entropy

MDPI

Article

Dynamics and Complexity of Computrons

Murat Erkurt

Department of Mathematics, Centre for Complexity Science, Imperial College London, South Kensington campus, London SW7 2AZ, UK; murat.erkurt09@imperial.ac.uk

Received: 29 October 2019; Accepted: 26 January 2020; Published: 27 January 2020

Abstract: We investigate chaoticity and complexity of a binary general network automata of finite size with external input which we call a *computron*. As a generalization of cellular automata, computrons can have non-uniform cell rules, non-regular cell connectivity and an external input. We show that any finite-state machine can be represented as a computron and develop two novel set-theoretic concepts: (i) *diversity space* as a metric space that captures similarity of configurations on a given graph and (ii) *basin complexity* as a measure of complexity of partitions of the diversity space. We use these concepts to quantify chaoticity of computrons' dynamics and the complexity of their basins of attraction. The theory is then extended into probabilistic machines where we define fuzzy basin partitioning of recurrent classes and introduce the concept of ergodic decomposition. A case study on 1D cyclic computron is provided with both deterministic and probabilistic versions.

Keywords: complexity; computron; cellular automata; general network automata; diversity measure; entanglement

1. Introduction

Complexity of dynamical systems and that of computational machines are mostly disconnected areas of research. The dynamical systems typically live in metric state spaces and evolve in time by iterating the rules of dynamics. The system is said to be complex if the information required to describe the evolving paths is high, a notion formally defined as Kolmogorov–Sinai entropy. On the other side of the world where computational machines live, the aim is to map a set of inputs to a set of outputs. We define *computron* as a computational machine that is composed of a finite number of cells placed on vertices of a directional graph. The cells have binary states which are updated according to local transition rules as a function of connected cells' states. Computron is the generalization of a cellular automata where local rules are not necessarily uniform, and graph not required to be a regular grid. Furthermore, a computron can have an external input, in which case it is called *driven*, otherwise *autonomous*. The computron's global state is the configuration of its cells' local states on its graph. It performs computation by mapping an initial configuration to an attractor, be it fixed or periodic by iterative dynamics. Such mapping partitions its configuration space.

We attempt to bridge disparate worlds of dynamical systems and computation by embedding computrons in a suitably defined metric space that captures similarity of its configurations. We develop notions of dynamical chaoticity using Lyapunov characterization of the machine's evolution and computational complexity by analyzing the basin partitioning of this proposed metric space. Our work contributes to literature in multiple ways:

1. Computron is introduced as a computational machine which is equivalent to a finite-state machine, but has the novelty that *flow of information* and *processing of information* is separated.
2. A new distance measure is introduced for configuration space on automata graphs. It is different than Hamming distance used in the literature since it captures pattern similarity of configurations on the underlying graph as opposed to solely counting mismatch between them.

3. A new complexity measure is proposed which captures the intrinsic structural complexity of a partitioning of the configuration space, not its randomness.
4. The above concepts are applied to quantify dynamics of generalized graph automata whereas literature is mainly focused on dynamics of cellular automata.
5. The framework is expanded to probabilistic automata using fuzzy basins and recurrent classes which enables us to quantify chaoticity and complexity of probabilistic machines, as well as their susceptibility to noise.

Section 2 provides background and literature review of Lyapunov characterization and entropy of dynamical systems including cellular automata. Section 3 defines the computron model. Section 4 introduces new set-theoretic concepts such as diversity measure and entanglement. Section 5 characterizes the dynamics of computron by embedding the machine into a metric space that captures similarity of configurations. Section 6 expands into probabilistic computation where each cell performs one of many rules probabilistically, driven by a random external input. This is a major step which pulls the formal domain into Markov chains and recurrent classes. Probabilistic computation lends itself to analysis of susceptibility of computrons, including effects of noise. Proofs of Lemmas introduced in various sections are provided in the Appendix A.

2. Background

In his *'A Philosophical Essay on Probabilities'* (1814), Laplace states that knowing the present and evolution laws gives full knowledge of the future. Yet, even if laws are truly known and are truly deterministic, the chaos could prevail, as described by Lorenz (1972): 'Chaos is when the present determines the future, but the approximate present does not approximately determine the future'. The sensitivity of future to present is formalized by Lyapunov characterization. The richness of what future may bring is quantified by Kolmogorov–Sinai entropy:

2.1. Lyapunov Characterization

Lyapunov exponents measure the typical rate of exponential divergence of nearby trajectories in a dynamical system. Consider the following continuous-time system:

$$\partial_t \mathbf{x} = \mathbf{F}(\mathbf{x}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ specifies the system state, and $\mathbf{F}$, a differentiable function, defines the evolution law. Let $\mathbf{x}(t)$ and $\mathbf{x}'(t)$ be two trajectories that start from close-by initial states $\mathbf{x}'(0) = \mathbf{x}(0) + \delta\mathbf{x}(0)$. The evolution of difference in trajectories, $\mathbf{z}(t) = \mathbf{x}'(t) - \mathbf{x}(t)$, can be approximated as a vector in tangent space:

$$\partial_t z_i = \sum_{j=1}^{d} \partial_{x_j} F_i|_{\mathbf{x}(t)} z_j(t) \tag{2}$$

and there exists an orthonormal basis $\{\mathbf{e}_i\}$ such that for large t, the difference in trajectories can be represented as exponential growth or decay:

$$\mathbf{z}(t) = \sum_{i=1}^{d} c_i \mathbf{e}_i e^{\lambda_i t} \tag{3}$$

The Lyapunov exponents $\{\lambda_i\}$ are independent of the initial state $\mathbf{x}(0)$ if the system is ergodic. Systems with a positive Lyapunov exponent are typically chaotic and error of prediction grows exponentially if the initial state is slightly off.

2.2. Kolmogorov–Sinai Entropy

If we were to discretize time and coarsen the state space into d-dimensional cubes of each side ϵ (indexing each cube with a symbol), then a trajectory of evolution could be approximated with a string of symbols based on the index of each cube visited at each time step. For a simple system with predictable trajectories, the information needed to represent the ensemble of such strings is less than that of a system where trajectories disperse rapidly. Let $K(\epsilon, t)$ be the total number of different strings generated by the system in t steps in a coarse-grained state space. The *topological entropy* of the system is defined as:

$$h = \lim_{\epsilon \to 0} \lim_{t \to \infty} \frac{1}{t} K(\epsilon, t) \tag{4}$$

Please note that in this definition, each different string generated by the system is counted as one. If we take into account the frequency of observing a given finite string of length T (a word w^T), and apply the Shannon entropy formulation to this stream of words, we get the information content of T-symbol paths generated by the system as:

$$H^T = -\sum_{\{w^T\}} P(w^T) \log P(w^T) \tag{5}$$

Kolmogorov–Sinai entropy is then defined as the average information content per time step of the path, chosen as the highest one over all possible coarse graining partitions of state space $\{\mathcal{G}\}$:

$$h^{KS} = \sup_{\mathcal{G}} \lim_{T \to \infty} \frac{1}{T} H^T \tag{6}$$

2.3. Basin Entropy

Dynamical systems theory has traditionally been *path-centric*. Chaoticity is described as divergence of close-by paths, and entropy is conceptualized as richness of paths emanating from a given present. Where paths end up in the long-term is called the *attractor* of a dynamical system. In systems with fixed-point or periodic attractors, future is rather dull. In chaotic systems however paths are entangled in such a way that bundles which are close-by soon end up far separated and the attractor is called *strange*. Some dynamical systems, and most computational automata, have multiple simple attractors. In these systems while the future is dull, present can be exciting if knowing the now gives little indication on which final simple destiny one ends up in. This is the *basin-centric* way of looking at the dynamics.

Consider a discrete-time dynamics $x \to x^2 - y^2 + a$ and $y \to 2xy + b$ with (a, b) system parameters. If one marks all possible initial states $x(0), y(0)$ according to two ultimate destiny scenarios (escape to infinity or not), we get the fractal Julia sets as given in Figure 1, and quite interesting ones for different system parameters (a, b). A basin of attraction is the set of initial states that end up in a given attractor. The *basin entropy* measures the entanglement of these basins.

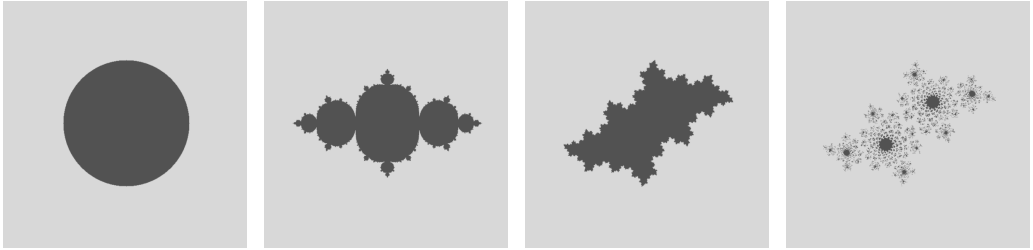

Figure 1. Basin partitioning in (*quadratic Julia sets*) with increasing basin entropy. System parameters (a, b) from left to right: $(0, 0)$, $(-0.75, 0)$, $(-0.35, -0.55)$, $(-0.40, -0.60)$. The last one tends to a disconnected Cantor set as time goes to infinity.

2.4. Entropy in Discrete State Space

A cellular automata ('CA') is a dynamical system that consists of a grid network of cells where each cell run the same look-up table-based evolution rule that tells it how to change its current state (which is drawn from a finite number of elements) into the next, given the neighbors' and its current state. The global state of CA is the configuration of local states of each cell on the grid. The local evolution rule and state information exchange between neighbors synchronously update all local states in each time step, changing the configuration of CA to its next, hence generating the dynamics on discrete configuration state space. A CA is called elementary if the grid is 1D, each cell takes binary state values and the neighborhood is just the left-right nearest cells.

Entropy for CA can be defined with the same spirit as Kolmogorov–Sinai entropy given above. Assuming a 1D grid, consider a window of L cells and ignore the rest. This is effectively a coarse graining of the configuration space since all configurations that have the same pattern within the window are grouped together. Now, let us look into strings of window patterns generated by the system in T time steps w_L^T with observation frequency $P(w_L^T)$. The entropy of this block (i.e., entropy of the system at this coarse graining) is:

$$H_L^T = -\sum_{\{w_L^T\}} P(w_L^T) \log P(w_L^T) \tag{7}$$

The equivalent Kolmogorov–Sinai entropy of CA is then given as the average information content per time step of the evolution path with no coarse graining, i.e., when the cell-space window is extended to full grid size:

$$h^{KS} = \lim_{L\to\infty} \frac{1}{L} \lim_{T\to\infty} \frac{1}{T} H_L^T \tag{8}$$

2.5. Dynamics of Cellular Automata

As per Equation (2), Lyapunov exponents rely on differentiability in state space. Since it is a Cantor set, the configuration space of CA does not lend itself to differentiability. Shereshevsky [1] in 1992, proposed the average propagation speed along the CA grid of a single cell disturbance as a proxy of CA's Lyapunov characterization. Shereshevsky showed that for ergodic 1D CA almost all configurations have unique left and right propagation velocities, λ^-, λ^+, and further established the inequality linking entropy of the CA to entropy of the shift operator σ and the propagation speeds as: $h(F) \leq h(\sigma)\,(\lambda^- + \lambda^+)$. Later in 2000, Tisseur [2] expanded the Sheresevsky's Lyapunov characterization by introducing average propagation velocities.

