Universal Criticality in Reservoir Computing Using Asynchronous Cellular Automata

Daisuke Uragami
Department of Mathematical Information Engineering
College of Industrial Technology
Nihon University, Chiba, Japan
uragami.daisuke@nihon-u.ac.jp

Yukio-Pegio Gunji
Department of Intermedia Art and Science
School of Fundamental Science and Engineering
Waseda University, Tokyo, Japan
yukio@waseda.jp

Elementary cellular automata (ECAs) generate critical spacetime patterns in a few local rules, which are expected to have advantages in reservoir computing (RC). However, previous studies have not revealed the advantages of critical spacetime patterns in RC. In this paper, we focus on the distractor's length in the time series data for learning and clarify the advantages of the critical spacetime patterns. Furthermore, we propose asynchronously tuned ECAs (AT_ECAs) to generate universally critical spacetime patterns in many local rules. Based on the results achieved in this study, we propose RC based on AT_ECAs. Moreover, we show that the universal criticality of AT_ECAs is effective for learning time series data.

Keywords: reservoir computing; cellular automata; edge of chaos; universal criticality; asynchronous updating

1. Introduction

Reservoir computing (RC) has recently attracted attention as a machine learning architecture for time series data [1–4]. RC is a recurrent neural network (RNN) based on a large-scale dynamical system called a reservoir. The learning method of the RC is characterized by fixed connections inside the reservoir and an adjustable relationship between the reservoir's state and the outputs. Consequently, the learning in RC is faster and more stable than the conventional RNN. Because the connection in the reservoir is fixed, any nonlinear dynamical system with a large degree of freedom can be used as a reservoir, which is one reason why RC is gaining attention from multiple

https://doi.org/10.25088/ComplexSystems.31.1.103

researchers [5–7]. Cellular automata (CAs) are a nonlinear dynamical system with a large degree of freedom, which can be used as reservoirs [8–11].

The reservoir must have a diversity of dynamics and stability for inputs to achieve an RC-based effective learning system [12]. Diversity of dynamics related to the storage capacity contributes to classifying and recording multiple instances of time series data. Moreover, the magnitude of change in the reservoir state for similar inputs should be small. Notably, from the viewpoint of maintaining memory, the reservoir dynamics should be stable for the inputs. In nonlinear dynamical systems, the diversity of dynamics is achieved by a chaotic mechanism. However, the stability for inputs is a property that is contrary to chaos. Therefore, it is generally difficult to achieve a system having both diversity and stability.

It is a critical state to include diversity and stability in a system. Generally, the critical state is achieved only in the extremely limited region of the order parameter, called the "edge of chaos," as shown in Figure 1 (top) [12–17]. However, such fine-tuning of order parameters is not desirable, considering the learning system's robustness. However, critical behavior is universally observed in biological systems. From the perspective of the edge of chaos, the universality of critical behavior in biological systems is a difficult phenomenon to explain. It is a mystery that should be called the "hard problem of life" [18].

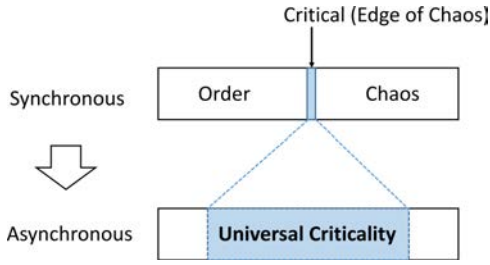

Figure 1. In a synchronous system, critical behavior appears only in the very narrow region of the order parameter, referred to as the edge of chaos (top). On the other hand, in an asynchronous system, critical behavior appears universally without depending on the order parameter (bottom).

Toward solving this problem, we focus on asynchrony in biological systems. We propose asynchronously tuned elementary cellular automata (AT_ECAs), which have been shown to achieve universally critical behavior [19–21]. The AT_ECAs generate critical spacetime patterns using a mechanism different from the edge of chaos, without the fine-tuning of order parameters (Figure 1, bottom). This property of AT_ECAs is expected to be useful as a reservoir.

In this paper, we propose a learning system that applies AT_ECAs to RC. Furthermore, we show that the universal criticality of AT_ECAs is effective in learning time series data. Section 2 introduces a learning system that uses the elementary cellular automaton (ECA) as a reservoir. In Section 3, we show effectiveness in learning specific time series data using criticality achieved by a few rules in elementary cellular automata (ECAs). We develop the learning system using AT_ECAs as a reservoir and identify its universal criticality effectiveness.

2. Reservoir Computing Using Elementary Cellular Automata

ECAs are one-dimensional, two-state, three-neighbor CAs [22, 23]. The state of the i^{th} cell at the time step t is expressed as $c_i(t)$; an ECA is defined as follows:

$$c_i(t) \in \{0, 1\} \tag{1}$$

$$f : \{0, 1\} \times \{0, 1\} \times \{0, 1\} \to \{0, 1\} \tag{2}$$

$$c_i(t + 1) = f\big(c_{i-1}(t), c_i(t), c_{i+1}(t)\big) \tag{3}$$

where the function f is called a local rule. Local rules determine the next state of a cell by referring to the states of the cell and the cells on both sides. The set of referenced states is called a neighboring state. There are eight patterns of neighboring states from 000 to 111. Therefore, the function f is determined by assigning 0 or 1 to $a_0 \sim a_7$ in the following:

$$\frac{000}{a_0}, \frac{001}{a_1}, \frac{010}{a_2}, \frac{011}{a_3}, \frac{100}{a_4}, \frac{101}{a_5}, \frac{110}{a_6}, \frac{111}{a_7}. \tag{4}$$

