Searching for Pattern-Forming Asynchronous Cellular Automata –An Evolutionary Approach

Tomoaki Suzudo

Department of Nuclear Energy Systems, Japan Atomic Energy Research Institute,
Tokai-mura, Japan 319-1195
suzudo@clsu3a0.tokai.jaeri.go.jp

Abstract. This paper discusses a class of 2-dimensional asynchronous cellular automata with conservation of mass, for the formation of patterns in groups. The previous study reported a methodology of searching, automatically, for pattern-forming cellular automata using a genetic algorithm; this approach successfully found a few types of pattern-forming rules. The current study is a series of statistical analyses of one of the classes found by the above methodology, with the hope of understanding the mechanisms of the pattern formation. These analyses lead to some basic logic necessary to the pattern formation, but not to enough information to elucidate the whole mechanism of the pattern formation. This result suggests that the existence of unidentified cooperative operations between the different transitions of the cellular automaton rule to carry out the pattern formation.

1 Introduction

Nature produces many kinds of patterns, and the mechanism of such pattern formations are often unknown. Cellular automaton (CA) is a mathematical tool suitable for such studies and has been exploited intensively by many researchers. Because this mathematical tool is especially suitable for visual analyses of the results of calculation, the self-organization of spatial patterns is a very common topic among studies. For instance, the hodgepodge machine[1], and the cyclic cellular automata[2] give intricate patterns and are reminiscent of real phenomena of some chemical reactions. The formation of crystal-like pattern is also studied by a class of CAs [13].

In common CAs, all of the cell states are updated synchronously, and the patterns formed by such CAs are usually heavily dependent on the synchronous timing of the events occurring at each cell. This means that such pattern formations are sensitive to a tiny delay of the events at each cell. Such perturbations always exist in the natural environment, yet self-organization observed in real physical, chemical, or biological systems is robust and not dependent upon synchronous timing. Several scientists have already started questioning the assumption of the synchronous adjustment of random boolean networks[7, 3, 4] and CAs[8, 12, 11, 16, 14].

Applications of artificial dynamical systems, such as random boolean networks or CAs, to the study of spatial pattern formations are expected to reveal the mechanism of advanced self-organizations frequently seen in biochemical systems and sometimes even in solid-state physical systems such as self-protection, self-replication, self-maintenance, etc.. However, the discrepancies, discussed above, between artificial and natural systems may spoil the fruit yielded by such studies.

In this paper, we will consider asynchronous CAs leading to a certain spatial order. As an example we will discuss pattern formations in 2-dimensional CAs defined on a regular lattice, and a genetic algorithm will be applied to searching, automatically, for pattern-forming CAs. Techniques to search for CAs using a genetic algorithm were exclusively studied for the collective behaviors such as a density task[16, 10], a classification[6], and a period three behavior[9]. However, few studies have been done for searching for spatial pattern formations. For this purpose, we will consider a kind of interactive particle system (or a kind of lattice gas), in which the total number of non-zero cells is conserved. This mass-conservative condition is essential to the searching methodology, because this eliminates CAs which unnaturally generate and annihilate particles to form spatial patterns; such CAs are meaningless for the study of pattern-forming mechanisms.

In the previous work [14], three types of pattern-forming CAs, *ckeckerboard*, *stripe*, and *sand-like*, were found (See Fig. 1). This paper describes a follow-up study that statistically analyzes the rules found by the above methodology hoping that we can obtain some fundamental mechanism related to pattern formations. The analyses in this paper are focused on the checkerboard type out of the patterns mentioned above; this pattern is already discussed in [14] in depth from the phase transition's point of view.

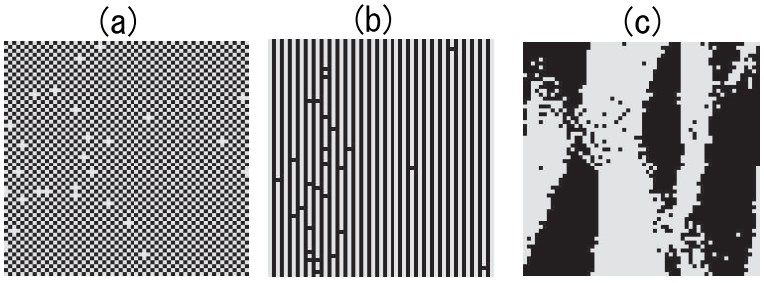

Fig. 1. The three kinds of patterns found by using genetic algorithms: (a)checkerboard, (b)stripe, and (c)sand-like.