Entropy given in Equation (8) is finite and calculable for trivial CAs such as left-shifting CA. However, Hurd, Kari and Culik [3] (1992) proved that it is uncomputable for an arbitrary CA. The undecidability result does not preclude the existence of large classes of examples, such as *linear* CA, for which explicit calculations are possible. A CA is called linear if the local state set S is endowed with the sum and product operations that make it a commutative ring, and the local evolution function f is given as the linear combination of the neighboring cell's local states, $f(x_{-r}, \ldots, x_{+r}) = \sum_i a_i x_i$ with $\{a_i\} \in S$. With sum and product operations defined on S, it is possible to describe a given configuration c with characteristic power series $P_c(z) = \sum_{i=-\infty}^{\infty} c_i z^i$ and the local function as a kernel $A_f(z) = \sum_{i=-r}^{r} a_i z^i$. For linear CA dynamics reduces to $P_{F(c)}(z) = P_c(z) A_f(z)$. Manzini et al. published a series of articles [4–7] on topological dynamics of linear 1D CA and explicitly calculated the entropy.

Topological entropy is a characterization of the richness of time streams of configuration patterns generated by the CA. This richness is closely linked to compressibility of these pattern streams. Excellent progress has been made on practical implementation of compression algorithms for patterns. Lempel–Ziv coding [8] introduced in 1976 is a universal lossless data compression algorithm which is at the core of GIF image format used today. Zenil (2010) [9] proposed to calculate entropy of CA based on compressed length of output patterns with Lempel–Ziv-like compression.

2.6. Complexity Classification of Cellular Automata

In the early 1980s, Wolfram ignited research interest in CA with a series of articles [10–12] where he classified the apparent spatio-temporal configuration complexity as: (1) tends to a spatially homogeneous state; (2) yields a sequence of simple stable or periodic structures; (3) exhibits chaotic aperiodic behavior; and (4) yields complicated localized structures, some propagating, i.e., *complex patterns*. Wolfram's classification made a distinction of chaotic vs. complex configuration evolution. This was important because entropy-based definition used in dynamical systems theory did not differentiate chaotic behavior from complex. He conjectured that complex CA performed universal computation. Culik and Yu (1988) [13] proposed a formal framework for Wolfram's complexity classes. Key demarcation was whether an algorithm existed to decide for any two finite configurations whether one ever evolves to the other. Since then there has been a wide range of classifications on elementary CA to capture the intuition that complex was something different than chaotic.

Li and Packard (1990) [14] looked into the structure of *'rule space'* and investigated bifurcations in the behavior of 1D CA by varying a classifier parameter λ defined as the ratio of non-zero entries in local rule table of CA. The rule space seemed to be split into ordered (fixed or periodic patterns) and chaotic regions with a sharp phase boundary. However, in certain parts of the rule space, between the ordered and chaotic domains, the phase boundary was a thick wall that housed critical rules which showed complex patterns. Kurka (1997) [15] provided three formal ways of classification for 1D CA: (i) based on the complexity of languages generated by partitions of the state space; (ii) based on Lyapunov stability; and (iii) based on attractors of the configuration space. Kurka demonstrated correspondence between the classes of these three schemes and Wolfram's heuristic classification.

Chua et al. (2002) [16] proposed a complexity index to categorize the rules of elementary CA: Consider a cube graph where each vertex corresponds to one of $2 \times 2 \times 2 = 8$ states of the input triplet of a cell. Let us color each such vertex with Red or Blue depending on what the output state ought to be based on the local rule table. Now, imagine partitioning the cube by planes such that same colored vertices remain in the same partition. Chua's complexity index is the number of planes required to make such partition. He showed that Class IV elementary CA of Wolfram had index 2 or 3. To overcome certain mis-categorizations of Chua index, Ewert recently (2019) [17] proposed a novel method based on residual circuit complexity. It takes into account not only minimal form representation of the rule table, but also the reduction of input entropy as the system evolves which further reduces the needed minimal form. Applied to elementary CA, Ewert showed that the proposed method matched well to Wolfram's heuristic classification.

2.7. Venturing into General Network Automata

Very few attempted to go beyond classical cellular automata which has two built-in regularities: (i) cells are arranged on a regular grid-like cell space, and (ii) each cell executes the same local rule, i.e., homogeneous. These properties arguably give the classical CA a physical-like feel with uniform laws and regular space, and make dynamical analysis tractable. It is, however, tempting to go beyond classical set up and consider generalized graph automata where cells are placed at vertices of a general directional graph, and each executes a different local rule. Many real-life systems with interacting agents are typically organized into networks such as Watts–Strogatz type small-world graphs, or Barabasi–Albert type scale-free graphs with non-uniform local rules.

Marr and Hutt (2005) [18] investigated the pattern formation capacity of binary graph automata where each cell was executing a local law based on the average state value of the neighbors passed through a threshold function parametrized by value κ. They investigated both small-world and scale-free type graphs, varying the graph incrementally to see impact on pattern formation. They measured Shannon entropy of time strings of a single cell averaged over all cells. Since graphs were not regular grids, the authors did not define any spatial pattern concept, and investigated time patterns of single cells only.

Tomassini (2006) [19] analyzed the performance of binary graph automata with homogeneous cell rules over small-world and scale-free graphs over two tasks. In density classification task, the cells, starting from random configurations are expected to converge to all 1 if their initial 1 density were greater than 0.5 and vice-versa. In synchronization task, the cells starting from random initial configurations are expected to synchronize into a global flip-flop cycle. Cattaneo et al. (2009) [20] introduced two types of non-homogeneous 1D CA: (i) local rules are variable only in a region outside which the rule is uniform, and (ii) the rule is non-uniform everywhere but the neighborhood has always same radius. The authors showed that most topological properties get demolished when CA becomes non-homogeneous.

2.8. Back to Basins

Basin entropy defined in Section 2.3 is a relatively new concept and plays a central role in this paper. Daza et al. (2016) [21] calculated basin entropy of two well-known physical dynamical systems with path chaoticity: (i) Duffing oscillator which is a periodically driven rod with a nonlinear elasticity, and (ii) Hénon-Heiles system which describes motion of a star around a galactic center. They defined basin entropy by coarse graining the basin into boxes and marking distribution of ultimate destinies' probability in each box, to which they applied an entropy measure.

Finite-size discrete systems have been mostly dismissed in the entropy world since paths are of finite size and ultimately fixed or periodic. However, these systems as computational machines are very rich in the structure of their basins of attraction. In fact, mapping initial states to ultimate destinies is what computation means for these machines. As we shall now formally see, our definition of chaoticity and more importantly complexity of such finite-state machines is all about quantifying the structure of basin partitioning, and as such fundamentally differs from previous work on complexity and entropy for automata.

3. Computron

"Things do not exist other than in arrangements". Tractatus—Ludwig Wittgenstein

3.1. Construction

A computron $M = [G, C, v_{\text{IN}}, v_{\text{OUT}}]$ is an arrangement of N cells with binary states ω_i which is formally constructed with:

1. A *generating graph* $G(V, E)$ where each cell is assigned to a vertex $v_i \in V$. A cell v_j is said to be a neighbor of cell v_i if there exists the directional edge $e_{ji} \in E$. The ordered set of neighbor cells of v_i is called its neighborhood denoted as $\delta_i = [v_a, v_b, \dots]$. Each cell is assumed to be also its own neighbor, i.e., $e_{ii} \in E, \forall i$.
2. A set of *connectors* $C = \{c_i\}$, each assigned to a vertex. A connector $c_i(\Omega_i)$ is a binary valued look-up table based on the state configuration of neighbors, $\Omega_i = [\omega_j | v_j \in \delta_i]$.
3. An *input cell* at vertex v_{IN} which acts as interface for the external binary input σ, if the computron is driven with an external signal, $\omega_{\text{IN}} = \sigma$.
4. A nominated *output cell* at vertex v_{OUT} which provides a binary output γ if required, $\gamma = \omega_{\text{OUT}}$.

Figure 2 depicts the construction of a computron. The local states are updated synchronously. The global state of computron $s = [\omega_0, \omega_1, \dots, \omega_{N-1}]$ is the configuration of local states on the generating graph. Local dynamics induce the global dynamics of the computron as follows:

$$\begin{aligned} \omega_i^n \to \omega_i^{n+1} &= c_i(\Omega_i^n) \\ s^n \to s^{n+1} &= [\omega_0^{n+1}, \omega_1^{n+1}, \dots, \omega_{N-1}^{n+1}] =: T(s^n) \end{aligned} \tag{9}$$

An autonomous computron performs computation by mapping an initial configuration to an attractor (fixed or periodic). Such mapping partitions the configuration space. A driven computron generates an output string for a given input string based on an initial starting configuration.

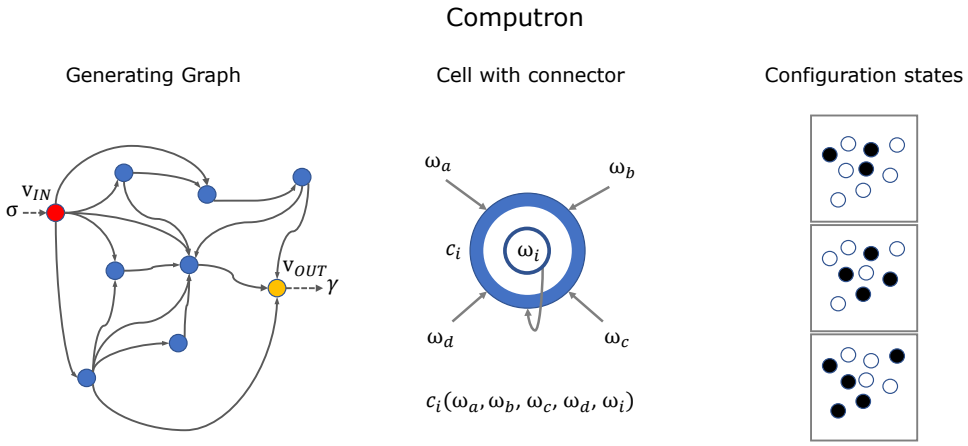

Figure 2. Computron is constructed on a generating graph. Cells are placed at vertices of the graph. Each cell has a binary state ω_i and is endowed with a local look-up table rule, called a connector. Global state of the computron is the configuration of local states on the generating graph. v_{IN} is the special cell that interfaces the external input and v_{OUT} is the nominated cell for reading the output.

3.2. Computron vs. CA and FSA

By construction, any finite-size cellular automata is a computron where generating graph is a regular grid, connectors are homogeneous, and there is no external input. Finite-state automata ('FSA') is a computational machine that maps an input string to an output one with no memory. Formally, the machine $M = [S, s_0, \Sigma, \Gamma, T, U]$ is specified by a set of states S, an initial state s_0, an input alphabet Σ, an output alphabet Γ, the *Transition Map* $T : S \times \Sigma \to S$, and the *Output Map* $U : S \times \Sigma \to \Gamma$. FSA generates an output string $y = \gamma_0\gamma_1\gamma_2... \in \Gamma^*$ driven by the input string $x = \sigma_0\sigma_1\sigma_2... \in \Sigma^*$. Without loss of generality we assume that input and output alphabets are binary.

Lemma 1. *For any FSA with* 2^N *states there is an equivalent computron of N cells plus the input cell.*

Figure 3 shows the state transition diagram of 32-state machine and its equivalent computron representation with 5 cells. We note that state transition diagram of finite-state automata lacks intuitive description of the inherent structure provided by the equivalent computron. The generating graph of the computron defines *'local flow of information'* whereas connectors of each unit describe *'local processing of information'*.

