in **Towards the Harnessing of Chaos**,
M. Yamaguti, editor, Elsevier Science, Amsterdam (1994).

Santa Fe Institute Working Paper 93-10-061

Critical Computation, Phase Transitions, and Hierarchical Learning

James P. Crutchfield

Physics Department
University of California
Berkeley, California 94720
chaos@gojira.berkeley.edu

Abstract

A study of the various routes to chaos in dynamical systems reveals that significant computation occurs at the onset of chaos. At first blush this is not surprising since statistical mechanics views these as phase transitions with infinite temporal correlations. In computational terms processes that are in a critical state, like those at the onset of chaos considered here, have an infinite memory. Storage capacity, however, is only part of information processing. The set of available nonlinear operations and just how the memory is organized are more important determinants of the class of computation that can be supported. This leads directly to studies of the architecture of information processing and to quantitative measures of complexity. There is a universal theory, for example, that indicates how complexity trades-off against the rate at which information is produced. This result suggests a new view of how learning and control systems can break out of inadequate internal models to discover genuinely new representations.

Proceedings of the 7th Toyota Conference, Lake Hamana, Shizuoka, Japan

1. DYNAMICS AND COMPUTATION

How does a physical device compute? The early history of analog and digital computer design is replete with diverse physical phenomena that implement the elements of formal computation: the objects and operators of differential calculus and automata theory. In light of the wide-ranging success of dynamical systems theory for modeling nonlinear physical processes, the following review asks the contemporary version of the above question — How does a dynamical system compute? The answer given here begins with interpreting computational devices as nonlinear dynamical systems and then explores what types of information processing are embedded in them. The ultimate goal is not only to help build new and better computers, but also to shed new light on the structure of dynamical systems. As a start a few words about dynamical systems and computation are in order.

The mechanical description of a physical system is given in terms of a set of equations that govern the change in system configuration from one time to the next. The dynamical systems reformulation of this considers the set of all configurations — the state space — and the mapping that takes states to states — the dynamic. An initial state under the action of the dynamic leads to a sequence of states — an orbit of the dynamical system. What the theory of dynamical systems contributes to the mechanistic view is an emphasis on how the dynamic induces global geometric structures in the system's state space and how these structures constrain a system's possible behavior. It also provides methods for analyzing behavioral stability and instability and how the constraints on behavior can change if parameters in the equations of motion are varied.

"Computation" in the context of physics and dynamical systems theory has several possible meanings. Unfortunately these more technical senses of the word are not often directly comparable to its contemporary use, which is highly colored by the present dominance of sequential digital computers. In preparation for understanding how dynamical systems might embody information processing, the next section defines several types of computation — usable, universal, and intrinsic computation. Subsequent sections review how several types of intrinsic computation are embedded in nonlinear dynamical systems. It turns out that there is a close relationship between a system being at the onset of chaos and its capacity to exhibit high levels of computational structure. As examples, both period-doubling and quasiperiodic systems are analyzed for intrinsic computation. The appropriate phase transition view of these systems' behavior — the renormalization group — then suggests a novel hierarchical learning method for an observer to discover a finite representation of the infinite memory structures these systems exhibit. The last sections consider the larger issue of how the state space of physical computing devices must be structured for them to support significant computation.

2. EMBODIMENTS OF COMPUTATION

The list of computation types seems almost endless: discrete[1] versus continuous,[2,3] temporal versus spatial,[4] probabilistic versus deterministic, and now classical versus quantum computation[5]. The theory of discrete computation, however, is the most highly developed. Very roughly, it distinguishes a number of computational classes, each capable of finitely representing a range of formal "languages" and each instantiated by a class of automata. There are the finite automata which recognize the "regular" languages using only a finite amount of memory; the

more powerful pushdown automata which recognize the "context-free" languages, akin to modern programming languages, using a last-in-first-out "stack" memory of arbitrary size; and the Turing machines, the most powerful discrete-computation devices, which recognize the "recursive" languages using an infinite storage tape with unrestricted access. Many decades of research comparing variously-structured automata have led to an extensive discrete-computation road map: a hierarchy of distinct classes spanning the range from finite memory devices to Turing machines. Rather than assessing relative capabilities within the single broad type of discrete computation, later sections will compare continuous and discrete computation. The results there give concrete examples of how discrete computation can be embedded in dynamical systems. To prepare for this, the notion of computation itself must be explored in more depth. Three kinds of computation — usable, universal, and intrinsic — will be considered.

2.1. Usable Computation

The most common meaning of "performing a computation" is that a dynamical system carries out some "useful" information processing task. Here, the equations of motion are interpreted as the "program" and the initial state is interpreted as the "input". The system runs for some specified time until it reaches a "goal" state at which it detects the task's completion. This final condition must be relatively easy to detect. It might be, for example, a fixed point state. In any case, some portion of its configuration is interpreted as the "output". There is a correspondence between the computation and the orbit in the system's state space. Examples of dynamical systems performing tasks in this way include integrating a differential equation on an analog computer to compute π, using a cellular automaton to generate the n^{th} row of Pascal's triangle, performing image edge enhancement with video feedback[6] or an oscillating chemical reaction in a petri dish,[7] running a recurrent neural network to "recover" a picture from an initially corrupted version, and — considering the notion of a dynamical system rather broadly — using a pinhole lens to estimate the Fourier transform of an image. (Further discussion of discrete computation in cellular automata can be found in Ref. [8].)

Requiring arbitrary "programmability" of a device is too strict, if one is interested in devices that are computationally useful. The dynamical system $\dot{x} = -x$ and $\dot{y} = 2\sqrt{x^3(2-x)}$, starting with $x(0) = -1$ and $y(0) = 0$, evolves to $y(\infty) = \pi$. Even though it's not programmable and requires specific initial conditions, its generation of π is a "useful" result in the sense that, in a domain of analog computation not allowing for the explicit representation of transcendentals, this information could only be obtained using this or a similar "algebraic" dynamical system as a subprocess.

If a transcendental number is produced using only algebraic operations, then there has been a "computational gain", since transcendentals require more computational resources than do algebraic numbers. The capability that leads to such a gain will be referred to as "usable" computation. "Usable" avoids the sense, implied by "useful", that the available information processing has some demonstrated utility.

2.2. Universal Computation

A second meaning of computation is for a dynamical system, given specially-crafted initial conditions, to be capable of universal computation. The latter is arguably the most general notion of "programmability": any computational task can be implemented on the universal device. That is, for each instance of a task there is an initial condition on which the system simulates, within given resource bounds, any other system of the same computational type. There is another sense of universality more appropriate to the comparison of discrete and continuous computation. This is that a continuous-state dynamical system mimic a programmable discrete-state computer, complete with information storage, logical gates, control mechanisms, and so on. In either case, the initial condition is considered to have two parts: an emulation program that makes the universal device implement the desired task and input data that specifies a task instance. The distinction between the underlying device, a program, and the input data is probably clearest for universal computation.

