COMPUTATION, DYNAMICS AND THE PHASE-TRANSITION

JEREMY AVNET

jeremy@theory.org

ABSTRACT. From a description of one of the simplest models of computation (discrete, finite state automata), a complex dynamical system is formed. Known as a cellular automaton, it has been proved that these systems can harbor the conditions necessary for universal computation. An exploration is made into the dynamics of cellular automata in order to answer the question: is their a deep connection between the theories of dynamical systems and computation? Is there an analog for the notion of universal computation in dynamical systems theory? The thermodynamic notion of a phase-transition is proposed as the model in which computation and dynamics commingle. Christopher Langton's thesis, *Chaos at the Edge of Computation*, represents the culmination of the ideas presented within.

This paper represents a single snapshot of a dynamic process. In order to facilitate public access and comments to the current and future versions of this work, it will be made available via the internet at http://www.theory.org/complexity/cdpt/.

Date: 6th June 2000.
Key words and phrases. Cellular automata, complexity, computation, decidability, dynamical systems, Halting Problem, phase-transition, Wolfram classes.

Acknowledgements

I would like to thank the ILL department at the University Library at the University of Michigan for loaning Christopher Langton's PhD dissertation for an amazing five months! His thesis was an unimaginibly powerful source of inspiration and ideas for this paper. Thank you to the UCSC Inter-Library Loan department for facilitating the loan from Michigan.

Thank you Ralph Abraham for advising me on this thesis topic and providing Chris Langton's wonderful video on Artificial Life in Cellular Automata.

Thank you mom and dad, whom without, this experience would not have happened.

CONTENTS

LIST OF FIGURES

1. THEORY OF COMPUTATION

1.1. Deterministic Finite Automata. Having a simple construction and easy to grasp definition, deterministic finite automata (DFA) are one of the simplest models of computation. When interconnected, discrete finite state automata have the ability to exhibit interesting dynamical behavior. These characteristics make DFA an ideal unit with which to build a system suitable for studying the connection between dynamical behavior and computational ability.

1.1.1. *Informal Description.* A deterministic finite automaton can be viewed as a simple machine consisting of three components: sensor, memory and instruction book.

The automaton sensor is the device by which data is input. For example the automaton could have a thermometer reading temperature, a magnet reading a hard drive, an antenna reading radio waves, or anything else that would import information. We restrict the accuracy of the imported information by allowing the automaton to recognize only a finite number of different input values or *symbols*. For example, if a discrete finite state automaton was taking values from a thermometer as input, we could restrict its recognition of temperatures between $0°$ and $100°$ in steps of $0.1°$.

The automaton memory has the limited capacity to store only a single symbol. The type of symbols that can be stored need not be the same as those being returned by the automaton sensor. We refer to this stored symbol as the automaton's *state*. We allow the automaton to exist in only a finite variety of different states.

The automaton instruction book is the set of rules by which the automaton changes its state. The rulebook dictates what state the automaton should be in based upon the automaton's current state and input symbol. The automaton consults this rulebook everytime an input symbol is read.

An automaton begins operation in a predefined state known as its initial state (or start state). The automaton then enters into this loop:

$$\text{read input symbol} \longrightarrow \text{consult instruction book} \longrightarrow \text{transition to new state}$$

Because operation continues until the input source has been exhausted, it is often practical to place a finite limit on the amount of information that may be made available to an automaton for input.

That an automaton may operate on a finite set of input data and end in a discernible final state motivates us to view the automaton as a mapping between the set of input symbols and its final state. This view is further enriched by naming some of the automaton's possible states *accepting* states. This allows us to say that an automaton either accepts or rejects its input data set based upon its final state.

Corollary 1. *Let A be an alphabet, $s^n \in A$ an n-letter string over A. Then a DFA is binary valued function M taking as input a finite set of input symbols:*

$$M : s^n \longrightarrow \{0, 1\}$$

Automata as binary valued mappings highlights their canonical use: to determine whether or not a string of symbols meets some particular criteria. The type of criteria that a finite automaton may check for is called the automaton's language[1]. This language is limited by the automaton's single symbol memory.

1.1.2. *Formal Definition.* We use J. C. Martin's [9] notation as our basis for

Definition 1. A *finite automaton* is a 5-tuple $(Q, \Sigma, q_0, \delta, A)$ where

Q is a finite set of states.
Σ is a finite set of input symbols.
$q_0 \in Q$ is the initial state.
$A \subseteq Q$ is the set of accepting states.
$\delta : Q \times \Sigma \to Q$ is the state transition function.

[1]The language recognized by finite state automata is known as a regular language. Regular languages appear as the simplest language in Chomsky's language hierarchy. For more information see J. C. Martin's book [9].

1.1.3. *Counting Transitions.* The transition function acts as an automaton behavioral rulebook. Based upon the automaton's current state and input symbol, the transition function dictates what state the automaton should next be in. Thus, a useful question to ask when studying the behavior of any finite automaton (FA) is: How many unique transition functions can I define for a particular FA?

Above, we defined the transition function for an FA as $\delta : Q \times \Sigma \to Q$. Q is the set of states the FA can be in. Σ is the set of symbols the FA can accept as input. By dissecting this definition, we shall count how many different ways there are to define the function δ.

The domain of the function says that our input is in $Q \times \Sigma$. Since Q and Σ are finite, the number of possible inputs is $|Q \times \Sigma| = |Q| \cdot |\Sigma|$. To count the number of unique ways δ can be defined, we need to count the number of different ways we can specify what output any particular input could have. The range of δ tells us that any input can have $|Q|$ possible outputs. In other words, for some $q_i \in Q$ and $\sigma_j \in \Sigma$:

$$\delta(q_i, \sigma_j) = \begin{cases} q_1 \\ q_2 \\ \vdots \\ q_{|Q|} \end{cases}$$

Thus, we have a choice of $|Q|$ different transitions that can occur for each of the $|Q| \cdot |\Sigma|$ inputs, or, $|Q|^{|Q||\Sigma|}$ possible transition functions. We record this result as

Proposition 1. *For any given FA,* $M = (Q, \Sigma, q_0, \delta, A)$*, the number of uniquely definable transition functions is*

$$\tau = |output|^{|input|} = |Q|^{|Q||\Sigma|}.$$

This counting problem can be reformulated as: For a given alphabet, how many words of a particular length are there? In the previous case, our word length was $|Q| \cdot |\Sigma|$ (the number of possible inputs) and our alphabet was the set of output symbols (having $|Q|$ "letters"). Thus, we would be choosing $|Q| \cdot |\Sigma|$ different letters for each position in our word $|Q|$ times. Therefore, there are $|Q|^{|Q||\Sigma|}$ different words that can be written down.

Example. Using our Roman alphabet, we can make 26^4 4-letter words.

This analogy constructs a bijective mapping from the set of possible transition functions to a set of strings.

Proposition 2. *For any given FA,* $M = (Q, \Sigma, q_0, \delta, A)$*, there exists a bijective mapping,* ψ*, from the set of all possible transition functions* $\delta \in \Delta$ *to the set of all strings of length* $|Q| \cdot |\Sigma|$ *over the alphabet* Q*.*

$$\psi : \Delta \longrightarrow Q^{|Q||\Sigma|}$$

1.1.4. *Two Counting Examples.* To get a better understanding of the size of Δ, we shall calculate the number of possible transition functions for a couple of example automata. The important point to remember is that the function τ, having exponential growth, creates a strong bounding condition on the complexity of the automata one can consider for practical, computational exploration.

