

THE UNIVERSITY OF SUSSEX

**Attractor Basins of Discrete Networks**  
Implications on self-organisation and memory

**Andrew Wuensche**

Submitted for the degree of D.Phil

October 1996  
Revised Feb 1997





# Contents

ABSTRACT .....	ix
ACKNOWLEDGEMENTS .....	x
OTHER PUBLICATIONS .....	xi
DECLARATION .....	xii
OVERVIEW .....	xiii
<b>1. Introduction</b>	
1.1. Discrete Dynamical Networks .....	1
1.2. Random Boolean Networks .....	2
1.3. Cellular Automata .....	2
1.4. Symmetries in Cellular Automata .....	3
1.5. Emergence in Cellular Automata .....	7
1.6. Cellular Automata Local and Global Measures .....	7
1.7. Cellular Automata parameters .....	8
1.8. Emergence in Random Boolean Networks .....	8
1.9. Random Boolean Networks Local and Global measures .....	9
1.10. Random Boolean Network Parameters .....	10
1.11. Learning and the inverse problem .....	10
<b>2. Basins of Attraction</b>	
2.1. Introduction .....	11
2.2. Computing Pre-Images and Attractor Basins .....	12
2.3. Constructing Attractor Basins .....	14
2.4. Portraying Attractor Basins .....	17
<b>3. Attractor Basins of Random Maps</b>	
3.1. Introduction .....	19
3.2. Random Graphs .....	20
3.3. Random Maps .....	20
3.4. Basins of Attraction of Random Maps .....	22
3.5. RBN and CA in the context of Random Maps .....	24
3.6. The Random Map Reverse Algorithm applied to CA and RBN .....	24
3.1. Conclusion .....	25

<b>4. Self-Organisation in One-D Cellular Automata</b>	
4.1. Introduction .....	26
4.2. One-Dimensional CA architecture .....	31
4.3. Space-time patterns and basins of attraction .....	32
4.4. Computing pre-images .....	34
4.5. Constructing and portraying CA attractor basins .....	37
4.6. Complex rules and gliders .....	37
4.7. Glider interactions and basins of attraction .....	49
4.8. Parameters and measures .....	49
4.9. Neighbourhood lookup frequency and entropy .....	50
4.9.1. Ordered Dynamics .....	56
4.9.2. Chaotic Dynamics .....	56
4.9.3. Complex Dynamics .....	56
4.10. Classifying rules by entropy-density .....	59
4.11. Automatically classifying rule-space by input-entropy variance .....	59
4.12. The Z-parameter .....	63
4.13. Calculating the Z-parameter .....	64
4.14. Relating the $\lambda$ parameter and Z .....	67
4.15. Attractor Basin Topology; $G$ -density and In-Degree frequency .....	68
4.15.1. $G$ -density .....	68
4.15.2. In-Degree frequency .....	69
4.16. $G$ -density - variation with system size .....	73
4.17. $G$ -density and the Z-parameter .....	76
4.18. Summary and Discussion .....	78
<b>5. Memory in Random Boolean Networks</b>	
5.1. Introduction .....	81
5.2. Random Boolean Networks .....	87
5.3. Random Boolean Network Architecture .....	87
5.4. Space-time patterns and basins of attraction .....	90
5.5. Computing RBN pre-images .....	91
5.6. Constructing and Portraying RBN Attractor Basins .....	94
5.7. Intermediate architecture between CA and RBN .....	97
5.7.1. Biased random Wiring .....	97
5.7.2. Biased random rule mix .....	99

5.8. Comparing CA and RBN .....	101
5.9. RBN Learning Algorithms .....	102
5.9.1. Learning by re-wiring. ....	104
5.9.2. Learning by mutating the rule scheme. ....	106
5.9.3. Sculpting the basin of attraction field and the inverse problem .....	107
5.10. The Emergence of Memory .....	109
5.10.1. Memory, far from equilibrium. ....	111
5.10.2. RBN as a neural model. ....	114
5.11. Genomic regulatory networks. ....	117
5.11.1. RBN as models of Genomic regulatory networks .....	118
5.11.2. Global and Local Measures .....	119
5.11.3. Attractor basins. ....	121
5.11.4. Frozen Regions and Frozen Skeletons .....	122
5.11.5. The Derrida plot. ....	126
5.11.6. Damage spread. ....	127
5.11.7. Input entropy. ....	129
5.11.8. RBN parameters as applied to genomic regulatory networks .....	129
5.11.9. Network wiring $k$ , and size $n$ .....	130
5.11.10. The $P$ parameter .....	130
5.11.11. Canalising inputs .....	131
5.11.12. Preliminary results .....	131
5.12 Memory in RBN - Conclusion. ....	133
<b>6. Conclusions and open questions .....</b>	<b>135</b>
<b>References .....</b>	<b>136</b>

## APPENDICES

<b>1. Garden-of-Eden density/ System size graphs</b>	
<i>K</i> =3 rules and <i>K</i> =5 totalistic rules .....	142
<b>2. Sample of glider rules, <i>k</i>=5</b>	
Space-time patterns and the basin of attraction field .....	146
<b>3. Sample of glider rules, <i>k</i>=5, 6, 7</b>	
Space-time patterns, neighbourhood lookup frequency and entropy .....	165
<b>4. Examples of subtrees for large lattice sizes <i>n</i>=20 to 150</b>	
for a <i>k</i> =5 rule glider .....	172
<b>5. Automatic Rule Sample</b> .....	174
5.1. <i>k</i> =5 rules .....	175
5.2. <i>k</i> =6 rules .....	180
5.3. <i>k</i> =7 rules .....	185
<b>6. Reverse Algorithm and Z parameter Code</b> .....	190
6.1. Random Maps .....	191
6.2. Cellular Automata .....	192
6.3. Random Boolean Networks .....	197
6.4 Z parameter .....	202
<b>7. Discrete Dynamics Lab</b> .....	206
7.2. DDLab Manual, <i>introductory sections only</i> .....	207
7.3. Network size limitations .....	223

## LIST OF FIGURES

<b>1. Introduction</b>	
1.1. The game-of-life .....	5
1.2. An example of 1d, $k=5$ CA rule with complex dynamics .....	6
<b>2. Basins of Attraction</b>	
2.1. State space and attractor basins .....	13
2.2. The entire basin of attraction field of the "game-of-life" on a $4 \times 4$ grid .....	15
2.3. The basin of attraction field of the "game-of-life" on a $4 \times 4$ grid showing only non equivalent basins .....	15
2.4. Two basins of attraction of the "game-of-life" shown in greater detail .....	16
<b>3. Attractor Basins of Random Maps</b>	
3.1. Examples of typical basin of attraction fields of random maps with Hamming bias ..	23
3.2. Example of a typical basin of attraction field of a random map without bias .....	24
<b>4. Self-Organisation in One-D Cellular Automata</b>	
4.1. Typical space-time patterns of a family $k=5$ rules ranging through <i>ordered-complexr-chaotic</i> dynamics .....	30
4.2. 1d CA architecture .....	31
4.3. The topology of a typical transient tree and the basin of attraction field .....	33
4.4. $k=5$ complex space-time patterns from the automatic sample .....	39
4.5. $k=6$ complex space-time patterns from the automatic sample .....	40
4.6. $k=7$ complex space-time patterns from the automatic sample .....	41
4.7. Gliders with various velocities and backgrounds .....	42
4.8. Glider collisions against a quiescent background .....	43
4.9. A glider forming the boundary between two different periodic backgrounds .....	43
4.10. Examples of glider-guns .....	44
4.11. Example of ordered domains co-existing with a chaotic domain .....	45
4.12. A glider colliding with a compound glider creating a complicated glider-gun .....	45
4.13. A glider with a large diameter and period .....	45
4.14. A glider with a period of 106 time-steps .....	46
4.15. A compound glider made up of two independent gliders .....	46
4.16. The basin of attraction field of a $k=7$ rule .....	48

4.17. Examples of glider dynamics, and of both ordered and chaotic dynamics, showing the lookup frequency histogram and entropy .....	51
4.18. The input-entropy relative to 1, 5 and 20 time-steps .....	52
4.19. $k=5$ . rule index 1 in the automatic sample with high entropy-variance .....	53
4.20. $k=6$ . rule index 1 in the automatic sample with high entropy-variance .....	54
4.21. $k=7$ . rule index 1 in the automatic sample with high entropy-variance .....	55
4.22. Input-entropy plotted against the pattern density for a chaotic rule, and a number of superimposed plots for complex rules. ....	58
4.23. Mean entropy plotted against standard deviation for a random sample of 17680 $k=5$ rules, and the distribution .....	60
4.24. Mean entropy plotted against standard deviation for a random sample of 15425 $k=6$ rules, and the distribution .....	61
4.25. Mean entropy plotted against standard deviation for a random sample of 14221 $k=7$ rules, and the distribution .....	62
4.26. The relationship between $\lambda_{\text{ratio}}$ and the $Z$ parameter for $k=5$ , $k=7$ and $k=9$ rules ...	66
4.27. The relationship between $\lambda_{\text{ratio}}$ and the $Z$ parameter for for the set of 256 $k=7$ totalistic rules .....	67
4.28. The basin of attraction field, and its in-degree frequency histogram, of an ordered $k=5$ rule, $n=18$ .....	69
4.29. The basin of attraction field, and its in-degree frequency histogram, of a complex $k=5$ rule, $n=18$ .....	70
4.30. The basin of attraction field, and its in-degree frequency histogram, of a chaotic $k=5$ rule, $n=18$ .....	71
4.31. In-degree histograms of subtree fragments, $n= 50$ . ....	72
4.32. $G$ -density plotted against $n$ for the 38 $k=5$ rules in appendix 2, and $G$ -density plotted against the $Z$ parameter. ....	73
4.33. Subtrees for $n=150$ .. ....	74
4.34. $G$ -density plotted against the $Z$ parameter for samples of $k=5$ , $k=7$ and $k=9$ rules ...	75
4.35. Plots of the $G$ -density against both $\lambda_{\text{ratio}}$ and the $Z$ parameter for the set of $k=7$ totalistic rules. ....	76
4.36. A typical space-time pattern and basin of attraction field of a $k=5$ rule taken from the sample in appendix 2. ....	77

<b>5. Memory in Random Boolean Networks</b>	
5.1. A basin of attraction of a random Boolean network .....	82
5.2. The basin of attraction field of a random Boolean network .....	83
5.3. RBN architecture .....	88
5.4. Computing RBN pre-images, the partial pre-images stack .....	94
5.5. The basin of attraction field of a randomly wired, single rule network .....	95
5.6. Space time patterns and pattern density, single rule, local wiring, showing the wiring being cumulatively randomised. ....	96
5.7. Intermediate architectures between CA and fully random networks .....	97
5.8. Examples of the space time patterns in non-local CA, including bi-stable pattern density .....	98
5.9. Space time patterns and pattern density, mixed rules, local wiring, showing the wiring being cumulatively randomised .....	100
5.10. The basin of attraction field of a random Boolean network with only one attractor, $n=13$ , $k=3$ .....	101
5.11. A sequence of learning steps .....	103
5.12. Learning pre-images .....	104
5.13. The basin of attraction field of an RBN showing states as patterns on a $3 \times 3$ grid ..	110
5.14. The wiring/rule scheme of the RBN shown in figure 5.12 .....	112
5.15. A network element of a RBN represented as an idealised neuron .....	115
5.16. A RBN network serving as an idealised model of a semi-autonomous population of idealised neurons .....	116
5.17. A neural model of weakly coupled semi-autonomous RBN .....	117
5.18. The consequences of mutating a single bit of a small RBN .....	120
5.19. Data on attractor basins for a $36 \times 36$ RBN, .....	122
5.20. Frozen regions and the pattern of canalising inputs in a $36 \times 36$ RBN, $C=20\%$ ...	123
5.21. Frozen skeleton types in a $36 \times 36$ RBN .....	124
5.22. A set of Derrida plots for a $36 \times 36$ RBN, $C=0\%-70\%$ .....	125
5.23. Derrida plots in greater detail, $C=52\%$ , fully random and confined wiring .....	125
5.24. Damage spread in a $36 \times 36$ RBN, $C=52\%$ .....	127
5.25. Damage spread distribution in $36 \times 36$ RBN, $C=40\%$ and $52\%$ .....	128
5.26. Random wiring of a gene in a $36 \times 36$ RBN .....	129
5.27. Frozen regions in a $36 \times 36$ RBN, $C=10\%$ , $30\%$ and $52\%$ .....	132





# THE UNIVERSITY OF SUSSEX

## **Attractor Basins of Discrete Networks** Implications on self-organisation and memory

**October 1996**

**Andrew Wuensche**

### ABSTRACT

New tools are available for reconstructing the attractor basins of discrete dynamical networks where state-space is linked according to the network's dynamics. In this thesis the computer software "Discrete Dynamics Lab" is applied to examine simple networks ranging from cellular automata (CA) to random Boolean networks (RBN), that have been widely applied as idealised models of physical and biological systems, to search for general principles underlying their dynamics. The algorithms and methods for generating pre-images for both CA and RBN, and reconstructing and representing attractor basins are described, and also considered in the mathematical context of random directed graphs.

RBN and CA provide contrasting notions of self-organisation. RBN provide models of hierarchical categorisation in biology, for example memory in neural and genomic networks. CA provide models at the lower level of emergent complex pattern. New measures and results are presented on CA attractor basins and how they relate to measures on local dynamics and the Z parameter, characterising ordered to "complex" to chaotic behaviour. A method is described for classifying CA rules by an entropy-variance measure which allows glider rules and related complex rules to be found automatically giving a virtually unlimited sample for further study.

The dynamics of RBN and intermediate network architectures are examined in the context of memory, where categorisation occurs at the roots of subtrees as well as at attractors. Learning algorithms are proposed for "sculpting" the basin of attraction field. RBN are proposed as a possible neural network model, and also discussed as a model of genomic regulatory networks, where cell types have been explained as attractors.

## Acknowledgements

I would like to express my gratitude to the many people who have contributed to this work through discussions, suggestions and encouragement.

At COGS, the School of Cognitive and Computing Sciences at the University of Sussex, I am especially grateful to Pedro Paulo Balbi de Oliviera, Inman Harvey, Phil Husbands, Harry Barrow, Guillaume Barreau and members of the "Alergic" discussion group.

I would like to thank the Santa Fe Institute and its staff for their continued support. My thanks to Christian Reidys, Brosl Hasslacher, Howard Gutowitz, Crayton Walker, Josh Smith and Steve Harris. I am especially grateful to Chris Langton and Stuart Kauffman for their support and involvement in my research over the years. My thanks to Simon Fraser for helping to port DDLab to UNIX and the Mac.

Andy Wuensche

The University of Sussex  
School of Cognitive and Computing Sciences  
*andywu@cogs.susx.ac.uk*

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

home address  
Rt.1. Box 92 ac  
Santa Fe, New Mexico 87501, USA  
tel 505 455 7330 fax 455

## Other publications

The following publications and software are closely related to this thesis.

### the author's book

Wuensche, A., and M.J. Lesser. *"The Global Dynamics of Cellular Automata; An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata"*, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, Reading, MA (1992).

Wuensche, A., *"The Ghost in the Machine; Basin of Attraction Fields of Random Boolean Networks"*, CSRP 281, University of Sussex (1993), published in *"Artificial Life III"*, ed C.G. Langton, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, Reading, MA, (1994).

Wuensche, A., *"Complexity in One-D Cellular Automata; Gliders, Basins of Attraction and the Z Parameter"*, Cognitive Science Research Paper 321, Univ. of Sussex, (1994). A updated version of this paper has been accepted for *"Complexity"*.

Wuensche, A., *"The Emergence of Memory, Categorisation far from Equilibrium"*, Cognitive Science Research Paper 346, Univ. of Sussex, (1994), published in *"Towards a Science of Consciousness"*, eds. S.R. Hameroff, A.W. Kaszniak, A.C. Scott, MIT Press (1996).

Wuensche, A., *"Discrete Dynamics Lab (DDLab)"*, (1994-1996), software and manual. Interactive graphics software for investigating discrete dynamical networks including their attractor basins. The software and documentation is published at MIT Press Alife Online, at the Santa Fe Institute.

Download instructions:

DDLab home page: <http://www.santafe.edu/~wuensch/ddlab.html>,

MIT Press Alife Online: <http://alife.santafe.edu/alife/software/ddlab.html>

ftp: <ftp://alife.santafe.edu/pub/SOFTWARE/ddlab>

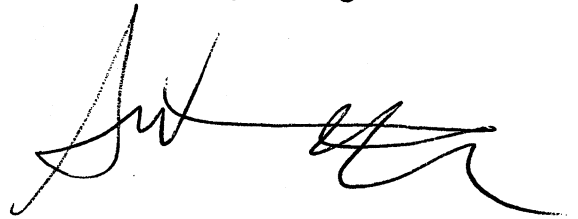
A preliminary *http* version of the DDLab manual is available, thanks to Pedro Paulo Balbi de Oliveira, at,

[http://www.lac.inpe.br/DLAB\\_MAN/ddlab.html](http://www.lac.inpe.br/DLAB_MAN/ddlab.html)

DDLab is available for various operating systems and platforms including UNIX/XWindows/SUN, hpux/X11/HP, DOS/PC, Linux and Mac. It is becoming widely used for research in complex systems, cellular automata theory and areas of theoretical biology, as well as in education. DDLab continues to be developed. See appendix 7 for further details.

## **Declaration**

**I hereby declare that I have not submitted this thesis, including the other publications and software mentioned, either in the same or a different form, to this or any other University for a degree.**

A handwritten signature in black ink, consisting of a series of fluid, connected strokes. The signature is positioned to the right of the declaration text.

## Overview

Discrete Dynamical Networks are central to our current perception of a wide range of natural and artificial phenomena drawn from many areas of science; from physics to biology to cognition; to social and economic organisation; to parallel computation and Artificial Life.

It can be argued that the dynamics of non-equilibrium parallel processing systems composed of discrete interacting components create and maintain much of the complex biological phenomena that dominate our world. Various types of discrete dynamical networks are increasingly being applied as idealised models of physical and biological phenomena. For this reason, and also because of their intrinsic interest as mathematical/physical systems, finding general principles underlying the dynamics of the idealised networks themselves has become an important task. Their behaviour is difficult to describe by classical mathematics using its tools of partial differential equations and continuous dynamics.

This thesis applies computer simulation tools developed by the author, known as "Discrete Dynamics Lab" (DDLab) and available on the Internet (Wuensche 1996), to examine a range of simple discrete dynamical networks ranging from cellular automata (CA) to random Boolean networks (RBN). In particular local dynamics is placed in the context of the global dynamics of the network represented by its basin of attraction field or fragment thereof. The terminology is borrowed from the notion of the "phase portrait" which proved so powerful in continuous dynamics. Constructing and visualising sub-trees, basins of attraction and the entire basin of attraction field of discrete dynamical networks (referred to collectively as attractor basins), and relating their characteristics to local dynamical trajectories was introduced in the author's book "The Global Dynamics Cellular Automata" (Wuensche and Lesser 1992a). This thesis extends the investigation into CA and embarks into new areas involving RBN. New results of computer experiments are presented, some of which are preliminary based on work in progress, and lines of future enquiry are described. The thesis also allows itself some interpretations, conjectures and speculations relating to the results, and how these might provide insights into self-organisation and memory. It is hoped that the distinction between hard results and more general discussion will be clear.

The capability of constructing attractor basins depends on reverse algorithms invented by the author for directly computing the predecessors (known as pre-images) of network states. Different reverse algorithms apply to CA and RBN, though the RBN method encompasses CA.

These methods are in general orders of magnitude faster than the brute force method, constructing an exhaustive map resulting from network dynamics. This exhaustive method rapidly becomes computationally intractable with increasing network size so is limited to small systems. It applies to all network types and also allows the attractor basins of random maps to be constructed.

These three independent methods together form a valuable reality check on the correctness of the computed pre-images and attractor basins.

The CA reverse algorithm forms the basis for a trajectory convergence parameter, named the  $Z$  parameter (Wuensche and Lesser 1992a, Wuensche 1994a), a sort of inverse Liapouov exponent, which predicts the bushiness of sub-trees in attractor basins. Measures of bushiness are captured by "garden-of-Eden" density and more generally by the distribution of in-degree. It is shown that these measures correlate with the quality of dynamics, ordered to chaotic, where order turns out to be the most bushy, chaos the least. A histogram of the in-degree distribution shows contrasting profiles for order and chaos, for complex rules the distribution seems to follow a power law.

Whereas RBN provide models in biology because of their non-local connections and heterogeneous rules, CA provide models in physics with its notions of continuous space and universal laws. Contrasting notions of self-organisation are apparent between the two systems. RBN provide models of hierarchical categorisation in biology, for example memory in neural, genomic, and immune networks. CA provide models of self-organisation at the lower level of emergent interacting space-time configurations, particles or "gliders", metaphors for the emergence of life from the stuff of inanimate physics by a process of self-organisation.

In general, coherent space-time configurations cannot emerge in RBN because of their non-locality. On the other hand, a CA's ability to flexibly categorise distributed patterns is subject to severe symmetry constraints. Different and contrasting "edge-of-chaos", or phase transition, notions are seen to apply to CA and RBN. In the case of CA, this is the phase in rule-space where glider behaviour might emerge, quantified by rule parameters such as the  $\lambda$  parameter (Langton 1990) and the author's  $Z$  parameter, measures on the dynamics such as the "input-entropy" and its variance, and by measures on the *topology* of attractor basins and sub-trees, such as the distribution of in-degree (Wuensche 1994a).

In RBN, as applied to gene regulatory networks (Kauffman 1969, Somogyi and Sniegoski 1996), the phase transition represents the phase in the wiring/rule setup where categorisation of patterns of gene activation, the network's memory, is sufficiently versatile for adaptive behaviour but short of chaotic to ensure reliable behaviour. Tuning the degree of "canalisation" at the level of rules (Kauffman 1984, Harris *et al* 1997) moves behaviour across the transition, though the characteristics of network connections also plays a role. A measure at the level of dynamics is given by the Derrida plot (Derrida and Stauffer 1986), analogous to the Liapouov exponent in continuous dynamics. There are further measures such as damage spreading, the percolation of frozen regions, input-entropy over time relating to single genes, and the transient/frozen skeleton/attractor characteristics.

The body of the thesis goes on to discuss these ideas in depth, based on published work by the author and work in progress. The introductory chapter describes and defines simple discrete

dynamical networks ranging from CA to RBN and explains how network dynamics can be understood globally in terms of attractor basins. Comparisons are made with attractor basins in classical continuous dynamics. Previous work and applications in the field are discussed. Ideas of emergence, phase transitions and various measures and parameters relating to CA and RBN dynamics are introduced.

Chapter 2 looks more closely at the methods for constructing and portraying attractor basins.

Chapter 3 considers discrete dynamical networks and their attractor basins in the mathematical context of directed maps in random graph theory. Discrete dynamical networks are a subclass of random maps. Attractor basins of random maps, possibly incorporating some bias in the random mapping, are amenable to probabilistic analysis, which can be checked against a numerical reconstruction using DDLab. This is work in progress in collaboration with Christian Reidys (Reidys and Wuensche 1997).

Chapter 4 focuses on CA and develops ideas first presented in the paper "Complexity in One-D Cellular Automata" (Wuensche 1994a) and subsequent work. The reverse algorithm for generating pre-images, and the  $Z$  parameter which derives from this procedure, are reviewed. The chapter looks in particular at new measures and results on attractor basins and how they relate to measures on local dynamics, characterising ordered to "complex" to chaotic behaviour, where complex behaviour corresponds to Wolfram's class 4 (Wolfram 1984a). A method is described for classifying CA rules by a measure of the variance of input-entropy over time. The method allows glider rules that support coherent interacting configurations and related complex rules to be found automatically giving a virtually unlimited sample for further study. Glider dynamics in CA is of prime interest as an example of self-organisation and emergence in simple systems, where all aspects of the system can be fully defined.

Chapters 5 focus on RBN, and develops ideas first presented in the papers "The Ghost in the Machine" (Wuensche 1993a) and "The Emergence of Memory" (Wuensche 1994b). Recent collaborative work on RBN as models of genomic regulatory networks is also described.

The reverse algorithm for generating pre-images in RBN is explained. The dynamics of RBN and intermediate network architectures are examined, in particular in the context of memory. The notion of memory far from equilibrium, categorisation by the roots of subtrees as well as by attractors is proposed. Algorithms for learning by "sculpting" the basin of attraction field are described. The inverse problem of finding the set of minimal RBN that will satisfy a predetermined set of transitions is discussed. Alternative methods for solving the inverse problem have recently been formulated independently by Manor Askenazi (1996) and John Myers (1996). The biological implications of memory far from equilibrium, and RBN or networks of RBN as the basis of a neural network model is discussed.

Chapter 5 goes on to discuss RBN as applied to model genomic regulatory networks, where cell types are explained as attractors or "frozen skeletons" in the dynamics of the network. DDLab is being used as a simulation platform for these models, and to extract various measures distinguishing dynamics between order and chaos. The results and methods are described. This is work in progress in collaboration with Stuart Kauffman and others. (Harris *et al* 1997, Somogyi and Sniegoski 1996a).

The thesis concludes with an overview of future work and open questions.



# Chapter 1

## Introduction

### 1.1. Discrete Dynamical Networks

A important challenge in science is the study of how complex behaviour at new levels of description is able to emerge from simpler components and interaction laws. In biology there is a growing awareness that self-organisation, as well as selection, is a force in evolution (Kauffman 1993, Goodwin 1994). Understanding self-organisation in complex systems is basic to confronting the enigma of how life and mind have emerged from matter.

One approach to such a quest is to model systems which are as simple as possible, yet support complex emergent behaviour. Take a system where all parameters are discrete; space, value and time. A collection of simple elements or "cells", sometimes referred to as "finite state machines", are connected up into a network by directed couplings. Each cell may be thought of as having one output terminal and a set of input "wires" which are arbitrarily connected to available output terminals. Each cell may take on a particular value or attribute, the cell's state, from a finite alphabet. For simplicity the alphabet may be taken as binary, for example a network of light bulbs that are either on or off. The state of each cell will depend on the states of the cells at the previous "time-step" that connect to it by its input wires, according to a logical rule. The rule can be thought of as a lookup table, a logical expression or as a physical arrangement of the input wiring into a combinatorial circuit with logical gates, AND, OR and NOT.

All cells update their states in parallel, that is synchronously, in discrete time-steps. An initial on-off pattern is assigned to the network and the system is allowed to iterate; connections and local rules stay constant. This is a dynamical system which moves through its set of all possible on-off network patterns, its phase-space or state-space. The system is deterministic; any initial pattern will result in one particular trajectory through state-space. It is dissipative; a given trajectory can merge with others but cannot diverge (there is an asymmetric arrow of time). Given a network of finite size any trajectory must eventually arrive at a repetitive sequence of network states, a point attractor or state cycle attractor. Merging trajectories flowing to the attractor form a *basin of attraction*, an object in space-time with a topology of trees rooted on cycles. Unravelling the structure of these objects provides new insights into network dynamics and is the principal motivation of this thesis.

Traditional mathematical methods and analysis cannot in general provide a description of the long term behaviour of discrete dynamical networks except for the simplest special cases. Their study relies on the cycle of theory and experiment by computer simulation.

## 1.2. Random Boolean Networks

On-off systems as described above have been named "random Boolean networks" by Stuart Kauffman (1969) after George Boole, the pioneer of symbolic logic. They have their roots in the automata theory of Turing and von Neuman (Burks 1970) and in Ross Ashby's work on dynamical systems and cybernetics (Ashby 1956, 1960). More complicated networks with a range of cell states exceeding two also conform to the dynamics described, but this thesis looks mainly at simple, sparsely connected binary networks. Random Boolean Networks (RBN) may be applied to model parallel processing systems with non-local connections typical in biology, for example in genomic regulatory networks, in microbial ecosystems, in coevolution on NK fitness landscapes (Kauffman 1969-93), in immune networks (for example Weisbuch *et al.*), and in neural networks (Wuensche 1993c, 1994b). They have close affinities with "weightless" artificial neural network models (Alexander *et al.* 1984).

## 1.3. Cellular Automata

In the special case where the connections in a network are made according to a regular geometry with a universal rule, that is with the same rule and coupling template everywhere, and connections to nearest neighbours or to a larger *local* "neighbourhood", the network is called a *cellular automaton* (CA) with particular spatial dimension, geometry and boundary conditions.

CA provide perhaps the simplest systems that support emergent self-organisation. They are of central interest in understanding complexity, chaos, and emergent phenomena. CA have been extensively studied as models in physics, mathematics, computation and biology. They are examples of discrete dynamical systems where large scale complex pattern formation may emerge from disordered initial conditions as a result of just local interactions.

The study of CA also has its origins in the automata theory of Turing and von Neuman (Burks 1970). von Neuman is said to have invented the general notion of a cellular automaton (at the time also called a tessellation structure or iterative circuit computer) to model self-reproduction following a suggestion by Stanislaw Ulam. It cannot be a coincidence that this shortly followed von Neuman's invention of one of the first programmable digital computers in the late forties. His unfinished automaton or "universal computer-constructor", completed and simplified by Authur Burks (1970), comprised a Turing machine embedded in an infinite two-dimensional lattice with a neighbourhood

of 5 (cross shaped) and 29-states per cell. The machine constructs the configuration described on its tape in a vacant area of the lattice together with a copy of the tape attached. Subsequent CA work has shown that far simpler architectures are capable of both universal computation (Conway 1982, Lindgren and Nordahl, 1990) and self-reproduction (Codd 1986, Langton 1984). Wolfram showed that very simple one-dimensional binary CA may also exhibit complex behaviour (Wolfram 1983,1984a).

The upsurge of interest in CA has reflecting the availability of inexpensive computer power, and can be traced back to John Conway (1982) and his remarkable "game-of-life" illustrated in figure 1.1, and to Stephen Wolfram who made the first in depth investigation of 1d CA. Wolfram's ultimate goal was "*to abstract from the study of cellular automata general features of "self organising" behaviour and perhaps devise universal laws analogous to the laws of thermodynamics*" (Wolfram 1983).

Chris Langton developed his concept of Artificial Life on the basis of CA (Langton 1986). He and others proposed the idea of self-organisation emerging at a phase transition in rule-space, close to the onset of chaos (Langton 1990, Li *et al* 1990), with implication for the origin and evolution of life. The discrete *lattice gas automaton* was invented by Brosl Hasslacher and others, promising a general approach to solving partial differential equations and to parallel computation (Hasslacher 1987).

There are many variants of CA and related systems, and a large literature has accumulated. Notable variants include lattice gasses (Frisch *et al* 1986, Hasslacher 1987), spin-glasses (Sherrington 1990), probabilistic CA, and continuous value CA or CA with a large alphabet or value range. CA may be constructed with a longer memory where the "neighbourhood" extends into the past beyond the previous time-step. A neighbourhood that includes a cell two time-steps back allows the construction of reversible CA (Toffoli and Margolus 1987). Reversible CA have both an in-degree and out-degree of one, allowing the system to be run deterministically both forward and backward, thus are closer to classical physics with its symmetric arrow of time.

CA thus provide models in physics, in computation, and in the emergence of complex spatial pattern. A CA may be regarded as a discretised artificial universe with its own local physics (Langton 1990) or as a parallel processing computer (Wolfram 1984a,b) where the initial state represents data to be processed.

#### **1.4. Symmetries in Cellular Automata**

In (Wuensche and Lesser 1992a) it was shown that there are a number of general constraints to CA dynamics with periodic boundary conditions. These are reflected in the structure and topology of CA

attractor basins. They do not necessarily apply to random networks. The constraints relate to various symmetries and hierarchies within state space and rule space, summarised below.

1. Rotation symmetry (the number of repeating segments in the bit pattern) is conserved. In a transient, rotation symmetry cannot decrease over time; in an attractor cycle, rotation symmetry must remain constant. In *symmetric* rules the same principles apply to bilateral symmetry.
2. Rotation equivalent states (that differ only by any rotation of the periodic lattice) are embedded in equivalent behaviour. This has been described as shift invariance in (Wolfram 1986b), and results in basins of attraction or subtrees with identical topology, but rotated states. Symmetries within basins occur if a sequence of rotation equivalent states repeat in the attractor cycle; transient trees with identical topology, but rotated states, will be rooted on the repeats.
3. Rule-space can be divided into symmetry categories by transformations within rule tables.
4. Equivalence classes and rule cluster relationships exist in rule space due to transformations between rule tables.

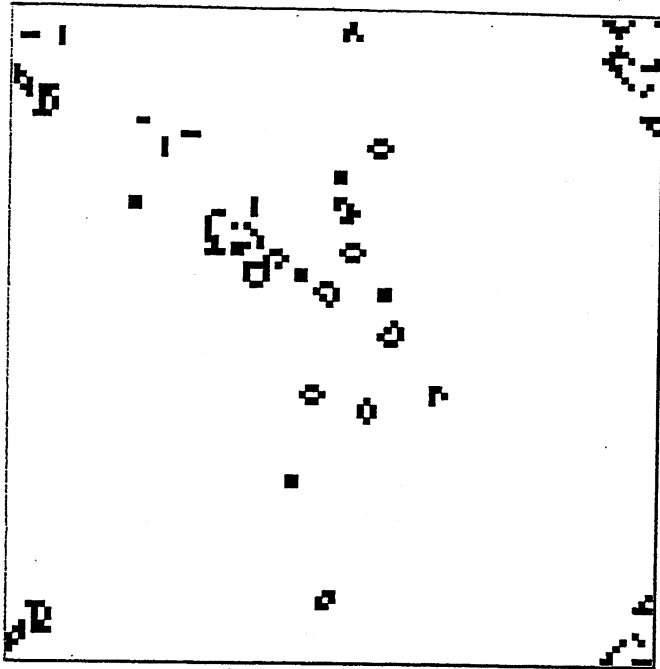
As well as for 1d CA, these constraints apply equivalently to 2d CA (e.g. an orthogonal grid with toroidal boundary condition), where the symmetries may occur in either or both of two directions. DDLab automatically recognises these symmetries to show only equivalent basins of attraction or subtrees, for example in figure 1.2 and 2.3. For 2d CA on a triangular grid, and for higher dimensions equivalent but more complicated symmetry constraints apply.

These constraints severely limit CA as flexible categorises of state-space, though this may be an advantage in some contexts such as pattern recognition. To achieve greater freedom for arbitrary categorisation it is necessary to relax the architectural definition of the system to allow arbitrary rules and connections - thus random Boolean networks.