There are $2^8 = 256$ local rules, referred to as rule 0–255 by Wolfram's coding. For example, rule 110 is:

$$\frac{000}{0}, \frac{001}{1}, \frac{010}{1}, \frac{011}{1}, \frac{100}{0}, \frac{101}{1}, \frac{110}{1}, \frac{111}{0}. \tag{5}$$

In an ECA, the states of all cells are updated using the same local rule. The state $c_i(t)$ of each cell at $t > 0$ can be computed by setting the initial states $c_i(0)$ and repeatedly applying the local rule. It is called a spacetime pattern. Figure 2 shows the spacetime patterns of four typical local rules when the initial states are set randomly. The horizontal rows represent the state of each cell in a specific time step, and the vertical direction represents the time, that is, the transition of the state of

https://doi.org/10.25088/ComplexSystems.31.1.103

each cell (white means $c_i(t) = 0$, black means $c_i(t) = 1$), as shown in Figure 2. The spacetime patterns are classified into four classes by Wolfram [22, 23]. The spacetime pattern of rule 250 is uniform and classified as class 1. The spacetime pattern of rule 156 is periodic and classified as class 2. The spacetime pattern of rule 150 is chaotic and classified as class 3. The spacetime pattern of rule 110 is critical and classified as class 4. Among the 256 local ECA rules, few generate critical spacetime patterns, such as rules 110 and 54 and their symmetric local rules, representing the idea of the edge of chaos [15].

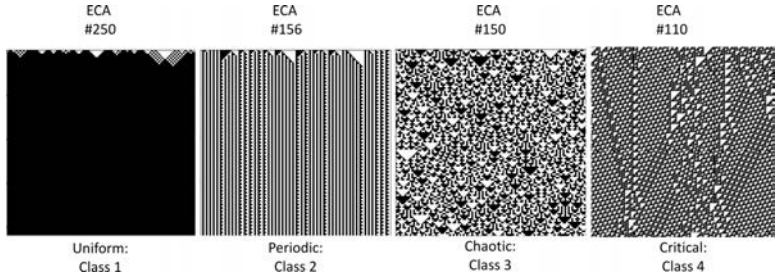

Figure 2. ECA spacetime patterns are classified into four classes. From the left, uniform (class 1), periodic (class 2), chaotic (class 3), critical (class 4). There are very few ECA local rules that generate Class 4 spacetime patterns.

RC is a type of RNN based on a nonlinear, large-scale dynamical system called a "reservoir" [1–4]. In RC learning, the weights of the connections in the reservoir are fixed, and only the correspondence between the reservoir and the output is tuned. Learning with RC is simple; therefore, it is faster and more stable than conventional recurrent neural networks (RNNs). Furthermore, any nonlinear dynamical system with a large degree of freedom can be used as a reservoir. Research is underway to utilize physical systems such as water stored in buckets and biological systems such as octopuses' legs as reservoirs [5–7].

ECA-based RC was proposed by Yilmaz [8] and improved to a recurrent method by Nichele and Gundersen [10]. In this paper, we use the latter method. The schematic diagram is shown in Figure 3. The data to be learned is time series data of multiple bits (the input is four bits and the output is three bits). First, the input data is copied into a one-dimensional cell via random connections, the initial state. A spacetime pattern is then generated using an ECA local rule, forming the reservoir state corresponding to the input. The time series data is not fed into the system all at once. Furthermore, the system calculates the output from the input of each time step using a recurrent method.

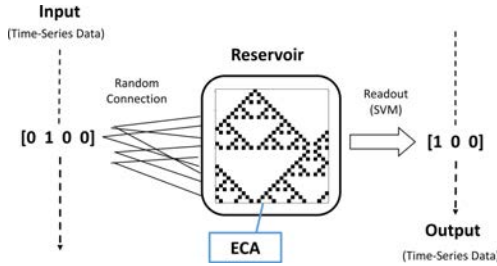

Figure 3. Schematic diagram of ECA-based RC. The input data is copied to the reservoir through random connections. The reservoir classifies and stores the input data. The substance of the reservoir is the ECA spacetime pattern. A support vector machine (SVM) learns the correspondence between the state of the reservoir (spacetime pattern of an ECA) and the output data.

Figure 4 shows the connections between the inputs and the reservoir and the recurrent connections. First, we prepare L cells in state 0 and randomly copy the first input data into those cells. Repeat it R times, and set them in an initial state. The spacetime pattern for I steps is generated from the initial state using an ECA local rule. The vectorized spacetime pattern is the reservoir state for the first input data. Therefore, the reservoir state is a vector consisting of $L \times R \times I$ elements, and the value of each element is 0 or 1 ($L = 20$, $R = 2$, $I = 4$ in the example shown in Figure 4). The last computed cell state is copied in the next time step, and the next time inputs are overwritten. The cell to be overwritten is the randomly determined cell in the

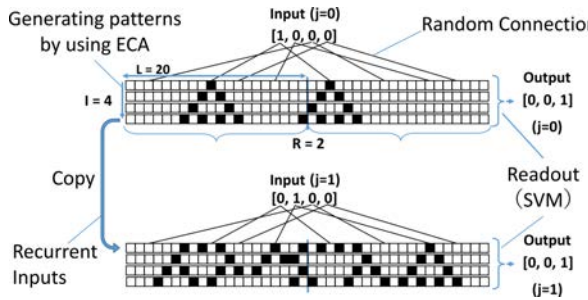

Figure 4. Details of the connections between the inputs and the reservoir. The first ($j = 0$) inputs are copied as cell states through random connections. As the initial state, a spacetime pattern generated based on the EAC's local rule is the reservoir state at $j = 0$. Next, the last state of the previous spacetime pattern is copied, and the input data at $j = 1$ is overwritten. With the new initial state, the spacetime pattern is generated again, which is the reservoir state at $j = 1$.

first time step. Subsequently, the ECA local rule is used to generate an I-step spacetime pattern, forming the reservoir state for the next input data. By repeating the process, the time series data is classified and stored as the reservoir state.

