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 μ_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 $\left\langle \theta_t^i \right\rangle$:

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

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

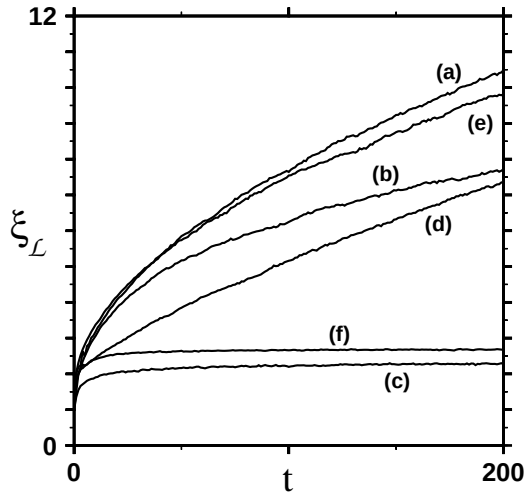

Figure 10 Interface plume volumes $\xi_{\mathcal{L}}(t)$ versus t. The labels (a) — (f) denote the same ensembles as in Figures 8 and 9.

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 Σ. 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:

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

   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 $\mathbf{\Phi}\left(\Lambda^0\right) \subseteq \Lambda^0$ and $\mathbf{\Phi}\left(\Lambda^1\right) \subseteq \Lambda^1$. We then performed a Monte Carlo search for a rule for which both languages were invariant, i.e., $\mathbf{\Phi}\left(\Lambda^0\right) = \Lambda^0$ and $\mathbf{\Phi}\left(\Lambda^1\right) = \Lambda^1$, 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 $\left\{\Lambda^i\right\}$ and the walls Γ^{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 σ 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 σ is the wall Γ^{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 Λ^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 T_Λ upon encountering the wall, and let a be the symbol read.
2. Consider the set $\mathcal{P}_j$ of words corresponding to paths in T_Λ 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:

$$\mathrm{suff}(\mathcal{W}_j) = \left\{\mathbf{s} : \ \mathbf{rs} \in \mathcal{W}_j \ , \ \mathbf{s} \in \Lambda \text{ for some } \Lambda \in \left\{\Lambda^i\right\}\right\} \tag{28}$$

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 Λ^i and Λ^j of its source and destination states. For a set of m domains, the output label of an edge representing a wall Γ^{ij} between domains Λ^i and Λ^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.