CA and RBN provide contrasting notions of emergence and self-organisation. RBN provide models in biology because of their non-local connections and heterogeneous rules. The hierarchical categorisation implicit in their attractor basins, unconstrained by the symmetries found in CA, provide models in biology, for example memory in neural, gene\*, and immune networks. CA on the other hand are systems having a regular space and universal laws. They provide models of biological processes in so far as these involve pattern formation, but are principally models in physics on the basis of self-organisation within their spatial pattern.

---

\* The "memory" of a genomic regulatory network is its cell type or gene expression pattern (see chapter 5.11).




**Figure 1.1.** An example of John Conway's "game-of-life" on a  $100 \times 100$  grid with toroidal boundary conditions. The initial state is this

pattern  at the centre of an otherwise empty grid.

*Right:* The space time pattern shown as an isometric projection (looking up at a transparent box). About 300 time-steps are shown. Diagonal features are gliders.

*Top left:* A 2d snapshot..

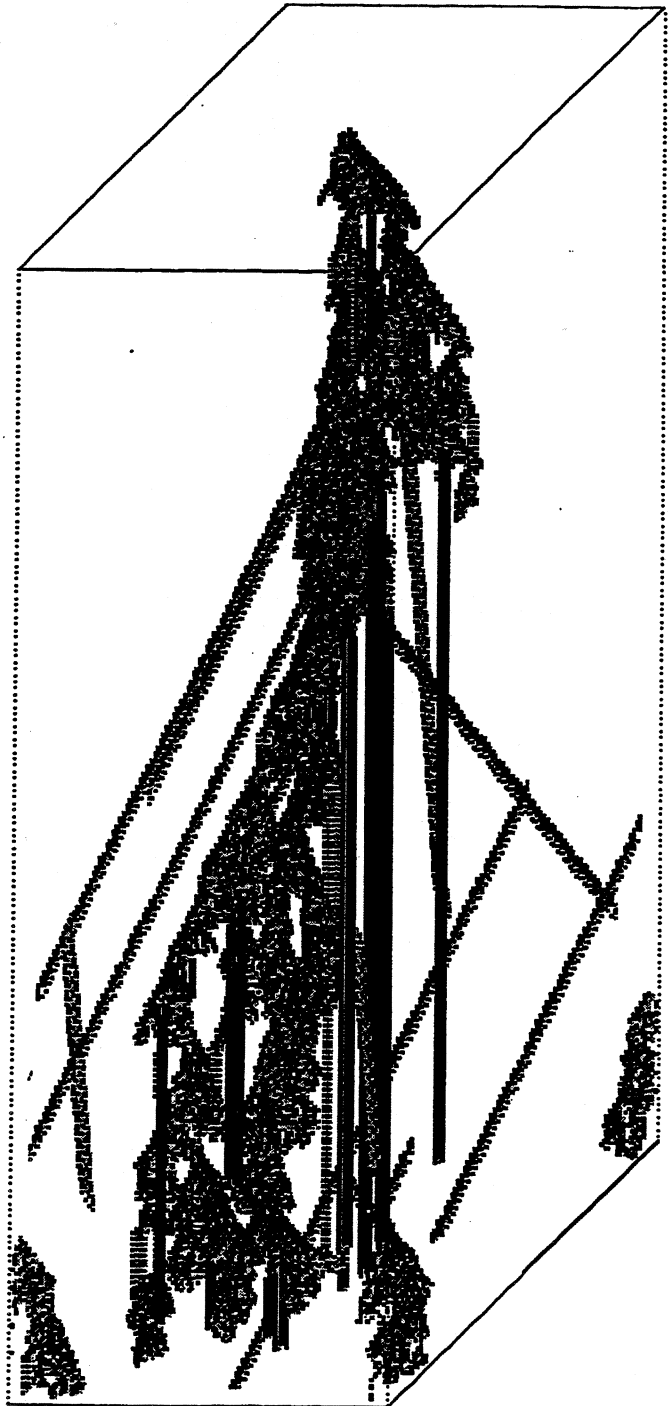
"Life" is a 2d binary CA. A cell updates according to the values of itself and its nearest neighbours relative to a  $3 \times 3$  block .

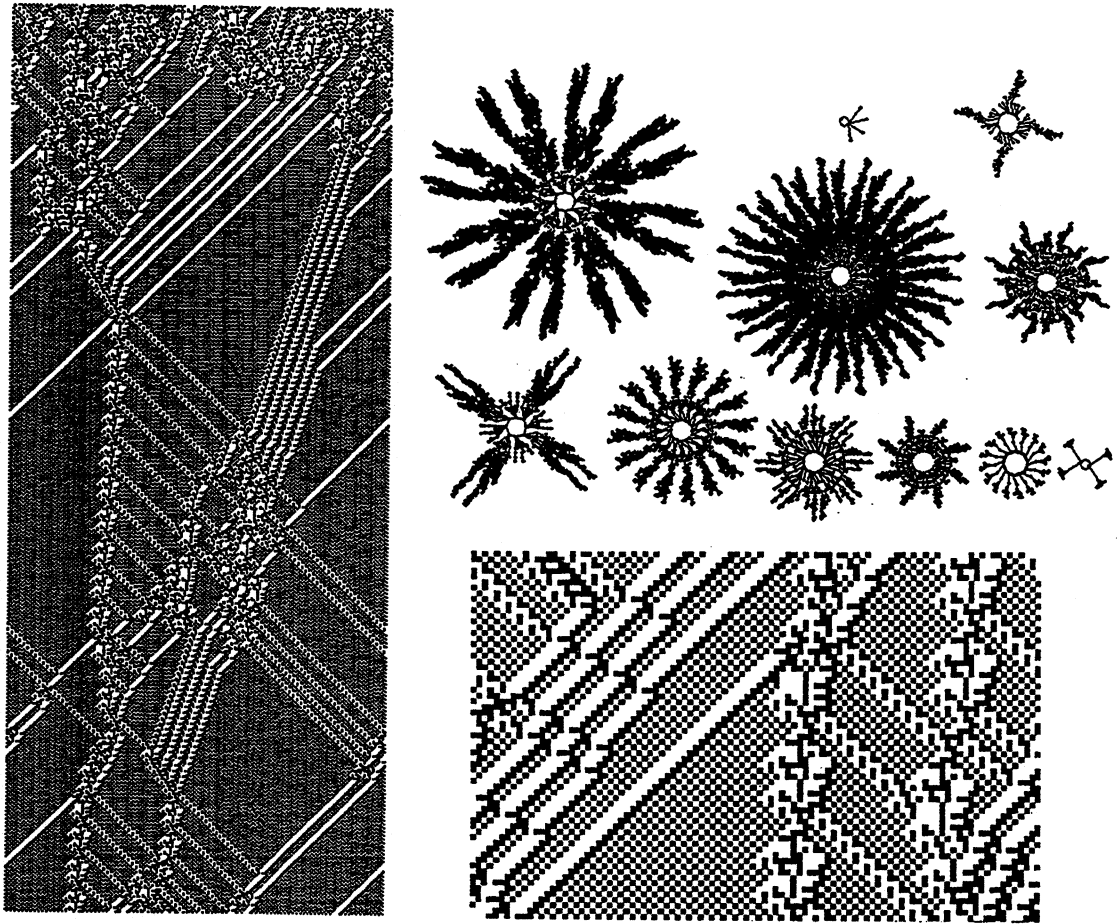
The "life" rule is as follows, where a black-cell is alive, and a white-cell is dead:

*Birth:* A dead cell comes to life if its neighbourhood had exactly three live cells.

*Death by overcrowding:* A live cell dies if its neighbourhood has 4 or more live cells apart from itself.

*Death by exposure:* A live cell dies if its neighbourhood has less than 2 live cells apart from itself.





**Figure 1.2.** An example of 1d,  $k=5$  CA rule with complex dynamics. The neighbourhood size  $k=5$ , the hex rule number\* is 36 0a 96 f9.

*Left.* the space-time pattern from a random initial state, system size  $n=200$  with periodic boundary conditions, 480 time-steps. Gliders (and a glider-gun) emerge against a checkerboard background after an initial sorting out phase.

*Top right.* The basin of attraction field for the same rule,  $n=16$ . The  $2^{16}$  states in state space are organised into 17 basins of attraction, only the 11 non-equivalent basins are shown.

The number of states in each basin, the number that are garden-of-Eden states (*g-of-E*), the attractor period, and the number of each type, is as follows (from left to right),

<u>states</u>	<u>g-of-E</u>	<u>period</u>	<u>types</u>
30928	14536	8	1
6	4	2	1
220	108	16	4
23808	11584	16	1
1136	528	16	2
1972	892	8	2
2064	992	16	1
568	232	16	1
448	224	16	2
144	64	32	1
26	20	1	1

Note the symmetries due to equivalent transient trees.

*Bottom right.* A detail of typical glider interactions.

\* The rule numbering conventions are described in chapter 4.2.

### 1.5. Emergence in Cellular Automata

Complex emergent behaviour in CA becomes apparent in the evolving space-time pattern, where self-sustaining configurations, particles or *gliders*, may emerge and interact. The system's behaviour can then in principle be described, and predictions made, at the higher level of gliders and their collision rules rather than at the lowest level of the CA's rule and neighbourhood geometry. Figures 1.1 and 1.2 show examples for 2d and 1d CA. These gliders are discrete analogues of Prigogine's *dissipative structures* (Prigogine and Stengers 1984), patterns in forced chemical reactions far from equilibrium as in the well known Belousov-Zhabotinski reaction, but a term now applied to non-linear effects producing emergent structures in any far from equilibrium continuous system, such as eddies in fluid flow or Bénard cells. Many of these phenomena are modelled by CA (Toffoli and Margolus 1987, Adamatzki 1994). However, in CA the process of formation, persistence and interaction of gliders and other dissipative structures can be traced at the lowest level of the system's basic components and their local interactions which are completely defined. This ability to see two levels of behaviour simultaneously, the underlying and emergent level, may lead to insights into the mechanics of self-organisation.

CA rules can be classified as ordered, complex and chaotic (Wolfram 1983). The small proportion of complex rules that support gliders are said to occur at a *phase transition* (Li *et al* 1990) in rule space, the so called "edge of chaos" (Langton 1990). Complex CA motivated the field of *Artificial Life* (Langton 1986,1989, Langton *et al.* 1991), and the long transients and glider dynamics typical of these rules provide metaphors for the emergence of life from the stuff of inanimate physics.

### 1.6. Cellular Automata Local and Global Measures

A method is described in chapter 4 for classifying CA rules by a measure of the variance of input-entropy over time. The frequency with which different neighbourhoods are looked up in the rule table (the frequency of blocks of size  $k$ , the neighbourhood size) is taken for a sample of trajectories over a moving window of time-steps. Low mean entropy of the frequency distribution indicates order, high mean entropy indicates chaos. In both cases the variance of the entropy over time is low. High variance is then a signature of large scale coherent pattern formation, typically made up of gliders interacting within periodic background domains, where the gliders act as domain boundaries. The method has allowed glider rules and related complex rules to be found automatically. A random sample of thousands of such rules sorted according to these measures have been assembled, though the source is virtually limitless. Glider dynamics in CA is of prime interest as an example of self-organisation and emergence in simple systems, where all aspects of the system can be fully defined.

The space of rules are sorted according to these local measures, and the measures are compared with global measures on the structure or topology of attractor basins and subtrees. It is shown that these global measures correlate with the local entropy measures, and also with the quality of dynamics as seen subjectively. A key measure of topology appears to be the characteristic in-degree, or bushiness, of subtrees in attractor basins. Ordered dynamics turns out to have relatively high in-degree with very short, bushy transient trees, rooted on very short attractor cycles. Chaos, on the other hand, is characterised by very long, sparsely branching transient trees rooted on very long attractor cycles. Measures of the bushiness of subtrees in attractor basins are captured by the density of states having no predecessors known as "garden-of-Eden" states, and more generally by the frequency distribution of in-degrees. In preliminary results, a histogram of the in-degree distribution shows contrasting profiles for order and chaos; for complex rules the distribution seems to follow a power law. These measures are taken on the whole basin of attraction field for small systems and just on sample sub-trees for large systems.

### 1.7. Cellular Automata parameters

Various rule parameters have been proposed that seek to predict CA behaviour based only on the structure of the lookup table (described in chapter 4.2), for example Langton's  $\lambda$  parameter which for binary CA is the fraction of 1s in the lookup table\* (Langton 1990), the equivalent idea of *internal homogeneity* introduced earlier by Walker (1966), the author's  $Z$  parameter (Wuensche and Lesser 1992a, Wuensche 1994a) depending on the distribution of 1s and 0s in the lookup table (described in chapter 4.12), and more recent parameters proposed by Zwick (1995). Parameters of this sort are interesting because they can be used in principle to tune a CA by selective bit flips though the order/chaos phase transition and provide another measure which may be correlated with measures on local and global dynamics. As yet the predictive powers of these parameters are only approximate to varying degrees. For binary 1d CA the  $Z$  parameter tracks behaviour more closely than  $\lambda$ . It derives from the algorithm for generating pre-images and gives an expectation of the bushiness of subtrees and thus of the convergence of trajectories, a sort of inverse Liapouov exponent. Plots of  $Z$  against garden-of-Eden density for a large sample of rules show a marked correlation (Wuensche 1994a).

### 1.8. Emergence in Random Boolean Networks

In RBN the emergence of coherent spatial patterns like those in CA is unlikely or impossible because of the irregular connectivity and non-local rules. As illustrated in figure 5.6, it can be shown

---

\* The  $\lambda$  parameter applies to CA with any value range including binary. If some arbitrary value (usually 0) is designated as *quiescent*, then  $\lambda$  is defined as the fraction of non-quiescent values in the lookup table.



that a system of interacting gliders is gradually degraded when regular CA connections are progressively randomised (Wuensche 1993a). What, then, are the emergent properties in a random Boolean network.

I will argue that there is an emergent property fundamental in biology and other complex systems. This property is memory, a network's ability to dynamically categorise its state-space. The emergent structure that embodies memory is the network's basin of attraction field, representing all possible trajectories through state-space. As will be shown, categorisation occurs *far from equilibrium* as well as at attractors, creating a complex hierarchy of content addressable memory represented by separate attractors within the basin of attraction field, and also by the roots of subtrees within basins of attraction.

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

In RBN as applied to gene regulatory networks (Kauffman 1969, Somogyi and Sniegoski 1996), the notion of a phase transition between order and chaos is just this balance. Tuning various parameters in the network's wiring/rule setup, in particular the degree of "canalisation" at the level of rules (Kauffman 1984, Harris *et al.* 1996) moves behaviour across the transition.

Work in collaboration with Stuart Kauffman and Steve Harris (Harris *et al.* 1996) and with Roland Somogyi (Somogyi and Sniegoski 1996, Somogyi *et al.* 1996) applies RBN as models of parallel processing genomic regulatory networks and develops Kauffman's long established ideas in this area (Kauffman 1969,1984). Particular settings of network parameters allow the dynamics to settle into a set of stable active (also inactive and dynamic) gene expression patterns which represent the various cell types, or cell pathology such as cancer. These are interpreted as the set of attractors (or basins) in the basin of attraction field of the genomic network, or as "frozen skeletons", trajectories outside the attractor cycle itself but where the pattern of active genes has largely stabilised.

### **1.9. Random Boolean Networks Local and Global measures**

Various order/chaos measures can characterise RBN dynamics and identify the phase transition (Kauffman 1969). The Derrida plot (Derrida and Stauffer 1986) is analogous to the Liapouov exponent in continuous dynamics and measures the divergence of trajectories based on Hamming distance. Further measures are available such as the frequency of sizes of damage spread resulting a

single bit perturbation of the network state, the percolation of "frozen" i.e. unchanging regions, input-entropy of single genes taken on the frequency of input patterns over time. Measures on the topology of attractor basins and subtrees such as their in-degree distribution can be made as for CA. The precise structure of attractor basins is also of interest as it reflects the stability of cell types to perturbation and allows methods of extracting genetic networks architecture from biological data (Somogyi *et al* 1996), an application of the *inverse problem* (see below). For larger RBN, methods are available for extracting data on transient/frozen skeleton/attractor characteristics such as the number and size of different attractor basins or skeleton sub-trees, the length of attractor cycles and of attractor/skeleton transients.

#### **1.10. Random Boolean Network Parameters**

Various network parameter settings may be tuned to move RBN dynamics across the order/chaos phase transition as defined by these measures. The  $P$  parameter (described in chapter 5.11.10) is equivalent to the  $\lambda$  parameter in CA. Another parameter which seems to be closer to observed cell biology is the degree of "canalisation" at the level of rules, though the characteristics of network connections also play an important role. Canalisation occurs when a particular input (0 or 1) on a connection determines a gene's behaviour irrespective of its other inputs. That connection is then said to be canalising. The canalisation setting on RBN corresponding to the phase transition is very far from random expectation. The sparse data that is just beginning to be assembled by Steve Harris (Harris *et al.* 1996) on real genomic regulatory networks seems to correspond to these settings, indicating that genomic regulatory networks in biological systems evolve to remain delicately posed close to the phase transition.

#### **1.11. Learning and the inverse problem**

Both Random networks and CA reliably categorise states within transient subtrees as well as at attractors (Wuensche 1993a, 1994b). However, RBN have a vastly greater parameter space than CA so that their potential for emergent memory is vastly more flexible. Interactions within a system of many interconnected networks, a network of networks, may lead to yet higher levels of emergence.

A network's basin of attraction field provides an overall picture of its hierarchical categorisation of state-space. Learning algorithms to modify or sculpt the resulting basin of attraction field were presented in (Wuensche 1993b). Alternative methods for solving the inverse problem, finding the RBN architecture that will satisfy a complete or partial set of predetermined transitions, have recently been formulated independently by Manor Askenazi (1996) and John Myers (1996).

## Chapter 2

### Basins of Attraction

#### 2.1. Introduction

In Ross Ashby's "An Introduction to Cybernetics" (Ashby 1956), he defines determinate machines or dynamical systems as corresponding to a closed, single valued, transformation. The dynamics can be represented as a "kinematic graph". The separate regions of the kinematic graph are the graph's basins of attraction.

Attractor basins of discrete dynamical networks have been mysterious objects hidden from view because they were seemingly computationally intractable (Wolfram 1986); however, methods to compute and reconstruct these objects have been invented by the author, making them readily accessible by computer generated graphics (Wuensche and Lesser 1992a, Wuensche 1992b-1996). Basins of attraction are possibly a key to understanding emergent behaviour in both CA and RBN.

A global perspective on a *continuous* dynamical system is provided by its vector field - the field of flow imposed on phase-space by a differential equation. This concept can be traced back to Lord Rayleigh amongst others and was developed by Henri Poincaré, founder of dynamical systems theory. The vector field is described by its phase portrait. A set of attractors, be they fixed point, limit cycles or chaotic, attract various regions of phase space. The phase portrait represents the basin of attraction field. Analogous concepts apply to discrete systems, but as there are no continuous coordinates of motion\*, thus no continuous flow or vector field, a very different kind of phase portrait is required, a graph linking state transitions.

CA and RBN are both examples of discrete deterministic dynamical networks made up from many simple components acting in parallel. The dynamics is driven by the iteration of a constant global updating procedure (the *transition function*) resulting in a succession of global states, the network's *trajectory*. Given a noise free, deterministic transition function within an autonomous system (cut off from outside influence), any pattern of activation (the global state) imposed on the network will seed a determined trajectory, though it may be unpredictable (Wolfram 1983, 1984a). In fact the system may be regarded as semi-autonomous, in the sense that a global initial state must be imposed or perturbed from outside to set the system going along a new trajectory. To be of interest, the system also needs a channel to communicate its internal state to the outside.

---

\* Continuity in terms of Hamming distance may be provided by representing trajectories on the edges of an  $n$ -dimensional hypercube where  $n$  is the size of the network (e.g. Lewis and Glass 1992), however this presents difficulties of visualisation for  $n > 3$ .

In discrete dynamical networks, a global state (or *image*) of the network may have any number (including zero) of immediate predecessors (called *pre-images*) converging onto it. Trajectories can therefore merge outside the attractor. In continuous systems they can only come arbitrarily close. In a finite network of size  $n$  and value range  $v$  there are  $v^n$  global states, thus  $2^n$  in binary networks. Any path must inevitably encounter a repeat. When this occurs the system is locked into a state cycle (the *attractor*). Many merging trajectories typically exist leading to the same attractor. The set of all such trajectories, including the attractor itself, make up a basin of attraction, whose topology is typically branching transient trees rooted on the attractor cycle, though this "cycle" may be a stable point, an attractor cycle with a period of one.

Separate basins of attraction typically exist within state space. A transition function will, in a sense, crystallise state-space into a set of basins, the basin of attraction *field*. The field may be represented as a directed graph showing the state at each node. As such it is an exact representation of the network's entire repertoire of behaviour, showing all possible trajectories .

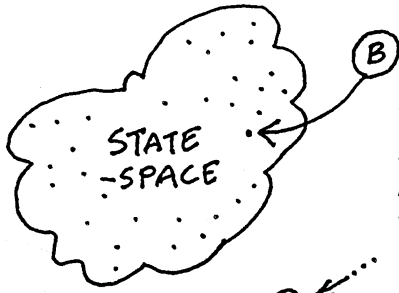
Attractor basins are mathematical objects in space-time that link the system's global states according to their dynamical transitions. Access to these objects opens up a new area of phenomenology, a global perspective on the dynamics of discrete networks, in addition to the study of space-time patterns alone. The relative length and "bushiness" of trees, and other features of attractor basin topology, reflect space-time phenomena.

Figure 2.1. summarises the concepts. Figures 2.2-2.5 provide an example, showing the basin of attraction field and individual basins of Conway's "game-of-life" on a  $4 \times 4$  (toroidal) grid, thus a state space of  $2^{16}$ . Figure 2.2 shows the entire basin of attraction field comprising 345 basins. Figure 2.3 shows only the 34 non equivalent basins (see chapter 1.4) with one attractor state drawn as a  $4 \times 4$  pattern. Figure 2.4 shows two basins in detail where all the nodes are drawn as  $4 \times 4$  patterns. See also figure 1.1 for a description of "life".

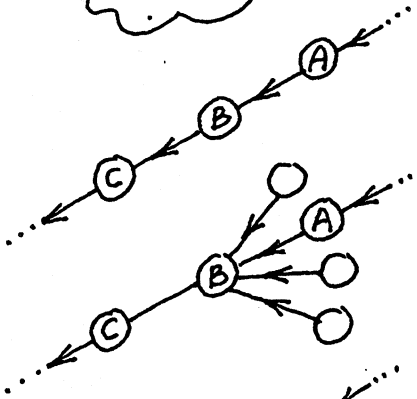
## 2.2. Computing Pre-images

The capability of constructing attractor basins depends on reverse algorithms invented by the author for directly computing the pre-images of network states. This allows the network's dynamics in effect to be run *backwards* in time; backward trajectories will, as a rule, diverge. These algorithms open up a window on the precise structure of the basins of attraction of discrete dynamical systems.

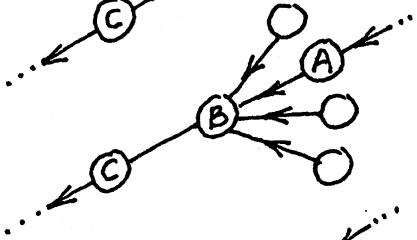
Different reverse algorithms apply to networks with different qualities of connectivity. The most computationally efficient algorithm applies to 1d networks where each site has the same connection template to a local neighbourhood. This algorithm thus applies to 1d CA or variants of 1d CA where the rules may be different at each site. An alternative algorithm is required for



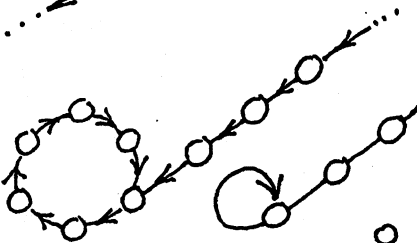
For a network size  $n$   $\leftarrow n \rightarrow$   
 an example of one of its states,  $B = 1010\dots\dots 0110$   
 State-space is made up of all  $2^n$  states,  
 the space of all possible bitstrings or patterns.



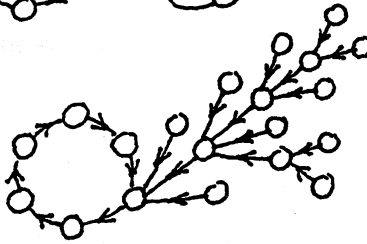
Part of a particular trajectory in state-space,  
 where  $C$  is a successor of  $B$ , and  $A$  is a predecessor (pre-image)  
 of  $B$ , according to the dynamics of the network.



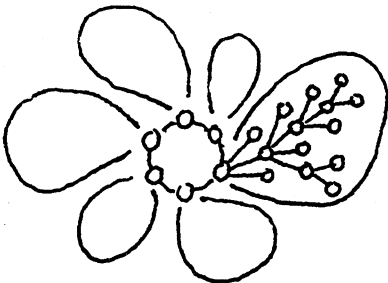
The state  $B$  may have other pre-images besides  $A$ , and  
 the pre-image states may have their own pre-images or none.  
 States without pre-images are known as *garden-of-Eden* states



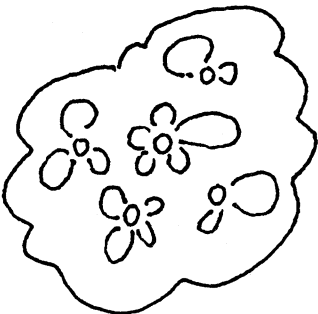
Any trajectory must sooner or later encounter a state that  
 occurred previously - it has entered an *attractor cycle*.  
 The trajectory leading to the attractor is known as a *transient*.  
 The *period* of the attractor is the number of states in its cycle,  
 which may be only one - this is known as 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 at all, just the attractor, a seeming  
 inconsistency in terminology but so be it.



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 2.1. State space and attractor basins.

networks with a non-local, arbitrary pattern of connections such as RBN, including RBN with mixed connectivity  $k$ . The non-local algorithm also applies to CA including 2d CA or CA of any dimensionality because these are just a sub-class of RBN.

These methods are in general orders magnitude faster than the brute force method, constructing an exhaustive map resulting from network dynamics. The exhaustive method rapidly becomes computationally intractable with increasing network size so is limited to small systems, but applies to all network types and also allows the attractor basins of random maps to be constructed as described in chapter 3. These three independent methods together form a valuable reality check on the correctness of the computed pre-images and attractor basins.

The reverse algorithm for 1d CA was introduced in the author's book (Wuensche and Lesser 1992a). The reverse algorithm for RBN was introduced in (Wuensche 1992b-1993b). The algorithms are described in chapters 4 and 5. These methods are applied in Discrete Dynamics Lab (DDLab), interactive graphics software available on the Internet (Wuensche 1996). A overview of DDLab is given in appendix 6.

Prior to the introduction of these reverse algorithms, there were very few examples of attractor basins to allow a study for their overall topology and structure. The few examples that were available were for small 1d CA or the exceptional cases of CA rules amenable to algebraic analysis (Wolfram and Peck 1986, Pitsianis *et al.* 1989, Martin *et al.* 1984). In the case of random Boolean networks there were no examples at all.

Previous investigations of attractor basin structure has been done by statistical methods (for example Wolfram 1984a,1985, Gutowitz 1991, Walker and Ashby 1996, Walker 1971-1987, Kauffman 1969-1993). In these methods, also now implemented in DDLab, many forward trajectories are run from random initial states looking for a repeat in the network pattern to identifying 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.

### **2.3. Constructing Attractor Basins**

Methods for constructing subtrees, single basins and the basin of attraction field as implemented in DDLab are described below.

To construct a basin of attraction containing a particular state or "seed", the network is iterated forward from the seed state until a repeat is encountered in the trajectory. The number of steps to achieve this is known as the disclosure length, the transient plus the attractor period (Walker and Aadryan 1971). This identifies the attractor cycle as the sequence of states from the state that was repeated.

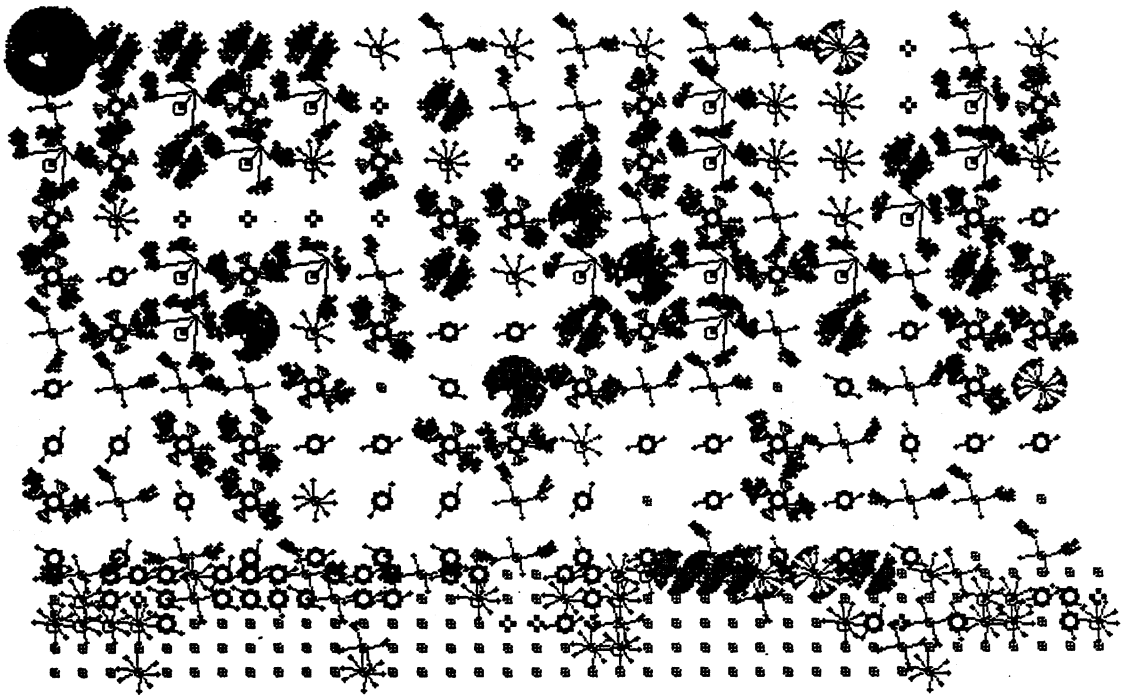


Figure 2.2. The entire basin of attraction field of the "game-of-life" on a  $4 \times 4$  grid comprising 345 basins. (overlaps between basins result from the graphics layout and should be ignored).

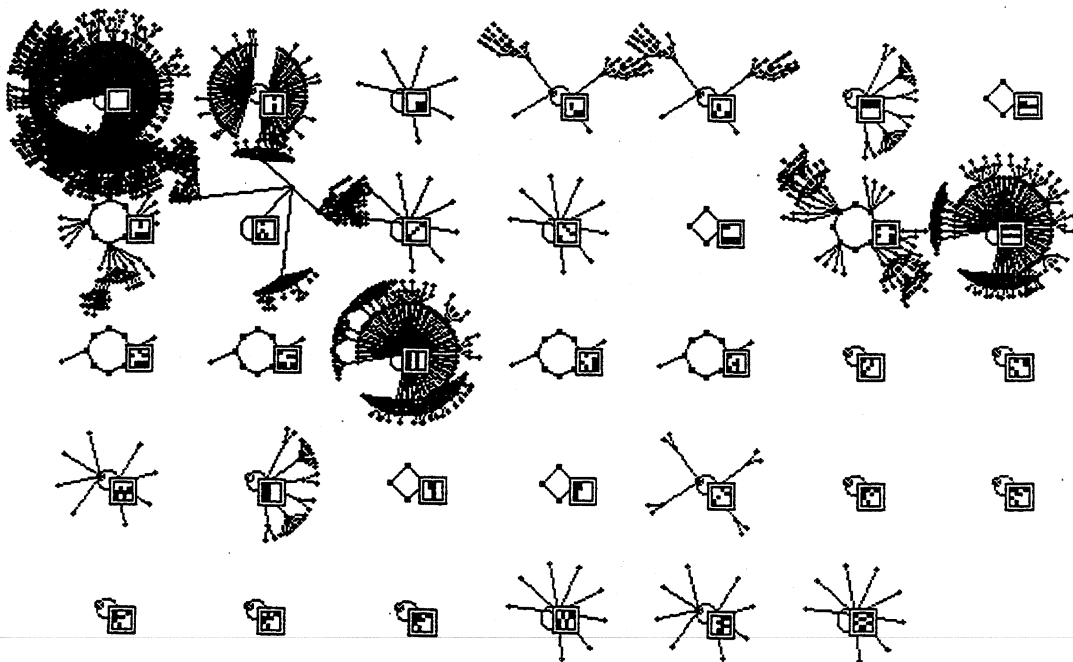
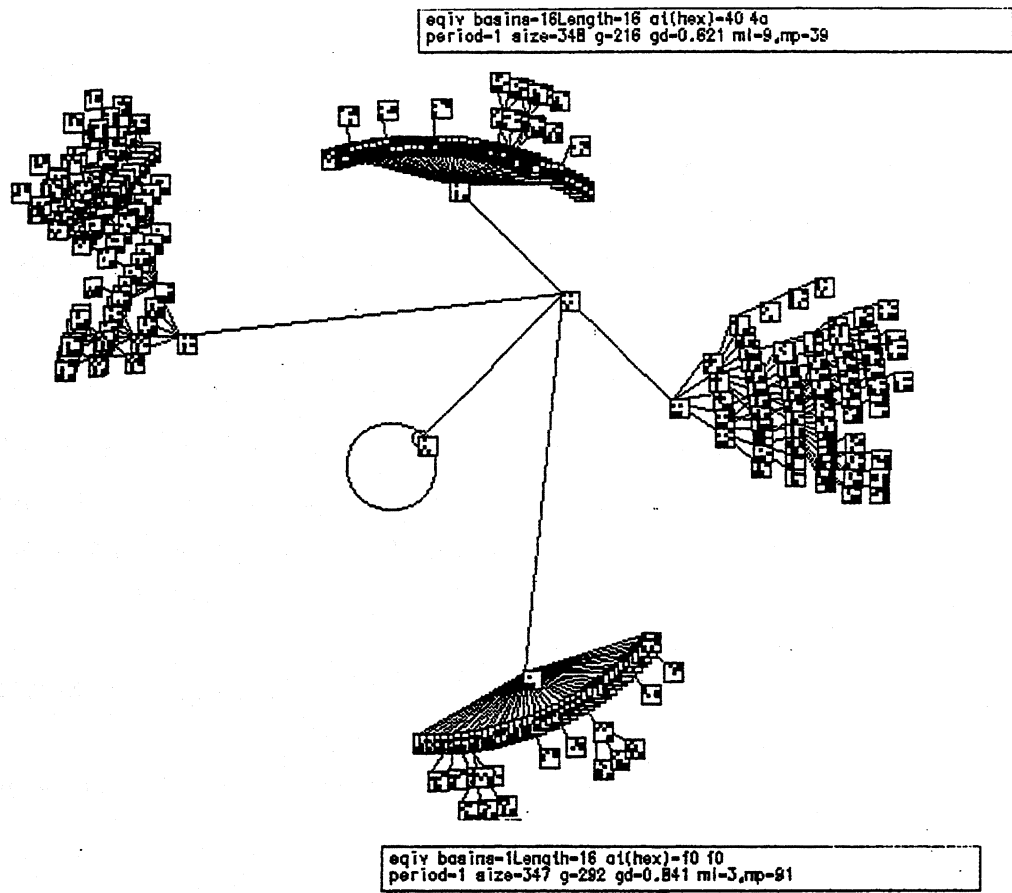


Figure 2.3. The basin of attraction field of the "game-of-life" on a  $4 \times 4$  grid showing only the 34 non equivalent basins (see chapter 1.4). One attractor state in each basin is drawn as a  $4 \times 4$  pattern.