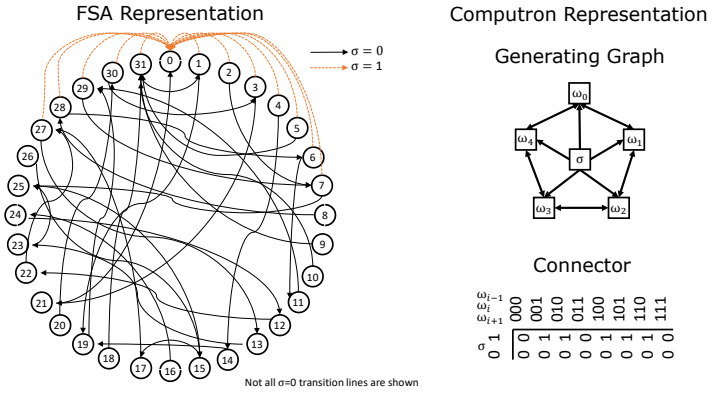

Figure 3. 32-state machine: Wolfram cellular automata Rule 110 on 5-circle and its equivalent computron with external input used as Reset (0) or Step (1)

4. Diversity Space

4.1. Purpose

Dynamical systems live on metric spaces whereas computation does not. Computron as a model of computation is constructed with global states defined as configurations of local states on its generating graph. In this section, we formally define a set measure called *diversity measure* which gives the cardinality of a set adjusted for similarity of its elements. Diversity measure induces a similarity distance between two sets as the measure of their symmetric difference. This similarity metric generates a metric space for the states of the computron called *diversity space*. We later perform all our dynamics investigations for the computron on this diversity space.

Various distance functions between sets with elements embedded in a metric space have been proposed in the literature. *Hamming* distance is arguably the simplest one which counts the number of different elements in two sets; it lacks the concept of similarity among the elements. *Hausdorff* distance between two sets is calculated as the furthest separation of one set's elements from closest of the other; final result reduces to distance between two most significant elements disregarding the rest.

$$d_h(S_1, S_2) = \max\{\max_{e \in S_1} \min_{f \in S_2} \Delta(e, f), \max_{e \in S_2} \min_{f \in S_1} \Delta(e, f)\} \tag{10}$$

The *sum-of-min-distances* incorporates all elements, but it is an aggregation of closest peers, hence presence of non-closest peers are shadowed, thus fails to capture an ensemble closeness.

$$d_m(S_1, S_2) = \frac{1}{2}(\sum_{e \in S_1} \Delta_m(e, S_2) + \sum_{e \in S_2} \Delta_m(e, S_1)) \tag{11}$$

Other distance measure between sets have been proposed to capture the overall similarity of the elements of two sets given the underlying metric space such as *Surjection* distance and *Link* distance [22]. Bennett, Vitanyi and Zurek et al. [23] introduced *absolute information distance* metric between two objects. Consistent with definition of absolute information content of an object as length of the shortest binary program $K(x)$ running on a universal machine, generating the given object x, they described the distance metric between two given objects x and y as length of the shortest binary program that generates one given the other: $d^*(x, y) = \max K(x|y) + \max K(y|x)$. They proved that absolute information distance is a universal pattern similarity metric. This theoretical metric is similar to that of algorithmic complexity in that while universal, it is not computable in its pure form.

4.2. Diversity Measure

We introduce *diversity measure* as the size of a set adjusted for the similarity of its elements. The intuition behind set-theoretic formulation below is as follows: Assume we are asked to form a *diverse team* from a pool of students. Each student has a profile which allows us to define how similar two students are. We build the team by adding one student at a time. First member contributes an *individuality* of 1. Second member's individuality is reduced by her similarity to the first member. Third member's is reduced by his similarity to the first two. Diversity of the team is sum of individualities.

Let (Q, f) be a metric space where Q is a finite set and f is a metric of Q, i.e., a function $f : Q \times Q \to \mathbb{R}^{0+}$ such that for any $x, y, z \in Q$, (i) $f(x, y) = 0 \Leftrightarrow x = y$, (ii) $f(x, y) = f(y, x)$, and (iii) $f(x, z) \leq f(x, y) + f(y, z)$. Let $\rho : \mathbb{R}^{0+} \to [0, 1]$ be a concave monotonic function with $\rho(0) = 0$ (*similarity kernel*) and define $\rho(x, y) = \rho(f(x, y))$, then (Q, ρ) is also a metric space.

Individuality of an element x given a set of constraining elements A is defined as:

$$d(x|A) = \prod_{y \in A} \rho(x, y) \tag{12}$$

if $\rho(x, y) = 0$, i.e., there exists an element $y \in A$ that is exactly the same as x, then $d(x|A) = 0$, meaning x has no individuality given the constraining set A.

Diversity of an ordered set $\vec{A} = [x_0, x_1, x_2 \dots]$ given a set of constraining elements, B is defined as:

$$\begin{aligned} d(\vec{A}|B) &= d(x_0|B) \\ &+ d(x_1|\{x_0\} \cup B) \\ &+ d(x_2|\{x_0, x_1\} \cup B) \dots \end{aligned} \tag{13}$$

Clearly, the ordering of the tuple $\vec{A}$ matters for the calculation of diversity, therefore we define the *diversity measure* of a set A constrained by set B as the supremum over all possible orderings of the tuple $\vec{A}$. For numerical experiments, we devised a routine whose computation load grows quadratically rather than combinatorially which made calculation of diversity measure feasible.

$$\|A|B\| = \sup_{\vec{A}} d(\vec{A}|B) \tag{14}$$

$$\begin{aligned} d(\vec{A}) &= d(\vec{A}|\emptyset) \\ \|A\| &= \|A|\emptyset\| \end{aligned} \tag{15}$$

Section 6 of this paper introduces probabilistic computron which generates probability distributions on states. Therefore, we extended diversity measure to handle *weighted sets*. Let $w : S \to \mathbb{R}^{0+}$ be a weight function. *Individuality* of a weighted element x given a set of constraining weighted elements A is given as:

$$d_w(x|A) = w(x) \prod_{y \in A \setminus x} \rho(x, y) \tag{16}$$

Diversity and diversity measure for weighted sets follow the same definitions as in Equations (13) and (14) using individuality for weighted sets. We use the following extension for union $\tilde{\cup}$, symmetric difference $\tilde{\triangle}$, and subset $\tilde{\subseteq}$:

$$\begin{aligned} A \tilde{\cup} B &= \{x : x \in A \cup B, w_C(x) = (w_A(x) + w_B(x))\} \\ A \tilde{\triangle} B &= \{x : x \in A \cup B, w_C(x) = |w_A(x) - w_B(x)|\} \\ A \tilde{\subseteq} B &\Leftrightarrow (x \in A \Rightarrow x \in B, w_A(x) \leq w_B(x)) \end{aligned} \tag{17}$$

Lemma 2 (Diversity measure)**.** *Diversity is a sub-additive measure on Q.*

$$A \tilde{\subseteq} B \implies \|A\|_w \leq \|B\|_w \tag{18}$$

$$\|A_0 \tilde{\cup} A_1\|_w \leq \|A_0\|_w + \|A_1\|_w \tag{19}$$

Diversity-induced distance (DiD) is an induced metric on Q^* that captures dissimilarity between two sets, $A, B \in Q^*$:

$$\varphi(A, B) = \frac{\|A \tilde{\triangle} B\|}{|Q|} \tag{20}$$

where $\varphi : Q^* \times Q^* \to [0, 1]$, $\tilde{\triangle}$ is the symmetric difference operator as defined in Equation (17) and $|Q| = \sum_Q w(x)$ is size of weighted space.

Lemma 3 (DiD metric)**.** *(Q^*, φ) is a metric space.*

4.3. Entanglement

We introduce the entanglement concept which will later be used in defining partitioning complexity of a computron. *Entanglement* of a set of sets $\boldsymbol{A} = \{A_0, A_1, A_2...\}$ is defined as:

$$\|\boldsymbol{A}\| = \sup_{\vec{\boldsymbol{A}}} D(\vec{\boldsymbol{A}}) \tag{21}$$

where

$$D(\vec{\boldsymbol{A}}) = \|A_0\| + \|A_1|A_0\| + \|A_2|A_1, A_0\| \dots \tag{22}$$

Entanglement captures the total diversity of a collection of sets where each set's diversity is constrained by the presence of the other sets. The more the sets are within each other's proximity entanglement increases. On the other hand, as one given set's elements get separated from each other, that set's diversity decreases. Entanglement captures the balancing of these two factors in calculating the total diversity of the set of sets. This is illustrated in Figure 4.

Figure 4. Illustration of entanglement: Each element is assumed to be a pixel with Manhattan distance metric. Space is partitioned into two sets, $\boldsymbol{A} = \{A_0, A_1\}$. Three different partitions with increasing entanglement measure is provided.

Lemma 4 (Maximal entanglement)**.** *Let* $\boldsymbol{P} = \{B_0, B_1...\}$ *be a partitioning of* Q *(i.e.,* $\cup_i B_i = Q$*, and* $B_i \cap B_j = \emptyset, i \neq j$*) with* $\boldsymbol{P}^0 = \{Q\}$ *(nil partition) and* $\boldsymbol{P}^\infty = \{\{x_0\}, \{x_1\}, \dots\}, \forall x_i \in Q$ *(full partition), then:*

$$\|\boldsymbol{P}^0\| = \|\boldsymbol{P}^\infty\| \geq \|\boldsymbol{P}\| \tag{23}$$

5. Dynamics of Autonomous Computrons

5.1. Embedding in Metric Space

Given a computron $M = [G, C]$ with a generating graph G and connectors set C, we embed it in a metric space that captures similarity of its configuration states as follows: The distance between two cells is set as the *shortest-path-length* on generating graph between the two vertices occupied by these cells mapped to a unit interval. The intuition of this distance definition between two cells is that a state change in one of the cells can possibly impact the state of other one in compute steps that are no less than the shortest-path-length.

$$d(v_i, v_j) := \rho(\text{SPL}(v_i, v_j)) \tag{24}$$

$\rho : \mathbb{R}^{0+} \to [0, 1]$ concave, monotonic with $\rho(0) = 0$. We then define the configuration state of the computron as set of cells that have local state of 1.

$$s \in S := \{v_i | \omega_i = 1\} \tag{25}$$

This leads to diversity-induced distance as a metric for dissimilarity of the states for computron:

$$\varphi(s_0, s_1) := \frac{\|s_0 \triangle s_1\|}{|S|} \tag{26}$$

By Lemma 3, (S, φ) is a metric space with each configuration of the computron an element of the space. This is the *diversity space* that we embed the computron where distance between two global states of the machine is based on the similarity of the corresponding configurations. Figure 5 illustrates the concept on a computron where generating graph is assumed to be regular 2D grid. Each configuration is a set of pixels that are have state 1 (a binary image). Diversity-induced distance between configuration s_0 and s_1 is less than that of s_0 and s_2, capturing their similarity, although Hamming distances are the same.

