author

Abstract—This paper addresses a detail characterization of the one-dimensional two-state 3-neighborhood asynchronous cellular automata (ACA) having multiple fixed-point attractors with the target to model this class of CA for designing efficient pattern classifier. The cells of ACA are independent, and they are updated independently. When an ACA cell is updated, it reads the states of its neighbors and then updates its state following the state transition function. The ACA rules are characterized considering an arbitrary cell is updated in each time step. A theorem is designed for the identification of ACA with only fixed-point attractors. From this theorem we get 146 out of 256 ACA in two-state 3-neighborhood interconnection which always approach towards fixed-Point attractors. To identify individual fixed-point attractors, a directed graph, namely Fixed-point graph (FPG) is proposed. The FPG guides us to point out the ACA having multiple fixed-points attractors. There are 83 such ACA (out of 146 ACA with only fixed-point attractors) that are utilized to design efficient pattern classifier. Finally, the proposed classifier is tested with real-life data sets, and it is observed that the performance of the proposed ACA based classifier is always better than that of traditional CA based classifier and is also better than many other well-known pattern classifiers.

Index Terms—Cellular automata model, asynchronous cellular automata (ACA), fixed-point attractor, Fixed-point graph (FPG), pattern classifier.

I. INTRODUCTION

With the increase of computing power, the researchers have been showing their interest on the study of global behavior of cellular automata (CA). The CA have become popular tool to simulate various real-world systems. CA were originally introduced as an abstract mathematical model capable of self-reproduction [12]. They are defined by a lattice of cells and a local rule by which state of a cell is determined as a function of the state of its neighbors. In early 1980, the CA structure was simplified, and a two-state, 3-neighborhood CA structure was proposed on an one-dimensional lattice [13]. The researchers working with VLSI design and test have got attracted to this type of CA due to their simplicity and ease of hardware implementations [1]. This contribution also concentrates on such one-dimensional two-state 3-neighborhood CA.

While characterizing the CA, a set of CA states have been identified, toward which neighboring states asymptotically approach in the course of dynamic evolution, called attractors [1]. The attractors can be of length 1 (fixed-point attractor) or of length greater than 1 (multi-state attractor). The CA with multiple fixed-point attractors have been targeted to design effective solutions for pattern classification [1]. The characterization of CA for pattern classification has received special attention for cost effective solutions of real life applications. The CA, contributing with this context, are addressed in [2], [3], [5]–[8].

However, all such contributions have dealt with synchronous CA, where all the CA cells are forced to be updated simultaneously. On the other hand, in asynchronous CA (ACA) the cells are independent, and as a result, the cells are free to get updated independently [4], [9]. This independence property of ACA cells may affect the performance of ACA while they are utilized to address some real life problems. The ACA, however, have rarely been considered to solve pattern classification problem. Two dimensional ACA are studied in [10], [11] to utilize them for pattern classification. To our knowledge, no target has been taken to study one-dimensional ACA as pattern classifier.

In this scenario, we concentrate on characterization of ACA in one-dimensional two-state 3-neighborhood interconnection, with the target to design an efficient model for pattern classification. The ACA with multiple fixed-point attractors are utilized for pattern classification. We refer such ACA as multiple attractor ACA (MAACA). Here, we design 2-class classifier. We explore the essential properties of attractors that enable us for such characterization. A theorem has been reported for the identification of ACA with fixed-point attractors. One of the major contributions of this work is, it introduces the idea of Fixed-point graph to identify the fixed-point attractors of ACA. An algorithm is also developed to find the fixed-point attractors from the graph. Finally, the pattern classifier is designed and tested with real-life data sets. It is shown that performance of ACA based classifier is much better than that of traditional (that is, synchronous) CA based classifier. Moreover, the proposed classifier is very much competitive with other well-known classifier, even better than them in many cases.

The next section introduces the preliminaries of ACA and some definitions which are relevant for the current work. Section III reports the characterization of ACA. A theorem and algorithms for characterization are addressed in this section.

