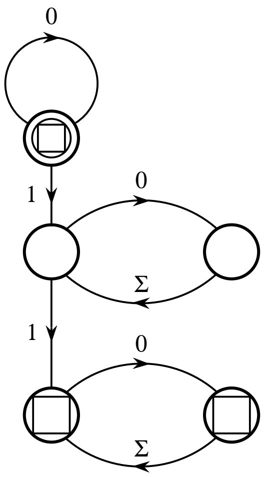

Fig. 10.27 Minimal DFA $\mathrm{M}\left(\Lambda_{18}^{0,0}\right)$ for domain-wall-domain language $\Lambda_{18}^{0,0}$.

spatial inhomogeneity of $\mathrm{M}\left(\Lambda_{18}^{0,0}\right)$ in combination with the spatial correlation of the input and output strings of the CA update transducer. The input to $\mathrm{T}_{18} \circ \mathrm{M}\left(\Lambda_{18}^{0,0}\right)$ consists of some portion of the spatial state s_{t-1}, starting an arbitrary distance from the dislocation. For input strings of the form $(01)^{*}10\Sigma0\Sigma...$, the observed region before the parent dislocation maps to a sequence of adjacent 0s. There is no nonzero cell to the left of the child dislocation to fix the spatial phase, and so the dislocation does not show up. As we discussed above, dislocations can only disappear in pairs, except when the state evolves to the null state. Thus the dislocation is still present, but merely unseen. We remove this "phase-ambiguity" artifact by requiring that for all non-null states the machine observe the dislocation before it can enter a final state. After the result is minimized, the final machine is seen to be identical to $\mathrm{M}\left(\Lambda_{18}^{0,0}\right)$, shown in fig. 10.27. Finally, $\Lambda_{18}^{0,0}$ is not a domain because it is not spatially homogeneous: its process graph has two strongly connected components. ▌

Next we consider the set of dislocation trajectories originating from a uniformly distributed ensemble of initial conditions belonging to $\Lambda_{18}^{0,0}$. We translate the states so that the dislocations lie initially at the same site. This initial distribution is formed by using $\mathrm{M}\left(\Lambda_{18}^{0,0}\right)$ as a pattern generator, taking uniform transition probabilities at branchings. The result obtained by evolving N_{total} members of this ensemble forward in time is the "dislocation plume" shown in fig. 10.29. There the gray scale darkness of a space-time cell (i,t) is proportional to the number $N_d(i,t)$ of dislocations visiting it. To improve contrast it has been normalized so that $\max\limits_{i}\{p_d(i)\} = 1$ at each time t, where $p_d = N_{\mathrm{total}}^{-1} N_d(i,t)$. Note the "checkerboard" pattern of the plume. This is due to the fact that in a single time step a dislocation can move in space by only an odd number of cells. Fig. 10.30 shows the plume histogram at times $t = 1,\ 10,$ and 100, in which the checkerboard pattern of the histogram has been factored out. The histogram's asymmetry, noticeable at the early time steps, is due to

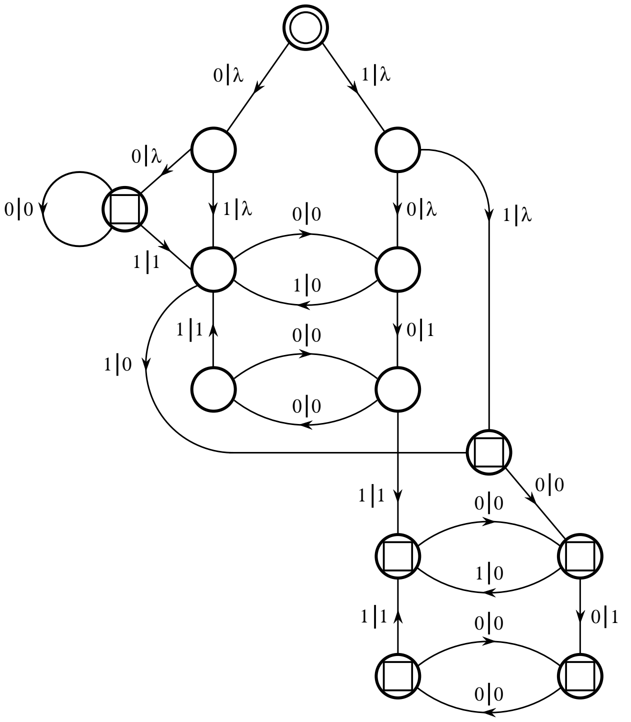

Fig. 10.28 $\Lambda_{18}^{0,0}$ is a fixed-point language: Machine $\mathrm{T}_{18} \circ \mathrm{M}(\Lambda_{18}^{0,0})$ obtained after composing the ECA 18 update transducer with the two-domain machine of fig. 10.27. On completion of the remaining steps of the proof, fig. 10.27 is found.

right-going dislocation detection. The relaxation from the early-time binomial process to the Gaussian is clearly demonstrated.