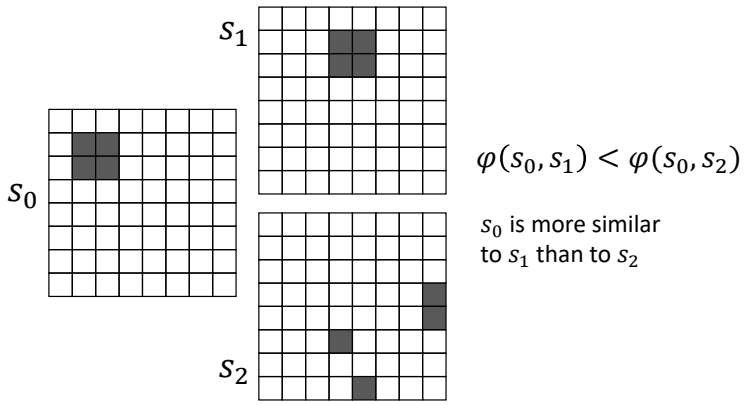

Figure 5. s_0, s_1, s_2 are configurations of a computron with 2D generating graph (binary images). Diversity-induced distance uses diversity measure to define similarity between configurations.

5.2. Computron as a Dynamical System

An autonomous computron has no external input. Starting from an initial configuration, it moves to next based on the transition map given in Equation (9). The dynamics is governed by discrete-time iteration on configuration metric space (S, φ):

$$\begin{aligned} s^{n+1} &= T(s^n) \\ T^{n+1} &:= T \circ T^n, T^0 = I, n \in \mathbb{N} \end{aligned} \tag{27}$$

An *attractor* of the computron is a tuple of configurations $\vec{A} = [s_0, s_1, ..s_{p-1}]$ where $s_{i+1} = T(s_i)$, $s_0 = T(s_{p-1})$ and $s_n \neq s_m \Leftrightarrow n \neq m$. For $p = 1$, it is called a fixed-point attractor with the corresponding state a *'fixed-state'*, else a periodic attractor. States that are elements of a periodic attractor are called *periodic states*. States that are not element of an attractor are called *transient states*. The *basin* of an attractor are those states that iterate into, or a part of, that attractor $B(\vec{A}) = \{s \in S : \exists k \ni T^k(s) \in \vec{A}\}$.

Lemma 5 (Basin partition)**.** *The set of all basins of an autonomous computron M is a partition of its configuration space S, denoted as $\boldsymbol{\mathcal{P}}_M = \{B_0, B_1, \dots\}$, with the corresponding attractors $\boldsymbol{\mathcal{A}}_M = \{\vec{A}_0, \vec{A}_1, \dots\}$ where $B_i \subset S$, $\cup_i B_i = S$ and $B_i \cap B_j = \emptyset, i \neq j$.*

Figure 6 gives a recap: A computron consists of a set of cells placed at vertices of a graph G with binary local states $\{\omega_i\}$. The configuration of the machine is specified as set of cells that are have local state 1. A basin is a set of configurations. As per Lemma A5, the configuration space is partitioned into basins by the computation performed by the computron.

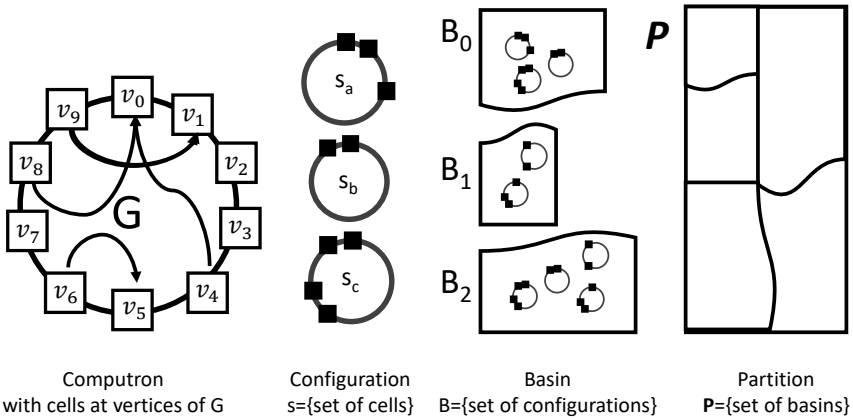

Figure 6. Dynamics of a computron is characterized by the structure of its basins of attraction. A partition is a set of basins; a basin is a set of configurations; a configuration is a set of cells that have local state 1.

5.3. Chaotic Computation

As defined in Equation (2) Lyapunov exponents characterize the rate of separation (or convergence) of infinitesimally close trajectories in a dynamical system. In discrete-time systems (maps), $x_{n+1} = f(x_n), n \in \mathbb{N}$, for an orbit starting with initial point x_0, Lyapunov exponent becomes:

$$\lambda(x_0) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln|f'(x_i)| \tag{28}$$

For a discrete-time discrete state-space dynamical system (i.e., dynamics of autonomous computron) on a metric space (S, φ), we define Lyapunov number for an initial configuration $s_0 \in S$ as follows: Let $\mathcal{B}_\epsilon(s) = \{q \in S : \varphi(s,q) < \epsilon\}$ denote the epsilon-ball around state s, where ϵ is set as the lowest value where epsilon-ball is non-empty. Let $\boldsymbol{\mathcal{A}}_M = \{A_0, A_1, \dots\}$ be the attractors of computron, and $\boldsymbol{\mathcal{P}}_M = \{B_0, B_1, \dots\}$ the corresponding basins of attraction. We calculate Lyapunov number of a state as the average separation of its neighboring states as iteration time goes to infinity. Since, each state converges to its attractor, we measure the dissimilarity of attractors using an induced DiD metric.

$$\lambda(s) = \frac{1}{|\mathcal{B}_\epsilon(s)|} \sum_{q\in\mathcal{B}_\epsilon(s)} \frac{\Phi(A_k, A_l)}{\Phi(\{s\},\{q\})} \tag{29}$$
$$s \in B(A_k), q \in B(A_l)$$

Lyapunov number for the computron is the average over the configuration space $\lambda_M := \langle\lambda(s)\rangle_S$.

5.4. Complex Computation

While chaos is a well-defined concept in dynamical systems theory, complexity has been more elusive with wide range of definitions as reviewed by Grassberger [24]. Observer's reference base using existing information is relevant for defining complexity, yet it leads to observer subjectivity. This can be avoided by either dealing with an equivalence class of the object as the reference base, or by treating the object on a self-referential basis using a generative description. In the former case, we are in the domain of information theoretical definition of complexity, whereas the latter is in the domain of algorithmic complexity. Either way, the complexity becomes a measure of difficulty in describing the object in question with regards to a reference base.

We introduce *diversity-based complexity* of a computron as the disentanglement of its basin partition. This is a basin-centric approach, not a path-centric one. Computron performs its computation by act of mapping the input configurations to attractors, generating a partitioning of configurations space

into basins of attraction. We consider the complexity of this partitioning as the defining characteristic of computron as the computational machine. This is different than Wolfram's complexity definition which is path-centric.

$$\Psi_M = 1 - \frac{\|\boldsymbol{\mathcal{P}}_M\|}{\|\boldsymbol{\mathcal{P}}^{\infty}{}_M\|} \tag{30}$$

with $\Psi_M \in [0,1]$ since $\|\boldsymbol{\mathcal{P}}_M\| \leq \|\boldsymbol{\mathcal{P}}^{\infty}{}_M\|$ by Lemma 4. This definition means that a complex computron has a basin partition where each basin consists of maximally diverse set of configurations with respect to other configurations in that basin, and furthermore basins themselves are maximally diverse with respect to each other. Therefore, a high complexity partitioning of the configuration space requires maximum information to describe such partitioning. As depicted in Figure 7 complexity of a partitioning increases initially with the number of partitions, then it starts decrease such that no partitioning and full partitioning have zero complexity.

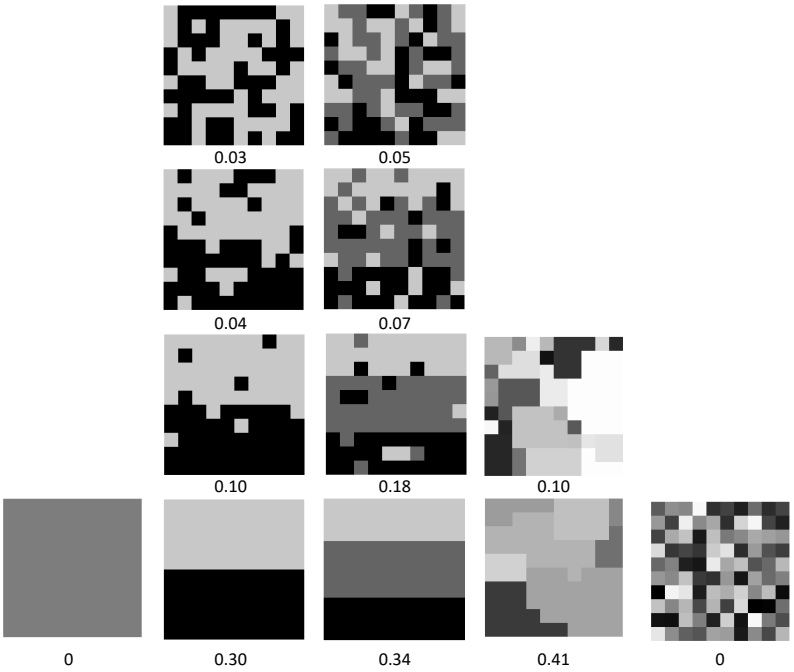

Figure 7. Illustration of diversity-based complexity measure for various partitionings. Nil partitioning and infinite partitioning has zero complexity. Complexity first increases with increasing number of partitions then starts to decrease. For a given number of partitions increasing entanglement reduces complexity. Complexity is maximized somewhere between order and chaos.

Lemma 6 (Minimal complexity)**.** *Absorbing and idle computrons have the least complexity (nil) compared to any other computron.*

$$\Psi_{M^0} = \Psi_{M^\infty} = 0 \leq \Psi_M \tag{31}$$

An *absorbing* computron has a single fixed-point attractor which all configurations iterate into denoted as $M^0 : \forall s \in S, \exists k \in \mathbb{N} \ni T^k(s) = s^*$, where T is the transfer map of computron and s^* is the absorbing state. An *idle* computron does nothing, i.e., each state is mapped to itself; and is denoted as $M^\infty : \forall s \in S, T(s) = s$.

5.5. Case Study—1D Cellular Automata

We analyzed complexity and chaoticity of autonomous computrons with linear cyclic generating graph and homogeneous connectors, i.e., elementary cellular automata. As expected, absorbing and idle computrons have minimal complexity and chaoticity. With same generating graph (linear cyclic of 5 cells), we analyzed different connector types. Chaoticity and complexity of each computron is plotted in Figure 8. There are several elementary CA that have very low complexity and not chaotic (less than 1 chaoticity). However, there is a group of machines that demonstrate high complexity while not chaotic.

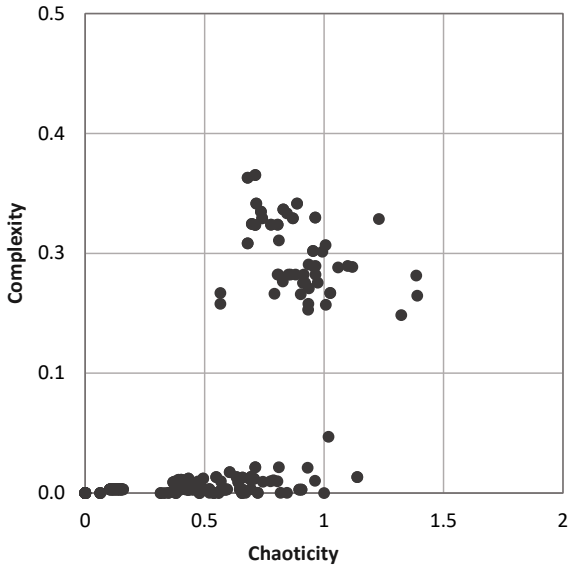

Figure 8. Chaoticity λ vs complexity Ψ of the universe of 32 state computrons with linear cyclic generating graph and homogeneous connectors

6. Dynamics of Driven Computrons

6.1. Computron as a Markov Chain

A *driven computron*, unlike an autonomous one, has a transition map that is a function of its current configuration and the external input drawn from binary alphabet. Dynamics is governed by discrete-time iteration:

$$s^{n+1} = T(s^n, \sigma^n) = \begin{cases} T^A(s^n) & \sigma^n = 0 \\ T^B(s^n) & \sigma^n = 1 \end{cases} \tag{32}$$

where σ^n is the external input at time n. We assume no a priori information about the input string other than observed frequency of alphabet letters, resulting in representing the input string as a sequence of independent identically distributed random variables on binary alphabet Σ with $p = Pr\{\sigma = 0\}$.