978-1-4799-0838-7/13/$31.00 ©2013 IEEE 311
The design of pattern classifier are devised in section IV. The algorithm for the design of pattern classifier is also reported. The performance of the proposed scheme with other are also reported in this section.

II. PRELIMINARIES OF ACA

Cellular automaton (CA) is a discrete dynamical system which evolves in discrete time and space. A CA consists of a lattice of cells, each of which stores a boolean variable at time $t$, that refers to the present state of the CA cell [13]. The next state of a cell of one-dimensional two-state 3-neighborhood CA is determined by the present states of left, self and right neighbors. Hence,

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

where $f$ 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 $t$ respectively. A collection of (local) states $S^t(S_1^t, S_2^t, \ldots, S_n^t)$ of cells at time $t$ is referred as a configuration or a (global) state of the CA at $t$. The function $f : \{0,1\}^3 \mapsto \{0,1\}$ can be expressed as a look-up table (see TABLE I). The decimal equivalent of the 8 outputs is called ‘rule’ [13]. There are $2^8 = 256$ CA rules in two-state 3-neighborhood dependency. Three such rules (36, 77 and 219) are shown in TABLE I. From the view point of Switching Theory, a combination of the present states can be considered as the Min Term of a 3-variable $(S_{i-1}^t, S_i^t, S_{i+1}^t)$ switching function. So, each column of the first row of TABLE I is referred to as Rule Min Term (RMT). The column (011) is the RMT 3. The next states corresponding to this RMT are 0 for rule 36 and, 1 for rule 77, and 1 for rule 219 (TABLE I). When the left most and right most cells of an $n$-cell CA are the neighbors of each other (that is, $S_0^t = S_n^t$ and $S_{n+1}^t = S_1^t$), the CA is called periodic boundary CA.

Traditionally, the cells of a CA are updated simultaneously. In asynchronous update, on the other hand, the cells are considered as independent and so, updated independently. We have considered here that a single arbitrary cell is updated in each time step. Such CA are referred as asynchronous cellular automata (ACA). This work deals with one-dimensional two-state 3-neighborhood ACA under periodic boundary condition.

The next state of a synchronous CA is determined by the rule only. However, in ACA, the next state does not only depend on the rule, but also on the cell which is being updated randomly. Fig. 1 shows state transition diagram of 4-cell rule 219 ACA in periodic boundary condition. The cells, updated randomly during state transition, are noted over arrows.

An ACA state can be viewed as a sequence of RMTs. For example, the state 1110 in periodic boundary condition can be viewed as $(3765)$, where 3, 7, 6 and 5 are corresponding RMTs on which the transitions of first, second, third and fourth cell can be made. To get a sequence of RMTs for a state, we consider an imaginary 3-bit window that slides over the state. The window contains a 3-bit binary value which is equivalent to RMT. To get the $i^{th}$ RMT, the window is loaded with $(i - 1)^{th}$, $i^{th}$ and $(i + 1)^{th}$ bits of the state. The window slides one bit right to report the $(i + 1)^{th}$ RMT. Now the current content of the window is $i^{th}$, $(i + 1)^{th}$ and $(i + 2)^{th}$ bit of the state. In the sequence of RMTs, however, two consecutive RMTs are related. If 5 (101) is the $i^{th}$ RMT in some sequence, then $(i + 1)^{th}$ RMT is either 2 (010) or 3 (011). Similarly, if 0 (000) or 4 (100) is the $i^{th}$ RMT, then 0 (000) or 1 (001) is the $(i + 1)^{th}$ RMT. The relations of two consecutive RMTs in a sequence of RMTs are noted in TABLE II. These relations play an important role in the design of pattern classifier, reported in this paper.

Definition 1: An RMT $r$ of a rule is active if an ACA cell flips its state (1 to 0 or 0 to 1) on $r$. Otherwise, the RMT $r$ is passive.