**Figure 2.4.**

Two basins of attraction of the "game-of-life" on a 4x4 grid shown in greater detail with all the nodes drawn as 4x4 patterns (with inevitable overlaps). Both basins have attractor period one.



Once the attractor cycle is known (and drawn as a circle of nodes), 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 (and drawn as the pre-image *fan*), but the pre-image lying on the attractor cycle itself is deliberately excluded to prevent endlessly tracing pre-images "backwards" around the attractor cycle. The reverse algorithm is then re-applied repeatedly, to the pre-images of pre-images, until all the garden-of-Eden states have been reached. These are the leaves of the transient tree having no pre-images. In this way the transient tree is completely specified (and simultaneously drawn).

In a similar way just a subtree or a fragment of a subtree may be constructed rooted on a seed state. However, because a seed chosen at random is very likely to be a garden-of-Eden state, to construct a subtree it is usually necessary to run the network forward by at least one step from the random state and use the state reached as the subtree seed. Running forward by more steps will reach a seed deeper in the subtree so allow a larger subtree to be constructed, though running too far forward may reach the attractor, in which case the subtree procedure will reconstruct the entire basin.

For CA the construction of transient trees and subtrees is simplified by taking advantage of shift invariance as described in chapter 1.4. There are global states that differ only by a rotation of the circular array. Such rotation equivalent states must have equivalent pre-images, rotated by the same amount, and must occupy equivalent positions in the same or an equivalent basin. If the pre-images of a given state have been computed, the pre-images of its rotation equivalents are known, and by extension so is the entire transient tree or subtree, which need not be re-computed. If rotation equivalent states belong to separate basins, the basins will be equivalent, so only one example needs be constructed.

Constructing the entire basin of attraction field requires setting up a tick-off table, an array with  $2^n$  bits where each bit represents a state in state space. The first basin of attraction making up the field is seeded with the state represented by the first bit in the array, say all 0s. As the basin is constructed all states belonging to it (and also to rotation equivalents in the case of CA) are ticked off in the tick-off table. When the basin is complete, the next available state, that has not been ticked off, is used as the seed for the next basin, and the procedure is repeated until no available states remain in the tick-off table. This method avoids basins being duplicated.

#### **2.4 Portraying Attractor Basins**

Attractor basins are portrayed as computer diagrams in the same graphic format as presented in (Wuensche and Lesser 1996). Global states are represented by nodes, by a bitstring pattern in 1d or 2d, or as the decimal or hex value of the state. The nodes are linked by directed arcs. Each node will have zero or more incoming arcs from its pre-image nodes, but because the system is deterministic,

exactly one outgoing arc (one *out-degree*). Nodes with no pre-images have no incoming arcs, and represent garden-of-Eden states. The number of incoming arcs to a node is its *in-degree*.

Figure 5.1 shows a typical basin of attraction of a random Boolean network (it is part of the basin of attraction field shown in figure 5.2). Figure 4.16 shows the basin of attraction field of a CA. Many symmetries are evident, a major difference between the topologies of the two systems.

In the graphic\* convention for drawing attractor basins (described in detail in Wuensche and Lesser 1992), the length of transition arcs decreases with distance away from the attractor, and the diameter of the attractor cycle asymptotically approaches an upper limit with increasing period. The forward direction of transitions is inward from garden-of-Eden states to the attractor, which is the only closed loop in the basin, and then clockwise around the attractor cycle.

Typically, the vast majority of states in a basin of attraction lie on transient trees outside the attractor cycle, and the vast majority of these states are garden-of-Eden states. A transient tree is the set of all paths from garden-of-Eden states leading to a particular state on the attractor cycle. A transient sub-tree is the set of all paths from garden-of-Eden states leading to a state within a transient tree.

---

\* The graphic convention has been devised to give as clear an impression of attractor basins as possible, but it should be remembered that the essential information is how states are connected, not the particular appearance of the basin images.

## Chapter 3

### Attractor Basins of Random Maps

#### 3.1. Introduction.

Discrete Dynamical Networks, and in particular their attractor basins, can be put into the wider context of random graph theory. This branch of mathematics combines graph and probability theory to predict the structures that arise when vertices are randomly linked by directed or undirected edges, usually applying some constraints to the edge architecture. For example, graph theory has been applied to identify *neutral* folding networks in idealised RNA or protein sequences space (Reidys 1996).

The structures that arise in a random graph are precisely defined in the terminology of random graph theory. To give just the flavour they include the key notions; *path* - a sequence of vertices linked by directed edges, a *cycle* is a closed path, a *connected* graph or subgraph - with any two vertices connected by a path, a *tree* - a connected graph without cycles, and *components* - the maximal connected subsets of vertices. Clearly these ideas directly relate to the attractor basins of discrete dynamical network made up of trajectories, subtrees, and attractor cycles. Random graph theory, and in particular "random mappings" where the out degree from each vertex is exactly one, could provide a powerful mathematical framework for the global dynamics of discrete networks.

Using DDLab the structures found in random maps\* can be computed numerically and portrayed just as the attractor basins of CA or RBN. In work in progress in collaboration with Christian Reidys (Reidys and Wuensche 1997), Reidys has used random graph theory to formulate expressions for the expectation of cycles of various lengths in random maps, including random maps with biases based on Hamming distance. These are compared with the basin of attraction fields of example mappings with the same biases generated by DDLab. Some results of this work in progress is presented below.

A discrete dynamical network is equivalent to a mapping with particular biases imposed by the transition function. A long term aim is to apply random graph theory to formulate expressions for the expectation of the various features of the basin of attraction fields of discrete dynamical networks. This especially applies to RBN where the mapping biases should be more mathematically tractable than in the case of CA.

---

\* A random map is equivalent to a RBN where  $k=n$  (neighbourhood = network size). The "random map model" has received considerable attention in the literature (e.g. Kauffman 1996, 1971, 1993, Wolfram 1983, 1984a, Gelfant and Walker 1984, Coste and Henon 1986, Derrida and Flyberg 1987, Derrida and Bessis 1988).

### 3.2. Random Graphs.

A random graph is constructed by taking a set of  $n$  vertices linked randomly by  $k$  edges. What then is the expectation of sizes and internal structure of separate connected components of the graph? Put another way, randomly connect  $n$  buttons with  $k$  threads and lift out any button. Other buttons that happen to be threaded directly or indirectly to that button will also lift out as a separate component with a structure consisting of loops and trailing ends. Graph theory combined with probability theory provides tools for calculating the expectation and distribution of the various general characteristics that describe the graph.

A classical result in random graph theory is the emergence of a *giant component* (Erdos and Renyi 1960). When  $k$  is small compared with  $n$  there will be many small separate components, but as more edges are added to the graph, at the threshold value of  $k=n$ , a giant component suddenly arises linking together almost all vertices. This is a phase transition related to the number of edges  $k$ , reminiscent of phase transitions between order and chaos in discrete dynamical networks.

### 3.3. Random Maps.

A special case of random graphs are random *directed* graphs with out-degree one, also known as *random maps*, where edges have a direction and each vertex has exactly one outgoing edge but an arbitrary number of incoming edges (including zero).

The structures found in random maps correspond to the topology found 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 structure or cycle. Note that the term *topology* is used here to describe the conformation of graphs or attractor basins, how vertices are linked by edges, independent of the appearance of the graphs drawn according to some graphic convention, and should not be confused with the standard use of the term *topology* in mathematics.

To construct a mapping of the Boolean hypercube of sequences of length  $n$ , (i.e. for a set  $Q_2^n$  of size  $2^n$  comprising all binary strings of length  $n$ ), a mapping from  $Q_2^n \rightarrow Q_2^n$ , to each element  $V_i$  of the set  $Q_2^n$ , independently assign one successor (or *image*)  $V_*$  also belonging to  $Q_2^n$ , chosen at random (or with some bias). The mapping is represented below as  $2^n$  pairs of elements (*states* in network terminology), where each image  $V_*$  represents a possibly different member of the set  $Q_2^n$ .

$V_{2^n-1}$	$V_{2^n-2}$	....	$V_i$ ....	$V_2$	$V_1$	$V_0$	all elements of the set $Q_2^n$
↓	↓		↓	↓	↓	↓	successors, <i>image</i> elements
$V_*$	$V_*$	....	$V_*$ ....	$V_*$	$V_*$	$V_*$	chosen at random (or with some bias)

The list of image elements is likely to contain repeats, and if so some other elements of  $Q_2^n$  must be missing from the image list. Transitions to some arbitrary element  $V_x$  may thus be *one*→*one* or *many*→*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.

We will consider the probability spaces of mappings of the Boolean hypercube with  $2^n$  vertices  $f:Q_2^n \rightarrow Q_2^n$  with various measures or biases. The number of all possible mappings is  $(2^n)^{(2^n)}$ , so an unbiased or uniform measure on each mapping is  $\mu(f) = 1/(2^n)^{(2^n)}$ . Here all mappings are equally probable. Several classical results (Bollobás 1985) on this probability space are,

$$\begin{aligned}
 P(G_f \text{ is connected}) &\approx \left(\frac{\pi}{2}\right)^{1/2} \times (2^n)^{-1/2} && \text{probability of one big basin of attraction} \\
 E(\text{ number of cycles}) &\approx \log(2^n) && \text{expectation of the number of basins of attraction} \\
 E(\text{ cycles of length } r) &\approx \binom{2^n}{r} \times (r-1)! \times (2^n)^{-r} && \text{expectation of the average number of cycles of length } r
 \end{aligned}$$

We will now introduce a Hamming distance bias on the mapping so that a state  $V_i$  maps only to the set of states at a Hamming distance of  $d$  from itself (described as the surface of a *ball* with radius  $d$ ), with uniform probability. Mappings  $f:Q_2^n \rightarrow Q_2^n$  now have the constraint  $d_h(v, f(v))=d$ , where  $d_h$  is the Hamming distance. The number of different mappings is now,

$$|B_d|^{(2^n)} \quad \text{where } B_d \text{ is the ball in } Q_2^n \text{ with radius } d \text{ and } |B_d| = \binom{n}{d} \quad \text{where } \binom{n}{d} = \frac{n!}{(n-d)! \times d!}$$

Following some further reasoning (Reidys and Wuensche 1997) it turns out that an upper and lower bound can be derived for the expectation  $E(C_L)$  of the number of cycles of length  $L$  in a system of size  $n$  with Hamming bias  $d$ , as follows,

$$\begin{aligned}
 &\text{Lower Bounds} && \text{Upper Bounds} \\
 \frac{1}{L} \left( \frac{2^n \binom{n}{d}^{L/2}}{\binom{n}{d}^L} \right) &\text{ or } && \frac{1}{L} \left( \frac{2^n \binom{n}{d}^{(L-1)/2}}{\binom{n}{d}^L} \right) \leq E(C_L) \leq \frac{1}{L} \left( \frac{2^n}{\binom{n}{d}} \right) && \text{equation 3.3.1}
 \end{aligned}$$

The variance of  $E(C_L)$  can also be computed (Reidys and Wuensche 1997). It turns out that the smallest variance, thus best prediction, is for short cycles in large systems. For low Hamming distance bias  $d$  the lower bound is most significant. Further results predict that random mappings without bias typically produce large cycles whereas random mapping where the pre-image and image are separated by a small Hamming distance typically produce short cycles.

### 3.4. Basins of Attraction of Random Maps.

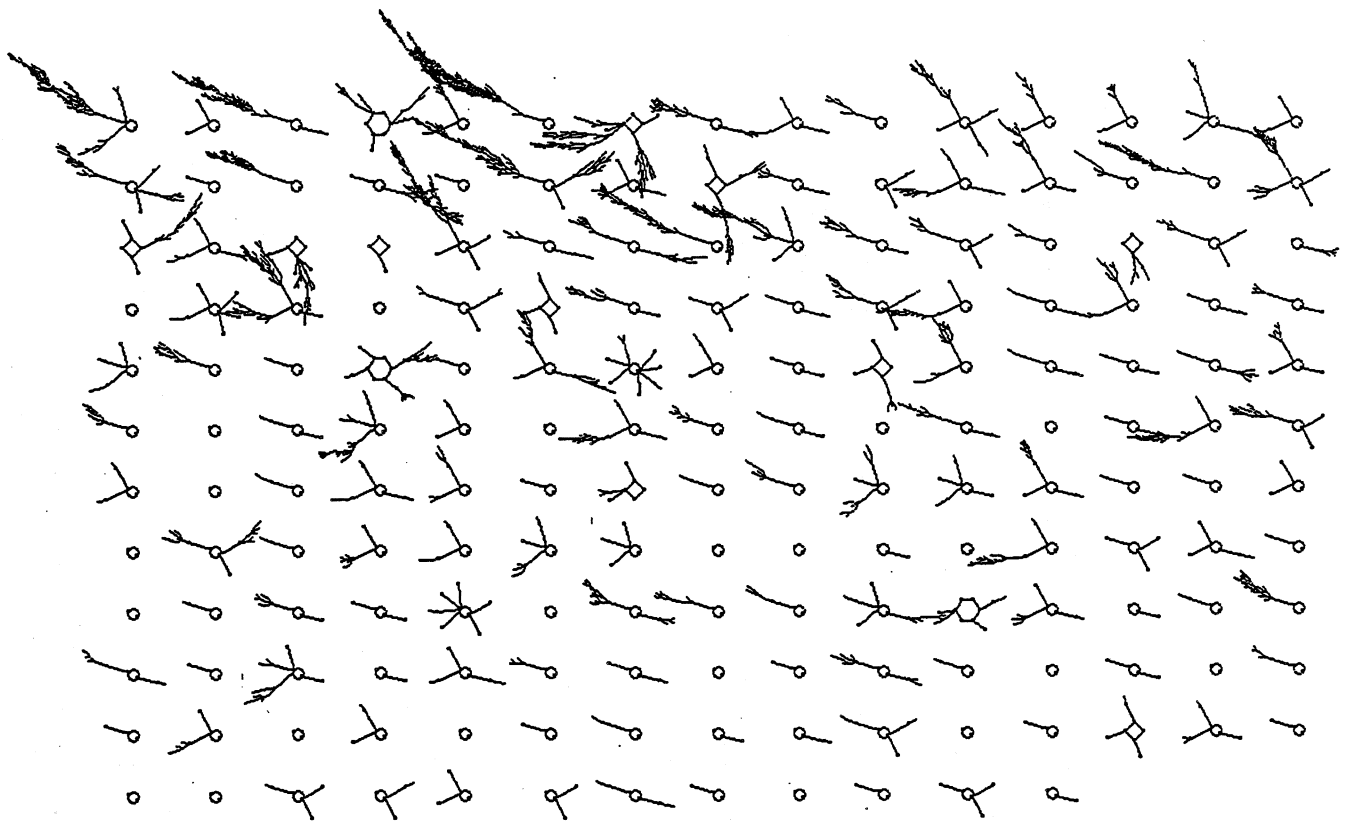
We can now test the expression for the expectation  $E(C_L)$  of the number of cycles of length  $L$ , by inserting parameter values in equation 3.3.1. These results can be compared with the basin of attraction fields of examples of mappings with the same parameters computed numerically.

A random map can be defined with the same Hamming biases as discussed above using DDLab, and the directed graph corresponding to the mapping, its basin of attraction field (or single basin or subtree), can be computed and drawn. The pre-images of an element (or state)  $V_x$  are found by scanning the mapping list of image elements, and noting the pre-image for every occurrence of  $V_x$ . If  $V_x$  does not appear in the list it is a garden-of-Eden state. The DDLab source code of the function that implements this algorithm is shown in appendix 6.1.

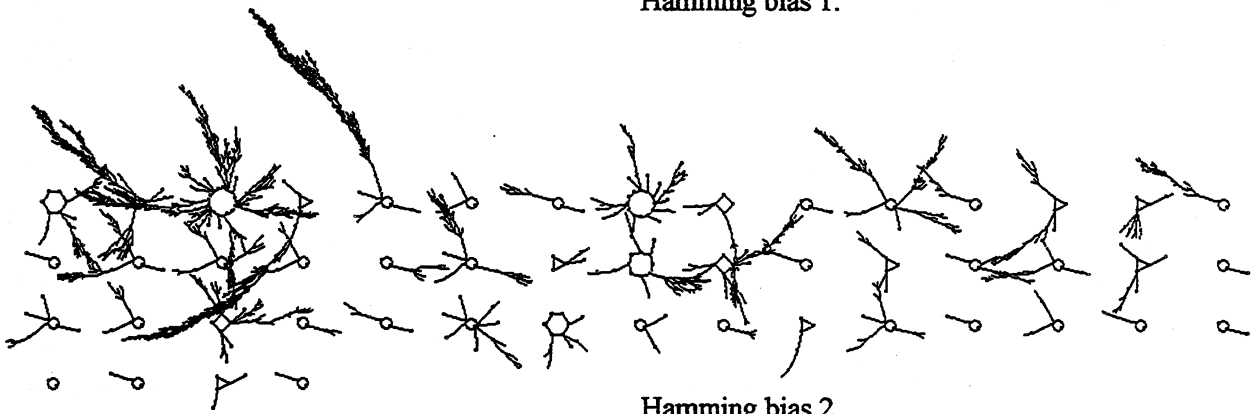
In the following examples, results from random graph theory (RGT), and by numerical simulation using DDLab, are compared. Samples of random maps are set up in DDLab with Hamming distance  $d$  of 1, 2 and 3. The length of the binary string  $n=12$ , giving a state space (number of vertices of the Boolean hypercube) of  $2^{12}=4096$ . DDLab generates the basin of attraction field for each map (figure 3.1 gives examples), and the frequencies of cycle lengths are counted. This is compared with the upper and lower bounds of the expectation  $E(C_L)$  of the number of small cycles of length  $L$  given by equation 3.3.1. Although this is a preliminary survey based on small sample sizes, the random graph theory seems to correctly predict the cycle length frequency. Comparative tables are given below.

Comparing analytical  $E$  and numerical (DDLab) cycle length frequencies for  $n=12$

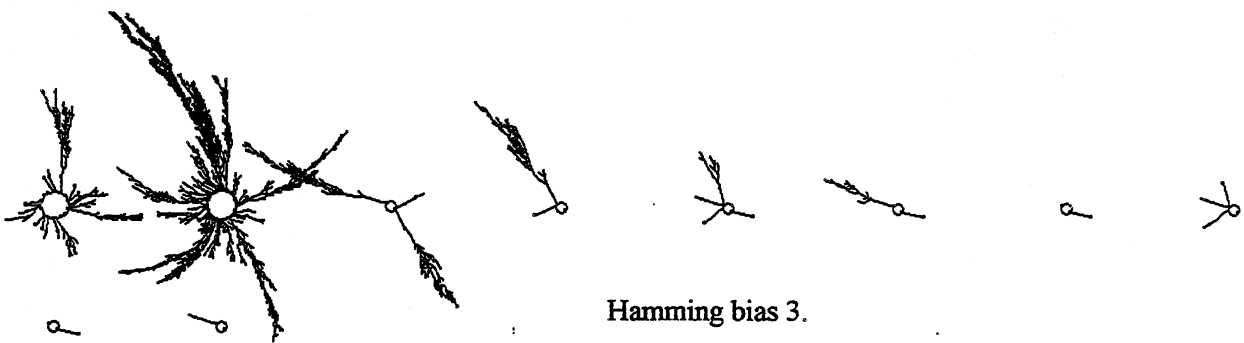
	$L$	$E(\text{lower-upper})$		<u>DDLab (sample size 20)</u>		note: for odd $d$ only even cycles are possible
				average	range	
<u><math>d=1</math></u>	$C_2$	170.6	170.6	172.4	160-191	
	$C_4$	7.1	21.8	7.1	5-10	
	$C_6$	0.39	58.9	1.0	0-3	
<u><math>d=2</math></u>	$C_2$	31.0	32.0	29.8	25-43	
	$C_3$	0.31	20.7	6.7	3-13	
	$C_4$	0.23	15.5	2.8	0-5	
<u><math>d=3</math></u>	$C_2$	9.3	9.3	9.8	4-16	
	$C_4$	0.02	4.6	1.25	0-3	



Hamming bias 1.

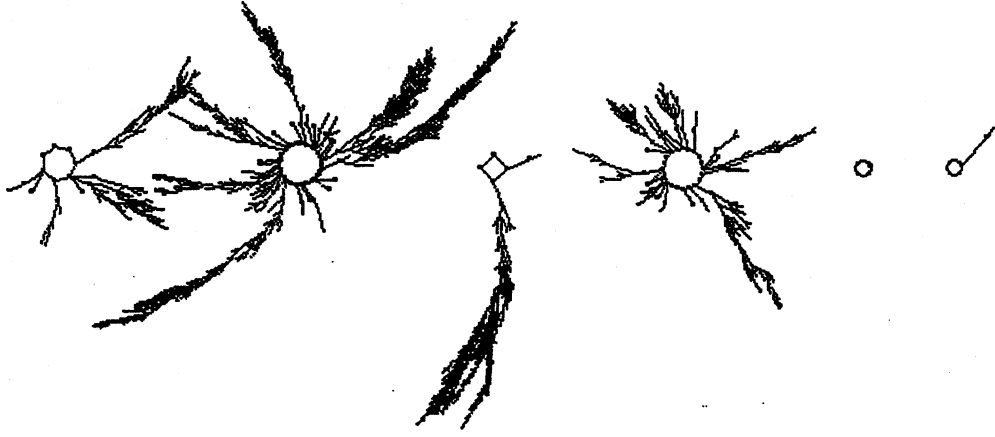


Hamming bias 2.



Hamming bias 3.

**Figure 3.1.** Examples of typical basin of attraction fields of the random maps with Hamming bias 1, 2 and 3. The length of the binary string,  $n=12$  (i.e.  $2^{12}=4096$  vertices). As the Hamming bias is relaxed more vertices are locked into transient trees and larger cycles



**Figure 3.2.** Example of a typical basin of attraction field of a random map with no bias for a binary string of length  $n=12$  (i.e.  $2^{12}=4096$  vertices).

### 3.5. RBN and CA in the context of Random Maps.

Examples of typical basin of attraction fields of the random maps with Hamming bias 1, 2 and 3 taken from the samples are shown in figures 3.1. As the Hamming bias increases, i.e. the Hamming constraints are relaxed, more vertices are locked into transient trees and larger cycles. An example without bias is shown in figure 3.2.

Random maps can be considered as 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, with its own exhaustive lookup table, can be assigned an arbitrary output for each network pattern.

RBN are usually applied as sparsely connected models where  $k \ll n$ , as in genomic regulatory networks or neural network models. Sparsely connected RBN may be regarded as a subclass of random maps just as CA are a subclass of RBN. The progression of subclasses is as follows,

$$CA \subset RBN \subset \text{Random Maps}$$

### 3.6. The Random Map Reverse Algorithm as applied to CA and RBN.

The "brute force" reverse algorithm for finding the pre-images of states in random maps can also be applied to discrete dynamical networks, CA and RBN, and is available in DDLab. The method depends on first constructing an exhaustive mapping  $Q_2^n \rightarrow Q_2^n$  as described in section 3.3 above. 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 created and held in a data structure. Once defined, 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. The C code function for the random map algorithm in DDLab is given in Appendix 6.1. The attractor basins are computed and drawn as described in chapter 2. This exhaustive method is restricted to small systems because the size of the mapping list grows exponentially with system size, and scanning the list for pre-images is slow compared to the direct reverse algorithms for CA and RBN, described in chapters 5 and 6.

### 3.7 Conclusion.

The results for cycle length frequencies predicted by the analytical methods described are tentatively confirmed by numerical simulations using DDLab. These preliminary results indicate that the structures in random maps, with biases based on Hamming distance, may be predicted by analytical methods. A random Boolean network is another type of bias on a mapping, though admittedly far more complex. In formulating expressions for expectations for particular structures in random maps with such complex biases, the interplay of theory and experiment will play an important role. Refining analytical methods by checking against numerical simulations has already proved useful in the preliminary work described here. Random graph theory may provide a powerful mathematical framework for discrete dynamical systems.

## Chapter 4

### Self-Organisation in One-D Cellular Automata\*

#### 4.1. Introduction

Conway's well known "game of life" is a two-dimensional cellular automata (CA) that supports coherent, periodic, space-time configurations that propagate and interact on a quiescent background (Conway 1982). An example of a space-time pattern of the game-of-life is shown in figure 1.1. The menagerie of configurations found have been grouped under various names such as gliders, glider-guns, eaters, blinkers etc. Interactions between glider streams can be contrived to create a system capable of universal computation (Conway 1982). Langton (1986) suggests that such *virtual state machines* may provide the "molecular" logic for artificial life embedded in CA.

In some 1d CA, analogous phenomena exist within a background that may be quiescent but is often periodic. These coherent structures are described variously as solitary waves, gliders, virtual automata, information structures, particle-like structures and domain boundaries or defects; for simplicity they are referred to here as gliders. The emergence of gliders would in principle allow the system to be described and predictions made at a higher level, on the basis of observed glider collision rules without reference to the underlying low level CA rules. Gliders may eject or absorb a regular glider stream, or spontaneously combine to form *compound* gliders, which then interact at yet higher levels of description. The process could unfold without limit in large enough systems. Glider dynamics in CA provide a stark instance of self-organisation in a simple system resulting from many local small scale parallel processes. This illustrates the concept of emergence, and one approach to the elusive notion of complexity.

Glider dynamics can be approached from a number of perspectives. It corresponds to Wolfram's class 4 behaviour (Wolfram 1984a), to notions of emergent computation at the edge of chaos, and to a phase transition between order and chaos (Langton 1990). Gliders are analogous to autocatalytic sets of polymers in the sense of Kauffman (1993), in that a configuration  $C_1$  sets off a sequence of transformations,  $\rightarrow C_2 \rightarrow C_3 \rightarrow \dots \rightarrow C_1$ , with catalytic closure. Members of such sets have a survival advantage in occupying space, and the set acquires its own identity as an observed object at a higher level. Gliders are also discrete examples of Prigogine's far-from-equilibrium dissipative structures (Prigogine 1984). However, in CA the process of formation, persistence and

---

\* This chapter is partly based on (Wuensche 1994a) and also on work in progress.

interaction of gliders and other dissipative structures can be traced at the lowest level of the system's basic components and their local interactions which are completely defined. This ability to see two levels of behaviour simultaneously, the underlying and emergent, may lead to insights into the mechanics of self-organisation (e.g. Hanson and Cruchfield 1995).

To illustrate a spectrum of CA behaviour, figure 4.1 shows typical space-time patterns of a family of 5-neighbour rules ranging progressively through ordered, complex and chaotic dynamics, corresponding to Wolfram's (1984a) classes (1 and 2) - (4) - (3). Starting with the glider rule in figure 4.1(d), the other rules were evolved by mutating the rule table to progressively force the  $Z$  parameter higher (towards chaos) and lower (towards order). Langton illustrated a similar sequence on the basis of the  $\lambda$  parameter\* (Langton 1990).  $Z$  is a trajectory convergence parameter which predicts the bushiness of sub-trees in attractor basins (Wuensche 1992a, 1994a), described in section 4.12.  $Z$  seems to approximately conform to a subjective view of behaviour, or behaviour as characterised by entropy variance, especially when tuned through its range of values by small mutations of a given rule table.

How simple can a CA be and yet support "interesting" glider behaviour, and what is this quality? How and why is such behaviour able to emerge? What quantitative measures can be used to identify glider dynamics? To answer these questions it may be helpful to examine a relatively large sample of CA rules that appear to us to support glider-like properties, but whose architecture is as simple as possible. In 1d binary CA, a number of glider rules have been identified. Among the 256 binary 3-neighbour rules, the "elementary rules" (Wolfram 1983), an exhaustive search reveals two sets of glider rules, rule 54 and 110, and their equivalents. Their glider interactions have been the subject of particular study (e.g. Li and Nordahl 1992, Boccara *et al.* 1991, Hanson and Cruchfield 1995). Some examples of glider rules in 5-neighbour rule-space have been documented (e.g. Wolfram 1984a, Li 1989, Aizawa *et al.* 1990, Wuensche 1992a, 1994a). Other examples are derived from 1d CA with more complicated architecture, such as a cell value range greater than binary (e.g. Langton 1990, Wolfram 1984a, Lindgren and Nordahl 1990).

Complex rules that feature gliders are supposed to be rare (Wolfram 1985). Most rules are either ordered or chaotic, though ordered rules become increasingly rare for larger neighbourhood size. Complex rules are defined here as those yielding large scale space-time configurations interacting over a relatively long time, that is having relatively long transients before reaching their attractor, where the dynamics along the transient is clearly "interesting". The human mind is uniquely qualified to recognise complex patterns such as interacting gliders, and to separate the interesting

---

\* The  $\lambda$  parameter for a binary CA is the fraction of 1s in the the lookup table (Langton 1990).  $\lambda_{ratio}$  (Wuensche and Lesser 1992a) also referred to in this chapter, is a normalised form of  $\lambda$  to allow direct comparison with the  $Z$  parameter.  $\lambda_{ratio}$  is defined in chapter 5.11.10.  $Z$  is defined in this chapter 4.12.

from the trivial, but it would be extremely useful to have a measure that corresponded closely to our subjective classification. An entropy variance measure on the dynamics seems to achieve this end, and allows a virtually unlimited sample of glider rules and related complex rules to be found, distinguishing rule-space between order, complexity and chaos.

Most rules captured by the entropy-variance measure turn out to be glider rules, but other types of interesting dynamics are also captured, for example two statistically different competing chaotic domains, or a glider domain co-existing with a chaotic domain. The method entails measuring the frequency distribution of blocks of size  $k$ , where  $k$  is the neighbourhood size. This corresponds to the frequency of lookup of different neighbourhoods in the CA rule's lookup table. The measure may be taken relative to each time-step or smoothed over a moving window of time steps. The entropy of this frequency distribution, referred to as the "input entropy" is calculated, followed by its variance (or the equivalent standard deviation) over an interval of time-steps.

Whereas order (low entropy) and chaos (high entropy) both exhibit low variance after the dynamics has settled from a random initial state, only complex dynamics continues to exhibit high variance for an extended time. To understand why this is imagine two converging gliders against a uniform background, the input entropy will be low because the frequency distribution of  $k$ -blocks will be uneven, made up just of background and glider configurations. The eventual collision may produce a temporary chaotic region in the space-time pattern causing the frequency distribution of  $k$ -blocks to even out and the entropy to rise. The chaotic region will then re-organise itself into new gliders causing the entropy to drop, and these gliders will re-collide in turn, and so on. If the block size is taken as larger than  $k$  this effect is amplified. Some glider rules may nevertheless have low input-entropy variance where gliders exist against background domains with large spatial and temporal periods, for example rules 110 and 54. However these cases may be captured by the measure if the block size is taken as larger than  $k$ .

The automatic method for classifying rule-space takes many thousands of rules at random and plots average entropy against entropy variance. Large samples of glider rules and related complex rules for various  $k$  values have been assembled. As  $k$  increases, both complex rules, and to an even greater extent ordered rules, become less frequent. The frequency can be estimated. The methods and results are described.

This method of finding glider rules supersedes a subjective method of artificially selecting random mutations described in (Wuensche 1994a) similar to a proposal by Li (1989). Here, a rule is selected at random from a likely region of rule space according to the  $Z$  parameter. While watching the space-time pattern iterating on the computer screen, the rule is mutated by random bitflips, bitflips-back, or bitflips that raise or lower  $Z$ , to mutate rules until interacting gliders in the space-time pattern become apparent. Alternatively, starting from a glider rule, other glider rules may be

easily found, suggesting that related glider rules separated by small Hamming distances occur in clusters in rule space. As  $k$  increases, the difficulty of the search rapidly escalates because the search space grows as  $2^{2^k}$ .

About 60 glider rules found by this method are shown in appendix 2 and 3 (also in Wuensche 1994b). Figures 4.4-4.6 gives some further examples of the hundreds of glider rules and related complex rules found by the automatic entropy-variance method. Glider dynamics is of prime interest as a simple example of emergence, and its study based on a large sample is now possible.

In addition to rule parameters and local measures on dynamics, a variety of global measures on the topology of attractor basins\* have become available. Using the program Discrete Dynamics Lab (DDLab, Wuensche 1996) attractor basins (described in chapter 2) can be reconstructed, and their topology can be characterised both statistically and explicitly. For example, measures can be taken on the number and size of basins, periods of attractors and length of transients. Related to these measures are the density of garden-of-Eden states, those without pre-images, and how this varies with system size. More generally measures can be made of the distribution of in-degree characterising the typical "bushiness" of attractor basins. These measures relate to order, complexity and chaos as seen in local dynamics. Order corresponds to short, bushy transient trees, attractors with small periods and high garden-of-Eden density. Chaos corresponds to the opposite, very long sparsely branched transient trees, attractors with very large periods and lower garden-of-Eden density. For complex dynamics these measures lie somewhere in between. A histogram of the in-degree distribution shows contrasting profiles for order and chaos, whereas for complexity the distribution appears to follow a power law. These results are presented.

