Discrete Dynamical Networks and their Attractor Basins

Andrew Wuensche
Santa Fe Institute, 1399 Hyde Park Road,
Santa Fe, New Mexico 87501 USA,
wuensch@santafe.edu, http://www.santafe.edu/∼wuensch/

Abstract

A key notion in the study of network dynamics is that state-space is connected into basins of attraction. Convergence in attractor basins correlates with order-complexity-chaos measures on space-time patterns. A network's "memory", its ability to categorize, is provided by the configuration of its separate basins, trees and sub-trees. Based on computer simulations using the software Discrete Dynamics Lab[19], this paper provides an overview of recent work describing some of the issues, methods, measures, results, applications and conjectures.

1 Introduction

Processes consisting of concurrent networks of interacting elements which affect each other's state over time are central to a wide range of natural and artificial systems drawn from many areas of science; from physics to biology to cognition; to social and economic organization; to computation and artificial life; to complex systems in general. The dynamics of these "decision making" networks depends on the connections and update logic for each element, resulting in complex feedback webs that are difficult to treat analytically. Understanding these systems depends on numerical simulations of idealized computer models known as discrete dynamical networks.

Cellular automata (CA) are a powerful yet simple class of network, characterized by a homogeneous rule and uniform nearest neighbour connections, providing models to study processes in physical systems such as reaction-diffusion[15], and self-organization by the emergence of coherent interacting structures[18]. By contrast, random Boolean networks (RBN) provide models for biological systems such as neural[3] and genetic[11] networks, where connections and rules must be less constrained. In addition, the idealized networks themselves hold intrinsic interest as mathematical/physical systems.

A key notion underlying network behavior is that state-space is organized into a number of basins of attraction, connecting states according to their transitions, and summing up the network's global dynamics, analogous to Poincaré's "phase portrait" which provided powerful insights in continuous dynamics.

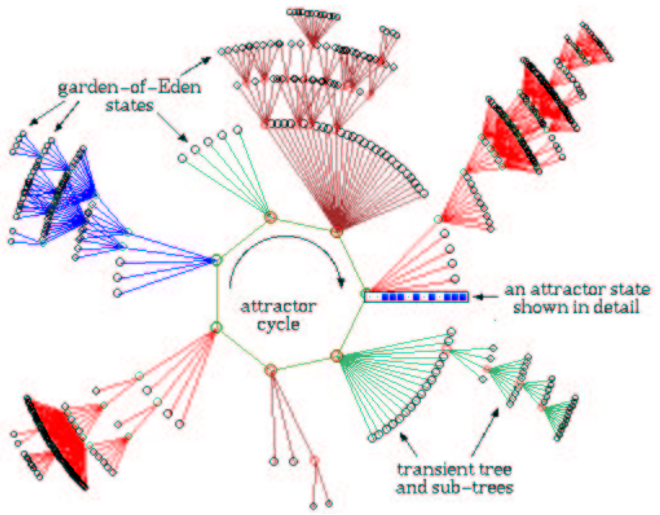

garden-of-Eden states
attractor cycle
an attractor state shown in detail
transient tree and sub-trees

Figure 1: A basin of attraction (one of 15) of a random Boolean network (n=13, k=3) shown in figure 15. The basin links 604 states, of which 523 are garden-of-Eden states. The attractor period = 7, and one of the attractor states is shown in detail as a bit pattern. The direction of time is inwards from garden-of-Eden states to the attractor, then clock-wise.

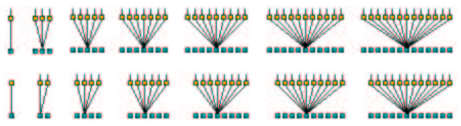
The quality of dynamical behaviour of CA, from ordered to chaotic[1], is reflected by convergence in attractor basins, their characteristic in-degree, which influences the length of transients and attractor cycles. The in-degree of a state is its number of pre-images (predecessors). Bushy subtrees with high in-degree imply high convergence and order. Sparsely branching subtrees imply low convergence and chaos. In the case of RBN, attractor basins reveal how the network is able to hierarchically categorizes state-space into separate basins, trees and sub-trees, the network's "memory". Changes to the network's wiring or rules change the memory categories, providing insights into learning[17, 20].

Traditionally, network dynamics has been investigated by running networks forward from many initial states to study space-time phenomenology[15], and for statistical measures on basins of attraction[8]. More recently, exact representations of basins of attraction and sub-trees have become accessible, where algorithms directly compute the pre-images of network states, allowing the network to be run "backwards" to disclose all possible historical paths[16, 17, 21]. Based on computer simulations using the software Discrete Dynamics Lab (DDLab)[19], this paper provides an overview of network architecture, the characteristics of space-time patterns, the methods and algorithms for reconstructing basins of attraction, and related parameters, measures, results, applications and conjectures, placing the dynamics along particular trajectories in the context of global dynamics.

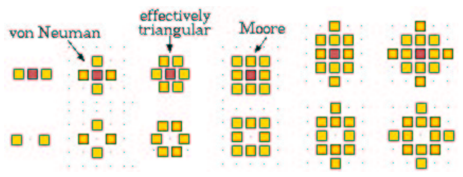
[1]The notion of "chaos" is used here by analogy only to its meaning in chaos theory, although there are many common properties, for example sensitivity to initial conditions.

2 Network Architecture

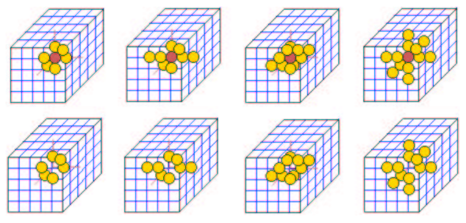
Discrete dynamical networks consist of a set of elements (cells) taking inputs from each other, and changing their cell-state according to some logical function on their inputs. The connectivity is usually sparse. The cell-state ranges over a discrete alphabet, in this paper just a binary alphabet (0 or 1) is considered. The updating is generally synchronous, though updating sequentially in a preset order or partial order is also of interest. A partial order is a sequence of sets of cells, where updating within each set is synchronous.



1d. k=0-13. The extra asymmetric cell in even k is on the right. The wiring is shown between two time-steps.



2d, k=2-13 (k=0-1 as in 1d). Note that k=6 and k=7 define an effectively triangular grid by changing between odd and even rows. The classical von Neuman and Moore neighbourhoods are indicated.

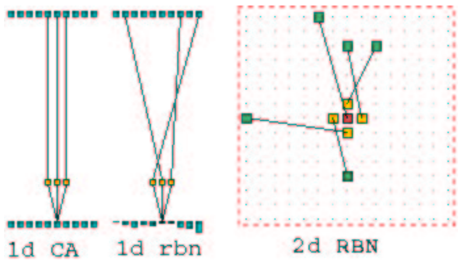


3d, k=6-13 (k=0-5 as in 2d), shown looking up into an axonometric cage.

Figure 2: 1d, 2d and 3d neighbourhood templates defined in DDLab. In 2d and 3d, to maximize symmetry, even k does not include the central target cell.

A CA is a very regular network, sometimes described as an artificial universe with its own physics. Cells take inputs from their nearest (and next nearest) neighbours (local "wiring") according to a fixed neighbourhood template, so issues of network geometry and boundary conditions are crucial. The same logical rule is applied everywhere. Figure 2 shows neighbourhood templates for 1d, 2d and 3d as applied in DDLab. An RBN relaxes these constraints, allowing arbitrary "wiring" and rules, as in figure 3. The number of input wires available to each cell may also vary. However, RBN architecture can be biased in countless ways, described in section 4, to constrain wiring or rules and approach CA, for example RBN wiring with a constant rule, or local wiring with mixed rules.

The wiring and rules can be tailored to very specific requirements, as in models of neural networks in the cortex[3]. The wiring can be constrained within a fixed distance from each cell, which confers meaning to network geometry and boundary conditions, whereas with completely arbitrary wiring the geometry just provides a convenient way of representing the network. A rule mix can be constrained to sample just a few rules or rules with a particular bias, as in genetic network models[5].



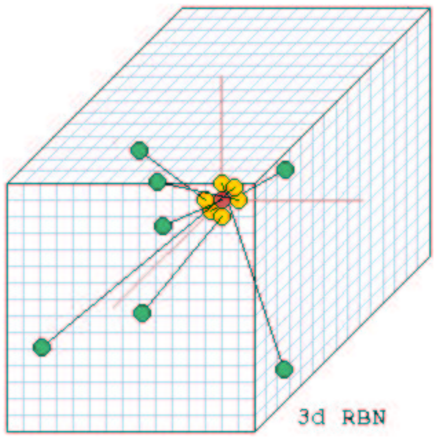


Figure 3: Network wiring. *left:* 1d, k=3, the wiring is shown between two time-steps. *centre:* 2d, k=5. *right:* 3d, k=7. In RBN, cells anywhere in the network are wired back to each position in the "pseudo-neighbourhood".

Hybrid networks can be constructed by putting an RBN within a CA or vice-versa. Networks of networks can be set up, with weak interactions so they perturb each other's dynamics. The functionality for setting up networks in these ways is present in DDLab.

