Diversity of complex systems produced by a class of cellular automata

T. Suzudo

1Research Group for Advanced Reactor Systems
Japan Atomic Energy Research Institute
Shirakata Tokai-mura, 319-1195 Japan
phone:+81-29-282-6077, fax:+81-29-282-6122
Email: suzudo@clsu3a0.tokai.jaeri.go.jp

Abstract

The variety of self-organized patterns is studied using simple cellular automaton rules. A class of rules to model an artificial ecological system is explored to demonstrate the diversity of complex spatio-temporal patterns appearing from similar but different rules. Three typical patterns are obtained from the numerical survey and each pattern is statistically analyzed to clarify its inherent property.

Keywords: diversity of complex systems, prey-predator interactions, 2-dimensional cellular automata, gliders, spirals, flocks, statistical analyses

1. Introduction

Nature contains many examples of complex systems creating different spatio-temporal patterns from similar constituents, for example clouds, animal flocks or fish schools. In addition to these examples, it is possible to recognize that the variety of self-organization is also a factor which contributes to the diversity of nature. For example, only 20 types of amino-acid molecules create many different cells in multi-cellular organism; genetically similar species can have quite different breeding strategies, with such species separation behavior becoming a driving force behind evolution; and human beings form many kinds of social groups and interactions between such groups affect their development. The variety of self-organization is therefore an essential property of complex systems in nature.

Complex systems created by mathematical dynamics have also proved to be varied. For example, Wolfram (Wolfram 1984) and Langton (Langton 1990) studied the complexity of one-dimensional cellular automata in depth, and produced many kinds of complex spatio-temporal patterns. Their cellular automata were not related to actual phenomena, as they were not specially interested in a particular complex phenomenon in nature, and their main goal was not to show the variety of patterns, but the results strongly suggested that cellular automata would be suitable for such a study.

This paper in turn attempts to establish the preliminary work necessary to elucidate the mechanism of such diversity in nature by using cellular automata, with a special focus on prey-predator interaction. The population dynamics of such interactions are typical complex phenomena, and have been studied by many mathematicians since Lotka and Volterra (Lotka 1920; Volterra 1931) employed a second-order nonlinear differential equation for their studies. Prey-predator interactions are a part of the food chain system which could be one factor in determining the diversity of ecosystem. These interactions may therefore help to explain the nature of diversity. Prey-predator interactions would, therefore, appear to be a suitable starting point for exploring the diversity of such complex systems and hence were selected as the subject of this study.

In order to explore the nature of diversity, the rules for prey-predator cellular automaton were not fully defined and some detailed parameters were intentionally left undetermined. In this way, it is possible to have a class of cellular automata that searches for different self-organized patterns. The search is conducted in a try-and-error style using a graphical user interface panel. Priority is placed on a demonstration of the diversity rather than simulating real phenomena. Thus, the model is intended to be simple rather than precise, so that the searching rule space does not become too large.

Section 2, below, describes the modeling strategy of the target cellular automata. Section 3 examines a class of the target cellular automata for their variety of self-organized patterns. Conclusions are given in Section 4.

2. A Simple Prey-Predator Model Using Cellular Automata

A cellular space composed of identical squares regularly arranged on a two-dimensional space is used for this model. This is a common way of arranging two-dimensional cellular automata. The model is programmed as prey and predators play a 'survival game' on this checker board. The total number of cell states of this model is set to three, i.e. 'zero', 'one' and 'two'. The meanings of each state are as follows:

- 'zero' for empty,
• ‘one’ for occupied by prey,
• ‘two’ for occupied by a predator.