The Game of Life cellular automaton (CA) on an infinite two-dimensional lattice is such a universal dynamical system. One construction for universal computation in the Game of Life — extremely complicated taken in its entirety, if not impractical — is given in [9]. Simpler constructions have been made for one-dimensional CA.[10] The simplest is to note that any universal Turing machine is a one-dimensional CA: its infinite tape is the spatial lattice with a bit is added at each site to uniquely locate the read-write head. When CA are viewed as dynamical systems, the main requirement for universality is to construct a set of equations of motion that allows one portion of the CA's initial pattern to specify the emulation of a device, with the remainder of the pattern specifying the device's input.

In the domain of continuous-state computation, standard analog computers — widely used up through the '70's — are universal devices. The "patching" of resistors, capacitors, operational amplifiers, and multiplier function modules constitutes an analog computer program. The versatility of possible patches allows analog computers to solve a wide range of integral and differential equations. In fact, many of these devices included voltage-controlled patching to enable self-modification of the "program" during the solution. Thus, the same ambiguity between program and input data that holds for discrete computation also obtains in the analog domain.

It has also been shown that universal discrete computation can be embedded in continuous-state systems. These constructions are based on establishing a "symbolic dynamics" equivalence between each possible Turing machine configuration — internal control state and infinite storage tape — and a continuous state. The equations of motion operating on the continuous states are designed to implement a universal Turing machine's finite state control. In Ref. [11] it was noted that a universal two-dimensional map of the plane can be constructed. A similar construction and more extensive analysis can be found in Ref. [12].

The obvious advantage of universal computers is that only one type of device needs to be built. Then the program portion of the input "reconfigures" the device to the particular task of interest. One disadvantage compared to a specialized device is the reduction in computation speed due to the emulation. Another is the reduction in the volume of system state space available for information processing, which lowers the density of computation. Conventional serial digital computers are examples, as are the universal CA examples just mentioned, of vanishingly small

computational density. The loci of information processing is concentrated in the control unit, while the memory and large portions of the control mechanism sit idle. It is worth noting, in contrast, that the discrete-in-continuous constructions just given use low-dimensional dynamical systems and so very few degrees of freedom to implement universality. This point should be kept in mind for the closing discussion.

2.3. Intrinsic Computation

A third meaning of computation in a dynamical system involves interpreting its behavior, or more properly the orbits it can generate, as a type of "intrinsic" computation. Here computation is not the transformation of an input to produce a "useful" output. Rather, it is measured in terms of elementary information processing structures — memory, information production, information transfer, logical operations, and so on. In other words, intrinsic computation in a dynamical system is an intrinsic property of its behavior that can be measured by an observer just as (say) the dimension of the system's attractor can be estimated. Intrinsic computation can be detected and quantified without reference to any specific "useful" computation performed by the dynamical system in question. In measuring it one looks at the typical information processing over the whole state space or large subsets of it. The equations of motion in this view are thought of as being the computational device in the sense that they determine the constraints that guide the flow of information in the state space. This notion of intrinsic computation is developed in Refs. [11,13,14].

Note that intrinsic computation makes no reference to the human, or other intelligent, manipulation of the input nor to the same interpreting the output. In contrast, to properly define usable computation one needs a theory of the semantic aspects of choosing inputs and interpreting outputs. The result is that intrinsic computation is a simpler and implementable notion of how systems process information. It relates more directly, for example, to how architectures for information processing spontaneously emerge from within evolutionary and developmental systems, since it does not rely on an outside "engineer" to handcraft computational elements, to prepare special initial configurations with "programs" built in, or to interpret the result.

2.4. Intrinsic versus Usable Computation

There is a key difference between computation theory and dynamical systems theory that complicates any comparison or synthesis. (This difference is also responsible for introducing a distinction like "intrinsic" computation.) On the one hand, according to computation theory, information processing devices during their operation do not produce information. Information is preserved though transformed, or it is lost. On the other hand, the more interesting dynamical systems — e.g. those exhibiting chaotic attractors — robustly generate information even though they, like the idealized devices of computation theory, are deterministic.

These considerations lead to the central question concerning the relationship between dynamics and computation — What structures in a dynamical system support the various embodiments of computation?

This question is answered for universal computation via the requirements and constructions noted above. Therefore universal computation will not be addressed further. Instead the

following focuses on the contrast between two views of a physical computing device. From the dynamical systems view, the interest is what intrinsic computation indicates about the computing device's behavior. From the computer science and engineering view, the concern is to know what degree of usable computation the physical device is capable of. Since they concern the same device, the contrasting views naturally suggest the question of how intrinsic and usable computation are related. After addressing the problem in this section, the next two give an answer to the main question as it applies to intrinsic computation in two families of dynamical systems.

Simply stated, intrinsic computation places an upper bound on what computations a dynamical system can support. That is, if a dynamical system $\mathcal{D}$, when observed and controlled with instrument $\mathcal{I}$, has an intrinsic computational capacity $C_{\rm in}$, then its useful computation capacity $C_{\rm us}$ can be at most $C_{\rm in}$:

$$C_{\rm us}(\mathcal{D},\mathcal{I}) \leq C_{\rm in}(\mathcal{D},\mathcal{I}) \tag{1}$$

This is somewhat reminiscent of universal computation, which indicates a "possible" capacity. If a system implements (say) a universal Turing machine then it can perform any discrete computation, even if the required emulation for a given task is extremely difficult to construct and even if the set of initial conditions employed is a vanishing fraction of the total available states. In contrast, the upper bound set by $C_{\rm in}$ indicates a "probable" computational capacity; even though there may be a small set of initial conditions that can be interpreted as leading to higher levels of computation, typically no more capacity is available than $C_{\rm in}$. For example, a dynamical system with an intrinsic computational capacity equivalent to a two-state machine cannot calculate the binary expansion of π, since the latter requires (at least) an amount of memory that grows with successive digits of π. Similarly, a finite memory device cannot check for balanced parentheses in arithmetic expressions. Here a device's capacity is specified as an entire class — $C_{\rm in} \sim$ finite automata — but the task's required capacity is in a class — $C_{\rm us} \sim$ pushdown stack automata — that is strictly more powerful and so violates the bound.

3. INTRINSIC COMPUTATION IN THE PERIOD-DOUBLING CASCADE

The following two sections review how intrinsic discrete computation is embedded in two well-known continuous-state dynamical systems. The connection between discrete computation and the continuous states is made via symbolic dynamics. In this approach a continuous-state orbit is observed through an instrument that produces very coarse, in fact binary, measurements. To detect the intrinsic computation the resulting binary data stream is fed into a modeling algorithm — ϵ-machine reconstruction[13] — to produce a minimal computational model of the data stream. The resulting model, referred to as an ϵ-machine, describes the intrinsic computational capability of the observed process — dynamical system plus instrument. Due to the choice of a particular type of instrument, the ϵ-machine also describes the computational capability of the hidden dynamical system.

The first dynamical system to be analyzed for computational structure is the logistic map and, in particular, its period-doubling route to chaos. The data stream used for reconstructing the model is derived from a trajectory of the logistic map when it is started with an initial condition

on its attractor. This makes the observed process stationary. The trajectory is generated by iterating the map

$$x_{n+1} = f(x_n) \tag{2}$$

with the logistic function $f(x) = rx(1-x)$ in which the nonlinearity parameter $r \in [0,4]$ and the initial condition $x_0 \in [0,1]$. Note that the map's maximum occurs at $x_c = \frac{1}{2}$. The orbit $\mathrm{x} = x_0 x_1 x_2 x_3 \ldots$ is converted to a discrete sequence by observing it via the binary partition

$$\mathcal{P} = \{x_n \in [0, x_c) \Rightarrow s = 0, x_n \in [x_c, 1] \Rightarrow s = 1\} \tag{3}$$

This partition is "generating" which means that sufficiently long binary sequences come from arbitrarily small intervals of initial conditions. Due to this, the information processing in the logistic map can be studied using the "coarse" measuring instrument $\mathcal{P}$.

