Santa Fe Institue Technical Report SFI 93–03–010 To appear in Physica D

# Turbulent Pattern Bases for Cellular Automata

**James P. Crutchfield**
**James E. Hanson**

**Department of Physics**
**University of California**
**Berkeley CA 94720 USA**[*]

## *Abstract*

Unpredictable patterns generated by cellular automata (CA) can be decomposed with respect to a turbulent, positive entropy rate pattern basis. The resulting filtered patterns uncover significant structural organization in a CA's dynamics and information processing capabilities. We illustrate the decomposition technique by analyzing a binary, range-2 cellular automaton having two invariant chaotic domains of different complexities and entropies. Once identified, the domains are seen to organize the CA's state space and to dominate its evolution. Starting from the domains' structures, we show how to construct a finite-state transducer that performs nonlinear spatial filtering such that the resulting space-time patterns reveal the domains and the intervening walls and dislocations. To show the statistical consequences of domain detection, we compare the entropy and complexity densities of each domain with the globally averaged quantities. A more graphical comparison uses difference patterns and difference plumes which trace the space-time influence of a single-site perturbation. We also investigate the diversity of walls and particles emanating from the interface between two adjacent domains.

**Keywords:**
cellular automata, coherent structure, complexity, pattern recognition, turbulence

[*]Internet addresses: JPC: chaos@gojira.berkeley.edu; JEH: hanson@gojira.berkeley.edu

# Contents

# List of Figures

## List of Tables

## 1 Pattern Bases, Coherent Structures, and Turbulence

One of the fundamental properties of any spatially-extended dynamical system is the variety of spatial configurations or patterns that arise during its evolution. Accordingly, the analysis of such systems usually starts with the choice of a representation that fits these patterns—in other words, a *pattern basis*. Once chosen, the pattern basis forms the underlying structure in terms of which the system is understood. Expansion in Fourier modes, the common choice of pattern basis for continuum systems, implicitly assumes that harmonic waves are appropriate for representing the system's spatial configurations. In many instances, this assumption is justified; for example, in Rayleigh-Bénard convection when the stationary state (heat conduction) destabilizes, the fluid motion organizes itself into spatially periodic roll patterns of a single (spatial) frequency. Representing the system in terms of its Fourier modes, therefore, projects the infinite-dimensional state space down onto a space of two dimensions: the amplitude and phase of the rolls. The small nonlinearities in the evolution then show up as modulations of the amplitude, which are governed by an "envelope" equation. Points at which the periodic pattern breaks down are "dislocations", and can be treated as quasiparticles with their own equations of motion.[1,2]

If the system is turbulent, however, the periodic representation is less appropriate. The spatial Fourier spectrum typically exhibits a continuum of excited modes. The diversity of observed configurations is so great that one is forced into the position of disregarding the details of the evolution in order to adopt some sort of stochastic model, such as in the case of fully-developed "hard" turbulence.[3] The problem with these models is that they virtually ignore the possibility of *any* regularity existing in the system, effectively masking it with purely stochastic dynamics. It is by now well-appreciated that for chaotic systems, this modeling procedure throws away important structural information; in particular, it fails to capture the possibility that coherent structures may be present in the system. What is needed instead is a *turbulent pattern basis*: a way of representing spatial configurations that captures the structural properties of the dynamics—e.g. via an appropriate notion of pattern—but that still allows for sufficient stochasticity that it is not burdened with describing the unpredictable details of high-entropy-rate "turbulent" behavior. In this way, coherent structures, if they exist, would show up as points at which the *turbulent* pattern breaks down. The system's evolution can then be decomposed into a turbulent background and coherent structures evolving against it. Recent work on two-dimensional turbulence[4,5] has shown how effective this type of decomposition can be.

The same situation appears in a much simpler class of spatial system, cellular automata (CA). Many CA evolve to patterns in which there is an easily-recognized spatio-temporally periodic background on which "particles" or "dislocations" move. In the same way as for the convection example above, the "particles" are defined *in terms of* the background pattern: they are points at which the pattern breaks down. Two relatively well known examples of this, for which the particles have been studied, are elementary CA rules 54 and 110.[6,7] But in chaotic CA, no periodic background pattern exists. If there is any regularity in the spatial configurations, it is hidden in the "turbulent" behavior of the CA. Again, what is needed is a turbulent pattern basis for the system, in terms of which its behavior is to be analyzed.

In Ref. [8] we introduced the notion of a *regular domain* as an organizing principle for the analysis of discrete spatially-extended dynamical systems, in particular for one-dimensional

CA, though it is also appropriate to other systems such as map lattices.[9] A regular domain is a regular language—i.e. a collection of symbol strings[10]—with the properties, defined more precisely below, of temporal invariance and spatial homogeneity. Regular domains can be spatially or temporally chaotic, or both, yet they can still capture the structural regularities that may exist. They give a precise notion of "spatial pattern" and the complexity of their patterns is determined by the size of the minimal stochastic automaton that describes all allowed domain configurations. It is these domains that serve as the turbulent pattern bases for cellular automata. For more information on regular domains in cellular automata, the reader is referred to Refs. [8] and [11].

In this article, we examine a CA having two distinct domains, each of which is chaotic, and in doing so we present a variety of generally applicable analytical and numerical techniques. Having reconstructively identified the domains from the space-time plot of the CA's evolution, we construct a filter for decomposing arbitrary spatial configurations into domains and intervening walls or coherent structures. We then analyze the structural and statistical properties of the domains and contrast them with those of the global state space. Finally, we briefly investigate the behavior of the coherent structures by tracing the evolution of inter-domain interfaces. Taken together, the techniques we develop represent a fundamental set of tools for analyzing the qualitative pattern dynamics of the system.

The domains have different spatial signatures, different entropy densities, and different complexities, yet they coexist on a single lattice. In analogy with statistical mechanics, we regard these domains as two distinct "thermodynamic phases" of the spatial dynamical system. Among CA, the phenomenon of multiple distinct domains appears to be a very common one. The methods we develop, and the conclusions we draw concerning the importance of domain detection for this one example, apply generally to a wide class of CA.

The CA is a binary-alphabet, range-2 CA with rule number 2614700074 in the conventional numbering scheme.[12] The rule was selected because it has two domains, both high-entropy "turbulent" processes, but with different internal structure. (See Appendix A.) Nevertheless, our formalism enables us to identify, separate and analyze them. This reveals more structural information there than would otherwise be apparent, especially when compared to what is afforded by the global view.

## 2 Raw and Filtered Space-time Data

To fix notation, we briefly recall the definition of a cellular automaton. The state of the CA at time $t$, denoted $\mathrm{s}_t$, consists of a one-dimensional array or lattice of cells $\sigma_t^i$ with values chosen from a finite alphabet $\mathcal{A} = \{0, 1, ..., k-1\}$. The local site-update function operating on neighborhoods of radius $r$ is written $\sigma_{t+1}^i = \phi(\sigma_t^{i-r}, ..., \sigma_t^{i+r})$. On a lattice of $N$ cells, $\mathrm{s}_t \in \mathcal{A}^N$. The global update operator $\Phi : \mathcal{A}^N \to \mathcal{A}^N$ applies $\phi$ in parallel to all neighborhoods in the lattice: $\mathrm{s}_{t+1} = \Phi(\mathrm{s}_t)$. For finite $N$, it is also necessary to specify the behavior at the edges of the array; in the following, we will always use periodic boundary conditions.

A space-time plot of the CA we are concerned with, binary range–two ($k = 2$, $r = 2$) rule 2614700074, is shown in Figure 1. The evolution is down the page, starting with a random

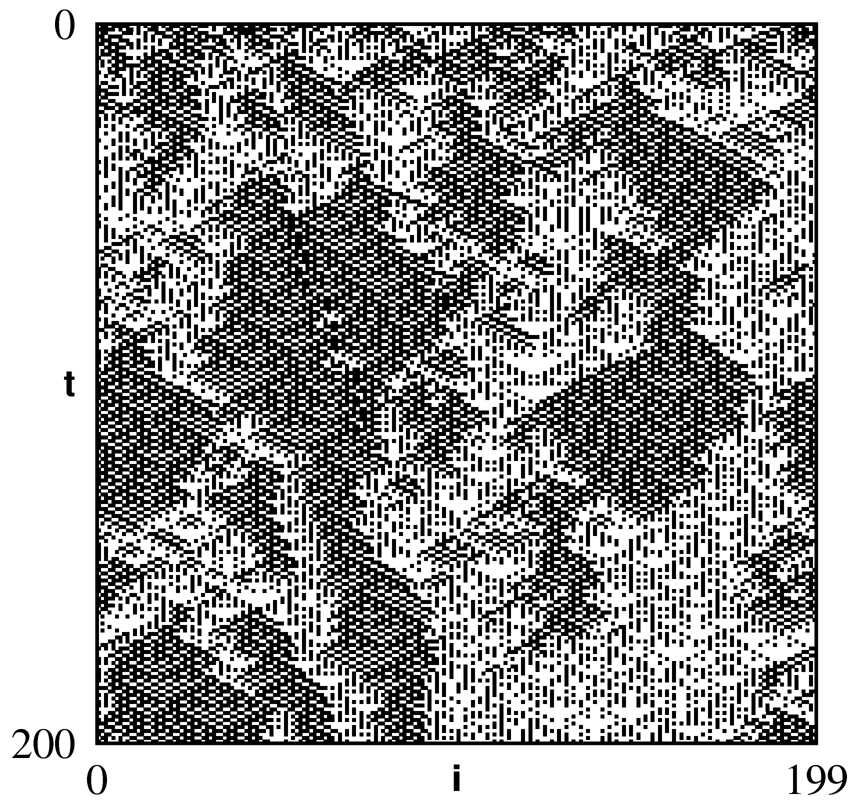


Figure 1 Space-time plot of the evolution of binary range-2 CA 2614700074, starting from a random initial condition on a 200 site lattice, over 200 time steps. The spatial configuration is displayed from left to right, with nonzero cells represented as black squares. Time is increasing down the page.

initial condition on a lattice of $N = 200$ cells with periodic boundary conditions. Cells with $\sigma_t^i = 0$ are white; $\sigma_t^i = 1$ are shown in black.

This sequence of spatial configurations was then fed as input into the machine reconstruction algorithm[13,14]. Machine reconstruction inductively constructs explicit models of the patterns that are present in the data. It acts as a substitute for visually identifying the patterns, and hence provides a method of discovering patterns that are too complex to be discovered by eye.

In this case, it is quite easy to visually identify space-time regions of two different types: one type of region is mostly white, in which there is a greater percentage of cells with $\sigma_t^i = 0$ than with $\sigma_t^i = 1$, while the other is predominantly black. Applying machine reconstruction to this data then serves to corroborate the visual evidence and to construct exact representations of the actual patterns present—and, incidentally, to illustrate the effectiveness of the reconstruction technique. But it is important to stress that machine reconstruction is an *alternative* to visual pattern-recognition, designed to operate on data for which visual identification is not effective.

Taking spatial sequences from each of the two types of space-time region in turn and applying machine reconstruction to them, we found that the first type of region consists of a tiling of the spatial pattern $(0\Sigma)$, and the second consists of a tiling of $(110\Sigma)$. Here, $\Sigma = \{0, 1\}$ denotes a "wild-card" cell that may be either $0$ or $1$. It is the appearance of the wild-card that indicates these patterns are unpredictable and chaotic. Using the notation $(s)^*$ to mean an arbitrary number of concatenations of the string $s$ and $\mathrm{sub}\mathcal{L}$ to indicate the set of all subwords of a language $\mathcal{L}$, we identify the two regions with the regular languages $\Lambda^0 \equiv \mathrm{sub}(0\Sigma)^*$ and $\Lambda^1 \equiv \mathrm{sub}(110\Sigma)^*$.

As we show in the next section, $\Lambda^0$ and $\Lambda^1$ are in fact regular domains. Each domain language represents the set of all spatial configurations with the given pattern. Depending on context, we will use the word "domain" to mean the domain language itself, or any single configuration in that language.