The plume's width $\sigma(t) \equiv \sqrt{\langle x^2 \rangle}$, plotted against time in fig. 10.31, quickly approaches the time dependence of a random walk. As the straight line in fig. 10.31 indicates we find $\sigma(t) \approx (2Dt)^{-\alpha}$ with a least-squares fit for $t > 512$ giving $D = .506$ and $\alpha = 0.501 \pm 0.004$. The systematic deviation from Gaussian diffusion seen at early times is again due to the recognition asymmetry which gives $2^{-\frac{i}{2}}$ tails to the distribution and to the binomial structure of the dislocation movement. The result of the exact analysis of these effects is that, for example, $\log \sigma(1) = \frac{3}{2}$. The simulations give $\log \sigma(1) = 1.52 \pm 0.05$.

Computation with dislocations

We have seen that space-time patterns generated by ECA 18 are composed of domains Λ_{18}^{0} separated by dislocations. The computational structure of a domain is extremely simple, since $\mathrm{M}(\Lambda_{18}^{0})$ possesses only two recurrent states, and is thus capable of storing only 1 bit of information. This is clearly too simple to be harnessed for useful computation—by which is meant such practical tasks as, to pick a random example, computing the digits of π. If it exists, the useful computation must be performed by the dislocations at the domain boundaries.

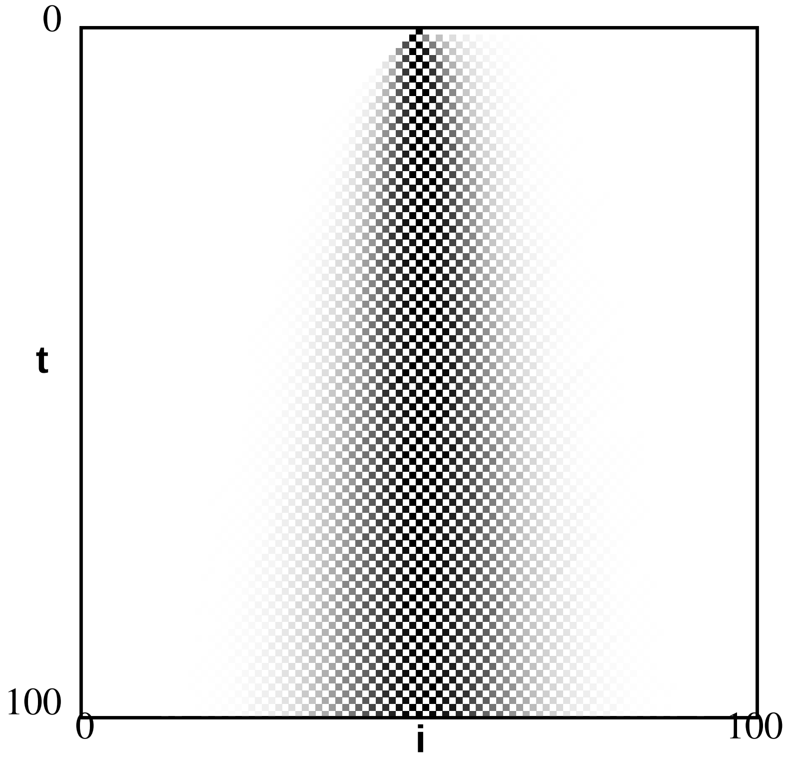

Fig. 10.29 The dislocation plume: Evolution of the initial ensemble $\Lambda_{18}^{0,0}$ consisting of configurations with a single dislocation. The gray scale intensity of a space-time cell is proportional to the number of dislocations visiting there. It is normalized at each time step to run from white to black so the distribution tails are more apparent.

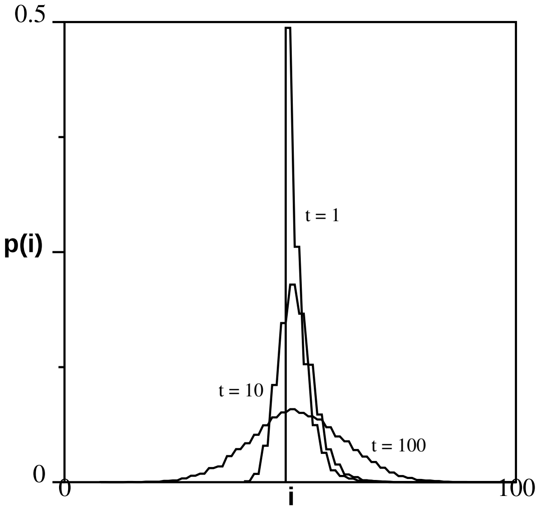

Fig. 10.30 Dislocation histogram at times $t = 1, 10,$ and 100.