Example. A binary state binary reader.

For this automaton, $Q = \{0, 1\}$ and $\Sigma = \{0, 1\}$. Thus, the number of unique transition functions we could define for this FA would be $\tau = 2^{(2)(2)} = 2^4 = 16$.

Example. A decimal state Roman alphabet reader.

For this automaton, $Q = \{0, 1, 2, ..., 7, 8, 9\}$ and $\Sigma = \{A, B, C, ...X, Y, Z\}$. Thus, the number of unique transition functions we could define for this FA would be $\tau = 10^{(10)(26)} = 10^{260}$.

1.2. **Turing Machines and Universal Computation.** What follows is a brief description of Turing machines and universality. The goal of this section is to describe what is meant by the phrase "universal computation". We shall avoid an in-depth discussion of Turing machine construction and operation as well as most of the interesting theories that surround and grow out of them. For a proper treatment of this most fascinating material, J. C. Martin's book [9] is recommended.

1.2.1. *Brief Overview*. Like a finite automaton, a *Turing machine* is a simple computing device consisting of a finite set of input states, finite set of input symbols, an initial state, and a transition function. We choose at least one of the input states to be special, calling it the *halt state,* and define our transition function such that there are no more possible transitions once the machine reaches the halt state. Unlike a finite automaton, a Turing machine's input data is said to be given on a finite, but unbounded, strip of magnetic tape. A tape head connected to the Turing machine is placed at the beginning of this input data and has the ability to move forwards and backwards along the tape. In addition, Turing machines have a finite set of tape symbols which they can both read and write to the tape via the tape head.

Dr. Alan Turing wrote a thesis which argued that Turing machines are as powerful as the most powerful computing device possible. In other words, no matter how fast or complex our computing devices become, they will have no more capabilities (power) than a Turing machine.

A special form of Turing machine is the *universal Turing machine.* Universal Turing machines have the ability to simulate any other Turing machine when given a description and set of input data for that other Turing machine.

One view of the relationship between Turing machines and universal Turing machines is that a Turing machine is like a single computer program and a universal Turing machine is like an operating system. In this view, a Turing machine is a defined algorithm acting on input data. A word processor, taking text input and outputting a formatted document, is a good example of a Turing machine. On the other hand, a universal Turing machine has the ability to "run" any other Turing machine along with its associated input. This is just as our home computers have the ability to simulate any other computing device that has been encoded as software. A prime example is running software that simulates a pocket calculator or word processor.

When we say a computational device is *universal,* we typically mean that the device has the capability to simulate any other computational device. A common method for proving a device is universal is to show that it is capable of simulating another device already known to be universal. For example, if you were to show a system capable of simulating the components necessary and sufficient to build a computer (gates, timers, etc.) then you would have shown that system to be a universal computer.

1.3. **Decidability and the Halting Problem.** One very important result obtained by Dr. Turing was proof that there exist completely deterministic processes for which it is impossible to decide whether or not they will complete. In terms of Turing machines, his proof implies that given a Turing machine and some input data, it is impossible to decide whether or not that Turing machine will halt. This result should not seem so surprising given the existence of such problems like the Three-Body Problem wherein you know all the details of the mathematical algorithms used to compute the system's evolution, yet are unable to give a general solution for how the system will evolve. One could say that there is a certain sensitivity to the system's initial configuration. This notion will be further explored in section 3.2.

In the course of the following Halting Problem proof and discussion there may be unfamiliar notation. We take the time now to describe what is meant:

Notation 1. If T is a Turing machine, then $e(T)$ is that Turing machine encoded in a format readable by the Turing machine receiving it as input. This is how we pass the description of one Turing machine to another.

Notation 2. The symbol Δ indicates a blank position on a Turing machine tape.

Notation 3. An underscore underneath another symbol, as in $\underline{\Delta}10110$, indicates the position of a Turing machine tape head. We use this symbol because it is customary for a Turing machine to end its processing with tape head positioned on the blank symbol before the beginning of its output data.

Notation 4. We say a Turing machine *halts,* if, in the course of processing its input data, the Turing machine enters into the halt state and has no further transitions.

We first give proof of the Halting Problem by constructing a machine capable of solving the Halting Problem and then showing that if such a machine existed it would generate a paradox. This proof is very similar in flavor to Cantor's diagonalization arguments.

Theorem. *The Halting Problem is undecidable.*

Proof. Assume the Halting Problem is decidable. Then we can construct a Turing machine, T_H, such that given an input $(e(T), x)$, where $e(T)$ is the description of another Turing machine T and x is the input string for T: running T_H on $(e(T), x)$ produces output "$\underline{\Delta}1$" if T halts on x, or "$\underline{\Delta}0$" if T does not halt on x.

We construct a universal Turing machine, T_U, which takes as input a string $e(T)$, and subsequently runs T_H with input $(e(T), e(T))$. We define T_U such that if T_H outputs "$\underline{\Delta}1$" then T_U should transition to an infinite loop and if T_H outputs "$\underline{\Delta}0$" then T_U transitions to the halt state.

What will happen if we ran T_U with input string $e(T_U)$? This will run T_H on $(e(T_U), e(T_U))$. There are two well defined possibilities for what T_U will do: either it will enter into an infinite loop or it will halt. We look at these outcomes:

T_U loops on $e(T_U) \Rightarrow T_H$ output "$\underline{\Delta}1$" $\Rightarrow T_U$ halts on $e(T_U) \Rightarrow T_U$ halts on $e(T_U) \Rightarrow$ T_H output $\underline{\Delta}0 \Rightarrow T_U$ loops on $e(T_U)$. A paradoxical cycle.

Since both cases end in a contradiction, our initial assumption that T_H is decidable must be incorrect. The Halting Problem is undecidable. □

In this next example, we show that if the Halting Problem was solvable, then we could give a simple solution to the unsolved Goldbach conjecture. The Goldbach conjecture states that any given even number greater than 2 can be written as the sum of two primes. For example, $4 = 2 + 2$, $6 = 3 + 3$, $8 = 5 + 3$, $42 = 19 + 23$, $140 = 23 + 117$.

Corollary. *If the Halting Problem was decidable then we could construct a machine to solve the Goldbach conjecture.*

Proof. Assume the Halting Problem is decidable. Then we can construct a Turing machine, T_H, such that given an input $(e(T), x)$, where $e(T)$ is the description of another Turing machine T and x is the input string for T: running T_H on $(e(T), x)$ produces output "$\underline{\Delta}1$" if T halts on x, or "$\underline{\Delta}0$" if T does not halt on x.

We construct a Turing machine, T_G, which determines whether or not an input even number greater than 2 can be written as the sum of two primes. This is a simple computable process which can be described as follows:

(1) Copy the input number to a new place, call it x.
(2) Create a second number $y = 0$.
(3) In a loop do the following:
    (a) Subtract one from x and add it to y.
    (b) If x is less than half of the input number then we have exhausted all possible prime number candidates and the input number fails the Goldbach conjecture: output "$\underline{\Delta}0$" and halt.
    (c) If x and y are both prime then the input number is the sum $x + y$ and it passes the Goldbach conjecture: output "$\underline{\Delta}1$" and halt.

The above process could easily be implemented in a variety of programming languages on a home computer, so we need not worry whether such a Turing machine could exist.

We construct another Turing machine, T_{test}, which runs T_G. T_{test} first gives T_G the number 4. T_{test} then enters the following loop:

(1) If T_G should output "$\underline{\Delta}0$" then halt.
(2) Run T_G with the next higher even number.