The learning system is defined as follows:

$$X_j = g(\boldsymbol{x}_j, X_{j-1}) \tag{6}$$

$$\boldsymbol{u}_j = h(X_j), \tag{7}$$

where $\boldsymbol{x}_j$ is the j^{th} input of the time series data, X_j is the corresponding reservoir state and $\boldsymbol{u}_j$ is the output of the learning system. The function g in equation (6) represents a series of procedures that copy the input data $\boldsymbol{x}_j$ into cells and generate a spacetime pattern. Since the procedure includes recurrent connections, the function g has the previous reservoir state X_{j-1} as an argument. The function h in equation (7) represents the procedure for determining the output from the state of the reservoir by the readout layer. When the output of time series data is written as $\boldsymbol{y}_j$, the readout layer is adjusted so that $\boldsymbol{u}_j$ matches $\boldsymbol{y}_j$ as much as possible. In this paper, as in [10], the readout layer uses a linear support vector machine (SVM) and is implemented using Python's scikit-learn. In the simulation, the parameters representing the reservoir's size were set as follows: $L = 20$, $R = 8$ and $I = 4$.

3. Criticality in Reservoir Computing Using Elementary Cellular Automata

In this section, we show that the ECA-based RC has higher learning ability when using the class 4 critical local rules than when using other local rules, classes 1, 2 and 3. Generally, the reservoir state should be critical because the reservoir requires dynamics diversity and input stability. On the one hand, there should be various dynamics (chaotic) considering memory capacity. On the other hand, it should be stable for input (not chaotic), considering memory maintenance. Consequently, the state where the balance between the two is maintained is the critical state.

Previous research has shown that the critical state achieved higher performance when using the Boolean network as a reservoir [12]. A CA-based reservoir with a spacetime pattern belonging to class 4 (critical) is expected to achieve higher performance. However, it has not been proven [8–10]. In the previous studies, the ECA local rules with

high learning ability are rules 60, 90 and 150, all of which belong to class 3 (chaotic). Conversely, in this paper, by investigating the influence of the distractor's length included in the time series data for learning, we show that learning ability is higher when the class 4 (critical) local rules such as rules 54 and 110 are used as reservoirs.

The subject of learning in this paper is the "five-bit task," which is time series data, as shown in Figure 5 [8, 10]. The input is 4-bit time series data, and the first five steps of the input are learning patterns. In the learning pattern, only the first or the second bit from the left is 1, and the other bits are 0. Therefore, there are $2^5 = 32$ possible learning patterns in total. The pattern input after the learning pattern is the distractor. This pattern continues for some length with an input in which only the third bit from the left is 1. The length is expressed as T_d. The next input after the distractor is the cue, in which only the fourth bit from the left is 1. After the cue, the input in which only the third bit from the left is 1 continues for five steps. Conversely, the output is 3-bit time series data. The outputs before the cue are the same, and only the third bit from the left is 1. The output after the cue is a pattern in which only the first or second bit from the left is 1, and the pattern is the same as the learning pattern. The time series data is not fed into the learning process simultaneously; four bits are fed as input to each step's learning system. Therefore, in this learning task, it is required to memorize the learning pattern step by step and keep it for the period during which the distractor is input. This learning task, which requires both short-term memory and long-term memory, is challenging for conventional RNNs [2].

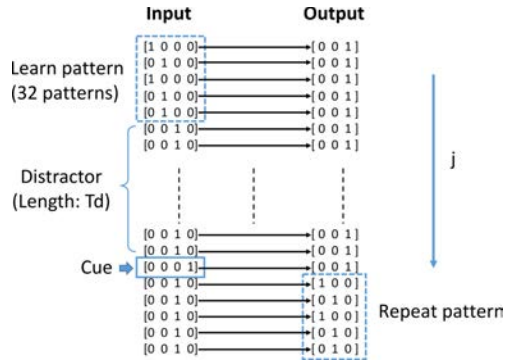

Figure 5. The five-bit task; learning data used in this study. The inputs are 4-bit time series data, and the outputs are 3-bit time series data. The learning task is to reproduce the learning patterns after passing through the distractor.

We used the five-bit task to evaluate the learning ability of the ECA-based RC. In this paper, we clarify the advantage of critical local rules by paying attention to the effect of distractor length.

https://doi.org/10.25088/ComplexSystems.31.1.103

Figure 6 shows the relationship between the distractor length T_d and the learning success rate. Here, the learning success rate is the ratio of successful learning out of 100 trials. One trial of learning is successful only if all 32 learning patterns can be repeated. The ECA local rules used as reservoirs are the five local rules, rules 54, 60, 90, 110 and 150. Rules 60, 90 and 150 are local rules that belong to class 3 (chaotic) and are considered to have high learning ability in previous studies [8–10]. Figure 6 shows that the learning success rate of these class 3 rules decreased significantly as the length of the distractor increased. Conversely, rules 54 and 110 are local rules belonging to class 4 (critical). We observe that the learning success rate remains almost one for these class 4 rules and seldom decreases as the distractor's length increases. Adding a very long distractor in the learning data reveals the advantage of the local rule of class 4 over class 3. This advantage has not been proven in previous studies and is one of the new results in this paper.