In Sections 2 and 3, the mathematical structure of the CA applied to this study and the methodology to search for pattern-forming rules are explained,

respectively. Section 4 shows the results of the statistical analyses, and some related discussions are given. The conclusion is given in Section 5.

2 Asynchronous Cellular Automata with Mass Conservation

Cellular automata are dynamical systems in which the current state of each cell is determined by the states of its neighboring cells at the previous time step. We consider, in this paper, an interactive particle system modeled by a CA having two kinds of cells, occupied and empty cells. In common cellular automaton schemes, each cell has its own neighborhood and its state is determined regardless of what states its neighboring cells will take. In other words, the update rule of a cell is a many-to-one mapping, which is why the total mass is not conserved in such schemes. One technique employed to realize mass-conservative dynamics within CA is to divide the whole cellular space into many blocks composed of several cells with the states of each block being updated with the total mass in the block conserved. In this case the rule becomes a many-to-many mapping. This technique is called partitioning, and such a rule is called a *block rule*. Margolus neighborhood is a typical method of partitioning and has been applied to various CAs with the above conservation laws [15].

While the partitioning technique has been applied mainly to synchronous CAs, for example Fredkin's Billiard Ball Machine[5], we will, in this study, apply the technique to asynchronous updating schemes: The position of the block, composed of several adjacent cells to be updated, is randomly selected and the states of cells in the selected block are updated while the remaining cells are left unchanged. From now on, we will simply call this block 'neighborhood'. The rule applied to neighborhood is constant over time, just as it is for the common synchronous scheme. See Fig. 2 for a schematic picture of the asynchronous partitioning in the case of von Neumann-type neighborhood being adopted. We will concentrate on this type of the neighborhood throughout this paper.

Because the influence of a single update to the dynamics is much smaller than that of an update in the common synchronous method, the unit time is newly defined as being composed of $\frac{x_{max} \cdot y_{max}}{N_{neighbor}}$ asynchronous updates, where $N_{neighbor}$ is the total number of neighbor cells, i.e. five for the von Neumann neighborhood; $x_{max}(= 100)$ and $y_{max}(= 100)$ are the dimensions of the cellular space. With this new definition, a cell is, on average, updated once in a unit time step, which is consistent with the common synchronous updating schemes. Note that some cells may not be updated in a unit time step, while others may be done more than once.

We adopt, in this study, a commonly used torus-type boundary condition in which the top and left edges of the cellular space are connected to the bottom and right ones, respectively. The state of each cell is assumed to take an integer equal to either 0 or 1. The '0' state denotes an empty cell and is depicted in our graphs as a white cell, while the other state is one occupied by a particle and is

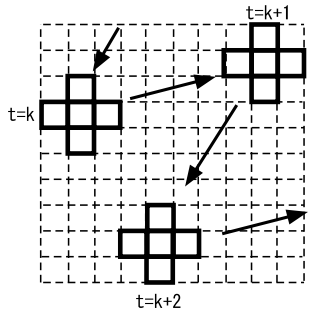

Fig. 2. Asynchronous partitioning cellular automata: A set of 5 neighbpring cells, randomly selected, are updated in the order indicated by the arrows.

depicted in our graphs as a black cell; i.e. the dynamical system is regarded as a mono-particle artificial chemical system or a kind of lattice-gas system. The particles merely move around the cellular space neither separating into two new pieces, nor merging with others to form a new particle. To satisfy this mass conservation law, the update rule must be defined such that the total number of particles in neighborhood does not vary.

The initial condition of each run is given by placing particles randomly on the cellular space with a specified "density", which is the ratio of the number of particles to the total number of cells in the cellular space. We here set the density at 0.5. For the random number generations, a unique randomizing seed is used for each space tried, ensuring that the initial configurations are always different.