Notice, if T_G should never find an even number that fails the Goldbach conjecture, then T_{test} will run forever. A prime target for the Halting Problem.

Run T_H on T_{test}. If T_H produces "$\underline{\Delta}1$" then we know the Goldbach conjecture failed for some even number; the Goldbach conjecture must be false. If T_H produces "$\underline{\Delta}0$" then we know that T_{test} runs forever; the Goldbach conjecture must be true.

Since all machines used must exist, we must be able to create the above composite machine to solve the Goldbach conjecture. $\square$

Proofs of this sort give an intuitive argument as to why the Halting Problem is undecidable.

2. Theory of Cellular Automata

Our discussion of finite automata outlined the workings of a class of simple machines. Though we did not study their behavior, when given carefully selected transition functions and placed into particular nested configurations, these simple machines can exhibit the full range of dynamical behavior: fixed point, periodic and even chaotic [4]. Since one of our aims is to look at whether or not an analog of universal computation exists in dynamical systems (and if so, what that analog is), we find that finite automata might provide a simply constructed system ideal for study. In fact, as we shall see, it is possible to generate a system of finite automata that not only exhibit the full spectrum of dynamical behaviors but are also already known to support universal computation!

2.1. A System of Automata.

Imagine a lattice with a finite automaton residing at each lattice site. Each automaton is designed so that its current state is visible to its own input sensor and the input sensor of the surrounding automata. Each automaton shall take as input the state configuration displayed by its local neighborhood.

Typically, a CA neighborhood is defined as an automaton and its immediate neighbors. The two most common neighborhood templates used for a two-dimensional lattice are:

five cell neighborhood   nine cell neighborhood

FIGURE 1. Two most common 2-D CA neighborhood templates.

Since the automata are using state configurations of a given template as their input source, the set of all possible configurations for a given template is isomorphic to the set of input symbols that the automata need recognize.

To simplify construction of CA systems, the following conventions are universally employed:

(1) All FA are defined to use the same set of state symbols Q.
(2) All FA use the same neighborhood template N.
(3) As a consequence of 1 and 2, all FA will be recognizing the exact same set of state configurations. Thus, it its natural to define all automata to use the same input symbol alphabet Σ.

Typically, an FA's accepting states is ignored since the goal of a cellular automaton is not to accept or reject an input set, but to process an initial configuration of start states. This leaves the last, arguably most important, ingredient in a CA, the transition function.

Traditionally, the transition function is defined identically for all FA. This effectively makes all FA in a cellular automaton duplicates of an archetype FA. This criteria is known as *uniformity*. Most of the research and applications involving CA have been with uniform CA. Recently, some interest has shifted to non-uniform CAs since it has been shown that they can solve problems that uniform CAs can not [10].

Time is spent simplifying the CA for study because a cellular automaton having as few parameters as necessary allows for a high-level search for behavior indicative of computational ability.

Summary 1. Our current system places a finite automaton at each site in a D-dimensional lattice. All automata have an identical set of states, use the same neighborhood template, recognize an identical set of input symbols and behave using the same transition function.

A valid concern might be that the system created thus far is too limited. Limited, possibly, to the extent that universal computation may in fact be impossible. However, the cellular automaton model constructed thus far is the same as that used in Conway's "Game of Life"[2]. That the Game of Life has been proved to be capable of universal computation is our insurance that such a limited system is not *too* limited.

Note 1. The Game of Life is one particular choice of transition function for a two-dimensional, binary-state, uniform cellular automata using the 9-neighborhood template. The Game of Life has been proved to be universal.

2.2. **Rule Space.** We calculate the total number of rules (transition functions) that are available to a cellular automata system:

For any neighborhood template, N, there are $|N|$ cells. Each contains a finite automaton of $|Q|$-states. Thus, there are $|\Sigma| = |Q|^{|N|}$ possible state configurations. By Proposition 1, the total number of possible rules is

$$\tau = |output|^{|input|} = |Q|^{\left(|Q|^{|N|}\right)}.$$

We record this as

Proposition 3. *For any uniform system of cellular automata using the FA* $M = (Q, \Sigma, q_0, \delta, A)$ *and having a neighborhood template* N*, there are* $|Q|^{|Q|^{|N|}}$ *rule sets to choose from.*

Example. The Game of Life is one particular rule choice for two-dimensional, binary-state, uniform cellular automata using the 9-neighborhood template. What is the total number of rules available in this system?

$$|Q|^{|Q|^{|N|}} = 2^{2^9} = 2^{512} \approx 10^{154}$$

[2]The Game of Life is a system well studied for both scientific and recreational purposes. Many resources exist for learning and exploring this system. One such resource is Paul's Page of Conway's Life Miscellany `http://www.cs.jhu.edu/~callahan/lifepage.html`.

2.2.1. *Partitioning.* Even though we have attempted to generate for ourselves a CA system of few parameters, the previous proposition shows us that the size of rule space grows at a phenomenal rate with the number of states and template size being used. Even if we were content to study just a single system, we are left with a large, undifferentiated space of CA rules [6, 7].

Our initial goal should then be to impose a structure on the space of rules that generates a nice ordering on which we could apply an index (a parameterization). The most ideal structure would partition the space of CA rules into regions correlated with the various dynamical behaviors observable in CA. By then probing the regions that exhibit computational abilities, we can begin to draw connections between the ability for computation and associated dynamical behaviors

We introduce the parameterization employed by Dr. Christopher Langton during his artificial life explorations of CA space [8].

Definition 2. The λ parameter. For a uniform, $|Q|$-state CA using neighborhood template N and FA $M = (Q, \Sigma, q_0, \delta, A)$, let $s_q \in Q$ be known as the *quiescent* state and let δ contain n transitions to s_q. Let the remaining $|Q|^{|N|} - n$ transitions in δ be filled by picking randomly and uniformly over the set of states $Q \setminus \{s_q\}$. Then we say

$$\lambda = \frac{|Q|^{|N|} - n}{|Q|^{|N|}}.$$

To get an idea of how the λ parameter correlates with the generated rule set, note the following particular values for λ:

$\lambda = 0.0$	All transitions in δ are to the quiescent state s_q.
$\lambda = 1.0$	No transitions in δ are to the quiescent state s_q.
$\lambda = 1.0 - 1/\|Q\|$	All states are equally represented.

We use the λ parameter precisely because it is deficient with respect to surveying the finer structure of CA space. This blurs the space we must search in, allowing us a low resolution survey in which we can identify the interesting areas for future, sharper exploration.[3]

2.2.2. *Searching & Restricting.* That $\lambda = 0.0$ represents the most homogeneous rule sets and $\lambda = 1.0 - 1/|Q|$ represents the most heterogenous rule sets gives some indication that λ is providing the structure we need to effectively search CA space. In fact, since these values represent end points in the spectrum of rule choice, we shall restrict our search of CA space to those rules having λ values lying within this range.

Searching rule space is the process of stepping through the range $0.0 \leq \lambda \leq 1.0 - 1/|Q|$ in discrete steps, and, at each sampled point, characterizing the CA behavior resulting for δ, the rule set randomly generated with respect to λ. The goal of this process is to describe the relation between λ value and CA behavior. From these results comes the ability to focus in on λ values displaying interesting dynamical behavior and possibly harboring computational ability.

In order to make the studies of CA more tractable, there are a couple of further restrictions that are commonly placed on how δ should be defined.