The process graphs for the domain languages $\Lambda^0$ and $\Lambda^1$ are shown in Figure 2. The circles represent the nodes or states of the graph. A symbol in $\mathcal{A}$ is assigned to each edge connecting two states; an edge labelled with the symbol $\Sigma$ is shorthand for a pair of parallel edges with symbols $0$ and $1$. States with inscribed circles or squares denote legal initial or final states, respectively. The graphs are interpreted as follows: a word in a language $\mathcal{L}$ is generated from its process graph $\mathrm{M}(\mathcal{L})$ by starting in any state of $\mathrm{M}(\mathcal{L})$ and taking any path along the graph's edges, reading off the edge labels as they are traversed. Each path on the graph corresponds to a single word in $\mathcal{L}$, although there can be several distinct paths generating the same word. The indices inscribed in the states of $\mathrm{M}(\Lambda^j)$ are the values of the domain's *phase* $\phi^j$. A symbol $\sigma^i$ in a word $\omega = ...\sigma^{i-1}\sigma^i\sigma^{i+1}... \in \Lambda^j$ has phase $n$ if $\mathrm{M}(\Lambda^j)$ is in the state with index $n$ after reading $\sigma^i$; this is written

$$\phi^j(i) = n \tag{1}$$

By convention, we have chosen the $\phi^j = 0$ state of each domain $\Lambda^j$ to be the state after the edge labelled with the wild-card $\Sigma$.

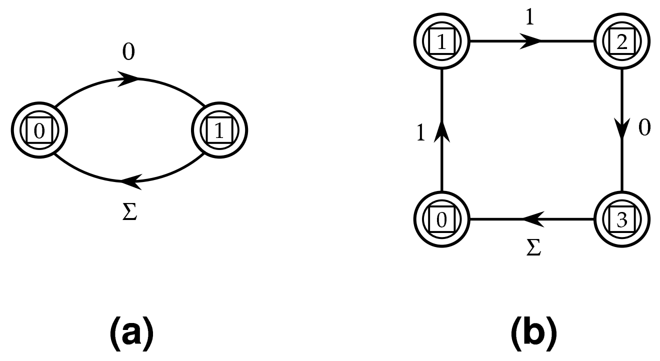


Figure 2 Process graphs for the two domains: (a) $\Lambda^0$; (b) $\Lambda^1$. Nodes or states of the graph are shown as circles. States with inscribed circles and squares denote start and final states, respectively; by definition, all states in a process graph are both start and final states. The number inscribed in each state gives the phase of the underlying pattern when the machine is in that state. An edge labelled with the symbol $\Sigma$ is shorthand for a pair of edges with labels $\{0, 1\}$.

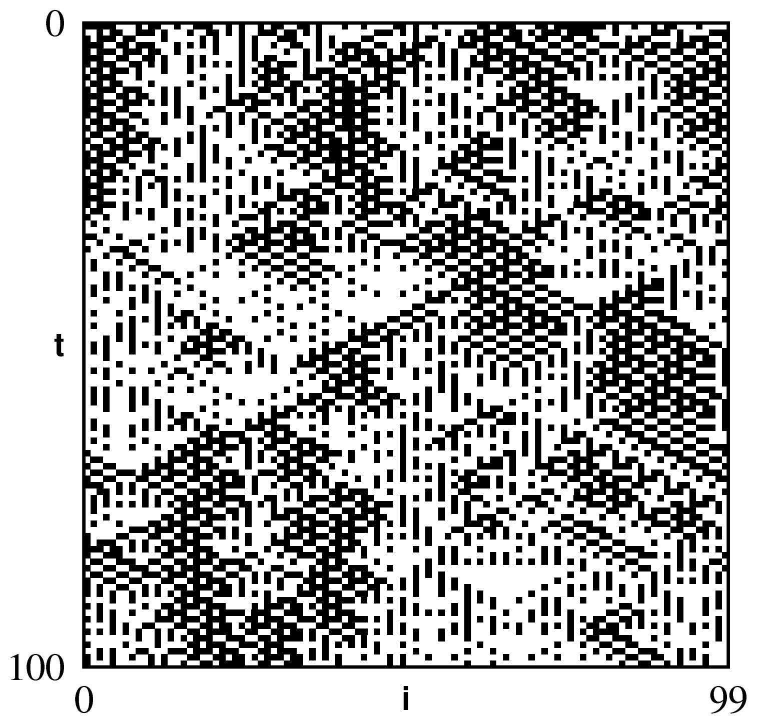
Once the domains have been identified, the next step is to find the domain walls, which are the boundaries between adjacent domains (see appendix B for the exact definition. From the two domains, we constructed a spatial filter $\mathrm{T}_\Lambda$ that maps arbitrary spatial configurations $\mathrm{s} \in \mathcal{A}^*$ onto the sequence of domains and walls the string contains. Each cell $\sigma^i$ of s is classified according to whether it is in a given domain or is a wall: cells in subsequences $\omega \in \Lambda^0$ are mapped to the symbol $0$, cells in subsequences $\omega \in \Lambda^1$ to the symbol $1$, and each wall between domains is mapped to another symbol whose value depends on the domains to either side. At each time-step $t$, $\mathrm{T}_\Lambda$ is run over the spatial configuration $\mathrm{s}_t$ from left to right, and the output string $\mathrm{T}_\Lambda(\mathrm{s}_t)$ is plotted. Using this filter on the space-time data shown in Figure 3(a), we generated the plot shown in Figure 3(b). In the latter, cells in $\Lambda^0$ are represented as white squares, cells in $\Lambda^1$ are gray squares, and all walls are black. Comparison of Figures 3(a) and 3(b) shows that the filtered pattern highlights the domains and walls present in the spatial configurations. The walls between different domains show up clearly, and are more precisely pinpointed than by eye in the original space-time data. Dislocations, which are walls separating domains of the same type, are made evident.

The filtered pattern represents a significant simplification of the apparent dynamics of CA rule 2614700074. The domains are temporally invariant, reducing the essential part of the temporal evolution to tracing the trajectories of the various walls. Additionally, each pattern is seen to be the concatenation of subpatterns from the two domains.

We traced the evolution of the initial condition shown in Figure 3 beyond time $t = 100$. Both types of domains were present up until $t = 700$, at which time the state as a whole fell into domain $\Lambda^1$ and remained there. In a survey of several dozen other initial conditions on a variety of lattice sizes, all of the states eventually fell into one or the other domain, with approximately $80\%$ falling into $\Lambda^0$. This is a preliminary indication that $\Lambda^0$ and $\Lambda^1$ may be regular attractors (defined in Ref. [8]), with $\Lambda^0$ having the larger basin measure. In any case, it is abundantly clear that the $\Lambda^0$ and $\Lambda^1$ are the fundamental structures organizing the high-dimensional state space.

The filter $\mathrm{T}_\Lambda$ we constructed is shown in Figure 4. It is a finite-state transducer that takes as input any sequence $\mathrm{s} \in \mathcal{A}^*$ and parses it into the sequence of domain and wall labels. In this way, it recognizes the structures implicit in the spatial configuration $\mathrm{s}$. There are three different types of state shown in Figure 4. States without inscribed labels are synchronization states, devoted to processing the first few symbols in $\mathrm{s}$. The machine starts in the state with the double circle at the top of the Figure, then moves along edges according to the input symbols. The first few symbols read serve to synchronize the machine to the underlying pattern, driving it from the start state of total ignorance into one of the asymptotic states. The edges are labelled $\sigma_{in}|\sigma_{out}$ for input and output symbols, respectively; however, during synchronization the machine produces no output, which is signified by putting $\sigma_{out} = \lambda$. Besides the synchronization states, there are two types of asymptotic state, one for each domain. States labelled $\Lambda^0$ or $\Lambda^1$ correspond to states visited when the input cell is in domain $\Lambda^0$ or $\Lambda^1$, respectively. Having synchronized to the pattern data stream, the transducer is in the asymptotic part of the graph, where it remains for the rest of its operation.

During synchronization, the transducer has yet to read in a sufficient amount of information to unambiguously distinguish the two domains and the several wall types. Once the machine has synchronized, every cell read is either in a domain or is a wall. Edges corresponding to cells in either domain are printed in bold lines, and have output symbols identifying the domain. Walls are printed in lighter lines, and have output symbols 2, 3, 4, or 5. These symbols denote the four types of wall, corresponding to the four possible combinations of domains on either side of the wall. Denoting a wall between domain $\Lambda^i$ and $\Lambda^j$ by $\Gamma^{ij}$, the four possible walls between domains $\Lambda^0$ and $\Lambda^1$ are $\Gamma^{00}$, $\Gamma^{01}$, $\Gamma^{10}$, and $\Gamma^{11}$. The complete domain transducer shown in Figure 4 assigns a unique symbol to each type of wall; in Figure 3(b), however, we have plotted all four types of walls as black squares. In making Figure 3(b), the transducer traversed the lattice from left to right, using the periodic boundary conditions to wrap around at the end in order



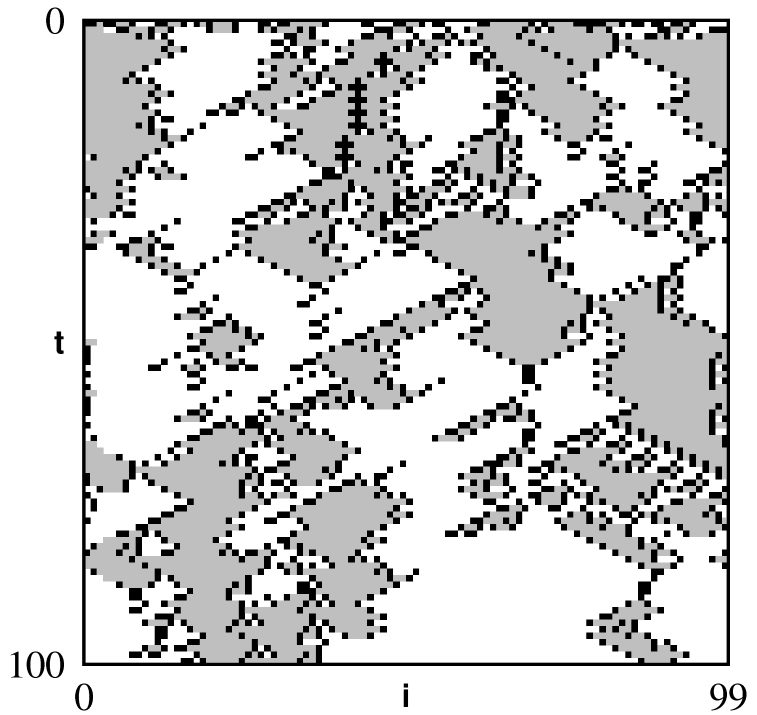


Figure 3 (a) Space-time plot of the evolution of binary range-2 CA 2614700074, starting from a random initial condition on a 100 site lattice over 100 time steps. (b) Filtered space-time plot of the same data shown in (a). Space-time regions in domain $\Lambda^0$ are shown in white, regions in $\Lambda^1$ are gray, and all walls are shown in black. All four varieties of wall are present. This figure was generated using the transducer of Figure 4.

to classify the first few cells, i.e., the cells used for synchronization. Except for the $0^*$ pattern, each cell in the lattice can be classified in this way with at most two passes over the data.

This machine $\mathrm{T}_\Lambda$ was built using a general construction procedure whereby, from any set of domain languages $\{\Lambda^0, \Lambda^1, ...\Lambda^{m-1}\}$, a minimal transducer is constructed that maps each cell in domain $\Lambda^i$ to symbol $i$, and maps each cell on wall $\Gamma^{ij}$ to symbol $(m + mi + j)$. (See Appendix B for more details on the construction.) Thus for a set of $m$ domains, there are $m(m+2)$ symbols in the transducer's output alphabet.