The network parameters can be listed as follows:

size: The system size n, the number of cells in the network.

connectivity: The number of input wires per cell k, or the k-mix if the connectivity is not homogeneous. The connectivity is usually sparse, i.e. $k \ll n$.

neighbourhood: The neighbourhood template for CA, or the pseudo-neighbourhood for RBN, as in figure 2.

wiring: For RBN, how each cell is wired relative to its pseudo-neighbourhood.

rule: The rule for CA, or the rule scheme scheme for RBN. Rules are generally defined as look-up tables.

updating: The updating, usually synchronous. Alternatively sequential according to a defined order or partial-order.

geometry: The underlying geometry and boundary conditions, 1d, 2d, 3d, orthogonal or triangular, or some other geometry, for example a hypercube. This is essentially a function of the the neighbourhood template and wiring scheme. The geometry for graphically representing the network may not necessarily correspond to the underlying geometry.

A CA neighborhood, or RBN pseudo-neighborhood, of size k has 2^k permutations of values. The most general expression of the Boolean function or rule is a lookup table (the rule-table) with 2^k entries, giving 2^{2^k} possible rules. Sub-categories of rules can also be expressed as simple algorithms, concise AND/OR/NOT logical statements (which could be implemented as combinatorial circuits), totalistic rules[14] or threshold functions.

By convention[14] the rule table is arranged in descending order of the values of neighborhoods, and the resulting bit string converts to a decimal or hexadecimal rule number. For example the k=3 rule-table for rule 30,

7	6	5	4	3	2	1	0	...neighbourhoods, decimal
111	110	101	100	011	010	001	000	...neighbourhoods, binary
0	0	0	1	1	1	1	0	...outputs, the rule table

The rule-table for other k values are set out in a corresponding way. $k \geq 4$ rules are referred to by their hexadecimal rule numbers. $k \leq 3$ rules are usually referred to by their more familiar decimal rule numbers.

For a given geometry, the behaviour space of CA depends on the size of rule-space, 2^{2^k}, though rule symmetries effectively reduce this number. For example, the $2^{2^3} = 256$ rules in $k = 3$ rule-space reduce to 88 equivalence classes[16]. The behaviour space of RBN is much greater, taking into account possible permutations of wiring and rule schemes, but there are also RBN equivalence classes relating to these permutations[10]. In general, the number of effectively different RBN of size n cannot exceed $(2^n)^{(2^n)}$ (see section 9).

3 Trajectories and space-time patterns

A state of a discrete dynamical network is the pattern of 0s and 1s at a given time-step. A trajectory is the sequence of states at successive time-steps, the systems *local* dynamics. Examples of 1d, 2d and 3d space-time patterns are shown in figures 4, 5 and 6. A time axis is only possible in representations of 1d or 2d systems. As well as showing cells as white(0) or black(1), an alternative presentation shows cells in colors (or shades) according to their look-up neighbourhood (figure 4). This allows the most frequently occurring colors to be progressively filtered to show up gliders and other space-time structures as in figure 5, which can be done interactively, on-the-fly, in DDLab for any CA. This is an alternative method to the “computational mechanics” approach[4].

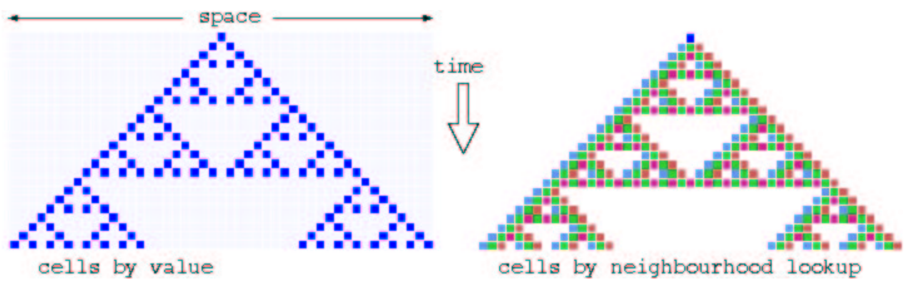


Figure 4: Space-time patterns of a CA (n=24, k = 3, rule 90). 24 time-steps from an initial state with a single central 1. Two alternative presentations are shown. *Left*: cells by value, white=0 black=1. *Right*: cells colored (or shaded) according to their look-up neighbourhood. This allows filtering, and improves the clarity of space-time patterns in 2d and 3d.

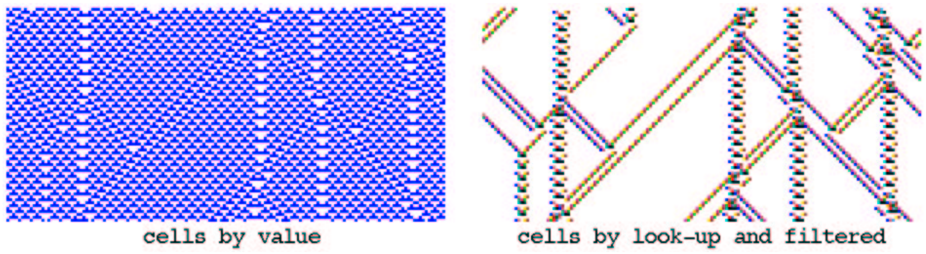
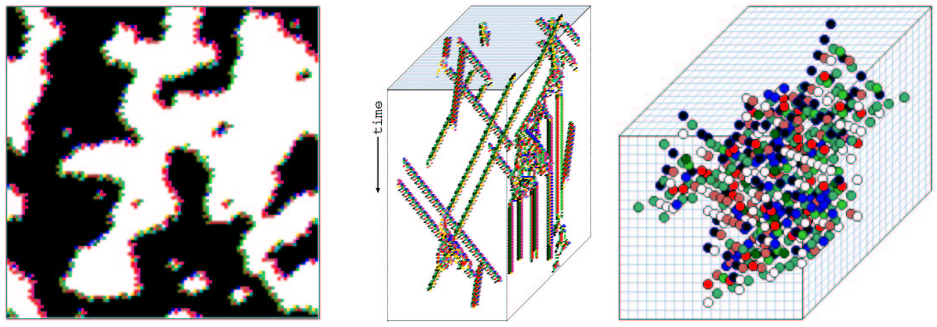


Figure 5: Space-time patterns of the k=3 rule 54 (n=150) from the same initial state showing interacting gliders, which are embedded in a complicated background. *Left*: cells by value. *Right*: cells by neighbourhood lookup, with the background filtered.



(a) 2d 100x100 triangular grid (b) 2d 56x56 +time (c) 3d 20x20x20

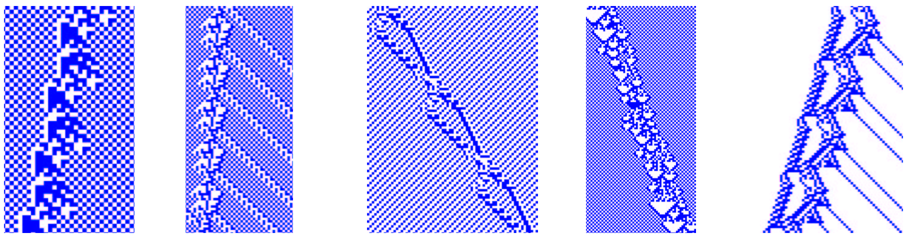
Figure 6: Examples of 2d and 3d CA space patterns. (a) is an evolved time-step of a 2d CA on a k=7 triangular lattice with a reaction-diffusion rule. (b) is the 2d game-of-Life on a 56x56 grid, but with a time dimension added, similar to a 1d space-time pattern. The initial state is set with a number of gliders. (c) is a time-step of a 3d k=7 CA with a randomly selected rule and starting from a single central 1.

3.1 Glider dynamics in CA

A large body of literature is devoted the study space-time patterns in CA. "Glider" or particle dynamics, where coherent configurations emerge and interact, provide a striking instance of self-organization in a simple system. Such dynamics are classified as complex, in contrast to ordered or chaotic[14], a well know example being Conway's 2d "game-of-Life"[1]. Because glider dynamics is relatively rare in CA rule spaces, much study has focused on the few known complex rules in 1d CA. However, an unlimited source of examples are now available, found by the methods described in sections 3.2 - 3.3.

Gliders are embedded within a uniform or periodic background or domain, and propagate at various velocities up the system's speed of light set by the neighbourhood diameter. Gliders are interpreted as dislocation in the background or as the boundary reconciling two different backgrounds[4, 18]. Gliders may absorbed or eject sub-gliders (glider-guns). Compound gliders may emerge made up of sub-gliders re-colliding periodically. Figure 7 shows some examples.

Glider dynamics has been interpreted as occurring at a phase transition in rule-space between order and chaos [9], relative to the rule parameters λ[9] and Z[16] (see section 6.2). Input-entropy provides a measure on space-time dynamics that allows the automatic classification of rule-space (see below).



(a)7e8696de (b)89ed7106 (c)89ed7106 (d)b51e9ce8 (e)b51e9ce8

Figure 7: Gliders, glider guns and compound gliders in k=5 1d CA. (c) is a compound glider made up of two independent gliders locked into a cycle of repeating collisions. (d) is a glider with a period of 106 time-steps. (e) is a compound glider-gun.