For example, RMT 1 (001) of rule 219 (see TABLE I) is active. Because, while a cell is acting on that particular RMT, then the cell’s present state is 0 and next state for the rule is 1. Hence, transition occurs. So, it can be said that if the middle bit of an RMT is unequal with the RMT value, the RMT is active. RMT 7 (111) of rule 219, on the other hand, is passive.

Definition 2: A fixed-point attractor is an ACA state, next state of which is the state itself for any style of update of cells. That is, if an ACA reaches to a fixed-point attractor, the ACA remains in that particular state forever.

In Fig. 1, the states 1101, 0111, 1111, 1011 and 1110 are fixed-point attractors for rule 219 ACA. The RMT sequence for the state 1111 is $(7777)$. The next state of 1111 is always 1111 for the update of any cell in any sequence. The RMT 7 is passive for rule 219. It can be observed that the RMTs in the RMT sequence of a fixed-point attractor are to be passive. Hence, we get the following lemma.

Lemma 1: Rule $R$ ACA forms a fixed-point attractor with state $S$ if the RMTs of the RMT sequence of $S$ are passive.

Since there are five, attractors in 4-cell rule 219 ACA, we can find five basins of attraction (Fig. 1). An attractor is the representative of the corresponding basin. If not specified otherwise, we shall refer fixed-point attractor as just attractor in our further discussion.

III. CHARACTERIZATION OF ACA

This section reports the properties of ACA to explore the fixed-point attractors. The proposed characterization is based
on a theorem and algorithms. The theoretical foundation is thus evolved and then employed to identify fixed-point attractors.

A. ACA with only fixed-point attractors

The following theorem states the necessary conditions of ACA, having only fixed-point attractors.

**Theorem 1:** Rule \( R \) ACA converges to fixed-point attractor if (i) RMT 0 (RMT 7) of \( R \) is passive and RMT 2 (RMT 5) is active, or (ii) RMTs 0 and 7 are passive and RMT 2 or 5 is active, or (iii) RMTs 0, 1, 2, and 4 (RMTs 3, 5, 6, and 7) are passive and RMT 3 or 6 (RMT 1 or 4) is active, or (iv) RMTs 1, 2, 4, and 5 (RMTs 2, 3, 5, and 6) are passive.

**Proof:** Proof of case (i): Let us consider RMT 0 of \( R \) is passive and RMT 2 is active. We shall show that the rule \( R \) ACA can reach to a fixed-point attractor from any initial state. Since RMT 0 is passive, the all-0 state (RMT sequence \( 00 \ldots 0 \)) is a fixed-point attractor. In any other state (except all-1), a sequence of consecutive 1s guided by 0s can always be found.

Consider, RMT 7 of \( R \) is active. Now in such a state (like \( \ldots 0111110 \ldots \)), we can find RMT 7 in its corresponding RMT sequence. If a cell with RMT 7 is selected to update, in that case, the sequence of consecutive 1s is divided into two sub-sequences of consecutive 1s guided by 0s. The new sequences have less number of 1s (like \( \ldots 0111110 \ldots \rightarrow \ldots 0101110 \ldots \)). After a number of similar updates, we can get a state with a number of single 1 and two consecutive 1s guided by 0s. A cell has state 1 with left and right neighbor’s states as 0s (010) implies that the cell can act on RMT 2. Since RMT 2 is active, all such cells can reach to state 0. So, finally, we get either the all-0 state (that is, the ACA is converged to all-0), or a state with two consecutive 1s guided by 0s (\( \ldots 0011100 \ldots \)). In second case, the RMT sequence contains RMT 0 and RMTs 1, 3, 6, and 4. If RMTs 1, 3, 4, and 6 are passive, the state \( \ldots 0110 \ldots \) itself is a fixed-point attractor. Otherwise, the ACA can reach to all-0 (fixed-point) attractor after some update of cells. Hence, from an arbitrary state with sequences of consecutive 1s guided by 0s, the ACA can reach to a fixed-point attractor. The all-1 state is a special state which contains no 0. However, if an arbitrary cell is updated, then we can get a 0, and then the new state can reach to fixed-point attractor with the above logic.