We can make a quantitative estimate of the computational capability of ECA 18 dislocations in the following way. After detecting dislocations with the domain transducer $\mathrm{T}_{\Lambda_{18}^{0}}$, we recode the motion of a single trajectory as a symbol stream by assigning the symbol L to a move to the left and R to a move right. We then apply the machine reconstruction technique to this R-L data stream. The resulting machine,

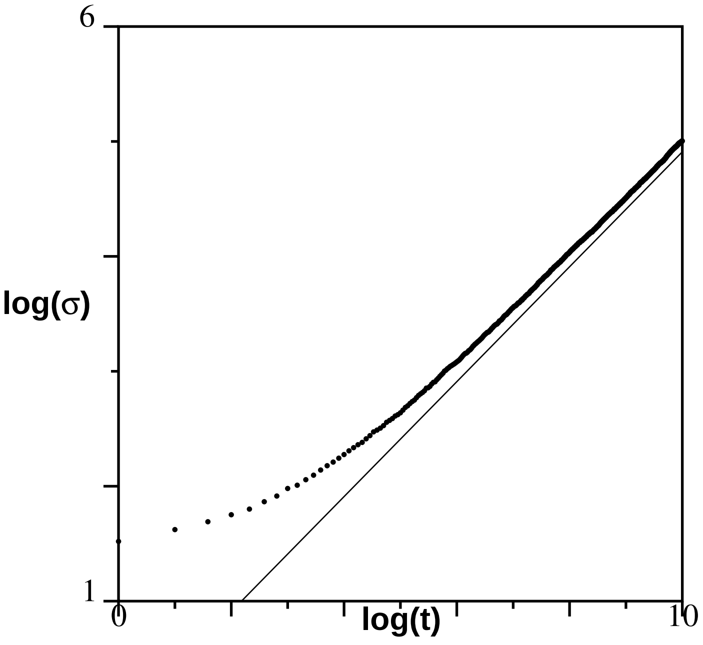

Fig. 10.31 Standard deviation of the dislocation histogram as a function of time. The measured slope of the asymptotic line is 0.501 ± 0.004 for $t > 100$. Logarithms are base 2. The straight line shown has slope $\frac{1}{2}$.

displayed in fig. 10.32, shows an asymmetric structure: every R must be followed by an L. Noting that the topological entropy of this machine is $\log_2 \phi$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden mean, we call it the golden mean machine M_{gm}. The asymmetry in M_{gm} is due to the asymmetry of the domain transducer $\mathrm{T}_{\Lambda^0_{18}}$ which identifies dislocations by a misplaced 1 in the spatial data set. This has the effect of a constraint on the motion of the observed dislocation: all moves to the left are a single cell, but moves to the right may be any odd-valued distance. Further, a move left may be followed by a move either to right or left, but a move right must be followed by one to the left. As we noted above, other dislocation-recognizing algorithms can be used for which this constraint would not be present. Apart from this, however, we see that all trajectories consistent with this single constraint are allowed. Numerical estimates indicate a uniform branching probability from M_{gm}'s start state. Thus, one type of computation of which ECA 18 is capable is the map from $\Lambda^{0,0}_{18}$ to $\mathcal{L}(\mathrm{M}_{\mathrm{gm}})$. This is a very restricted class of computation since it is a mapping between such simple regular languages.

The computational properties of ECA 18 are captured by a model with two components: (i) computationally simple invariant domains Λ^0_{18}, and (ii) dislocations evolving along computationally simple trajectories given by M_{gm}. Neither of these components has anything like the complexity of the global machine $\mathrm{M}(\Omega_t)$, which grows extremely rapidly with time (see section 12.4 below). The global machine is misleading in that it might be seen as indicating that ECA 18 is very computationally capable, when for the *typical* evolution, just the opposite is true.

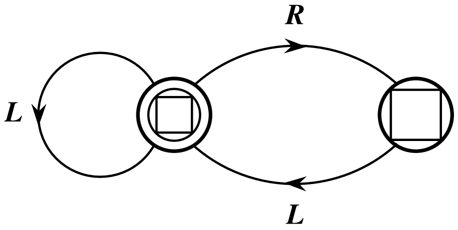

Fig. 10.32 The golden mean machine M_{gm} reconstructed from the space-time trajectory of a single dislocation. The symbol L denotes a dislocation move to the left; R a move right. Its entropy is $\log_2 \phi$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden mean.