A *noisy computron* is a computron exposed to thermal fluctuations in the binary states of its cells. The dynamics is governed by probabilistic discrete-time iteration:

$$\begin{aligned} s^{n+1} &= T^{\varepsilon}(s^n, \sigma^n) \\ &= \begin{cases} T^A(s^n) & \text{with prob. } p(1-\varepsilon) \\ T^B(s^n) & \text{with prob. } (1-p)(1-\varepsilon) \\ s_r \in \mathcal{B}_\epsilon(s^n) & \text{with prob. } \varepsilon \end{cases} \end{aligned} \tag{33}$$

where $\mathcal{B}_\epsilon(s^n)$ is epsilon-ball neighborhood of configuration s^n and ϵ is the probability of malfunction due to thermal noise.

We now enter the domain of Markov chains to describe the probabilistic dynamics of the driven and noisy computrons. A *discrete Markov chain* is a sequence of random variables with domain S, $\{X_n, n \in \mathbb{N}\}$, where each random variable X_n is independent of prior random variables in the sequence given its previous one.

$$Pr\{X_n = j \mid X_{n-1} = i, X_{n-2} = k, \dots\} = P_{ij} \tag{34}$$

Probabilistic computrons are equivalent to a discrete Markov chain. They can be represented with directed labeled graphs where each vertex is assigned a configuration $s_i \in S$, and an edge from configuration s_i to s_j with transition probability $P_{ij} \neq 0$. It can also be represented by a $M \times M$ *transition probability matrix* $\mathbf{P}$ where $M = |S|$ is the number of all configurations and $\mathbf{P}_{ij} = P_{ij}$. From matrix representation, it follows that probability of reaching k from i in exactly n steps is $\mathbf{P}^n_{ik}$.

Terminology

- n-step *walk* is an ordered set of states $(i_0, i_1, \dots, i_n)$ in which there is a directed edge from i_{m-1} to i_m and no states are repeated
- State j is *accessible* from state i if there exists a walk from i to j, shown as $i \to j$
- Two states *communicate* if $i \to j$ and $j \to i$
- *Class* is a set of states where each pair of states in the set communicate, and none communicate with a state outside the class
- State i is called *recurrent* if $i \to j \Rightarrow j \to i$
- *Transient* state is a state that is not recurrent
- *Period* of a state i is the greatest-common-divisor of those values of n for which $P^n_{ii} > 0$. If the period is 1, the state is called *aperiodic*.
- It follows from definitions that for a given class, either all states are recurrent, or they are all transient. It can also be shown that all states in the same class have the same period.
- A class is called *ergodic* if it is recurrent and aperiodic.
- A Markov chain consisting entirely of single ergodic class is called and *ergodic chain*.
- A *unichain* contains a single recurrent class plus, possibly, some transient states. An *ergodic unichain* is a unichain for which the recurrent class is ergodic.
- A *steady-state distribution* for an M state Markov chain with transition matrix $\mathbf{P}$ is a row vector $\boldsymbol{\pi}$ that satisfies:

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}, \text{where } \pi_i \geq 0, \Sigma_i \pi_i = 1 \tag{35}$$

For a given Markov chain, Equation (35) always has a solution, and the solution is unique if and only if the Markov chain is a unichain. If there are c recurrent classes, then there are c linearly independent solutions each with non-zero entries only for the states of corresponding recurrent class. $\mathbf{P}^n$ converges if and only if recurrent classes are aperiodic with each row of $\mathbf{P}^n$ converging to $\boldsymbol{\pi}$ of one class.

We compare dynamical systems concepts for autonomous computron to that of Markov chain. An attractor is analogous to a recurrent class. An ergodic dynamical system has a single attractor, and similarly an ergodic chain has a single recurrent aperiodic class. In autonomous computron a transient state feeds into a single attractor, therefore the concept of basin of attraction is well-defined. However, in driven automaton, a transient state can feed into multiple recurrent classes with overlapping basins.

Let P_{ij} be the probability of transition from state s_i to s_j as described in Equation (33), and $\mathbf{P}$ the corresponding transition probability matrix. Let $p_n(s_i)$ be the probability of computron being in configuration s_i at time n, and $\mathbf{p}_n$ be the corresponding probability distribution vector over configuration space S. Its evolution starting from an initial distribution $\mathbf{p}_0$ is governed by the *master equation*:

$$\mathbf{p}_{n+1} = \mathbf{p}_n\mathbf{P} \tag{36}$$

Impact set R_s of a given configuration of computron is a set weighted by the impulse response distribution $\boldsymbol{\pi}_s$ defined as follows:

$$\boldsymbol{\pi}_s = \lim_{n\to\infty} \mathbf{p}_s^{\delta}\mathbf{P}^n$$
$$p_s^{\delta}(x) = \begin{cases} 1 & \text{if } x = s \\ 0 & \text{else} \end{cases} \tag{37}$$

$$R_s = \{x \in S : \pi_s(x) > 0\} \tag{38}$$

Since Equation (36) is linear, steady-state distribution of a driven computron can be written as linear combination of the impulse responses of all states weighted by the initial probability distribution. Let $\mathbf{p}_0$ be an arbitrary initial distribution stated as linear combination of delta–dirac distributions over all configurations:

$$\mathbf{p}_0 = \sum_{s\in S} p_0(s)\mathbf{p}_s^{\delta} \tag{39}$$

Then, steady-state distribution $\boldsymbol{\pi}$ becomes:

$$\boldsymbol{\pi} = \sum_{s\in S} p_0(s)\boldsymbol{\pi}_s \tag{40}$$

6.2. Ergodic Decomposition

A computron is said to be ergodic if its Markov chain representation is ergodic (i.e., single recurrent aperiodic class). Typically, computrons are non-ergodic, i.e., they have more than one recurrent class, and can be periodic. A periodic computron would not have a stationary steady-state distribution as $\mathbf{P}^n$ would not converge. Stationarity can be introduced by imposing *'asynchronicity'* where we modify each state's transition map with non-zero probability λ to *stay idle*.

$$\tilde{\mathbf{P}} = (1-\lambda)\mathbf{P} + \lambda\mathbf{I} \tag{41}$$

As per the definition of periodicity (see Terminology), the greatest-common-divisor of n for which $\tilde{P}_{ii}^n$ becomes 1 for all states (since they all have a transition to self with λ probability), and therefore the machine would become aperiodic. An aperiodic computron would have $\tilde{\mathbf{P}}^n$ converge with each row becoming $\tilde{\boldsymbol{\pi}}$ of one recurrent class.

The Coloring Lemma provides a technique for identifying the recurrent classes $\boldsymbol{\mathcal{C}}_M = \{C_1, C_2, \ldots, C_K\}$ of a computron. Let $\mathbf{p}_a, \mathbf{p}_b$ two random initial distributions with non-zero entries for each state, and let $\boldsymbol{\pi}_a$ and $\boldsymbol{\pi}_b$ be the corresponding steady-state distributions in an asynchronous computron. Define, the *'gray color'* of a state $c(s)$ as:

$$c(s) = \begin{cases} \pi_b(s)/\pi_a(s) & \text{if } \pi_a(s) > 0 \\ 0 & \text{else} \end{cases} \tag{42}$$

Lemma 7 (Coloring)**.** *Two states will, almost always, have the same shade of gray if and only if they are elements of same recurrent class, provided all classes of the computron are 'distinguishable' (as defined in the proof); i.e., for two random non-zero initial distributions,* $c(s) = c(q) > 0 \Leftrightarrow s, q \in C_i$*, almost always.*

Having identified the recurrent classes of a computron using the Coloring Lemma, we now use them as basis for ergodic decomposition:

Lemma 8 (Ergodic bases)**.** *Recurrent classes form a non-overlapping spanning set (basis) for the impact sets of all states. Let* $\boldsymbol{\mathcal{C}}_M = \{C_1, C_2, \ldots, C_K\}$ *be the set of recurrent classes of a computron and* $\boldsymbol{\chi}_s = \{\chi_1(s), \chi_2(s), \ldots \chi_K(s)\}$ *the ergodic decomposition of impulse response* $\boldsymbol{\pi}_s$*:*

$$\chi_k(s) = \sum_{q \in C_k} \boldsymbol{\pi}_s(q) \tag{43}$$

then:

$$\sum_k \chi_k(s) = 1, \textit{with } \chi_k(s) \geq 0, \forall s \in S \tag{44}$$

We can now formally define *basin of attraction for a recurrent class* as a weighted-set $\tilde{B}(C_k)$:

$$\tilde{B}(C_k) = \{s : \chi_k(s) > 0, w_{\tilde{B}}(s) = \chi_k(s)\} \tag{45}$$

Lemma 9 (Fuzzy partition)**.** *The set of basins of attraction of a probabilistic computron is a fuzzy partition of its configuration space, denoted as* $\tilde{\boldsymbol{\mathcal{P}}}_M = \{\tilde{B}(C_1), \tilde{B}(C_2), \ldots, \tilde{B}(C_K)\}$*. A fuzzy partition of S is the set of weighted sets* $\tilde{\boldsymbol{\mathcal{P}}} = \{\tilde{B}_1, \tilde{B}_2, ..\tilde{B}_K\}$*, where* $\cup_i \tilde{B}_i = S$*, and* $\sum_k w_{\tilde{B}_k}(s) = 1, \forall s \in S$*.*

6.3. Chaoticity—Complexity of Probabilistic Automata

A computron acting on a configuration space transforms each configuration into its impact set. Configurations that are similar to each other (based on diversity-induced distance) can be transformed to similar or different impact sets. We define the ratio of DiD distance among the impact sets of two configuration to distance between them as the *'bending ratio'*. Computron's action on the configuration space can be conceptualized as transforming the distance between each pairs of configurations with the bending ratio.

$$\tau(s, q) = \frac{\tilde{\Phi}(R_s, R_q)}{\Phi(\{s\}, \{q\})} \tag{46}$$

Lyapunov number for a state is the average bending of the ϵ-ball around such state:

$$\lambda(s) = \frac{1}{|\mathcal{B}_\epsilon(s)|} \sum_{q \in \mathcal{B}_\epsilon(s)} \tau(s, q) \tag{47}$$

Chaoticity of the machine is measured by the average Lyapunov number over configuration space. $\lambda_M := \langle \lambda(s) \rangle_S$.