Now consider, RMT 7 is passive. Then, the all-1 state (RMT sequence \( 77 \ldots 7 \)) is another fixed-point attractor. A state with a sequence of consecutive 1s guided by 0s contains RMTs 3 and 6. If any one of them is active, the ACA with that state can reach to all-0 fixed-point. If both (RMT 3 and 6) are passive but RMT 1, 4 or 5 is active, the ACA can reach to all-1 fixed-points. If all RMTs except 2 are passive, then the state itself is a fixed-point attractor. Hence, the rule \( R \) ACA that can reach to fixed-point attractors from any initial state if RMT 0 of \( R \) is passive and RMT 2 is active.

While RMT 7 is passive and RMT 5 is active (and the rest RMTs are active or passive) it can be shown by similar logic that the rule \( R \) ACA converge to fixed-point attractors

**Proof of case (ii):** While RMTs 0 and 7 of \( R \) are passive and RMT 2 or RMT 5 is active, the ACA converges to fixed-point attractor. This proof can obviously followed from the proof of case (i).

**Proof of case (iii):** Consider RMTs 0, 1, 2, and 4 of \( R \) are passive. Then, the states where two 1s are separated by at least two consecutive 0s (like \( \ldots 001000100 \ldots \)) are fixed-point attractors, because the corresponding RMT sequences of these states contain only RMTs 0, 1, 2 and 4 (Lemma 1). Now consider a state which contains two or more consecutive 1s. Therefore, the corresponding RMT sequence of the state contains RMTs 3, 7 and 6 (along with others). If RMTs 3 or 6 is active, the number of 1s can be reduced to a single 1 separated by 0s during evolution of the ACA. The resultant state is a fixed-point attractor if corresponding RMT sequence contains RMTs 0, 1, 2 and 4 only. If the resultant state (like \( \ldots 001010 \ldots \)) contains any other RMT which is active (except 3 or 6) then updating the cells properly, we can reach to a state which is a fixed-point attractor.

While RMTs 3, 5, 6, and 7 are passive, the states where two 0s are separated by at least two consecutive 1s (like \( \ldots 110111011 \ldots \)) are fixed-point attractors because the corresponding RMT sequences of these states contain RMTs 3, 5, 6, and 7 only (Lemma 1). Now consider a state which contains two or more consecutive 0s. Therefore, the corresponding RMT sequence of state contains RMTs 4, 0 and 1 (along with others). If RMT 1 or 4 is active, the number of 0s can be reduced to single 0 separated by 1s during the evolution of the ACA. The resultant state is a fixed-point attractor if corresponding RMT sequence contains RMTs 3, 5, 6, 7 only. If the resultant state contains any other RMT which is active (except RMT 1 or 4) then we can reach to a fixed-point attractor.

**Proof of case (iv):** Consider RMTs 1, 2, 4 and 5 of \( R \) are passive. We shall next show that from any state, the ACA can reach to a fixed-point attractor while RMT 0, 3, 6, or 7 or none is active. Let us consider RMTs 0 and 7 are active. Now, from all-0 state, updating cell with RMT 0
we can reach to 0101 ... after a number of time steps. The state 0101 ... is a fixed-point attractor. The transitions are: 0000 ... → 0100 ... → ... → 0101 ...

Similarly, from all-1 state, the ACA can reach to 0101 ... 01 or 0101 ... 011 (depending on the number of cells). The state 0101 ... 01 is itself a fixed-point attractor and 0101 ... 011 can be a fixed-point attractor if RMTs 3 and 6 are passive. If RMT 3 or 6 is active, the ACA can obviously reach to a fixed-point attractors. Now it can easily be shown that the ACA can reach to a fixed-point attractor from any state.

Lastly, it can also be shown that rule \( R \) ACA can reach to a fixed-point attractor, if RMTs 2, 3, 5 and 6 are passive. We omit the detail steps of this proof because the rationale is similar with other cases.