Definition 3. *Quiescence restriction***.** Any neighborhood that is uniform in cell-state s_q, the quiescent state, should map to s_q.

A stronger form of this restriction is known as

Definition 4. *Strong quiescence.* A neighborhood uniform in any state, $s_i \in Q$, should map to s_i.

Almost universally observed in CA research is the restriction known as

Definition 5. *Spatial isotropy.* All planar rotations of a neighborhood-state will map to the same state.

This has the effect that the local dynamics is unable to make use of the global property of orientation [6].

[3]Chris Langton notes that H. A. Gutowitz has defined a hierarchy of parameterization schemes in which λ is the simplest. References are given in his PhD Thesis [6] and Physica D article [7].

2.3. **A Relationship with Dynamical Systems.** The study of dynamical systems involves the analysis of *phase space*. Phase space is the space of all possible values that a system's variables can take on. Each point in phase space represents particular values for all of the system's variables and we refer to this point as the system's *state*. A system's phase space (*state space*) covers all of the states in which a system could possibly be in.[1]

Example. A thermometer has a one-dimensional state space which can be mathematically idealized as the real number line. Each point on this line represents a particular state that the thermometer could be in corresponding with measured temperature. Note that this particular state space would have a practical lower bound at the point marking absolute zero.

Example. Relative to the center of Earth, every body has a 6-dimensional state space which can be represented by a 6 dimensional real vector space. Each point in this vector space represents a particular state that the body could be in corresponding with its position and velocity relative to the center of Earth. Three axes could be devoted to position, (x, y, z), and three devoted to velocity, (v_x, v_y, v_z).

As a system changes over time its variables change value. As variables change value, the state point forms a trajectory path in state space. Dynamical systems theory has been interested in generalizing system behavior through studying the geometry of the long term trajectories. Discovered is that there are three distinct behaviors any system may evolve into:

(1) *fixed point attractive*: Behavior stabilizes to a single point in phase space that is unchanging over time. In general, there is a clear correspondence between system start state and long term behavior; similar start states producing similar state path trajectories.
(2) *periodic attractive*: Behavior stabilizes into a closed path. In general, there is a clear correspondence between system start state and long term behavior; similar start states producing similar state path trajectories.

(3) *chaotic attractive*: Behavior never seems to stabilize, however the subspace in which the state path trajectory moves is restricted to a bounded manifold. This manifold often possesses complex, detailed structure. In general, similar start states do not produce similar state path trajectories. This last property is called *sensitivity to initial conditions.*

We say that the behavior is attractive because we take the view that there is some sort of *attractor*, or *potential force*, towards which the state path is evolving.

Example 1. A thermometer, initially at room temperature, is heated. Over time, the thermometer cools down to room temperature again. In phase space, this cooling process would appear as a point on the real number line sliding towards the value representing room temperature. We say that the state path trajectory is being attracted to the point representing room temperature. What would the attractor look like?

Fixed point attractive systems are attracted to a *fixed point attractor*, periodic attractive systems are attracted to a *periodic attractor* and chaotic attractive systems are attracted to a *strange attractor*. (See Figure[4] 2).

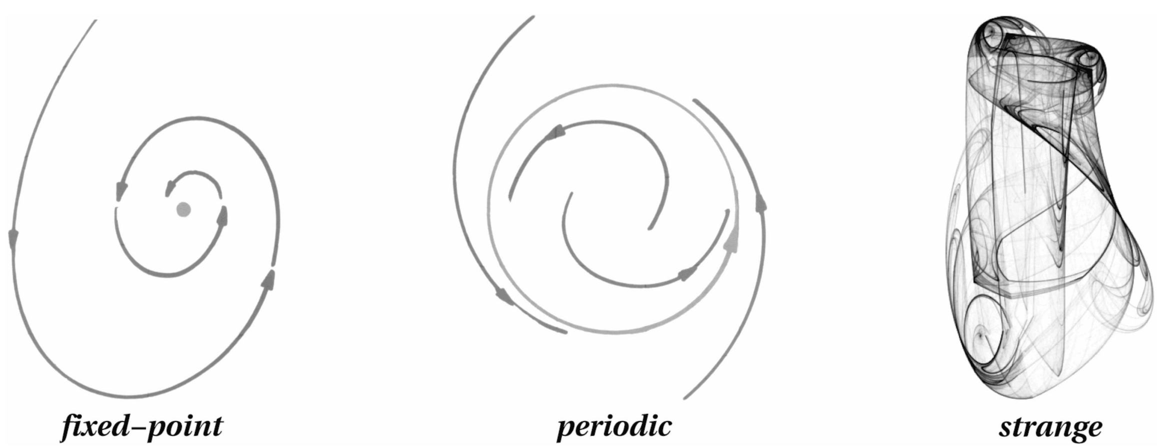

FIGURE 2. Three classes of attractor.

[4]Fixed-point and periodic attractor images were borrowed from *Dynamics: The geometry of Behavior* [1]. Strange attractor was generated with the Theory.org *BFE software suite [11].

We are interested in studying how the behavior of cellular automata correlates with the choice of rule set and whether or not there is a sub-rule-space that lends itself to computational ability.

The obvious analogue for studying a dynamical system phase space is studying a cellular automata state space. We described the state of a dynamical system as the point in phase space representing the current values for all of its variables. In a similar manner, we can describe the global state of a CA by its current total configuration of cell states. This makes the CA phase space the set of all possible global configurations. Thus, the time evolution of a CA could be studied by observing the state-path trajectory in this set of global configurations. The natural question following the this analogy is: What kinds of dynamical behavior do cellular automata exhibit?

2.3.1. *Wolfram's Qualitative CA Classes.* In 1984, Stephen Wolfram undertook a detailed study of one-dimensional cellular automata and their relationship to dynamical systems [13]. His study identified four distinct classes of cellular automaton behavior (see Figure[5] 3):

- **Class I:** From almost any initial configuration, cellular automata evolve to a homogeneous state after a finite number of time steps. Cellular automata in this class exhibit the maximal possible order on both global and local scales.
- **Class II:** Cellular automata usually evolve to short period structures. Local and global order is exhibited, although not maximal.
- **Class III:** Evolution of cellular automata from almost all possible initial states leads to aperiodic patterns. After sufficiently many time steps, the statistical properties of these patterns are typically the same for almost all initial

[5]Images were generated using the "elementary CA" form at Santa Fe Institute's Artificial Life OnLine website [2].

configurations. Cellular automata in this class can exhibit maximal *disorder* on both global and local scales.

Class IV: Yields stable, periodic and propagating structures which can persist over arbitrary lengths of time. By properly arranging the propagating structures, final states with any cycle length may be obtained. The cellular automata in this class exhibit a great deal of local order.