3 A Methodology to Search Automatically for Pattern-Forming Cellular Automata

We, in this paper, measure the degree of pattern formation by the spatial entropy of global cellular states. Thus, finding pattern-forming CAs means finding those with the spatial entropy being relatively small. The space of CA rules is discrete and two closely similar rules do not necessarily produce a similar development. The optimization of cellular automaton rules therefore tends to become a messy problem; classical tools such as steepest descent method are not applicable. Genetic algorithms are, however, useful for problems in which the function to be minimized (or maximized) has a rugged surface, and we apply this technique to our problem as described below.

Each rule in consideration is composed of $2^{N_{neighbor}} = 32$ mappings. The graphic patterns of all the 32 entries of the mapping are shown in Fig. 3. Because such graphic patterns can be indexed by integers, the rules of our interest can be represented by the array of 32 integers, say $T[i](i = 0, 1, \ldots, 31)$. This integer array becomes the chromosome of this genetic algorithm. The value of

$T[i]$, i.e. the output of i'th entry assumes an integer between 0 and 31, but as described above each transition must satisfy the mass-conservative condition, and the transitions changing the mass inside the neighborhood are prohibited; then the whole rule space is approximately composed of $\sim 10^{14}$ rules, which is much smaller than that of the general rule space of 32^{32} rules.

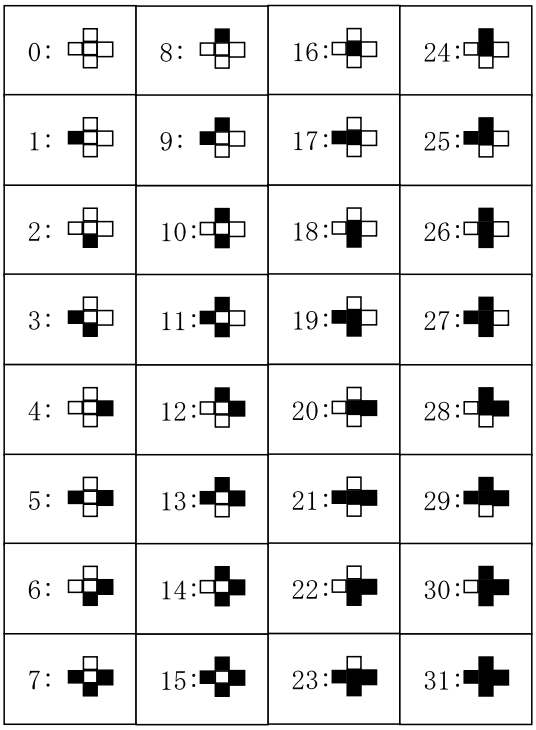

Fig. 3. All the entries of the transitions of the cellular automata.

The algorithm of the searching is:

1. Choose M chromosomes, i.e. mass-conservative cellular automaton rules at random, where M is called a *population*.
2. Run all of these CAs.
3. If at least one of the CAs satisfies a condition of being optimized (see below for the detail of the condition), stop the program; continue otherwise.
4. Calculate the fitness functions (given below) of all the CAs.
5. Choose the $M_e (0 < M_e < M)$ best-fit CAs for the next generation; they are called *elites*.
6. Choose M_n pairs of individuals at random with the probability proportional to their fitness functions out of the remaining CAs for the next generation

Table 1. The parameters of the genetic algorithms

symbol	explanation	value
M	population	20
M_e	the total number of the elite	6
P_c	crossover rate	0.5
P_m	mutation rate	0.05
H_s^o	the target value of H_s	0.57

so that the total of the elite and non-elite individuals becomes M again in the next generation, i.e. $M = M_e + 2M_n$.

7. Cross over each pair of non-elites at a randomly chosen point with probabilities of P_c, crossover rate.
8. Mutate each non-elites with probabilities of P_m per a transition. Note that new transitions also satisfy the mass-conservative condition.
9. Go to Step 2.