3.2 Input entropy

Keeping track of the frequency of rule-table look-ups (the k-block frequency, or "look-up frequency") in a window of time-steps, provides a measure, the variance of input-entropy over time, which is used to classify 1d CA automatically for a spectrum of ordered, complex and chaotic dynamics[22]. The method allows screening out rules that support glider dynamics and related complex rules, giving an unlimited source for further study. The method also shows the distribution of rule classes in the rule-spaces of varying neighbourhood sizes. The classification produced seems to correspond to our subjective view of space-time dynamics, and to global measures on the "bushiness" of typical sub-trees in attractor basins, characterized by the distribution of in-degree sizes in their branching structure.

The look-up frequency can be represented by a histogram (figure 8) which distributes the total of $n \times w$ lookups among the 2^k neighbourhoods (shown as the fraction of total lookups), where n=system size, w=the window of time-steps defined and k=neighbourhood size. The Shannon entropy of this frequency distribution, the "input-entropy" S, at time-step t, for one time-step (w=1), is given by, $S^t = -\sum_{i=1}^{2^k} \left(\frac{Q_i^t}{n} \times log \left(\frac{Q_i^t}{n} \right) \right)$, where Q_i^t is the look-up frequency of neighbourhood i at time t. In practice the measures are smoothed by being taken over a moving window of time-steps (w=10 in figure 8).

Figure 8 shows typical examples of ordered, complex and chaotic dynamics in 1d CA, with input-entropy plots and a snapshot of the lookup frequency histogram alongside. In ordered dynamics the entropy quickly settles at a low value with low or zero variance. In chaotic dynamics the entropy settles at a high value, but again with low variance. Both ordered and chaotic dynamics have low input-entropy variance. By contrast, in complex dynamics the entropy

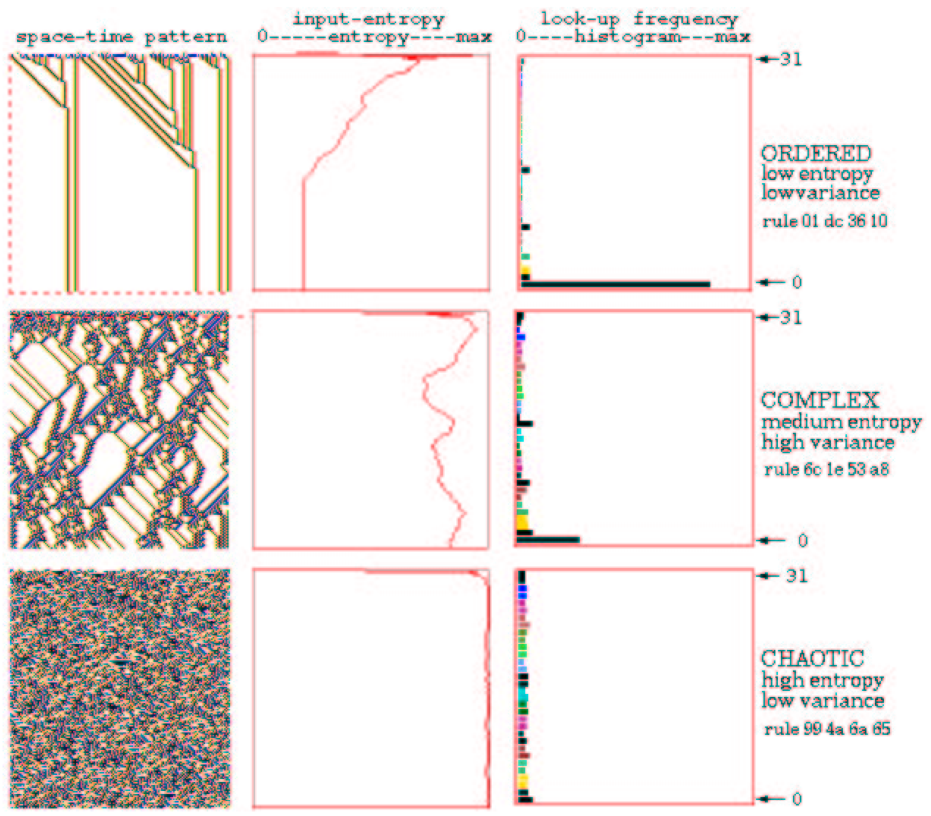


Figure 8: Typical 1d CA Space-time patterns showing ordered, complex and chaotic dynamics (n=150, k=5,rule numbers shown in hex). Alongside each space-time pattern is a plot of the input-entropy, where only complex dynamics (*centre*) exhibits high variance because glider collisions make new gliders.

fluctuates erratically both up and down for an extended time, because glider collisions produce new gliders, often via a temporary zone of chaotic dynamics. Complex rules can be recognized by eye, subjectively. Input-entropy variance provides a non-subjective measure for recognizing complex rules automatically.

A related method of visualizing the entropy-variance is to plot input-entropy against the density of 1s relative to a moving window of time-steps. Each rule produces a characteristic cloud of points which lie within a parabolic envelope because high entropy is most probable at medium density, low entropy at either low or high density. Each complex rule produces a plot with its own distinctive signature, with high input-entropy variance. Chaotic rules, on the other hand, will give a flat, compact cloud at high entropy (at the top of the parabola). For ordered rules the entropy rapidly falls off with very few data points because the system moves rapidly to an attractor.

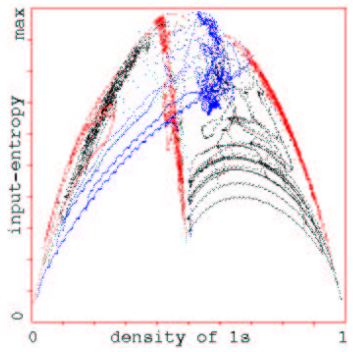


Figure 9: Entropy-density scatter plot. Input-entropy is plotted against the density of 1s relative to a moving window of time-steps w=10. k=5, n=150. Plots for a number of complex rules from the automatic sample (section 3.3) are show superimposed, each of which has its own distinctive signature, with a marked vertical extent, i.e. high input-entropy variance. About 1000 time-steps are plotted from several random initial states for each rule.

3.3 Automatically classifying rule-space

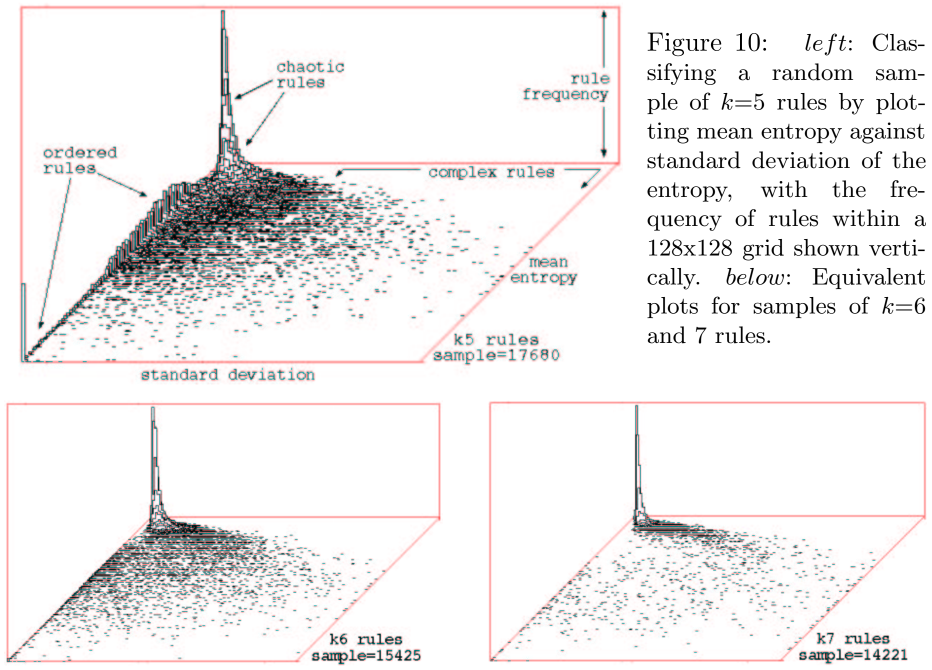


Figure 10: *left*: Classifying a random sample of k=5 rules by plotting mean entropy against standard deviation of the entropy, with the frequency of rules within a 128x128 grid shown vertically. *below*: Equivalent plots for samples of k=6 and 7 rules.

To distinguish ordered, complex and chaotic rules automatically, the mean input-entropy taken over a span of time-steps is plotted against the standard deviation of the input entropy. Figure 10 summarizes how random samples of k=5, 6 an 7 rules where classified by this method. For each rule, the data was gathered from 5 runs from random initial states, for 430 time-steps, discounting the first 30 to allow the system to settle, with w=5 as the size of the moving window of time-steps.