The states of all cells are updated synchronously in discrete time steps. The state of each cell at the next time step is determined by the current states of the cell and its neighbors. It is therefore necessary to define a transition table to stipulate the cell state at the next time step for all the possible configurations of neighboring cells. This table is often called a rule. There are two major neighbor types for two-dimensional cellular automata; they are von Neumann and Moore neighborhoods. The von Neumann neighborhood is composed of a cell and its four direct neighbors, while the Moore neighborhood includes four diagonal neighbors in addition to the von Neumann neighborhood. For the sake of simplicity, I selected the von Neumann neighborhood for this study. Because there are three possible states per cell, the total number of entries in the transition table is $3^9 = 19683$. There are three possible output states for each entry, and it is possible to define as many as $3^{18} \approx 10^{116}$ of cellular automaton rules; this is the size for a rule space of three-state cellular automaton with a von Neumann neighborhood.

The next modeling process is to select from the whole space the partial rule space corresponding to the prey-predator game. First, we assume the isotropic cellular space for the ‘survival game’, i.e. there are neither favorable nor unfavorable directions for prey and predators. This condition can be achieved by selecting only rules being rotationally and reflectionally symmetric. This study adopted an outer-totalistic rule, in which each entry is defined by the state of a cell and the total of each state in the neighborhood. Note that outer-totalistic rules satisfy rotational and reflectional symmetries. Second, several principles were adopted to narrow down the list of the potential prey-predator rules. These principles are listed in the following, where the words ‘is born’ mean that an empty cell becomes occupied by prey or a predator in the next time step, and the word ‘survive’ means that a prey-occupied or predator-occupied cell becomes occupied by the same agent.

1. No prey is born unless at least one prey cell is in the neighborhood or if there are any predators in the neighborhood. This ensures that prey can breed in the absence of predators, but it cannot appear from nothing.

2. In addition to the previous item concerning the breeding of prey, prey is supposed to breed indefinitely if no predators exist at all. Because only prey and predator are defined in this cellular space, this simplification does not seem too unreasonable. Note that the Lotka-Volterra equation also leads to this solution when the number of predators is nil (Reichl 1980). For the outer-totalistic rule of a von Neumann neighborhood this condition is attained simply by the transition, ‘prey is born when the four outer-neighbors are composed of one prey cell and three empty cells’. How this transition satisfies the above condition is explained in the Appendix of a paper by Suzudo (Suzudo 1999).

3. No predator is born without either prey or a predator in the neighborhood, i.e. predators can breed, but only when there are a parent and food in the neighborhood.

4. No prey survives if any predators are in the neighborhood. This ensures that prey is killed by a neighboring predator.

5. For a prey-occupied cell to become predator-occupied, at least one neighbor must be a predator. This ensures that predators need a parent when they are born.

6. No predator survives without at least one prey cell in the neighborhood. This ensures that predators with no food die. Note that the Lotka-Volterra equation also leads to the extinction of predators when no prey is around (Reichl 1980). Predators in nature can survive without food for a certain period, but in the model proposed here they become extinct immediately after their prey disappears. A time margin for the extinction of predators was abandoned for the sake of simplicity in this study.

7. Naturally, a predator is never replaced by prey.

The rules satisfying the principles described above form a class of prey-predator cellular automata rules. To fully define the rule, it is necessary to answer the YES/NO checklist shown in Appendix 1. The total number of entries in the checklist is 34, and the total number of rules included in this class is $2^{34} \approx 10^{10}$. The above design principles eventually leave a large rule space, and it is therefore not realistic to analyze every single cellular automaton. Fortunately many items on the checklist have little influence on the dynamics, so it could reasonably be hoped that most of the typical complex dynamical patterns have been attained by the try-and-error style search.

3. Analysis and Discussion

Dynamical Spatial Patterns

The first step in the analysis is to collect self-organized patterns from the class of cellular automata obtained in the previous section. The word ‘self-organized pattern’, however, may not be clear, because it is a subjective matter. It is generally accepted that this technical term has the same meaning as ‘complexity’, and there are some descriptive criteria necessary in order to be labeled ‘complex’. These are the Wolfram Class IV cellular automata (Wolfram 1984) and the ‘edge of chaos’ proposed by C.
G. Langton (Langton 1990). In these concepts, the following cellular automata are classified as not complex:

1. The cellular space evolves into homogeneous configuration or a few isolated simple periodic patterns in the early stage of time development.