This chapter gives a brief review of CA architecture, space-time dynamics, and the  $Z$  parameter. The reverse algorithm for computing CA pre-images and thus reconstructing their attractor basins is described. With the benefit of a large sample, glider and related complex dynamics is discussed with examples. The method of categorising rules by input-entropy variance is explained. Results on the distribution of large random samples of rules according this measure are presented. These are tested against a subjective expectation of behaviour. These local measures and attributes, together with the global measures on attractor basins and the rule parameter  $Z$ , are related to each other with the aim of quantifying notions of order, complexity and chaos in one-dimensional CA.

---

\* The term attractor basins refers to all of the following: basin of attraction fields, individual basins of attraction, subtrees or fragments of sub-trees.

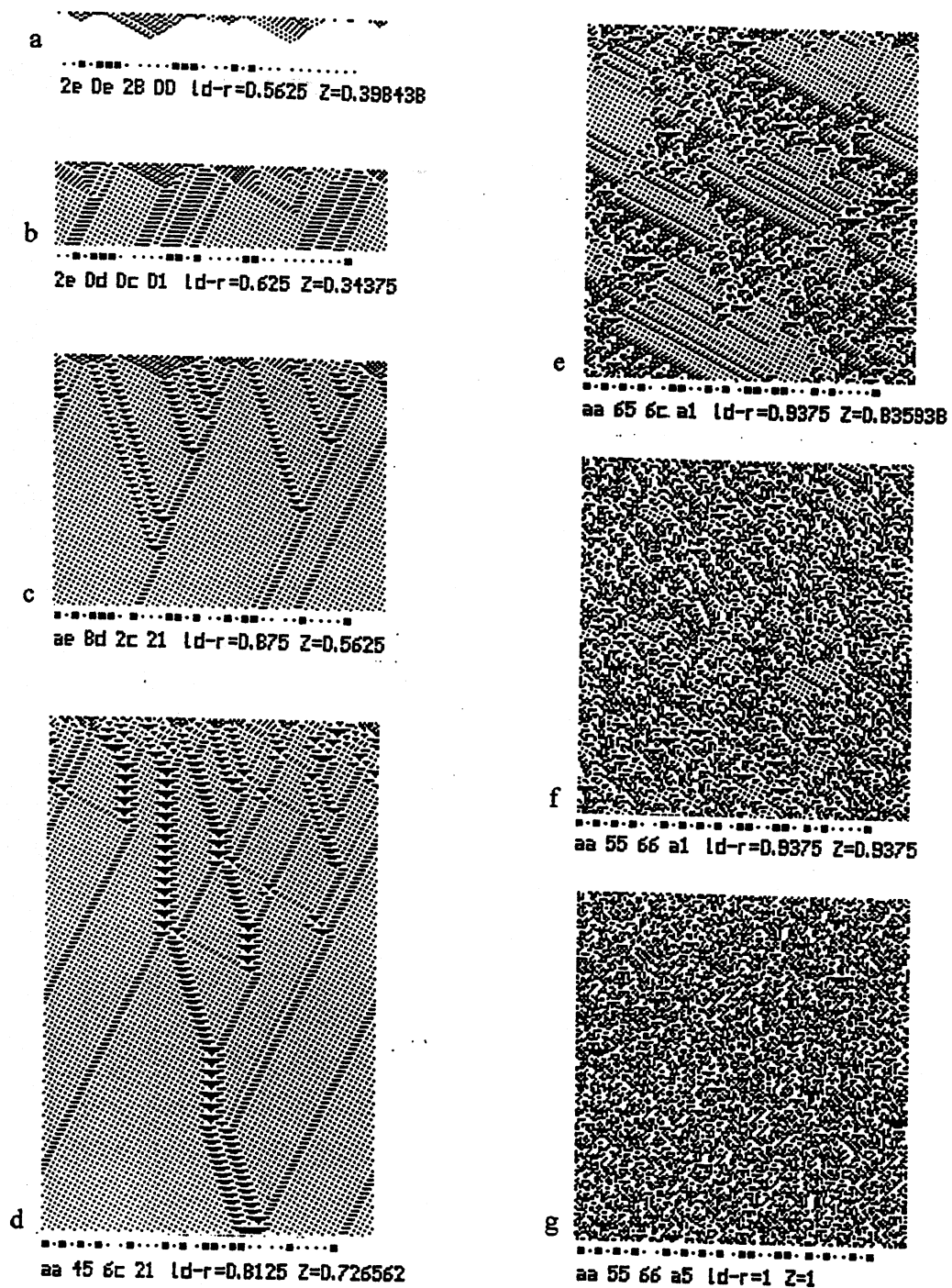
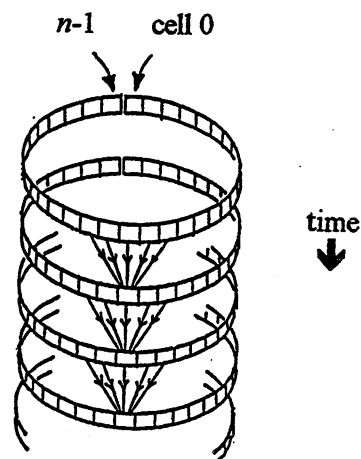


Figure 4.1. Typical space-time patterns of a family of 5-neighbour ( $k=5$ ) rules. The rules a, b, c, d, e, f, g, range through *ordered-complex-chaotic* dynamics. a, b, d, g, correspond to Wolfram's classes 1,2,4,3, where "d" (class 4) shows glider interactions. Starting with this rule, the other rules were evolved by mutating the rule table to progressively force the Z parameter higher (towards chaos) and lower (towards order). The rule number is shown in hex (see this chapter 4.2), together with its  $\lambda_{ratio}$  (ld-r) and Z parameter. The space-time patterns were generated from the same random initial state, system size  $n=150$ . Time proceeds from the top down. Attractor basins and in-degree histograms for a, d and f are shown in figures 4.28 - 4.31.

**Figure 4.2.1d** CA architecture, periodic boundary conditions. Neighbourhood size  $k=5$ , system size  $n$ . All states update synchronously according to the states of the  $k$  neighbours at the previous time-step. Flattening out the "cylinder", split between  $n-1$  and  $0$ , gives the space-time pattern representation as shown in figure 4.1 and elsewhere, where the direction of time is from the top down.



## 4.2. One-Dimensional CA architecture

A CA is a self-contained discrete dynamical system, where space is a lattice of cells with a regular geometry. Cells update their values, chosen from a finite alphabet, as an invariant function of a standard neighbourhood template (the *neighbourhood*). Updating is synchronous in discrete time-steps.

This chapter considers the simplest CA architecture. The alphabet's size is just 2 (0,1). Space is a one-dimensional ring of  $n$  cells (periodic boundary conditions). A cell's neighbourhood (size  $k$ ) is a continuous zone of cells, centred for odd  $k$ , and with an extra cell on the right for even  $k$  (in the convention used here). Most examples in this chapter are for  $k=5, 6$  and  $7$ . A diagram of the system is shown in figure 4.2.

Consider a periodic 1d lattice with  $n$  cells and a size  $k$  neighbourhood.  $r_{\text{left}}$  and  $r_{\text{right}}$  are the neighbourhood radii to the left and right, where  $k = r_{\text{left}} + 1 + r_{\text{right}}$ . The time evolution of the  $i$ -th cell is given by,

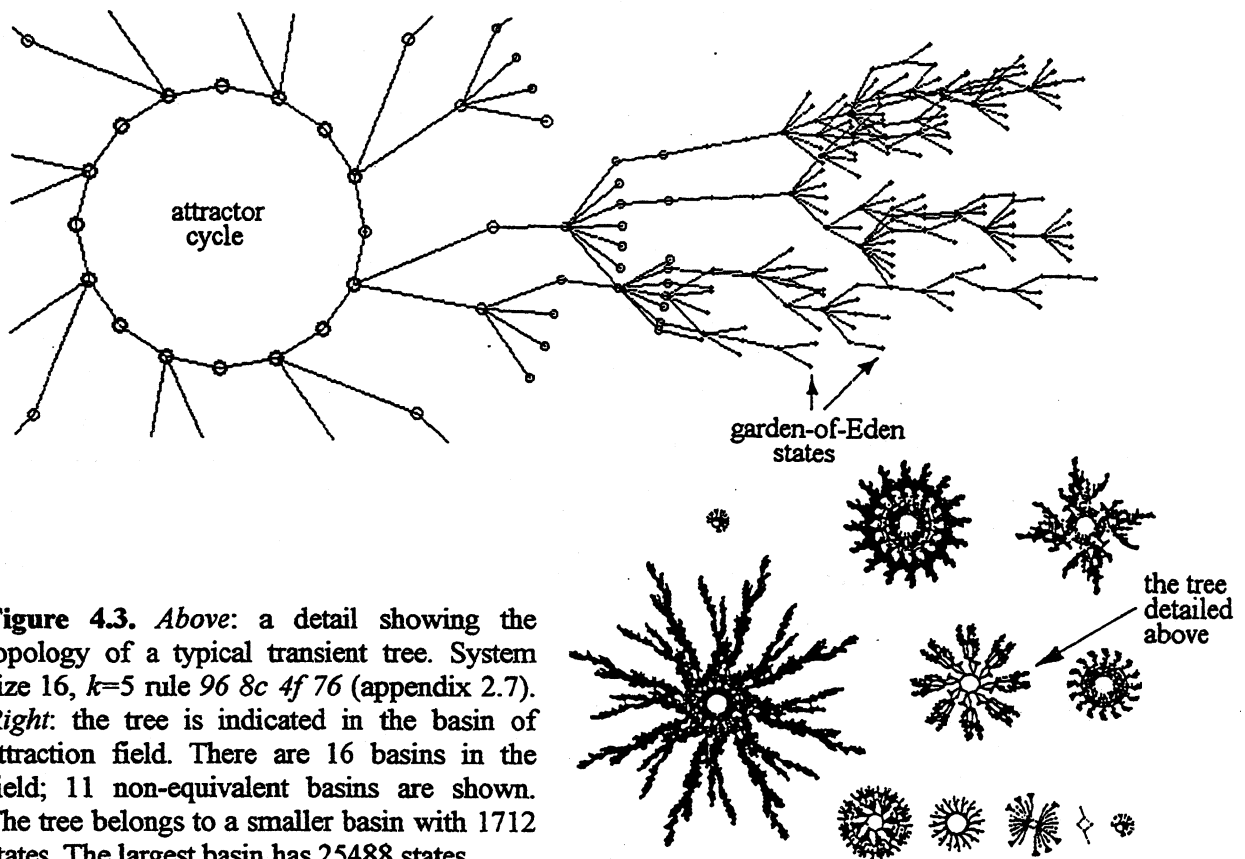
$$C_i^{(t+1)} = f\left(C_{i-r_{\text{left}}}^{(t)}, \dots, C_{i-1}^{(t)}, C_i^{(t)}, C_{i+1}^{(t)}, \dots, C_{i+r_{\text{right}}}^{(t)}\right)$$

to satisfy periodic boundary conditions, for  $x < 1$ ,  $C_x = C_{n+x}$  for  $x > n$ ,  $C_x = C_{x-n}$

A neighbourhood of size  $k$  has  $2^k$  permutations of values. The most general expression of the transition function  $f$  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, Boolean derivatives (Vichniac 1990), totalistic rules (Wolfram 1984a) or threshold functions. The number of effectively different rules is reduced by symmetries in the rule table (Walker 1971, Wuensche and Lesser 1992a). By convention







**Figure 4.3.** Above: a detail showing the topology of a typical transient tree. System size 16,  $k=5$  rule 96 8c 4f 76 (appendix 2.7). Right: the tree is indicated in the basin of attraction field. There are 16 basins in the field; 11 non-equivalent basins are shown. The tree belongs to a smaller basin with 1712 states. The largest basin has 25488 states.

The state-space of a CA with  $n$  cells is  $2^n$ . Any path inevitably encounters a repeat of a previous state, and must lead to a state cycle (the *attractor*). The attractor may have just one state, a stable point cycling to itself, or may have an arbitrarily long period\*. The set of all possible paths leading to the same attractor, including the attractor itself, make up a *basin of attraction*, a concept familiar from continuous dynamical systems. State-space is typically divided into many basins, the *basin of attraction field*.

A trajectory is just one particular path within a basin of attraction. A *transient* is the portion of the trajectory outside the attractor cycle and usually merges with other transients to form a branching tree with garden-of-Eden states as the leaves. A *sub-tree* is a branch of the *transient tree*. Basins of attraction typically have a topology of trees rooted on cycles.

\*The attractor period for a 1d CA with periodic boundary conditions cannot, however, exceed  $2^n - m$ , where  $m$  is the number of states in state space made up of repeating segments on the circular lattice (Wuensche and Lesser 1992a).

#### 4.4. Computing CA pre-images

Constructing a basin of attraction or subtree poses the problem of finding the pre-images of each state. A possible method is to construct an exhaustive map resulting from network dynamics and to scan the map for pre-images, (as described in chapter 3.7) but this becomes computationally intractable as the network's size increases beyond modest limits. However, a *reverse algorithm* that directly computes pre-images for one-D CA, without the need for an exhaustive map, has been invented by the author (Wuensche and Lesser 1992a). Using this algorithm the network's dynamics can be run *backwards* in time; backward trajectories will, as a rule, diverge. The DDLab source code of the function that implements the algorithm (for any  $k$ , and for mixed rules) is shown in appendix 6.2. Appendix 7.2 describes network size limitations and time issues.

A different though related reverse algorithm for RBN (Wuensche 1993a) described in chapter 5 also applies to CA which are a subset of RBN. These alternative methods serve as a useful reality check on the correctness of the computed pre-images.

The CA algorithm is described below.

Consider a 1d CA with  $n$  cells and neighbourhood size  $k$ . The algorithm will be demonstrated for a neighbourhood  $k=3$ . Equivalent algorithms apply for other values of  $k$ . The system is shown as a 1d array,  $A_{n-1} A_{n-2} \dots A_1 A_0$ .

Consider also a  $k=3$  lookup table defining the CA rule as follows, where  $T_d$  denotes the output of the neighbourhood whose decimal equivalent value is  $d$ .

rule-table...	111	110	101	100	011	010	001	000	...neighbourhoods
	0	0	1	1	0	0	1	0	...example outputs (0 or 1)
	$T_7$	$T_6$	$T_5$	$T_4$	$T_3$	$T_2$	$T_1$	$T_0$	

To derive the pre-images of an arbitrary global state, consider a *partial* pre-image where the *start string*, the left  $k-1$  cells, is assigned a pattern of 0s and 1s from the  $2^{k-1}$  possible.

Because the boundary conditions are periodic, start strings are assigned to  $P_0 P_{n-1}$  for  $k=3$ . For other values of  $k$  appropriate start strings are assigned in the same way, for example for  $k=5$  the start string is  $P_1 P_0 P_{n-1} P_{n-2}$  for an asymmetric neighbourhood such as  $k=4$  with the extra cell on the right the start string is  $P_0 P_{n-1} P_{n-2}$ .

The remaining cells are empty or unknown, unallocated as either 0 or 1. Empty cells are denoted by the wild card star symbol  $\star$ , known cells (with values established as 0 or 1) are denoted by the block symbol  $\square$ .

Consider a known network state,  $A_{n-1} A_{n-2} \dots A_0$  and the partial pre-image state  $P_{n-1} P_{n-2} \dots P_0$ .

	$P_0$	$P_{n-1}$	$P_{n-2}$	$P_{n-3}$	$P_{n-4}$	·	·	$P_3$	$P_2$	$P_1$	$P_0$	$P_{n-1}$
partial pre-image ...	□	□	☆	☆	☆	☆	☆	☆	☆	☆	□	□
known state ...		□	□	□	□	□	□	□	□	□	□	
		$A_{n-1}$	$A_{n-2}$	$A_{n-3}$	$A_{n-4}$	·	·	$A_3$	$A_2$	$A_1$	$A_0$	

Starting with the known cell,  $P_{n-1}$  (at the centre of the  $k=3$  neighbourhood  $P_0P_{n-1}P_{n-2}$ ) find the value of the next unknown cell to the right at  $P_{n-2}$ , consistent with the lookup table. More generally, knowing the partial pre-image from the left up to  $P_i$  find the value of the next unknown cell to the right of  $P_i$  at  $P_{i-1}$ . The known cell values at  $P_{i+1}P_i☆$  correspond to two entries in the rule table. When the outputs of these two entries of are compared with each other and with  $A_i$  there are three possible consequences. The permutation is either *deterministic*, *ambiguous* or *forbidden*.

	$P_{i+1}$	$P_i$	$P_{i-1}$	
neighbourhood .....	□	□	☆	compare the two outputs of $P_{i+1}P_i☆$
known cell ...		□		with each other and with and $A_i$
		$A_i$		

(1). Deterministic permutations

if the outputs of the two neighbourhoods  $P_{i+1}P_i☆$  are different,  
 i.e. if.  $P_{i+1}P_i 0 \rightarrow T$ , and  $P_{i+1}P_i 1 \rightarrow \text{not}T$   
 then the value of  $P_{i-1}$  is uniquely determined, and can be filled in as known.

(2). Ambiguous or forbidden permutations

if the outputs of the two neighbourhoods  $P_{i+1}P_i☆$  are the same,  
 i.e. both.  $P_{i+1}P_i 0$  and  $P_{i+1}P_i 1 \rightarrow T$ , there are two possible consequences,

(2a). Ambiguous permutation

If  $T = A_i$ , then both 0 and 1 are equally valid solutions for  $P_{i-1}$ . The partial pre-image must be duplicated with  $P_{i-1}=0$  in one partial pre-image and  $P_{i-1}=1$  in the other.

(2b). Forbidden permutation

If  $T \neq A_i$  then  $P_{i-1}$  has no valid solution and the partial pre-image is rejected.

If  $P_{i-1}$  has a valid solution (i.e. the permutation is deterministic or ambiguous) the procedure is continued to find the value of the next empty cell to the right,  $P_{i-2}$ . If the permutation is ambiguous both alternative partial pre-images will need to 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 applied to determine each successive unknown  $P_{i-1}$  towards the right, any incidence of an ambiguous permutation will require a partial pre-image to be added to the stack.

To increase speed and minimise the growth of the partial pre-image stack, forbidden permutations are also checked towards the left, i.e. of the form,

	$P_{i+1}$	$P_i$	$P_{i-1}$
neighbourhood . . . . .	☆	□	□
known cell . . .		□	
		$A_i$	

In the case of an ambiguous permutation, given one of the solutions i.e.  $P_{i-1}=0$ , a check is made on whether the next permutation one place further to the right at  $P_{i-2}$  (as shown below) is forbidden,

	$P_i$	$P_{i-1}$	$P_{i-2}$
neighbourhood . . . . .	□	□	☆
known cell . . .		□	
		$A_{i-1}$	

If this is the case the unique solution  $P_{i-1}=1$  is entered.

The procedure is continued to the right to overlap the assumed start string  $P_0$  and  $P_{n-1}$  to check if periodic boundary conditions are satisfied. If not the pre-image is rejected. For  $k \neq 3$  a larger or smaller overlap needs to be checked. If the boundary conditions are satisfied 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 (2a) occurs a new partial pre-image must be added to the stack, but the stack will eventually become exhausted, at which point all the valid pre-images containing the assumed start string of  $P_0 P_{n-1}$  will have been found. The procedure is repeated for each of the remaining values of  $P_0 P_{n-1}$ , (or the appropriate start string for  $k \neq 3$ ).

The reverse algorithm is applied from left to right in DDLab, but is equally valid when applied from right to left.

The reverse algorithm for computing pre-images works for 1d CA and also for networks with CA-like wiring, i.e. with a local neighbourhood of homogeneous size  $k$ , but where the rules may be different at each site. Provided that the neighbourhood is small as compared to network size,  $k \ll n$ , the algorithm is many orders of magnitude faster than constructing and scanning an exhaustive map as described in chapter 3.6.

#### 4.5. Constructing and Portraying CA Attractor Basins

Once the pre-images of a state are known, the information is applied to construct the pre-image *fan*, from the given state to its set of pre-images (if any). The pre-image fan for each pre-image is then computed, and so on, until only garden-of-Eden states remain. In this way transient trees (or subtrees) are constructed. A basin of attraction (or the complete basin of attraction field) is constructed by first running the network forward to reveal the attractor cycle, then computing each transient tree in turn.

In the computer diagrams of attractor basins drawn in DDLab, representations of global states are linked by directed arcs. Each node will have zero or more incoming arcs from its pre-image nodes, but because the system is deterministic, exactly one outgoing arc (one *out-degree*). Nodes with no pre-images have no incoming arcs, and represent garden-of-Eden states. The number of incoming arcs to a node is its *in-degree*.

The methods and the graphic conventions are explained more fully in chapter 2 and (Wuensche and Lesser 1992a).

Figure 4.3 shows the basin of attraction field of a  $k=5$ ,  $n=16$  CA, with a detail of the topology of a typical transient tree rooted on an attractor cycle. The CA has organised the  $2^{16}=65536$  states in state-space into a field with 16 basins of attraction. Because of rotation symmetries in the circle of cells, some of these basins, and also transient trees within basins, are equivalent to each other (see Chapter 1.4). Only the 11 non-equivalent basins in the basin of attraction field are shown.

#### 4.6. Complex rules and Gliders

Whether or not a rule is described as ordered, complex or chaotic has depended to a large extent on a subjective appraisal its typical emergent space-time patterns. Each CA rule self-organises its space-time patterns in a characteristic way, and these patterns are often recognisable given our talent for pattern recognition. For  $k \leq 5$  rules a characteristic structure to the pattern is apparent even when the space-time patterns appears chaotic. This becomes less obvious for larger  $k$ . The characteristic pattern structure of different rules can be analysed in formal language theory as a "regular language" with a vocabulary made up of bit sequences and a "grammar" made up of succession rules between sequences (e.g. Wolfram 1984b), and by a related "computational mechanics" approach (Hanson and Cruchfield 1996).

Certain space-time patterns appear especially interesting or intriguing. Within the overall space-time pattern, periodic sub-patterns may emerge, move across a regular background, and interact with other periodic sub-patterns in a particle-like manner. Conway's "game of Life" is perhaps the most interesting example of a 2d rule that support this behaviour (Conway 1992). The term "gliders" used

in this chapter is borrowed from him. Figure 1.1. shows an example of a 2d space-time pattern of the "life" rule as an isometric projection (as if looking up at a transparent box). Time proceeds from the top-down on the vertical time axis. Gliders appear as regular diagonal features.

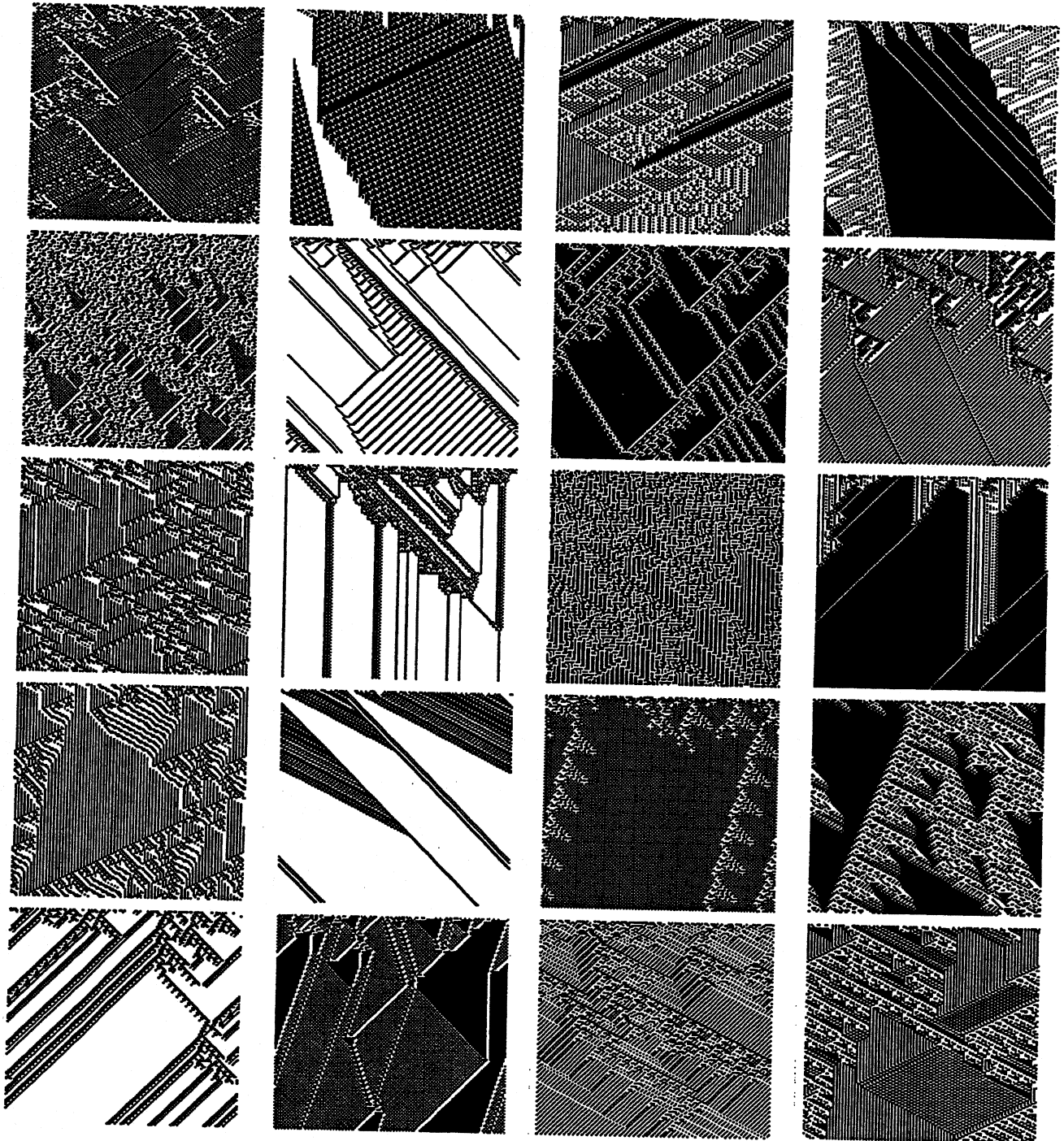
Gliders dynamics corresponds to Wolfram's Class 4 behaviour in his classification of CA dynamics (Wolfram's 1994a). He conjectures that this most complex class 4 behaviour is capable of universal computation. Wolfram lists his classes in order of increasing complexity of their typical space-time patterns as measured in formal language theory (Wolfram 1984b), and draws analogies with classical continuous dynamical systems theory in terms of the attractors typical of each class. His classes are as follows,

<u>class</u>	<u>description in CA dynamics</u>	<u>dynamical systems analogue</u>
1.	Tends to a spatially homogeneous state.	..... limit points.
2.	Yields a sequence of simple stable or periodic structures. ....	..... limit cycles.
3.	Exhibits chaotic aperiodic behaviour. ....	..... chaotic (strange) attractors.
4.	Yields complicated localised structures, some propagating. ....	..... attractors unspecified.

Langton (1990) and others have argued correctly that Wolfram's class 4 more naturally belongs between classes 2 and 3, at a phase transition between order and chaos, the so called "edge of chaos", which can be traversed by tuning the  $\lambda$  parameter. For binary rules tuning the  $Z$  parameter gives a finer correspondence with observed space-time pattern behaviour, as well as relating behaviour to the topology of attractor basins. It will be shown that complex rules have intermediate topology between order and chaos, including measures of the garden-of-Eden density and the profile of the in-degree histogram. For these reasons class 4 is relocated between 2 and 3. Classes 1 and 2 are combined because many ordered rules have both limit points and short limit cycles, though one or the other may predominate. Wolfram's rule classes are readjusted here as follows:

ordered (class 1 and 2) - - - - - complex (class 4) - - - - - chaotic (class 3)

Figures 4.4-4.6 shows some examples of complex dynamics taken from the sample automatically generated by input-entropy variance for  $k=5$ ,  $k=6$  and  $k=7$  rules, and further examples are shown in appendix 2 and 3. The space-time patterns shown were to some extent selected for an interesting view of glider interactions by varying the initial random seed. The gliders and other



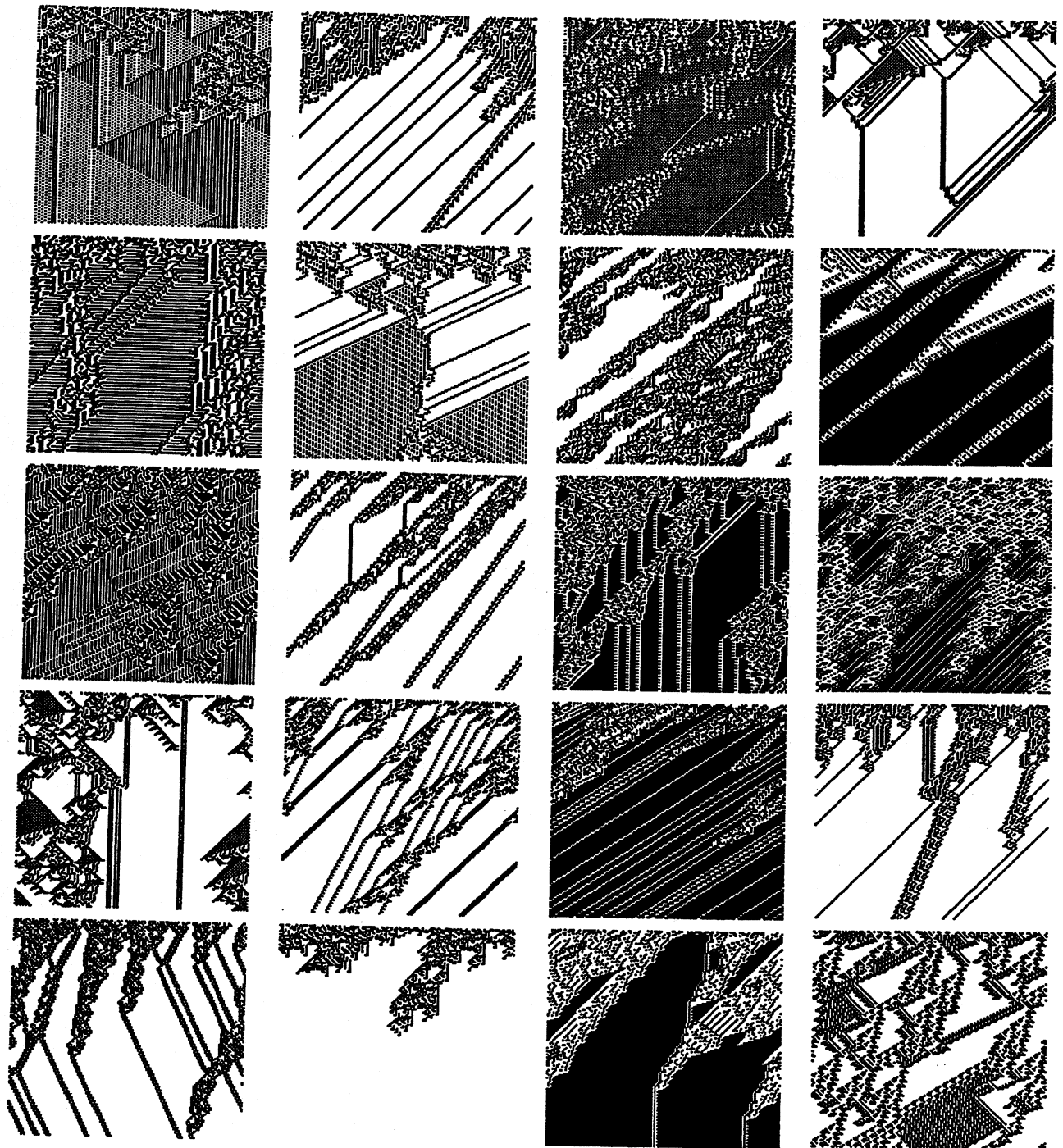
index 2-6

index 7-11

index 12-16

index 17-21

**Figure 4.4.**  $k=5$  complex space-time patterns with high input-entropy variance from the automatic sample,  $n=150$ , 140 time steps from a random initial. The rule numbers are shown in appendix 5.1.5. index 2-21. See appendix 5.1.1 for examples of ordered and chaotic rules.



index 2-6

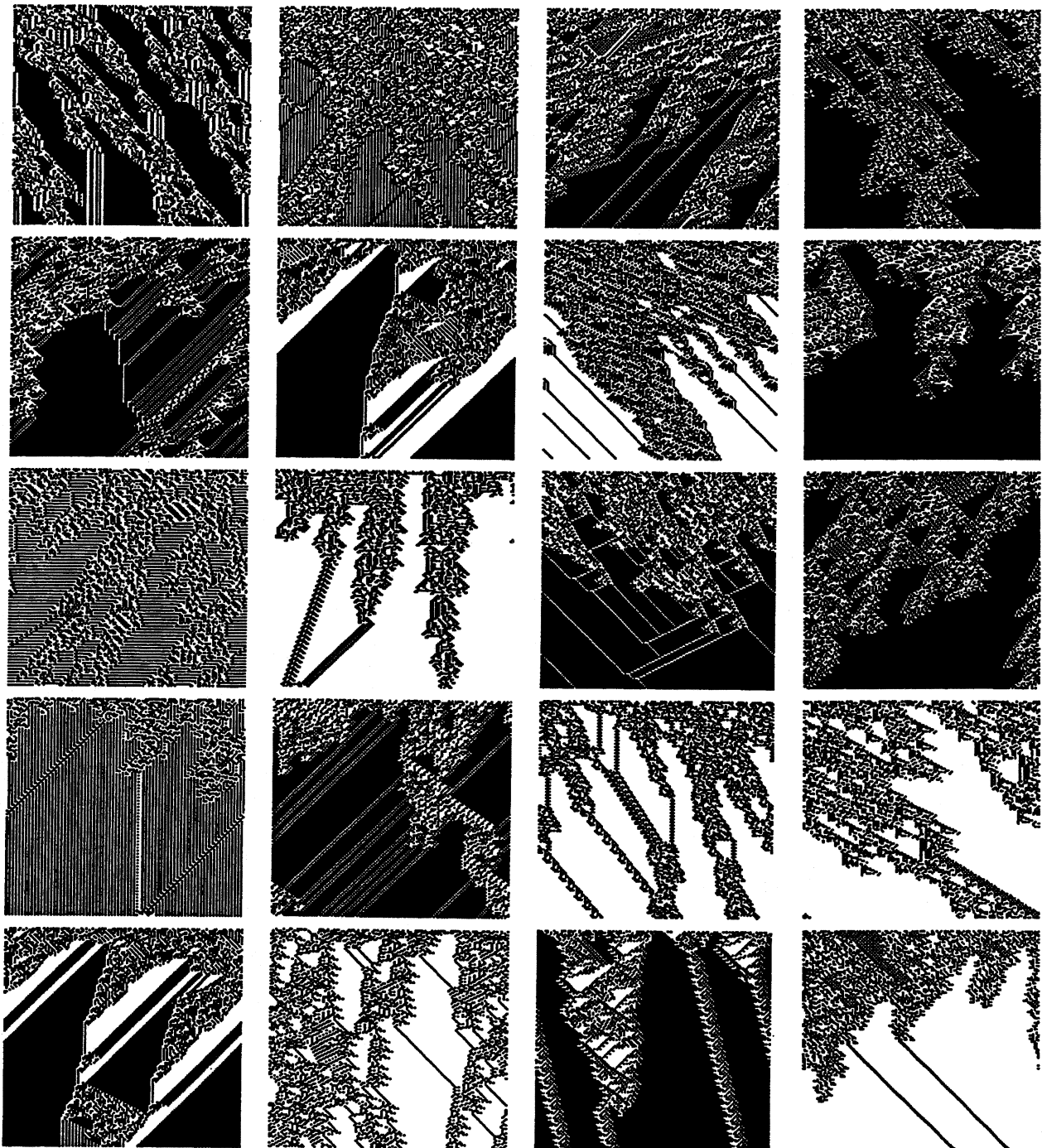
index 7-11

index 12-16

index 17-21

**Figure 4.5.**  $k=6$  complex space-time patterns with high input-entropy variance from the automatic sample,  $n=150$ , 140 time steps from a random initial state. The rule numbers are shown in appendix 5.2.5. index 2-21. See appendix 5.2.1 for examples of ordered and chaotic rules.