As an extension of the complexity definition we used for autonomous computrons, Equation (30), complexity of a probabilistic computron is the disentanglement of its fuzzy basin partition.

$$\tilde{\Psi}_M = 1 - \frac{\|\tilde{\boldsymbol{\mathcal{P}}}_M\|}{\|\tilde{\boldsymbol{\mathcal{P}}}_M\|} \in [0, 1] \tag{48}$$

As a case study we analyzed the complexity of hybrid elementary CA that execute two alternative rules based on the input signal. Figure 9 shows the variation in complexity of hybrid CAs with the driver mix $p(\sigma = 1)$ changing in $[0, 1]$. For $p = 0$ and $p = 1$ the machine is regular elementary CA. We plotted the deviation of complexity from a linear mix, so-called synergy, as p is varied. Creating hybrid CAs using high and low complexity constituents results in drop in overall complexity (i.e., non-synergistic mix). The drop happens at even small mixing around the higher complexity CA. This compares to synergistic case when both constituents are of high complexity.

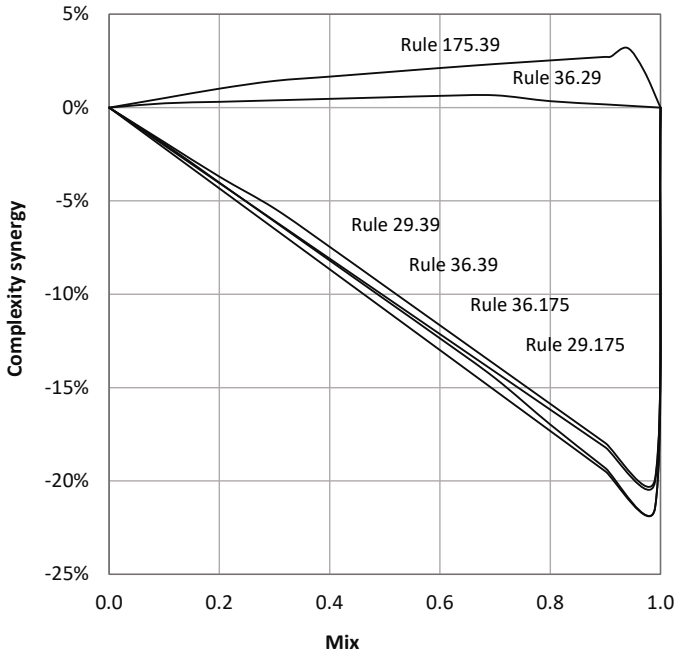

Figure 9. Increase (decrease) in complexity of driven cellular automata compared to linear combination of autonomous CA, with changing intensity of driving signal $p(\sigma = 1) \in [0, 1]$.

6.4. Susceptibility

Let M^A, M^B be two computrons operating on the same state space. We provide a measure of topological similarity between them by comparing their bendings of the configuration space. The bending ratio $\tau(s, q)$ was defined in Equation (46). The similarity of two computrons is given as:

$$\mathbf{\Phi}(M^A, M^B) = \sum_{s,q \in S \times S} |\tau_A(s, q) - \tau_B(s, q)| \tag{49}$$

Equipped with a measure for similarity of two computrons, we can now analyze distortions in a machine upon small structural changes. As an application, we investigate distortion of a computron due to noise which is given as:

$$\xi(\mu) = \mathbf{\Phi}(M_{p_\mu = \mu}, M_{p_\mu = 0}) \tag{50}$$

Figure 10 shows noise distortion curves on log-log scale for a computron executing Rule 29.175 with varying levels of driver intensity. The machine exhibits linear noise distortion curves with saturation at high noise levels when it is driven in region Rule 29 towards Rule 175. However, as it gets closer to Rule 175, noise distortion become exponential with power $\gamma \approx 1.45$. This characterization is consistent with the sudden change in complexity around Rule 175 as was provided in Figure 9.

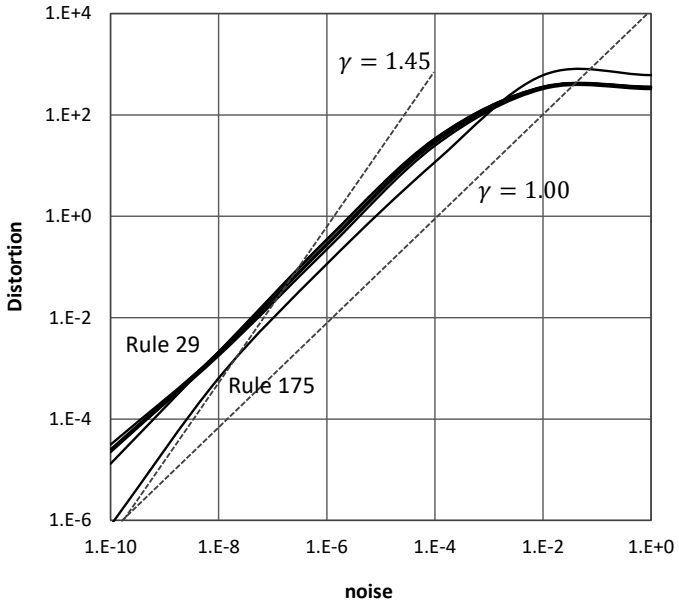

Figure 10. Topological distortion to driven computron due to noise. Distortion vs noise levels are plotted in log-log scale. Curves correspond to different levels of driver intensity which results in varying the machine's behavior from Rule 29 to Rule 175 gradually.

7. Discussion

This paper is an investigation into dynamics and complexity of a computation machine, computron, which is equivalent to a finite-state machine and generalization of a cellular automata. We embedded the computron into a metric space that captures similarity of configurations as patterns on the generating graph of computron. We analyzed the chaoticity and complexity of computron in this metric space. Our approach to complexity was basin-centric rather than path-centric. We also expanded the framework into non-autonomous machines with external drivers which led to Markov chain formalism. We applied our theory to hybrid elementary cellular automata with noise. Case studies showed that machines with high complexity are more susceptible to externalities, such as mixing and noise.

Our framework of conceptualizing automata as computrons in diversity space opens up a series of avenues for future research, including the study of robustness and complexity of various generating graphs (e.g., hierarchical) or connector families (e.g., reversible, linear). We can also investigate evolutionary dynamics of a computational machine by *'growing'* (attaching cells to an existing machine) and *'mutating'* (altering generating graph or the connectors) computrons.

Author Contributions: M.E. is the sole author for this article, including conceptualization, theory, proofs, coding and simulations. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We would like to thank Henrik Jeldtoft Jensen (Leader of the Centre for Complexity Science at Imperial College London) and Ada Diaconescu (Computer Science and Networks Department, Telecom ParisTech) for insightful discussions on this work.

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FSA	finite-state automata
CA	cellular automata
DiD	diversity-induced distance

Appendix A. Proof of Lemmas

Lemma A1. *For any finite-state automaton with 2^N states there is an equivalent computron of N cells plus the input cell.*

Two machines $M = [S^M, s_0, \Sigma, \Gamma, T^M, U^M]$ and $\Omega = [S^\Omega, s_0, \Sigma, \Gamma, T^\Omega, U^\Omega]$ are said to be *equivalent* if there exists a one-to-one mapping between the states $\mu : S^M \leftrightarrow S^\Omega$, and for all $s \in S^M$ and $\sigma \in \Sigma$:

$$\begin{aligned} \mu(T^M(s,\sigma)) &= T^\Omega(\mu(s),\sigma) \\ U^M(s,\sigma) &= U^\Omega(\mu(s),\sigma) \end{aligned} \tag{A1}$$

A generating graph $G(V,E)$ is said to be *fully connected* if $\forall v_i, v_j \in V, \exists e_{ji} \in E$. A state of a computron is the configuration of its cell states $s^\Omega = [\omega_0\omega_1...\omega_{N-1}], \omega_i \in \{0,1\}$ with *base indexing* defined as $k(s^\Omega) = \sum_{i=0}^{N-1} 2^{\omega_i}$, where $N = |V_G|$ is the number of vertices of the graph. following 3 propositions construct the proof for the Lemma.

Proposition A1. *For any computron Ω with an arbitrary generating graph G there exists an equivalent computron Ω^* with a fully connected generating graph G^* with same number of vertices.*

We pair up cells of Ω and Ω^* as $v_i \leftrightarrow v_i^*$ and we use base indexing to enumerate configurations s^Ω and s^{Ω^*}. This establishes the one-to-one mapping between the two configuration spaces $\mu : S^\Omega \leftrightarrow S^{\Omega^*}$. Then we construct the connectors $[C^*]$ of Ω^* as follows: Let $\delta_i = [v_{i_0}...v_{i_{n(i)}}]$ be the neighbors of cell v_i, and let $\delta_i^{'}$ be the rest of the cells (i.e., not-connected to v_i.), then we set the loop-up tables of connectors C_i^* for each cell as follows:

$$C_i^*([\delta_i, \delta_i^{'}], \sigma) = C_i([\delta_i], \sigma) \tag{A2}$$

With output maps also set identical, $U^*(s,\sigma) = U(s,\sigma)$, then the implied Transition Maps $T^\Omega(s)$ and $T^{\Omega^*}(s)$ satisfy Equation (A1). Therefore, $\Omega \equiv \Omega^*$.

Proposition A2. *There is one-to-one correspondence between finite-state automata M of 2^N states with binary input and fully connected computrons Ω^* with $N+1$ cells, including the input cell.*

Let $\mathbb{U}_{\Omega^*}$ denote set of all fully connected computrons with $N+1$ cells (including the input cell).

$$|\mathbb{U}_{\Omega^*}| = (2^N)^{(2^N)} \times (2^N)^{(2^N)} \tag{A3}$$

Each cell of Ω^* (other than the special input cell) has $N+1$ inputs from other cells and itself (fully connected) with 2^{N+1} input configurations. With each input configuration having a binary output set by the connector, the number of different connector types are $2^{(2^{N+1})}$. Each cell, other than the input cell, can have any of the connector types (input cell has a unique connector that solely acts as interface to external input). Therefore, total number of different computrons with fully connected generating graph is $(2^{2^{N+1}})^N = (2^N)^{(2^N)} \times (2^N)^{(2^N)}$.

Let M be a finite-state automaton with 2^N states, and let $\mathbb{U}_M$ denote set of all such machines. Since each state can go to any state based on the binary external input, we have:

$$\begin{aligned} |\mathbb{U}_M| &= (2^N)^{(2^N)} \times (2^N)^{(2^N)} \\ &= |\mathbb{U}_{\Omega^*}| \end{aligned} \tag{A4}$$

Base indexing provides the one-to-one map between the configurations of Ω^* and (excluding the input cell) and the states of M. From Proposition A1, for each computron there is an equivalent fully connected computron. From A2, there is a one-to-one mapping between finite-state automata of 2^N states with binary input and a fully connected computron of $N+1$ cells, including the input cell. Therefore, for any finite-state automata there exists an equivalent computron.