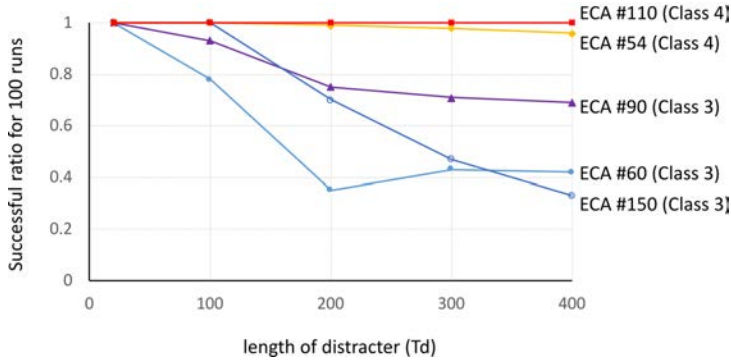

Figure 6. For RC using ECAs, the learning success rate is plotted against the length of the distractor (T_d). Utilizing class 3 (chaotic) local rules for the reservoir decreases the learning success rates as T_d increases (ECA rules 60, 90, 150). Conversely, utilizing the class 4 (critical) local rule did not significantly change success rates as T_d increases (ECA rules 54, 110).

It can be qualitatively understood that the critical local rule is superior as a reservoir for the reasons mentioned. For the five-bit task, various dynamics are required to identify the 32 learning patterns, and for that purpose, the state of the reservoir should be chaotic. Conversely, to retain the memory during the period when the distractor is input, the state of the reservoir should not change significantly. Both conditions are expected to be satisfied when class 4 local rules are used. In order to show this quantitatively, this paper proposes a new analysis index, as described in the following.

First, for the two states X_j and X_k of the reservoir with different time steps, $d(X_j, X_k)$ is defined as follows:

$$d(X_j, X_k) = \frac{|X_j - X_k|}{L \cdot R \cdot I}, \tag{8}$$

where $|X_j - X_k|$ is the number of cells contained in the reservoir that have different state values. $L \cdot R \cdot I$ is the total number of cells contained in the reservoir. Furthermore, $d(X_j, X_k)$ represents the distance between the two reservoir states X_j and X_k, where $0 \le d(X_j, X_k) \le 1$.

Second, using $d(X_j, X_k)$, we define the index $D(j)$, which indicates how different the reservoir state for the j^{th} input is from all states of the reservoir generated so far, as follows:

$$D(j) = \min_{0 \le k < j} \left(d(X_j, X_k) \right), \tag{9}$$

where $\min_{0 \le k < t} \left(d(X_j, X_k) \right)$ is the minimum value of $d(X_j, X_k)$ calculated for all X_k in the range of $0 \le k < j$. $D(j)$ will be a small value (close to 0) if any of all the generated reservoir states are similar to the current reservoir state; otherwise, it will be a large value (close to 1). Therefore, $D(j)$ approximately represents the distance between the current reservoir state X_j and the set of the previous reservoir states $\{X_k \mid 0 \le k < j\}$.

$D(j)$ was computed for RC using the local rules of ECA classes 1, 2, 3 and 4 as the reservoir. Figure 7 shows the $D(j)$ when one instance of the time series data of the five-bit task is input. The length of the distractor $T_d = 100$, and the total length of the time series data is $5 + 100 + 1 + 5 = 111$ $(0 \le j < 111)$. The graphs show the characteristics of classes 1, 2, 3 and 4. For rule 250 (class 1) and rule 156 (class 2), $D(j)$ decreased sharply and $D(j) = 0$ for the period when the distractor was input. Subsequently, $D(j)$ increased slightly because of the cue's input, and then returned to 0. These results show that the inputs do not affect the reservoir state, indicating they cannot distinguish learning patterns in classes 1 and 2. However, for class 3, $D(j)$ remained large when the distractor was input, indicating that the reservoir state is always new, which suggests that the memory of the learning pattern is lost.

https://doi.org/10.25088/ComplexSystems.31.1.103

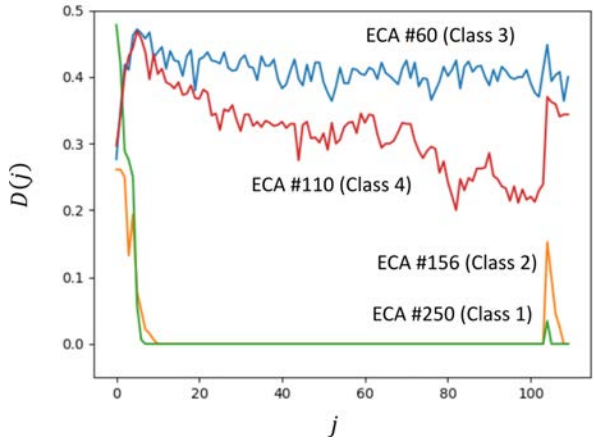

Figure 7. The transition of $D(j)$ when ECA classes 1 to 4 are used as reservoirs. While $D(j)$ of ECA rule 60 (class 3) is constantly changing at a large value, $D(j)$ of ECA rule 110 (class 4) falls once and then rises again at the end.

Conversely, rule 110 (class 4) has the characteristics of classes 1, 2 and 3. The $D(j)$ of rule 110 (class 4) decreased gradually while the distractor was input, but increased sharply after the cue was input. This result indicates that the reservoir state changed slightly when the distractor was input, which is beneficial for memory maintenance, while the state of the reservoir after the cue is input is highly variable and diverse. It is advantageous to distinguish and recall learning patterns. In other words, it can be confirmed from the transition of $D(j)$ that stability and diversity for input are achieved when the local rule of class 4 is used as a reservoir. Moreover, criticality is the realization of stability and diversity. Therefore, this numerical analysis shows that criticality is useful in RC using ECAs as a reservoir.

4. Asynchronously Tuned Elementary Cellular Automaton

The AT_ECA introduced the following three mechanisms into ECA [19, 20].

1. Local rules are applied asynchronously.
2. There are two types of local rules.
3. Local rules change at each cell.

First, asynchronous means that the state of a different time step may be referred to when updating the cell state. In asynchronous updating, the order in which the cell states are updated is significant.