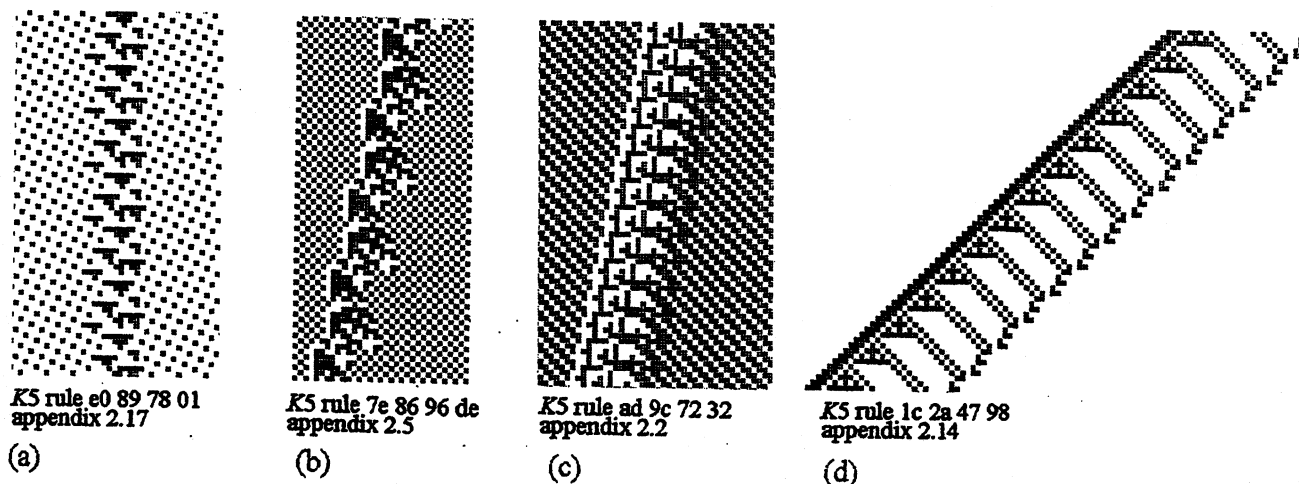
index 2-6

index 7-11

index 12-16

index 17-21

**Figure 4.6.**  $k=7$  complex space-time patterns with high input-entropy variance from the automatic sample,  $n=150$ , 140 time steps from a random initial state. The rule numbers are shown in appendix 5.3.5. index 2-21. See appendix 5.3.1 for examples of ordered and chaotic rules.



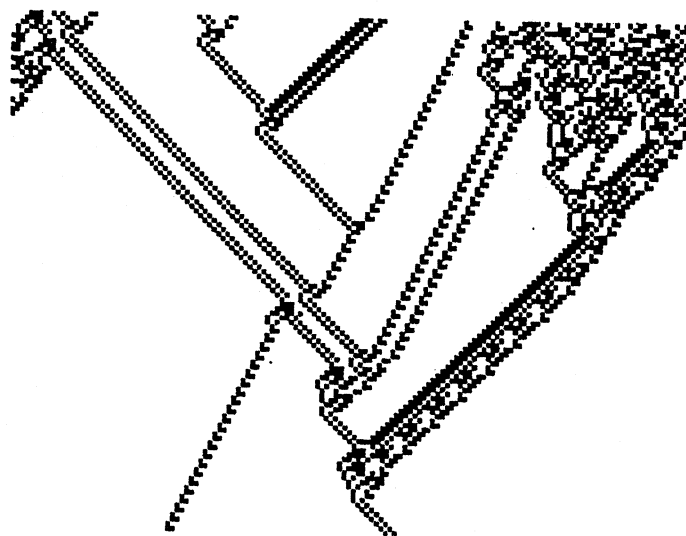
**Figure 4.7.** Gliders with various velocities and backgrounds. The hex rule numbers (defined in 4.2) and relevant appendix pages are shown.

complex structures are seen emerging rapidly in periodic arrays of size 150-200. It should be noted that although this is a reasonable size, there may be rules that appear chaotic when limited to this size but which allow complex dynamics in larger sized systems, requiring a longer time for wider gliders to emerge.

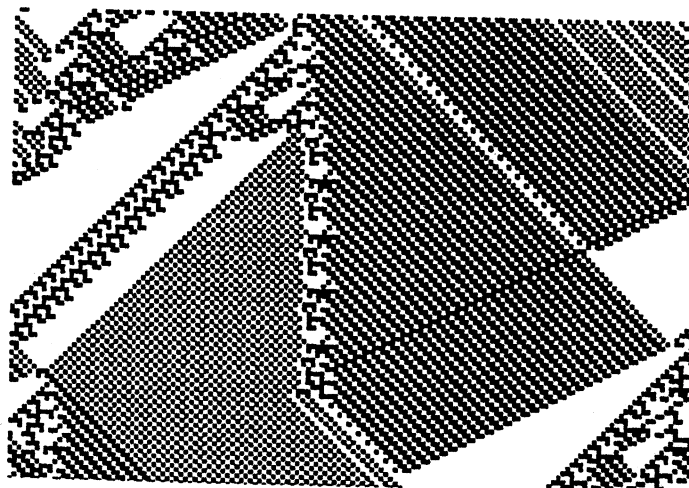
What are the essential features of glider behaviour based on our sample? Glider dynamics occurs if a limited set of self-sustaining configurations emerge from random initial states, and if the interactions between configurations persist for an extended time before settling into a relatively short attractor. The configurations are static, or propagate at various velocities up to a maximum, the system's *speed of light*. They exist against a uniform or periodic space-time background which of necessity has simultaneously emerged. This regular background may be simple, such as a checkerboard, or a more complicated pattern.

Appendices 2 and 3 present samples of about 60 1d CA rules with glider-like characteristics. These rules were found before the automatic method for finding complex rules was implemented. The rules were evolved by the method described in section 4.1, or found by accident. Several are borrowed from other sources (Aizawa *et al* 1990, Li 1989, Wolfram 1986a, Wuensche and Lesser 1992a). Appendix 2 presents 36 rules, each with a typical space-time pattern, 200 cells  $\times$  480 time-steps, a detail showing glider collisions, and the basin of attraction field including significant data for a system size of 16. Appendix 3 presents 26 further rules, each with a typical space-time pattern, 150 cells  $\times$  460 time-steps, showing the lookup frequency histogram and entropy plot alongside (described in section 4.17 below).

**Figure 4.8.**  
 Glider collisions against a  
 quiescent background (all  
 0s).  $k=5$  rule *5c 6a 4d 98*  
 (appendix 2.4).

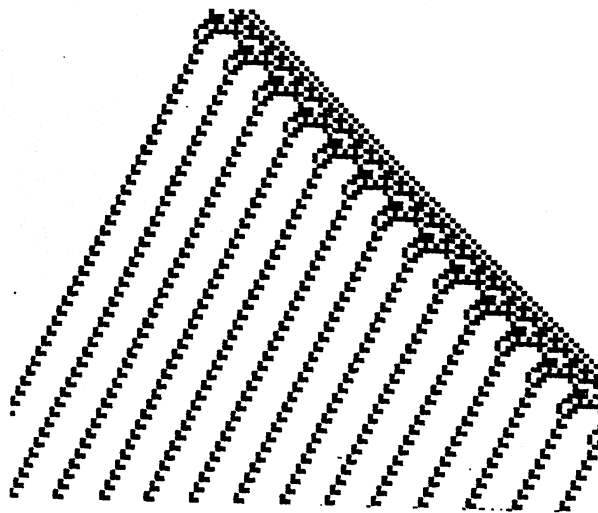


**Figure 4.9.**  
 A glider forming the  
 boundary between two  
 different periodic back-  
 grounds.  
 $k=5$  rule *bc 82 27 1c*  
 (appendix 2.9).



Although there are borderline cases, space-time patterns made up of "interesting" glider interactions are generally easily recognised in contrast to patterns that stabilise rapidly to fixed points or short periods on the one hand, or where chaotic patterns persist on the other. The borderline cases verge either on ordered or chaotic behaviour. Chaotic behaviour may also contain distinct chaotic backgrounds or domains (Crutchfield and Hanson 1993), where *filtering* is required to uncover domain walls analogous to gliders.

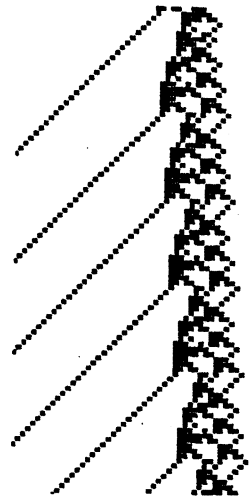
The space-time patterns in our sample show many examples of gliders within a periodic space-time background or a uniform background. Some examples are given in figure 4.7-4.15. A uniform background (all white or black) has a period of one in both space and time. Gliders may be regarded



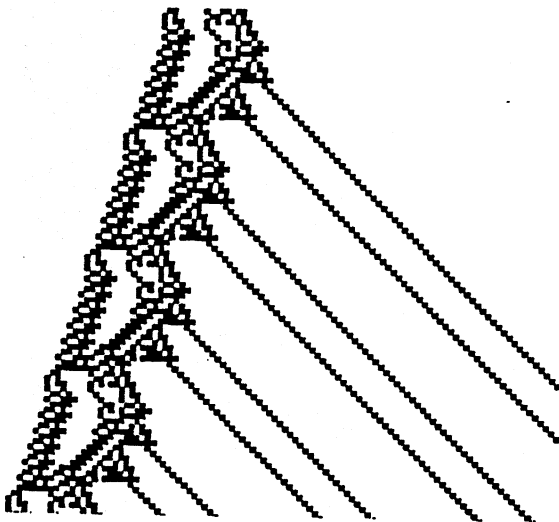
K5 rule 5c 6a 4d 98, appendix 2.4  
(a)



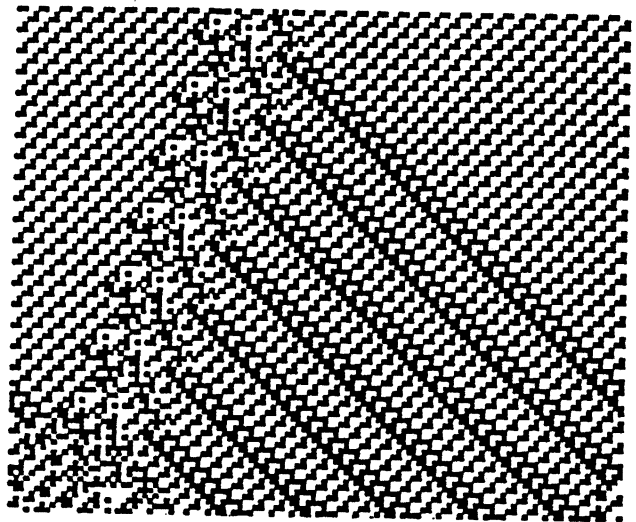
K5 rule 36 0a 96 f9, appendix 2.6  
(b)



K5 rule 97 8e ce e4, appendix 2.18  
(c)



K5 rule = 6c 1e 53 a8, figure 12  
(d)

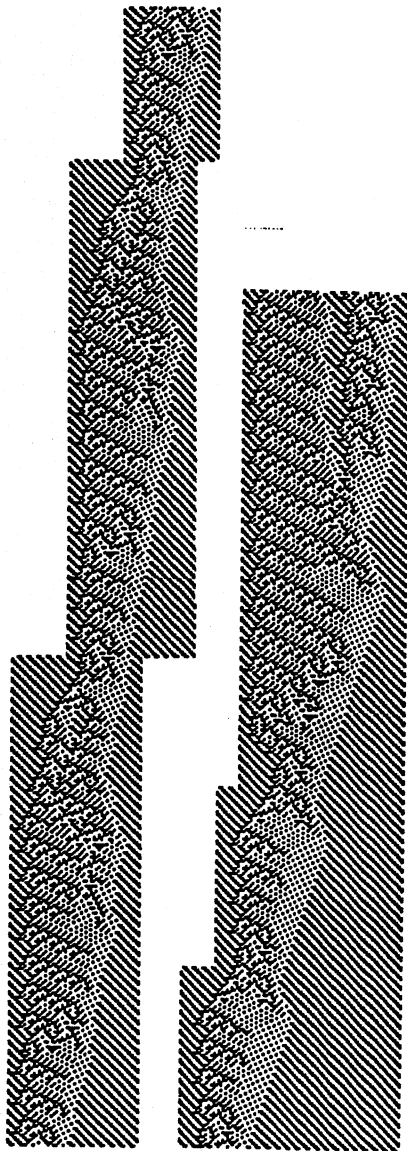
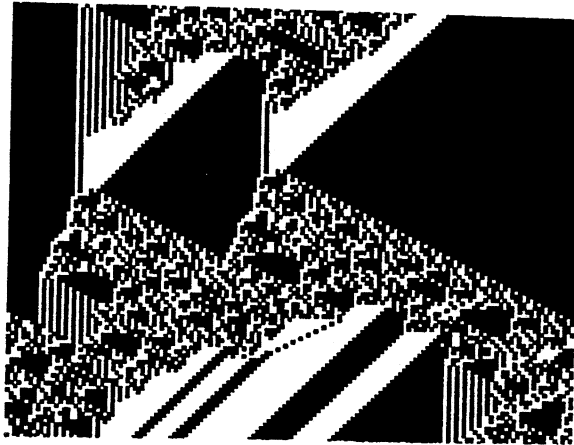


K5 rule = 98 8d 66 3c, appendix 3.2  
(e)

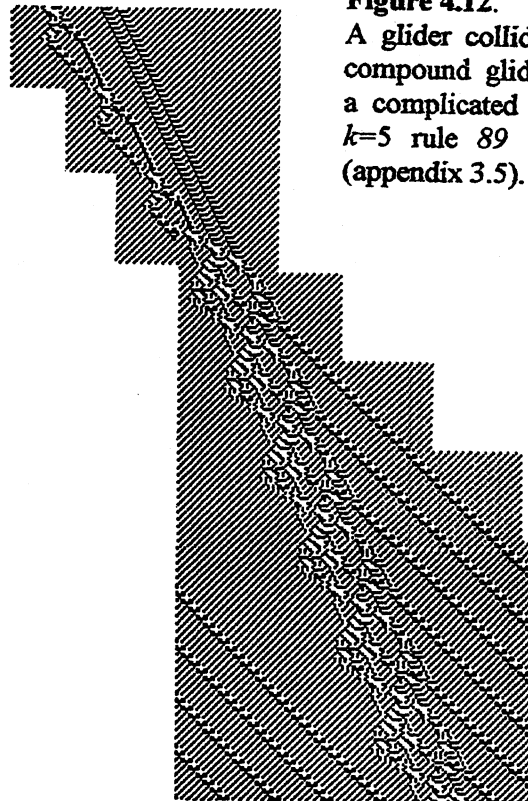
**Figure 4.10.** Examples of glider-guns. The hex rule numbers and relevant appendix pages are shown.

as solitary waves within the background. Gliders may have the special property of *solitons* (Aizawa *et al* 1990, and see appendix 2.12), preserving their shape and velocity after interacting with other solitons. For a neighbourhood of radius  $r$ , glider velocity varies from 0 to a maximum of  $r$  cells per time step towards the left or right. A glider configuration that repeats at each time-step, i.e. with period one, is limited to velocities of  $0, 1, 2, \dots, r$  per time-step. Gliders with periods greater than one may have intermediate fractional velocities.

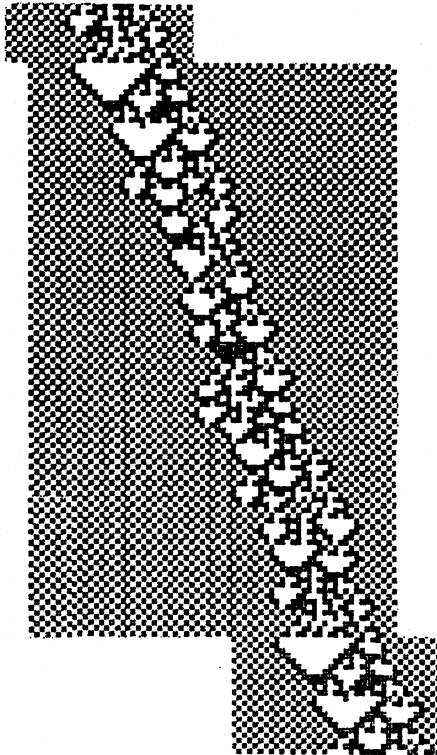
**Figure 4.11.** Example of ordered domains co-existing with a chaotic domain.  $k=7$  rule *fe 40 f5 96 18 oa 98 6e ba ba 9e 06 fe ac d4 e4*



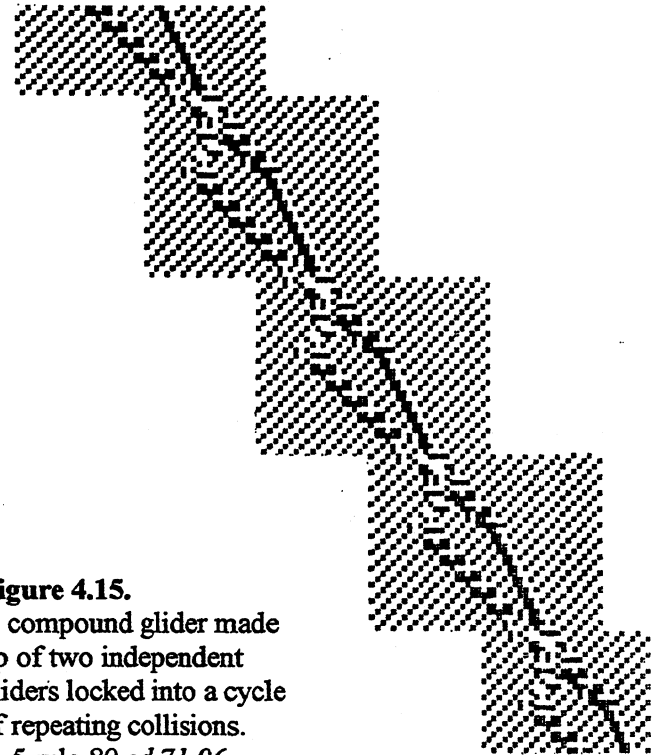
**Figure 4.12.** A glider colliding with a compound glider creating a complicated glider-gun.  $k=5$  rule *89 ed 71 06*. (appendix 3.5).



**Figure 4.13.** *Far left:* a glider with a large diameter and period. The diameter varies between 22 and 42 cells, the period is 402 time-steps. *Left:* a collision between two large gliders creating a third large glider.  $k=5$  rule *82 26 dc 23* (appendix 3.2).



**Figure 4.14.**  
A glider with a period of 106 time-steps.  
 $k=5$  rule *b5 1e 9c e8* (also figure 4.17)



**Figure 4.15.**  
A compound glider made up of two independent gliders locked into a cycle of repeating collisions.  
 $k=5$  rule *89 ed 71 06*,  
(appendix 3.5).

A glider's attributes are the background pattern and spatio-temporal period (on both sides of the glider), the glider's temporal period and velocity, its changing diameter, and the list of its repeating configurations. The same description might be applied recursively to each sub-glider component of a *compound* glider.

From a random seed, a limited number of different glider types emerge after an initial sorting out phase and continue to interact by collisions for an extended time, for example figure 4.1(d). Collisions between two glider types often result in a third glider type (or more). One or both of the gliders may survive a collision with a possible shift in trajectory, or both gliders may be destroyed. In some cases a collision initially results in a chaotic interaction phase, before the final outcome emerges. The outcome of a collision is sensitive to the point of impact relative to the space-time period of each glider.

The emergence of gliders implies the emergence of one or more periodic backgrounds. A glider generally represents a dislocation or defect of varying width in the background, which is often out of phase on either side of the glider, analogous to fracture planes in a crystal lattice. Alternatively, a

glider may be seen as the zone that reconciles the two areas of out-of-phase background. A glider may separate two entirely different backgrounds, acting as the boundary. The example in figure 4.9 has three different backgrounds. Gliders that eject a stream of sub-gliders at regular intervals, as in figure 4.10, and gliders that survive by absorbing a regular glider stream, as in figure 4.7(d), are relatively common in the samples. They are analogous to *glider-guns* and *eaters* in the "game of Life" (Conway 1982). Because a regular glider stream is essentially the same as a regular periodic background, a glider-gun creates a background, and a glider-eater absorbs a background. Glider-guns/eaters are thus equivalent to a glider forming the boundary between two backgrounds.

Both the period and diameter of a glider may be considerable. The diameter may show a large variation within the period. The example in figure 4.14 shows a glider with a period of 106 time-steps. The example in figure 4.13 (*left*), shows a glider with a period of 402 time steps; its diameter varies between 22 and 42 cells. A further example of the same rule in figure 4.13 (*right*) shows a collision between two large gliders creating a third large glider. Clearly such glider interactions can only emerge in systems large enough to contain them.

The existence of *compound* gliders made up of sub-gliders colliding periodically may be expected in large enough systems. Compound gliders could combine into yet higher order structures (Langton 1986), and the process could unfold hierarchically without limit. The example in figure 4.7(d) shows a compound glider made from a glider-gun and a parallel glider-eater which absorbs the sub-glider stream; the compound glider can have an arbitrary diameter. A compound glider-gun is shown in figure 4.10(d). Figure 4.12 shows a compound glider colliding with two gliders creating a compound glider-gun. The compound glider is made of two independent gliders locked into a cycle of repeating collisions, a detail is shown in figure 4.15.

Compound gliders are analogous to the glider/gun/eater interactions engineered by Conway (1982) to make logical gates and an external memory to demonstrate that his 2d "game of Life" CA is capable of universal computation. Others have demonstrated universal computation by glider interactions in simple 1d CA, but with cell states greater than binary (Smith 1971, Lindgren and Nordahl 1990). An example of a kind of self-reproduction is seen in appendix 2.6 (rule *3a 48 b5 c4*), where gliders eject close mirror image copies of themselves with opposite velocity. A third glider kills off the reproducing gliders, checking overcrowding. Gliders may be seen as analogous to autocatalytic sets of polymers in that a configuration  $C_1$  sets off a sequence of transformations,  $\rightarrow C_2 \rightarrow C_3 \rightarrow \dots \rightarrow C_1$ , with *catalytic closure* (Kauffman 1993). Members of such sets have a survival advantage in occupying space, and the set acquires its own identity as an independent higher level object.

Once gliders have emerged, CA dynamics may in principle be described at a higher level, by glider collision rules as opposed to the underlying CA rules.

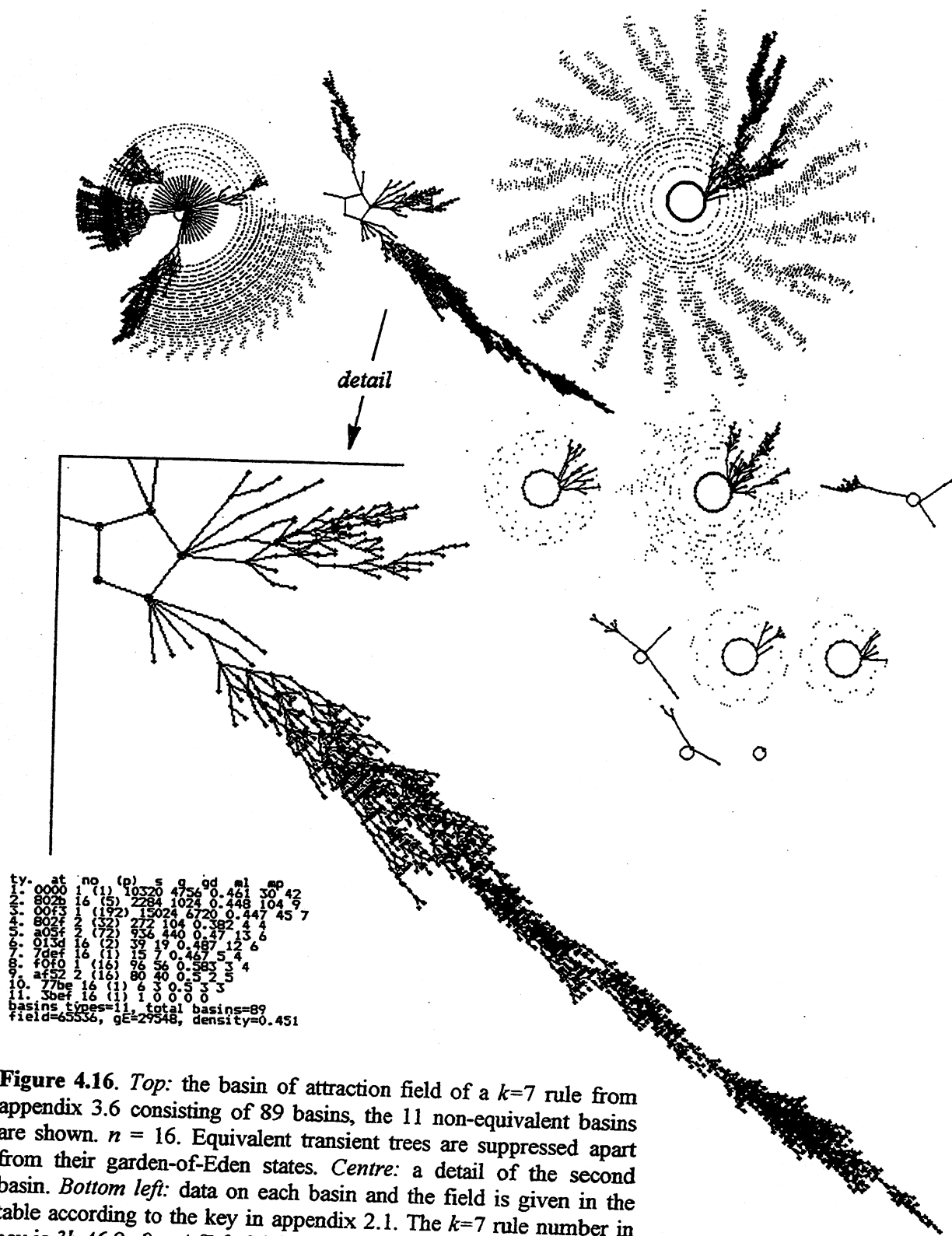


Figure 4.16. Top: the basin of attraction field of a  $k=7$  rule from appendix 3.6 consisting of 89 basins, the 11 non-equivalent basins are shown.  $n = 16$ . Equivalent transient trees are suppressed apart from their garden-of-Eden states. Centre: a detail of the second basin. Bottom left: data on each basin and the field is given in the table according to the key in appendix 2.1. The  $k=7$  rule number in hex is  $3b\ 46\ 9c\ 0e\ e4\ f7\ fa\ 96\ f9\ 3b\ 4d\ 32\ b0\ 9e\ d0\ e0$ , (see appendix 3.6, bottom left).



#### 4.7. Glider interactions and basins of attraction.

It is possible to identify classes of configurations that make up different components of the basin of attraction field in glider rules. In states chosen at random, all configurations occur with equal probability, so the special glider-background configurations are unlikely to be picked. Non-glider/background states make up the majority of state space, and are likely to be garden-of-Eden states, or states just a few steps forward in time from garden-of-Eden states. They occur in the initial sorting out phase of the dynamics and appear as short bushy dead-end side branches along the length of long transients, as well as at their tips.

States dominated by glider and background configurations are special cases, a small subcategory of state-space. They constitute the glider interaction phase, making up the main lines of flow within the long transients. This has also been noted by (Gutowitz and Domain 1995), who described the main lines of flow as the topological skeleton of physically relevant states and the short dead end side branches from garden-of-Eden states as a skin of non-physical transitional states, comprising the bulk of the nodes in an attractor basin.

Gliders in the interaction phase can be regarded as competing sub-attractors, with the final survivors persisting in the attractor cycle. Finally, states made up solely of non-interacting gliders configurations (i.e. having equal velocity), or backgrounds free of gliders, must cycle and therefore constitute the relatively short attractors, with a period depending on the glider velocity. Attractor states themselves are a small subcategory of possible glider/background configurations, and thus form a tiny subcategory of state-space. By simply looking at the space-time patterns of a glider rule from a number of different initial states, most gliders in its glider repertoire (relative to the system size) may be identified. A complete list would allow a complete description of all the attractors in state-space, by finding all possible permutation of non-interacting gliders.

Figure 4.16 shows a typical basin of attraction field of a glider rule for  $k=7$ ,  $n=16$ . Here the system size is too small to support glider interactions, nevertheless the figure gives some flavour of the topology expected. Fragments of subtrees can, however, be computed for sizes that are large enough to support gliders. Examples are shown in figures 4.31 and 4.33 where  $n=50$  and 150.

#### 4.8. Parameters and measures.

What parameters and measures are there to distinguish the range of CA dynamics from ordered to chaotic, and in particular to identify complex rules?

A number of alternative measures of CA behaviour, including complexity measures, have been proposed. For example regular language complexity (Wolfram 1984a), related measures based on deBruijn diagrams (McIntosh 1990), complexity measures related to computation (Langton 1990,

Li *et al.* 1990, Crutchfield and Young 1989), information storage and prediction (Grassberger 1986), net information gain (Bates and Shepherd 1993), block probability distribution (Lindgren and Nordahl 1988), statistics on transient length (Gutowitz 1991), mean field theory (Gutowitz and Langton 1995), and correlation function and mutual information (Li 1990a). Other measures of complexity relate to entropy and information theory (e.g. Zurek 1990), such as algorithmic complexity and logical depth (Bennet 1986). These various approaches relate to each other and possibly to glider emergence, but a comparative discussion must be left for another occasion.

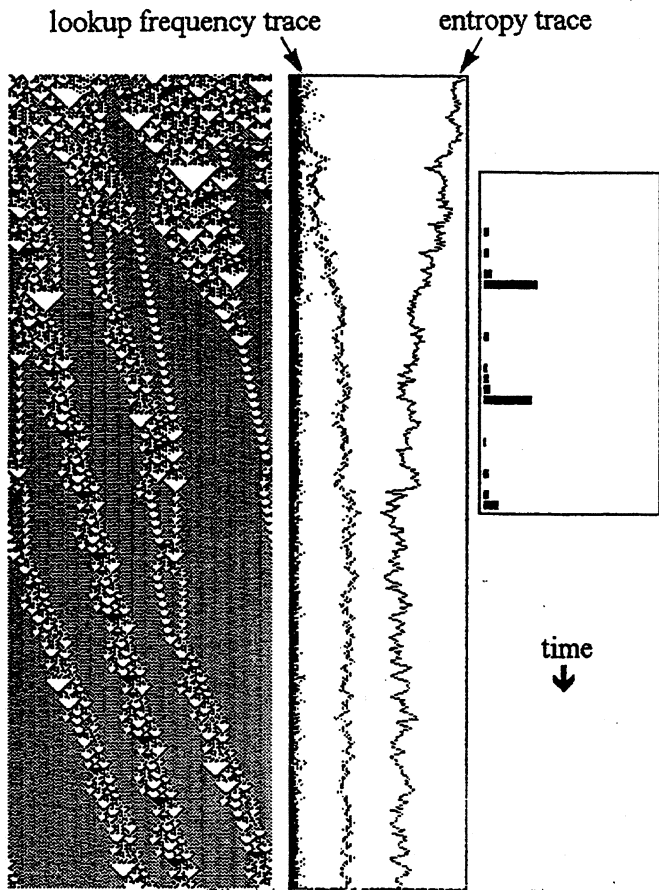
The following sections describe results based on a number of new measures. Firstly the variance of the input-entropy, a local measure on space-time patterns to distinguish glider rules, where the mean entropy distinguishes ordered and chaotic rules. Secondly, the new software tools available for constructing attractor basins and subtrees allows a close look at the relationship between basin topology and space-time patterns. Global measures on basin topology are provided by garden-of-Eden density and more generally by a histogram of in-degree distribution. These measures may be taken on the whole basin of attraction field for small systems. For large systems the measures are taken on subtrees or fragments of sub-trees. Local and global measures are related to each other and to the rule-parameter  $Z$ , which is defined in 4.12.  $Z$  is compared and related to the well known  $\lambda$  parameter (Langton 1990).