Chaotic rules are concentrated in the top left corner "tower", ordered rules on the left with lower entropy. Complex rules have higher standard deviation, and are spread out towards the right. There is a fairly distinct boundary between ordered and chaotic rules, but a gradual transition from both towards the complex rules. As the standard deviation decreases glider interactions either become more frequent, transients longer, tending towards chaos, or less frequent, transients shorter, tending towards order. The plots for k=6 and k=7 rules indicate a greater frequency of chaotic rules at the expense of ordered and complex rules for greater k. The decrease in ordered rules is especially marked.

To check whether the expected dynamics (recognized subjectively) corresponds to the measures as plotted, the dynamics of particular rules at different positions on the plots can be easily examined in DDLab, for example with a mouse click on the scatter plot. Preliminary scans confirm that the expected behaviour is indeed found, but further investigation is required to properly demarcate the space between ordered, complex and chaotic rules and to estimate the proportion of different rule classes for different k.

Input entropy is a local measure on the space-time patterns of typical trajectories. The distribution of the rule samples according to these local measures may be compared with global measures on convergence in attractor basins, G-density and the in-degree frequency, described in section 8. Preliminary results indicate a strong relationship between these global measures and the rule sample input-entropy plots.

4 RBN space-time patterns

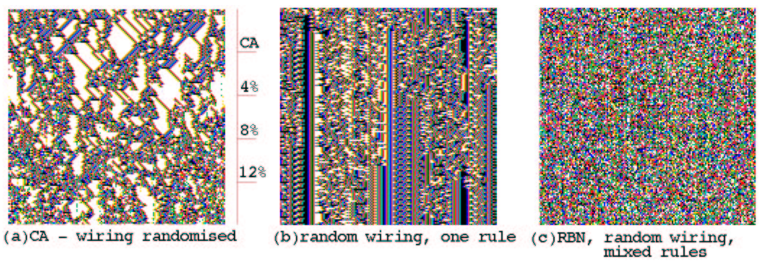
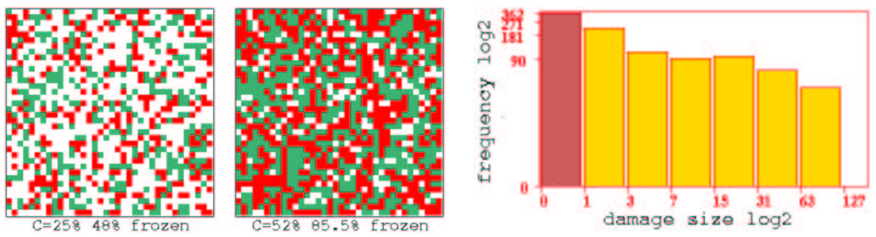

CA
4%
8%
12%
(a)CA – wiring randomised
(b)random wiring, one rule
(c)RBN, random wiring, mixed rules

Figure 11: Space-time patterns for intermediate 1d architecture, from CA to RBN. n=150, k=5, 150 time-steps from a random initial state. (a) Starting off as a complex CA (rule 6c1e53a8 as in figure 8), 4% (30/750) of available wires are randomized at 30 time-step intervals. The coherent pattern is progressively degraded. (b) A network with local wiring but mixed rules, vertical features are evident. (c) RBN, random wiring and mixed rules, with no bias, shows maximal chaotic dynamics.

In contrast to CA, glider dynamics in general cannot occur in RBN because of their irregular architecture. Figure 11 (left) shows glider dynamics degrading as local wiring is progressively scrambled. An alternative order-chaos notion in RBN is the balance between "frozen", stabilized, regions and changing regions in the space-time pattern[8]. Stable regions are characteristic of RBN with low connectivity, $k \leq 3$, because rules which induce stability are relatively frequent in these rule-spaces. To induce stability for $k \geq 4$, where chaotic rules become overwhelmingly predominant, biases on rules must be imposed, low λ (see section 6.2) or a high proportion of "canalizing" inputs. In a rule's lookup table, an input wire is canalizing if a particular input (0 or 1) determines, by itself, the neighbourhood's output. A rule's degree of canalization can be from 0 to k, for the same output; for the network it is the percentage of all inputs that are canalizing, C. An RBN's order-chaos characteristics, for varying C, are captured by the measures illustrated in figure 12, and described below.



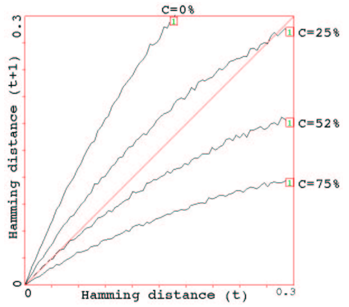


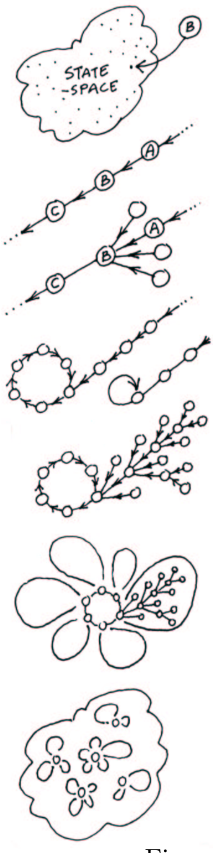
Figure 12: Order-chaos measures for a RBN 36×36, $k = 5$. C = the percentage of canalizing inputs in the randomly biased network. *top left*: frozen elements that have stabilized for 20 time-steps are shown, 0s-green, 1s red, otherwise white, for C=25% and 52%. *top right*: the log-log "damage spread" histogram for C=52%, sample size about 1000. *left*: the Derrida plot for C=0%, 25%, 52%, and 75%, for 1 time-step, H_t=0-0.3, interval = 5, sample for each H_t = 25.

The "Derrida plot"[2], is analogous to the Liaponov exponent in continuous dynamics, and measures the divergence of trajectories based on normalized Hamming distance H, the fraction of bits that differ between two patterns. Pairs of random states separated by H_t, are independently iterated forward by one (or more) time-steps. For a sample of random pairs, the average H_{t+1} is plotted against H_t, and the plot is repeated for increasing H_t (from 0 to 0.3 in figure 12). A curve above the main diagonal indicates divergent trajectories and chaos, below - convergence and order. A curve tangential to the main diagonal indicates a balance.

A related measure is the distribution of "damage spread" resulting from a single bit change at a random position in a random state, for a sample of random states. The size of damage is measured once it has stabilized, i.e. not changed for say 5 time-steps. A histogram (figure 12) is plotted of damage size against the frequency of sizes. Its shape indicates order or chaos in the network, where a balance between order and chaos approximates to a power law distribution. Results by these measures for $k = 5$, indicate a balance at $C = 52\%$ (see figure 12). There are further measures on basins of attraction as in figure 14.

These methods are applied in the context of RBN models of genetic regulatory networks[8] discussed in section 10. The conjecture is that evolution maintains genetic regulatory networks marginally on the ordered side of the order-chaos boundary to achieve stability and adaptability in the pattern of gene expression which defines the cell type[5].

5 Basins of Attraction



For a network size n, an example of one of its states B might be $1010\dots0110$. *State-space* is made up of all 2^n states, the space of all possible bitstrings or patterns.

Part of a *trajectory* in state-space, where C is a successor of B, and A is a *pre-image* of B, according to the dynamics of the network.

The state B may have other pre-images besides A, the total is the *in-degree*. The pre-image states may have their own pre-images or none. States without pre-images are known as *garden-of-Eden* states.

Any trajectory must sooner or later encounter a state that occurred previously - it has entered an attractor cycle. The trajectory leading to the attractor a *transient*. The period of the attractor is the number of states in its cycle, which may be only just one - a point attractor.

Take a state on the attractor, find its pre-images (excluding the pre-image on the attractor). Now find the pre-images of each pre-image, and so on, until all garden-of-Eden states are reached. The graph of linked states is a *transient tree* rooted on the attractor state. Part of the transient tree is a subtree defined by its root.

Construct each transient tree (if any) from each attractor state. The complete graph is the *basin of attraction*. Some basins of attraction have no transient trees, just the bare "attractor".

Now find every attractor cycle in state-space and construct its basin of attraction. This is the *basin of attraction field* containing all 2^n states in state-space, but now linked according to the dynamics of the network. Each discrete dynamical network imposes a particular basin of attraction field on state-space.

Figure 13: State space and basins of attraction.

The idea of basins of attraction in discrete dynamical networks is summarized in figure 13. Given invariant network architecture and the absence of noise, a discrete dynamical network is deterministic, and follows a unique (though in general, unpredictable) trajectory from any initial state. When a state that occurred previously is re-visited, which must happen in a finite state-space, the dynamics becomes trapped in a perpetual cycle of repetitions defining the attractor (state cycle) and its period (minimum one, a stable point).

These systems are dissipative. A state may have multiple "pre-images" (predecessors), or none, but just one successor. The number of pre-images is the state's "in-degree". In-degrees greater than one require that transient states exist outside the attractor. Tracing connections backwards to successive pre-images of transient states will reveals a tree-like topology where the "leaves" are states without pre-images, known as garden-of-Eden states. Conversely, the flow in state-space is convergent. Measures of convergence are G-density, the fraction of states that are garden-of-Eden, and the distribution of in-degrees, described in section 8. The set of transient trees rooted on the attractor is its basin of attraction (figure 1). The local dynamics connects state-space into a number of basins, the basin of attraction field, representing the systems global dynamics (figure 15).