There are 64 rules where the RMT 0 is passive and RMT 2 is active (Theorem 1(ii)). Similarly, identifying this way theorem 1 dictates that there are 146 ACA out of 256, which always approach to some fixed-point attractors. Such ACA rules are listed in TABLE III.

**Example 1:** Let us consider the rule 36 ACA, in which RMTs 0, 1, 2 and 4 are passive. Therefore, the rule satisfies the condition for convergence to fixed-point attractor (Theorem 1(iii)). Here, we assume number of cells are 4 and the initial state is 1111. The ACA can reach to a fixed-point attractor after some random update of cells from the initial state. One possible transition is: 1111(1) → 0111(4) → 0110(4) → 0110(2) → 0010 (the cell updated in a step is noted in bracket).

**Algorithm 1:** Construction of FPG

**Input:** Rule \( R \), TABLE II.

**Output:** Fixed-point graph \((G)\) with all passive RMTs are its vertices.

**Step 1.** Point out the passive RMTs of \( R \).

**Step 2.** Make forest with passive RMTs of \( R \).

**Step 3.** For each vertex \( u \) repeat step 4 and step 5.

**Step 4.** Find the next possible RMTs from TABLE II.

**Step 5.** If the next possible RMT is a vertex \( v \), then draw a directed edge from \( u \) to \( v \).

**Step 6.** Report the graph as FPG

**Example 2:** This Example illustrates the steps of Algorithm 1. Let us consider rule 77 ACA. Here, the passive RMTs are 1, 2, 3, 4, 5 and 6 (see TABLE I). As per step 2 of Algorithm 1, the vertices of the graph are 1, 2, 3, 4, 5 and 6 (see Fig. 2). Now considering the first vertex as vertex 1, we find the next RMTs of vertex 1 are RMT 2 and 3 (from TABLE II). As these two RMTs are also vertices, we draw directed edges from vertex 1 to both vertices 2 and 3 (see Fig. 2). Similarly, for vertex 3 the next possible RMTs are 6 and 7. But only one directed edge is possible from vertex 3 to vertex 6, since vertex 7 is absent (Fig. 2). After the construction of directed edges for all the vertices, the graph is the desired FPG. Fig. 2 is the FPG, resulted from the Algorithm 1 of rule 77 ACA.

Following Algorithm 1, we can get the FPGs for the rules of ACA, which always approach to some fixed-point attractors. Algorithm 2, reported next, provides the procedure for counting the number of attractors from a FPG of a particular ACA. For each vertex in the FPG, we count number of cycles of length equal to the ACA size. The sum of all cycles for each vertex is the total number of fixed-point attractors for that particular ACA.

\[
\begin{array}{cccccccccccc}
0 & 2 & 4 & 5 & 8 & 10 & 12 & 13 \\
16 & 18 & 24 & 26 & 32 & 34 & 36 & 40 \\
42 & 44 & 48 & 50 & 56 & 58 & 64 & 66 \\
68 & 69 & 72 & 74 & 76 & 77 & 78 & 79 \\
80 & 82 & 88 & 90 & 92 & 93 & 94 & 95 \\
96 & 98 & 100 & 104 & 106 & 112 & 114 & 120 \\
122 & 128 & 130 & 132 & 133 & 136 & 138 & 140 \\
141 & 144 & 146 & 152 & 154 & 160 & 161 & 162 \\
163 & 164 & 165 & 166 & 167 & 168 & 169 & 170 \\
171 & 172 & 173 & 174 & 175 & 176 & 177 & 178 \\
179 & 180 & 181 & 182 & 183 & 184 & 185 & 186 \\
187 & 188 & 189 & 190 & 191 & 192 & 194 & 196 \\
197 & 200 & 202 & 203 & 204 & 205 & 206 & 207 \\
208 & 210 & 216 & 217 & 218 & 219 & 220 & 221 \\
222 & 223 & 224 & 225 & 226 & 227 & 228 & 229 \\
230 & 231 & 232 & 233 & 234 & 235 & 236 & 237 \\
238 & 239 & 240 & 241 & 242 & 243 & 244 & 245 \\
246 & 247 & 248 & 249 & 250 & 251 & 252 & 253 \\
254 & 255 \\
\end{array}
\]
Algorithm 2: Counting Attractors
Input: FPG (G) from Algorithm 1, n (CA size)
Output: Number of fixed-point attractors