#### 4.9. Neighbourhood lookup frequency and input-entropy

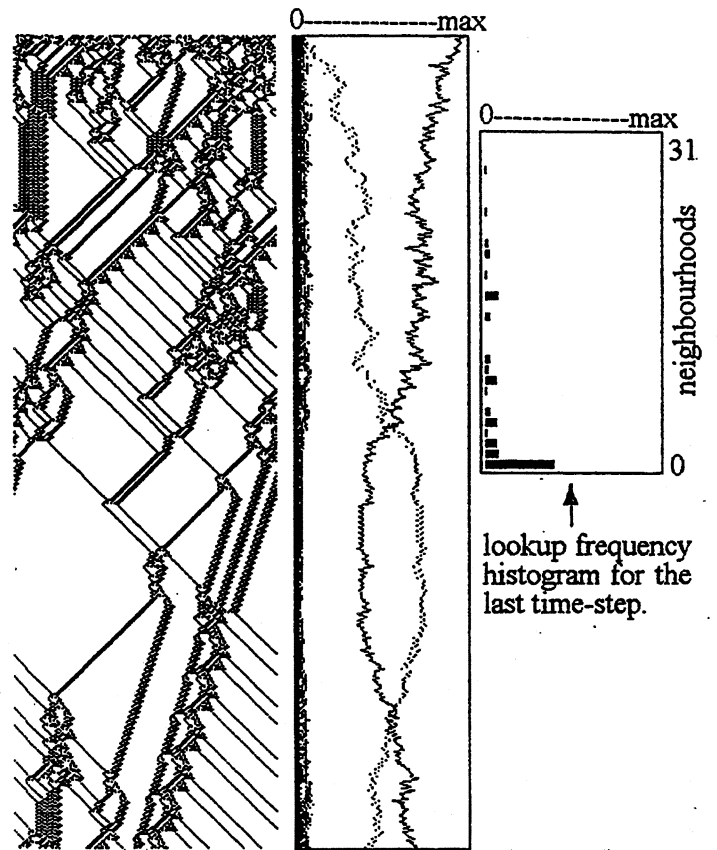
Gliders and backgrounds are built from a set of self sustaining configurations that emerge from an initial chaotic phase, crowding out all the other many possible configurations to dominate the CA's future behaviour. In an initial random seed all  $2^k$  neighbourhood configurations are equally probable. If randomly chosen, the seed is likely to be a garden-of-Eden state because garden-of-Eden states usually make up most of of state space.

**Figure 4.17.** (*opposite page*)

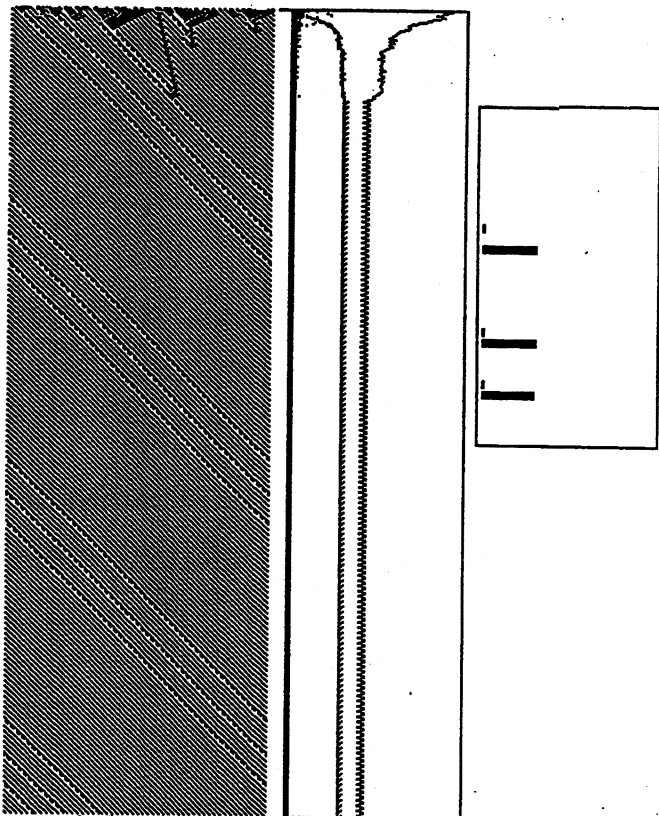
Two examples of glider dynamics (*top left* and *top right*), and an example of both ordered dynamics (*bottom left*) and chaotic dynamics (*bottom right*). System size 150 with periodic boundary conditions. 460 time-steps from a random seed. An example of the lookup frequency histogram is shown for the last time-step. A *superimposed* frequency spectrum is plotted alongside each time-step, with  $2^k$  points corresponding to the histogram values superimposed on one line. The entropy at each time-step is also shown on the same plot. Rule numbers are shown in hex.



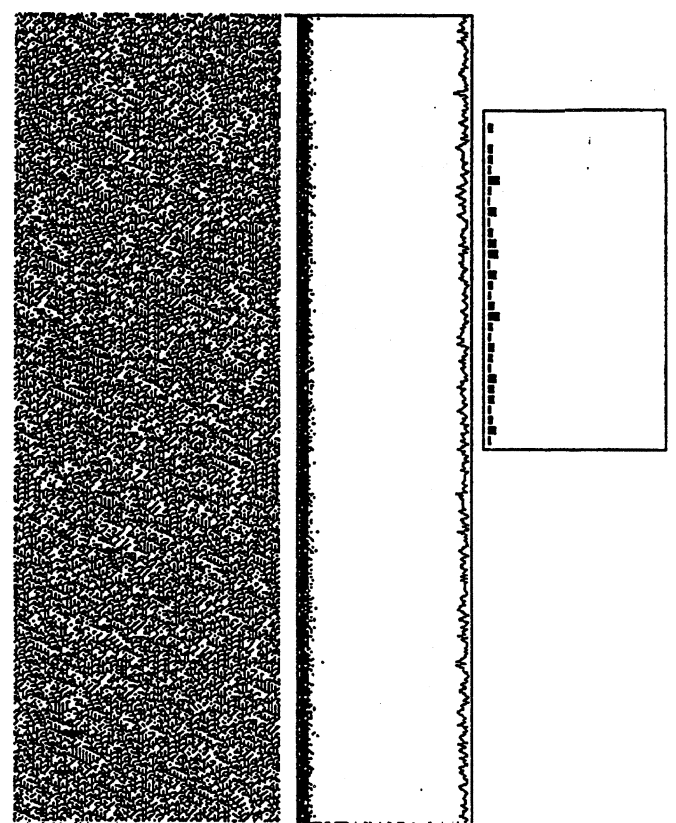
Glider  $K=5$  rule *b5 1e 9c e8*,  $\lambda_{ratio}=0.938$ ,  $Z=0.617188$



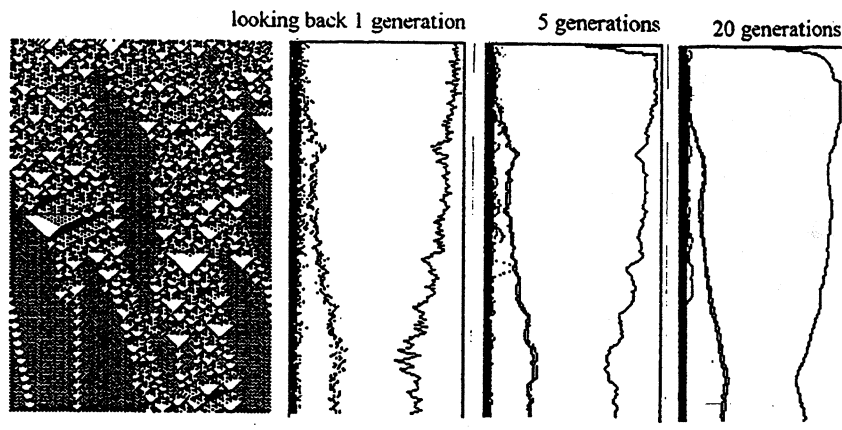
Glider  $K=6c$  1e 53 a8,  $\lambda_{ratio}=0.938$ ,  $Z=0.726562$



Ordered  $K=5$  rule *af d1 db 47*,  $\lambda_{ratio}=0.75$ ,  $Z=0.5625$



Chaotic  $K=bc$  7a 2a 65,  $\lambda_{ratio}=0.727$ ,  $Z=0.726562$



**Figure 4.18.** The frequency spectrum and its input-entropy plot shown alongside the space-time pattern from the same initial random seed, relative to one time-step (*left*), 5 time-steps (*centre*), and 20 time-steps (*bottom*).  $k=5$  rule,  $b5\ 1e\ 9c\ e8$ , as in figure 4.17 (*top left*).

As the CA evolves beyond the initial sorting out phase, some neighbourhoods will occur more frequently, others less. A special case of block probability and entropy (Wolfram 1984a) is input frequency and entropy. The frequency with which each of the neighbourhoods in the rule-table is "looked up" at a given time-step can be represented by a histogram, or lookup spectrum, as in figure 4.17, which distributes the total of  $n$  lookups among the  $2^k$  neighbourhoods (shown as the fraction of maximum lookups  $n$ , where  $n$ =system size,  $k$ =neighbourhood size).

The Shannon entropy of the lookup frequency histogram, the input-entropy  $S$  at time-step  $t$ , is given by

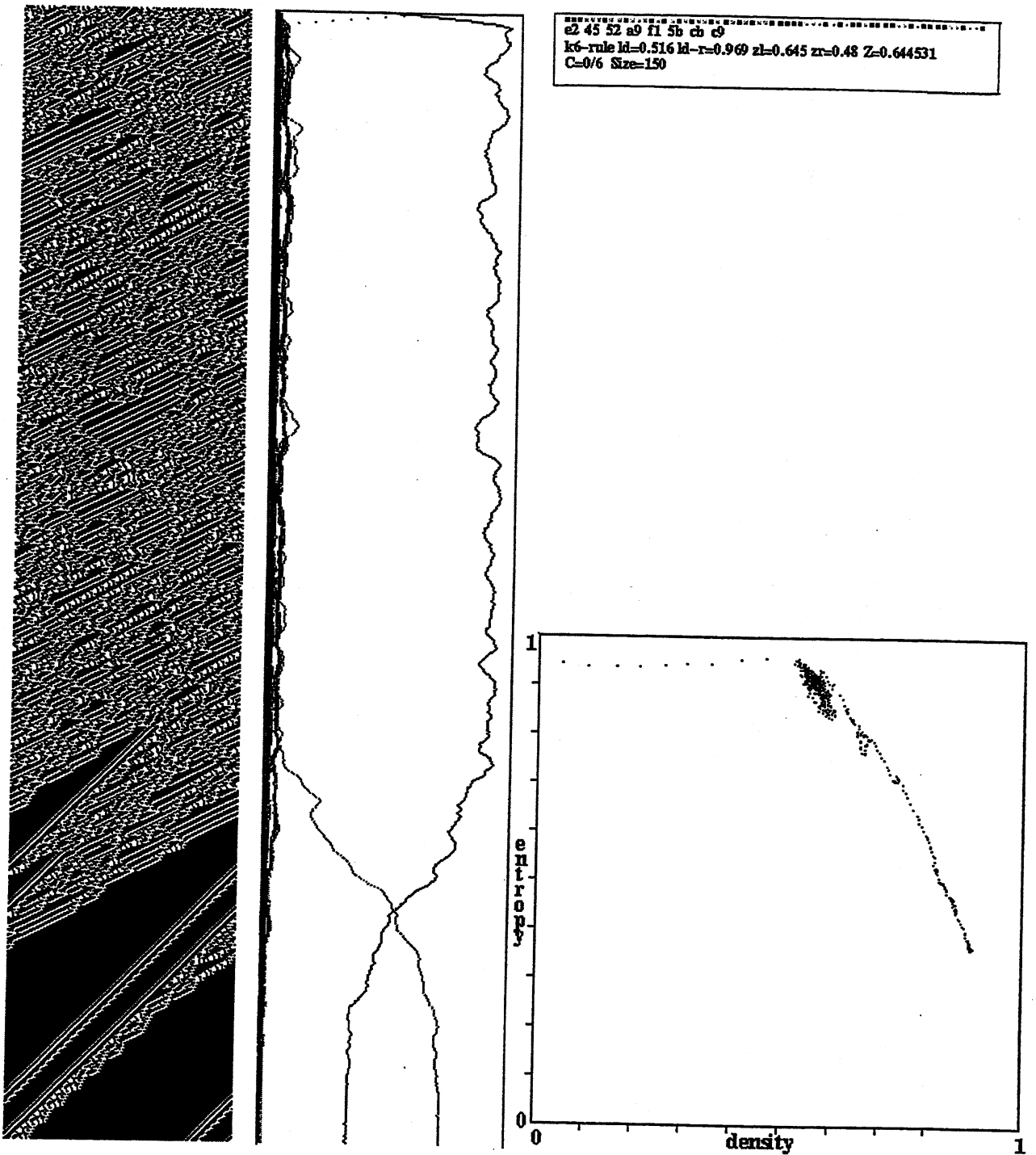
$$S^{(t)} = - \sum_{i=1}^{2^k} \left( \left( \frac{Q_i^{(t)}}{n} \right) \times \log \left( \frac{Q_i^{(t)}}{n} \right) \right)$$

Where  $Q_i^{(t)}$  is the lookup frequency of neighbourhood  $i$  at time  $t$ .

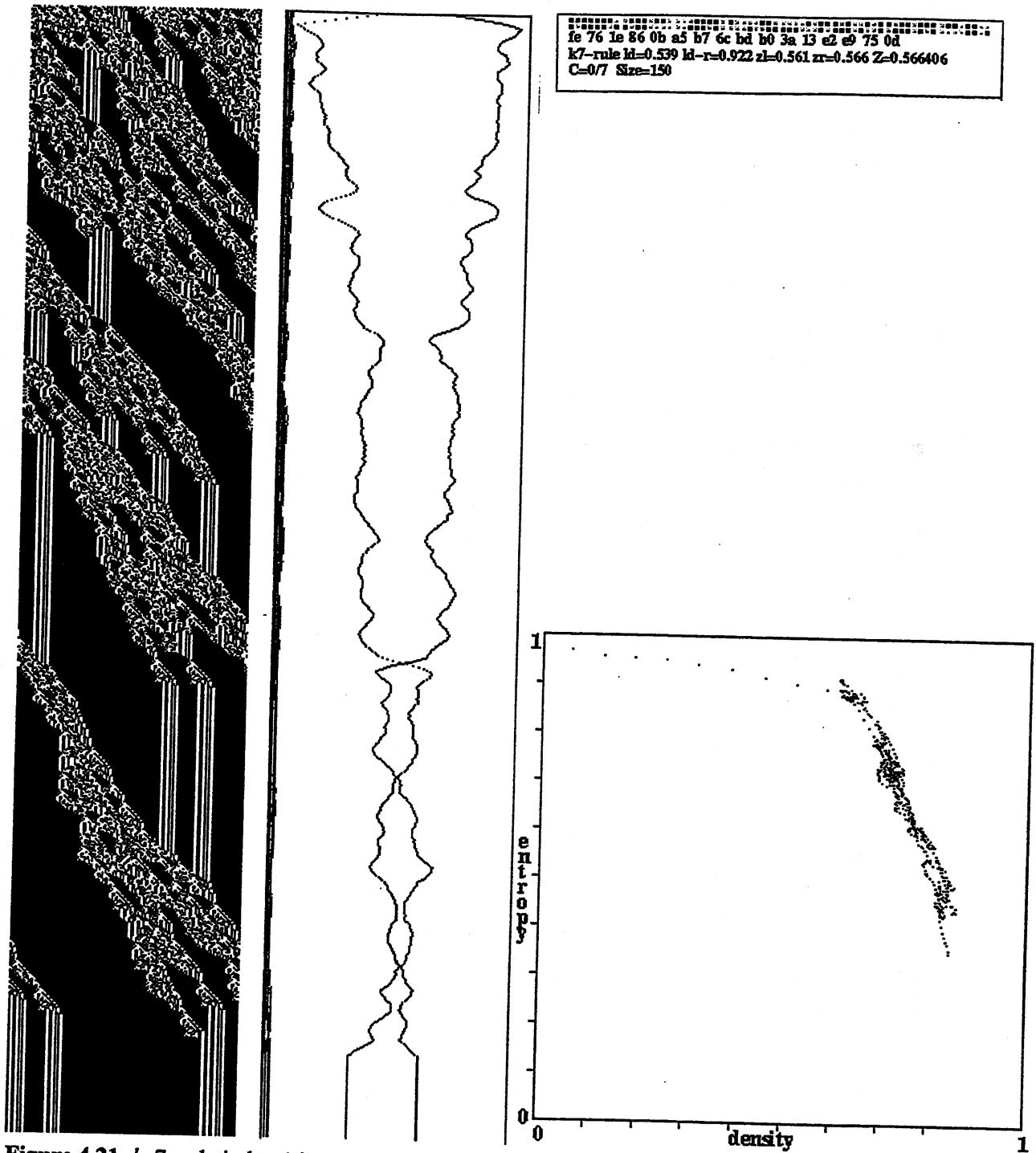
For a smoother measure the histogram is taken over a window of time-steps.

Figure 4.17 shows the space-time patterns of two complex rules with emerging gliders, and also an ordered and chaotic rule for comparison, evolving for 460 time-steps from a random initial state,  $n=150$ . Figures 4.19-4.21 show further examples from the automatic sample. The lookup frequency histogram is shown for the very last time-step. A *superimposed* frequency spectrum is plotted alongside each time-step in the space-time pattern, with  $2^k$  points corresponding to the top of the histogram bars superimposed on one line. The entropy of the lookup frequency spectrum is also shown for each time step on the same graph.





**Figure 4.20.**  $k=6$ . rule index 1 in the automatic sample with high entropy-variance. Measures are made relative to a moving window of 5 time-steps.  
*Left:* The space-time pattern  $n=150$  from a random initial state.  
*Centre:* The frequency spectrum and input-entropy plot.  
*Top right:* The rule number,  $\lambda$  and  $Z$  data.  
*Bottom right:* Input-entropy (vertical axis) plotted against the pattern density (the density of 1s).



**Figure 4.21.**  $k=7$ . rule index 1 in the automatic sample with high entropy-variance. Measures are made relative to a moving window of 5 time-steps.  
*Left:* The space-time pattern  $n=150$  from a random initial state.  
*Centre:* The frequency spectrum and input-entropy plot.  
*Top right:* The rule number,  $\lambda$  and  $Z$  data.  
*Bottom right:* Input-entropy (vertical axis) plotted against the pattern density (the density of 1s).

To compare these measures between ordered, complex and chaotic rules, an example of an ordered and a chaotic  $k=5$  rule is shown in figure 4.17(*bottom*), with the same lookup frequency and entropy data. If a  $k=5$  rule is selected at random, with a probability of 0.5 of setting a 0 or 1 for each rule table entry, the rule is likely to be either ordered or chaotic, although there is a significant chance that it might be a complex rule. The large number of ordered and chaotic rules from the automatic input-entropy samples (see below) confirm that these examples are typical. Note that as the neighbourhood  $k$  increases, a rule selected at random is more likely to be chaotic.

Because the initial state is set at random, given a big enough system size, the  $2^k$  bars in the lookup frequency histogram are likely to be distributed close together at low values. The different neighbourhoods occur with equal probability. The starting entropy will be correspondingly high. Below we discuss the characteristic evolution of the lookup frequency spectrum for successive iterations of the CA, and the input entropy, for ordered, chaotic and complex dynamics.

#### **4.9.1. Ordered Dynamics**

In ordered dynamics the lookup frequency spectrum will rapidly become highly unbalanced, with most neighbourhoods never looked at (their lookup frequency = 0). The few remaining high frequencies settle on constant or periodic values. The entropy will settle at a low constant or periodic value, corresponding to a fixed point or short cycle attractor. Ordered behaviour produces extremely short and bushy transient trees with a high  $G$ -density. Ordered rules decrease disorder and entropy.

#### **4.9.2. Chaotic Dynamics**

In chaotic dynamics, the lookup frequency spectrum will fluctuate irregularly within a narrow band of low values, and the entropy will fluctuate irregularly within a narrow high band, corresponding to dynamics on very long transients or cycles, analogous to strange attractors in continuous dynamical systems. Transient trees will be sparsely branched thus will tend to be very long with relatively low  $G$ -density. Chaotic rules increase or conserve disorder and entropy.

#### **4.9.3. Complex Dynamics**

In complex dynamics, the frequency spectrum becomes unbalanced, breaking up into high and low strands that exhibit large erratic fluctuations, reflected in large erratic fluctuations in entropy. As in ordered behaviour, a proportion of neighbourhoods are never looked at again after the initial sorting out phase; these neighbourhoods are, in a sense, leached out of the system. After an extended time,



(100s or 1000s of time-steps) the system generally settles onto a short attractor cycle. The high strands in the superimposed lookup spectrum result from the neighbourhoods making up the emergent background (or multiple high strands for multiple backgrounds). The low strands are the lookup frequencies of interacting gliders.

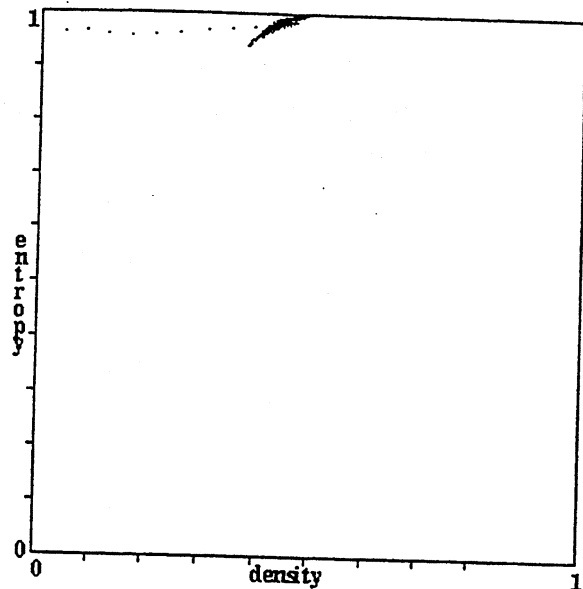
Glider dynamics is subject to two countervailing tendencies. On the one hand a tendency towards order because of the dominant periodic background(s), but the zones of order are mobile, their boundaries form the moving particles or gliders. When these collide there is a tendency toward chaos. The collisions may form a temporary zone of chaotic dynamics before new gliders emerge. In systems of the order of size considered here, order or chaos may predominate at different times causing the entropy to vary. For large networks where colliding and non-colliding zones co-exist, the entropy variance will be reduced, and could disappear in the limit of infinite size. Thus glider dynamics on a transient in relatively small networks can both increase and decrease entropy by large amounts in an erratic manner producing an erratic entropy curve. The entropy starts off high for the random seed and the initial sorting out phase. It then fluctuates erratically over a wide range of values during the glider interaction phase, and settles at a periodic or constant minimum level at the attractor cycle.

Appendix 3 shows some examples of complex rules and their input-entropy plots for  $k=5$ ,  $k=6$  and  $k=7$ . In these plots the measures are taken relative to each time-step. In subsequent results the measures are taken over a moving window of time-steps to smooth out the entropy curve and show only the longer term variation. This is illustrated in figure 4.18, which shows the  $k=5$  glider rule as in figure 4.17(*top-left*). From the same initial random seed, the frequency spectrum and its entropy plot is shown firstly relative to just one time-step, then for a moving window of 5, and finally for 20 time-steps.

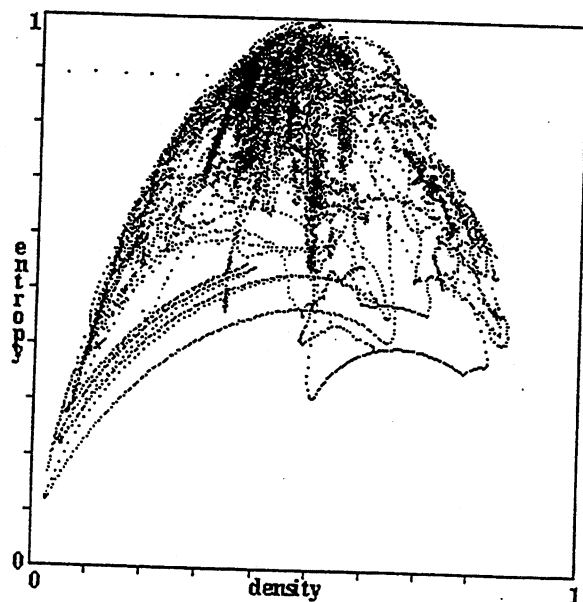
A measure of the variability of the input-entropy curve is its variance or standard deviation from the mean, high entropy variance is then a sure sign of complex space-time dynamics.

High entropy variance is not only characteristic of glider dynamics but also of other less frequent types of complex dynamics. For example two chaotic co-existing (competing) domains which are qualitatively different, having different block probabilities will produce an erratic entropy curve as one or the other domain becomes dominant. The boundary between the chaotic domains may be seen as a different type of particle lacking the regularity of a glider. Such particles may be isolated by filtering out the chaotic domains (Crutchfield and Hanson 1993). A related type of complex dynamics occurs when a chaotic domain co-exists with glider dynamics or with just a regular background as in figure 4.11.

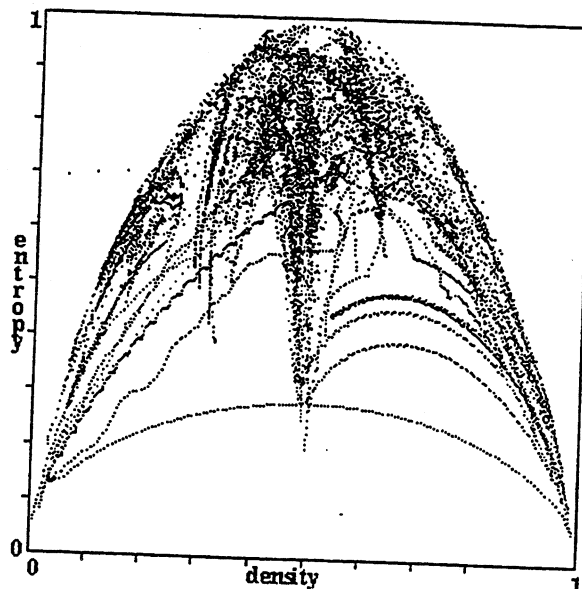
The chaotic rule *aa 55 66 a1* from figure 4.1(f). The scatter plot is concentrated at high entropy with low variance.



Some  $k=5$  complex rules (with high input-entropy variance) from the sample in appendix 2 and 3.



$k=5$  complex rules (with high input-entropy variance) from the automatic rule sample. Their space-time patterns are shown in figure 4.4. The rule numbers are shown in appendix 5.1.5, index 2-22.



**Figure 4.22.** Input-entropy (vertical axis) is plotted against the pattern density (the density of 1s).  $k=5$  rules,  $n=150$ . The measures are made relative to a moving window of 5 time-steps. *Top:* the chaotic rule *aa 55 66 a1* from figure 4.1(f). *Centre and bottom:* A number of superimposed plots of complex rules. Note the high variance of the input-entropy. Each rule produces a plot with its own distinctive "signature". Plots were made for trajectories of about 1000 time-steps, and from several random initial states for each rule.

Entropy variance is the basis of an automatic procedure for finding glider rules and related complex rules, whereas the mean entropy segregates rule-space between order and chaos, explained in section 4.10-11 below.

#### 4.10. Classifying rules by entropy-density

Entropy-density plots for a number of complex rules are shown in figure 4.19-4.22. Input-entropy (vertical axis) is plotted against the pattern density (the density of 1s). The plots may be from one or more random initial states, and the measures are made relative to a moving window of 5 time-steps. Each rule gives a characteristic cloud of points. For complex rules the clouds have a marked vertical extent because the input-entropy varies significantly. Each complex rule produces a plot with its own distinctive signature. By contrast, chaotic rules will give a flat compact cloud at high entropy. For ordered rules the plot also has a large vertical extent as the entropy falls off, but there are very few data points because the system moves very rapidly to an attractor.

Gutowitz (1995) has also shown entropy-density plots for large samples of rule-space, but his plots show a single point for each rule where the measures on that rule have settled down, whereas the plots shown here focus on the transient history of the system. These plots distinguish order, complexity and chaos by the vertical extent and density of the cloud.

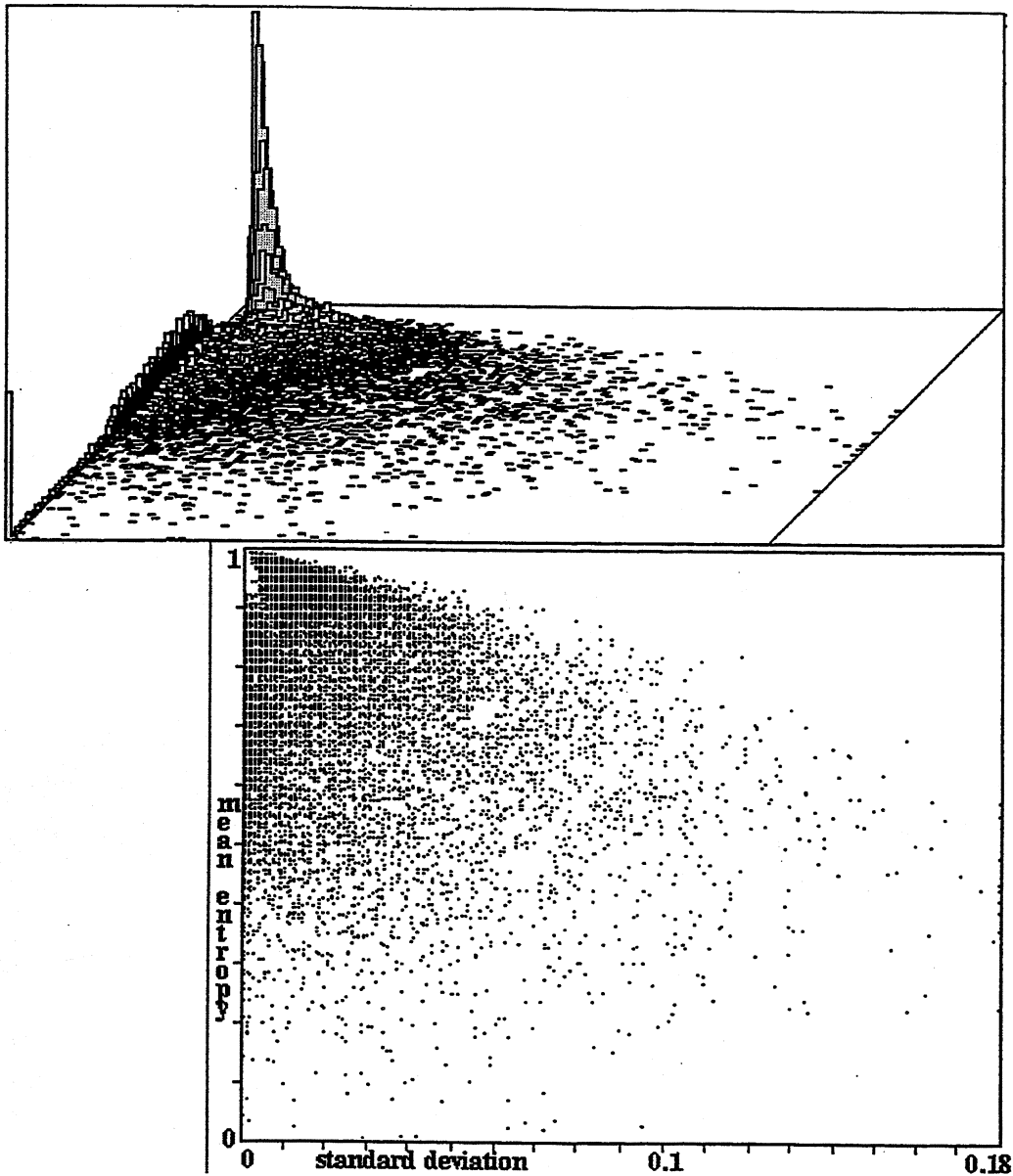
#### 4.11. Automatically classifying rule-space by input-entropy variance

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. The standard deviation is given by,

$$\sigma = \sqrt{\frac{\sum x_i^2}{n}} \quad \text{where } x_i = \text{deviation of each measure from the mean, and } n = \text{number of measures.}$$

The variance =  $\sigma^2$

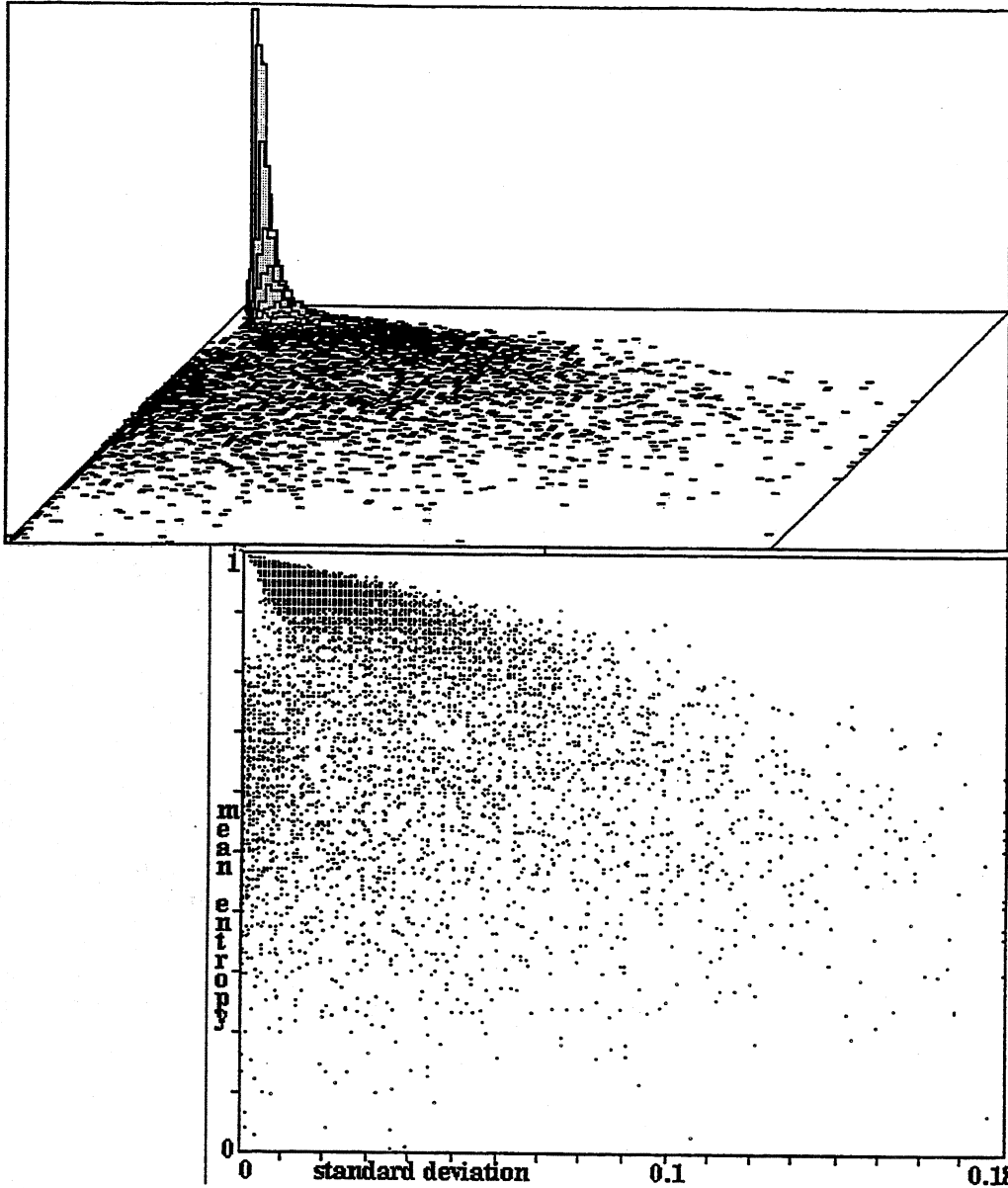
In the following results for  $k=5, 6$  and  $7$  rules, the input-entropy was measured over a window of 5 time-steps as the system was run for 430 time-steps from a random initial state. The measures were only taken into account for the last 400 time-steps, the first 30 were ignored to allow the system to evolve beyond the initial sorting out phase. The mean input-entropy, and the standard deviation from this mean, were calculated relative to these 400 time-steps. This procedure was repeated 5 times from different random initial states for each rule. The measures were averaged and a point was plotted of mean input-entropy ( $y$  axis) against standard deviation. Rules (and initial



**Figure 4.23.** *Below:* Mean entropy ( $y$ -axis) plotted against standard deviation for a random sample of 17680  $k=5$  rules,  $n=150$  (see section 4.11). *Above:* The distribution of points falling within a  $128 \times 128$  grid. Chaotic rules are concentrated in the top left "tower", ordered rules in the "ridge" close to the  $y$ -axis with lower mean entropy. Complex rules are spread in the area to the right with higher standard deviation.