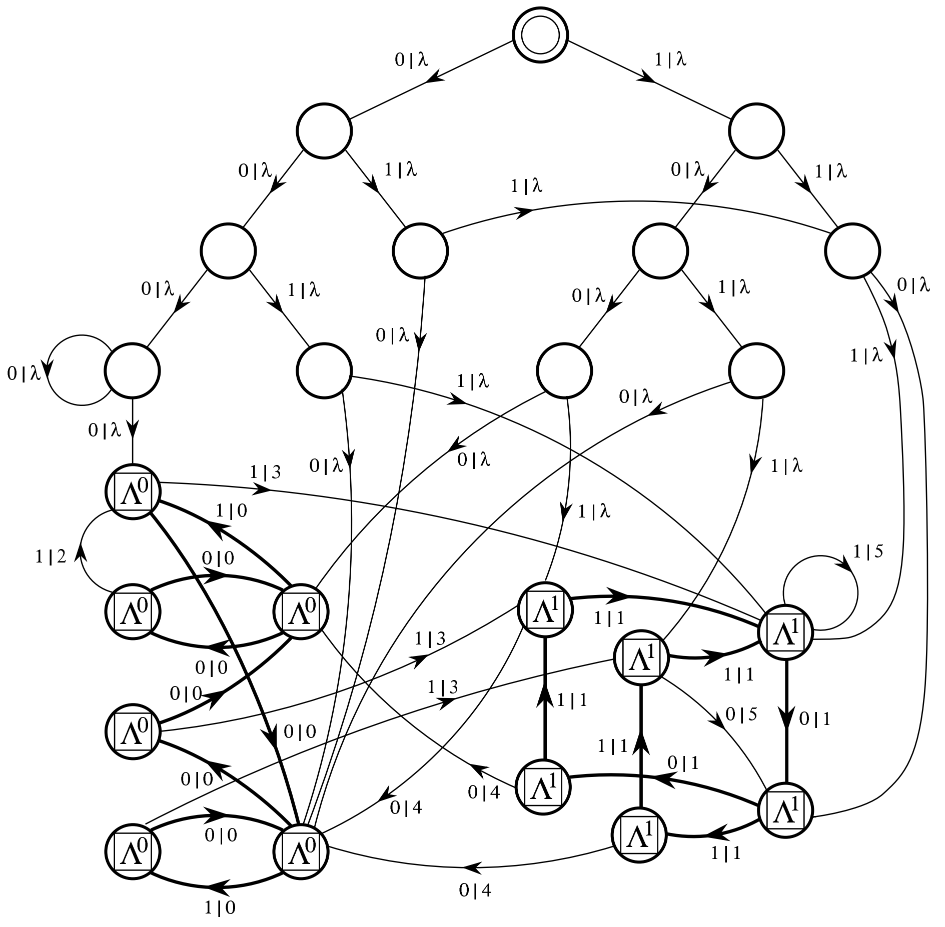


Figure 4 Domain transducer. Cells in domain $\Lambda^0$ are mapped onto 0's, cells in domain $\Lambda^1$ are mapped onto 1's, and walls $\Gamma^{ij}$ are mapped onto symbol $(2i + j + 2)$. States in the transducer corresponding to cells in a domain are labelled with that domain. Edges are labeled $\sigma_{in}|\sigma_{out}$ for input and output symbols, respectively. For the first few symbols read, the transducer is synchronizing to the data stream, and no output is generated; this is indicated by the null symbol $\lambda$. The edges shown in bold correspond to cells within a domain.

## 3 Domain Properties

### 3.1 Regular Domains

Having reconstructed the machines for the two regular languages $\Lambda^0$ and $\Lambda^1$ from the observed space-time data, we must now prove that they are domains. As mentioned in the introduction, this entails showing that each language satisfies two requirements: temporal invariance and spatial homogeneity. For more details on the notion of domain used here, including the formal definition of regular domain, see Ref. [8].

A language $\mathcal{L}$ is temporally invariant if it is mapped onto itself by the dynamic. Thus, we generate the image of $\mathcal{L}$ under the CA rule $\mathbf{\Phi}$:

$$\mathbf{\Phi}(\mathcal{L}) = \{\mathrm{s}' : \mathrm{s}' = \Phi(\mathrm{s}),\ \mathrm{s} \in \mathcal{L}\} \tag{2}$$

The language is temporally invariant if $\mathbf{\Phi}(\mathcal{L}) = \mathcal{L}$. If $\mathcal{L}$ is a regular language, then so is $\mathbf{\Phi}(\mathcal{L})$, and the "finite machine evolution" (FME) algorithm developed in Ref. [8] gives a prescription for constructing $\mathbf{\Phi}(\mathcal{L})$ explicitly. This makes testing for invariance a practical task, even when $\mathcal{L}$ contains an infinite number of words, as is the case here. Application of the FME algorithm to the regular languages $\Lambda^0$ and $\Lambda^1$ shows that, under the action of CA rule 2614700074, they are both invariant: $\mathbf{\Phi}(\Lambda^0) = \Lambda^0$ and $\mathbf{\Phi}(\Lambda^1) = \Lambda^1$. This constitutes a computer proof of invariance. Details of this proof technique can be found, for a much simpler example, in Ref. [8].

The second defining characteristic of a regular domain, spatial homogeneity, is a form of spatial translation invariance. It is similar to statistical stationarity of the spatial pattern, and is defined in Ref. [8] as the requirement that the process graph of the domain be strongly connected. From Figure 2 it is obvious that the process graphs of $\Lambda^0$ and $\Lambda^1$ are strongly connected, and hence $\Lambda^0$ and $\Lambda^1$ are both spatially homogeneous. This completes the proof of the following result.

**Proposition**: $\Lambda^0$ and $\Lambda^1$ are regular domains.

## 3.2 Pattern Quantifiers

From the structure of the process graphs and the transition probabilities estimated from the reconstructed machines, we can directly calculate the spatial entropies and complexities of the domains.[13,14]

The topological entropy per symbol $h_0(\mathcal{L})$ of a regular language $\mathcal{L}$ is related to the exponential growth rate in number of distinct patterns in $\mathcal{L}$:

$$h_0(\mathcal{L}) = \lim_{L\to\infty} \frac{\log N_{\mathcal{L}}(L)}{L} \tag{3}$$

where $N_{\mathcal{L}}(L)$ is the number of distinct patterns (or "words") of length $L$ in $\mathcal{L}$. From the minimal process graph $\mathrm{M}(\mathcal{L})$ of the domain language $\mathcal{L}$, we construct the connection matrix $\mathrm{T}(\mathcal{L})$ by setting each element $t_{ij}$ of $\mathrm{T}(\mathcal{L})$ equal to the number of edges in $\mathrm{M}(\mathcal{L})$ that lead from state $i$ to state $j$. Then $h_0(\mathcal{L})$ is given by the formula

$$h_0(\mathcal{L}) = \log \lambda_{\max} \tag{4}$$

where $\lambda_{\max}$ is the largest eigenvalue of $\mathrm{T}(\mathcal{L})$. Similarly, the metric entropy, which takes into account probabilistic structure of the domain, is defined as

$$h_1(\mathcal{L}) = \lim_{L\to\infty} -\frac{1}{L} \sum_{\substack{\omega\in\mathcal{L} \\ |\omega|=L}} p(\omega) \log p(\omega) \tag{5}$$

where $p(\omega)$ is the probability of occurrence of the word $\omega$ and the sum is taken over all distinct words of length $L$. The metric entropy is estimated from the empirical machine by

$$h_1(\mathcal{L}) = -\sum_{u\in\mathbf{V}} p(u) \sum_{\substack{v\in\mathbf{V} \\ \sigma\in A}} p_{\underset{\sigma}{u\to v}} \log p_{\underset{\sigma}{u\to v}} \tag{6}$$

where $\mathbf{V}$ is the set of states of $\mathrm{M}(\mathcal{L})$, $p(u)$ is the asymptotic probability of visiting state $u$ and $p_{\underset{\sigma}{u \rightarrow v}}$ is transition probability from state $u$ to state $v$ on symbol $\sigma$, i.e., the probability that the machine will move into state $v$ upon receiving the symbol $\sigma$ as input, given that it is currently in state $u$.

The (finitary) topological and statistical complexities of a regular language, $C_0(\mathcal{L})$ and $C_1(\mathcal{L})$ respectively, are related to the amount of "memory" in the process that generates that language, and are given as

$$\begin{aligned} C_0(\mathcal{L}) &= \log \|\mathbf{V}\| \\ C_1(\mathcal{L}) &= -\sum_{v \in \mathbf{V}} p(v) \log p(v) \end{aligned} \tag{7}$$

where again, $\mathbf{V}$ is the set of states in the minimal process graph of $\mathcal{L}$, $\|\mathbf{V}\|$ is the cardinality of $\mathbf{V}$, and $p(v)$ is the asymptotic probability of visiting state $v$. More information on $C_0(\mathcal{L})$ and $C_1(\mathcal{L})$, and on the calculation of entropy and complexity of a language from its minimal process graph, can be found in Refs. [13,14].

Table 1 shows the entropies and complexities of the domains $\Lambda^0$ and $\Lambda^1$, as calculated from their process graphs. The interesting thing to note is that the two domains differ in both entropy and complexity: $\Lambda^0$ has higher entropy, and lower complexity, than $\Lambda^1$. For each domain $h_0 = h_1$, indicating that the branching states in each domain have equal transition probabilities. Because the process graphs of both $\Lambda^0$ and $\Lambda^1$ are cyclic loops, and because the single branching state has equal branch probabilities, $C_0 = C_1$ as well. In general, however, the metric and topological quantities are related by $h_1 \leq h_0$, $C_1 \leq C_0$.[†]

The invariance of the domain languages, proved using the FME operator, implies the spatial and temporal invariance of the topological entropy and complexity densities $h_0$ and $C_0$ within each domain. The equivalent proof of invariance of the metric quantities $h_1$ and $C_1$ uses an extension of the FME operator to machines with probabilities on the edges. We chose a different tack here, however. For each domain $\Lambda^i$, we generated a random initial condition $\mathrm{s}_0 \in \Lambda^i$ of size $N = 1 \times 10^6$ and allowed it to evolve under the CA rule for $t = 200$ steps. We then applied machine reconstruction to both $\mathrm{s}_0$ and $\mathrm{s}_{200}$, and measured the metric entropy and statistical complexity at the two times. The comparison showed that for both $\Lambda^0$ and $\Lambda^1$, $h_1$ and $C_1$ are invariant.

The next pair of quantities we measure relate to the single-site statistics of states in the ensemble. Again, there are topological and metric versions of this. The metric quantity, $\rho_1(\mathcal{L})$, is the fraction of cells with nonzero values, averaged over all states in the ensemble:

$$\rho_1(\mathcal{L}) = \|\mathcal{L}\|^{-1} \sum_{\omega \in \mathcal{L}} \frac{1}{|\omega|} \sum_{i=1}^{|\omega|} \Theta(\sigma^i) \tag{8}$$

where $\|\mathcal{L}\|$ is the cardinality of $\mathcal{L}$, $\sigma^i$ is the $i$th symbol of $\omega$, and $\Theta(\sigma) = 0$ if $\sigma = 0$ and $\Theta(\sigma) = 1$ for all $\sigma \neq 0$.[‡] The corresponding topological quantity, $\rho_0(\mathcal{L})$, is the fraction of

[†]Because the equal branching probabilities, the fluctuation spectra (see [15]) for the domains are both single points.

[‡]This is the formula for $\mathcal{L}$ finite, which is used in numerical Monte Carlo estimates or estimates made at fixed lattice size $N$. For $\mathcal{L}$ infinite, we restrict eq. 8 to words of length $|\omega| = L$, and then take the limit $L \rightarrow \infty$.

nonzero cells in the CA lookup table outputs of parent neighborhoods occurring in $\mathcal{L}$. The set $\Pi_{\mathcal{L}}$ of parent neighborhoods in $\mathcal{L}$ is identical to the set of words in $\mathcal{L}$ of length $2r+1$, where $r$ is the radius of the CA rule:

$$\Pi_{\mathcal{L}} = \{\omega : \ \omega \in \mathcal{L}, \ |\omega| = 2r+1\} \tag{9}$$

Then $\rho_0(\mathcal{L})$ is just the fraction of the set $\Pi_{\mathcal{L}}$ that map to nonzero values under the rule:

$$\rho_0(\mathcal{L}) = \frac{\sum\limits_{\substack{\sigma\in\mathcal{A}\\ \sigma\neq 0}} \eta_{\mathcal{L}}(\sigma)}{\sum\limits_{\sigma\in\mathcal{A}} \eta_{\mathcal{L}}(\sigma)} \tag{10}$$

where $\eta_{\mathcal{L}}(\sigma)$ is the number of parent neighborhoods in $\Pi_{\mathcal{L}}$ that map to the value $\sigma$ under the CA rule $\phi$. If all possible parent neighborhoods occur in the ensemble $\mathcal{L}$, then $\rho_0(\mathcal{L})$ is equivalent to the "lambda parameter" defined in [16]. Our choice of notation for these quantities emphasizes the ensemble-dependence of their definitions and differentiates them from the "Lyapunov exponents" defined below. The values of $\rho_0$ and $\rho_1$ for both domains are shown in Table 1. The large difference between the values of $\rho_1(\Lambda^0)$ and $\rho_1(\Lambda^1)$ is what makes the two domains so easy to identify visually in Figure 1. However, we must emphasize that $\rho_0(\mathcal{L})$ does not reflect the domains' evolution, and so may or may not be relevant, and $\rho_1(\mathcal{L})$ is not indicative of the domains' structural regularity.