If the order in which the i^{th} cell at time step t is updated is expressed as $\text{Ord}^t(i)$, it is randomly determined as long as the following equation is satisfied:

$$\text{Ord}^t(i) \in \{1, 2, \ldots, N\},\ i \neq s \Rightarrow \text{Ord}^t(i) \neq \text{Ord}^t(s), \tag{10}$$

where N is the total number of cells. If $\text{Ord}^t(i) = 1$, the i^{th} cell is updated first, and if $\text{Ord}^t(i) = N$, the i^{th} cell is updated last. Moreover, no cells are updated in the same order.

Second, based on the given order, we apply the local rules in the following formula to determine the cell states at the next time step:

$$\begin{aligned}&\text{Ord}^t(i-1) < \text{Ord}^t(i) < \text{Ord}^t(i+1) \Rightarrow \\ &\quad c_i(t+1) = f\big(c_{i-1}(t+1),\, c_i(t),\, c_{i+1}(t)\big)\end{aligned} \tag{11}$$

$$\begin{aligned}&\text{Ord}^t(i-1) > \text{Ord}^t(i) > \text{Ord}^t(i+1) \Rightarrow \\ &\quad c_i(t+1) = f\big(c_{i-1}(t),\, c_i(t),\, c_{i+1}(t+1)\big)\end{aligned} \tag{12}$$

$$\begin{aligned}&\text{Ord}^t(i-1) < \text{Ord}^t(i) > \text{Ord}^t(i+1) \Rightarrow \\ &\quad c_i(t+1) = f\big(c_{i-1}(t+1),\, c_i(t),\, c_{i+1}(t+1)\big)\end{aligned} \tag{13}$$

$$\begin{aligned}&\text{Ord}^t(i-1) > \text{Ord}^t(i) < \text{Ord}^t(i+1) \Rightarrow \\ &\quad c_i(t+1) = g_i^t\big(c_{i-1}(t),\, c_i(t),\, c_{i+1}(t)\big).\end{aligned} \tag{14}$$

Note that equations (11)–(14) apply different functions or local rules. It is the second mechanism that characterizes AT_ECAs. Let us call the functions f and g_i^t the passive rule and the active rule, respectively. The function f is the same as the conventional ECA local rule, which is defined by equation (4); subsequently, we explain the definition of the function g_i^t. The condition of equations (11)–(13) is that at least one cell on either side has already been updated. In this case, the passive rule f is applied. Here, the updated state is referred to for the cells that have already been updated. The condition of equation (14) is that the cells on both sides are yet to be updated. In this case, we use the active rules g_i^t to update the cell state.

The active rules g_i^t change in each cell in the process of generating the spacetime pattern, and as a result, various local rules are different for each cell. This is the third mechanism that characterizes AT_ECAs. The active rules g_i^t are defined as follows:

$$\frac{000}{e_{i,0}^t}, \frac{001}{e_{i,1}^t}, \frac{010}{e_{i,2}^t}, \frac{011}{e_{i,3}^t}, \frac{100}{e_{i,4}^t}, \frac{101}{e_{i,5}^t}, \frac{110}{e_{i,6}^t}, \frac{111}{e_{i,7}^t}, \tag{15}$$

https://doi.org/10.25088/ComplexSystems.31.1.103

where $e^t_{i,0} \sim e^t_{i,7}$ are 0 or 1. These are the same in all cells at time step $t = 0$, and match one of ECA's 256 rules. For all i,

$$e^0_{i,n} = a_n (n = 0, \ldots, 7), \tag{16}$$

where a_n is the same as in equation (4). After the next time step ($t > 0$), if equation (13) is applied for a cell in the previous time step, we change the active rule of the cell based on the following equation:

$$e^{t+1}_{i,m} = c_i(t+1) \text{ for } m = 4 \cdot c_{i-1}(t) + 2 \cdot c_i(t) + c_i(t) \tag{17}$$

$$e^{t+1}_{i,n} = e^t_{i,n} \text{ for } n \neq m. \tag{18}$$

We change the active rule to match the result of the last update. Changing the active rules this way has the effect of locally mitigating the discrepancy caused by asynchronous updating. As a result, AT_ECAs generate universally critical spacetime patterns. If equation (11) or equation (12) is applied, change it to $e^{t+1}_{i,m} = a_0$. Applying equation (14) does not change the active rule. These are the three mechanisms that characterize AT_ECAs.

The process of generating a spacetime pattern in an AT_ECA is described next. First, one local rule out of 256 ECA rules is selected as a passive rule. The active rules are initially the same as the passive rule and are the same in all cells. Furthermore, we set the initial state $c_i(0)$ appropriately. Second, the updating order $\mathrm{Ord}^0(i)$ at time step $t = 0$ is set (the update order is randomly set for each time step). Based on the updating order, one of the equations (11)–(14) is applied to determine the cell state $c_i(1)$ at the next time step, $t = 1$. At this time, the active rules are also changed, and the modified active rules are used in the next time step. By repeating the process, a spacetime pattern is generated. Since active rules are individually variable for each cell, there are various active rules in the process of generating a spacetime pattern.

Figure 8 shows the spacetime patterns of an AT_ECA and ECA. The local rule (passive rule) is rule 60, and the initial state is random. Although the spacetime pattern of the ECA is typically chaotic, AT_ECAs tend to grow branched patterns that are characteristic of critical spacetime patterns. This is not limited to the case where the passive rule is set to rule 60, but AT_ECAs generate critical spacetime patterns in many local rules, as shown in Figure 9. The local rules are rules 250, 156, 150, 110 (in ECAs, these local rules belong to classes 1 to 4, respectively, as shown in Figure 2). In all these rules, AT_ECAs generate critical spacetime patterns. Moreover, such a tendency is confirmed in the whole rule space [19, 20]. The spacetime

patterns generated by AT_ECAs have the characteristics of a critical system such as "1/f noise" [21].