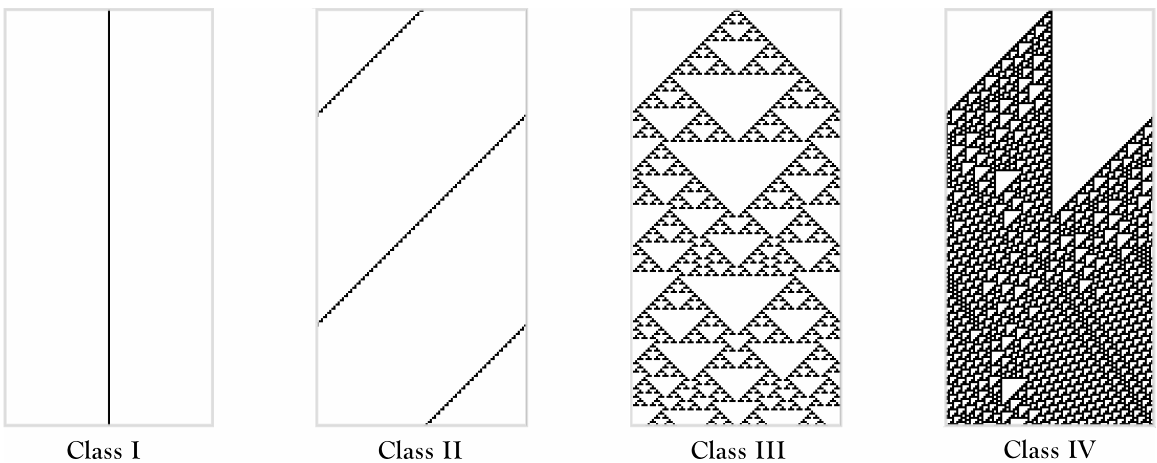

FIGURE 3. Wolfram class examples. Each image a time-evolution of a 2-state CA on a circle using neighborhood size of 3 (central cell and its two neighbors).

For each CA class, Wolfram described the behavior in terms of dynamical systems theory:

Class I	Cellular automata evolve to *limit points*.
Class II	Cellular automata evolve to *limit cycles*.
Class III	Cellular automata evolve to *strange attractors*.
Class IV	Cellular automata exhibit very long transient lengths, having no direct analogue in the field of dynamical systems.

2.3.2. *Length of Transients.* One way of probing a system's behavior is to put it into a state of non-typical behavior and watch the resulting behavior as the system moves towards its attractor. This interim behavior, between perturbed and settled behavior, is known as the *transient behavior*. Of interest is how the transient times change with regard to the size of the system under study.

For fixed-point and periodic systems, the effects of perturbations are restricted within a finite region of the FA array. In other words, beyond this region, the other cells in the array feel neither direct nor indirect effects of the perturbation. These finite regions are typically small and independent of the array size. The relation between the size of a cellular automaton and the average length of its transients is summarized:

fixed-point systems: Transient length is independent of system size.

periodic systems: Transient length is dependent on system size taking on a number of different forms related to how long it takes for localized regions to settle.

chaotic systems: Perturbations are quickly randomized, making their effect on local regions independent of system size. This would be akin to throwing a rock into a pond during a rain storm.

For systems in between, transients show a dependence on system size taking on a variety of functional forms. It is also possible for transients in this dynamical system regime to become effectively infinite — the system spends virtually all its time on a transient.

2.4. **Computation in CA.** "Cellular automata can be viewed either as computers themselves or as *logical universes*, within which computers may be embedded."[6]

As "computers themselves", cellular automata use their starting configuration as the data which their transition function is to process. This means the function δ is representing the algorithm being computed for the input data.

As "logical universes within which computers may be embedded", a cellular automaton's starting configuration constitutes a computer with input data and algorithm. Suitable initial configurations can specify arbitrary algorithmic procedures

[6]Chris Langton, Physica D 42, p. 15. [7]

and input data, thus the system serves as a general purpose computer. In this scenario, the CA transition function is acting as the "physics" under which this general purpose computer will operate.

The Game of Life is a concrete example that CA can support universal computation. However, that CA can be functionally universal has been known since their invention by Ulam and von Neumann in the 1940s. Von Neumann invented cellular automata because he was searching for a mechanism by which machines could self-reproduce. In his existence proof of such self-reproducing machines, von Neumann demonstrated the existence of a universal computer using a two-dimensional, 29-state cellular automaton with the 5-neighborhood template. From this proof stemmed subsequent proofs of automata with much fewer states supporting universal computation.

The technique for showing a system universal general involves constructing the necessary and sufficient parts of a computer inside of that proposed system. Some of these proofs involve the construction of Turing machines, other involve the construction of the more modern notion of a stored-program computer. An example of this latter technique was recently given when the spatialized prisoner's dilemma was shown to be universal. An appropriate choice for dilemma payoff matrix allows for the construction of a virtual Minsky register machine [5]. A Minsky register machine is a general purpose electronic computer.

What sort of factors or criteria are needed for a system to be capable of universal computation? It is the dynamics of the transition function "physics" that provides the conditions for these constructive proofs of universality. In particular, three fundamental features have been determined as necessary and sufficient for these constructions:

(1) The dynamics must support the *storage* of information. This entails the ability for local regions of state information to be preserved for arbitrarily long times.

(2) The dynamics must support the *transmission* of information. This entails the ability for small (local) regions of state information to propagate over arbitrarily long distances.

(3) Stored and transmitted information must be able to interact, resulting in a possible *modification* of one or the other.

Taken together, we see that a dynamical system must be able to correlate storage and transmission abilities (condition 3) and that this correlation can be arbitrarily long (conditions 1 & 2). Like a Turing machine tape, these correlation lengths should be finite, but unbounded.

In his development of the CA classification, Wolfram suggested that Class IV CA are capable of supporting computation, noting similarities between structures observed in his studies and the Game of Life [13]. Langton theorized that it is the association between Class IV CA and very long transients that explains their computational capacity and the difficulty to characterize their behavior.

2.5. **A Search of CA Space.** We give a description of one of Langton's [6, 7] explorations of CA space via the λ parameter.

The search took place on a one-dimensional lattice having 128 sites wrapped into a circle. Thus, the time evolution of this system would be best displayed on a cylinder; each additional time-step being, conventionally, placed below the last. The lattice is uniform with 4-state FAs using a neighborhood of size 5 (the central cell, two cells on its left and two cells on its right). Each CA was initialized in on of two ways:

(1) A uniform, random configuration of initial states is given to all 128 cells.

(2) On a quiescent background, a uniform, random configuration of non-quiescent initial states is given to the center 20 cells.

The search produced the following primary dynamical features (see Figure[7] 4):

[7]Images were generated using the "lambda CA form" at Santa Fe Institute's Artificial Life OnLine website [2].