Lemma A2 (Diversity measure)**.** *Diversity is a sub-additive measure on Q.*

Monotonicity:

$$A \tilde{\subset} B \implies \|A\|_w < \|B\|_w \tag{A5}$$

Let $A = \{a, b, c\}, B = \{a, b, c, x\}$ with corresponding weights $\{w_a^A, w_b^A, w_c^A\}$ and $\{w_a^B, w_b^B, w_c^B, w_x^B\}$, respectively, satisfying subset definition given in Equation (17). Let $\vec{A}^* = [bac]$ be the optimal ordering of A-tuple that maximizes $\sup_{\vec{A}} \|\vec{A}\|_w$, and construct $\vec{B}^* = [bacx]$.

$$\begin{aligned} \|\vec{A}^*\| &= w_a^A + \quad w_b^A \rho_{ab} + w_c^A \rho_{cb} \rho_{ca} \\ \|\vec{B}^*\| &= w_a^B + \quad w_b^B \rho_{ab} + w_c^B \rho_{cb} \rho_{ca} + \quad w_x^B \rho_{xb} \rho_{xa} \rho_{xc} \end{aligned} \tag{A6}$$

Therefore, $\|A\|_w = \|\vec{A}^*\|_w < \|\vec{B}^*\|_w < \sup_{\vec{B}} \|\vec{B}\|_w = \|B\|_w \square$

Sub-additivity:

$$\|A_0 \tilde{\cup} A_1\|_w \leq \|A_0\|_w + \|A_1\|_w \tag{A7}$$

Let $A = \{a, x\}$, $B = \{b, x\}$ and $C = A \tilde{\cup} B = \{a, b, x\}$ with corresponding weights $\{w_a^A, w_x^A\}$, $\{w_b^B, w_x^B\}$, and $\{w_a^A, w_b^B, w_x^A + w_x^B\}$, respectively. Let $\vec{C}^* = [bxa]$ be the optimal ordering of C-tuple that maximizes $\sup_{\vec{C}} \|\vec{C}\|_w$. Construct $\vec{A}^* = [xa]$ and $\vec{B}^* = [bx]$ as lifting from $\vec{C}^*$ with same ordering within. Then:

$$\begin{aligned} \|\vec{C}^*\|_w &= (w_b^B) + (w_x^A + w_x^B)\rho_{xb} + (w_a^A)\rho_{ab}\rho_{ax} \\ \|\vec{A}^*\|_w &= \quad\quad + (w_x^A) \quad\quad\quad + (w_a^A)\rho_{ax} \\ \|\vec{B}^*\|_w &= (w_b^B) + (w_x^B)\rho_{xb} \end{aligned}$$

Since $\rho \in [0, 1]$, $\|C\|_w = \|\vec{C}^*\|_w \leq \|\vec{A}^*\|_w + \|\vec{B}^*\|_w \leq \|A\|_w + \|B\|_w$.

Lemma A3 (Diversity-Induced Distance)**.** *'DiD' defined as* $\varphi(A, B) = \frac{\|A \tilde{\triangle} B\|_w}{|Q|}$ *is a metric of* Q^*

Let $A, B, C \in Q^*$ and diversity-induced distance $\varphi(A, B) = \frac{\|A \tilde{\triangle} B\|_w}{|Q|}$, then from definitions $\varphi(A, A) = 0$, and $\varphi(A, B) = \varphi(B, A) \geq 0$. We need to show the triangle-inequality property to establish DiD as a metric: $\varphi(A, B) \leq \varphi(A, C) + \varphi(C, B)$. First, we will show that $A \tilde{\triangle} B \subseteq (A \tilde{\triangle} C) \tilde{\cup} (C \tilde{\triangle} B)$. Let $A = \{a, x, y, p\}$, $B = \{b, x, x, p\}$, and $C = \{c, y, z, p\}$ with corresponding weights:

$A\tilde{\triangle}C$	$C\tilde{\triangle}B$	$A\tilde{\triangle}B$	$(A\tilde{\triangle}C)\tilde{\cup}(C\tilde{\triangle}B)$
w_a		w_a	w_a
	w_b	w_b	w_b
w_c	w_c		w_c
w_x^A	w_x^B	$\lvert w_x^A - w_x^B\rvert$	$(w_x^A + w_x^B)$
$\lvert w_y^A - w_y^C\rvert$	w_y^C	w_y^A	$\lvert w_y^A - w_y^C\rvert + w_y^C$
w_z^C	$\lvert w_z^C - w_z^B\rvert$	w_z^B	$w_z^C + \lvert w_z^C - w_z^B\rvert$
$\lvert w_p^A - w_p^C\rvert$	$\lvert w_p^C - w_p^B\rvert$	$\lvert w_p^A - w_p^B\rvert$	$\lvert w_p^A - w_p^C\rvert$ $+\lvert w_p^C - w_p^B\rvert$

For each row, weight for $A\tilde{\triangle}B$ is less than or equal to the weight for $(A\tilde{\triangle}C)\tilde{\cup}(C\tilde{\triangle}B)$. Therefore, $A\tilde{\triangle}B \subseteq (A\tilde{\triangle}C)\tilde{\cup}(C\tilde{\triangle}B)$. Then the rest follows from Lemma A2:

$$\begin{aligned} A\tilde{\triangle}B &\subseteq (A\tilde{\triangle}C)\tilde{\cup}(C\tilde{\triangle}B) \\ \|A\tilde{\triangle}B\|_w &\leq \|(A\tilde{\triangle}C)\tilde{\cup}(C\tilde{\triangle}B)\|_w \\ &\leq \|A\tilde{\triangle}C\|_w + \|C\tilde{\triangle}B\|_w \end{aligned} \tag{A8}$$

Lemma A4 (Maximal entanglement)**.** *Let* $\boldsymbol{P} = \{B_0, B_1 \dots\}$ *be a partitioning of* Q *(i.e.,* $\cup_i B_i = Q$*,and* $B_i \cap B_j = \emptyset, i \neq j$*) with* $\boldsymbol{P}^0 = \{Q\}$ *and* $\boldsymbol{P}^\infty = \{\{x_0\}, \{x_1\}, \dots\}, \forall x_i \in Q$*, then:*

$$\|\boldsymbol{P}^0\| = \|\boldsymbol{P}^\infty\| \geq \|\boldsymbol{P}\| \tag{A9}$$

The proof is based on a combinatorics argument. Entanglement of $\boldsymbol{P}^0$ is calculated by shuffling all the elements of universe Q within a single set, whereas entanglement of $\boldsymbol{P}^\infty$ is calculated by shuffling sets of all single elements, to find the supremum.

$$\begin{aligned} \|\boldsymbol{P}^0\| &= \sup_{\vec{\boldsymbol{P}}} D(\vec{\boldsymbol{P}}) = \|Q\| = \sup_{\vec{Q}} d(\vec{Q}) \\ \|\boldsymbol{P}^\infty\| &= \sup_{\vec{\boldsymbol{P}}} D(\vec{\boldsymbol{P}}) = \|\{x_i\}\| + \|\{x_j\}|\{x_i\}\| + \dots \\ &= d(x_i) + d(x_j|x_i) + \dots \\ &= \sup_{\vec{Q}} d(\vec{Q}) \end{aligned} \tag{A10}$$

Therefore, $\|\boldsymbol{P}^0\| = \|\boldsymbol{P}^\infty\|$. For any other partition $\boldsymbol{P}$, entanglement is calculated by shuffling the set blocks of the partition together with shuffling the elements within each set block. Therefore, its supremum is less than or equal to that of $\boldsymbol{P}^0$ which seeks the supremum through all possible orderings. As a simple case consider $Q = \{a, b, c\}$. Let $\boldsymbol{P}^0 = \{\{a, b, c\}\}, \boldsymbol{P}^\infty = \{\{a\}, \{b\}, \{c\}\}$, and $\boldsymbol{P}^1 = \{\{a, b\}, \{c\}\}$

$\|\boldsymbol{P}^0\|$ $\sup_A d(.)$	$\|\boldsymbol{P}^\infty\|$ $\sup_B d(.)$	$\|\boldsymbol{P}^1\|$ $\sup_C d(.)$
A	B	C
abc	abc	abc
acb	acb	
bac	bac	bac
bca	bca	
cab	cab	cab
cba	cba	cba

Therefore, $\|\boldsymbol{P}^0\| = \|\boldsymbol{P}^\infty\| \geq \|\boldsymbol{P}\|$.

Lemma A5 (Basin partition)**.** *The set of all basins of an autonomous computron is a partition of its configuration space, denoted as* $\boldsymbol{\mathcal{P}}_M = \{B_0, B_1, \dots\}$*, with the corresponding attractors* $\boldsymbol{\mathcal{A}}_M = \{\vec{A}_0, \vec{A}_1, \dots\}$.

A partition of S is $\boldsymbol{\mathcal{P}} = \{B_0, B_1, \dots\}$, where $B_i \subset S$, $\cup_i B_i = S$ and $B_i \cap B_j = \emptyset, i \neq j$. So, we need to show that each state belongs to some basin and no state belongs to more than one basin. Due to finite-state nature of the computron this is trivial. Let $P_n(s_0) = \{s_0, s_1, ..s_n\}$ be the n-step forward path of state s_0. Since $|S| < \infty, \exists k \leq |S|$ such that $s_{k+1} \in P_k(s_0)$ and $s_{i+1} \notin P_i(s_0), \forall i < k$. Denote $s_{k+1} = s_q (q \leq k)$. Then, $\vec{A} = [s_q, s_{q+1} ... s_k]$ is an attractor of the computron, as per definition, and $s_0 \in B(\vec{A})$. Therefore, each state is an element of some basin.

Now we show that no state belongs to more than one basin. Assume $s \in B(\vec{A}_a)$ and $s \in B(\vec{A}_b)$ with $\vec{A}_a = [s_{q_a} \dots s_{k_a}]$ and $\vec{A}_b = [s_{q_b} \dots s_{k_b}]$. If $k_b < k_a$ then $\exists i = k_b \ni s_{i+1} \in P_i(s)$ which means $\vec{A}_b$ is not an attractor as it violates attractor definition, and similarly if $k_a < k_b$ then $\vec{B}_b$ is not an attractor. Therefore $k_a = k_b$ and $\vec{A}_a = \vec{A}_b$.

Lemma A6 (Minimal complexity)**.** *Absorbing and idle computrons have the least complexity (nil) compared to any other computron.*

$$\Psi_{M^0} = \Psi_{M^\infty} = 0 \leq \Psi_M \tag{A11}$$

Absorbing computron M^0 has a single fixed-point attractor which all states iterate into therefore its basin partitioning is $\boldsymbol{\mathcal{P}}_{M^0} = \{S\} = \boldsymbol{P}^0$, and idle computron M^∞ each state is mapped to itself, therefore its basin partitioning is $\boldsymbol{\mathcal{P}}_{M^\infty} = \{\{s_0\}, \{s_1\} \dots\} = \boldsymbol{P}^\infty$. Hence from definition of complexity, $\Psi_{M^0} = \Psi_{M^\infty} = 0 \leq \Psi_M$.

Lemma A7 (Coloring)**.** *Let* $\mathbf{p}_a, \mathbf{p}_b$ *two random initial distributions with non-zero entries for each state, and let* $\boldsymbol{\pi}_a$ *and* $\boldsymbol{\pi}_b$ *be the corresponding steady-state distributions in an asynchronous computron. Define, the* 'gray color' *of a state* $c(s)$ *as:*

$$c(s) = \begin{cases} \pi_b(s)/\pi_a(s) & \text{if } \pi_a(s) > 0 \\ 0 & \text{else} \end{cases} \tag{A12}$$

Two states will, almost always, have the same shade of gray if and only if they are elements of same recurrent class, provided all classes of the computron are *'distinguishable'*; i.e., for two random non-zero initial distributions, $c(s) = c(q) > 0 \iff s, q \in C_i$, almost always.