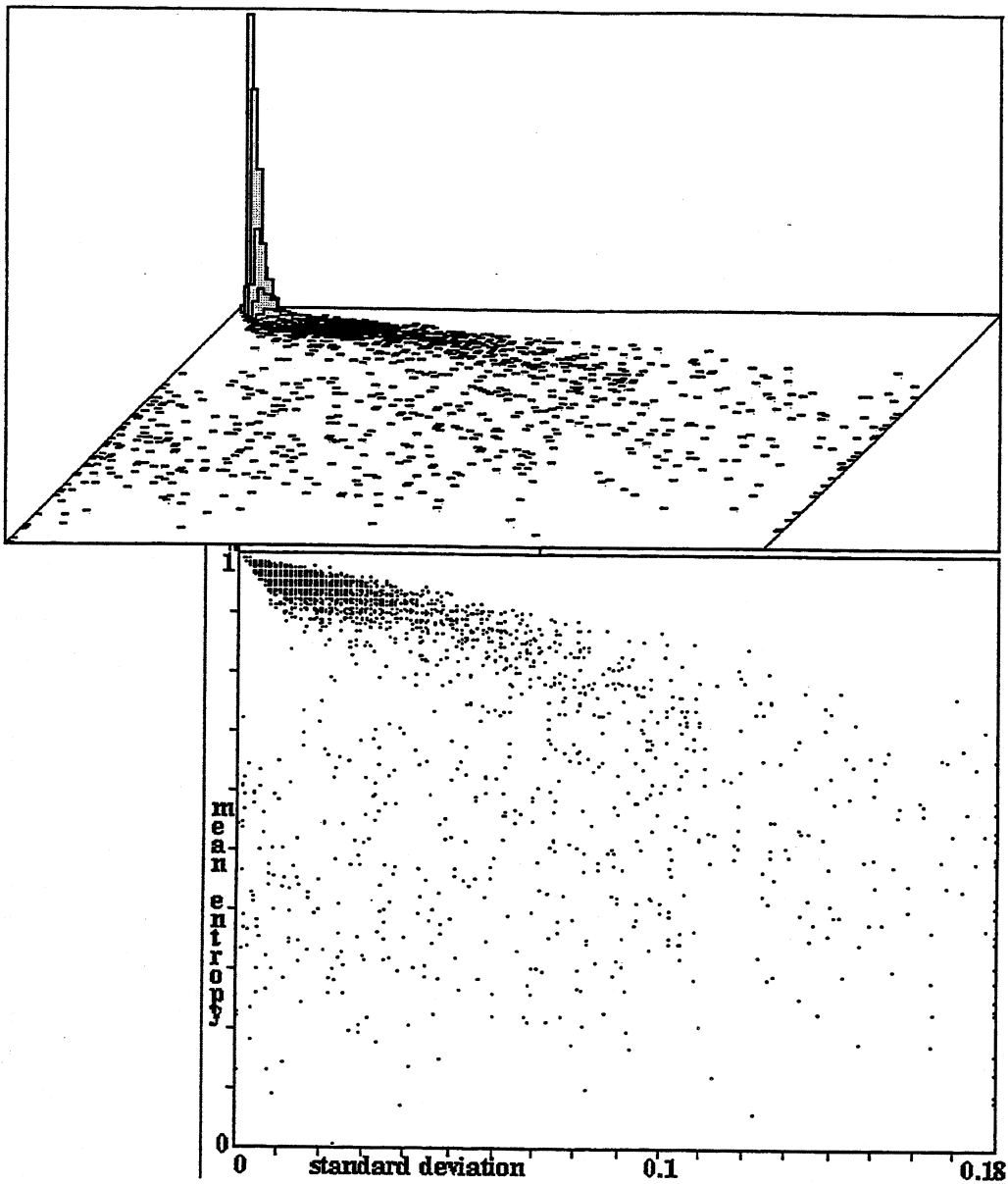
states) were selected at random by setting a 1 or 0 with equal probability for each entry in the rule's look-up table (and each bit in the initial state).

Figures 2.23-2.25 show the resulting scatter plots for random samples of  $k=5$ ,  $k=6$  and  $k=7$  rules. The sample sizes are as follows; 17680  $k=5$ , 15425  $k=6$ , 14221  $k=7$ . Each plot is accompanied by a 2d histogram showing the frequency distribution of points falling within blocks on a  $128 \times 128$  grid overlaid over the scatter plot.



**Figure 4.24.** *Below:* Mean entropy ( $y$ -axis) plotted against standard deviation for a random sample of 15425  $k=6$  rules,  $n=150$  (see section 4.11). *Above:* The distribution of points falling within a  $128 \times 128$  grid. Chaotic rules are concentrated in the top left "tower", ordered rules close to the  $y$ -axis with lower mean entropy. Complex rules are spread in the area to the right with higher standard deviation.

Looking at the  $k=5$  2d frequency histogram, the "tower" in the upper left represents chaotic rules with low standard deviation and high mean entropy. The ridge on the left represents ordered rules with low standard deviation and a spread of lower mean entropy. Rules to the right of the plot, with higher standard deviation are complex rules, though further work is required to specify a boundary with ordered and chaotic rules. Any standard deviation above the maximum scale has been



**Figure 4.25.** *Below:* Mean entropy ( $y$ -axis) plotted against standard deviation for a random sample of 14221  $k=7$  rules,  $n=150$  (see section 4.11). *Above:* The distribution of points falling within a  $128 \times 128$  grid. Chaotic rules are concentrated in the top left "tower", the few ordered rules are close to the  $y$ -axis with lower mean entropy. Complex rules are spread in the area to the right with higher standard deviation.

reset to the maximum of 1.8. The  $k=6$  and  $k=7$  plots show a greater frequency of chaotic rules and a declining frequency of ordered and complex rules as  $k$  increases.

The rule samples and measures, including the rule's  $\lambda$  and  $Z$  parameters, were saved to file sorted by decreasing standard deviation, and decreasing mean entropy for each measure of standard deviation. In appendix 5 examples of complex, chaotic and ordered rules from the sorted samples

are shown. The rule samples are also shown in relation to both the rule's  $\lambda$  and  $Z$  parameter ( $y$  axis) plotted against standard deviation.

To check whether the expected dynamics (recognised subjectively) corresponds to the measures as plotted, the dynamics of particular rules at different positions on the plots may be re-examined. This may be done efficiently using DDLab. Examples of dynamics from the plots are shown in figures 4.4-4.6 and in appendix 5. A preliminary scan indicates 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 pattern of typical trajectories. The distribution of the rule samples according to these local measures may be compared with *global* measures on attractor basin topology,  $G$ -density and the in-degree frequency (section 4.15). A preliminary scan indicates a strong relationship between these global measures and the rule sample input-entropy plots, but again a systematic investigation of the space is required.

#### 4.12. The $Z$ -parameter

Various parameters have been defined on CA rule tables, notably Langton's (1990)  $\lambda$  parameter and the equivalent idea of *internal homogeneity* introduced earlier by Walker (1966). An alternative parameter,  $Z$  (Wuensche 1992,1994b) for binary CA tracks behaviour more closely. Whereas  $\lambda$  simply counts the fraction of 1s in a binary rule table,  $Z$  takes into account the allocation of rule-table values to sub-categories of related neighbourhoods. It is derived from the algorithm for computing pre-images reviewed in section 4.4 and predicts the characteristic in-degree, the bushiness of sub-trees in attractor basins corresponding to the degree of convergence of the dynamical flow in state-space. It is thus a trajectory convergence parameter analogous to an inverse Liapouov exponent in continuous dynamics. The DDLab source code for calculating  $Z$  is shown in appendix 6.4.

As well as predicting convergence, an example of *global* behaviour,  $Z$  gives a good indication of at least the extremes of *local* behaviour between order and chaos, both subjectively and as characterised by input-entropy variance. If a given rule table is tuned through its range of  $Z$  values by small mutations, a progressive change in behaviour between order and chaos is observed. Figure 4.1 shows an example. Suppose that we know part of a pre-image (a partial pre-image) of an arbitrary CA state, and attempt to deduce the value of successive cells from *left* to *right*. The  $Z_{left}$  parameter is the probability that the next unknown cell to the right in the partial pre-image has a unique value, and is calculated directly from the CA rule-table by counting *deterministic* neighbourhoods, defined in the next section.  $Z_{right}$  is the converse, from *right* to *left*. The  $Z$  parameter itself is the greater of  $Z_{left}$  and  $Z_{right}$ .

If a partial pre-image ends with a pattern corresponding to a deterministic neighbourhood, then the next unknown cell must have a unique value, thus preventing the partial pre-image from bifurcating into two valid partial pre-images, and restricting the potential for growth in the number of pre-images. This is the basis of the reverse algorithm for determining pre-images (Wuensche and Lesser 1992a) reviewed in section 4.4, where the leading edge of each partial pre-image is continuously checked for a match with a deterministic neighbourhood.

$Z$  is a probability, so varies between 0 and 1. If  $Z$  is high, the number of pre-images of an arbitrary CA state is likely to be small relative to system size. For  $Z=1$  it was shown in (Wuensche and Lesser 1992a) that the maximum in-degree cannot exceed  $2^{k-1}$ , and if only one of  $Z_{left}$  or  $Z_{right}=1$ , maximum in-degree must be smaller than  $2^{k-1}$ . Conversely, if  $Z$  is low, the in-degree is likely to be relatively high. For  $Z=0$ , all state space becomes a single pre-image fan converging on the state all-0s or all-1s in one step.

#### 4.13. Calculating the $Z$ -parameter

A general procedure to calculate  $Z$  from the rule table is described below\*.

Let  $n_k$  = the count of rule-table entries belonging to deterministic *pairs* of neighbourhoods such that,

$$\begin{array}{ll} \text{the neighbourhood,} & a_1, a_2, \dots, a_{k-1}, 1 \rightarrow T \dots (\text{its output}) \\ \text{and} & a_1, a_2, \dots, a_{k-1}, 0 \rightarrow \text{not } T \end{array}$$

The probability that the *next cell* is determined because of the above is given by,  $R_k = n_k / 2^k$

This is a first approximation of  $Z_{left}$ , as deterministic *pairs* occur with the highest probability in a random rule-table.

Let  $n_{k-1}$  = the count of rule-table entries belonging to deterministic *4-tuples* of neighbourhoods (where "\*" is a wildcard value, 0 or 1 with equal validity) such that,

$$\begin{array}{ll} \text{the neighbourhood,} & a_1, a_2, \dots, a_{k-2}, 1, * \rightarrow T \text{ (its output)} \\ \text{and} & a_1, a_2, \dots, a_{k-2}, 0, * \rightarrow \text{not } T \end{array}$$

The probability that the *next cell* is determined because of the above is given by,  $R_{k-1} = n_{k-1} / 2^k$

Let  $n_{k-2}$  = the count of rule-table entries belonging to deterministic *8-tuples* of neighbourhoods such that,

$$\begin{array}{ll} \text{the neighbourhood,} & a_1, a_2, \dots, a_{k-3}, 1, *, * \rightarrow T \text{ (its output)} \\ \text{and} & a_1, a_2, \dots, a_{k-3}, 0, *, * \rightarrow \text{not } T \end{array}$$

The probability that the *next cell* is determined because of the above given by,  $R_{k-2} = n_{k-2} / 2^k$

---

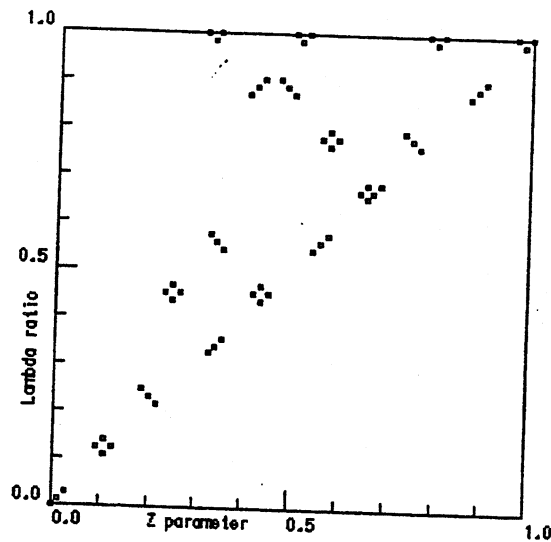
\*Refer to (Wuensche and Lesser 1992a) for a fuller discussion relating specifically to  $k=3$  and  $k=5$  rules.







**Figure 4.27.**  
 The relationship between  $\lambda_{\text{ratio}}$  and the  $Z$  parameter for the set of 256  $k=7$  totalistic rules. Because of rule table symmetries this number reduces to 136 non-equivalent rules belonging to 72 clusters having equal  $\lambda_{\text{ratio}}$  and  $Z$ .



#### 4.14. Relating the $\lambda$ parameter and $Z$

$\lambda$  is the proportion of non-quiescent entries in a CA rule table, where 0s are usually taken as quiescent. In a binary CA  $\lambda$  is the proportion of 1s. The  $\lambda$  parameter and  $Z$  are related because  $\lambda$  indicates the probability of the value of  $Z$ . Exceptions to  $\lambda$ 's predictions of CA behaviour (see figure 4.35) are due to the fact that  $\lambda$  measures just the proportions of the different values in the rule table irrespective of their distribution.  $\lambda_{\text{ratio}} = 2 \times$  the ratio of 0s or 1s in the rule table, whichever is the less. This normalises the  $\lambda$  parameter, making 0s and 1s equivalent for direct comparison with  $Z$ . The three graphs in figure 4.26 show  $\lambda_{\text{ratio}}$  plotted against  $Z$  for a sample of  $k=5, 7$  and  $9$  rules. 2200 random rules are plotted for each  $k$ . The rule sample is biased to include a representative spread of  $\lambda_{\text{ratio}}$  values by varying the probability of setting a 1 in the rule table from 0.05 to 0.5, in steps of 0.05. An equal mix of 0s and 1s gives the highest probability of high  $Z$  values at  $\lambda_{\text{ratio}} = 1$  ( $\lambda = 0.5$ ).

The graphs show a cloud of points diverging above the  $x=y$  diagonal, closer to it for lower  $\lambda_{\text{ratio}}$ , but with  $Z \leq \lambda_{\text{ratio}}$ . Points do occur between the cloud and the  $x=y$  diagonal in the graphs of  $k=7$  and  $k=9$  rules as well as  $k=5$  rules, but become more rare to the right of the cloud as  $k$  (and the rule-table size  $2^k$ ) increases, so are unlikely to be seen in this area when rules are set at random because they depend on unlikely distributions of 0s and 1s in the rule table. This explains why the  $\lambda_{\text{ratio}} - Z$  relationship as shown is more focused towards the left edge of the cloud for larger  $k$ . Note also that a rule for a given  $k$  has an equivalent rule-table at any larger  $k$ , so the points on the  $k=5$

graph could legitimately be added to the  $k=7$  and  $k=9$  graphs. High  $Z$  values may be evolved (see 4.17), but this was not done in these  $\lambda_{\text{ratio}} - Z$  graphs. For smaller  $k$  more points on the graph are superimposed as both  $Z$  and  $\lambda_{\text{ratio}}$  may take on specific values only, with a smaller number of values for smaller  $k$ .

Figure 4.27 shows  $\lambda$  plotted against  $Z$  for the set of totalistic  $k=7$  rules, of which there are  $2^{k+1}=256$ . Rule table symmetries produce sets of 4 rules with equivalent space-time dynamics, the equivalence class<sup>1</sup>, consisting of the start pattern, its negative, its reflection, and its negative-reflection, though some of these equivalents may be identical. It turns out that the 256 totalistic<sup>2</sup>  $k=7$  rules may be reduced to 136 non-equivalent rules belonging to 72 clusters (the equivalent rules and their rule table complements) having equal  $\lambda_{\text{ratio}}$  and  $Z$ , as well as other measures related to their attractor basins. This set of rules has a conveniently even spread over the values of both  $\lambda_{\text{ratio}}$  and  $Z$ .

#### 4.15. Attractor Basin Topology; $G$ -density and the In-Degree Frequency

For CA rules in general, the quality of dynamical behaviour from ordered to chaotic is reflected by the bushiness of subtrees. This characterises the degree of convergence of dynamical flow seen in attractor basins, or the degree of dissipation viewing the CA as a dissipative dynamical system.

Consider a transient sub-tree with  $n$  nodes linked by  $n-1$  edges. In the space of all possible topologies there are two extreme cases. Maximum convergence occurs where  $n-1$  nodes converge in one step onto a single node. Here the garden-of-Eden density is close to 1. Minimum convergence occurs where the nodes are strung out in a chain with pre-imaging never exceeding one. Here the garden-of-Eden density is close to 0. Between these two extremes there lies a spectrum of degrees of convergence characterised by the topology of typical transient trees in terms of their "bushiness". High convergence implies short and dense trees with branching points having a high in-degree, signifying order. Low convergence implies long sparse trees, with branching points having a low in-degree, signifying chaos. Figures 4.28 - 4.31 show some examples.

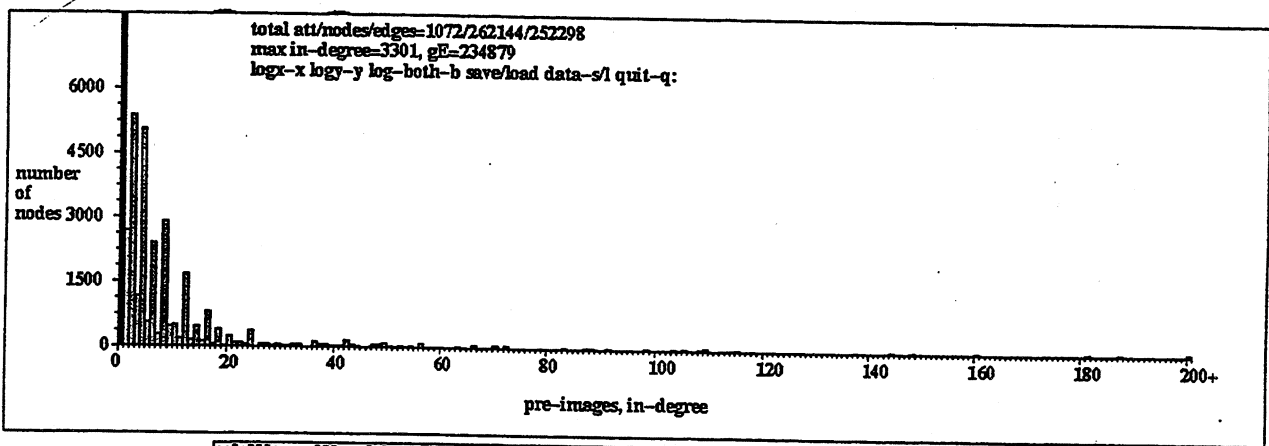
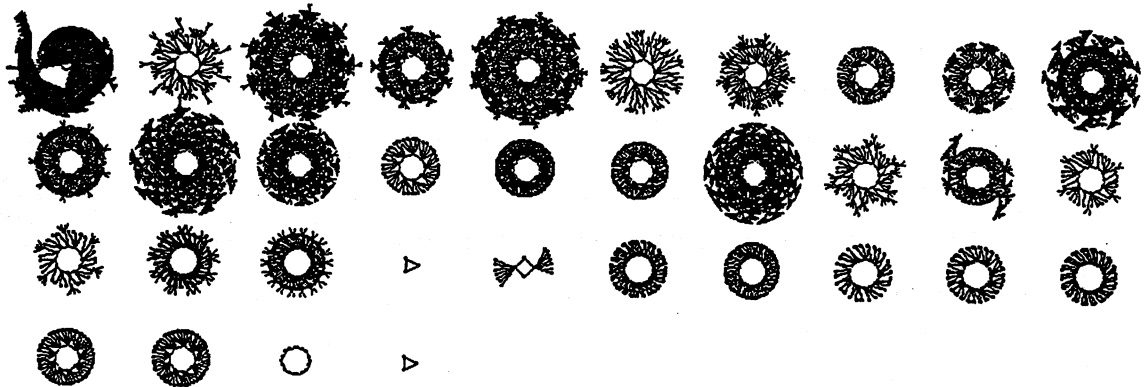
##### 4.15.1. $G$ -density

A simple measure that captures the degree of convergence is the density of garden-of-Eden states ( $G$ -density) counted in attractor basins or sub-trees, the rate of increase of  $G$ -density with  $n$  is a further indication. Because average in-degree (including in-degrees of zero) must equal one for any basin of attraction, high  $G$ -density (and a high rate of increase) signifies high convergence, low

<sup>1</sup>See (Wuensche and Lesser 1992a) for a discussion of equivalence classes and rule clusters.

<sup>2</sup>This should not to be confused with the 256  $k=3$  rules, which reduce to 88 equivalence classes and 48 clusters (Wuensche and Lesser 1992a).

basins types=34 total basins=71  
 field=262144 gE=234879 density=0.896 2min 48.903sec



k5-rule(dec)772679680 (hex)2e 0e 28 00  
 Size=18 ld=0.281 ld-r=0.562 P=0.719 zl=0.312 zr=0.398 Z=0.398438 C=0/5

**Figure 4.28.** The basin of attraction field, and its in-degree frequency histogram, of the ordered  $k=5$  rule  $2e\ 0e\ 28\ 00$ .  $n=18$ . Its space-time patterns are shown in figure 4.1(a). Note high  $G$ -density (0.896, off the scale), high in-degree (maximum=3301, off the scale) and short transients.

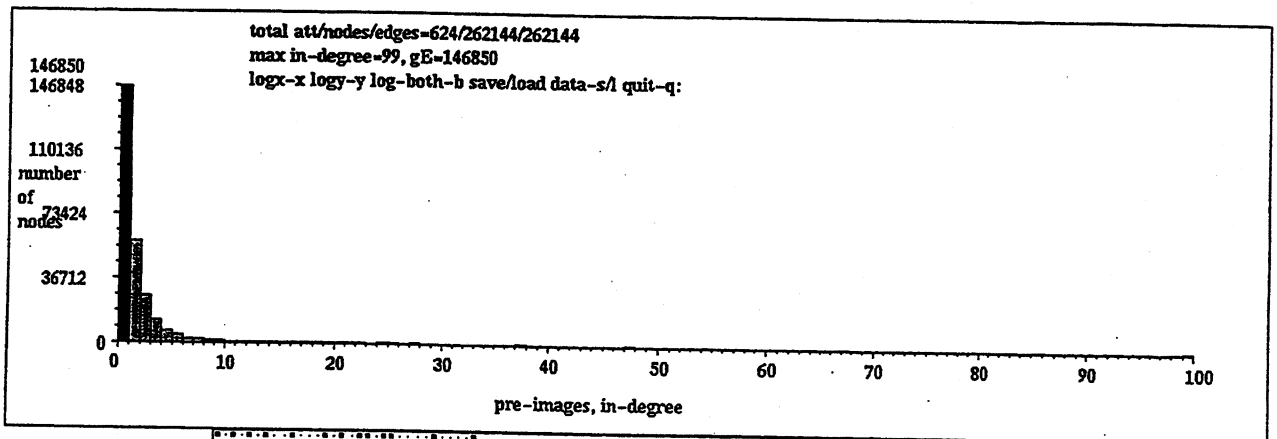
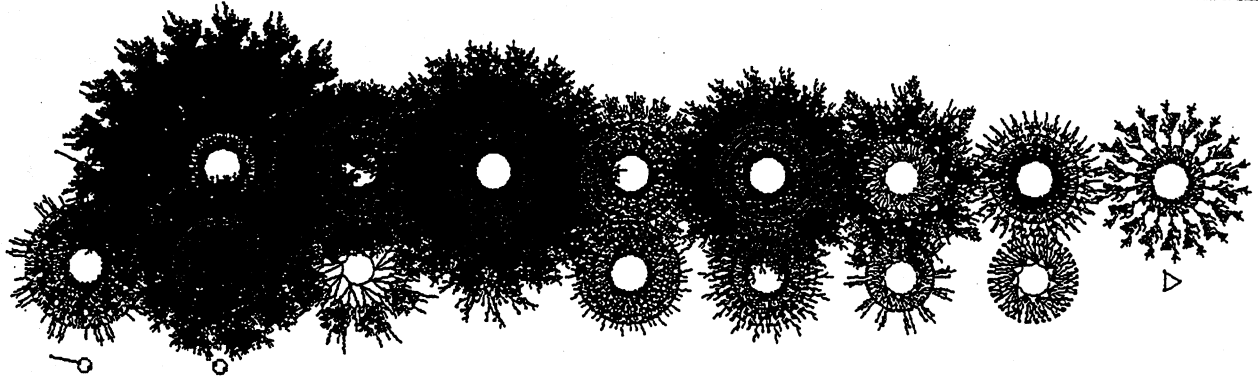
$G$ -density (and a low rate of increase) signifies low convergence. High and low convergence in turn indicate ordered and chaotic dynamics.

#### 4.15.2. In-Degree Frequency.

To gain further insights into how the convergence typical of a rule is structured, the in-degree frequency distribution of the basin of attraction field for small systems, or of a subtree for larger systems, may be plotted as a histogram. 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.

Figures 4.28 - 4.30 show the entire basin of attraction fields and corresponding in-degree histograms, for system size 18, of the ordered, complex and chaotic rules, whose typical space-time

basins types=20 total basins=28  
 field=262144 gE=146850 density=0.56 10min 54.646sec



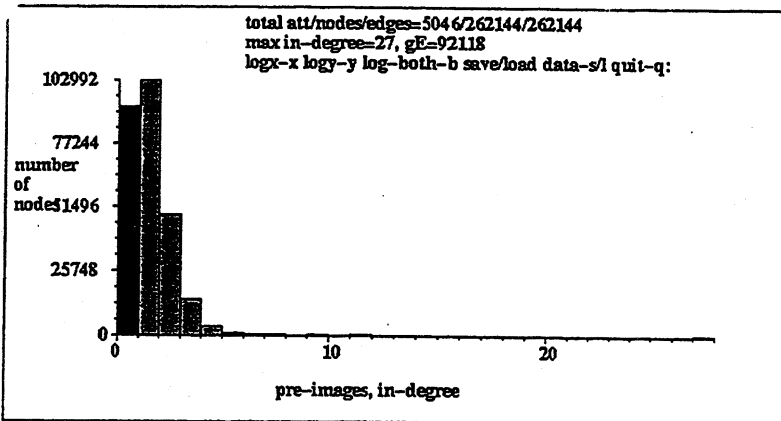
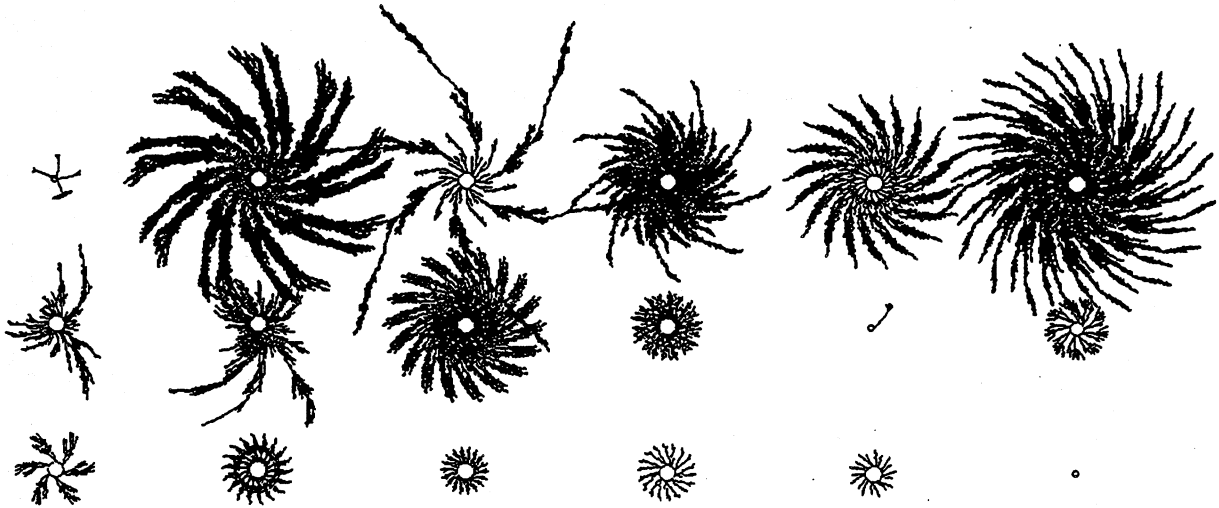
k5-rule(dec)2856676385 (hex)aa 45 6c 21  
 Size=18 ld=0.406 ld-r=0.812 P=0.594 zl=0.727 zr=0.578 Z=0.726562 C=0/5

**Figure 4.29.** The basin of attraction field, and its in-degree frequency histogram, of the complex  $k=5$  rule *aa 45 6c 21*.  $n=18$ . Note intermediate  $G$ -density (0.56), in-degree (maximum 99) and transient length. See figure 4.31 for the in-degree histogram of a subtree of this rule for  $n=50$ . Its space-time patterns are shown in figure 4.1(d).

patterns are shown in figure 4.1(a, d, and f). The distribution of in-degree is a finer measure of attractor basin topology and convergence than  $G$ -density alone, and the in-degree histogram gives characteristically different profiles for order, complexity and chaos.

Gliders can only interact fleetingly in such small systems. The in-degree histogram needs to be computed for much larger systems, where glider interactions can develop fully, to see to what extent these profiles are typical and independent of size. Subtrees for much larger systems may be computed, but only for complex and chaotic rules where the in-degree remains within reasonable bounds. For ordered rules the in-degree becomes astronomically large with large system size thus beyond the bounds of computation. However, it is a plausible assumption that the characteristic in-degree histogram profile for ordered rules does not change as the system size increases, so can be inferred from smaller systems.

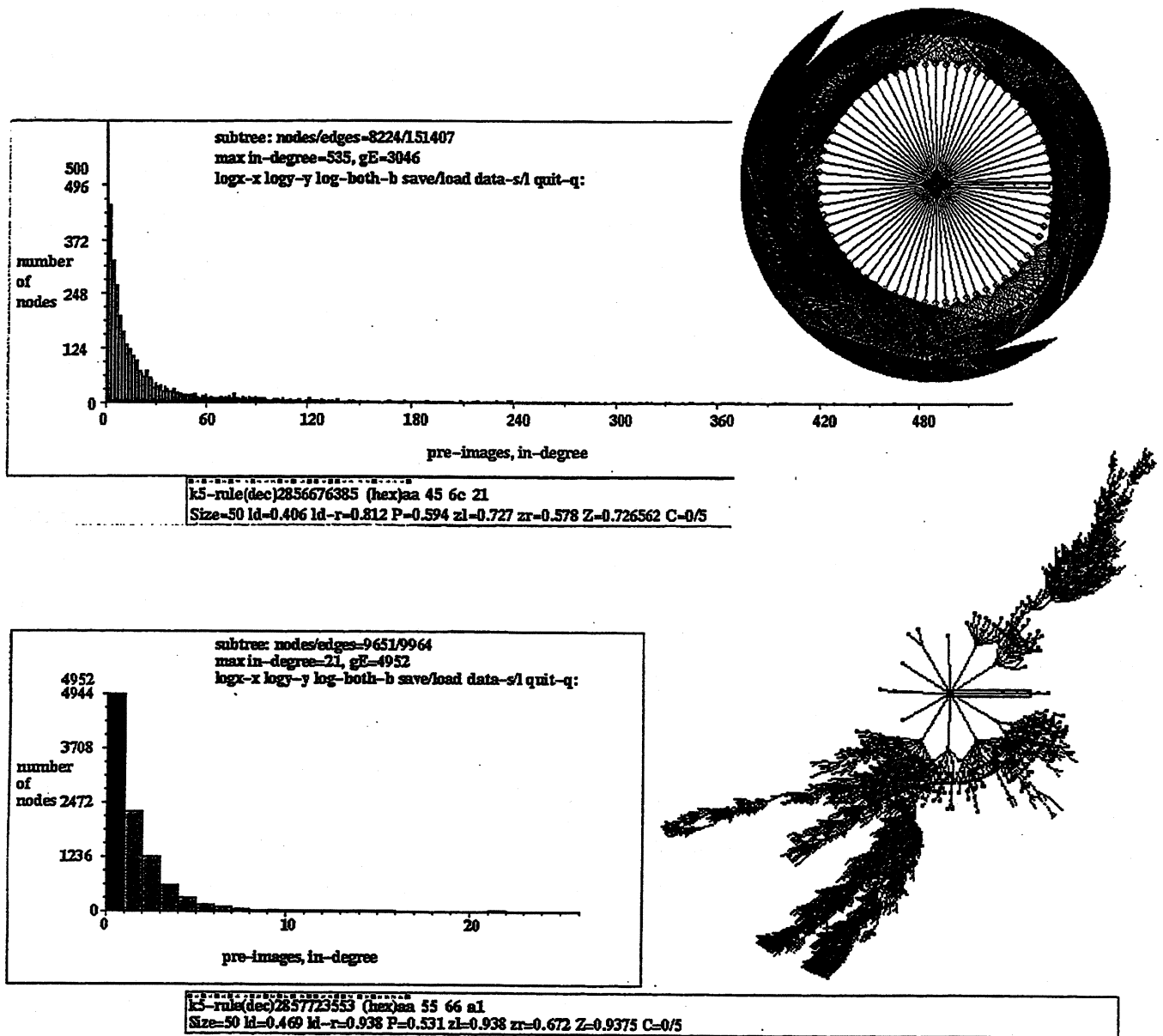
basins types=18 total basins=64  
 field=262144 gE=92118 density=0.351 2min 23.335sec



k5-rule(dec)2857723553 (hex)aa 55 66 a1  
 Size=18 kd=0.469 kd-r=0.938 P=0.531 z=0.938 zr=0.672 Z=0.9375 C=0/5

**Figure 4.30.** The basin of attraction field, and its in-degree frequency histogram, of the chaotic  $k=5$  rule  $aa\ 55\ 66\ a1$ ,  $n=14$ . Note low  $G$ -density (0.351), low in-degree (maximum 27) and long transients. See figure 4.28 for the in-degree histogram of a subtree of this rule for  $n=50$ . Its space-time patterns are shown in figure 4.1(f).