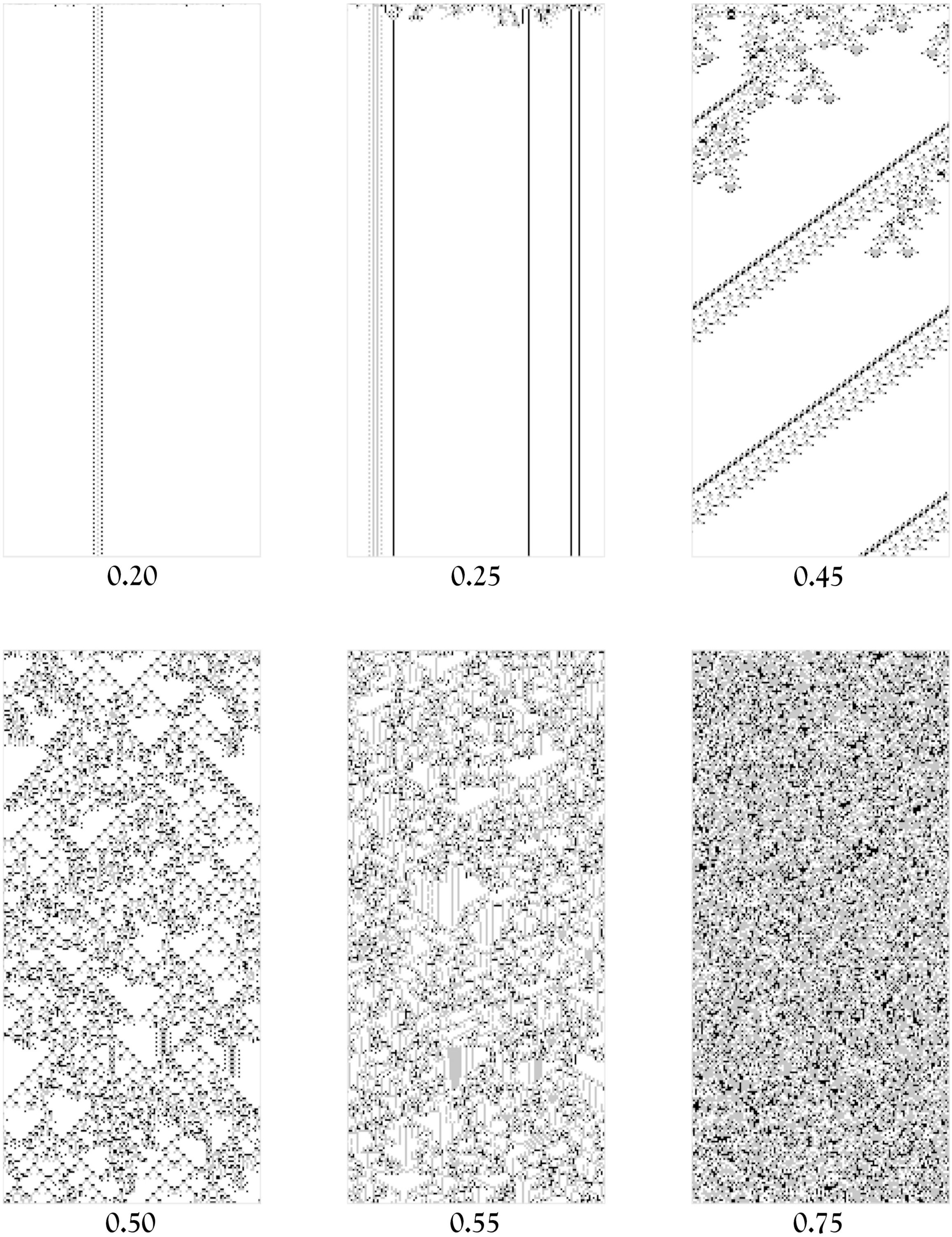

FIGURE 4. For selected λ-values, a 256 step time-evolution of a corresponding CA.

$\lambda \approx 0.00$ After a single time-step, the array is uniform in s_q. Systems are characterizable as immediately decaying to a fixed-point attractor.

$\lambda \approx 0.15$ Decay to uniformity in s_q takes between 4 or 5 time-steps. We refer to this activity as the *transients*, since it precedes the "typical", long-term behavior.

$\lambda \approx 0.20$ Systems can now also produce potentially permanent periodic structures. Transient length is between 7 and 10 time-steps.

$\lambda \approx 0.25$ Systems can now also produce cells that get stuck in a single, non-quiescent state. Langton referred to these as period 1 structures.

$\lambda \approx 0.40$ Periodic structures are observed with periods of up to 40 time-steps. Transient length has increased to 60 time-steps. Globally, systems can be characterized as collapsing into isolated areas of periodic activity.

$\lambda \approx 0.45$ Transient length has gone through a dramatic increase to almost 1000 time-steps. Globally, systems appear to be near a balance between collapsing and expanding dynamical activity. Being near this balance seems to give systems the ability to produce propagating structures. These propagating structures may appear, on first look, to be period 100 structures. However, extended observation of the cycling allows one to notice that a whole structure is slowly gliding around the lattice circle. Thus, the true period length of this structure includes the time it takes to orbit around the circle until reaching the initial location and configuration. In Langton's study, the structure under his observation had to orbit 3 times before repeating exactly. Taking 116 time-steps to shift 3 sites, the true period of this structure was $\frac{128}{3} \cdot 116 \cdot 3 = 14,848$ time-steps! Systems generated with respect to this λ value do not *necessarily* produce these propagating structures.

$\lambda \approx 0.50$ Transient length is dramatically increasing; now on the order of 12,000 time-steps. It is unclear whether or not the global behavior is expanding or contracting. Langton hypothesizes an overall expansive nature that is eventually destroyed by the wild fluctuations of dynamic activity, leaving periodic structure residue.

$\lambda \approx 0.55$ Transient activity has become so long that *it* can be considered to be the "typical", long-term behavior. In this new dynamical regime, long-term behavior now tends to chaotic behavior.

$\lambda \to 0.75$ Transient activity decreases as λ increases. Again, this means that the time it takes for the system to reach is "typical", long-term behavior has become increasingly shorter. We say that a chaotic system has reached its "typical", long-term behavior when the density of occupied lattice sites is within 1% of the long-term average. When $\lambda \approx 0.75$, systems become "typically" chaotic after only a single time step. Compare this to $\lambda \approx 0.65$ where it takes approximately 10 time-steps. In other words, systems are characterizable by very quick decays to a strange attractor.

By varying the λ parameter, we are able progress through CAs exhibiting maximal possible order to CAs exhibiting maximal possible disorder. At either end of the spectrum, behavior can be characterized as quickly decaying to its long-term attractor. In the middle of the spectrum, behavior becomes very unpredictable with long transients preceding a break down to long-term behavior. Also, we were able to witness a very rapid increase in transient length. This ultimately led to a new regime of dynamical behavior which had its own transient lengths very rapidly decrease as λ increased. This dynamical regime separating the periodic regime from the chaotic regime is known as the *transition regime*.

The transition regime is able to support propagating structures (as discussed above for $\lambda \approx 0.45$). These structures are very much like the "gliders"[8] seen in the Game of Life. It is interesting to note, that for CAs using the parameters of the Game of Life, the λ value corresponding to the Game of Life's particular rule choice lies within the transition regime. By increasing the lattice size for the above experiment, the existence of several different kinds of propagating structures can be observed. Langton found that the collision between a propagating structure and a static structure produces a structure which propagates in the opposite direction. It is exciting to realize that these interactions can create a basis for transmission, storage and modification of structurally embedded information. Thus, this transition regime seems to harbor the ingredients for constructing a universal computer.

[8]Glider Description `http://www.brunel.ac.uk/depts/AI/alife/al-osc.htm##glider`

3. PHASE-TRANSITIONS AND COMPUTATION

3.1. **The Phase-Transition.** As discussed in the previous section, there are two primary regimes of rules, *periodic* (Class II) and *chaotic* (Class III), separated by a third, *transition regime* (Class IV). Schematically, this appears as Langton's famous egg diagram:

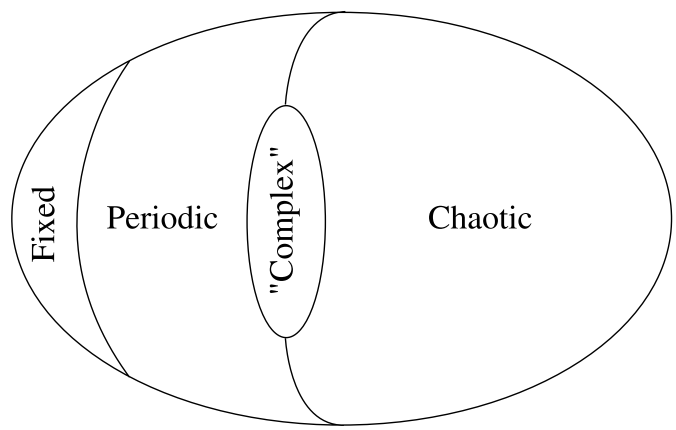

FIGURE 5. Schematic drawing of CA rule space indicating relative location of periodic, chaotic, and "complex" transition regimes.