The final row in Table 1 gives the values of the left- and right-Lyapunov exponents $\lambda_L$ and $\lambda_R$, discussed in Section 3.4 below.

| Quantifier | $\Lambda^0$ | $\Lambda^1$ | $\mathbf{\Phi}_t(\mathcal{A}^*)$ |
|---|---|---|---|
| $h_0$ | 1/2 | 1/4 | 0.9(†) |
| $h_1$ | 0.500 | 0.250 | 0.7(†) |
| $C_0$ | 1 | 2 | 5.5(†) |
| $C_1$ | 1.000 | 2.000 | 2.1(†) |
| $\rho_0$ | 4/11 | 6/10 | 15/32 |
| $\rho_1$ | 0.250 | 0.625 | 0.41(†) |
| $\lambda_L, \lambda_R$ | -2, 2 | -2, 2 | -1.64, 1.73 |

Table 1 Quantifiers for the two domains $\Lambda^0$ and $\Lambda^1$, and for the full state space $\mathcal{A}^*$. See the text for definitions of the quantifiers. Fractional numbers and integers without decimal points are exact quantities. All others were estimated numerically, and have an uncertainty of $\pm 1$ in the last decimal place. Numbers marked with the symbol (†) are nonstationary quantities, and were measured on a lattice of size $N = 1 \times 10^6$ at time $t = 200$.

For comparison, we have also included a third column in Table 1 that gives estimates of values of the domain quantifiers for the full state space $\mathcal{A}^*$. This is despite the fact that most of the quantifiers are nonstationary, due to the global contraction of the state space under the dynamic, i.e.,

$$\mathbf{\Phi}_{t+1}(\mathcal{A}^*) \subset \mathbf{\Phi}_t(\mathcal{A}^*), \ \forall t \geq 0 \tag{11}$$

and generally to the creation and annihilation of domains, walls, particles, and other structures. For example, the topological complexity $C_0(\mathbf{\Phi}_t(\mathcal{A}^*))$ appears to grow without bound, as the number of states in the machines for the sequence of languages $\{\mathbf{\Phi}_t(\mathcal{A}^*),\ t = 0, 1, 2, ...\}$ rapidly diverges under the action of the FME operator.[17] For the entropies, complexities, and single-site statistics in Table 1, we estimated the values using a single state $\mathrm{s}_t$ evolved from a random initial condition $\mathrm{s}_0 \in \mathcal{A}^*$ of size $N = 1 \times 10^6$, iterated up to time $t = 200$. The entropies $h_0(\mathbf{\Phi}_t(\mathcal{A}^*))$ and $h_1(\mathbf{\Phi}_t(\mathcal{A}^*))$, and the density of nonzero cells $\rho_1(\mathbf{\Phi}_t(\mathcal{A}^*))$ were estimated from the statistics of subsequences of $\mathrm{s}_t$ of length $L = 18$. The complexities $C_0(\mathbf{\Phi}_t(\mathcal{A}^*))$ and $C_1(\mathbf{\Phi}_t(\mathcal{A}^*))$ were estimated by the topological and metric total excess entropies, again measured over subsequences of length $L = 18$.[18] The (topological) fraction of nonzero cells $\rho_0(\mathbf{\Phi}_t(\mathcal{A}^*))$, being a property of the rule table and initial ensemble alone, was calculated exactly.

The most important thing to realize from the comparison between the values of domain quantities for the two domains on the one hand, and the full state space on the other, is that the global average can be misleading. The average density $\rho_1$, for example, is misleading in the same way that the mean of a bimodal distribution is misleading: it fails to capture the fact that there are two distinct types of behavior. For example, take any small region of space-time in the evolution of a random initial condition. Typically, this region is in one or the other domain. The density of nonzero cells in the region is not halfway between the densities of the two domains, it is simply one or the other. That is, there is no region in which the density of nonzero cells is $\rho_1 \approx 0.41$. Excepting the domain wall regions, which do not play a dominant role in the average (though for other rules they could), there are only regions with a density of nonzero cells close to either $\rho_1 \approx 0.25$ or $\rho_1 \approx 0.63$.

## 3.3 Domain Difference Patterns

One graphical measure of the structure of information flow in a CA is the *difference pattern*.[19] Here we introduce two important modifications of this technique. We refer to the resulting method as domain-dependent difference patterns. The consequences are striking and demonstrate the important structural role played by regular domains.

To generate a difference pattern, we start with a given "fiducial" state $\mathrm{s}_0 = ...\sigma_0^{i-1}\sigma_0^i\sigma_0^{i+1}...$ in some pattern ensemble $\mathcal{L}$ and create a new state $\tilde{\mathrm{s}}_0 = ...\tilde{\sigma}_0^{i-1}\tilde{\sigma}_0^i\tilde{\sigma}_0^{i+1}...$ that differs from $\mathrm{s}_0$ in a single cell. This is done by adding to $\mathrm{s}_0$ the perturbation $\delta^j$ consisting of a single nonzero cell at site $j$: $\sigma^j = 1$, $\sigma^{i \neq j} = 0$; this is the smallest possible variation in the local state. The result is

$$\tilde{\mathrm{s}}_0 = \mathrm{s}_0 \dot{+} \delta^j \tag{12}$$

where "$\dot{+}$" denotes cell-by-cell addition modulo $k = \|\mathcal{A}\|$, the alphabet size. In addition, we require that cell $j$ be chosen so that the perturbed state $\tilde{\mathrm{s}}_0$ is still in $\mathcal{L}$; for $\Lambda^0$ and $\Lambda^1$, this is accomplished by requiring that cell $j$ be a wild-card. Both the fiducial and perturbed states are iterated independently using the CA rule, and at each time step we plot the difference pattern

$$\mathbf{\Delta}_t^{\mathcal{L}} = \mathrm{s}_t \dot{-} \tilde{\mathrm{s}}_t \tag{13}$$

where $\dot{-}$ is subtraction modulo $k$. The resulting picture shows the space-time cells whose values are dependent on the value of the perturbed cell $\sigma_0^j$. Not surprisingly, the difference pattern is dependent on the particular choice of initial condition $\mathrm{s}_0$. Figure 5 shows space-time plots of the difference patterns $\mathbf{\Delta}_t^{\mathcal{L}}$ generated by states chosen randomly from three different sets: $\mathcal{L} = \mathcal{A}^N$ (Figure 5(a)), $\mathcal{L} = \Lambda^0$ (Figure 5(b)), and $\mathcal{L} = \Lambda^1$ (Figure 5(c)). In each case, the evolution was on a lattice of size $N = 100$ with periodic boundary conditions, and the perturbed cell was located at $j = N/2$. Note that for each of these patterns, the maximum instantaneous rate at which the perturbation spreads in either direction is equal to the "speed of light" of this rule, $c = 2$ cells/iteration. However, beneath this spreading light cone, the patterns differ markedly. The difference pattern $\mathbf{\Delta}_t^{\mathcal{A}^N}$ for $\mathrm{s}_0 \in \mathcal{A}^N$ shows an irregular structure, while the patterns for the domains $\Lambda^0$ and $\Lambda^1$ both have the same regular structure. The latter two are in fact identical to each other, $\mathbf{\Delta}_t^{\Lambda^0} = \mathbf{\Delta}_t^{\Lambda^1}$ for all $t$. When every other cell in space is removed — using the decimation operator $\mathrm{T}_{\mathrm{decimate}}(\Lambda)$ of Ref. [8] — they are identical to the space-time diagram of the chaotic, linear elementary CA rule 90 with an initial condition consisting of a single nonzero cell. That is,

$$\mathrm{T}_{\mathrm{decimate}}\left(\mathbf{\Delta}_t^{\Lambda^i}\right) = \Phi_t^{90}(0^*10^*) \tag{14}$$

## 3.4 Domain Difference Plumes

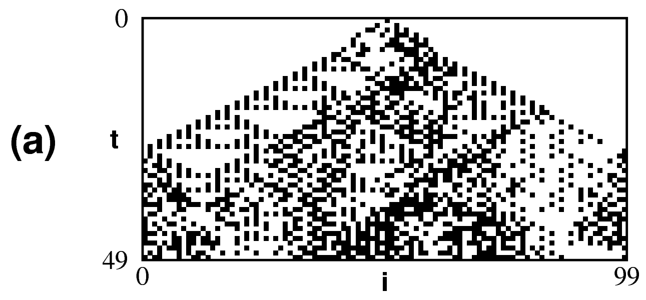
To investigate the statistical information transmission properties of an entire ensemble, we use the *difference plume*, or averaged difference pattern. For each state $\mathrm{s}_0$ in an ensemble $\mathcal{L}$, we form the difference pattern as above, creating an ensemble of "difference pairs" for a given perturbation $\delta^j$: $\left\{(\mathrm{s}_0, \tilde{\mathrm{s}}_0) : \tilde{\mathrm{s}}_0 = \mathrm{s}_0 \dot{+} \delta^j,\ \mathrm{s}_0, \tilde{\mathrm{s}}_0 \in \mathcal{L}\right\}$. As above, both the original and the perturbed state must be in the ensemble; i.e. the $\sigma^j = 1$ in $\delta^j$ appears only at wild card sites. Then at each point $(i, t)$ in space-time, we average $\Delta_t^i = \sigma_t^i \dot{-} \tilde{\sigma}_t^i$ over the ensemble:

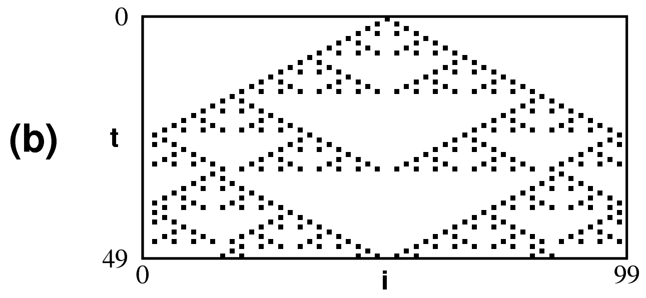
$$\langle \Delta_t^i \rangle = \|\mathcal{L}\|^{-1} \sum_{\mathrm{s}_t \in \mathcal{L}} \Delta_t^i \tag{15}$$

Thus $\langle \Delta_t^i \rangle$ is the fraction of difference pairs for which $\sigma_t^i \neq \tilde{\sigma}_t^i$. At $t = 0$, for example, $\langle \Delta_0^i \rangle = 1$ for $i = j$ and $\langle \Delta_0^i \rangle = 0$ for all $i \neq j$.

The difference plumes for the three ensembles $\mathcal{A}^N$, $\Lambda^0$, and $\Lambda^1$ are shown in Figures 6(a), 6(b), and 6(c), respectively. Each plume was calculated using a Monte Carlo sample of $1000$ state pairs on a lattice of size $N = 100$, with the perturbation in cell $j = N/2$. The difference plume for $\mathcal{A}^N$ (Figure 6(a)) is devoid of internal structure, with $\langle \Delta_t^i \rangle$ smoothly decreasing toward the edges of the light cone. The plumes for $\Lambda^0$ and $\Lambda^1$ are quite different from this. In both of the latter, the averaged difference plume is identical to the difference pattern generated by a single initial condition. At each space-time point $(i, t)$, $\langle \Delta_t^i \rangle$ is either identically zero or one. Thus all states in $\Lambda^0$ have the same difference pattern—i.e., the propagation of information is identical for all states in the ensemble. The situation is exactly the same for $\Lambda^1$, even though the two domains have strikingly different spatial structure. The important thing to note is that, without having identified the ensembles $\Lambda^0$ and $\Lambda^1$, we would never have noticed any structure in the difference plumes at all. The average over all of state space washes it out entirely.