The fitness function is defined as:

$$F = (\frac{1}{H_s(t_c)}) \cdot t_c, \tag{1}$$

where $H_s(t)$ is a spatial entropy of global cellular states at the time step t (A rigorous definition of H_s is given in Appendix A.); t_c is the transition time, that is, the time when the time development of H_s becomes almost steady. The length of t_c is determined by the sequential tests of

$$|H_s(t+1) - H_s(t)| < \epsilon, \tag{2}$$

$$|H_s(t+2) - H_s(t+1)| < \epsilon, \tag{3}$$

and

$$|H_s(t+3) - H_s(t+2)| < \epsilon. \tag{4}$$

The smallest t satisfying all of eqs.(2)-(4) was considered as t_c, where ϵ was empirically set at 0.003. In case a pattern formation occurs, the transition time (i.e. t_c) is long. For this reason, the multiplication by t_c in eq.(1) was added to a simple $\frac{1}{H_s(t_c)}$, and it was empirically found that this multiplication improved the optimization speed. The condition to stop the program is when at least one of the individual was found $H_s(t_c) < H_s^o$; where H_s^o is the target value of optimization of H_s. The parameters for running the genetic algorithm are summarized in Table 1.

4 Statistical Analyses of the Rules Found by the Genetic Algorithm

Many runs of the optimization trial by the genetic algorithm were conducted, and the rules that satisfied the condition were collected. The number of gener-

Table 2. The Shannon entropy of the selected transitions

entry	entropy	entry	entropy	entry	entropy	entry	entropy
0	0.0	8	0.61	16	0.0	24	0.60
1	0.52	9	0.98	17	0.60	25	0.98
2	0.61	10	0.99	18	0.60	26	1.00
3	0.98	11	0.60	19	0.99	27	0.60
4	0.61	12	0.98	20	0.60	28	0.98
5	0.99	13	0.60	21	0.99	29	0.58
6	0.98	14	0.60	22	0.99	30	0.54
7	0.59	15	0.0	23	0.67	31	0.0

ations needed to reach the condition of being optimized is sometimes less than a hundred, but a few other cases reached 300 generations at which the program gave up the further optimization and went to a next trial. The spatial patterns that the rules found by the searching give are the same as those in [14], i.e. , (a)checkerboard, (b)stripe, and (c)sand-like patterns (see Fig.1). In the following, we will concentrate on the analyses only of the rules that develop the checkerboard pattern.

The total number of the collected checkerboard rules is ~ 1000. For these rules, the probability of what transition is selected through the optimization process, say p_{ij}, were calculated, where i and j are indices for the entry and the output of the transition, respectively. Using p_{ij}, the Shannon entropy of p_{ij} for each entry i was calculated; the results are shown in Table 2.

From the entropy values, it is possible to classify each entry into 4 groups:

I Entry-0 and Entry-31 (The entropy is exactly 0; each entry has only one possible transition because of the mass-conservative restriction.),

II Entry-15 and Entry-16 (The entropy is exactly 0; the entry pattern already matches the checkerboard pattern and the identical mapping is selected.),

III Entry-3, Entry-5, Entry-6, Entry-9, Entry-10, Entry-12, Entry-19, Entry-21, Entry-22, Entry-25, Entry-26, and Entry-28 (The entropy is almost 1, thus the transition is almost randomly selected. From Fig. 3, we can recognize that these entries are not relevant to produce the checkerboard pattern.), and

IV The remaining entries (The entropy is in the range of $0.52-0.67$; The further analysis will be given below.).

The key point of the above grouping is whether or not the entry is one of the matching patterns of which the checkerboard pattern is composed, that is, 15th and 16th graphic patterns in Fig. 3.

The most-frequently selected transition for each entry and its probability, $\sup_i p_{ij}$, were also calculated (See Table 3). From Table 3, we notice that Group IV can be classified into two subgroups:

Table 3. Transition most-frequently selected by the optimization; the values in the parenthesis is its probability (i.e. . $\sup_i p_{ij}$.)