6 Computing Pre-images

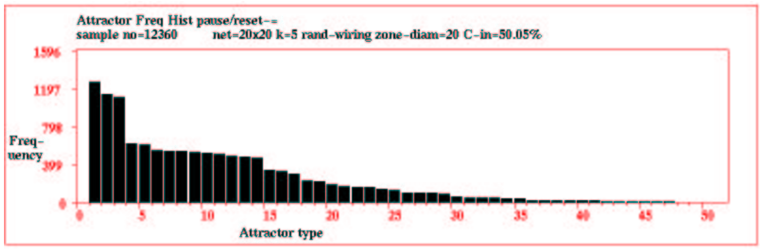


Figure 14: Statistical data on attractor basins for a large network; a 2d RBN 20×20, k=5, with fully random wiring and a fraction of canalizing inputs C=50%. The histogram shows attractor types and the frequency of reaching each type from 12,360 random initial states, sorted by frequency. 46 different attractors types where found, their periods ranging from 4 to 102, with average transient length from 21 to 113 time-steps. The frequency of arriving at each attractor type indicates the relative size of the basin of attraction.

Attractor basins are constructed with algorithms that directly compute the pre-images of network states[16, 17, 21]. This allows the network's dynamics, in effect, to be run backwards in time. Backward trajectories will, as a rule, diverge. Different reverse algorithms apply to networks with different sorts of connectivity. The most computationally efficient algorithm applies to 1d networks with local wiring, taking advantage of the regularity of connections. The wiring must be uniform, as for 1d CA, but the network may have a mix of rules. Analogous algorithms could be derived for 2d and 3d networks, but have not been implemented. An alternative algorithm is required for RBN with their non-local connections and possibly mixed k. This algorithm also applies to CA of any dimension or geometry, as CA are just a sub-class of RBN.

Provided $k << n$, these methods are in general orders of magnitude faster than the brute force method (section 9.1), constructing an exhaustive map resulting from network dynamics, a method which rapidly becomes intractable with increasing network size and so is limited to very small systems. However, the exhaustive method may be applied to all types of network, and also allows the attractor basins of random maps to be constructed, as described in section 9. The agreement of these three independent methods, and other checks, give considerable confidence in the accuracy of the pre-image computations.

Some basic information on attractor basin structure can be found by statistical methods, first applied by Walker[13], as shown in figure 14. These are also implemented in DDLab and are appropriate for large networks. Trajectories are run forward from many random initial states looking for a repeat in the network pattern to identify the range of attractor types reached. The frequency of reaching a given attractor type indicates the relative size of the basin of attraction, and other data are extracted such as the number of basins, and the length of transients and attractor cycles.

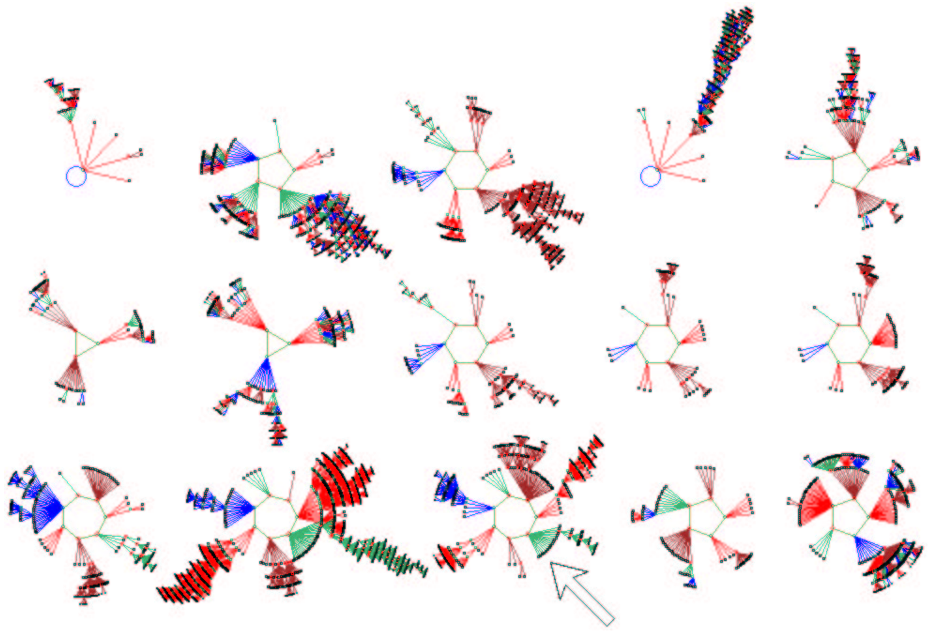


Figure 15: The basin of attraction field of a random Boolean network (n=13, k=3). The 2^{13} = 8192 states in state space are organized into 15 basins, with attractor periods ranging between 1 and 7. The number of states in each basin is: 68, 984, 784, 1300, 264, 76, 316, 120, 64, 120, 256, 2724, 604, 84, 428. Figure 1 shows the arrowed basin in more detail. Right: the network's architecture, its wiring/rule scheme.

cell	wiring	rule
12	10,1,7	86
11	6,2,9	4
10	10,10,12	196
9	2,10,4	52
8	5,6,8	234
7	12,5,12	100
6	1,9,0	6
5	5,7,5	100
4	4,11,7	6
3	8,12,12	94
2	11,6,12	74
1	6,5,9	214
0	12,9,6	188

6.1 The CA reverse algorithm

Consider a 1d CA size n (indexed $n-1\ldots0$) and neighbourhood k. To find the all pre-images of a state A, let P be a "partial pre-image" where at least $k-1$

continuous bits (on the left) up to and including P_i, are known. Now find the next unknown bit to the right, P_{i-1}, consistent with the rule-table. ($\bullet$ indicates known, $\star$ unknown, bits),

	P_{i+1}	P_i	P_{i-1}	
... partial pre-image P ...	$\bullet$	$\bullet$	$\star$	compare the outputs of $P_{i+1}, P_i, \star$
		$\bullet$		with each other and with A_i
... known state A ...		A_i		

If $k = 3$ (for example), the bitstring $P_{i+1}, P_i, \star$ corresponds to two neighbourhood entries in the rule-table. When their outputs, T_1 and T_2, are compared with each other and with A_i there are three possible consequences. The permutation is either deterministic, ambiguous or forbidden.

1. deterministic: if $T_1 \neq T_2$, then P_{i-1} is uniquely determined, as there is only one valid neighbourhood with the output A_i.

2. ambiguous: if $T_1 = T_2 = A_i$, then both 0 and 1 are valid solutions for P_{i-1}. The partial pre-image must be duplicated, with $P_{i-1} = 0$ in one version and $P_{i-1} = 1$ in the other.

3. forbidden: if $(T_1 = T_2) \neq A_i$, then P_{i-1} has no valid solution.

If forbidden (3) the partial pre-image P is rejected. If deterministic or ambiguous (1 or 2) the procedure is continued to find the next unknown bit to the right. However, in the ambiguous case (2), both alternative partial pre-images must be continued. In practice one is assigned to a stack of partial pre-images to be continued at a later stage. As the procedure is re-applied to determine each successive unknown bit towards the right, each incidence of ambiguous permutations will require another partial pre-image to be added to the stack. Various refinements can limit this growth.

The procedure is continued to the right to overlap the assumed start string, to check if periodic boundary conditions are satisfied; if so the the pre-image is valid. The procedure is re-applied to each partial pre-image taken from the partial pre-image stack, starting at the first unknown cell. Each time an ambiguous permutation (2) occurs a new partial pre-image must be added to the stack, but the stack will eventually be exhausted, at which point all the valid pre-images containing the assumed start string will have been found. The procedure is applied for 2^{k-1} start strings, assuming the different possible values of the first $k-1$ bits. The reverse algorithm is applied from left to right in DDLab, but is equally valid when applied from right to left. Examples are given in [16, 21].

6.2 The Z parameter

A by product of the CA reverse algorithm is the probability of the next unknown bit being *deterministic* (section 6.1(1)). Two versions of this probability are calculated from the rule-table. Z_{left} for the reverse algorithm applied from left to right, and Z_{right} for the converse. The Z parameter is the greater of these values. For Z=1 it can be shown[16] that for any system size n, the maximum in-degree, $I_{max} \leq 2^{k-1}$, because the next unknown bit is always uniquely determined, so the assumed start string of length $k-1$ may generate at most 2^{k-1} pre-images. If only one of Z_{left} or Z_{right}=1, $I_{max} < 2^{k-1}$, because at least one assumed start string must be forbidden (section 6.1(3)). At the other extreme, for Z=0, all state space converges on the state all-0s or all-1s in one step. For high Z, low in-degree (relative to system size n) is expected in attractor basins, growing at a slow rate with respect to n. Conversely, for low Z, high relative in-degree is expected growing quickly with respect to n. High Z predicts low convergence and chaos, low Z predicts high convergence and order.