Many investigations of the logistic map concentrate on how its time-asymptotic behavior, its attractor, changes with the nonlinearity parameter r. Here, however, the interest is in how its information processing capabilities are related to one another. There are two basic measures of this that can be directly taken from the reconstructed ϵ-machines. The first is the statistical complexity C, which is the size of the reconstructed ϵ-machine. It has units of "bits" and, roughly speaking, is measured by taking the logarithm base 2 of the number of ϵ-machine states. In short, the statistical complexity is the amount of memory in the process that produced the data stream. The second measure of information processing is the entropy rate h_μ, which is the rate in bits per time step at which information is produced. The net result of using just the complexity and entropy rate is that the original equations of motion and the nonlinearity parameter are simply forgotten. All that is of interest is how the complexity C of the data stream depends on the rate h_μ of information production.

The complexity-entropy plot of Figure 1(a) summarizes this relationship by showing the results of reconstructing ϵ-machines from data streams produced at different parameter values. For each data set produced, an ϵ-machine is reconstructed and its statistical complexity C and entropy rate h_μ are estimated. The latter is estimated as $h_\mu(L) = H(L)/L$ where $H(L)$ is the Shannon information of length L sequences. Figure 1(a) is simply a scatter plot of the complexity-entropy pairs.

There are a number of important features exhibited by the complexity-entropy diagram. (Details are given in Refs. [11] and [13].) The first is that the extreme values of entropy lead to zero complexity. That is, the simplest periodic process at $H(L)/L = 0$ and the most random one at $H(L)/L = 1$ are statistically simple. They both have zero complexity since they are described by ϵ-machines with a single state. Between the extremes the processes are noticeably more complex with an apparent peak about a critical entropy value denoted H_c. Below this entropy, it turns out, all of the data streams come from parameters at which the logistic map is periodic — including parameters within the "periodic windows" found in the map's chaotic regime. The data sets with $H(L)/L > H_c$ are produced at chaotic parameter values.

A theory was developed to explain the emergence of high computational capability between the ordered and disordered regimes. For processes with $H(L)/L < H_c$ the entropy and complexity are equivalent

$$C = H \tag{4}$$

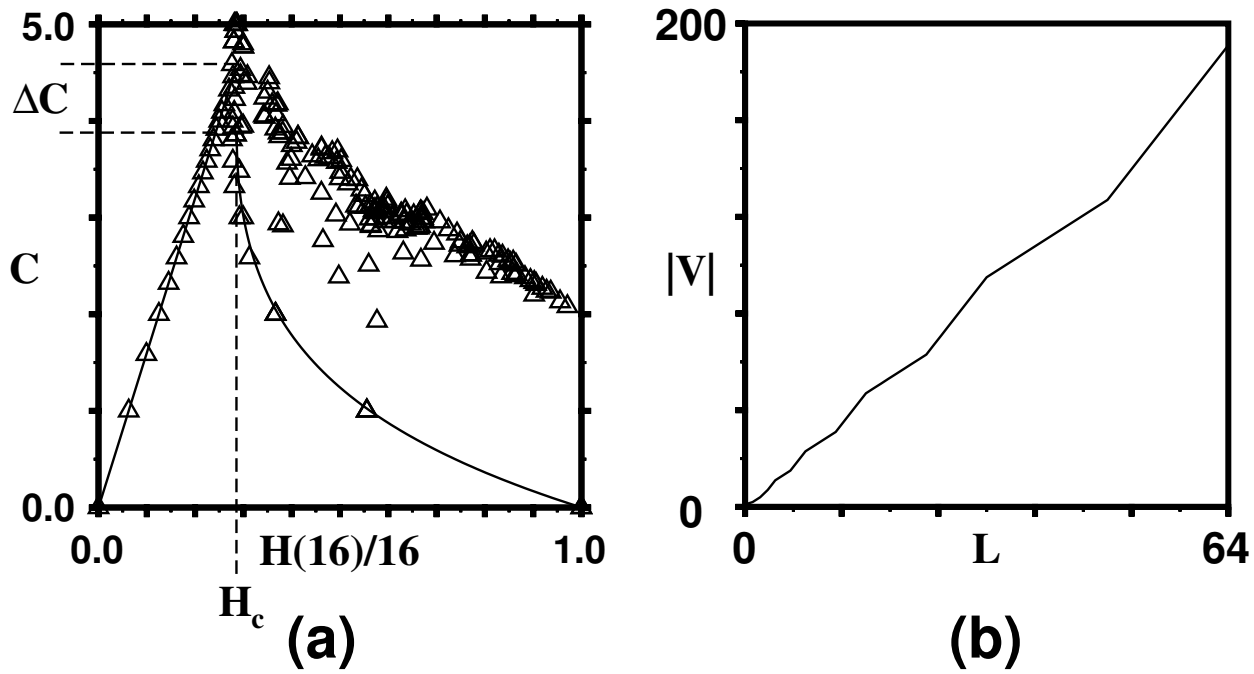

Figure 1 (a) Statistical complexity C versus specific entropy $H(L)/L$ for the period-doubling route to chaos. Triangles denote estimated $(C, H(L)/L)$ at 193 values of the logistic map nonlinearity parameter. ϵ-machines were reconstructed using a subsequence length of $L = 16$. The heavy solid lines overlaying some of this empirical data are the analytical curves derived for C versus $H(L)/L$. (After [13].) (b) At one of the critical parameter values of the period-doubling cascade in the Logistic map the number $|\mathbf{V}|$ of inferred states grows without bound. Here $r = r_c \approx 3.5699456718695445\ldots$ and the sequence length ranges up to $L = 64$ where $|\mathbf{V}| = 196$ states are found. It can be shown, and can be inferred from the figure, that the per symbol density of states $|\mathbf{V}(L)|/L$ does not have a limiting value as $L \to \infty$. (After [11].)

This is shown as a solid straight line on the left portion of Figure 1(a). For processes with $H(L)/L > H_c$ the dependence of complexity on entropy is more interesting. In fact, the solution is given in terms of the dependence of the entropy on the complexity. The result is that

$$H(L) = C + \log_2\left(2^{L2^{-C}} - 2^{-1}\right) \tag{5}$$

The curved solid line in Figure 1(a) shows the relevant portion of Eq. (5).