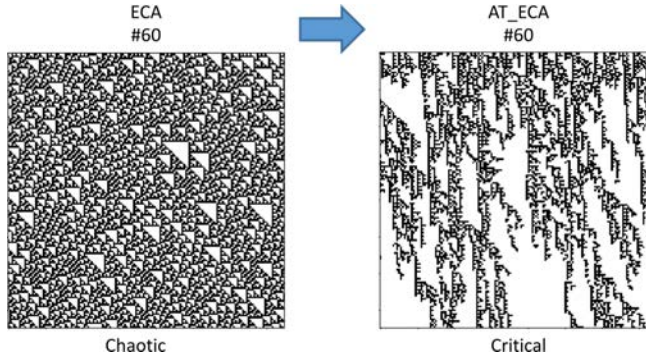

Figure 8. Comparison of spacetime patterns of an ECA and AT_ECA. Both are based on the same local rules (rule 60). The spacetime pattern of the ECA is chaotic, whereas the spacetime pattern of the AT_ECA is critical.

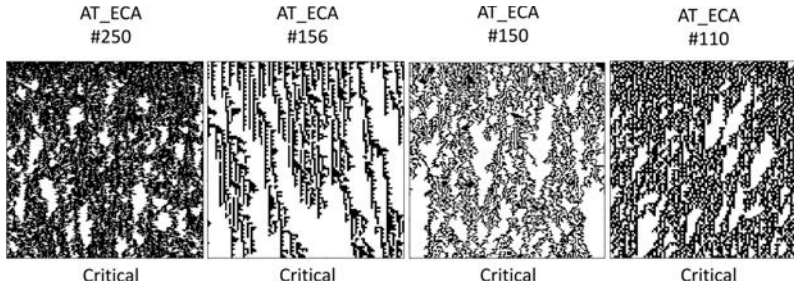

Figure 9. AT_ECA spacetime patterns. The local rules are the same as those shown in Figure 2, and they belong to classes 1 to 4 in ECAs, respectively. AT_ECAs generate critical spacetime patterns in all the classes.

5. Universal Criticality in Reservoir Computing Using Asynchronously Tuned Elementary Cellular Automata

This section shows that the criticality of AT_ECAs introduced in the previous section is useful in RC. The method of constructing AT_ECA-based RC is similar to the ECA-based RC explained in Section 3. The only difference is that AT_ECAs are used instead of ECAs when generating the spacetime pattern as the reservoir state. However, in order to guarantee the reproducibility of the spacetime patterns in the reservoir using AT_ECAs, the updating order $\mathrm{Ord}^t(i)$ randomly generated for $I = 4$ steps is repeatedly used in a trial.

First, we consider the effect of the distractor's length on the five-bit task. Figure 10 shows the relationship between the learning success

rate and the distractor's length when AT_ECA rules 60, 90 and 150 are used as reservoirs. For comparison, the results of using the same local rules in ECAs are also shown. The simulation conditions are the same as in Figure 6 in Section 3. Using AT_ECAs as a reservoir shows that the learning success rates are hardly reduced when the distractor's length is increased. This tendency is the same as that of ECA rule 110 (class 4) shown in Figure 6.

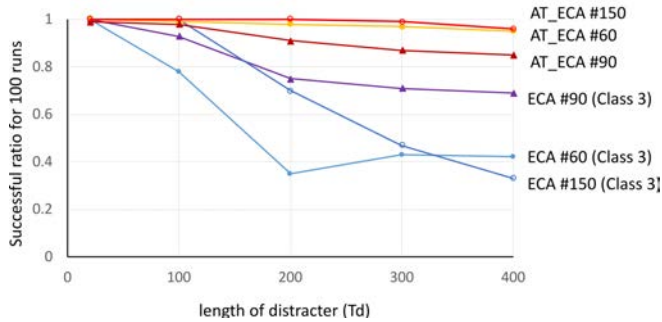

Figure 10. The relationship between the length of the distractor and the success rate of learning for RC using AT_ECA and ECA. For AT_ECA, the learning success rates seldom decreased based on the distractor's length. For ECA, the results are the same for the three local rules of class 3. See Figure 6 for comparison.

Second, for AT_ECA-based RC, the index $D(j)$ defined in equation (9) in the previous section is computed. Figure 11 shows the results of rules 33 and 60, which are local rules that belong to classes

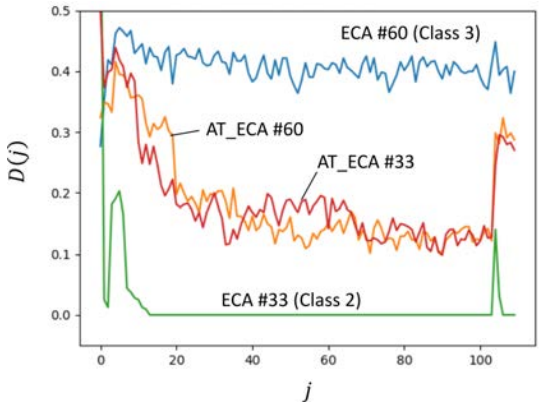

Figure 11. Comparison of $D(j)$ for an AT_ECA and ECA. For AT_ECA rules 33 and 60, $D(j)$ decreases once and then increases sharply at the end. This result is similar to ECA rule 110 (class 4) in Figure 7.