Step 1. For each vertex v of (G), repeat steps 2 and 3.
Step 2. Find whether v can be reached after exploring (n-1) vertices of (G)
Step 3. If there are m ways to return back to v, add this m with the total no. of fixed-point attractors to be reported.
Step 4. If no such path is found, a fixed-point attractor can not be formed with vertex v.
Step 5. Report the total no. of fixed-point attractors.

Example 3: This Example illustrates the execution steps of Algorithm 2. Let us consider, for the graph G (Fig. 2), we count the number of fixed-point attractors for 4 cell ACA. Starting from vertex 1, the vertex 1 can be reached after exploring vertices 3, 6 and 4 (step 2 of Algorithm 2). The path is 1 $\rightarrow$ 3 $\rightarrow$ 6 $\rightarrow$ 4 $\rightarrow$ 1 (Fig. 2). Similarly, other cycles of length 4 (as 4 cell ACA) in G can be found, and the total cycles of length 4 are the total number of fixed-point attractors in G. The cycles of length 4 in G (Fig. 2) are:

1) $1 \rightarrow 3 \rightarrow 6 \rightarrow 4 \rightarrow 1$
2) $3 \rightarrow 6 \rightarrow 4 \rightarrow 1 \rightarrow 3$
3) $6 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 6$
4) $4 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 4$
5) $2 \rightarrow 5 \rightarrow 2 \rightarrow 5 \rightarrow 2$
6) $5 \rightarrow 2 \rightarrow 5 \rightarrow 2 \rightarrow 5$

Hence, the total number of attractors in G is 6 (see Fig. 2).

TABLE IV
THE RULES OF ACA WITH MULTIPLE FIXED-POINT ATTRACTORS (MAACA)

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>5</th>
<th>12</th>
<th>13</th>
<th>36</th>
<th>44</th>
<th>68</th>
<th>69</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>100</td>
<td>104</td>
<td>128</td>
<td>130</td>
<td>132</td>
<td>133</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>140</td>
<td>141</td>
<td>144</td>
<td>146</td>
<td>152</td>
<td>154</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>162</td>
<td>164</td>
<td>166</td>
<td>168</td>
<td>170</td>
<td>172</td>
<td>174</td>
<td>176</td>
<td></td>
</tr>
<tr>
<td>178</td>
<td>180</td>
<td>182</td>
<td>184</td>
<td>186</td>
<td>188</td>
<td>190</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>196</td>
<td>197</td>
<td>200</td>
<td>202</td>
<td>203</td>
<td>204</td>
<td>205</td>
<td></td>
</tr>
<tr>
<td>206</td>
<td>207</td>
<td>208</td>
<td>210</td>
<td>216</td>
<td>217</td>
<td>218</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>221</td>
<td>222</td>
<td>223</td>
<td>224</td>
<td>228</td>
<td>232</td>
<td>234</td>
<td></td>
</tr>
<tr>
<td>236</td>
<td>237</td>
<td>238</td>
<td>240</td>
<td>242</td>
<td>244</td>
<td>246</td>
<td>248</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>252</td>
<td>254</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Out of 256, 146 ACA in two-state 3-neighborhood interconnection that always approach towards fixed-point attractors are already listed in TABLE III (Theorem 1). However, we can further identify a set of ACA which are having multiple attractors. Algorithm 2 guides us to find such set of ACA. There are 83 ACA (TABLE IV) which have multiple attractors and are referred as multiple attractors ACA (MAACA). These MAACA are utilized in designing pattern classifier.