Comparing the periodic and chaotic analyses — i.e. Eqs. (4) and (5) — provides a detailed picture of the complexity-entropy phase transition. The critical entropy H_c at each sequence length L is given

$$H_c(L) = C'(L) + \log_2\left(by - 2^{-1}\right) \tag{6}$$

where $C'(L) = \log_2 L - \log_2\log_2 y$ is the complexity on the high entropy side at H_c, $y \approx 2.155535$ is the solution of $y\log_e y - y + 2^{-1} = 0$, and $1 \leq b \leq 3 \cdot 2^{-\frac{3}{2}} \approx 1.06066$ is a constant. From Eq. (4) it follows immediately that the complexity C'' on the low-entropy side of the transition is itself $H_c \cdot L$. The difference is a finite constant — the latent complexity of the transition $\Delta C = C'' - C' \approx 0.7272976$ bits. The latent complexity is independent of the sequence length.

This analysis of the interdependence of complexity and entropy is nonasymptotic in the sense that it applies at each sequence length L. If, as done for Figure 1(a), this length is fixed

at $L = 16$, the preceding results predict the transition's location. The critical entropy there, for example, is $H_c \approx 0.286205$. But for any L the overall behavior is universal. All behaviors with specific entropies $H(L)/L < H_c$ are periodic. All behaviors with higher entropies are chaotic. The functional forms in Eqs. (4) and (5) are general lower bounds. The statistical complexity is maximized at the border between the predictable and unpredictable "thermodynamic phases". It is important to emphasize that the complexity-entropy diagram makes no explicit reference to the system's nonlinearity parameter. The diagram is defined this way in order to show those properties which depend only on the intrinsic information production and intrinsic computational structure.

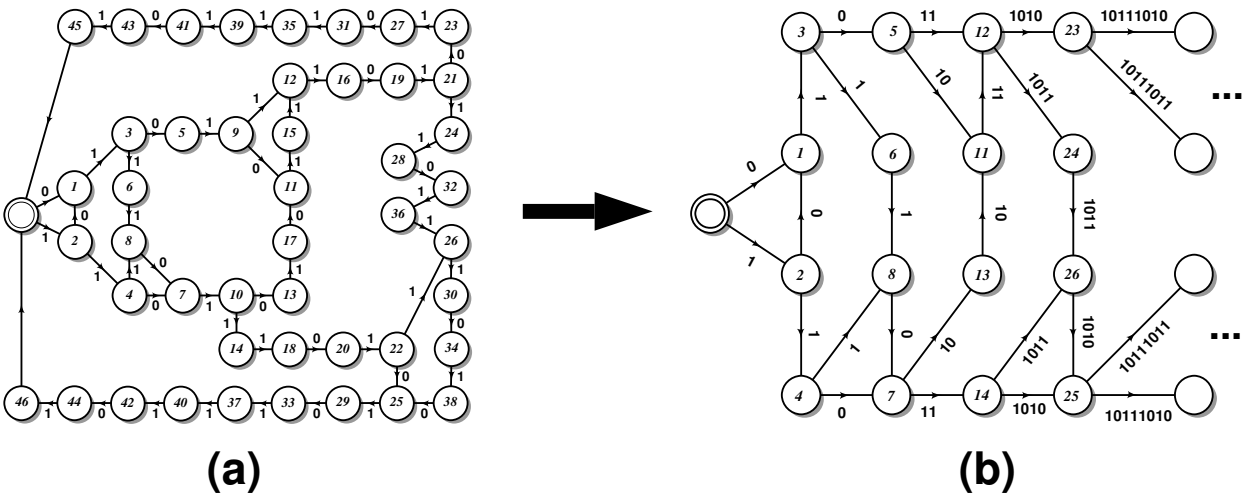

Figure 2 (a) Approximation of the critical ϵ-machine at the period-doubling onset of chaos. (After [13].) (b) The dedecorated version of the machine in (a). Here the deterministic state chains have been replaced by their equivalent strings. (After [11].)

Up to this point the overall interplay between complexity and entropy for the period-doubling cascade has been reviewed. But what happens at the phase transition; i.e. at the critical entropy density H_c? One parameter value, out of the many possible, corresponding to $H(L)/L = H_c$ is the first period-doubling onset of chaos at $r = r_c \approx 3.5699456718695445\ldots$. Figure 2(a) shows the 47 state ϵ-machine reconstructed with window size $L = 16$ at this parameter setting. An improved approximation can be attempted by increasing the window length L to take into account structure in longer subsequences. Figure 1(b) shows the result of doing just this: at the onset of period-doubling chaos the number $|\mathbf{V}|$ of states for the reconstructed ϵ-machines grows without bound.

The consequence is that the data stream produced at the onset of chaos leads to an infinite machine. This is consonant with the view introduced by Feigenbaum that this onset of chaos can be viewed as a phase transition at which the correlation length diverges.[15] The computational analog of the latter is that the process intrinsically has an infinite memory capacity. But there is more that the computational analysis yields. As will now be shown, for example, the infinite memory is organized in a particular way such that the logistic map is not equivalent to a universal Turing machine, but is instead a less powerful stack automaton.

The "explicit state" representation of Figure 2(a) does not directly indicate what type of information processing is occurring at the phase transition. Nor does the unbounded growth of

machine size shown in Figure 1(b) give much help. A simple transformation of the 47 state machine in 2(a) goes some distance in uncovering what is happening. Replacing the unbranched "chains" in the machine with the corresponding sequences produces the "dedecorated" critical machine of Figure 2(b). In this representation is it evident that the branching states are quite regularly organized. Beyond the discovery of this higher-order regularity, there is an additional element that consists of manipulating the intervening strings between the branching states.

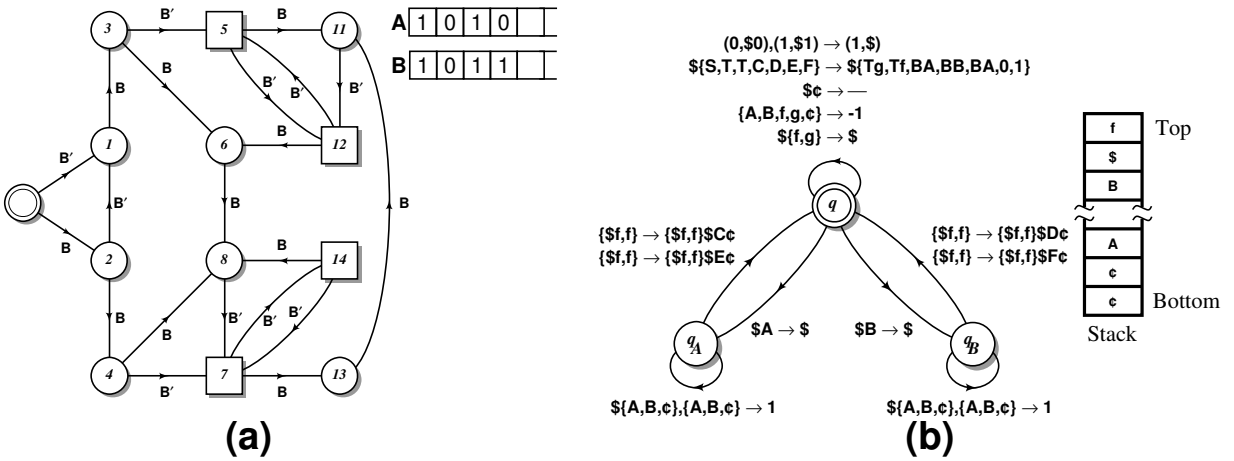

Figure 3 (a) The finite version of Figure 2(b)'s infinite critical ϵ-machine. This is a string production machine that, when making a transition from the square states, updates two string registers with the productions **A** → **BB** and **B** → **BA**. **B'** is the contents of **B** with the last bit flipped. (b) Another finite representation of the period-doubling critical ϵ-machine — a one way nondeterministic nested stack automaton — that produces symbols sequentially. (After [11].)