2. The cellular space becomes chaotic, i.e. random and dynamic all over the space and no specific pattern is seen.

A follow-up study by Suzudo (Suzudo 1999) added one more condition to the above list:

3. The cellular space becomes random and static in the early stage of time development.

Using these criteria, many cellular automata were tested to search for complex patterns. Those satisfying the above criteria were abandoned, and the rest were further investigated.

The above investigation revealed three typical dynamical spatial patterns of complexity. The first pattern is a ‘glider’, composed of several prey and predator cells (see Fig. 1). Prey in the glider breed toward a certain direction, but at the same time predators keep chasing and reducing them, and as a result gliders can travel while keeping their shape. Gliders travel horizontally or vertically in the cellular space, but not diagonally like the one in the Game of Life (Gardner 1970). Every complex cellular automaton has a set of inherent gliders, and a slight change of the rule can cause another set of gliders. These gliders are usually produced by some template, which is usually another kind of glider traveling in the opposite direction. This kind of template can be seen in the top right-hand corner of Fig. 1. Gliders are the most basic pattern to make a system complex, because they can act as the medium for remote communications. In the prey-predator system in particular, they may be interpreted as traveling animals carrying food.

The second pattern is a spiral, see Fig. 2. This kind of pattern appears only in some particular complex cellular automata. The appearance of the spiral is more inanimate than the gliders described above. Some types of spirals have a finite size, and they are destroyed after a limited life time. Other types of spirals grow indefinitely while covering the whole cellular space. The latter type sometimes destroy themselves from the focus point. This destruction can cause the whole cellular space to become homogeneously empty. Thus the spiral can cause a catastrophic extinction of the two species. Spiral patterns appear to simulate unstable population dynamics such as extinction and population explosion. Which type of the spirals appears, depends not only on the fine tuning of the rule, but also on the initial configuration.

The third pattern is a flock of prey as seen in Fig. 3. This kind of pattern appears only in some particular complex cellular automata. The prey is caught by the predators at the edge of the flock, but the prey breeds at the same time. The population of prey sometime increases sharply, but it is soon reduced by the predators. This feedback give the system a population balance. As ecological systems, cellular automata with this pattern are stable, i.e. neither population explosion nor extinction occurs. So far the coexistence of the second and third pattern in the same cellular automaton has not been found.

Population trends for the above complex cellular automata are classified into two kinds. The first kind is when the prey trends and the predators change almost in-phase. Cellular automata composed of gliders and spirals belong to the first kind. See the population trend graph in which spirals appear (Fig. 4). This in-phase trend occurs because more predators can thrive when more prey are born.

On the other hand, this is not the case when many prey form a flock, because prey can breed without being
caught by predators to some extent. As predators catch the prey at the flock edges, the number of prey decreases and the number of predators increase. Thus, the population trends of both species change in an out-of-phase manner (see Fig. 5).

**Statistical Analysis**

Generally speaking, every complex cellular automaton looks different. This implies that some statistical measure of spatio-temporal dynamics can describe the inherent properties of complex cellular automata. I proposed ‘the entropy trajectory’ (Suzudo 2000) which is briefly explained below and is used to reinforce the analyses of the self-organized patterns introduced above.

Consider four adjacent sites of a cellular automaton space such as \((i, j), (i + 1, j), (i, j + 1)\) and \((i + 1, j + 1)\), where \(i\) and \(j\) are the x-axis and y-axis positions of an arbitrary cell. There are \(3^4 = 81\) possible patterns for this local patch if each cell takes either a ‘zero’, ‘one’ or ‘two’ state. The entropy of the spatial pattern of configuration, \(H_s\), at an arbitrary time step \(\tau\) can be defined by

\[
H_s(\tau) \equiv - \sum_k P^k_s(\tau) \log_n(P^k_s(\tau)),
\]

(1)

where \(P^k_s(\tau)\) is the probability for a particular pattern of the local patch at the time step \(\tau\). Note that the base for the logarithmic function \(n\) is set to be \(n^4\) where \(n\) is the total number of states per cell. The entropy assumes, in this way, a value between nil and unity.