IV. DESIGN OF CLASSIFIER

An n-cell ACA with multiple fixed-point attractors (MAACA) can be viewed as a natural classifier. It classifies a set of patterns into different distinct classes, each class contains set of states of the attractor basin. For the identification of a class of patterns, the attractors, representing the classes, need to be stored in memory (Fig. 3). To identify the class of an input pattern $P$, the MAACA is loaded with $P$ and updated till it reaches to an attractor. Then, from the attractor and the stored information, one can declare the class of the pattern $P$. In Fig. 3, the class of $P$ is I. However, if there are more than two attractors, then a set of attractors identify a class.

Example 4: Let the MAACA of Fig. 1 be employed to classify patterns into two classes (say 1 and 2), where class 1 is represented by the states of three attractor basins and class 2 is represented by the states in the rest of basins. Let the attractors 1111, 0111 and 1110 be for class 1 and the rest attractors 1101 and 1011 be for class 2. Hence, the MAACA of Fig. 1 can act as 2-class pattern classifier.

The design of MAACA based 2-class classifier for pattern classification demands the proper distribution of the patterns among the CA attractor basins. The design of classifier for two pattern sets $P_1$ and $P_2$ should ensure that elements of one class (say $P_1$) are covered by a set of attractor basins that do not include any member from the class $P_2$. However, for real-life data sets, the attractor basins may mix up the patterns of two classes. Therefore, the primary metric for evaluating classifier performance is the classification accuracy. It is measured as:

$$
\text{efficiency} = \frac{\# \text{ patterns properly classified}}{\text{Total No. of patterns}} \times 100 \quad (2)
$$

Example 5: Let us consider two pattern sets of class 1 say $P_1$={$1111, 0011, 1100, 0110$} and class 2 say $P_2$={$0000, 0011, 1000$}. Suppose the MAACA of Fig. 1 is considered as the classifier. The patterns 1111, 0011, 1100 and 0110 from set $P_1$ are under attractor basin class 1 while patterns 0000, 0011 from set $P_2$ are in attractor basin class 2. However, the pattern 1000 is in class 2, but it is wrongly identified by the classifier as class 1. Hence, The total number of properly identified patterns are 6. So, by using the formula of equation 2 the efficiency of the classifier utilizing the above given patterns, for the particular ACA is 85.714%. Depending on the pattern sets and the MAACA, the efficiency can vary. The above discussion shows that the MAACA are the candidates to qualify as a pattern classifier. Out of 83 MAACA (TABLE IV), however, we exclude rule 204 ACA, because all the states of 204 ACA are attractors. For the rest 82 rules, we design Algorithm 3 for getting the target classifier. Algorithm 3 takes two pattern sets ($P_1$ and $P_2$) and a list of 82 ACA. Then
the algorithm reports the ACA having maximum efficiency as pattern classifier.

Algorithm 3: Classifier
Input: TABLE IV – {204 ACA}, n (Size of ACA), Two pattern set \( P_1 \) and \( P_2 \).
Output: ACA with maximum efficiency
Step 1. For each ACA, \( A \) ∈ TABLE IV –{204}
repeat Step 2 to Step 8.
Step 2. Identify the attractors of \( A \) utilizing FPG (Section III-B).
Step 3. Repeat Step 4 and Step 5 for each pattern \( p \) of \( P_1 \) and \( P_2 \).
Step 4. Load \( A \) with \( p \).
Step 5. Run \( A \) until it reaches to an attractor, \( attr \).
Step 6. Suppose \( n_1 \) and \( n_2 \) are the number of patterns from \( P_1 \) and \( P_2 \) respectively mapped into an attractor, \( attr \)
   if \( n_1 > n_2 \), then declare the \( attr \) for \( P_1 \).
   if \( n_1 < n_2 \), then declare the \( attr \) for \( P_2 \).
   if \( n_1 = n_2 \), then declare \( attr \) for \( P_1 \) or \( P_2 \) arbitrarily.
Step 7. Repeat Step 6 for each attractor.
Step 8. Find efficiency as \( \sum_{\{\max (n_1,n_2)\}} \frac{|P_1|+|P_2|}{|P_1|} \).
Step 9. Report the ACA with maximum efficiency as our target classifier.