Wolfram's early investigation of CA computational capability, described in section 2.2, was based on visual examination of the space-time patterns they generate. Recall that class 4, conjectured to be the most computationally capable, was defined by the presence of spatially isolated transients of many shapes evolving for arbitrarily long times. The conjecture was that using these propagating "blobs" a model of universal serial computation could be constructed.

A simple recoding of ECA 18, however, produces space-time patterns satisfying the visual criteria of class 4. This recoding, elsewhere called the "blocking transformation",[107] is a special type of a decimation transducer. Recall that in the blocking transformation every other cell, for example, in space and time is excised. The new pattern represents the evolution of the recoded initial condition under a different CA rule, typically one with a larger neighborhood size. General blocking transformations replace a fixed length block of cells with a single symbol. Blocking transformations, though, are neither as versatile as transducers nor do they exploit a CA's qualitative dynamics.

Visually, blocked space-time patterns of arbitrary states evolving under ECA 18 display spatially isolated patches of fluctuating cells moving against an invariant background of 0s (see appendix 3 in ref. [107]). It is the spatial period 2 nature of Λ^0_{18}, however, that is responsible for this. When a domain is in phase with the blocking scheme's internal state, arbitrary spatial patterns will be observed. When it is out of phase, all the nonzero sites will be removed. The resulting blocked region is a homogeneous patch of 0s. Thus, the structures in the blocked pattern are caused by the wall dynamics. They are misinterpreted by the blocking transformation since the domains alternate between being in-phase and out-of-phase with the blocking. The evolution of the boundaries of an in-phase domain governs the evolution of the blocked patch between them. When the boundaries meet, they annihilate and the "spatially isolated transient" disappears.

Based on the appearance of propagating blobs under a simple recoding, one might be tempted to argue that ECA 18 should be placed in class 4. However, its computational capability appears quite limited, being dominated by diffusive behavior of the dislocations. The simplicity of M_{gm} makes it doubtful that ECA 18 can be harnessed to perform useful computations. The misleading "transients" in the blocked patterns are an artifact introduced by the blocking transformation. They occur because the transducer used in the blocking was unsuited to the underlying domain structure.

The ambiguities of Wolfram‘s classification suggest a reevaluation of how computation is identified in CA. The use of transducers in concert with machine reconstruction gives a principled approach to analyzing a CA’s intrinsic computational capability. The first step employs machine reconstruction to search for computational structures such as domains. When found, these are factored out using the domain transducers. The resulting encoded pattern then can be searched for higher-order structures, again using machine reconstruction. In this way, one builds a hierarchy of computational models, with structures in each level describing computational properties missed at the previous level. The hierarchy as a whole describes a CA’s information processing architecture. The hierarchy’s depth in this view is one simple measure of CA’s global computational capability.

For intrinsic computation to be useful, there are the additional requirements that individual computations be robust and that a compiler exist for a class of nontrivial tasks. The first means that, at the very least, small perturbations do not change the computational class of the temporal evolution. This in turn requires an understanding of the global organization the CA induces in regular language space; that is, of the attractor-basin portrait. The second requirement, a compiler, determines how a computational task is mapped onto the CA’s dynamics. Nontrivial tasks can be supported either by the CA’s evolution toward attractors or by propagating wall structures. A dislocation compiler for ECA 18, for example, is a mapping of a computational task specification T to an initial condition $\mathrm{s}_0 \in \Lambda_{18}^{0,0}$ such that the time series $\mathrm{s}_0, \mathrm{s}_1, \mathrm{s}_2, \ldots, \mathrm{s}_t, \ldots$ executes the task via the $R - L$ movement of wall and domain structures. The output is a string in $\mathcal{L}(\mathrm{M}_{\mathrm{gm}})$. Since ECA 18’s computation appears quite limited, we must leave the question of useful intrinsic computation to a future analysis of more complex CA.

Although formulated in the present context of spatially-extended systems, the general questions raised concerning the design and existence of compilers for CA are germane to parallel programming and distributed processing. Should such compilers exist, then analyses along the lines developed here would indicate where useful computation can lie in high-dimensional state space. They also would give a global view of the geometry underlying the computation. One practical use of this appears to be in the analysis of errors and stability of a given CA implementation.

11 Multiple Coexisting Domains

In this chapter 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. As such, this analysis is intended to serve as a model for the analysis of general CA.*

Having reconstructively identified the domains from the space-time plot of the CA's evolution, we construct the domain transducer 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.