$0 \to 0$ (1.0)	$8 \to 16$ (0.72)	$16 \to 16$ (1.0)	$24 \to 24$ (0.32)
$1 \to 16$ (0.77)	$9 \to 9$ (0.17)	$17 \to 17$ (0.31)	$25 \to 25$ (0.16)
$2 \to 16$ (0.72)	$10 \to 10$ (0.14)	$18 \to 18$ (0.31)	$26 \to 26$ (0.14)
$3 \to 3$ (0.15)	$11 \to 11$ (0.28)	$19 \to 19$ (0.14)	$27 \to 15$ (0.73)
$4 \to 16$ (0.72)	$12 \to 12$ (0.14)	$20 \to 20$ (0.31)	$28 \to 28$ (0.15)
$5 \to 5$ (0.14)	$13 \to 13$ (0.31)	$21 \to 22$ (0.14)	$29 \to 15$ (0.74)
$6 \to 6$ (0.18)	$14 \to 14$ (0.31)	$22 \to 22$ (0.15)	$30 \to 15$ (0.77)
$7 \to 7$ (0.31)	$15 \to 15$ (1.0)	$23 \to 15$ (0.68)	$31 \to 31$ (1.0)

IVa Entry-1, Entry-2, Entry-4, Entry-8, Entry-23, Entry-27, Entry-29, and Entry-30 (The value of $\sup_i p_{ij}$ is around $0.65 - 0.8$; The patterns for these entries are possible to transit to either of the two matching patterns of which the checkerboard pattern is composed, and such transitions are most-frequently selected.), and

IVb Entry-7, Entry-11, Entry-13, Entry-14, Entry-17, Entry-18, Entry-20, and Entry-24 (The value of $\sup_i p_{ij}$ is ~ 0.3; The patterns of these entries are close to one of the matching patterns, but the transition to them are prohibited under the mass-conservative condition; the identical mappings in these entries are most-frequently selected.).

Through the above classification, we seem to understand how each transition is selected to achieve the checkerboard pattern. Natural question is then; "Is this all about the tricks of the pattern formation?". If so, a rule created by putting all the most-frequently selected transitions of Table 3 together must have a performance of making the checkerboard pattern as good as that given by the "raw" rules obtained by the optimization. Figure 4 gives the time development of the CA composed of the most-frequently selected transitions. The graph at $t = 25$ has the spatial entropy of ~ 0.69, much larger than 0.57, the optimization target. The configuration of the cellular states at $t = 25$ (Fig. 4b)is almost identical to that at $t = 50$ (Fig. 4c), and the spatial entropy of the cellular states are not improved even in the further time-development. Such a poor performance does not meaningfully vary with the initial condition of the cellular states, so this rule's performance cannot compare with that of the rule composed of a raw set of transitions found by the searching. See Fig. 1(a); this graph is a cellular states developed by a typical rule found by the searching.

Although the statistical analyses given above seem theoretically sound, these are not enough to elucidate the whole mechanism of the pattern formation. In other words, the genetic algorithm selects each transition more cleverly than we can intuitively do. This result is not, however, surprising because complex phenomena such as pattern formations cannot be decomposed into a simple

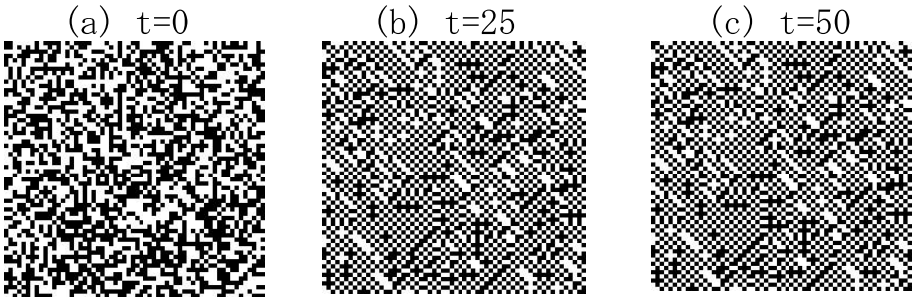

Fig. 4. The time development of the cellular automaton given by the rule in Table 3.

addition of many simple actions, but are organized by combinations or sometimes networks of them.