The transition regime has proven to be made of a complicated structure. In some experiments with the λ parameter, the transition regime acts as nothing more than a boundary line between periodic and chaotic behavior. Crossing over this line (increasing or decreasing λ) produces a discrete jump between behaviors. Quantitative analysis shows a discrete jump in statistical measures[9] of the kind usually associated with *first-order transitions*[10]. In other experiments with the λ parameter, the transition regime acts as a smooth transition between periodic and chaotic activity. This smooth change in dynamical behavior was described in the

[9]Langton utilized theories from thermodynamics and information theory to analyze the changes in dynamical activity of evolving CAs as the λ-parameter varied. In particular, complexity was measured using Shannon entropy which Langton first derived from Boltzmann entropy.
Average mutual information was plotted against the single cell entropy. This plot gave away that the transition regime optimized system entropy to handle both storage of information (which reduces entropy) and transmission of information (which increases entropy). The plot took on the same shape that is characteristic of thermodynamic phase-transition graphs; particular, the λ-transition [3].

[10]A transition for which the first derivative is discontinuous with respect to some, usually physical, quantity, such as temperature.

experiment discussed in section 2.5. This type of transition is primarily *second-order*[11], also called a critical transition.

To summarize, giving a qualitative description of CA behavior as the λ parameter is varied produces the full spectrum of dynamical behaviors. Furthermore, the pattern of transient length (a complexity metric) forms a sort of butterfly pattern; the transition regime acting as the butterfly's body. Thus, we can order the dynamical behaviors with respect to their λ-distance from the critical transition regime. This is precisely what one expects to see as one approaches and passes through a second-order phase-transition. Typically, physical systems which pass through this type of phase-transition have transients that diverge to infinity and behavior becomes maximally complex and unpredictable. Thus, Langton hypothesized that it is this phase-transition structure which "is responsible for the existence of most of the surface-level features in the diverse phenomenology of CA behaviors."

Using this notion of a phase-transition, we describe[12] the existence of and relationship between the four Wolfram classes in Figure 6.

3.2. **CA and Computation Space.** A natural question to ask is whether or not this phase-transition structure underlying CA "phenomenology" bares any resemblance to the "phenomenology" of computations. In other words, we would like to find CA behaviors that parallel the behaviors we see in computation. We already know that CAs can form logical universes in which we may construct universal computers. Hence, we look for analogues of computability and complexity.

3.2.1. *Computability.* The phase-transition structure tells us that near a critical λ value, the complexity of CA behavior rapidly increases. What this means is that it becomes much more difficult to characterize the long-term behavior of CAs near this critical value. This is primarily due to the increased length of transients.

[11] A transition for which the second-derivative is discontinuous with respect to some, usually physical, quantity, such as temperature.

[12] This figure emphasizes that any scheme which attempts to partition CA rule space into a small set of qualities will inevitably be only a low resolution rasterization of the underlying, continuous reality.

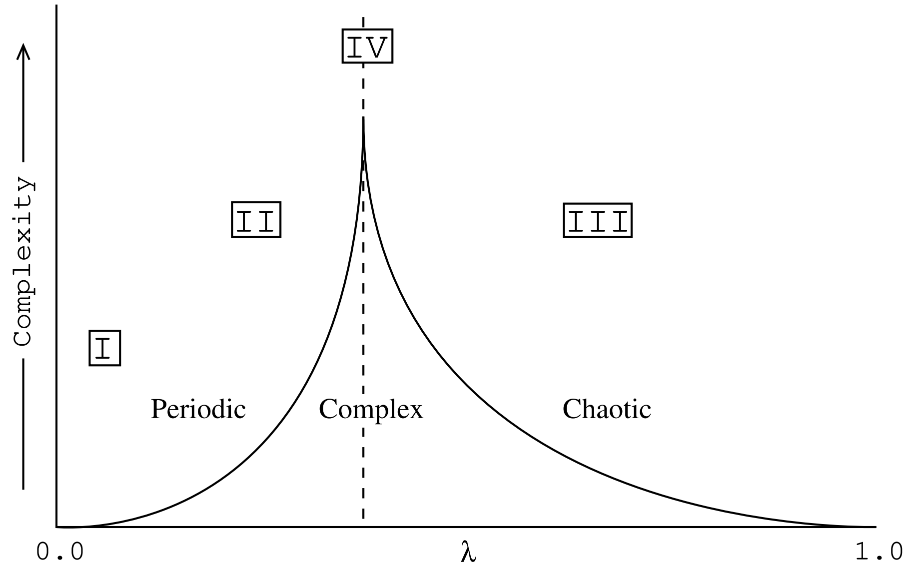

FIGURE 6. Schematic drawing of CA rule space showing the relationship between the Wolfram classes and the underlying phase-transition structure.

For relatively[13] low values of λ, it becomes quickly clear that the system under observation is heading towards long-term periodic behavior. For relatively high values of λ, it becomes quickly clear that the system under observation is heading towards long-term chaotic behavior. However, for λ near the critical value, either of these long-term outcomes is possible. Coupled with very long transient lengths, it becomes effectively undecidable as to whether a given rule set and starting configuration will tend towards periodic or chaotic behavior. A genesis of the Halting Problem[14]!

3.2.2. *Complexity.* Complexity classes in computation refer to the time it takes for a halting computation to complete with respect to the size of its input. Typically, these classes have names such as *constant*, *polynomial*, and *exponential*, referring to the class of algebraic formula representing the relation between input size and computation time. The time it takes for a computation to complete can be thought of as the transient time leading to the fixed-point state of solution. Thus, the

[13]With respect to the critical λ value.

[14]See section 1.3.

complexity classes of computation are analagous to the structures produced in answering the question: Given a rule set and starting configuration, how long does it take for the CA to exhibit "typical", long-term behavior.

3.3. **The "Deep-Structure" of Computation.** We have been using the notion of an underlying phase-transition structure as an analogy for discussing the relation between computational and dynamical behaviors. In fact, what we want to show is that by assuming this underlying phase-transition structure, we are provided with "a simple and straightforward explanation for the existence of, and relationship between, many significant features of the phenomenology of computation"[15]. (See Figure 7).

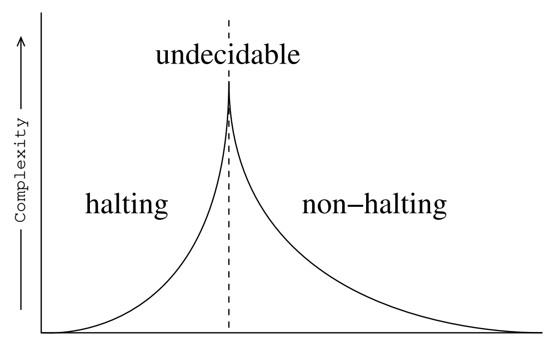

FIGURE 7. Schematic drawing of computation space, indicating relative location of halting, non-halting, and undecidable regimes.

Langton gave the following four arguments for making this assumption:

(1) The existence of computability classes is explained by the separation between ordered and disordered regimes being dominated by a phase-transition.
(2) The existence of undecidability is explained by transients diverging to infinity on approach to a second-order phase-transition. This gives rise to the Halting Problem.
(3) The existence of complexity classes is explained by the divergence of transient length on approach to a second-order phase-transition.