This particular CA was selected because it has two domains, both high entropy-density "turbulent" processes, but with different internal structure. Nevertheless, our techniques enable us to identify, separate and analyze them. This reveals more structural information than would otherwise be apparent, especially when compared to what is afforded by the global view.

11.1 Binary Range–2 CA 2614700074

A space-time plot of the CA we are concerned with, binary range–two ($k = 2$, $r = 2$) rule 2614700074, is shown in fig. 11.1. The initial condition is random, and the lattice size is $N = 200$ cells with periodic boundary conditions.

A similar sequence of spatial configurations was fed as input into the machine reconstruction algorithm. 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 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

---

*This investigation originally appeared in ref. [25].

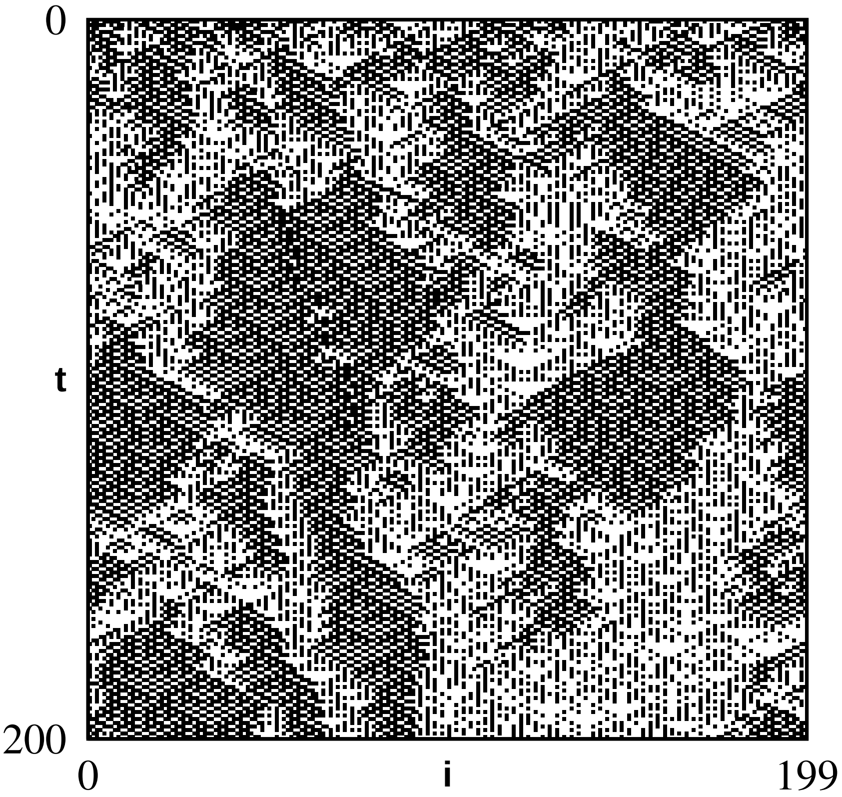

Fig. 11.1 Space-time plot of the evolution of binary range-2 CA 2614700074, starting from a random initial condition on a 200 site lattice, over 200 time steps.

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 region in turn and applying machine reconstruction to them, we found that the first region consists of a tiling of the spatial pattern (0Σ), and the second consists of a tiling of (110Σ). The symbol $\Sigma = \{0, 1\}$ 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. The two regions are therefore identified with the regular process languages $\Lambda^0 \equiv \mathrm{sub}(0\Sigma)^*$ and $\Lambda^1 \equiv \mathrm{sub}(110\Sigma)^*$, where $\mathrm{sub}(\mathcal{L})$ indicates the set of all subwords of $\mathcal{L}$. As we show in the next section, Λ^0 and Λ^1 are in fact regular domains.

The process graphs $\mathrm{M}(\Lambda^0)$ and $\mathrm{M}(\Lambda^1)$ for the domain languages Λ^0 and Λ^1 are shown in figs. 11.2(a) and (b), respectively. An edge labeled with the symbol Σ is shorthand for a pair of parallel edges with symbols 0 and 1. The indices inscribed in the states of $\mathrm{M}(\Lambda^j)$ are the values of the domain's phase ϕ^j. By convention, we have chosen the $\phi^j = 0$ state of each domain Λ^j to be the state after the edge labeled with the wild-card Σ. Note that Λ^0 is identical to the domain Λ^0_{18} of ECA 18, although the temporal behavior of individual states in Λ^0 differs under the two CA rules, as can be seen by comparison of figs. 11.1 and 10.17.