Similarly to \(H_s\), it is possible to define the entropy associated with a temporal change. Consider again an arbitrary site \((i, j)\). If the state of the site changes from the last time step, the ‘temporal-change state’ of the site is set to ‘1’, otherwise it is set to ‘0’. Let \(P^k_t(\tau)\) be the probability for a particular change-state pattern of the local patch described above at the time step \(\tau\). The spatial entropy of the change state can be regarded as the entropy associated with temporal change such as

\[
H_t(\tau) \equiv - \sum_k P^k_t(\tau) \log_{16}(P^k_t(\tau)).
\]

(2)

Again note that a base of 16 is used for the logarithmic function as the entropy assumes a value between nil and unity. The entropy trajectory is a two-dimensional time series composed of \(H_t\) and \(H_s\), and is displayed by plotting on \(H_t - H_s\) plain (see Fig. 6). In this plot, trajectory points in the bottom left-hand corner correspond to homogeneous or simple isolated periodic patterns, those at the top right-hand corner to completely chaotic patterns, and those around the top left-hand corner to the random static patterns. Note that the trajectories at the three corners correspond to the not complex criteria introduced in the top of this section. The trajectory in the other range implies the corresponding dynamics are complex. The trajectory never goes to the bottom right-
hand corner, because it is impossible to have random temporal changes with spatially regular patterns.

Entropy trajectories for the GOL are shown in Fig. 6. An inherent ‘attractor’ of the entropy trajectory is observed for the GOL (Suzudo 2000). Although the ‘attractor’ is not completely asymptotic and may move slightly mainly in the direction of the length, this is a powerful means to characterize complex systems.

Cellular automata shown in Figs. 1-3 are called Types I-III cellular automata, respectively, for convenience. The entropy trajectories for these cellular automata are also shown in Fig. 6. Type-I cellular automaton is composed of only gliders, and its entropy trajectory forms a long line similar to that of the GOL. This suggests a proportional relation between $H_t$ and $H_s$. The length of the line indicates how changeable the population of gliders is. The trajectory for the GOL has almost the same $H_s$ as Type I, but a smaller $H_t$, this means the GOL is more static than Type I. This difference is caused by the fact that many inactive (or invariant) debris exist in the GOL. There is little such debris in Type I.

Type-II cellular automaton is composed of gliders and spirals. The trajectory is similar to that of Type I, but the range is shorter. The location of the trajectory suggests that the cellular space of Type II is busier than Type I. The extension of the Type-I trajectory toward the right-upward direction leads to a Type-II trajectory. This seems to suggest that the spiral is formed by the growth of gliders. Because Type-II cellular automaton creates only spirals with limited size, the trajectory is relatively stable and the length of the trajectory range is short.

Type-III cellular automaton is composed of gliders and flocks. The trajectory range is shorter than that of Type I, thus the dynamics are more stable. It can also be seen that the trajectory is located above that of the Type-II cellular automaton. This suggests the existence of inactive parts in cellular space. Remember that internal prey cells of flocks do not change until they are exposed to predators.

In summary, the statistical analyses by the entropy trajectory yield objective information on self-organized patterns. As described above, the observed patterns in the cellular space were recognized to have different statistical properties, and the diversity of the complex systems was confirmed with this measure.