The 2^k neighborhoods of size k, each indexed $k-1 \ldots 0$, each have an output T (0 or 1) which makes up the rule-table (section 2), and may be expressed as $a_{k-1}, a_{k-2}, \ldots a_1, a_0 \rightarrow T$. To calculate Z_{left} from the rule table, let n_k be the count of rule-table entries belonging to deterministic pairs, such that,

$a_{k-1}, a_{k-2}, \ldots a_1, 0 \rightarrow T$ and $a_{k-1}, a_{k-2}, \ldots a_1, 1 \rightarrow \overline{T}$ (not T)

The probability that the *next bit* is determined because of the above is given by, $R_k = n_k/2^k$. This is a first approximation of Z_{left}.

Let n_{k-1} be the count of rule-table entries belonging to deterministic 4-tuples (where "$\star$" may be 0 or 1), such that,

$a_{k-1}, a_{k-2}, \ldots a_2, 0, \star \rightarrow T$ and $a_{k-1}, a_{k-2}, \ldots a_2, 1, \star \rightarrow \overline{T}$

The probability that the *next bit* is determined because of the above is given by, $R_{k-1} = n_{k-1}/2^k$. This count is repeated if necessary for deterministic 8-tuples where $R_{k-2} = n_{k-2}/2^k$, 16-tuples, 32-tuples, ... up to the special case of just one 2^k-tuple which occupies the whole rule-table. These are are independent non-exclusive probabilities that the *next bit* is determined. The union of the probabilities $R_k \cup R_{k-1} \cup R_{k-2} \ldots = Z_{left}$, is given by the following expression (the order of the probabilities makes no difference to the result),

$Z_{left} = R_k + R_{k-1}(1 - R_k) + R_{k-2}(1 - R_k + R_{k-1}(1 - R_k))) + R_{k-3}(1 - (R_{k-2}(1 - R_k + R_{k-1}(1 - R_k))))) + \cdots$ which simplifies to,

$Z_{left} = R_k + R_{k-1}(1-R_k) + R_{k-2}(1-R_{k-1})(1-R_k) + R_{k-3}(1-R_{k-2})(1-R_{k-1})(1-R_k) + \cdots$

and may be expressed as[2] $Z_{left} = R_k + \sum_{i=1}^{k-1} R_{k-1} \left(\prod_{j=k-i+1}^{k} (1 - R_j) \right)$

where $R_i = n_i/2^k$, and n_i = the count of rule-table entries belonging to deterministic 2^{k-i}-tuples. A converse procedure gives Z_{right}, and the Z parameter = the greater of Z_{left} and Z_{right}. Examples are given in [16, 21].

By virtue of being a convergence parameter, Z is also an order-chaos parameter varying from 0(order) - 1(chaos). Z can be compared with Langton's[9] well known λ parameter[3]. λ is an order-chaos parameter for CA which may have values greater than binary, and measures the density of "non-quiescent" outputs in a rule-table, so for just binary CA, $\lambda = c/2^k$ where c=the count of

[2]Acknowledgment and thanks to Guillaume Barreau and Phil Husbands at COGS, Univ. of Sussex, for deriving this expression.

[3]Other versions of binary λ are "internal homogeneity" introduced earlier by Walker[13], and the P parameter, applied for RBN, which varies between 0.5(chaos)-1(order). $P = c_{max}/2^k$ where c_{max} is the count of 0s or 1s in the rule-table, whichever is *more*, $P = 1 - \lambda_{ratio}/2$. A number of alternative order-chaos parameters have also recently been proposed, for example by Marty Zwick and Burton Voorhees.

1s a rule-table on k inputs. λ varies between 0(order)-0.5(chaos)-1(order). To allow Z and λ to be compared, a normalized version of binary λ is defined[16], $\lambda_{ratio} = 2 \times c_{min}/2^k$ where c_{min} is the count of 0s or 1s in the rule-table, whichever is *less*. λ_{ratio} then varies from 0(order)-1(chaos) just as Z.

Plots of G-density against both the λ_{ratio} and Z parameters, showing the discrepancies as well as similarities, are shown in figure 16, for the 256 k=7 totalistic rules, which reduce to 136 non-equivalent rules in 72 clusters (having equal λ_{ratio} and Z). Points plotted in the top right corner of the λ_{ratio} graph represent λ_{ratio} values that do not correspond to behaviour as expected.

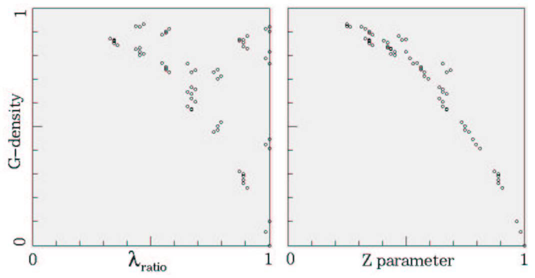


Figure 16: G-density against both λ_{ratio} and Z for the set of k=7 totalistic rules, n=16, for $Z \geq 0.25$. The complete basin of attraction field was generated for each rule and garden-of-Eden states counted.

6.3 The RBN reverse algorithm

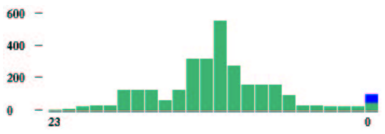


Figure 17: Computing RBN pre-images. The changing size of a typical partial pre-image stack at successive elements. n=24, k=3.

Consider an RBN of size n. Find all pre-images of a state A, $(A_{n-1}, A_{n-2}, \ldots, A_0)$. Each network element A_i, has a pseudo-neighbourhood size k_i (assuming a mixed k network), indexed $k_i - 1, k_i - 2, \ldots, 0$, a wiring scheme W_i, $(w_{k_i-1}, w_{k_i-2}, \ldots, w_0)$, where w_j is a number between $n-1$ and 0, the position of the wire connection from the jth branch of the pseudo-neighbourhood, and a rule-table R_i.

To find the all pre-images of A, let P, $(P_{n-1}, P_{n-2}, \ldots, P_0)$, be a candidate pre-image consisting of *empty* network elements, as yet unassigned to either 0 or 1. Starting with an element of A, A_i, assign bits from all valid pseudo-neighbourhoods in the rule-table R_i, i.e. that are consistent with A_i, to separate copies of P according to the wiring scheme W_i. As there will be a mix of 0s and 1s in R_i, only some of the 2^{k_i} possible pseudo-neighbourhoods will be valid. This will produce a stack of "partial pre-images" with some bits allocated and the remainder empty.

Now repeat the procedure for another element of A, say A_{i-1}, but this time independently for each partial pre-image previously created. If the allocation of a bit to a given partial pre-image conflicts with the bit already assigned, then the partial pre-image is rejected. Otherwise, the partial pre-image is added to the next generation of partial pre-images in a new stack. The allocation will be valid if it is made to an "empty" element, or to an allocated element with an equal bit. Valid allocation increases the size of the partial pre-image stack, conflicts reduce the size of the stack.

This procedure is repeated in turn for the remaining network elements of A. If the stack size is reduced to zero at any stage A has no pre-images. The algorithm works for any ordering of elements in A, though to minimizes the growth of the partial pre-image stack, the order should correspond to the greatest overlap of wiring schemes. The changing size of the stack at successive elements can be displayed in DDLab, an example is shown in figure 17. When the procedure is complete, the final pre-image stack may still have empty network elements, which did not figure in any wiring scheme. These are duplicated so that all possible configurations at empty element positions are represented. The resulting pre-image stack is the complete set of pre-images of A without duplication.

The reverse algorithm for RBN works for networks with any degree of intermediate architecture between RBN and CA, including CA of any dimension. More detailed explanations of the algorithm are given in [17, 21].

7 Constructing and portraying attractor basins

To construct a basin of attraction containing a particular state, the network is iterated forward from the state until a repeat is found and the attractor identified. The transient tree (if it exists) rooted on each attractor state is constructed in turn. Using one of the reverse algorithms, the pre-images of the attractor state are computed, ignoring the pre-image lying on the attractor itself. Then the pre-images of pre-images are computed, until all "garden-of-Eden" states have been reached.

In a similar way, just a subtree may be constructed rooted on a state. Because a state chosen at random is very likely to be a garden-of-Eden state, it is usually necessary to run the network forward by at least one time-step, and use the state reached as the subtree root. Running forward by more steps will reach a state deeper in the subtree so allow a larger subtree to be constructed.

For CA, a considerable speedup in computation is achieved by taking advantage of "rotational symmetry"[16], a property of the regularity of CA and periodic boundary conditions, resulting in equivalent subtrees and basins.

Attractor basins are portrayed as state transition graphs, vertices (nodes) connected by directed edges. States are represented by nodes, by a bit pattern in 1d or 2d, or as the decimal or hex value of the state. In the graphic convention[16, 19], the length of edges decreases with distance away from the attractor, and the diameter of the attractor cycle approaches an upper limit with increasing period. The direction of edges (i.e. time) is inward from garden-of-Eden states to the attractor, and then clockwise around the attractor cycle, as shown in figure 1. Typically, the vast majority of states in a basin of attraction lie on transient trees outside the attractor, and the vast majority of these states are garden-of-Eden states.