2 and 3 in ECA, respectively. In AT_ECAs with both rules, $D(j)$ goes down once and then goes up sharply after the cue is entered. Such an aspect is considered to be derived from the criticality of AT_ECAs, which is evident based on the comparison with Figure 7 in Section 3. This implies that the change in the reservoir state is small when the distractor is input, and the cue's input diversifies the state. Moreover, as discussed in Section 3, such a property is useful in this learning task.

Third, we show the tendency of the entire rule space to clarify the universal criticality in AT_ECA-based RC. As mentioned earlier, there are 256 local rules for ECAs, but considering the symmetry of 0 and 1 and the left and right symmetry, there are for practical purposes, 88 local rules [24]. For these 88 local rules, the learning ability when ECAs and AT_ECAs were used as reservoirs was verified. The learning task is the five-bit task. However, the length of the distractor was set to $T_d = 1000$. Learning simulations of 100 trials were conducted for each of the 88 local rules, and the results were aggregated using the "λ-parameter." The λ-parameter is defined as follows for the ECA local rule:

$$\lambda = \frac{\sum_{n=0}^{7} a_n}{8}, \tag{19}$$

where a_n is the same as that in equation (4). The λ-parameter takes a value from 0 to 1. However, 0.5 or more has a symmetric local rule, so 0.5 is in reality the maximum value. The λ-parameter is widely used as an order parameter that determines the behavior of ECAs and other CAs [15].

Figure 12 shows the average and maximum learning success rates calculated and plotted for local rules with the same λ-parameter value. However, both ECAs and AT_ECAs with a learning success rate of 0 are excluded from the average value. The ECA has a convex graph shape for both the average value (left figure) and the maximum value (right figure). For ECAs, the learning success rate increased when the λ-parameter was an intermediate value. This result represents the idea of the edge of chaos. Conversely, the shape of the AT_ECA graph is flat, and the learning success rate is universally high. In AT_ECA-based RC, the reservoir's state is critical without depending on the λ-parameter, referred to as universal criticality. Furthermore, the simulation result shows that high learning ability is universally achieved by universal criticality in AT_ECA-based RC. Such a learning system is considered to be robust. Furthermore, since most physical systems and biological systems are asynchronous in principle, the results are important when using them as reservoirs.

https://doi.org/10.25088/ComplexSystems.31.1.103

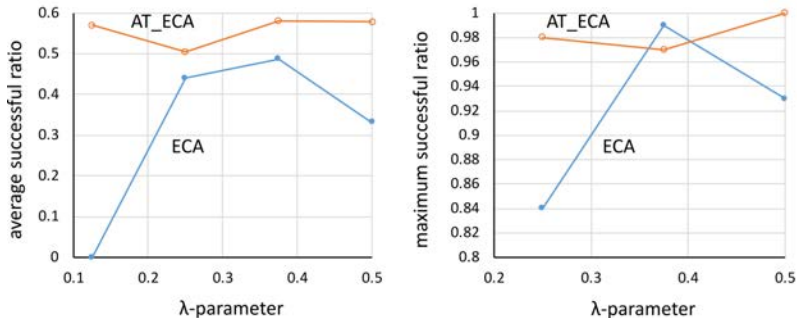

Figure 12. Relationship between λ-parameters and learning success rates. For the 88 independent local rules, the average and maximum values of the rules' learning success rate with the same λ-parameter value are calculated and plotted (the left figure is the average value and the right figure is the maximum value). For ECAs, the learning success rates are high only when the λ-parameters are intermediate. This phenomenon is called the "edge of chaos." Conversely, AT_ECAs have high learning success rates irrespective of the λ-parameters. This phenomenon result is called the "universal criticality."

6. Concluding Remarks

In this paper, we show that the universal criticality of the proposed asynchronously tuned elementary cellular automaton (AT_ECA) enhances the learning ability of reservoir computing (RC). In a conventional system, critical behavior is achieved only in a very limited area of order parameters. In this study, we show that in elementary cellular automaton (ECA)-based RC, critical behavior that enhances learning ability appears only with limited local rules commonly referred to as the edge of chaos. In contrast, this paper revealed that the edge of chaos is derived from synchronous updating. That is, AT_ECAs produce universally critical behavior because of asynchronous updating. This mechanism explains universally critical behavior observed in real biological systems.

In recent years, studies on open-ended evolution and computational universality in cellular automata (CAs) have advocated the need to remove the doctrine that certain special rules realize open-ended evolution or computational universality [25, 26]. Open-ended evolution and computational universality are closely related to the criticality that is the subject of this paper [27]. Our past study has also shown that changes in the interpretation of rules in the ECA enhance criticality [28]. The results of these studies reveal that state-dependent changing rules or compliers are more important than the specific rules themselves. The AT_ECA is a form of dynamic system

based on state-dependent changing rules, as local rules change due to the effects of asynchronous updating. Therefore, what we have shown in this paper is considered to be a strong support for what these studies have revealed.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP19K12143.

References

[1] H. Jaeger, "The 'Echo State' Approach to Analysing and Training Recurrent Neural Networks—With an Erratum Note," GMD Report 148, German National Research Center for Information Technology, 2010. www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf.

[2] H. Jaeger, "Long Short-Term Memory in Echo State Networks: Details of a Simulation Study," Technical Report 27, Jacobs University, Bremen, 2012.

[3] W. Maass, T. Natschläger and H. Markram, "Real-Time Computing without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, **14**(11), 2002 pp. 2531–2560. doi:10.1162/089976602760407955.

[4] M. Lukoševičius and H. Jaeger, "Reservoir Computing Approaches to Recurrent Neural Network Training," *Computer Science Review*, **3**(3), 2009 pp. 127–149. doi:10.1016/j.cosrev.2009.03.005.