Using the general procedure outlined in section 7.5, we constructed a domain filter T_Λ that maps arbitrary spatial configurations $\mathrm{s} \in \mathcal{A}^*$ onto the sequence of domains

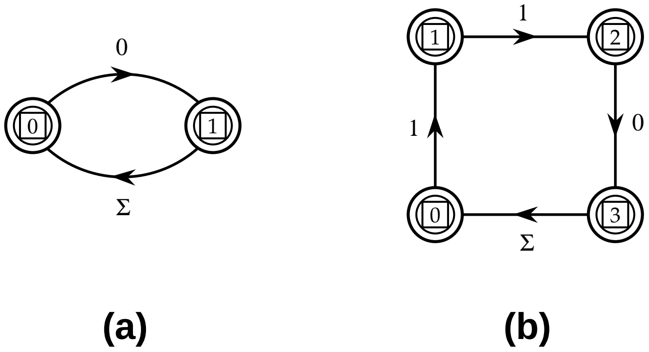

Fig. 11.2 Process graphs for the two domains of binary range 2 CA 2614700074: (a) Λ^0; (b) Λ^1.

and walls the string contains. Each cell σ^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 cell $\sigma = \Gamma^{ij}$ between domains Λ^i and Λ^j is mapped to symbol $(2i + j + 2)$. At each time t, T_Λ is run over the spatial configuration 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 fig. 11.3(top), we generated the plot shown in fig. 11.3(bottom). In the latter, cells in Λ^0 are represented as white squares, cells in Λ^1 are gray squares, and all walls are black. Comparison of figs. 11.3(top) and 11.3(bottom) 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 in the original space-time data. Dislocations, i.e., 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 spatial configuration is seen to be the concatenation of subpatterns from the two domains.

We traced the evolution of the initial condition shown in fig. 11.3 beyond time $t = 100$. Both types of domains were present up to $t = 700$, at which time the state as a whole fell into domain Λ^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 Λ^0. This is a preliminary indication that Λ^0 and Λ^1 may be fixed-point regular attractors, with Λ^0 having the larger basin measure. In any case, it is abundantly clear that the Λ^0 and Λ^1 are the fundamental structures organizing the high-dimensional state space.

The domain filter T_Λ is shown in fig. 11.4. Three different types of state are shown. States without inscribed labels are synchronization states, devoted to processing the first few symbols in s. 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. Besides the synchronization states, there are two types of asymptotic state, one for each domain. States labeled Λ^0 or Λ^1 correspond