8 Attractor basin measures

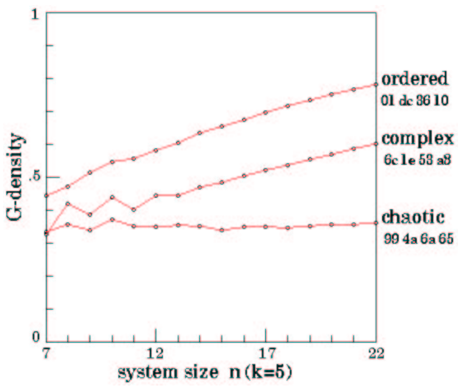


Figure 18: The G-density plotted against system size n, for the ordered, complex and chaotic rules shown in figures 8 and 19. The the entire basin of attraction field was plotted for n= 7 to 22, and garden-of-Eden states counted. The relative G-density and rate of increase with n provides a simple measure of convergence.

Measures on attractor basins include the number of attractors, attractor periods, size of basins, characteristic length of transients and the characteristic branching within trees. The last in particular gives a good measure of the convergence of the dynamical flow in state-space, where high convergence indicates ordered, and low convergence indicates chaotic dynamics.

The simplest measure that captures the degree of convergence is the density of garden-of-Eden states[18], G-density, counted in attractor basins or sub-trees, and the rate of increase of G-density with n as shown in figure 18. A more comprehensive measure is the in-degree frequency distribution, plotted as a histogram. The in-degree of a state is the number of its immediate pre-images. This can be taken on a basin of attraction field, a single basin, a subtrees, or on just part of a subtree for larger systems. Subtrees are portrayed as graphs showing trajectories merging onto the sub-tree root state.

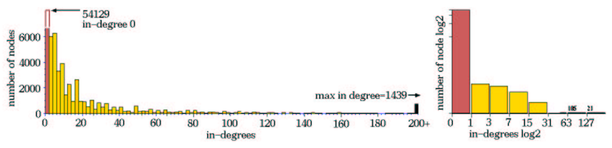
Examples of in-degree histograms for typical sub-trees of ordered, complex, and chaotic rules are shown in figure 19. The horizontal axis represents in-degree size, from zero (garden-of-Eden states) upwards, the vertical axis represents the frequency of the different in-degrees. The system size n=50 for the complex and chaotic rules. For very ordered rules in-degrees become astronomical. The ordered rule shown is only moderately ordered, however the system size was reduced to n=40 to allow easier computation.

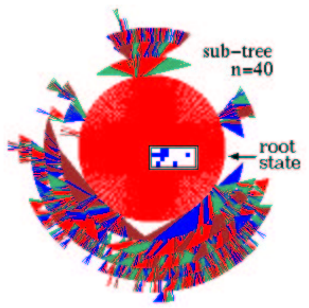
From the preliminary data gathered so far, the profile of the in-degree histogram for different classes of rule is as follows:

Ordered rules: Very high garden-of-Eden frequency and significant frequency of high in-degrees. High convergence.

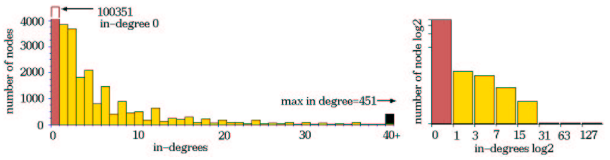
Complex rules: Approximates a power law distribution. Medium convergence.

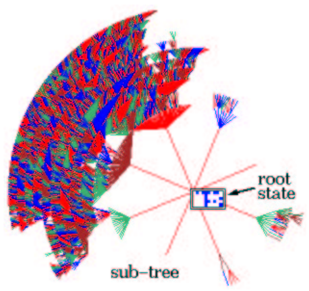
Chaotic rules: Lower garden-of-Eden frequency compared to complex rules, and a higher frequency of low in degrees. Low convergence.



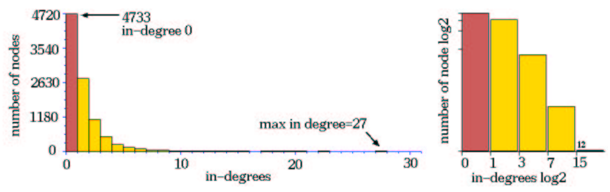


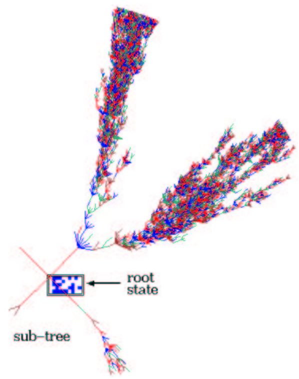
Ordered dynamics. Rule 01dc3610, n=40, Z=0.5625, λ_{ratio}=0.668. *right:* The complete sub-tree 7 levels deep, with 58153 nodes, G-density=0.931.





Complex dynamics. Rule 6c1e53a8, n=50, Z=0.727, λ_{ratio}=0.938. *right:* The sub-tree, stopped after 12 levels, with 144876 nodes, G-density=0.692.





Chaotic dynamics. Rule 994a6a65, n=50, Z=0.938, λ_{ratio}=0.938. *right:* The sub-tree, stopped after about 75 levels, with 9446 nodes, G-density=0.487.

Figure 19: Ordered - Complex - Chaotic CA dynamics. The space-time patterns of the rules are shown in figure 8. The in-degree histogram of a typical sub-tree shown in normal and log-log form.

9 Random Maps

The attractor basins of discrete dynamical networks can be put into the wider context of random graph theory. CA belong to the set of RBN which in turn belong to the set of random directed graphs with out degree one, known as random maps. This is a mapping of the Boolean hypercube of sequences of length n (i.e. a set Q_2^n of size 2^n comprising all binary strings of length n), a mapping from $Q_2^n \rightarrow Q_2^n$. The structures found in random maps correspond to those in the attractor basins of discrete dynamical networks, where each separate component of the graph is made up of trees rooted on just one closed cycle. The structures can be computed in DDLab just as the attractor basins of CA or RBN.

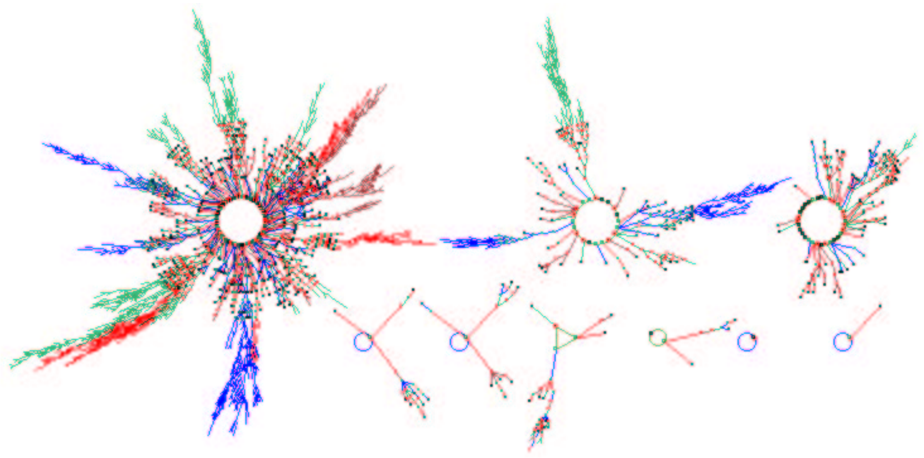


Figure 20: The basin of attraction field of a typical unbiased random map, n=12. The 2^{12} = 4096 states in state space are connected into 9 basins of attraction. The period (p) and size (s) of the biggest three (top row), including their percentage of state-space, are as follows: (1) p=118 s=3204=78.2%. (2) p=20 s=599=14.6%. (3) p=32 s=194=4.74%. The field's G-density=0.37, this is a low value implying chaotic dynamics.

A random map can be constructed by assigning a successor to each state in state-space, i.e. independently assign one successor (or image) V_* also belonging to Q_2^n, chosen at random (or with some bias) to each element V_i of the set Q_2^n. There are $(2^n)^{(2^n)}$ possible mappings. The mapping is represented below as 2^n pairs of strings (or states in state-space), where each image V_* represents a possibly different member of the set Q_2^n.

$$\begin{array}{ccccccccc} V_{2^n-1} & V_{2^n-2} & \dots & V_i & \dots & V_2 & V_1 & V_0 \\ \downarrow & \downarrow & & \downarrow & & \downarrow & \downarrow & \downarrow \\ V_* & V_* & \dots & V_* & \dots & V_* & V_* & V_* \end{array}$$

The list of images is likely to contain repeats, and if so some other members of Q_2^n must be missing from the list. Transitions to some arbitrary element V_x may thus be one-to-one or many-to-one, or may not exist. The latter is a garden-of-Eden state in the terminology of discrete dynamical networks. A representation of the particular mapping may be drawn as a basin of attraction field or fragment thereof just as for CA or RBN, and will have the same general topology of trees rooted on attractor cycles, as shown in figure 20.

Random maps provide the most general context for a discrete dynamical system and are equivalent to a fully connected RBN where $k = n$, (the neighbourhood = the network size). This follows because each cell in the RBN can be assigned an arbitrary output for any network state.