By following the increasing-accuracy modeling experiment shown in Figure 1(b) in detail, one can ask how the machines in a series of successively improved models grow in size. The result, as disclosed by the dedecorated machine, is that only the branching states and "string productions" are needed to describe the regularity in the growth of the machines. This, in turn, leads to the discovery, shown in Figure 3(a), of a finite machine with two kinds of states (the new type is denoted with squares) and two registers **A** and **B** that hold binary strings. Simple inspection of the dedecorated machine shows that the string manipulations can be described by appending a copy of **A**'s contents onto **B** and replacing the contents of **A** with two copies of **B**'s contents. These string productions are denoted **A** → **BB** and **B** → **BA**. At the outset, register **A** contains "0" and **B** contains "1".

One problem with the string production machine of Figure 3(a) is that the length of strings in the registers grows exponentially fast, which contrasts sharply with the sequential production of symbols by the logistic map. Figure 3(b) gives an alternative, but equivalent, serial machine that produces a single symbol at a time. It is called a one-way nondeterministic nested stack automaton. The memory in this machine is organized not as string registers, but as a pushdown stack. The latter is a type of memory whose only accessible element is on the top. In fact, the automaton shown has a slightly more sophisticated stack that allows the finite control to begin a new "nested" stack within the existing one. The only restriction is that the automaton cannot move on to higher levels in the outer stack(s) until it is finished with its most recently created stack.

The net effect of these constructions is that a finite representation has been discovered from an infinite one. One of the main benefits of this, aside from producing a manageable description, is that the type of information processing in the critical "state" of the logistic map has been made transparent.

4. INTRINSIC COMPUTATION AT THE ONSET OF QUASIPERIODIC CHAOS

The second route to chaos of interest, which also has received extensive study, is that through quasiperiodicity. In the simplest terms, this route to chaos and the models that exhibit it describe the coupling of two oscillators whose periods are incommensurate — the ratio is not rational. The ratio of the number of periods of one oscillator to the other in order to complete a full cycle for both is called the winding number $\hat{\omega}$. This is a key parameter that controls the entire system's behavior: when rational the two oscillators are phase-locked. Quasiperiodic behavior is common in nature and underlies such disparate phenomena as cardiac arrhythmia, the stability of the solar system, and the puzzling synchronization of two mechanical clocks located in close proximity.

The simplest model of two "competing" oscillators is the discrete-time circle map

$$\begin{aligned} \phi_{n+1} &= f(\phi_n) \mod 1 \\ \text{where} \quad f(\phi) &= \omega + \phi + \frac{k}{2\pi} \sin 2\pi\phi \end{aligned} \tag{7}$$

The map's name derives from the fact that the $\mod 1$ operation keeps the state ϕ_n on the unit circle. One thinks of ϕ_n then as a phase — or, more properly, the relative phase of the two original oscillators. There are two control parameters, ω and k. The former directly sets the phase advance and the latter the degree of nonlinearity, which can be roughly interpreted as the coupling strength between the two oscillators.

As a function of the nonlinearity parameter the behavior makes a transition to chaos. Like the logistic map, there is a signature to the path by which chaotic behavior is approached from periodic behavior. Furthermore, the circle map's signature has the basic character of a phase transition.[16]

The following will investigate one arc through (ω, k)-space that exhibits just such a phase transition to chaos. This is a path that includes the golden mean circle map — so-called since its winding number is the golden mean $\hat{\omega} = \frac{1+\sqrt{5}}{2}$. The easiest way to implement this is to set $\omega = \hat{\omega}$. Varying $k \in [0, 6]$ then gives a wide sample of behavior types on the quasiperiodic route to chaos. $k = 1$ is the threshold of nonlinear behavior, since the map for larger values becomes many-to-one; $k > 1$ is also a necessary, but not sufficient condition for chaos.

The measuring instrument uses three types of partition depending on the parameter range: $k = 0$, $k \in (0, 1]$, and $k > 1$. Generally, the instrument is a binary partition that labels $\phi_n \in (\phi', \phi'']$ with $s = 0$ and $\phi_n = (\phi'', \phi']$ with $s = 1$. For $k = 0$, $\phi' = \frac{1}{2}$ and $\phi'' = 0$; for $k \in (0, 1]$, $\phi' = \frac{1}{2}$ and $\phi'' = f^{-1}\left(\frac{1}{2}\right)$; and, for $k > 1$, ϕ' is the larger and ϕ'' the smaller value of $(2\pi)^{-1} \cos^{-1}\left(-k^{-1}\right)$ on the interval. By iterating the map many times on an initial condition a time series $\boldsymbol{\phi} = \phi_0\phi_1\phi_2\ldots$ is produced. When observed with an instrument the time series is converted to a binary string $\mathrm{s} = s_0s_1s_2\ldots$ of coarse measurements $s_i \in \{0, 1\}$.

Figure 4(a) shows the complexities and entropies estimated for the quasiperiodic route to chaos at several hundred settings along the chosen parameter arc. As with period-doubling,