[5] C. Fernando and S. Sojakka, "Pattern Recognition in a Bucket," in *Proceedings of Advances in Artificial Life (ECAL 2003)*, Dortmund, Germany (W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich and J. T. Kim, eds.), Berlin, Heidelberg: Springer, 2003 pp. 588–597. doi:10.1007/978-3-540-39432-7_63.

[6] B. Jones, D. Stekel, J. Rowe and C. Fernando, "Is There a Liquid State Machine in the Bacterium *Escherichia coli*?," in *The First IEEE Symposium on Artificial Life, (IEEE-ALife'07)* Honolulu, HI, Piscataway, NJ: IEEE, 2007 pp. 187–191. doi:10.1109/ALIFE.2007.367795.

[7] K. Nakajima, H. Hauser, T. Li and R. Pfeifer, "Exploiting the Dynamics of Soft Materials for Machine Learning," *Soft Robotics*, **5**(3), 2018 pp. 339–347. doi:10.1089/soro.2017.0075.

[8] O. Yilmaz, "Reservoir Computing Using Cellular Automata." arxiv.org/abs/1410.0162.

[9] E. T. Bye, "Investigation of Elementary Cellular Automata for Reservoir Computing," Master's thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, 2016.

[10] S. Nichele and M. S. Gundersen, "Reservoir Computing Using Nonuniform Binary Cellular Automata," *Complex Systems*, **26**(3), 2017 pp. 225–245. doi:10.25088/ComplexSystems.26.3.225.

[11] N. Babson and C. Teuscher, "Reservoir Computing with Complex Cellular Automata," *Complex Systems*, **28**(4), 2019 pp. 433–455. doi:10.25088/ComplexSystems.28.4.433.

[12] N. Bertschinger and T. Natschläger, "Real-Time Computation at the Edge of Chaos in Recurrent Neural Networks," *Neural Computation*, **16**(7), 2004 pp. 1413–1436. doi:10.1162/089976604323057443.

[13] S. A. Kauffman, "Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets," *Journal of Theoretical Biology*, **22**(3), 1969 pp. 437–467. doi:10.1016/0022-5193(69)90015-0.

[14] S. A. Kauffman and S. Johnsen, "Coevolution to the Edge of Chaos: Coupled Fitness Landscapes, Poised States, and Coevolutionary Avalanches," *Journal of Theoretical Evolution*, **149**(4), 1991 pp. 467–505. doi:10.1016/S0022-5193(05)80094-3.

[15] C. G. Langton, "Computation at the Edge of Chaos: Phase Transitions and Emergent Computation," *Physica D: Nonlinear Phenomena*, **42**(1–3), 1990 pp. 12–37. doi:10.1016/0167-2789(90)90064-V.

[16] D. R. Chialvo and P. Bak, "Learning from Mistakes," *Neuroscience*, **90**(4), 1999 pp. 1137–1148. doi:10.1016/S0306-4522(98)00472-2.

[17] T. Rohlf, N. Gulbahce and C. Teuscher, "Damage Spreading and Criticality in Finite Random Dynamical Networks," *Physical Review Letters*, **99**(24), 2007 248701. doi:10.1103/PhysRevLett.99.248701.

[18] S. I. Walker and P. C. W. Davies, "The 'Hard Problem' of Life," *From Matter to Life: Information and Causality* (S. I. Walker, P. C. W. Davies and G. F. R. Ellis, eds.), New York: Cambridge University Press, 2017 pp. 19–37. doi:10.1017/9781316584200.002.

[19] Y.-P. Gunji, "Self-Organized Criticality in Asynchronously Tuned Elementary Cellular Automata," *Complex Systems*, **23**(1), 2014 pp. 55–69. doi:10.25088/ComplexSystems.23.1.55.

[20] Y.-P. Gunji, "Extended Self Organised Criticality in Asynchronously Tuned Cellular Automata," *Chaos, Information Processing and Paradoxical Games: The Legacy of John S. Nicolis* (G. Nicolis and V. Basios, eds.), Hackensack, NJ: World Scientific, 2015 pp. 411–429. doi:10.1142/9789814602136_0021.

[21] D. Uragami and Y.-P. Gunji, "Universal Emergence of $1/f$ Noise in Asynchronously Tuned Elementary Cellular Automata," *Complex Systems*, **27**(4), 2018 pp. 399–414. doi:10.25088/ComplexSystems.27.4.399.

[22] S. Wolfram, "Universality and Complexity in Cellular Automata," *Physica D: Nonlinear Phenomena*, **10**(1–2), 1984 pp. 1–35. doi:10.1016/0167-2789(84)90245-8.

[23] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[24] W. Li and N. Packard, "The Structure of the Elementary Cellular Automata Rule Space," *Complex Systems*, **4**(3), 1990 pp. 281–297. complex-systems.com/pdf/04-3-3.pdf.

[25] A. Adams, H. Zenil, P. C. W. Davies and S. I. Walker, "Formal Definitions of Unbounded Evolution and Innovation Reveal Universal Mechanisms for Open-Ended Evolution in Dynamical Systems," *Scientific Reports*, **7**(1), 2017 997. doi:10.1038/s41598-017-00810-8.

[26] J. Riedel and H. Zenil, "Cross-Boundary Behavioural Reprogrammability Reveals Evidence of Pervasive Universality," *International Journal of Unconventional Computing*, **13**(4–5), 2018 pp. 309–357.

[27] Y.-P. Gunji and D. Uragami, "Breaking of the Trade-Off Principle between Computational Universality and Efficiency by Asynchronous Updating," *Entropy*, **22**(9), 2020 1049. doi:10.3390/e22091049.

[28] D. Uragami and Y.-P. Gunji, "Lattice-Driven Cellular Automata Implementing Local Semantics," *Physica D: Nonlinear Phenomena*, **237**(2), 2008 pp. 187–197. doi:10.1016/j.physd.2007.08.010.