Let $\boldsymbol{\mathcal{C}}_M = \{C_1, C_2, \dots, C_K\}$ be the recurrent classes of a computron, and $\boldsymbol{\pi}_M = \{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \dots \boldsymbol{\pi}_K, \}$ the corresponding steady-state distributions of the class. Then:

$$\boldsymbol{\pi_a} = \sum_{k=1}^{K} \chi_k(\mathbf{p}_a)\boldsymbol{\pi}_k, \ \boldsymbol{\pi_b} = \sum_{k=1}^{K} \chi_k(\mathbf{p}_b)\boldsymbol{\pi}_k \tag{A13}$$

First we show, $s, q \in C_m \implies c(s) = c(q)$. By definition recurrent (see Terminology) classes are non-overlapping. If $s, q \in C_m$, then:

$$\begin{aligned} \pi_a(s) &= \chi_m(\mathbf{p}_a)\pi_m(s), \ \pi_a(q) = \chi_m(\mathbf{p}_q)\pi_m(q) \\ \pi_b(s) &= \chi_m(\mathbf{p}_b)\pi_m(s), \ \pi_b(q) = \chi_m(\mathbf{p}_b)\pi_m(q) \end{aligned} \tag{A14}$$

Therefore,

$$c(s) = \pi_a(s)/\pi_b(s) = \frac{\chi_m(\mathbf{p}_a)}{\chi_m(\mathbf{p}_b)} = \pi_a(q)/\pi_b(q) = c(q) \tag{A15}$$

We define *'distinguishable'* as follows: Let $\vec{W}_{x\to y} = x_0 x_1 ... x_k$ be a *walk* state s_x to state s_y. We define the walk probability as $p(\vec{W}_{x\to y}) = \prod_{i=0}^{k-1} \mathbf{P}_{x_i x_{i+1}}$. Let $\tilde{W}_{x\to y}$ denote set of all different walks from state

s_x to state s_y. We define reach probability from s_x to s_y as $p_{x\to y} = \sum_{\vec{W}\in\vec{\mathcal{W}}} p(\vec{W}_{x\to y})$. We define reach probability from a state s_x to an attractor A as $p_{x\to A} = \sum_{s_y\in A} p_{x\to y}$, where the walk from $s_y \in A$ there does not have any other element of A. Two recurrent classes A and B are said to be *'equidistant'* with respect to state s_x, if $p_{x\to A} = p_{x\to B}$. Two recurrent classes A and B are said to be not *'distinguishable'* if they are equidistant with respect to all states in S. Define the $|S| \times |\boldsymbol{\mathcal{C}}|$ matrix $\mathbf{P}^{\boldsymbol{\mathcal{C}}}$ as follows:

$$\mathbf{P}^{\boldsymbol{\mathcal{C}}} = \begin{bmatrix} p_{0\to C_1} & p_{0\to C_2} & \cdots & p_{0\to C_K} \\ p_{1\to C_1} & p_{1\to C_2} & \cdots & p_{1\to C_K} \\ \vdots & & & \end{bmatrix} \tag{A16}$$

If any two columns of $\mathbf{P}^{\boldsymbol{\mathcal{C}}}$ are equal then the corresponding recurrent classes are indistinguishable. Let $\boldsymbol{\chi}(\mathbf{p}_a) = [\chi_1(\mathbf{p}_a) \dots \chi_K(\mathbf{p}_a)]$, then:

$$\boldsymbol{\chi}(\mathbf{p}_a) = \mathbf{p}_a \mathbf{P}^{\boldsymbol{\mathcal{C}}} \tag{A17}$$

Now, we show that almost always, for any two recurrent classes C_m, C_n that are *'distinguishable'*, $s \in C_m, q \in C_n \implies c(s) \neq c(q)$. Assume the contrary:

$$\begin{aligned} c(s) = c(q) &\implies \frac{\chi_m(\mathbf{p}_a)}{\chi_m(\mathbf{p}_b)} = \frac{\chi_n(\mathbf{p}_a)}{\chi_n(\mathbf{p}_b)} \\ &\implies \frac{\mathbf{p}_a.\mathbf{P}^{\boldsymbol{\mathcal{C}}}[m]}{\mathbf{p}_a.\mathbf{P}^{\boldsymbol{\mathcal{C}}}[n]} = \frac{\mathbf{p}_b.\mathbf{P}^{\boldsymbol{\mathcal{C}}}[m]}{\mathbf{p}_b.\mathbf{P}^{\boldsymbol{\mathcal{C}}}[n]} \end{aligned} \tag{A18}$$

The set of distinct initial distributions pairs $\mathbf{p}_a, \mathbf{p}_b$ that are solution to Equation (A18) represent a plane in $\Re^{|S|}$ which may or may not be a probability distribution. The ratio of size of this set to size of all possible initial distribution pairs is $\frac{\mathcal{O}(|S|-1)}{\mathcal{O}(|S|)} \approx 0$, hence contrary assumption leads to contradiction almost always.

Lemma A8 (Ergodic bases)**.** *Recurrent classes form a non-overlapping spanning set (basis) for the impact sets of all states. Let* $\boldsymbol{\mathcal{C}}_M = \{C_1, C_2, \dots, C_K\}$ *be the set of recurrent classes of a computron and* $\boldsymbol{\chi}_s = \{\chi_1(s), \chi_2(s), \dots \chi_K(s)\}$ *the ergodic decomposition of impulse response* $\boldsymbol{\pi}_s$*:*

$$\chi_k(s) = \sum_{q\in C_k} \boldsymbol{\pi}_s(q) \tag{A19}$$

then:

$$\sum_k \chi_k(s) = 1, \textit{ with } \chi_k(s) \geq 0, \forall s \in S \tag{A20}$$

$$\begin{aligned} \boldsymbol{\chi}(\mathbf{p}_a) &= \mathbf{p}_a \mathbf{P}^{\boldsymbol{\mathcal{C}}} \\ \chi_k(s) &= \mathbf{p}_s^{\delta} \mathbf{P}^{\boldsymbol{\mathcal{C}}}[k] = \mathbf{P}^{\boldsymbol{\mathcal{C}}}[s][k] = p_{s\to C_k} \end{aligned} \tag{A21}$$

$$\sum_k \chi_k(s) = \sum_k p_{s\to C_k} = 1 \tag{A22}$$

Lemma A9 (Fuzzy Partition)**.** *The set of all basins of attraction of a driven computron is a fuzzy partition of its configuration space, denoted as* $\tilde{\boldsymbol{\mathcal{P}}}_M = \{\tilde{B}(C_1), \tilde{B}(C_2), \dots, \tilde{B}(C_K)\}$.

Basins of attraction for an asynchronous computron is a weighted-set $\tilde{B}(C_k)$:

$$\tilde{B}(C_k) = \{s : \chi_k(s) > 0, w_{\tilde{B}}(s) = \chi_k(s)\} \tag{A23}$$

A fuzzy partition of S is the set of weighted sets $\tilde{\boldsymbol{\mathcal{P}}} = \{\tilde{B}_1, \tilde{B}_2, ..\tilde{B}_K\}$, where $\cup_i \tilde{B}_i = S$, and $\sum_k w_{\tilde{B}_k}(s) = 1, \forall s \in S$. The proof follows directly from definition of basin, Equation (A23) and Lemma (A8).

References

1. Shereshevsky, M.A. Lyapunov Exponents for One-Dimensional Cellular Automata. *J. Nonlinear. Sci.* **1992**, *2*, 1–8. [CrossRef]
2. Tisseur, P. Cellular automata and Lyapunov exponents. *Nonlinearity* **2000**, *13*, 1547–1560. [CrossRef]
3. Hurd, L.P.; Kari, J.; Culik, K. The Topological-Entropy of Cellular Automata Is Uncomputable. *Ergod. Theor. Dyn. Syst.* **1992**, *12*, 255–265. [CrossRef]
4. Finelli, M.; Manzini, G.; Margara, L. Lyapunov exponents versus expansivity and sensitivity in cellular automata. *J. Complex.* **1998**, *14*, 210–233. [CrossRef]
5. Manzini, G.; Margara, L. Attractors of linear cellular automata. *J. Comput. Syst. Sci.* **1999**, *58*, 597–610. [CrossRef]
6. Cattaneo, G.; Formenti, E.; Manzini, G.; Margara, L. Ergodicity, transitivity, and regularity for linear cellular automata over Z(m). *Theor. Comput. Sci.* **2000**, *233*, 147–164. [CrossRef]
7. D'amico, M.; Manzini, G.; Margara, L. On computing the entropy of cellular automata. *Theor. Comput. Sci.* **2003**, *290*, 1629–1646. [CrossRef]
8. Lempel, A.; Ziv, J. Complexity of Finite Sequences. *IEEE Trans. Inf. Theory* **1976**, *22*, 75–81. [CrossRef]
9. Zenil, H. Compression-Based Investigation of the Dynamical Properties of Cellular Automata and Other Systems. *Complex Syst.* **2010**, *19*. [CrossRef]
10. Wolfram, S. Computation Theory of Cellular Automata. *Commun. Math. Phys.* **1984**, *96*, 15–57. [CrossRef]
11. Wolfram, S. Cellular Automata as Models of Complexity. *Nature* **1984**, *311*, 419–424. [CrossRef]
12. Wolfram, S. 20 Problems in the Theory of Cellular Automata. *Phys. Scr.* **1985**, *1985*, 170. [CrossRef]
13. Culik, K.; Yu, S. Undecidedability of CA classification schemes. In *Complex Systems 2*; Complex Systems Publications, Inc: Champaign, IL, USA, 1988; pp. 177–190.
14. Li, W.T.; Packard, N.H.; Langton, C.G. Transition Phenomena in Cellular Automata Rule Space. *Phys. D* **1990**, *45*, 77–94. [CrossRef]
15. Kurka, P. Languages, equicontinuity and attractors in cellular automata. *Ergod. Theor. Dyn. Syst.* **1997**, *17*, 417–433. [CrossRef]
16. Chua, L.O.; Yoon, S.; Dogaru, R. A nonlinear dynamics perspective of Wolfram's new kind of science. Part I: Threshold of complexity. *Int. J. Bifurc. Chaos* **2002**, *12*, 2655–2766. [CrossRef]
17. Ewert, T. A Measure for the Complexity of Elementary Cellular Automata. *Complex Syst.* **2019**, *28*. [CrossRef]
18. Marr, C.; Hutt, M.T. Topology regulates pattern formation capacity of binary cellular automata on graphs. *Phys. A* **2005**, *354*, 641–662. [CrossRef]
19. Tomassini, M. Generalized automata networks. In *Cellular Automata*; Lecture Notes in Computer Science; El Yacoubi, S., Chopard, B., Bandini, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4173, pp. 14–28.
20. Cattaneo, G.; Dennunzio, A.; Formenti, E.; Provillard, J. Non-uniform Cellular Automata. In *Language and Automata Theory and Applications*; Lecture Notes in Computer Science; Dediu, A.H., Ionescu, A.M., Martín-Vide, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5457, pp. 302–313. [CrossRef]
21. Daza, A.; Wagemakers, A.; Georgeot, B.; Guery-Odelin, D.; Sanjuan, M.A.F. Basin entropy: A new tool to analyze uncertainty in dynamical systems. *Sci. Rep.* **2016**, *6*. [CrossRef]
22. Eiter, T.; Mannila, H. Distance measures for point sets and their computation. *Acta Inform.* **1997**, *34*, 109–133. [CrossRef]

23. Bennett, C.H.; Gacs, P.; Li, M.; Vitanyi, F.M.B.; Zurek, W.H. Information distance. *IEEE Trans. Inf. Theory* **1998**, *44*, 1407–1423. [CrossRef]
24. Shreim, A.; Grassberger, P.; Nadler, W.; Samuelsson, B.; Socolar, J.E.S.; Paczuski, M. Network analysis of the state space of discrete dynamical systems. *Phys. Rev. Lett.* **2007**, *98*, 198701. [CrossRef] [PubMed]