Three quantitative measures of the plume serve to emphasize this difference. First, we measure the average rate at which the difference plume spreads. This is given by the two quantities $\lambda_L(\mathcal{L})$, $\lambda_R(\mathcal{L})$, which are the linearized average velocities of the left and right edges





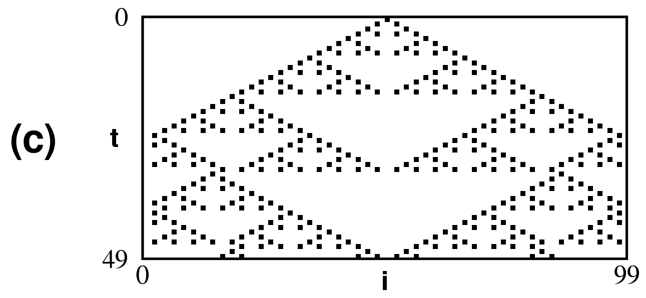


Figure 5 Difference patterns $\Delta_t^{\mathcal{L}}$: (a) $\mathcal{L} = \mathcal{A}^N$. Global difference pattern in which the initial condition was randomly selected over the entire state space: $\mathrm{s}_0 \in \mathcal{A}^N$; (b) random IC in domain $\mathcal{L} = \Lambda^0$; (c) random IC in domain $\mathcal{L} = \Lambda^1$. In each plot, evolution was on a lattice of size $N = 100$ and the initial condition was perturbed in cell $j = 50$. The patterns of (b) and (c) are identical, and are also identical to an "expanded" space-time diagram of elementary CA rule 90 in which a 0 is inserted between every horizontal pair of cells.

of the plume, respectively. These are the left- and right-Lyapunov exponents [20,21] restricted to the ensemble $\mathcal{L}$. They are defined as follows.

For each difference pattern in the ensemble, denote the leftmost and rightmost cells $\sigma_t^i$ for which $\Delta_t^i \neq 0$ by $\mathrm{Left}(\Delta_t)$ and $\mathrm{Right}(\Delta_t)$. That is,

$$\begin{aligned} \mathrm{Left}(\Delta_t) &= i \ : \ \Delta_t^i \neq 0, \ \Delta_t^j = 0 \,\forall\, j < i \\ \mathrm{Right}(\Delta_t) &= i \ : \ \Delta_t^i \neq 0, \ \Delta_t^j = 0 \,\forall\, j > i \end{aligned} \tag{16}$$

Then the left and right edges are defined by

$$\begin{aligned} \mu_L(t) &= \frac{1}{\|\mathcal{L}\|} \sum_{\mathrm{s}_0 \in \mathcal{L}} \mathrm{Left}(\Delta_t) \\ \mu_R(t) &= \frac{1}{\|\mathcal{L}\|} \sum_{\mathrm{s}_0 \in \mathcal{L}} \mathrm{Right}(\Delta_t) \end{aligned} \tag{17}$$

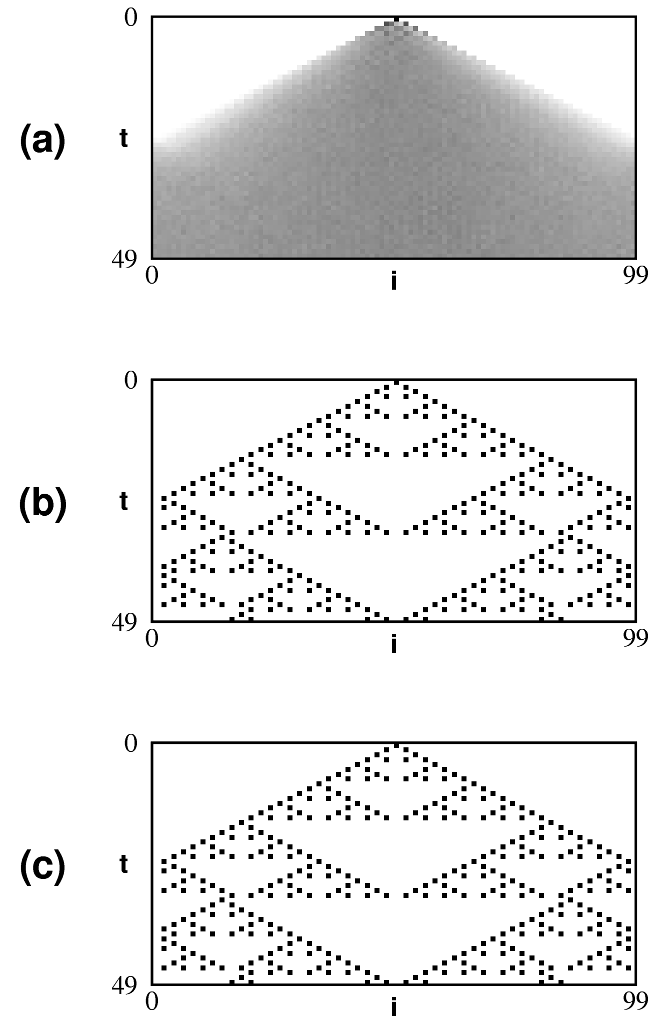


Figure 6 Difference plumes $\langle \Delta_t^i \rangle$: (a) $\mathcal{L} = \mathcal{A}^N$. Global difference plume averaged over initial conditions randomly selected over the entire state space; (b) initial conditions in domain $\mathcal{L} = \Lambda^0$; (c) initial conditions in domain $\mathcal{L} = \Lambda^1$. Note that while (a) appears to be a smooth surface, both (b) and (c) show structure. In fact, (b) and (c) are identical to the patterns of Figure 5(b) and 5(c). For each plot, $M = 1000$ pairs of states were iterated on a lattice of size $N = 100$. As above, each initial condition was perturbed in cell $j = 50$.

The Lyapunov exponents $\lambda_L$ and $\lambda_R$ are the linearized growth rates of $\mu_L(t)$ and $\mu_R(t)$ with time:

$$\begin{aligned} \mu_L(t) &= j + \lambda_L t + \mathcal{O}(t^2) \\ \mu_R(t) &= j + \lambda_R t + \mathcal{O}(t^2) \end{aligned} \tag{18}$$

For both of the domains, $\lambda_L$ and $\lambda_R$ are evidently $-2$ and $+2$, respectively. We measured the values for $\mathcal{A}^*$ using an ensemble of $M = 1000$ pairs of states on a lattice of size $N = 1000$. For each pair, the left- and right-most cells in which the two states differed was recorded, and the average positions $\mu_L(t)$ and $\mu_R(t)$ over the ensemble were measured. We plotted both of these quantities versus time, and fit each curve to a polynomial, identifying $\lambda_L$ and $\lambda_R$ with the linear terms. The values are shown in Table 1.

The third quantity associated with the difference plume is its total volume $\zeta_{\mathcal{L}}(t)$, defined as the average number of cells in which the perturbed and unperturbed states differ at time $t$:

$$\zeta_{\mathcal{L}}(t) = \sum_i \langle \Delta_t^i \rangle \tag{19}$$

Note that at $t = 0$, $\zeta_{\mathcal{L}}(0) = 1$ for all $\mathcal{L}$. From Figure 6(a) it is clear that $\zeta_{\mathcal{A}^*}(t)$ increases smoothly with $t$, while $\zeta_{\Lambda^0}(t)$ and $\zeta_{\Lambda^1}(t)$ fluctuate in an orderly but discontinuous fashion. To further illustrate this point, we show in Figure 7 a plot of $\zeta_{\mathcal{L}}(t)$ versus $t$ for $\mathcal{L} = \Lambda^0$ and $\mathcal{L} = \mathcal{A}^*$. As expected, the curve of $\zeta_{\mathcal{A}^*}(t)$ versus $t$ (the dashed line) increases smoothly from $\zeta_{\mathcal{A}^*}(0) = 1$, with a very slight upward curvature. A least-squares fit to a power law form $\zeta_{\mathcal{A}^*}(t) = c_1 t^{\alpha} + c_2$ gives an exponent of $\alpha = 1.03$; alternatively, fitting $\zeta_{\mathcal{A}^*}(t)$ to a polynomial gives a linear slope of $m = 1.42$. The plot of $\zeta_{\Lambda^0}(t)$ versus $t$, printed as a solid line, is quite different. All values of $\zeta_{\Lambda^0}(t)$ are powers of 2. Applying the above-mentioned equivalence of the difference pattern $\mathbf{\Delta}_t^{\Lambda^0}$ and the evolution of a state with a single nonzero cell under elementary CA 90, we find the exact formula

$$\zeta_{\Lambda^0}(t) = 2^{\kappa(t)} \tag{20}$$

where $\kappa(n)$ is the number of 1's in the binary expansion of $n$.[22] Since the difference plumes for $\Lambda^0$ and $\Lambda^1$ are identical, so are the plume volumes: $\zeta_{\Lambda^1}(t) = \zeta_{\Lambda^0}(t)$ for all $t$.

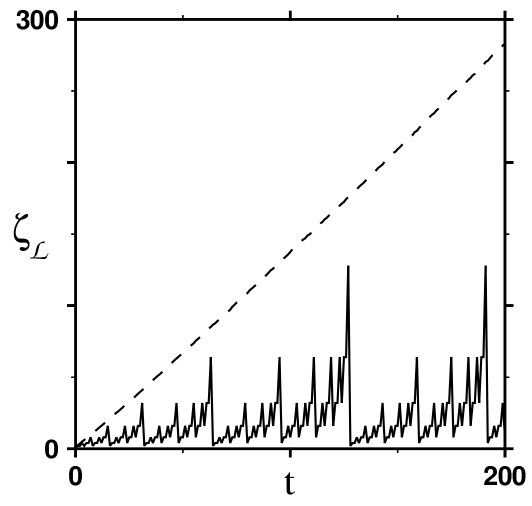


Figure 7 Difference plume volumes $\zeta_{\mathcal{L}}(t)$ versus $t$ for the entire state space $\mathcal{L} = \mathcal{A}^*$ (dashed line) and for domain $\mathcal{L} = \Lambda^0$ (solid line). The plume volume for the other domain, $\mathcal{L} = \Lambda^1$, is identical to $\zeta_{\Lambda^0}(t)$.

## 4 Domain Interfaces

Having identified the regular domains $\Lambda^0$ and $\Lambda^1$ and measured their properties, it is natural to turn next to the study of the interface between two adjacent domains. This section gives a brief survey of the temporal evolution of states consisting of two domains laid side-by-side, focusing in particular on the region where they meet. We make no attempt at a complete characterization, but merely indicate the starting point. Our purpose is to illustrate the utility of analyzing the state space in terms of the pattern bases: i.e., to show how knowledge of the domains gives a new systematic view of the variety of the space-time structures—walls, dislocations, and (possibly) "particles"—that a spatial system can generate. The analysis that we indicate here has been carried out in detail for a simpler system in Ref. [11].

We are concerned with the behavior of states initially consisting of two concatenated domains $\Lambda^i$ and $\Lambda^j$, with the wall between the two at a given cell $m$. Each domain has a specific phase, but is otherwise chosen at random. As in Ref. [8], we denote ensembles of such “two-domain” states by $\Lambda^{i,j}$. We then allow the state to evolve, and analyze the structure of the interface between the domains. On a finite lattice, the periodic boundary conditions create another wall at cell 0, but in our investigations we will be using sufficiently large lattices that this second wall can be ignored.