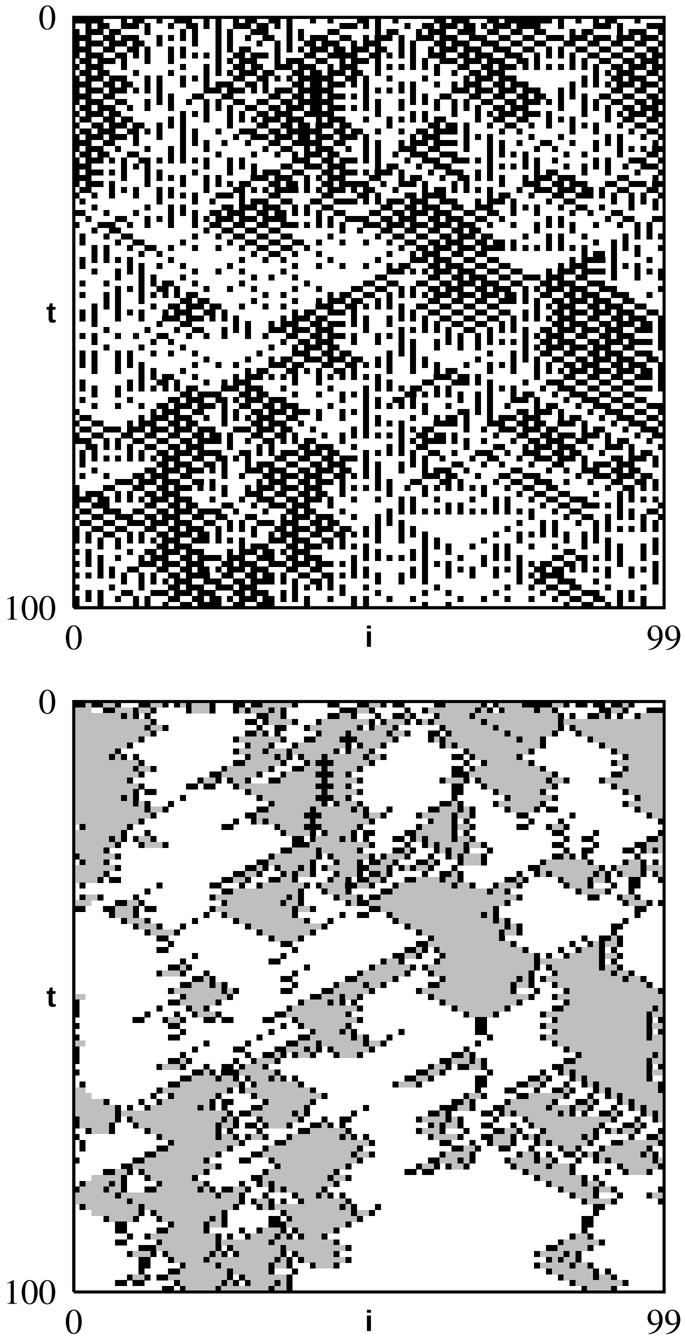

Fig. 11.3 (top) Space-time plot of the evolution of binary range-2 CA 2614700074, starting from a random initial condition on a 100 site lattice over 100 time steps. (bottom) Filtered space-time plot of the same data shown in (top). Space-time regions in domain Λ^0 are shown in white, regions in Λ^1 are gray, and all walls are shown in black. All four varieties of walls are present. This figure was generated using the transducer of fig. 11.4.

to states visited when the input cell is in domain Λ^0 or Λ^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 a sufficient amount of information to unambiguously distinguish among 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, respectively Γ^{00}, Γ^{01}, Γ^{10}, and Γ^{11}, as noted above.

The complete domain transducer shown in fig. 11.4 assigns a unique symbol to each type of wall; in fig. 11.3(bottom), however, we have plotted all four types of walls as black squares, effectively changing the transducer to output the same symbol on a wall of any type. Alternatively, we could have assigned a unique symbol to each of the nine wall transitions. These minor variations on the machine of fig. 11.4 are just different visualizations of the same fact, namely that the system organizes itself into spatial regions of domains Λ^0 and Λ^1, with walls in between.

In making fig. 11.3(bottom), the transducer traversed the lattice from left to right, using the periodic boundary conditions to wrap around at the end in order to classify the first few cells, i.e., the cells used for synchronization. Each cell in the lattice can be classified in this way with at most two passes over the data.

11.2 Domain Properties

Having reconstructed the machines for the two regular languages Λ^0 and Λ^1 from the observed space-time data, we must now prove that they are domains. This is done in the same way as in previous chapters.

Proposition: Λ^0 and Λ^1 are regular domains of binary range-2 CA 2614700074.

Proof: (i) Temporal invariance: $\mathbf{\Phi}(\Lambda^0) = \Lambda^0$ and $\mathbf{\Phi}(\Lambda^1) = \Lambda^1$ by explicit construction using the FME algorithm. (ii) Spatial homogeneity: by inspection of figs. 11.2(a) and 11.2(b). ▌

Domain quantifiers

This subsection describes the application of the various pattern quantifiers described in chapter 9 to the domains Λ^0 and Λ^1. The results, shown in the first two columns of table 11.1, are discussed in sequence below. The third column of the table gives estimated values for the quantifiers measured on the iterates Ω_t of the extended state space $\Omega_0 = \mathcal{A}^*$, which will be used for comparison with the values for the domains.

The first four rows of table 11.1 show the entropies and complexities for each domain. The interesting thing to note is that the two domains differ in both entropy and complexity: Λ^0 has higher entropy, and lower complexity, than Λ^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 Λ^0 and Λ^1 are cyclic loops, and because the single branching state has equal branch probabilities, the state probabilities of each machine are uniform, and so $C_0 = C_1$ as well.