9.1 The Random Map Reverse Algorithm

The "brute force" reverse algorithm for finding the pre-images of states in random maps can also be applied to discrete dynamical networks, RBN and CA. The method depends on first constructing an exhaustive mapping $Q_2^n \rightarrow Q_2^n$. For discrete dynamical networks, the mapping is defined by iterating the network forward by one step from every state in state-space and filling in the image list accordingly. A list of 2^n pairs, each state and its image (successor), is held in a data structure. The pre-images of an arbitrary state S are found by scanning the image list; any occurrence of S in the list gives a pre-image, the state paired with S. If S does not occur in the list it has no pre-images, a garden-of-Eden state.

10 Biological networks

Genetic regulatory networks have been thought of as discrete dynamical networks, to explain how gene expression is able to settle into a number of distinct stable patterns or cell types, despite the fact that all eukaryotic cells in an organism carry an identical set of genes[5, 7, 11, 23]. The gene expression pattern of a cell needs to be stable but also adaptable. Section 4 described biases to RBN to achieve such a balance, and related measures.

Cell types have been interpreted as the separate attractors or basins of attraction into which network dynamics settles from various initial states. Trajectories leading to attractors are seen as the pathways of differentiation. The attractor basins in RBN are idealized models for the stability of cell types against mutations, and also perturbations of the current state of gene activation. Figure 21 illustrates both effects. If a particular reference state (pattern of gene activation) undergoes a 1 bit perturbation, the dynamics may return to the same subtree, the same basin, or it may be flipped to another basin, a different cell type. In this case the basin of attraction field remains unchanged. Alternatively, the network itself my undergo a mutation (in the genotype), resulting in an an altered basin of attraction field (the phenotype).

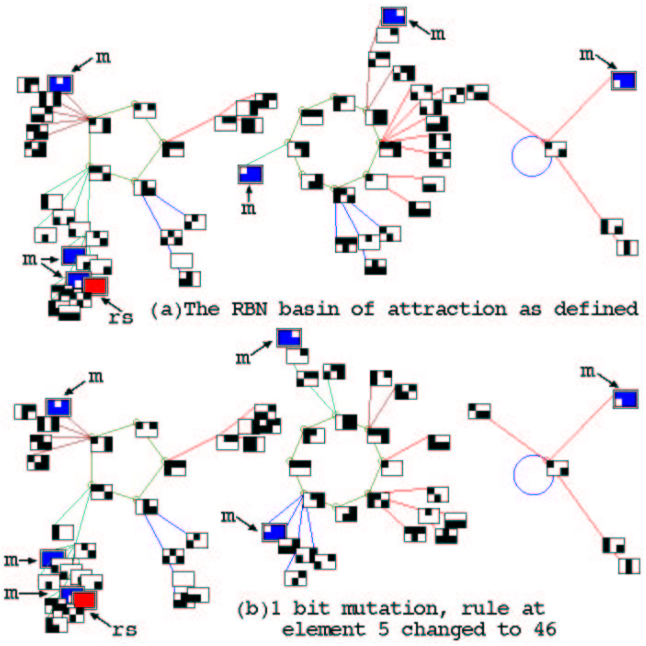
The examples in figure 21 are small so that the pattern at each node can be shown. Larger networks are affected in analogous ways. The consequences of a one bit mutation has a relatively smaller effect with increasing network size. However, a particular one bit mutation may cause drastic consequences whatever the size, such as breaking an attractor cycle. The consequences of moving a connection wire is usually greater than a one bit mutation in a rule.

10.1 Memory

Attractors classify state-space into broad categories, the network's "content addressable" memory in the sense of Hopfield[6]. Furthermore, state-space is categorized along transients, by the root of each subtree forming a hierarchy of subcategories. This notion of memory far from the equilibrium condition of attractors greatly extends the classical concept of memory by attractors alone[17, 20].

It can be argued that in biological networks such as neural networks in the brain or networks of genes regulating the differentiation and adaptive behavior of cells, attractor basins and subtrees, *the network's memory*, must be just right for effective categorization. The dynamics need to be sufficiently versatile for adaptive behavior but short of chaotic to ensure reliable behavior, and this in turn implies a balance between order and chaos in the network.

A current research topic, known as the "inverse problem", is to find ways to deduce network architecture from usually incomplete data on transitions, such as a trajectory. This is significant in genetics, to infer the genetic regulatory network (modeled as RBN) from data on successive patterns of gene expression in the developing embryo[12]. In pattern recognition and similar applications in the area of artificial neural networks, solutions to the inverse problem would provide "learning" methods for RBN to make useful categories[17, 20].


m
m
m
m
m
rs
(a)The RBN basin of attraction as defined
m
m
m
m
m
m
rs
(b)1 bit mutation, rule at element 5 changed to 46

el.	wiring	rule
5	2,4,5	62
4	5,0,1	61
3	4,3,5	108
2	2,5,0	5
1	4,2,1	64
0	3,1,2	231

Figure 21: The basin of attraction field of (a) The RBN (n=6, k=3) as defined in the table (above), and (b) the RBN following a 1 bit mutation to one of its rules. Some differences in the fields are evident. The result or a 1 bit perturbation to a reference state of all 1s (rs) is indicated by its 1 bit mutants (m).

11 Conclusion

Important insights may be gained by considering network dynamics in the context of attractor basins. Some methods of achieving this have been presented, including parameters and measures on particular trajectories that may be related to those on global dynamics. It is hoped that these methods may provide a basis for future research, both in theory and applications, in the many areas of complex systems where network dynamics plays a central role.

Acknowledgments: Thanks to Cosma Shalizi for suggestions, and to the many people who have contributed to this work over the years, notably Chris Langton and Stuart Kauffman.

References

[1] Conway,J.H., (1982) "What is Life?" in *Winning ways for your mathematical plays*, Berlekamp,E, J.H.Conway and R.Guy, Vol.2, chap.25, Academic Press, New York.

[2] Derrida,B., and D.Stauffer, (1986) "Phase transitions in Two-Dimensional Kauffman Random Network Automata", *Europhys.Lett.* 2, 739.

[3] Douglas,R., and A.Wuensche, work in progress.

[4] Hanson,J.E., and J.P.Cruchfield (1997) "Computational Mechanics of Cellular Automata, An example", *Pysica D*, vil. 103, 169-189.

[5] Harris,E.S., B.K.Sawhill, A.Wuensche, and S.Kauffman, (1997) "Biased Eukaryotic Gene Regulation Rules Suggest Genome Behaviour is Near Edge of Chaos", Santa Fe Institute Working Paper 97-05-039.

[6] Hopfield,J.J. (1982) "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of NAS 79*: 2554-2558.

[7] Kauffman,S.A., (1969) "Metabolic stability and epigenisis in randomly constructed genetic nets", *Journal of Theoretical Biology*, 22, 437-467.

[8] Kauffman,S.A., (1993) "*The Origins of Order*", Oxford University Press.

[9] Langton,C.G., (1990) "Computation at the Edge of Chaos, phase transitions and emergent computation", *Physica D 42*, 12-37.

[10] Myers,J.E., (1997) "Random Boolean Networks - Three Recent Results", to be appear in *Complexity.*

[11] Somogyi,R., and C.Sniegoski, (1996) "Modeling the Complexity of Genetic Networks: Understanding Multigenetic and Pleiotropic Regulation", *Complexity*, Vol.1/No.6,45-63.

[12] Somogyi,R, S.Fuhrman, M.Askenazi, A.Wuensche., (1997) "The Gene Expression Matrix", in *Proceedings of the World Congress of Non-Linear Analysis*, in press.

[13] Walker,C.C., and W.R.Ashby, (1966) "On the temporal characteristics of behavior in certain complex systems", *Kybernetik 3*, 100-108.

[14] Wolfram,S., (1984) "Universality and complexity in cellular automata", *Physica 10D*, 1-35.

[15] Wolfram,S., ed. (1986) "*Theory and Application of Cellular Automata*", World Scientific.

[16] Wuensche,A., and M.J.Lesser. (1992) "*The Global Dynamics of Cellular Automata*", Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.

[17] Wuensche,A., (1994) "The Ghost in the Machine", in *Artificial Life III*, ed C.G.Langton, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.

[18] Wuensche,A., (1994) "Complexity in One-D Cellular Automata", Santa Fe Institute Working Paper 94-04-025.

[19] Wuensche,A., (1996) "Discrete Dynamics Lab (DDLab)", http://www.santafe.edu/ wuensch/ddlab.html

[20] Wuensche,A., (1996) "The Emergence of Memory", in to *Towards a Science of Consciousness*, eds. S.R.Hameroff, A.W.Kaszniak, A.C.Scott, MIT Press.

[21] Wuensche,A., (1997) "Attractor Basins of Discrete Networks", CSRP 461, Univ. of Sussex (D.Phil thesis).

[22] Wuensche,A., (1998) "Classifying Cellular Automata Automatically", Santa Fe Institute Working Paper 98-02-018.

[23] Wuensche,A., (1998) "Genomic regulation modeled as a network with basins of attraction", in procceedings of *Pacific Symposium on Biocomputing'98*, World Scientific.