One important factor in the evolution of two-domain states is the *relative phase* between the domains. This is defined as follows. Let the wall between the domains $\Lambda^i$ and $\Lambda^j$ be situated at cell $m$. Recalling the definition of the phase $\phi^i(m)$ of $\Lambda^i$ from Section 2 above, define the *relative phase* $\delta\phi^{ij}$ between $\Lambda^i$ and $\Lambda^j$ as the difference in phase of the two domains at the wall:

$$\delta\phi^{ij} = \phi^j(m) - \phi^i(m) \tag{21}$$

Strictly speaking, the phase of $\Lambda^i$ at the wall cell $m$ is undefined, so we take $\phi^i(m)$ to be the phase that $\Lambda^i$ *would have had* if the wall had not been present. For binary cell values, this latter quantity is unambiguous; extending the definition to larger alphabets requires a slight alteration of the formulation. The range of allowed values of the relative phase depends on the number of states in the process graphs of the domains on either side. For the two domains we are studying, we have $\delta\phi^{00} \in \{-1,0,1\}$, $\delta\phi^{01} \in \{-1,...,3\}$, $\delta\phi^{01} \in \{-3,...,1\}$, and $\delta\phi^{11} \in \{-3,...,3\}$. Note also that a state in $\Lambda^{i,i}$ with $\delta\phi^{ii} = 0$ does not in fact contain a wall, since the two parts of the state have the same domain type and same phase at the junction. However, for states in $\Lambda^{i,j}$ with $i \neq j$, there is a wall for all values of $\delta\phi^{ij}$. Note also that some values of relative phase are equivalent; for example, in our case, $\delta\phi^{00} = +1$ and $\delta\phi^{00} = -1$ describe the same situation. Similar degeneracies for the other domain pairs reduce the number of distinct interface types to eight, six of which are discussed below.

## 4.1 Interface Plots

Filtered space-time plots of six randomly selected two-domain states, representing a subset of the possible combinations of domains and relative phases, are shown in Figure 8. Examples of all four possible domain pairs $\Lambda^{i,j}$ are shown, and in addition we show two examples each of $\Lambda^{1,0}$ and $\Lambda^{1,1}$ with different values of $\delta\phi$, to investigate the importance of the relative phase. Reading the Figure from left to right, top to bottom, they are (a) $\Lambda^{0,0}$ with relative phase $\delta\phi^{00} = 1$, (b) $\Lambda^{0,1}$ with $\delta\phi^{01} = 0$, (c) $\Lambda^{1,0}$ with $\delta\phi^{10} = 0$ (d) $\Lambda^{1,0}$ with $\delta\phi^{10} = 1$, (e) $\Lambda^{1,1}$ with $\delta\phi^{11} = 1$, and (f) $\Lambda^{1,1}$ with $\delta\phi^{11} = -1$. The two interface types not shown are (g) $\Lambda^{0,1}$ with $\delta\phi^{01} = 1$ and (h) $\Lambda^{1,1}$ with $\delta\phi^{11} = 2$, for which the evolution of walls is like interfaces (a) and (e), respectively. Cells in $\Lambda^0$ are shown in white, cells in $\Lambda^1$ are gray, and all walls are black. The lattice size is $N = 1000$, of which only the central portion is displayed. Each state starts out at $t = 0$ with a single wall at cell $m = 500$.

Perhaps the most obvious feature of the plots in Figure 8 is that behavior of the interface can vary widely depending on the domains to either side. In the roughest approximation, there are two qualitatively different types of evolution: (i) spreading interface, and (ii) localized drift.

In Figures 8(a), 8(b) and 8(e), the number of walls is plainly not conserved. The single wall at $t = 0$ splits into a whole population of walls that move about, annihilate each other, and themselves split into multiple offspring. The interfacial region, i.e., the region in which walls are

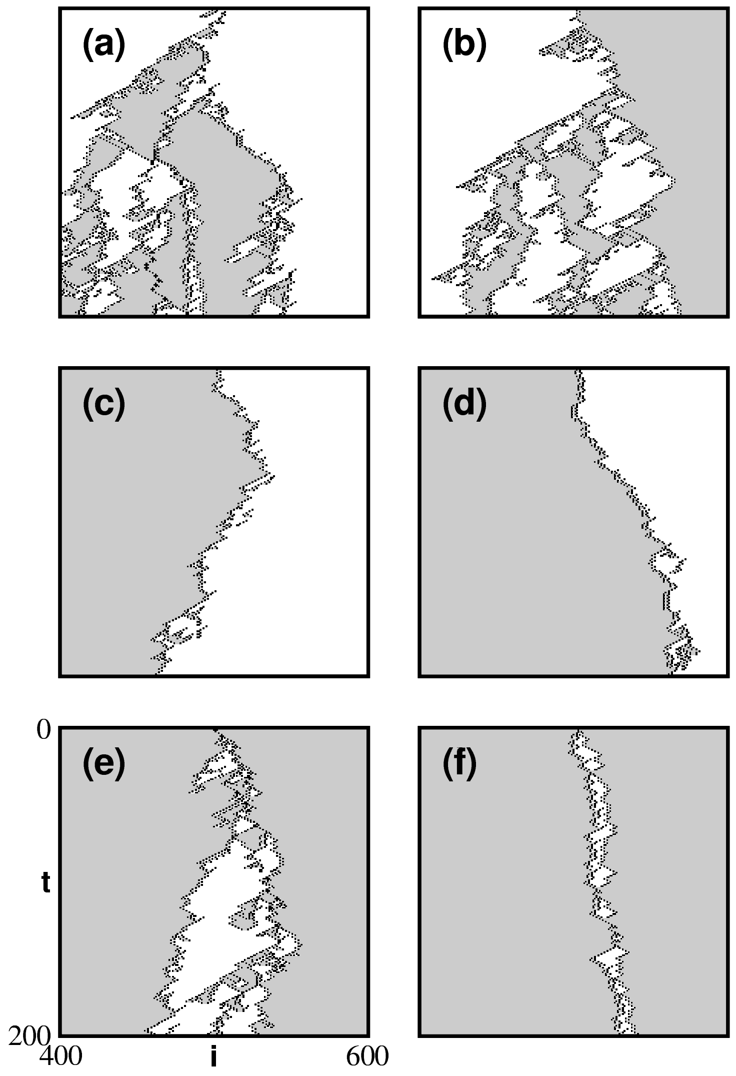


Figure 8 Evolution of domain interfaces. Each state starts out with a single wall at cell $j = 500$. The initial conditions were chosen randomly from the following ensembles: (a) $\Lambda^{0,0}$ with relative phase $\delta\phi^{00} = 1$; (b) $\Lambda^{0,1}$ with $\delta\phi^{01} = 0$; (c) $\Lambda^{1,0}$ with $\delta\phi^{10} = 0$; (d) $\Lambda^{1,0}$ with $\delta\phi^{10} = 1$; (e) $\Lambda^{1,1}$ with $\delta\phi^{11} = 1$; and (f) $\Lambda^{1,1}$ with $\delta\phi^{11} = -1$. Cells in the interior of $\Lambda^0$ are shown in white, cells in $\Lambda^1$ are gray, and all walls are black.

present, spreads rapidly to either side. In these plots, the identification of walls as "particles" is problematic at best, since the interfaces do not possess even the most basic attribute of particles, spatial localization.

In Figures 8(c), 8(d), and 8(f), on the other hand, the interfacial region remains localized in space, and drifts as a whole in one direction or another. The number of walls fluctuates but does not blow up as in the other plots. From time to time, a single wall does split into multiple offspring, but these offspring are quickly annihilated. These interfaces behave as localized "real" particles surrounded by a cloud of interacting "virtual" particles.

The diffusive character of wall motion is also immediately apparent. The walls in each plot appear to following some type of random walk, with or without a net drift. This statement will be made more quantitative in the next section.

The dependence on relative phase is illustrated in Figures 8(e) and (f). Both plots start with a state in $\Lambda^{1,1}$, but the relative phase differs. In Figure 8(e), the interface spreads, whereas in Figure 8(f), it remains localized.

## 4.2 Interface Plumes

The plots in Figure 8 show only a single state in each of their respective ensembles. It is possible, however, that these plots show behavior that is atypical of the ensembles. To characterize the ensembles as a whole, it is necessary to gather statistics over all the states, or at least a Monte Carlo sample of them. We do this by plotting the *interface plumes*, that is, the sum of the interfaces of the states in each two-domain ensemble. For each of the ensembles plotted in Figure 8, we generated a Monte Carlo sample of $M = 1000$ states of size $N = 1000$, and evolved each state up to time $t = 200$. The structure of the plumes is made clearer by plotting a few simple statistics, defined below. These are all measures of the positions of the walls in the interface, and are closely related to the difference plume statistics defined in the last section.

Let $\theta_t^i$ be the sign function of the output of the domain filter $\mathrm{T}_\Lambda$ upon reading the symbol $\sigma_t^i$. That is, let $\theta_t^i = 1$ whenever $\sigma_t^i$ is a wall, and $\theta_t^i = 0$ otherwise. The ensemble average of $\theta_t^i$ gives the fraction of states in the ensemble $\mathcal{L}$ for which there is a wall at cell $(i, t)$:

$$\langle \theta_t^i \rangle = \frac{1}{||\mathcal{L}||} \sum_{\mathrm{s}_0 \in \mathcal{L}} \theta_t^i \tag{22}$$

The roughest measure of interface evolution is just its *mean position* (or center) $\mu_C$, defined as the average position of the walls in all states in the ensemble. This is given by the formula

$$\mu_C(t) = \sum_{i=a}^{b} i \langle \theta_t^i \rangle \tag{23}$$

with $a$ and $b$ chosen outside the interfacial region. The mean position will capture the average drift of the walls, but it is insensitive to the width of the interface. For this reason, we also look at the edges of the plume. For any iterate $\mathrm{s}_t$ of a two-domain initial condition $\mathrm{s}_0 \in \Lambda^{i,j}$ there will be a left-most and right-most wall cell. Denote these by $\mathrm{Left}(\theta_t)$ and $\mathrm{Right}(\theta_t)$, respectively. The *mean left edge* $\mu_L$ of the interface is given by the ensemble average

$$\mu_L(t) = \frac{1}{||\mathcal{L}||} \sum_{\mathrm{s}_0 \in \mathcal{L}} \mathrm{Left}(\theta_t) \tag{24}$$

where $\mathcal{L} = \Lambda^{i,j}$ with the chosen relative phase. Similarly, the *mean right edge* $\mu_R$ of the interface is given by

$$\mu_R(t) = \frac{1}{||\mathcal{L}||} \sum_{\mathrm{s}_0 \in \mathcal{L}} \mathrm{Right}(\theta_t) \tag{25}$$

The time-evolution of $\mu_L$ and $\mu_R$ measures the spreading of the interface to the left and right, respectively. The left and right spreading rates are clearly analogous to the Lyapunov exponents defined in the last section. The difference $\mu_R - \mu_L$ is a rough measure of the average interface width. If the interface remains localized, then $\mathrm{Left}(\theta_t) \approx \mathrm{Right}(\theta_t)$ for each state in $\mathcal{L}$, and $\mu_L \approx \mu_R$.

The three quantities $\mu_L$, $\mu_C$ and $\mu_R$ are plotted versus time in Figure 9, for the same ensembles as in Figure 8. They are the three solid lines in each plot, with $\mu_L(t) \le \mu_C(t) \le \mu_R(t)$ at each time $t$. Each plot was generated numerically using a Monte Carlo sample of the appropriate ensemble with $M = 1 \times 10^4$ states on a periodic lattice of size $N = 1000$. In addition to the mean quantities, the absolute left-most and right-most wall detected at each time are plotted as dashed lines.

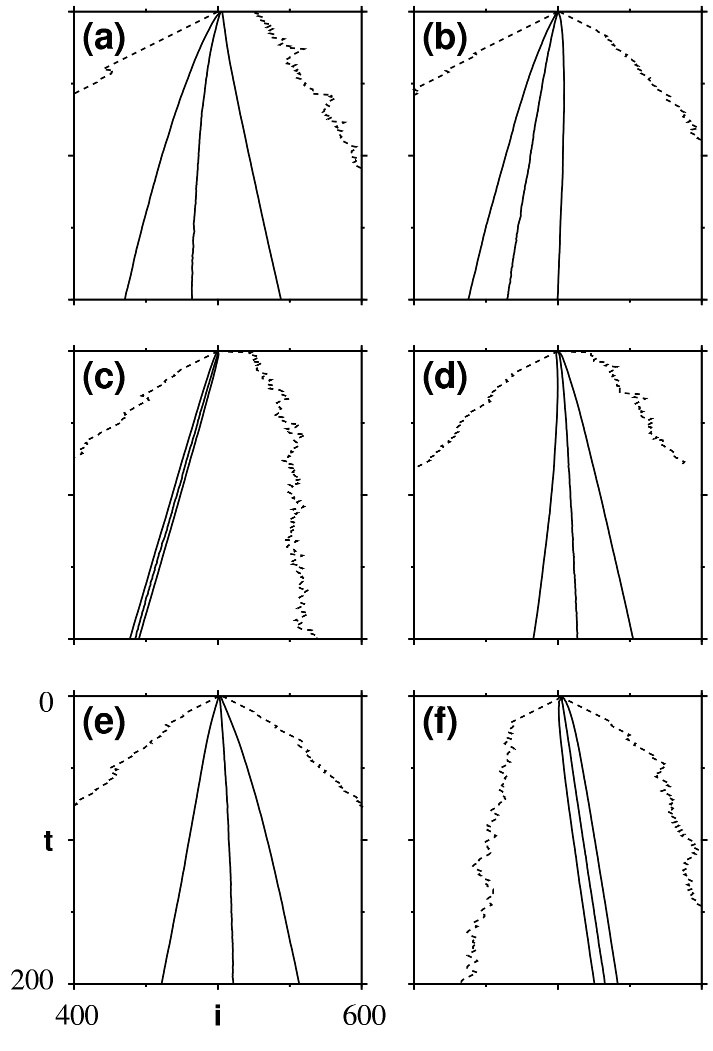


Figure 9 Interface plumes for ensembles of the six types of state shown in Figure 8. The labels (a) — (f) denote the same ensembles as in that Figure.