4. Conclusion

In order to demonstrate the diversity of complex systems, an artificial prey-predator population dynamics was considered, and a class of cellular automata introduced for this purpose. Visual analyses indicated that various spatio-temporal patterns appeared from similar rules, and that these patterns were roughly classified into three sorts, i.e., a glider, a spiral and a flock. The entropy trajectory was proposed as a statistical measure for describing the property of cellular automaton dynamics, and the analyses of using this quantity confirmed each pattern has an inherent property of complexity. Future work could include more realistic modeling with greater interpretation of detail in each pattern. Cellular automata discussed in this paper can be seen using demonstration tools. See Appendix 2 for more detail.

References


Appendix 1: Full Rule Descriptions

In the case of outer-totalistic rules, the total number of rule entries is the multiplication of the possible cell states
by the possible combinations of outer-neighbors. For our cellular automata, this is $3 \times 15 = 45$, but some are already determined in advance because of the principles introduced in Section 2. Special notation to express each entry is used hereafter, such as $\{1,2,0\}$. The first number denotes the state of the center cell, the second the total number of prey cells, and the third the total number of predator cells in the neighborhood.

The first group of entries is on whether prey is born or not. The state of the center cell and the total number of prey cells must be 0. When the total number of prey cells is 1, the prey is always born according to Principl 2. Thus the remaining entries are:

1. $\{0,2,0\}$
2. $\{0,3,0\}$ and
3. $\{0,4,0\}$

The second group is on whether a predator is born or not. At least one prey cell and one predator are around, so we have six entries as follows:

4. $\{0,1,1\}$
5. $\{0,1,2\}$
6. $\{0,1,3\}$
7. $\{0,2,1\}$
8. $\{0,2,2\}$ and
9. $\{0,3,1\}$

Prey may survive when no predator is around, so we have five entries for whether prey survives or not, as follows:

10. $\{1,0,0\}$
11. $\{1,0,1\}$
12. $\{1,0,2\}$
13. $\{1,0,3\}$ and
14. $\{1,0,4\}$

Prey may be replaced with a predator when at least a predator is around, so there are ten entries for this change, as follows:

15. $\{1,1,0\}$
16. $\{1,1,1\}$
17. $\{1,1,2\}$
18. $\{1,1,3\}$
19. $\{1,1,4\}$
20. $\{1,2,0\}$
21. $\{1,2,1\}$
22. $\{1,2,2\}$ and
23. $\{1,3,1\}$

Predator may survive when prey is around, so there are ten entries for survival of the predator, as follows:

24. $\{2,1,0\}$
25. $\{2,1,1\}$
26. $\{2,1,2\}$
27. $\{2,1,3\}$
28. $\{2,2,0\}$
29. $\{2,2,1\}$
30. $\{2,2,2\}$
31. $\{2,3,0\}$
32. $\{2,3,1\}$ and
33. $\{2,4,0\}$

The rules used in this paper are shown in Table 1. The entry numbers in the table correspond to the number listed above. Note that the list includes only entries which are not determined by the principles given in Section 2, and the combination of the principles and the table gives the full definition of the rules.

Appendix 2: Demonstration Tools

There are a few methods for looking at the cellular automata discussed in this paper. The easiest way is to connect

"http://www001.udpp.so-net.ne.jp/suzuudo/" and to go to the English page. The applets showing the cellular automata introduced in this paper can be executed from the Aquarium in the paragraph on ‘Java applet demos ...’ at this site. It is also possible to download the whole set of the programs at the same site. The program is called CA демо, and is written in Java language version 1.1, and has also a capability of the statistical analyses mentioned in Section 3. The command to execute the cellular automata introduced in this paper by using CA демо is

```
> java CAdemo ruleClass=AquaRule
```

The default rule is ‘pond’, which is a Type-I rule. To edit the rule, press ‘rule’ button in the CAdemo window, then you will see the graphical user interface to select all the items shown in Appendix 1 (see Fig. 7). The other preset rules introduced in Table 1 are easily selected by using the ‘Choice’ interface at the bottom of the interface window.