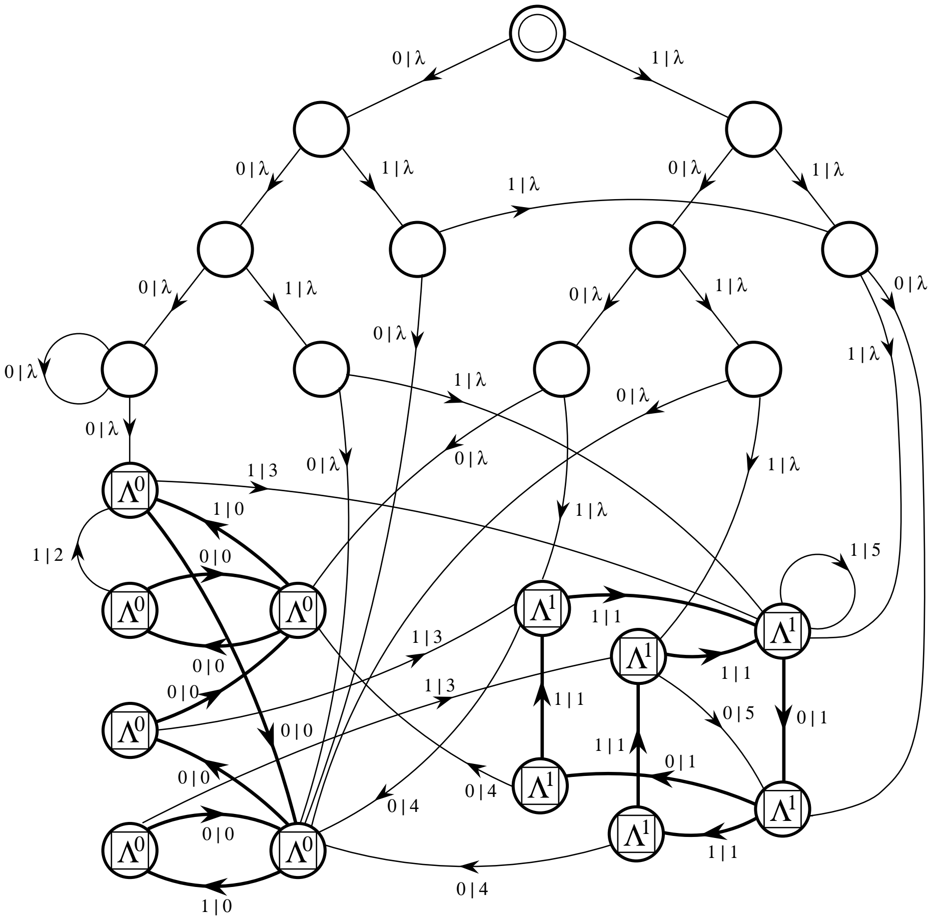

Fig. 11.4 Domain transducer. Cells in domain Λ^0 are mapped onto 0s, cells in domain Λ^1 are mapped onto 1s, and walls Γ^{ij} are mapped onto symbol $(2i + j + 2)$. For the first few symbols read, the transducer is synchronizing to the data stream, and no output is generated, as indicated by the null symbol λ. The edges shown in bold correspond to cells within a domain.

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 Λ^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 s_0 and s_{200}, and measured the metric entropy and statistical complexity at the two times. The comparison showed that for both Λ^0 and Λ^1, h_1 and C_1 are invariant.

The values of the single-site statistics ρ_0 and ρ_1 for both domains are shown in the fourth and fifth rows of table 11.1. For this rule, the "quiescent" cell is $\hat{\sigma} = 0$, so ρ_0 and ρ_1 measure the density of nonzero cells. The large difference in the values of $\rho_1(\Lambda^0)$ and $\rho_1(\Lambda^1)$ is what makes the two domains so easy to identify visually in fig. 11.1.

The final row in table 11.1 gives the values of the left and right "Lyapunov exponents" λ_L and λ_R, measured from the difference plumes discussed more fully in the next section.

Quantifier	Λ^0	Λ^1	Ω_t
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(†)
ρ_0	4/11	6/10	15/32
ρ_1	0.250	0.625	0.41(†)
λ_L, λ_R	-2, 2	-2, 2	-1.64, 1.73

Table 11.1 Quantifiers for the two domains Λ^0 and Λ^1, and for the iterates of the state space, $\Omega_t = \mathbf{\Phi}^t(\mathcal{A}^*)$. See chapter 9 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 ± 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 11.1 that gives estimates of values of the domain quantifiers for the iterates Ω_t of the extended state space $\Omega_0 = \mathcal{A}^*$. 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.,

$$\Omega_{t+1} \subset \Omega_t, \; \forall t \geq 0, \tag{11.1}$$

and generally to the creation and annihilation of domains, walls, particles, and other structures. For example, the topological complexity $C_0(\Omega_t)$ appears to grow without bound, as the number of states in the machines for the sequence of languages $\{\Omega_t, \; t = 0, 1, 2, ...\}$ rapidly diverges under the action of the FME operator. For the entropies, complexities, and single-site statistics in table 11.1, we estimated the values using a single state 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 $h_0(\Omega_t)$ and $h_1(\Omega_t)$, and the density of nonzero cells $\rho_1(\Omega_t)$ were estimated from the statistics of subsequences of s_t of length $L = 18$. The complexities $C_0(\Omega_t)$ and $C_1(\Omega_t)$ were approximated by the estimated topological and metric total excess entropies, again measured over subsequences of length $L = 18$.[26] The (topological) fraction of nonzero cells $\rho_0(\Omega_t)$, 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 $\rho_1(\Omega_t)$, for example, is misleading in the same way that the mean of a bimodal distribution