Figures 9(a) and 9(e) are similar, though plume (a) drifts slowly left, while plume (e) drifts slowly right. States in both of these ensembles are of the "spreading interface" type. The plume in Figure 9(b) also spreads, though more slowly than plumes (a) and (e); in addition it shows a faster net drift to the left.

Figure 9(d) shows a slower rate of spreading, though it is still nonzero. This indicates that the apparently localized interface shown in Figure 8(d) is atypical of the ensemble. While some initial conditions have interfaces that remain localized, there are others in which it spreads more in the fashion of Figure 8(a), 8(b) or 8(e). This suggests that the ensemble plotted Figure 9(d) is composed of sub-ensembles with qualitatively different behavior. It is interesting to note that $\mu_L(t)$ moves to the left at first, but then slowly curves back to the right. We do not at this point have an explanation for this highly unusual behavior.

Figures 9(c) and 9(f), on the other hand, show little or no spreading at all. After an initial growth period, the mean left edge and mean right edge of the plume remain close together. This correlates well with the behavior seen in Figures 8(c) and 8(f). The mean drift velocity, given by the slope of $\mu_C$, is surprisingly constant, although as the dashed lines indicate, there is a great deal of variation in the individual trajectories. It appears that for these two ensembles, the interface does behave as a "particle", at least in an average sense.

As we observed in the last section, the evolution of walls in the plots of Figure 8 appears to be a diffusive process. One very rough measure of the diffusivity of wall motion is given by the growth in the standard deviation $\sigma(t)$ of the distribution of walls in the interface. This statistic neglects the fact that many walls may be present in a single interfacial region, and that those walls are constantly interacting, annihilating each other and giving birth to multiple offspring. Nevertheless it serves to give an indication of the average evolution of the ensemble. For comparison with what follows, we note that the expected form for an ideal random walk is $\sigma(t) = Dt^{1/2}$ with $D$ constant. For each of the ensembles shown in Fig. 9, we measured $\sigma(t)$ for $0 \leq t \leq 200$ and fitted the resulting curve to both power-law and exponential functional forms. For $t \geq 10$, all six curves showed power-law growth $\sigma(t) \propto t^\alpha$ with the following exponents: plumes (a), (b), (d), and (e) all had $\alpha = 0.6$, plume (c) had $\alpha = 0.5$, and plume (f) had $\alpha = 0.4$.

A quantitative measure of the degree of localization of the interface is provided by the distance $\mu_{R-L} = \mu_R - \mu_L$ between the mean left edge and mean right edge of the plume. Least-squares fits of the curves of $\mu_{R-L}(t)$ vs. $t$ indicated that plumes (a), (b), and (e) are well fit to a power law $\mu_{R-L}(t) \propto t^\alpha$ with the exponents as follows: (a) $\alpha = 0.80$; (b) $\alpha = 0.67$; (e) $\alpha = 0.79$. Plumes (c) and (f) for long times ($t > 100$) showed very slow linear growth $\mu_{R-L}(t) = mt + const$ with rates of $m = 0.008$ for plume (c) and $m = 0.04$ for plume (f). The curve for plume (d) was nearly a straight line, with a fit to a quadratic polynomial $\mu_{R-L}(t) = a + bt + ct^2$ giving coefficients $a = 2.9$, $b = 0.29$ and $c = 2.1 \times 10^{-4}$.

Another statistic that differentiates among the plumes is the *interface plume volume* $\xi_{\mathcal{L}}(t)$. This is the total number of walls in the interface, given by the sum over space of $\langle \theta_t^i \rangle$:

$$\xi_{\mathcal{L}}(t) = \sum_{i=a}^{b} \langle \theta_t^i \rangle \tag{26}$$

where the limits $a$ and $b$ are chosen to lie outside the envelope of the interface. The plume volumes for the six ensembles shown in Figures 8 and 9 are plotted versus time in Figure 10.

The plume volume provides another test of whether the interface is behaving as a particle or whether it is unstable. Since a particle is by definition localized in space, it necessarily must have a roughly constant—or at least bounded—number of walls. This indeed is the case for

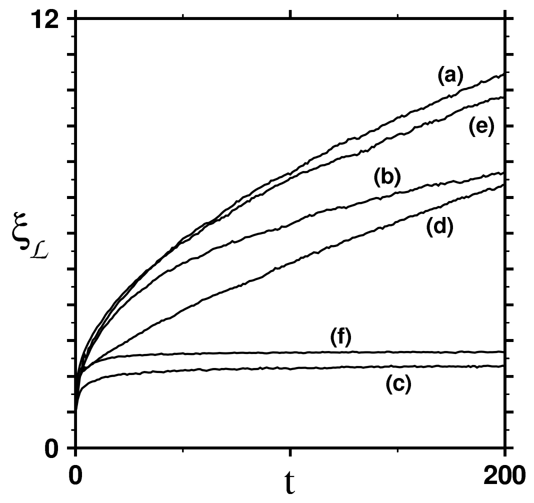


Figure 10 Interface plume volumes $\xi_{\mathcal{L}}(t)$ versus $t$. The labels (a) — (f) denote the same ensembles as in Figures 8 and 9.

curves (c) and (f) of Figure 10. After a short initial period of rapid growth, they flatten out quickly so that by $t \approx 100$ they are both roughly constant. It is interesting to note, however, that the average number of walls in the two "particles" differ. The values of $\xi_{\mathcal{L}}(t)$ for the two plumes, averaged over times $100 \leq t \leq 200$ were $\overline{\xi_{\Lambda^{1,0}}} = 2.26 \pm 0.05$ for curve (c) and $\overline{\xi_{\Lambda^{1,1}}} = 2.67 \pm 0.02$ for curve (f).

In addition, the curves in Figure 10 appear to fall into three classes: curves (a), (b) and (e) all have similar shape, as do curves (f) and (c). The third class is exemplified by curve (d). Note that it was ensemble (d) whose plume showed interesting spreading properties (Figure 9(d)). Least-squares fits to exponential, power-law and polynomial functional forms indicated that curves (a), (b) and (e) showed power-law growth $\xi \propto t^{\alpha}$ with exponents of $\alpha = 0.36$ for both (a) and (e), and $\alpha = 0.30$ for (b). Curve (d) was best fit by a quadratic polynomial $\xi = a + bt + ct^2$ with $a = 2.08$, $b = 0.035$ and $c = -4.7 \times 10^{-5}$.

There is clearly room for further investigation into the behavior of domain interfaces. We have only touched on the phenomenology, without even attempting to trace the causal mechanisms responsible for the different behaviors. Nevertheless, we want to emphasize that without the simplification accomplished by identifying the domains, we would never have even gotten this far. Any analysis of domain interfaces—for example, tracing the behavior back to the CA rule table, or deriving statistical models of interface evolution—must necessarily be based on the pattern basis the domains represent.

## 5 Discussion and Conclusion

The decomposition of turbulent states into coherent structures and background has been a major and recent breakthrough in the general understanding of nonlinear spatially-extended systems. It has led to greatly improved statistical predictions for two-dimensional turbulence at large Reynolds number[4] and provided a means of studying the onset of vortex turbulence in the two-dimensional Ginzburg-Landau equation.[5] The key element behind this progress is the identification of coherent structures and the development of an appropriate analytical framework adapted to those structures. In both of the above examples, however, the relevant coherent structures were known *a priori* to be vortices. General applicability requires precise, quantitative definitions of both "pattern" and "coherent structure", as well as a battery of analytical and numerical tools for discovering, visualizing, and quantifying them in arbitrary space-time data. In turbulent systems especially, structural features may be obscured by the chaotic background, making automatic discovery of domains essential to identifying whatever coherent structures may be present.

As we have seen, regular domains, or more generally, formal languages, provide the necessary notion of "pattern" appropriate for cellular automata. Because domains are reconstructed from the observed space-time data, it is not necessary to start by assuming a pattern basis: rather, the appropriate pattern basis for the system is automatically discovered from the behavior of the system itself. Regular domains represent the fundamental spatio-temporal structures that govern the system's evolution, and into which the system self-organizes. They can be fully ordered, fully disordered, or anywhere in between. They are the pattern bases underlying a wide class of CA behavior. Coherent structures such as dislocations, domain walls, grain boundaries, and particles are defined as space-time loci at which the underlying domain structure breaks down. It is the background—i.e., the domains—that dominates the system's evolution and determines the properties of the coherent structures. Any analysis of the spatial structure of CA needs to take this into account.

The contrast in the values of quantities such as entropy and complexity density as measured over the domains and over the full state space demonstrates the importance of domains in determining the statistical properties of the system. The global averages are nonstationary and highly misleading, especially for systems with more than one domain, as in the example we analyzed. Even if the global average is restricted to long-time "asymptotic" behavior, the presence of multiple attractors means that the averaging is done over configurations with qualitatively different structural properties. Indeed, it is unclear that globally averaged statistics have any meaning at all in such a situation. Certainly, their interpretation as characterizing the system's typical behavior is dubious.

The domains also play an essential role in the structure of information processing and transmission in the system. This is evident from the difference patterns and plumes, which measure the space-time flow of information by tracing the propagating effect of a single-cell perturbation. The difference patterns within each domain are highly structured and independent of the detailed spatial configuration, while the average information flow over the full state space appears to be structureless, being essentially governed by the diffusive evolution of domain walls.

Domain-filtering gives a greatly simplified picture of the system's behavior, which would not have been otherwise accessible. Once the domains have been identified, they can be automatically filtered out, so that only the walls are visible. The domain walls can then be analyzed on their own terms, for example by modeling them as interacting particles. Generally this leads to a hierarchical understanding of the system's behavior, which at each level identifies structural features still hidden on the preceding level. In this article we started the analysis at the first level—identifying the domains—and indicated the beginnings of the second in the plots of the domain interfaces showing "particle" creation and annihilation. For a more detailed example of this type of analysis applied to a simpler system, see Ref. [11].

We have chosen to work on these issues in the context of cellular automata, since they have extremely simple local dynamics and are as discrete as possible, yet they still exhibit the same wide range of behavior—from simply ordered to complex to fully turbulent—found in nature. A well-developed mathematical formalism appropriate to CA is found in symbolic dynamics and the theory of computation. In addition, the discreteness and simplicity of the local dynamics means that CA are ideally suited to fast computer simulation. These features make CA convenient model systems with which to develop generally applicable ideas.

Many of the techniques we have described generalize to other systems, for example the symbolic dynamics of map lattices obtained via a generating partition on the real-valued spatial sites. Presumably they generalize further to continuous-time and continuous space-time dynamical systems. The connection with experiment is also close, since real data are inevitably quantized by experimental resolution anyway. In some sense, treating experimental data as discrete symbols builds *fewer*, rather than more, assumptions into the analysis than does treating it as a continuous signal.