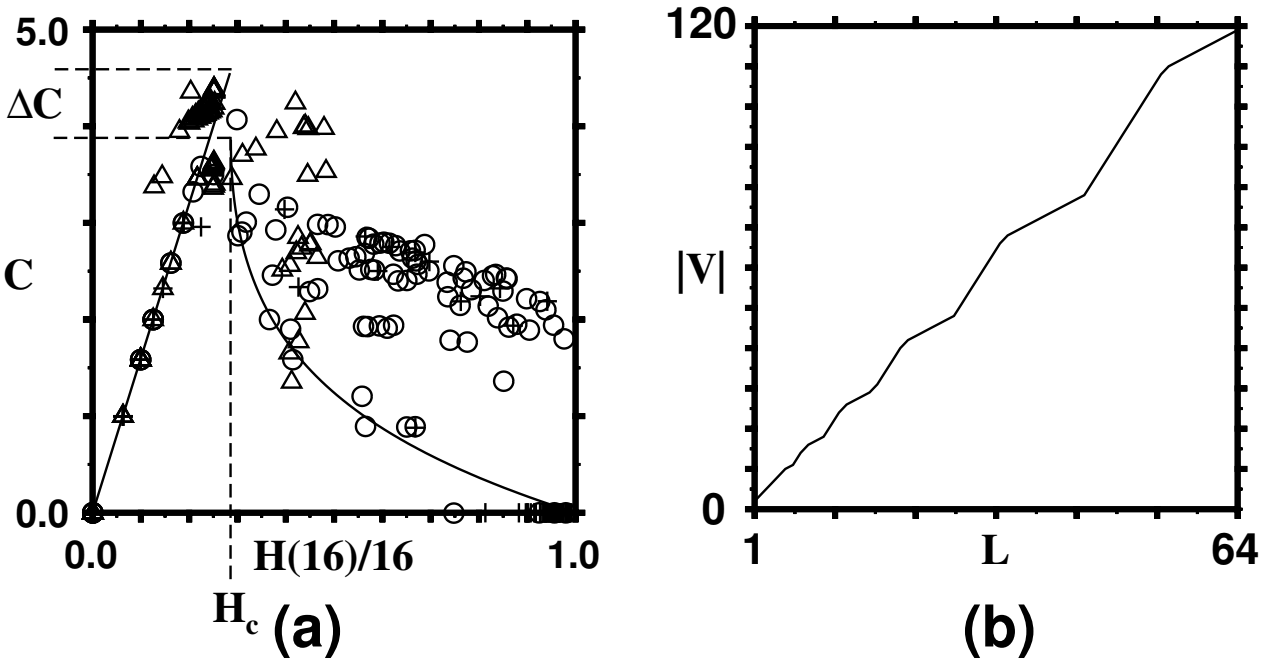

Figure 4 (a) Statistical complexity C versus specific entropy $H(L)/L$ for the quasiperiodic route to chaos. Triangles denote estimated $(C, H/L)$ at 303 values of the circle map with $\omega = \frac{1+\sqrt{5}}{2}$ and nonlinearity parameter k in three different ranges: 101 values for $k \in [0, 2]$, 101 values for $k \in [3.5, 3.9]$, and 101 values for $k \in [4.5, 6]$. These are ranges in which the behavior is more than simple periodic. ϵ-machine reconstruction used a tree depth of $D = 32$ and a morph depth of $L = 16$ for the first range and $(D, L) = (16, 8)$ for the second two ranges, which typically have higher entropy rates. The entropy density was estimated with a subsequence length of $L = 16$. Refer to Figure 1(a) for details of the annotations. (b) At the golden mean critical winding number in the quasiperiodic route to chaos the number $|\mathbf{V}|$ of inferred states grows without bound. Here the sequence length ranges up to $L = 64$ where $|\mathbf{V}| = 119$ states are found.

the quasiperiodic behavior with entropies $H(L)/L < H_c$ are periodic. All those with higher entropies are unpredictable. The statistical complexity is maximized at the border between the ordered and chaotic "thermodynamic phases". The lower bounds, Eqs. (4) and (5), are shown again as solid lines for both phases. The circle map clearly obeys them, as did the logistic map, though the scatter differs. For example, there is a cluster of points just below H_c at high complexity. These are all due to the "irrational" quasiperiodic behavior that is predictable. The complexity derives from the fact that the map essentially "reads out" the digits of their irrational winding number. This leads to data streams that require large machines to model. There is also some scatter at high entropy and low complexity. This is due to highly intermittent behavior that results in all subsequences being observed, but with an underlying probability distribution that is far from uniform. The result is that ϵ-machine reconstruction approximates the behavior as a biased coin — zero complexity, since it has a single state, and entropy less than unity.

What happens at the quasiperiodic onset at $k = 1$? The metric entropy is zero here, since the number of length L subwords increases strictly linearly: $N(L) = L + 1$. The single symbol entropy is high, $H(1) \approx 0.959419$ bits, since the frequency of isolated zeros is $\lim_{L\to\infty} \frac{F_{L-2}}{F_L} = \hat{\omega}^{-2} \approx 0.381966$, where F_L is the L^{th} Fibonacci number.

ϵ-machine reconstruction applied to this "critical" data stream does not lead to a finite state machine. In fact, just as for the logistic map at the onset of chaos, the machine size

keeps diverging. (See Figure 4(b).) A finite approximation to the presumably infinite "critical" machine is shown in Figure 5(a).

Notably, the intrinsic computation in quasiperiodicity can be finitely represented at a next higher level. When the average winding number is the golden mean, one finds the "Fibonacci" machine shown in Figure 5(b). There is a two state finite control automaton shown at the top portion of Figure 5(b) that determines copying operations on two registers, **A** and **B**, that contain binary strings. The finite control is started in the left-most, double-circled state, **A** begins with "1", and **B** with "0". The finite control machine's edges are labelled with the actions to be taken on each state-to-state transition. The first symbol on each edge label is a zero or one read from the input data stream that is to be recognized. The symbol read determines the edge taken when in a given state. The backward slash indicates that a string production is performed on registers **A** and **B**. This consists of copying the previous contents of **A** to **B** and appending the previous contents of **B** to **A**. The string productions are denoted $\mathbf{A} \rightarrow \mathbf{AB}$ and $\mathbf{B} \rightarrow \mathbf{A}$. They are applied simultaneously. If there are two backward slashes, then two "Fibonacci" productions are performed. The input string must match the contents of register **A**, when register **A** is read in reverse. The latter is denoted by the left-going arrow above **A** in the edge label.

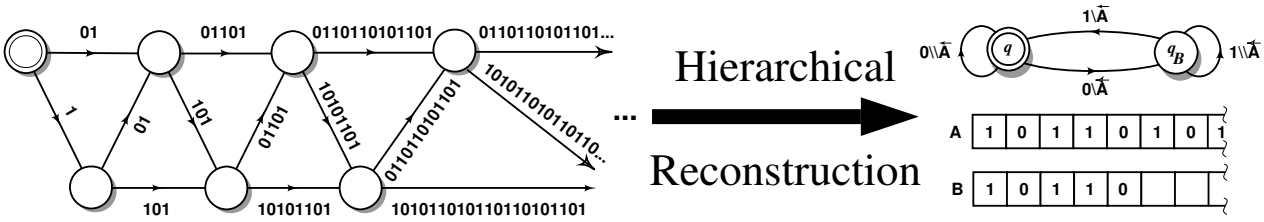

Figure 5 (a) A portion of the infinite critical machine for the quasiperiodic route to chaos at the golden mean winding number. Note that the dedecorated machine is shown — that is, the intervening states along deterministic chains have been suppressed. (b) The Fibonacci machine: the finite representation of the infinite machine in (a).

The basic computation step describing the quasiperiodic critical dynamics employs a pair of string productions. The computational class here is quite similar to that for period-doubling behavior — that is, nested stack automata. It is at this higher level that a finite description of the golden mean critical behavior is found. This is demonstrated, as for period-doubling, by noting that the productions are context-free deterministic Lindenmayer productions and that these can be mapped first to an indexed context-free grammar and then to nested stack automaton.[1] Thus, rather than Figure 5(b) the Fibonacci machine can be represented with a stack automaton analogous to that shown in Figure 3(b) for the period-doubling onset of chaos.

The required length of the Fibonacci machine registers grows as a function of the number of applications of the production at an exponential rate which is the golden mean, since the string length grows like the Fibonacci numbers — an observation directly following from the productions. Thus, with very few transitions in the machine input strings of substantial length can be recognized.

Another interpretation of the recognition performed by the Fibonacci machine in Figure 5(b) is that it phase locks to the quasiperiodic data stream. That is, the Fibonacci machine can jump in at any point in the "critical" string, not necessarily some special starting time, and, from that symbol on, determine if the subword it is reading is in the language of all Fibonacci subwords.