Another on-line source for demonstrations is MCell; this is a MS Windows-based cellular automaton tool created by Mirek Wojtowicz. The download site is "http://psoup.math.wisc.edu/mcell/". The rules are included in the ‘Neumann binary’ folder. There is a Java version of MCell called MCell which can be found at the same site.

Cellsprings by J. M. G. Elliott is an applet for cellular automaton demonstration tools available on the Web. This also includes the cellular automata in this paper. The URL for access is "http://jmge.net/java/csprings/". The rules discussed in this paper can be selected from the ‘General 2Bit Rule’ item in the ‘Presets’ pull-down menu.

In both of the alternative tools, the names of the cellular automata are consistent with Table 1. Note that rules irrelevant to this study are also included in these tools.
Aquarium rule is the group of CA rules which are a sort of the prey-predator system. The state zero (black) cell means empty, the state one (white) and two (red) cells correspond to prey and predator, respectively. The von Neumann neighborhood is applied. Each rule entry is defined by the total of each state in the neighbor sites. As the three cell states are defined, nine update patterns are possible.

1) empty $\rightarrow$ prey: At least one prey must be at neighbor sites and any predators must not be at neighbor sites. You have to determine if born or not according to the total number of neighboring prey.

2) empty $\rightarrow$ predator: At least one predator and one prey must be at neighboring sites. You have to determine what combination cause the birth of predator.

3) empty $\rightarrow$ empty: The empty cell continue to be empty if any conditions defined above 1) and 2) are not satisfied. You do not have to define this.

4) prey $\rightarrow$ prey: Any predators must not be at neighbor sites. You have to determine if survive or not.

Figure 7: The user interface to select the rule of CADemo: The top pane gives explanatory information on the class of rules.

Table 1: Examples of the prey-predator cellular automaton rules satisfying the principles in Section 2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Pattern</th>
<th>Ent. 1-3</th>
<th>Ent. 4-9</th>
<th>Ent. 10-14</th>
<th>Ent. 15-24</th>
<th>Ent. 25-34</th>
</tr>
</thead>
<tbody>
<tr>
<td>pond</td>
<td>glider</td>
<td>oox</td>
<td>o00ox0</td>
<td>ox00x</td>
<td>ox000xx0</td>
<td>oxx000xx0</td>
</tr>
<tr>
<td>plankton</td>
<td>glider</td>
<td>oox</td>
<td>o00ox0</td>
<td>ox00x</td>
<td>ox000xx0</td>
<td>oxx000xx0</td>
</tr>
<tr>
<td>tanks</td>
<td>glider</td>
<td>oox</td>
<td>o00ox0</td>
<td>ox00x</td>
<td>ox000xx0</td>
<td>ox000xx0</td>
</tr>
<tr>
<td>lake</td>
<td>glider</td>
<td>oxx</td>
<td>o00xx</td>
<td>x00xx</td>
<td>ox0xx00</td>
<td>ox0xx000</td>
</tr>
<tr>
<td>galaxy</td>
<td>glider, spiral</td>
<td>ooo</td>
<td>ox0xx</td>
<td>xxxx</td>
<td>ox0xx00</td>
<td>ox0xx000</td>
</tr>
<tr>
<td>wave</td>
<td>glider, spiral</td>
<td>oox</td>
<td>ox0xx0</td>
<td>ox0xx</td>
<td>ox0xx0x0</td>
<td>ox0xx000</td>
</tr>
<tr>
<td>typhoon</td>
<td>glider, spiral</td>
<td>ooo</td>
<td>ox0xx</td>
<td>xxxx</td>
<td>ox0xx00</td>
<td>ox0xx000</td>
</tr>
<tr>
<td>honeycomb</td>
<td>glider, flock</td>
<td>xxx</td>
<td>oox00</td>
<td>o00xx</td>
<td>xx00xx0</td>
<td>oxx000xx0</td>
</tr>
</tbody>
</table>

Note. The symbol ‘o’ and ‘x’ mean yes and no, respectively.