Algorithm 3, designs the classifier for given pattern sets. This phase is commonly referred as training. However, to understand the effectiveness of the designed classifier, we need to test it with a new set of patterns. Here, we report the efficiencies of training phase as well as testing phase for some data sets. For our purpose, we have taken standard and widely studied data sets, available at http://www.ics.uci.edu/~mlearn/MLRepository.html. Six data sets are considered for experimentation—Monk 1, Monk 2, Monk 3, Haberman, Tic-Tac-Toe, Spect Heart.

All the data sets taken into consideration have two classes. Two handle such real data, data sets are suitably modified to fit the input characteristics of the proposed pattern classifier. To calculate the efficiency of the proposed classifier, the training of data sets are performed, and the ACA with maximum efficiency is reported as the proposed classifier. Then, the classifier (that is, the ACA) is tested with different data sets to measure the classification efficiency. Since the cells of ACA are updated arbitrarily, classification efficiency for a single ACA rule may little change in different runs.

TABLE V shows the maximum efficiency of different ACA using monk-1 data set obtained during training of the classifier. It shows that efficiency of a data set (in training) changes if the ACA changes. TABLE VI reports the performance result of the proposed classifier using different real-life data sets. Column 1 shows the name of data set while column 2 reports the size of ACA. The efficiencies of the classifier during training and testing are depicted in next two columns, and the ACA, acting as the classifier is in the last column. We further compare the performance of proposed classifier with others [2], [3]. TABLE VII reports a comparative study of efficiency of various classifier algorithms with our classifier. The efficiency of the proposed classifier, with respective ACA rule is reported in the last column.

V. Conclusion
This paper has reported a detail characterization of asynchronous cellular automata having multiple fixed-point attractors with the target to model this class of ACA in designing the efficient pattern classifier. Theorem and algorithms have been designed for the characterization of these ACA. The concept of FPG has proposed for the counting of attractors, which identify the MAACA, utilized for the pattern classification. The algorithm for the design of pattern classifier has also been proposed which calculates the efficiency of the classifier.
TABLE VII
COMPARISON OF CLASSIFICATION ACCURACY

<table>
<thead>
<tr>
<th>Data set</th>
<th>Algorithms</th>
<th>Efficiency in %</th>
<th>Efficiency in % (proposed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monk 1</td>
<td>Bayesian</td>
<td>99.9</td>
<td>92.129</td>
</tr>
<tr>
<td></td>
<td>C4.5</td>
<td>100</td>
<td>(rule 76)</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></td>
<td>Traditional CA</td>
<td>61.111</td>
<td></td>
</tr>
<tr>
<td>Monk 2</td>
<td>Bayesian</td>
<td>69.4</td>
<td>91.435</td>
</tr>
<tr>
<td></td>
<td>C4.5</td>
<td>66.2</td>
<td>(rule 76)</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></td>
<td>Traditional CA</td>
<td>67.129</td>
<td></td>
</tr>
<tr>
<td>Monk 3</td>
<td>Bayesian</td>
<td>92.12</td>
<td>94.212</td>
</tr>
<tr>
<td></td>
<td>C4.5</td>
<td>96.3</td>
<td>(rule 76)</td>
</tr>
<tr>
<td></td>
<td>TCC</td>
<td>76.58</td>
<td></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></td>
<td>Traditional CA</td>
<td>80.645</td>
<td></td>
</tr>
<tr>
<td>Haber-man</td>
<td>Traditional CA</td>
<td>73.499</td>
<td>84.615 (rule 164)</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>Sparse grid</td>
<td>98.33</td>
<td>99.432 (rule 76)</td>
</tr>
<tr>
<td></td>
<td>ASVM</td>
<td>70.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LSVM</td>
<td>93.330</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Traditional CA</td>
<td>63.159</td>
<td></td>
</tr>
</tbody>
</table>

The performance of the proposed classifier has also tested with real-life date sets and compared with some well-known classifier. It has been reported that, the proposed ACA based classifier, performs better than them, and always better than traditional CA based classifier.

REFERENCES