5 Conclusion

As a study of self-organization, the spatial pattern formation of CAs was studied. Especially, to seek the pattern formation more consistent to Nature, asynchronous CAs under the mass conservation law was explored and the asynchronous partitioning scheme was applied for this purpose. A methodology for automatically searching for the pattern-forming CAs was introduced, and its performance was confirmed. The methodology can be extended to more complicated situations for more realistic simulations of natural phenomena.

In addition, the rules giving the checkerboard pattern were collected by using the searching method for the purpose of their statistical analyses. The probability how often a certain transition is selected by the searching was discussed. It was possible to explain why certain transitions are dominantly selected by the searching, but the success of such analysis was limited because it failed to explain all the mechanisms of the pattern formation. This indicated that the checkerboard-pattern formation cannot be explained by a deduction of a simple addition of the mappings.

A further development of this study is to analyze the interaction between different transitions. For this purpose we perhaps need much more samples of the rules, and therefore it is necessary to drastically improve the searching speed. I hope that this extension could open a wide range of potential studies for approaching the origin of self-organization in Nature.

A Definition of spatial entropy of the global configuration of CAs

Consider four adjacent sites of the 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 position

of an arbitrary cell, respectively. There are $2^4 = 16$ possible patterns for this local patch if each cell takes either '0' or '1' state. In this paper, we use

$$H_s(\tau) \equiv -\sum_k P_s^k(\tau) log_{16}(P_s^k(\tau)), \tag{5}$$

as the definition of the spatial entropy of global configuration at an arbitrary time step τ, where $P_s^k(\tau)$ is the probability for a particular pattern of the local patch at the time step τ. Note that a base of 16 is used for the logarithmic function as the entropy assumes a real number between 0 and 1.

References

1. Dewdney, A.K.: Computer recreations. Sci. Amer. **August** (1988) 86–89
2. Dewdney, A.K.: Computer recreations. Sci. Amer. **August** (1989) 102–105
3. Di Paolo, E. A.: Searching for rhythms in asynchronous random boolean networks. Alife **VII**, The MIT Press (2000) 73–80
4. Di Paolo, E. A.: Rhythmic and non-rhythmic attractors in asynchronous random boolean networks. BioSystems **59(3)** (2001) 185–195
5. Fredkin, E., Toffoli, T.: Conservative logic. Int. J. Theor. Phys. **21** (1982) 219–253
6. Ganguly, N., et al.: Evolving cellular automata as pattern classifier. In S. Bandini, B. Chopart and M. Tomassini (Eds.), Cellular Automata, Lecture Notes in Computer Science **2493** (2002) 56–68
7. Harvey, I., Bossomaier, T.: Time out of joint: Attractors in asynchronous random boolean networks. Proceedings of the Fourth European Conference of Artificial Life, The MIT Press (1997) 67–75
8. Ingerson, T. E., Buvel, R. L.: Structure in asynchronous cellular automata. Physica D **10** (1984) 59–68
9. Jiménez-Morales, F.: An evolutionary approach to the study of non-trivial collective behavior in cellular automata. In S. Bandini, B. Chopart and M. Tomassini (Eds.), Cellular Automata, Lecture Notes in Computer Science **2493** (2002) 32–43
10. Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press (1996).
11. Nehaniv, C. L.: Self-reproduction in asynchronous cellular automata. University of Herfordshire, Faculty of Engineering and Information Sciences, Technical Report **368** (2002).
12. Schonfisch, B., de Roos, A.: Synchronous and asynchronous updating in cellular automata. BioSystems **51(3)** (1999) 123–143
13. Suzudo, T.: Crystallisation of 2-dimensional cellular automata. Complexity International **6** (1999) (online journal at http://www.complexity.org.au/).
14. Suzudo, T.: Spatial pattern formation of asynchronous cellular automata with mass conservation. Physica A in press
15. Toffoli, T., Margolus, N.: Cellular Automata Machines - A New Environment for Modeling. The MIT Press (1987)
16. Tomassini, M., Venzi, M.: Artificial evolved asynchronous cellular automata for the density task. In S. Bandini, B. Chopart and M. Tomassini (Eds.), Cellular Automata, Lecture Notes in Computer Science **2493** (2002) 44–55