To close this section, let us summarize both its and the last section's approach to intrinsic computation. Both sections demonstrated how intrinsic discrete computation can be detected in a continuous-state dynamical system. They both also showed that to get a simple model that captures the system's *true* computational capability, as determined by observations, it is sometimes necessary to jump up to a more powerful computational class. At both onsets of chaos the computational analysis identified structures that were higher than finite memory devices. The onset of chaos led to infinite memory and, just as importantly, to memory that is organized in a particular way to facilitate some types of computation and to proscribe others. The logistic and circle maps at their respective onsets of chaos are far less than Turing machines, especially ones that are universal. At the onset the information processing embedded in them jumps from the finitary level to the level of stack automata. One practical consequence of failing to change to a more powerful representation for these critical systems is that an observer will conclude that they are more random, less predictable, and less complex, than they actually are. More generally, appreciating how infinite complexity can arise at the onset of chaos leads one to expect that highly nonlinear systems can perform significant amounts of and particular forms of information processing.

5. TOWARDS A THEORY OF HIERARCHICAL LEARNING

But how would such high complexity computation elements be detected? Figures 1(b) and 4(b) suggest that significant difficulties arise when one tries to model a critical process using a representation adapted to finite memory — infinite models appear. In both the critical period-doubling and critical quasiperiodic cases a resolution to this was found by dedecorating the machines and inferring the productions governing the successively longer strings.

One interpretation of this situation is that the initial representation selected was not appropriate. In each of the above cases it was not clear, until the higher level finite representation was found, what type of information processing the dynamical systems were implementing. Is there a way to automatically discover better model classes — i.e. to detect the emergence of high levels of complexity and of new types of computational capability? In [17] a modeling technique — hierarchical ϵ-machine reconstruction — was introduced that begins to answer this question. It turns out that the basic method of changing from one representation to another is much more general than the logistic and circle map examples would suggest. There are other process classes — e.g. hidden Markov processes and spatio-temporal dynamics — where the generalization has provided essential insights to discovering finite models of demonstrably complex data. Hierarchical machine reconstruction gives a way to break out of weak model classes, to learn more powerful ones.[14,17]

To be clear about this, the first step is to note the common aspects of any hierarchy of computational models. At each level there are a number of elements:

1. **Models** M, in some class $\mathcal{M}$, consisting of states and transitions. These are observed only indirectly via a measurement function.
2. **Languages** being the ensembles of finitely representable behaviors.

3. **Symmetries** reflecting the observer's assumptions about a process's structure. These determine the semantic content of the model class $\mathcal{M}$, which is defined by equivalence relations $\sim$ corresponding to each symmetry.
4. **Reconstruction** being the procedure for producing estimated models. It factors out a symmetry from a data stream s. Formally, reconstruction of a model $\mathrm{M} \in \mathcal{M}$ is denoted as $\mathrm{M} = \mathrm{s}/\sim$.
5. **Complexity** of a process being the size of a reconstructed model M with respect to the given class $\mathcal{M}$: $C(\mathrm{s}|\mathcal{M}) = \|\mathrm{M}\|$.
6. **Predictability** being estimated with reference to the states that are distinguished by M.

It is important that reconstructed models $\mathrm{M} \in \mathcal{M}$ be minimal. This is so that M contains no more structure than and no additional properties beyond the underlying dynamics. The simplest explication of this is to note that there are many multiple state representations of an ideal random binary string. But if the size of representation is to have any meaning, such as the amount of memory, only the single state process can be allowed as the model from which it is computed.

With this view of individual levels in a hierarchy of model classes, hierarchical machine reconstruction is the search for an ϵ-machine, which is the *minimal* model at the *least* computationally powerful level yielding a *finite* description. This definition builds in an adaptive notion that an observer initially might not have the correct model class. How does the observer find a better representation? By moving up an inductive hierarchy through the innovation of new notions of "state".

A large part of innovating a new model class is simply a reapplication of machine reconstruction as introduced in Ref. [13]. The central method of discovering structure is to group lower-level states into equivalence classes that lead to the same range of future behavior. These equivalence classes then become the notion of state at the new level. A series of increasingly accurate lower level models are, in this sense, a data stream — $\mathrm{M}(\epsilon), \mathrm{M}\left(\frac{\epsilon}{2}\right), \mathrm{M}\left(\frac{\epsilon}{4}\right), \mathrm{M}\left(\frac{\epsilon}{8}\right), \ldots$ — for reconstruction at the next higher level. The regularity, if any, between the models *is* the next higher level representation. For example, at the onset of chaos hierarchical machine reconstruction involves four levels — data stream, trees, finite automata, and stack automata — before finding a finite representation.

There is an additional step beyond grouping states at one level according to their transition structure. This step is seen in the deterministic modeling of recurrent hidden Markov models as the innovation of a resettable counter,[18] at the onsets of chaos reviewed here as the innovation of string productions,[11] and in discrete spatial processes as the innovation of local state machines to break away from cellular automata look-up tables.[19] In each case it was quite straightforward to find the additional computational element governing the lower level information processing. But since an exhaustive and (partially) ordered spectrum of basic computational elements is not yet available, innovation must contain a component, albeit small, of undetermined discovery.

6. WHAT IS A MECHANISM SUCH THAT IT COMPUTES?

With an eye toward the present technological age and the likely improvements in microscopic engineering, let us consider a engineering gedanken experiment. This will cast the notion

of intrinsic computation in a higher relief — hopefully without giving a false impression of feasibility. At the present time an integrated circuit designer in effect employs many millions, if not trillions or more, of degrees of freedom in a physical device in order to implement a basic computational element, such as the storage, transmission, or logical manipulation of a single bit of information. The vast amount of available microscopic information processing is forced onto the procrustean bed of Boolean logic by the relatively macroscopic methods of materials science.

Rather than appealing to the view of 19th century logic for a computational paradigm, perhaps a direct look at the nonlinear physics that governs molecular solid state interactions will yield new efficiencies and new elementary components for information processing. Is there anything to be gained by using the intrinsic computational capability of the microscopic physics? As shown above there certainly can be high levels of computational structure in simple nonlinear dynamical systems.

Holding the available physical resources constant, there are two information processing approaches that can be compared. The first is the conventional digital computer as designed at the present time. The second is the possibility of architecting molecular structures in as much detail as is consistent with solid state physics. In each, given a mole of silicon molecules and a satchel of dopants, how high a density of computation can be produced? And what, if anything, can dynamical systems contribute to our understanding of the limitations and possibilities?

The preceding discussion has contrasted two perspectives — the state-space geometry view of dynamical systems theory and the computation theoretic view of a process's symbolic dynamics in terms of automata. Figure 6 attempts to integrate these views and to provide a base for comparing intrinsic and usable computation in a physical device.

There are two submanifolds — a grouping of the many stable and unstable manifolds that foliate the state space. The vertical submanifold of states is that of an entropy cascade. Within it organized, low-entropy energy enters at the top and cascades down toward the heat bath of thermal fluctuations, in strict obeyance of the second law of thermodynamics. The horizontal submanifold of states, vastly smaller in dimension, is the "critical" submanifold that supports computation. There are several ways in which it does this. First, it must allow for arbitrarily long space and time correlations — i.e. the most basic requirement of reliable computation. Second, evolution within it must have nonlinear elements in order to do nontrivial information processing: storage, transmission, and logical operations. Third, it must "borrow" energy from and "return" energy to the entropy cascade in such a way that the computation is not corrupted. In effect organized energy must be introduced into the critical submanifold to drive the computation forward, but without destroying the infinite spatio-temporal correlations. Finally, the coupling between the submanifolds must be nonlinear in order to effect this transfer. To the extent that an essentially unidirectional coupling of organized energy is achieved then the information processing in the critical submanifold will be robust.