Finally, we note a few of the connections of these techniques to other fields. Domain filtering represents a useful step in image processing and the visualization of high-dimensional systems, since the parts of the image that are "understood" are filtered out, clarifying the remainder for further analysis. It is also a method of image compression in which only the trajectories of the domain walls are recorded. The close connection of domains with computation theory also makes this work relevant to the problem of parallel (i.e. spatial) computation. Because these techniques explicitly discover the emergent computational structures arising in a spatial system's evolution, they provide a means of defining and investigating the intrinsic computational properties of the system. They provide a way of mapping out the architecture of information flow and processing of a given system. This is a necessary starting point for designing spatial systems to perform specific computational tasks.

## 6 Acknowledgments

This work was partially funded by ONR through grant N00014-92-J-4024 and administered by the Santa Fe Institute. The work at UC Berkeley was partially supported by grant AFOSR-91–0293.

# Appendix A  CA Invariant Set Design (with Dan Upper)

This appendix summarizes the method used to come up with radius 2 CA candidates containing two regular domains. The design approach is to initially write down the output portion of the rule table as a series of wild-card symbols, and to gradually replace the wild-cards with specific values according to constraints imposed by the domains. At the end of the process we are left with a rule table which may contain leftover wild-cards, and which then specifies a set of rules consistent with the constrained values.

The specific series of steps in the design process is detailed below. At each step, a choice is made about the domain structures, which has the effect of constraining a few more symbols in the rule table. It is often the case that a choice turns out to be inconsistent with previous choices because it fixes one of the symbols that has already been fixed at a different value. In such an event, it is necessary to back up and try again, even to the extent of starting over completely with different domains. At this point, there does not exist an algorithm guaranteed to find a CA rule possessing any two given domains. We must also emphasize that this design technique does not necessarily come up with rules for which the desired languages are mapped *onto* themselves by the dynamic—i.e., invariant, and hence domains—but rather with rules for which the languages are mapped *into* themselves by the dynamic.

The series of steps in the design is as follows. As noted, after each step, the self-consistency of the rule table is tested, and if there is a conflict, a different choice is made.

1. Choose the radius $r$ and the alphabet $\mathcal{A}$, to fix the number of entries in the rule table. In our case, we chose $r = 2$, $\mathcal{A} = \{0, 1\}$.
2. Choose a domain language, of the form $(xyz...)^*$, where each of $\{x, y, z, ...\}$ is either a symbol $\sigma \in \mathcal{A}$ or a wild-card $\Sigma$. Thus all domains designed by this method are of the form "periodic tiling with wild-cards". Our final choices, after a number of dead-ends, were the two patterns $(0\Sigma)^*$ and $(110\Sigma)^*$.
3. Choose the relative phase between a domain and its iterate. We ended up with a relative phase of $1$ between $(0\Sigma)^*$ and its iterate, and of $2$ between $(110\Sigma)^*$ and its iterate.
4. Write down pairs of parent/child words of sufficient size that the child word is more than one period long. This is done for our case as follows:

$$\begin{array}{lll} 0\Sigma0\Sigma0\Sigma0\Sigma & 110\Sigma110\Sigma & \sigma_t^i \; ... \, \sigma_t^{i+7} \\ \Sigma0\Sigma0\Sigma0\Sigma0 & 0\Sigma110\Sigma11 & \sigma_{t+1}^i \cdots \sigma_{t+1}^{i+7} \end{array} \qquad (27)$$

   Both parent and child words still contain wild-cards. This step fixes the bits in the output portion of the rule table that do *not* contain wild-cards. For example, the left parent/child pair of Eq. 27 forces the outputs of all parent neighborhoods of form $\Sigma0\Sigma0\Sigma$ to be $0$.
5. Fill in the wild-card cells in the parent/child pairs. This fixes all or most of the remaining wild-cards bits in the output portion of the rule table. In our case, the left parent/child pair in Eq. 27 indicates that parent neighborhoods of form $0\Sigma0\Sigma0$ may have outputs of either $0$ or $1$. The outputs of some of these parent neighborhoods may have already been fixed by other constraints such as those found in step 4. But others are still unconstrained.
6. If there are any remaining wild-cards in the rule-table, conduct a search of the rules generated by fixing them in all possible ways.

The rule studied in this article was found by following the above procedure, which provided us with a rule table containing $11$ leftover wild-cards, or in other words a set of $2^{11}$ candidate rules. Each rule in this set was designed to have $\mathbf{\Phi}(\Lambda^0) \subseteq \Lambda^0$ and $\mathbf{\Phi}(\Lambda^1) \subseteq \Lambda^1$. We then performed a Monte Carlo search for a rule for which both languages were invariant, i.e., $\mathbf{\Phi}(\Lambda^0) = \Lambda^0$ and $\mathbf{\Phi}(\Lambda^1) = \Lambda^1$, and for which both domains were stable. The rule that we found was CA 2614700074.

## Appendix B Filter Construction

This appendix describes a procedure for constructing a filter that will classify an arbitrary CA state $\mathrm{s}_t \in \mathcal{A}^*$ into a given set of domains $\{\Lambda^i\}$ and the walls $\Gamma^{ij}$ between them. The construction procedure guarantees that the resulting transducer is minimal—i.e., that there is no transducer having fewer states that implements the same filtration task. The procedure can be fully automated.

A wall is defined as follows. We consider a finite string $\omega = z\sigma$ such that $z \in \Lambda^i$ for some unique $i$, $\sigma \in \mathcal{A}$, $\omega \notin \Lambda^i$ for any $i$. Then $\sigma$ is a wall. Further, we break $z$ into two parts $z = xy$, with $y$ a string such that $y\sigma \in \Lambda^j$ for some unique $j$. Then $\sigma$ is the wall $\Gamma^{ij}$.

Construction starts with the process graphs of the domains. There are three distinct steps in the construction procedure, related to three steps in the operation of the filter: synchronization, encountering a wall, and re-synchronization.

Before the transducer reads the first cell in the spatial array, it is in the start state representing total ignorance of the context. It has not yet read enough cells to identify which of the domains the cell is in, or if it is a wall. Therefore, the transducer must pass through a series of "synchronization" states, during which it reads symbols and branches on their values, but does not emit anything. These states are automatically generated from the set of domain process graphs by considering the entire set as a single non-deterministic finite automaton (NFA)—nondeterministic because all states are start states—and performing the standard NFA-to-DFA conversion on it. This results in a single machine with the various domain machines as recurrent components and a branching tree of transient synchronization states leading down into them. The transducer begins in the unique start state, parses its way down through the tree of synchronization states, and eventually passes into some state in one or another of the original domain process graphs, at which time it is synchronized and begins emitting the symbol $\sigma_{out} = i$ for the domain $\Lambda^i$ it is in.

If the machine is in one of the recurrent states, and if a symbol is read for which there is no corresponding edge, this constitutes the observation of a domain wall. It is a symbol that is not consistent with the domain. To deal with domain walls, it is necessary to draw an extra edge, to be labeled with a symbol identifying the two domains it separates, for every disallowed transition in the original machine. The domain transducer is thereby constructed to accept as inputs all words in $\mathcal{A}^*$, i.e., all possible spatial configurations.

After encountering a wall and beginning to draw the extra edge, the immediate question is, where should the edge lead to? The naive answer is to reset to the start state, since the domain-structure of the input word has broken down. This is in fact not optimal, since there is more information at hand: specifically, the state the transducer was in when the wall was encountered, and the value of the wall cell. To take advantage of this information we apply the following scheme.

1. Let $V_j$ be the state of the transducer $\mathrm{T}_\Lambda$ upon encountering the wall, and let $a$ be the symbol read.
2. Consider the set $\mathcal{P}_j$ of words corresponding to paths in $\mathrm{T}_\Lambda$ that end in state $V_j$. Form the set $\mathcal{W}_j = \{\omega_j a\}$ by concatenating each word $\omega_j \in \mathcal{P}_j$ with symbol $a$.
3. Define the set $\mathrm{suff}(\mathcal{W}_j)$ of *legal* suffixes of words in $\mathcal{W}_j$, i.e., the suffixes that are in any domain language:
$$\mathrm{suff}(\mathcal{W}_j) = \{\mathbf{s} : \mathbf{rs} \in \mathcal{W}_j \ , \ \mathbf{s} \in \Lambda \text{ for some } \Lambda \in \{\Lambda^i\}\} \qquad (28)$$
4. Determine the set $\{U_j\}$ of recurrent (domain) states reached by all words in $\mathrm{suff}(\mathcal{W}_j)$. If $\{U_j\}$ has a single member, the dislocation edge goes to that state. If it has more than one member, state $V_j$ is split into parallel states, one for each member of $\mathrm{suff}(\mathcal{W}_j)$. This in general does not introduce indeterminism into the transducer, since the size of $\mathrm{suff}(\mathcal{W}_j)$ is related to the number of distinct paths in the transducer converging on $V_j$. The splitting refines the equivalence class that $V_j$ represents by separating paths leading into $V_j$ that were previously identified.
5. Each newly drawn edge is classified according to the domains $\Lambda^i$ and $\Lambda^j$ of its source and destination states. For a set of $m$ domains, the output label of an edge representing a wall $\Gamma^{ij}$ between domains $\Lambda^i$ and $\Lambda^j$ is $\sigma_{out} = (m + mi + j)$.

## Bibliography

[1] P. Manneville. *Dissipative Structures and Weak Turbulence*. Academic Press, New York, 1990.

[2] E. Bodenschatz, W. Pesch, and L. Kramer. Structure and dynamics of dislocations in anisotropic pattern-forming systems. *Physica D*, 32:135, 1988.

[3] F. Heslot, B. Castaing, and A. Libchaber. Transitions to turbulence in helium gas. *Phys. Rev. A*, 36:5870, 1987.

[4] G. F. Carnevale, J. C. McWilliams, Y. Pomeau, J. B. Weiss, and W. R. Young. Rates, pathways, and end states of nonlinear evolution in decaying two-dimensional turbulence: Scaling theory versus selective decay. *Phys. Fluids A*, 4:1314, 1992.

[5] G. Huber, P. Alstrom, and T. Bohr. Nucleation and transients at the onset of vortex turbulence. *Phys. Rev. Lett.*, 69:2380, 1992.

[6] N. Boccara, J. Nasser, and M. Roger. Particle-like structures and their interactions in spatio-temporal patterns generated by one-dimensional deterministic cellular automaton rules. *Phys. Rev. A*, 44:866, 1991.

[7] D. A. Lind. Appendix, table 15: Structures in rule 110. In S. Wolfram, editor, *Theory and Applications of Cellular Automata*. World Scientific, Singapore, 1986.

[8] J. E. Hanson and J. P. Crutchfield. The attractor-basin portrait of a cellular automaton. *J. Stat. Phys.*, 66:1415, 1992.

[9] J. P. Crutchfield and K. Kaneko. Phenomenology of spatio-temporal chaos. In Hao Bai-lin, editor, *Directions in Chaos*, page 272. World Scientific Publishers, Singapore, 1987.

[10]J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.

[11]J. P. Crutchfield and J. E. Hanson. Attractor vicinity decay for a cellular automaton. *CHAOS*, 3:215, 1993.

[12]S. Wolfram. *Theory and Applications of Cellular Automata*. World Scientific Publishers, Singapore, 1986.

[13]J. P. Crutchfield and K. Young. Inferring statistical complexity. *Phys. Rev. Lett.*, 63:105, 1989.

[14]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. Addison-Wesley, 1990.

[15]K. Young and J. P. Crutchfield. Fluctuation spectroscopy. In press, 1993.

[16]C. G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. In S. Forrest, editor, *Emergent Computation*, page 12. North-Holland, Amsterdam, 1990.

[17]L. P. Hurd. Appendix, table 10: Regular language complexities. In S. Wolfram, editor, *Theory and Applications of Cellular Automata*. World Scientific, Singapore, 1986.

[18]J. P. Crutchfield and N. H. Packard. Symbolic dynamics of noisy chaos. *Physica*, 7D:201, 1983.

[19]S. Wolfram. Universality and complexity in cellular automata. *Physica*, 10D:1, 1984.

[20]N. H. Packard. Complexity of growing patterns in cellular automata. In J. Demongeot, E. Goles, and M. Tchuente, editors, *Dynamical Systems and Cellular Automata*. Academic Press, 1985.

[21]M. A. Shereshevsky. Lyapunov exponents for one-dimensional cellular automata. *Journal of Nonlinear Science*, 2:1, 1992.

[22]S. Wolfram. Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55:601, 1983.