[15]Chris Langton in his thesis, p. 130. [6]

(4) The existence of universal computation is explained by the quality of "susceptibility". Langton describes this quality as self-sensitivity to minute details of internal structure and external perturbations. "This self-sensitivity in the vicinity of a critical transition is manifested in universal computations as their 'programmability'."[16]

Thus, by assuming computation space has an underlying phase-transition structure, we are able to produce exactly those behaviors expected in a study of computational behavior.

3.4. **Some Implications.** Arising out of the existence of a fundamental connection between computation and phase-transitions is a wide variety of beautiful and powerful implications. We briefly discuss the implications introduced at the end of Langton's thesis.

3.4.1. *Phase-Transitions and Information Processing.* There is a fundamental equivalence between the dynamics of phase-transitions and the dynamics of information processing. This means that computation can be thought of as a special case of phase-transition phenomena and that the theory of computation can be used to explain phenomena in the theory of phase-transitions. Thus, we may look to the natural systems around us for emergent, information processing capabilities.

3.4.2. *Material Phases as Dynamics.* We all know the three phases of matter: solid, liquid and gas. However, there exists a *critical temperature* at which the surface between the gas and liquid phases disappears. The resulting matter is known as a *supercritical fluid* [3]. Thus, we may say that there exists two phases of matter: solid and fluid. What is very interesting is that even though the CA experiments existed in a non-physical substrate, we still ended up with two primary regimes of behavior: the static, *solid*, dynamics of fixed-point and periodic structures and the non-static, *fluid*, chaotic dynamics. Langton hypothesizes that

[16]Chris Langton in his thesis, p. 129. [6]

the categories, solid and fluid, are not merely material qualities but instead are dynamical qualities separated by a critical transition.

If the phases of matter can indeed be generalized as dynamical qualities, then the difference between a natural and artificial system seems less real. For, if the behavior of matter can be reproduced in a computer world "then it is only a matter of organization to turn 'hardware' into 'wetware' and, ultimately, for hardware to achieve everything that has been achieved by wetware, and more."[17]

3.4.3. *Dynamics and Life Support.* If we were to look for a beginning of life, we are tempted to focus on the phase-transitions that were occurring on pre-biotic earth. From them would have emerged information processing abilities. Those abilities may have become coupled with one another to form complex, meta-stable information structures in the physical substrate of earth (the primordial "soup"). Those structures which gained a local control over their parameters survived, while others were pushed passed the event horizon of an attractor. In this sense, the "living" systems are precisely those systems that are able to *avoid* attractors. Evolution is then a repeated iteration whereby a system climbs out of one attractor into a higher dimensional phase-space, only to find itself heading for a higher dimensional attractor; a Red Queen[18].

Looking at the processes and structures of living cells, we see the same sort of phase-transition conditions. A particularly good example of this is the dynamics of the olfactory bulb which during rest takes on low-amplitude chaotic activity and during inhalation falls into the basin of a periodic attractor. Thus, the olfactory bulb spends a good portion of its time in the transition regime between chaotic and periodic behavior. (See Figure[19] 8).

[17] Chris Langton in his thesis, p. 135. [6]

[18] In Lewis Carol's, *Through the Looking Glass*, the Red Queen is seen running as fast as she can, but does not get anywhere because the scenery moves with her. This notion is taken up later as an analogy to evolution in Matt Ridley's, *Red Queen: Sex and the Evolution of Human Nature*.

[19] Image from Theory.org's *Chaos and Neurodynamics* [12].

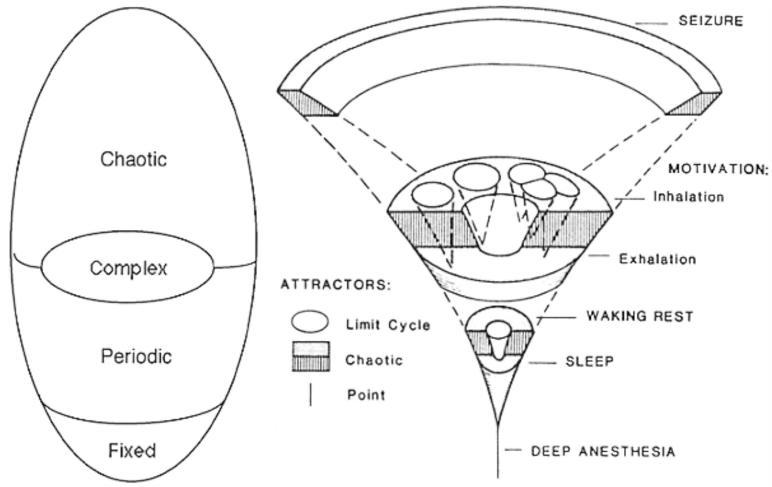

FIGURE 8. Comparison between schematic of CA rule space and bifurcation diagram of the phase portrait of olfactory bulb states.

3.5. **Conclusion.** That a phase-transition structure is underlying CA state space is not just a convenient analogy for describing how behavior varies with the λ parameter. By assuming this structure we are provided with powerful means for explaining the existence of many of the fundamental features of computational theory by allowing them to naturally arise out of a dynamical systems context. Conversely, the dynamical behaviors that are found in dynamical systems near a critical transition can be explained in terms of the computational abilities they may harbor. Thus, we are presented with new ways in which we may analyze and reanalyze both the natural and artificial systems surrounding us.

REFERENCES

[1] R. H. Abraham and C. D. Shaw. *Dynamics: The Geometry of Behavior*. Parts I, II and III. Santa Cruz, CA: Aerial Press, 1982, 1983, 1985.

[2] Artificial Life Online. *Exploring the Space of Cellular Automata*. `http://alife.santafe.edu/alife/topics/ca/caweb/`

[3] P. W. Atkins. *Physical Chemistry*. New York: W. H. Freeman and Company, 1998.

[4] J. P. Crutchfield and K. Young. "Computation and the Onset of Chaos." In *Complexity, Entropy and the Physics of Information*, edited by W. H. Zurek. Redwood City, CA: Addison-Wesley, 1990.

[5] P. Grim, G. Mar, and P. St. Denis. *The Philosophical Computer: Exploratory essays in philosophical computer modeling*. Cambridge, MA: The MIT Press, 1998.

[6] C. G. Langton. *Computation at the edge of Chaos: Phase-Transitions and Emergent Computation*. Ph.D. Thesis, University of Michigan (1990).

[7] C. G. Langton. "Computation at the edge of Chaos: Phase-Transitions and Emergent Computation." *Physica D* 42 (1990) 12-37.

[8] C. G. Langton. "Studying Artificial Life with Cellular Automata." *Physica D* 22 (1986) 120-149.

[9] J. C. Martin. *Introduction to Languages and the Theory of Computation*. Boston, MA: McGraw-Hill, 1997.

[10] M. Sipper. "Evolution of Parallel Cellular Machines: The Cellular Programming Approach." *Lecture Notes in Computer Science* 1194. Berlin: Springer-Verlag, 1997.

[11] Theory.org. *Theory.org Software*. URL: `http://www.theory.org/software/`

[12] Theory.org. *Chaos and Neurodynamics*. `http://www.theory.org/fracdyn/neurodyn/`

[13] S. Wolfram. "Universality and Complexity in Cellular Automata." *Physica D* 10 (1984) 1-35.