Note that the total thermodynamic entropy density can be quite high — as it is in present-day digital and analog computers — even in the presence of substantial intrinsic and usable computation. Thus, to support computation there is no requirement that the whole device have zero thermodynamic entropy density or vanishingly small dynamical entropy rate or that the system be near the onset of chaos. All that is required are the properties just delineated for the critical submanifold.

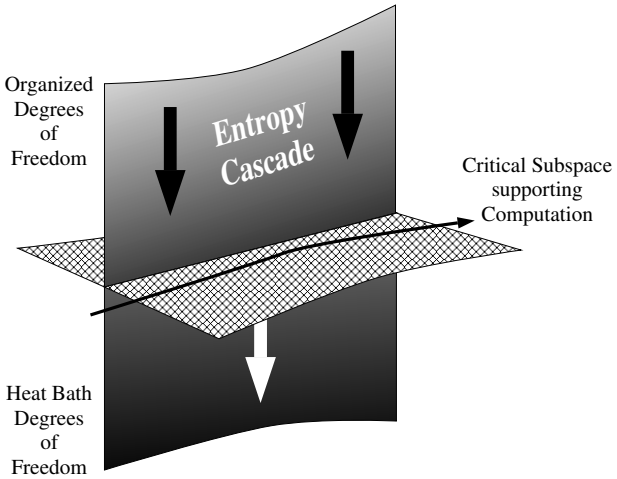

Figure 6 The manifolds of computation: A highly schematic view of the state space of a physical computing device. The state space is split into an entropy-cascade submanifolds that brings in energy at low thermodynamic entropy. Typically only a small fraction of that is garnished to drive the computation forward in the critical submanifold.

The preceding sections suggest two possible avenues to increasing the density of computation. The first would be to use the higher level computational processing available in nonlinear dynamical systems. The second would be to improve the energy transfer to and isolation of the critical submanifold with highly nonlinear coupling. Linear coupling, in contrast, is less effective in meeting the goals of unidirectional energy transfer and isolation. Indeed, in lieu of nonlinearity, present designs employ a large number of degrees of freedom, relying on the law of large numbers to reduce the net fluctuation in the critical submanifold. In any case, both possibilities of increasing computation density necessarily take advantage of nonlinear microscopic physics.

7. CONCLUDING REMARKS

There is an amusing difficulty in discussing "computation" at the present time. The term evokes a strong sense of discreteness and of mathematical logic. One is given the impression that many think "computation" only refers to discrete symbol manipulation. Such a perception could not be further from our knowledge of physics, human neurophysiology, and the history of computing machinery. Digital electronic technology has hijacked our scientific language. Credit for both the introduction and the resolution of this bias perhaps lay in Turing's use of the phrase "computer" to refer to a human performing calculations with pencil and paper. While he argued that the markings made on paper are discrete symbols from a finite set, the process leading to them being put down in the first place is initiated by and supported on an apparently continuous process — quantum mechanics not withstanding. A thoroughgoing discreteness does not seem justified. The relationship between the written symbols and the originating thought process strikes one as more akin to how the symbolic dynamics stood in relation to the instrumentation-obscured continuous-state logistic and circle maps considered above. That discrete computation then appeared embedded in those dynamical systems is itself more like our apprehension of meaning in written language.

It seems clear from the questions raised here that a synthesis of computation and dynamics could be fruitful. In particular, the theories of information transformation need to be rectified

with the theories of information creation. A synthesis would go some distance to harnessing the dynamical and nonlinear aspects of microscopic nature to perform a much wider range of information processing. Though this might reveal new regimes of information processing and help us push computation closer to the limits of physical devices, we should not forget that an equally important limitation is our own notion of what computation is.

ACKNOWLEDGMENTS

Thanks are due to Jim Hanson and Melanie Mitchell for helpful discussions. This work was supported by AFOSR 91-0293.

REFERENCES

1. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, 1979.
2. L. Blum, M. Shub, and S. Smale. On a theory of computation over the real numbers. *Bull. AMS*, 21:1, 1989.
3. C. Moore. Real-valued, continuous-time computers: A model of analog computation, part I. Technical Report 93-04-018, Santa Fe Institute, 1993.
4. A. R. Smith. Simple computation-universal cellular spaces. *J. ACM*, 18:339, 1971.
5. D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proc. Roy. Soc. Ser. A*, 439:553, 1992.
6. J. P. Crutchfield. Spatio-temporal complexity in nonlinear image processing. *IEEE Trans. Circ. Sys.*, 37:770, 1988.
7. V. I. Krinsky, V. N. Biktashev, and I. R. Efimov. Autowave principles for parallel image processing. *Physica*, 49D:247, 1991.
8. M. Mitchell, J. P. Crutchfield, and P. Hraber. Dynamics, computation, and the "edge of chaos": A re-examination. In G. Cowan, D. Pines, and D. Melzner, editors, *Integrative Themes*, volume XIX of *Santa Fe Institute Studies in the Sciences of Complexity*, page in press, Reading, MA, 1993. Addison-Wesley. Santa Fe Institute Technical Report 93-06-040.
9. E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for your Mathematical Plays*, volume 2. Academic Press, New York, 1984.
10. K. Lindgren and M. G. Nordahl. Universal computation in a simple one-dimensional cellular automaton. *Complex Systems*, 4:299–318, 1990.
11. J. P. Crutchfield and K. Young. Computation at the onset of chaos. In W. Zurek, editor, *Entropy, Complexity, and the Physics of Information*, volume VIII of *SFI Studies in the Sciences of Complexity*, page 223, Reading, Massachusetts, 1990. Addison-Wesley.
12. C. Moore. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Lett.*, 64:2354, 1990.
13. J. P. Crutchfield and K. Young. Inferring statistical complexity. *Phys. Rev. Let.*, 63:105, 1989.
14. J. P. Crutchfield and J. E. Hanson. Turbulent pattern bases for cellular automata. *Physica D*, in press, December 1993. Santa Fe Institute Report SFI-93-03-010.
15. M. J. Feigenbaum. Universal behavior in nonlinear systems. *Physica*, 7D:16, 1983.

16. L. P. Kadanoff, M. J. Feigenbaum, and S. J. Shenker. Quasiperiodicity in dissipative systems: a renormalization group analysis. *Physica*, 5D:370, 1982.
17. J. P. Crutchfield. Reconstructing language hierarchies. In H. A. Atmanspracher and H. Scheingraber, editors, *Information Dynamics*, page 45, New York, 1991. Plenum.
18. J. P. Crutchfield. Observing complexity and the complexity of observation. In H. Atmanspacher, editor, *Inside versus Outside*, pages 235 – 272, Berlin, February 1994. Springer-Verlag. Santa Fe Institute Technical Report 93-06-035.
19. J. P. Crutchfield. Unreconstructible at any radius. *Phys. Lett. A*, 171:52 – 60, 1992.