In figure 4.31 the in-degree histogram of subtrees for  $n=50$  are shown for typical  $k=5$  complex and chaotic rules in figure 4.1 (d and f), also shown for the whole basin of attraction field in figures 4.29 - 4.30 for  $n=18$ . In future work a systematic survey of the in-degree histograms of subtrees, based on the automatic samples described in section 4.11, is proposed. Randomly selected rules from representative positions in the mean entropy/standard deviation scatter plot will be surveyed. It is also proposed to survey in-degree histograms based on the subtrees of particular trajectories, especially in relation to glider dynamics.

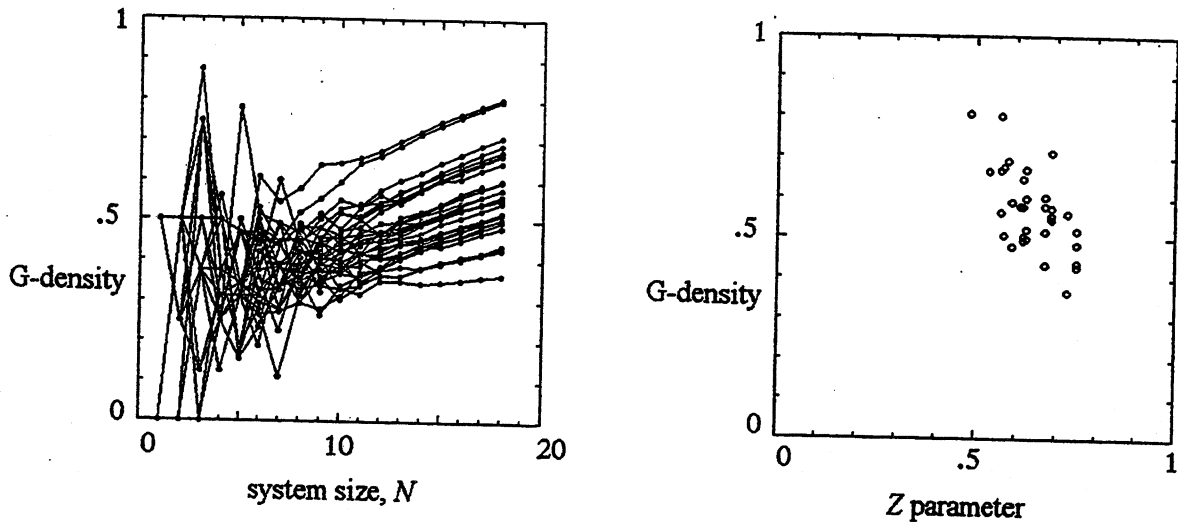


**Figure 4.31.** In-degree histograms of subtree fragments,  $n=50$ . The systems was run forward 50 time-steps from a random initial state, and the subtrees were generated from the state reached.

*Top:* The in-degree frequency histogram of the complex  $k=5$  rule *aa 45 6c 21* (rule (d) in figure 4.1) Note  $G$ -density (0.37, off the scale) and maximum in-degree 535. See figure 4.26 for the in-degree histogram of the basin of attraction field of this rule for  $n=18$ . *Inset right:* The subtree so far, with pre-images computed for 8224 nodes.

*Bottom:* The in-degree frequency histogram of the chaotic  $k=5$  rule *aa 45 6c a1* (rule (f) in figure 4.1). Note  $G$ -density (0.57) and maximum in-degree 21. See figure 4.27 for the in-degree histogram of the basin of attraction field of this rule for  $n=14$ . *Inset right:* The subtree so far, with pre-images computed for 9651 nodes.





**Figure 4.32.** (left)  $G$ -density plotted against  $n$  for  $n=1$  to 18, showing superimposed plots for the 38  $k=5$  rules in appendix 2, which gives the same plots separately. The  $G$ -density is based on the complete basin of attraction field. (right)  $G$ -density as above for  $n=18$  plotted against the  $Z$  parameter.

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

*Ordered rules:* A bi-modal distribution, very high garden-of-Eden frequency and significant frequency of high in-degrees.

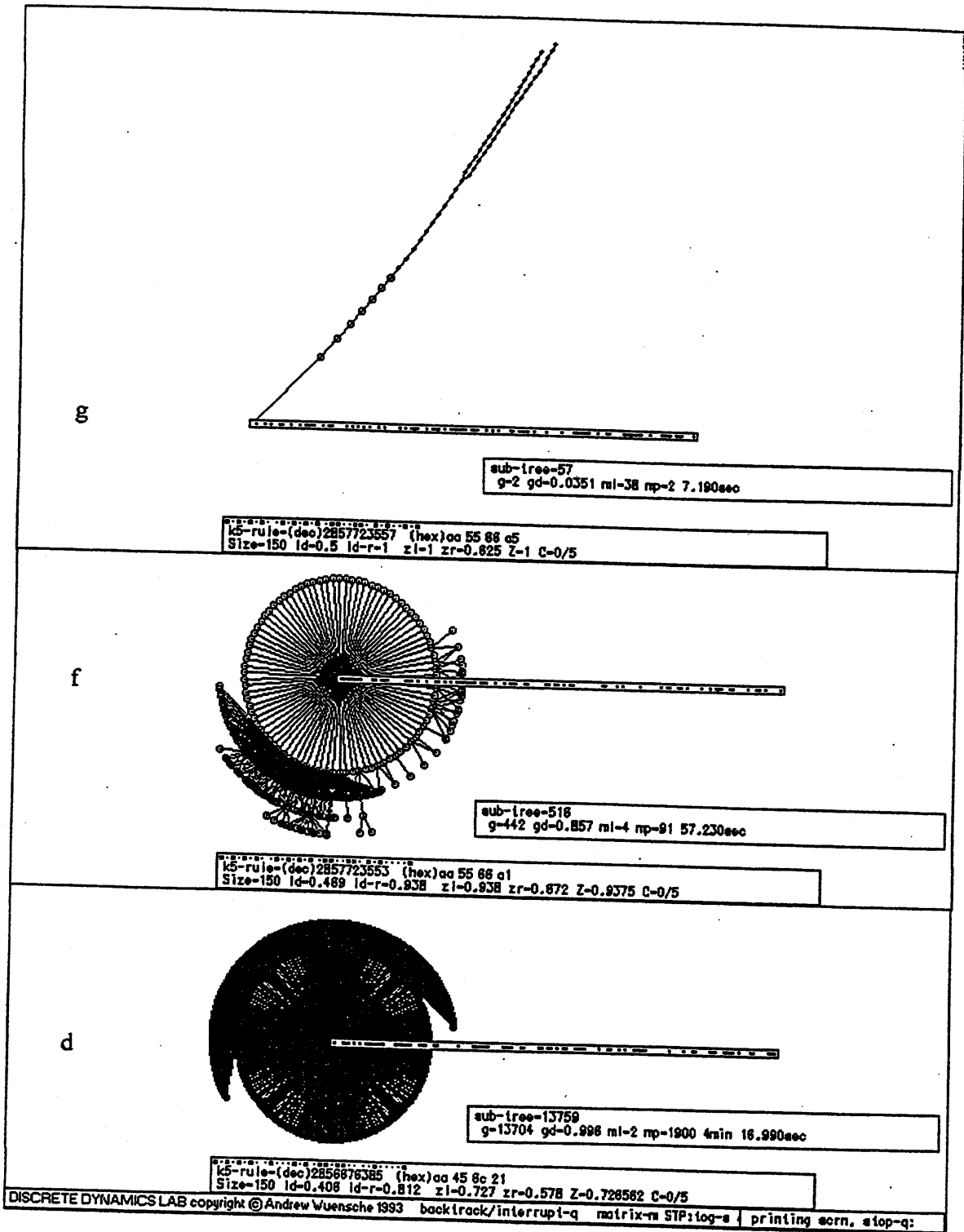
*Complex rules:* A power law distribution.

*Chaotic rules:* A relatively lower garden-of-Eden frequency compared to ordered and complex rules, and a high frequency of very low in degrees.

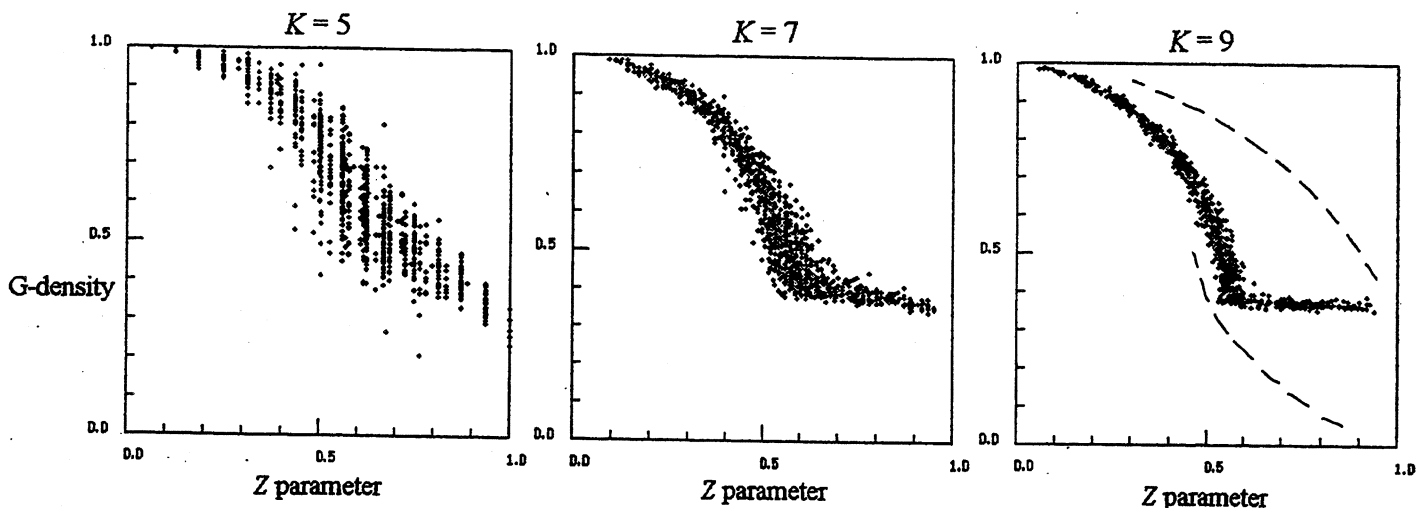
#### 4.16. $G$ -density - variation with system size

The variation of garden-of-Eden density ( $G$ -density) with system size  $n$ , as  $n$  is increased from 1 to 18 has been plotted for the  $k=3$  (elementary) rules, the  $k=5$  totalistic rules, and also for the sample of glider rules in appendix 2. For each rule, the complete basin of attraction field was generated for each value of  $n$ , and a count made of the number of garden-of-Eden states. These graphs are shown in appendix 1. A superimposed plot for the sample of glider rules in appendix 2 is shown in figure 4.32 (left).

The graphs in appendix 1 are presented in order of increasing  $Z$ . After an initial unstable phase for low  $n$  values, the plot generally stabilises; the  $G$ -density increases but to a progressively smaller



**Figure 4.33.** A subtree for a lattice size  $n=150$ . for the chaotic rules (*g*) and (*f*), and the glider rule (*d*), in figure 4.1 (*f* and *d* are also featured in figures 4.29 - 4.31 showing the entire field for  $n=14$ ). The subtree size, number of garden-of-Eden nodes (*g*), *G*-density (*gd*), and maximum in-degree (*mp*) are noted. The same random seed was run forward by 2 steps, and the subtrees were generated from the state reached, which is shown graphically.

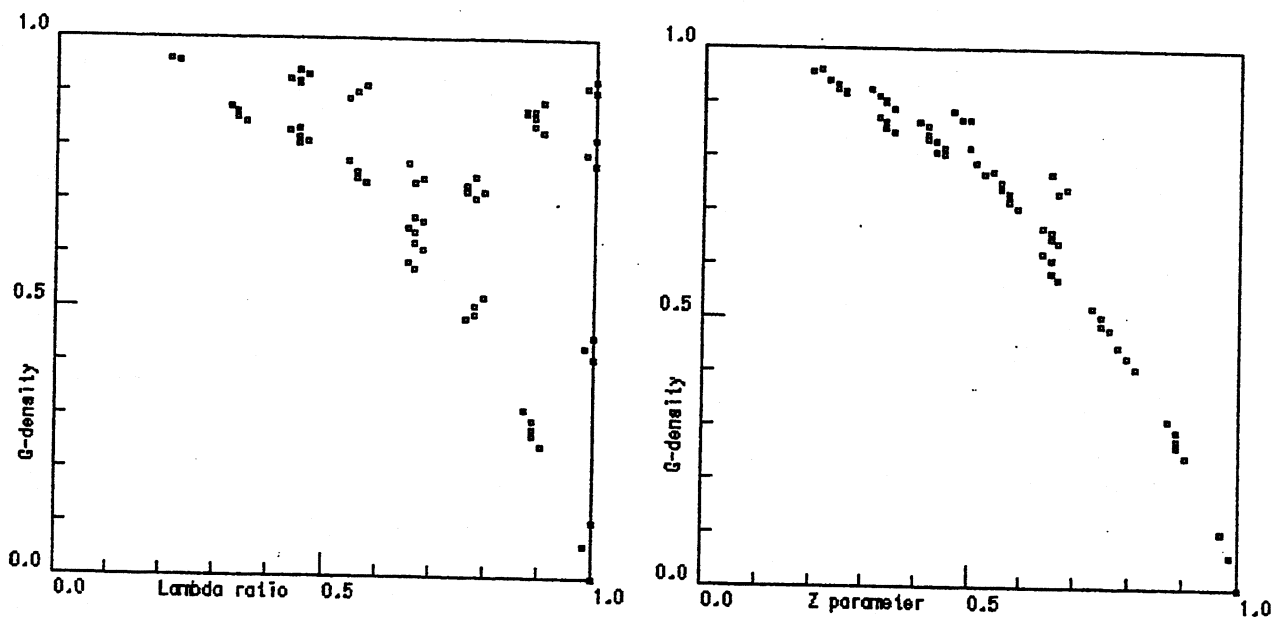


**Figure 4.34.**  $G$ -density plotted against the  $Z$  parameter for  $k=5$ ,  $k=7$  and  $k=9$  rules. Each graph shows over 1000 rules chosen at random but biased to include a representative spread of  $Z$ . Very high  $Z$  values were forced starting only from an even probability of 0s and 1s, which explains the abrupt change in direction of the cloud of points. The expected outer envelope of the cloud is indicated by the dotted line in the  $k=9$  graph, with points away from the cloud being increasingly rare. The complete basin of attraction field for each rule was generated for  $n=15$ , and garden-of-Eden states counted.

extent with increased  $n$ , confirmed by measures on larger systems based just on subtrees. Though generating entire basin of attraction fields for large lattice sizes to count garden-of-Eden states is computationally intractable, it is possible to gather this data on subtrees.

The assumption is made that the statistics on unbiased subtrees or subtree fragments will be representative of the global dynamics, provided that the subtree is located on the "skeleton", the main lines of flow of physically relevant states (Gutowitz and Domain 1995), as opposed to the short dead end side branches a little way in from garden-of-Eden states, which appear to occur uniformly throughout transient trees. The skeleton is quickly reached by running the CA forward from a randomly chosen state (which is likely to be a garden-of-Eden state) beyond this "sorting-out" stage, for example as described in the context of glider rules in section 4.6. However, it will be important to further investigate how the statistics on subtrees depend on the location of the subtree root, whether it is on or close to the attractor, at different depths on the skeleton, or within the sorting-out phase, or dead end side branches close to garden-of-Eden states.

Figure 4.33 shows some examples of subtrees for  $n=150$ . Appendix 4 shows subtrees for a range of lattice sizes  $n=20$  to 150. This and many other examples not shown indicate that for nearly all rules in rule-space,  $G$ -density increases at varying rates as a smooth function of lattice size. In the case of the *limited pre-image* rules where  $Z=1$  (Wuensche and Lesser 1992a), the plot behaves exceptionally; generally  $G$ -density approaches zero for greater  $n$ , or oscillates periodically.



**Figure 4.35.**

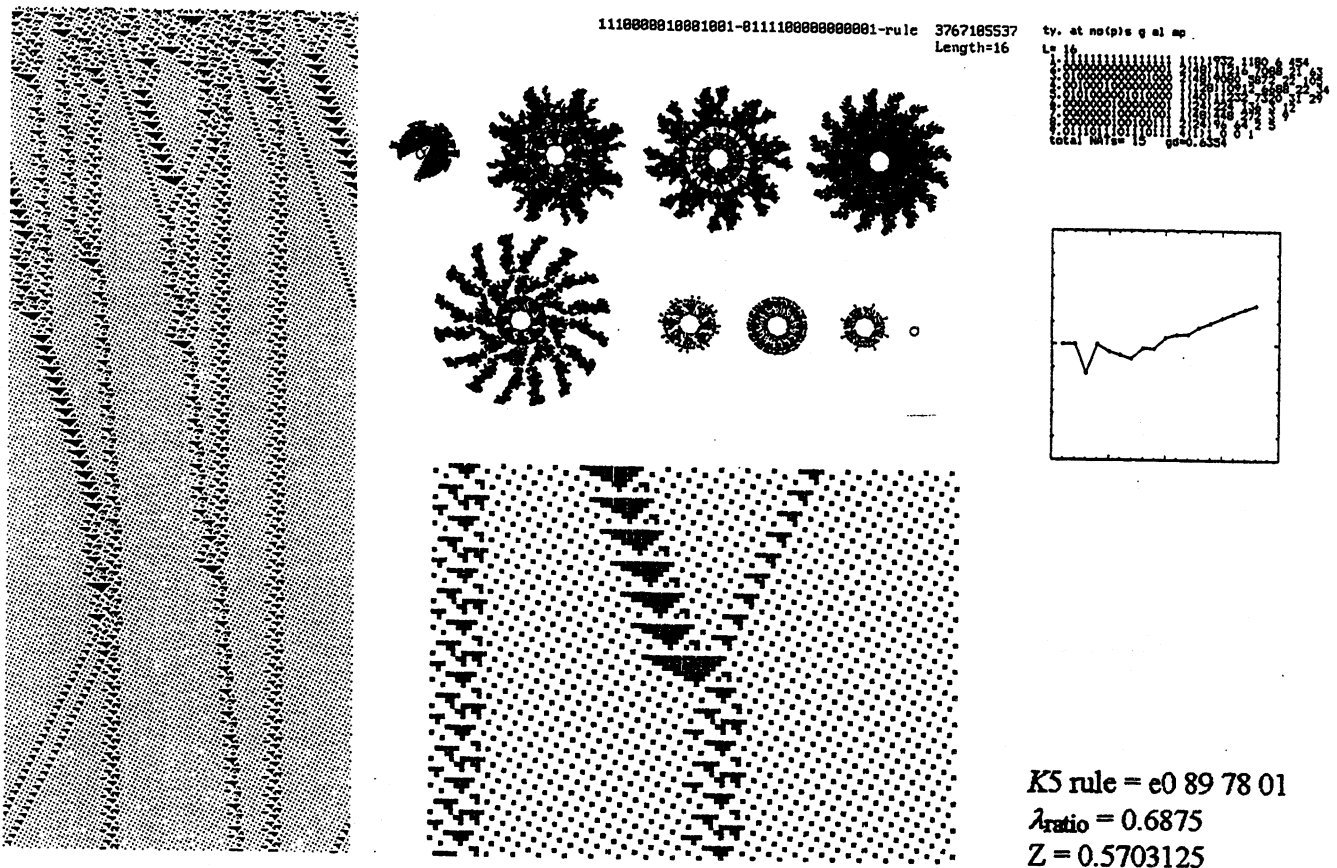
Plots of the  $G$ -density against both  $\lambda_{\text{ratio}}$  and the  $Z$  parameter for the set of  $k=7$  rules, showing the discrepancies as well as similarities between  $\lambda$  and  $Z$ . Points plotted in the top right corner of the  $\lambda_{\text{ratio}}$  graph represent rules with a  $\lambda$  value that does not correspond to behaviour as expected. There are 256  $k=7$  totalistic rules, but only 136 non equivalents in 72 clusters (having equal  $\lambda_{\text{ratio}}$  and  $Z$ ). A representative rule from each cluster was plotted, except where  $Z < 0.2$ . The complete basin of attraction field for each rule was generated for  $n=16$ , and garden-of-Eden states counted.

Rules in the same equivalence class have equivalent behaviour and thus identical  $G$ -density and in-degree distribution for a given  $n$ . This also appears to be true for complimentary rules (rule table entries flipped) that have different though related attractor basins. Although no proof of this statement is offered, it has been confirmed by many trials. Thus it has been assumed that the same graph showing  $G$ -density against  $n$  is valid for all rules in a rule cluster\*. The graphs have therefore been plotted for the 48  $k=3$  rule clusters and 20  $k=5$  totalistic code clusters.

#### 4.17. $G$ -Density and the $Z$ parameter

A number of plots of  $G$ -density against the  $Z$  parameter have been made. Again, for each rule, the whole basin of attraction field was generated. This was plotted against the rule's  $Z$  parameter

\*See (Wuensche and Lesser 1992a) for a discussion of rule clusters, complimentary rules and their attractor basins.



**Figure 4.36.** A typical space-time pattern and basin of attraction field of a  $k=5$  rule that supports emerging gliders, taken from the sample in appendix 2.17. *Left:* A typical space-time pattern, system size 200, periodic boundary conditions, 480 time-steps from the top down. *Top centre:* The basin of attraction field,  $n=16$ , showing the 9 non-equivalent basins from a total of 15 in state space. *Top right:* Basin field data (for key see appendix 2.1). *Centre right:* A graph of garden-of-Eden density ( $y$ -axis) against  $n$  for  $n = 1$  to 18. *Bottom centre:* A detail of glider interactions. *Bottom right:* The rule number in hex, its  $\lambda_{ratio}$  and  $Z$  parameter.

computed from its rule table. Appendix 1 shows graphs of  $G$ -density against the  $Z$  parameter for all  $k=3$  rules and  $k=5$  totalistic rules for system size 18. The same plot is shown in figure 4.32(right) for a the sample of glider rules in appendix 2.

Figure 4.34 shows three graphs of  $G$ -density plotted against the  $Z$  parameter for a sample of  $k=5, 7$  and  $9$  rules, for  $n=15$ . Each sample plots over 1000 rules. The rule sample was biased to include a representative spread of  $Z$  values by biasing  $\lambda$  (as described above for the plot of  $\lambda_{ratio}$  against  $Z$  in section 4.4). However, values in the high ranges of  $Z$  had to be evolved. This was done by starting with an unbiased random rule table and flipping (flipping back if necessary) random bits until the required  $Z$  parameter threshold was achieved.

The plots produce a characteristic cloud of points with an inverse correlation between  $G$ -density and  $Z$ , becoming more focused towards the left edge with higher  $k$ . Note that, as in the  $\lambda_{\text{ratio}} - Z$  plots, in principle points do occur to the right of the cloud in the plots of  $k=7$  and  $k=9$  rules as well as  $k=5$  rules, but become increasingly rare with larger  $k$ . Points are unlikely to occur in this area when rules are set at random as they depend on unlikely distributions of 0s and 1s in the rule table. This explains why the points are more focused towards the left edge of the cloud for larger  $k$ . Note also that a rule for a given  $k$  has an equivalent rule table relative to any larger  $k$ , so the points on the  $k=5$  plot could legitimately be added to the  $k=7$  and  $k=9$  plots. High  $Z$  values were evolved only from an even probability distribution of 0s and 1s, which explains the abrupt change in direction of the cloud for  $k=7$  and  $k=9$  rules. The conjectured outer envelope of the cloud is indicated by the dotted line in the  $k=9$  plot, with points away from the cloud becoming increasingly rare.

Plots of the  $G$ -density against both  $\lambda_{\text{ratio}}$  and the  $Z$  parameter for the set of  $k=7$  totalistic rules are shown in figure 4.35. The totalistic rules provide a conveniently even distribution over  $\lambda$  and  $Z$  to test the correspondence with  $G$ -density. A representative rule from each of the 72 totalistic rule clusters was plotted, except where  $Z < 0.2$ . The complete basin of attraction field for each rule was generated for  $n=16$ , and garden-of-Eden states counted, showing the discrepancies as well as similarities between  $\lambda$  and  $Z$ . Points plotted in the top right corner of the  $\lambda_{\text{ratio}}$  graph represent rules with a  $\lambda$  value that does not correspond to behaviour as expected. This may be compared to the  $\lambda_{\text{ratio}} - Z$  graphs for the  $k=7$  totalistic rules in figure 4.27.

The intention of these plots is to investigate  $G$ -density and its variation with system size for different classes of CA, and to what extent  $Z$  correlates with  $G$ -density, and thus with the convergence of state-space and basin field topology. It is clear that for the vast majority of rules, garden-of-Eden states occupy most of state-space. Rules with lower  $Z$  have higher  $G$ -density, increasing with system size at a fast rate. Rules in the most crowded area of rulespace according to  $Z$  (about 0.5 - 0.65) have moderate  $G$ -density, increasing at an intermediate rate with system size. Rules with very high  $Z$  have low  $G$ -density, increasing slowly with system size. Where  $Z=1$   $G$ -density is likely to remain constant or decrease.

#### 4.18. Summary and Discussion

Complex behaviour in simple 1d CA, especially the emergence of gliders, mirrors our intuitive notion of complex forms and processes emerging in nature. CA are arguably the simplest systems where complex phenomena arise. Their simplicity allows a description of global as well as local behaviour, and how this varies across rule-space.

A global perspective on CA dynamics and rule-space is provided by the notion of attractor basins. The basin of attraction fields of complex rules are typically composed of a small number of basins with long, moderately bushy, transient trees rooted on short attractor cycles. Gliders interacting aperiodically belong to the main lines of flow within the transient trees. Configurations where gliders interact periodically, or have ceased to interact, make up the attractor cycles.

Gliders have a distinct identity. Their interactions are predictable. A collision-table could be formulated empirically, without knowing the underlying rule-table mechanism. The collision-table would probably need to hold much more information than the rule-table. It would need to describe all possible permutations of collisions at different points of impact between gliders in a given glider rule. However, compared to the rule-table, the collision table would provide a far more useful description of *established* behaviour, enabling some prediction of the system's future evolution. On the other hand only the rule-table can account for the *origins* of gliders, their emergence by a process of self-organisation from random patterns.

Interacting gliders may combine to create compound gliders, interacting at yet higher levels of description, and conceivably the process could unfold hierarchically without limit in large enough systems. This is analogous to describing matter in terms of chemistry as opposed to the underlying sub-atomic particles, or in terms of biology as opposed to the underlying chemistry. There are any number of further analogies that might be drawn from nature or society. However, the origins of the higher level entities must refer to the lower level. The possibility of new levels of description of dynamical behaviour, on the basis of observed glider collision rules without reference to the underlying low level CA rules, provides a prototype of emergence by self-organisation. According to this approach, a system's complexity is the number of levels of description that underpin it in a descending hierarchy.

In this chapter, large samples of glider rules and other complex 1d CA rules have been assembled by an automatic procedure based on a local measure, input-entropy variance, which correlates well with a subjective assessment of CA space-time patterns. Global measures on the topology of basins of attraction and subtrees have been related to the local measures, in particular  $G$ -density, in-degree frequency, and the rate at which  $G$ -density changes with system size. These measures have in turn been related to the rule parameters  $\lambda$  and  $Z$ . The combination of these measures and parameters may help to characterise rule-space and complex rules in particular.

The local measure, entropy variance, has been found to distinguish glider dynamics between order, complexity and chaos, allowing virtually unlimited samples of complex rules to be assembled. The  $Z$  parameter approximately predicts the degree of convergence in attractor basins, which is measured by  $G$ -density, and also by a more general measure, the distribution of in-degree frequency. These global measures discriminate at least between the extremes of order and chaos, and

might also be capable of finer discrimination. Complex dynamics seems to achieve a fine balance between high and low convergence where the in-degree frequency of complex rules follow a power law distribution, according to the preliminary results.

Further investigation is required to systematically test both the local and global behaviour of the automatic rule samples to see whether the behaviour space can be more finely demarcated. The computer tools for such an investigation are largely in place.



## Chapter 5

### Memory in Random Boolean Networks

#### 5.1. Introduction

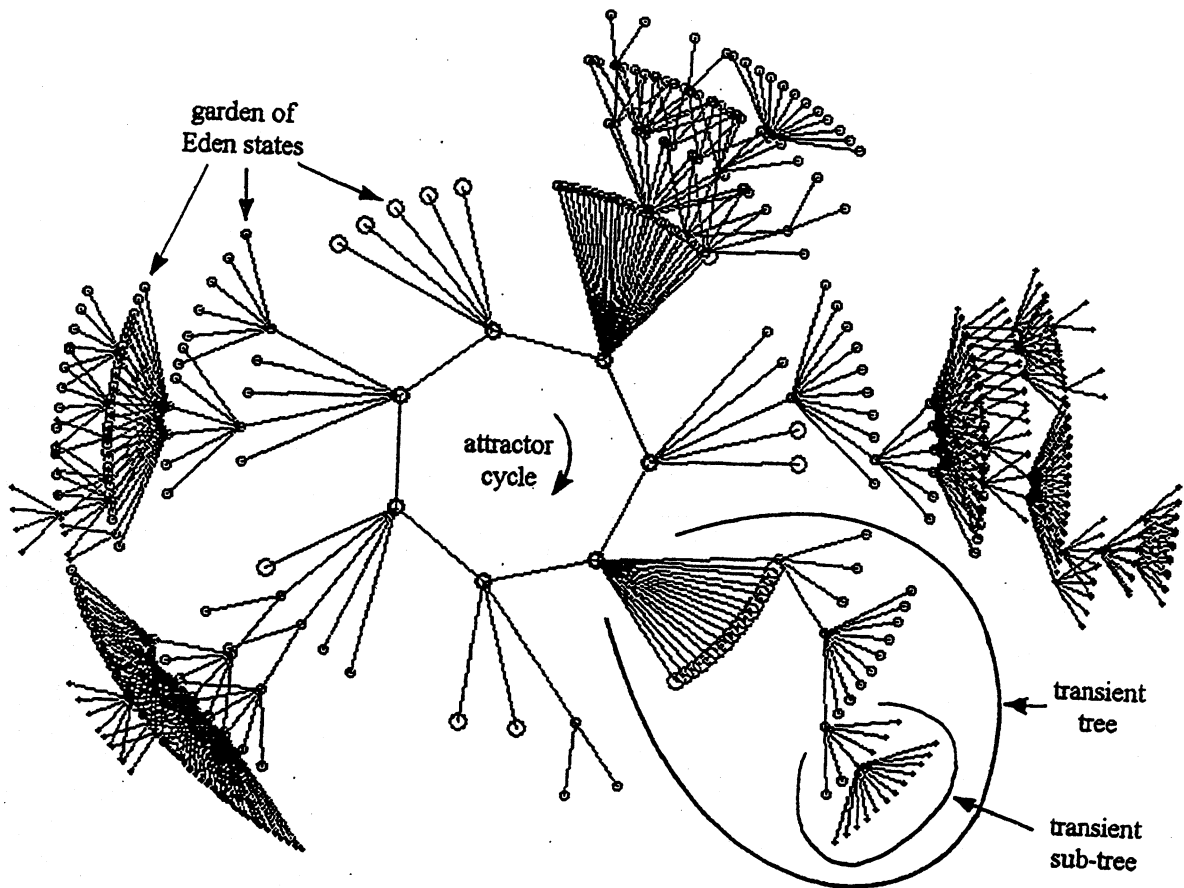
The ideas and methods described in chapter 4 for unravelling the global dynamics of cellular automata (CA) (Wuensche and Lesser 1992a, Wuensche 1994a) have been extended to random Boolean networks (Wuensche 1992b,c,1993a,b,c,1994b). Random Boolean networks (RBN) are perhaps the most general instance of a synchronous discrete dynamical network. A fully connected RBN is equivalent to a random mapping (Kauffman 1969) as described in chapter 2, but RBN are usually taken as sparsely connected systems where  $k \ll n$ , the number of input connections available to each network element,  $k$ , is much smaller than  $n$ , the number of elements. This accords with the applications of RBN in cell biology, where a gene is regulated directly by just a few of the many other genes in the genome, and in neural networks where a neuron has *relatively* few direct inputs (though the number may be many 1000s) from other neurons in the brain or its local region. In both cases a high proportion of inputs may be local to the gene or neuron, i.e. *cis* as opposed to *trans* acting DNA sequences in a genomic regulatory network, and local circuits as opposed to long range interregional pathways in the brain.

Random wiring confined within a local neighbourhood relative to each network element\* implies a network geometry. In that case the dimensionality, geometry (i.e. orthogonal or triangular, etc.), and boundary conditions of a network lattice would need to be specified as for CA. In a network with fully random wiring there is no sense of geometry, how each network element is assigned to the lattice is irrelevant to the dynamics. Localised random wiring introduces a "speed of light" limitation to the influence of any element in the network based on the radius of the local neighbourhood, as described in chapter 4.6 for CA, but the overall effect on dynamics is as yet unclear.

The attractor basins of RBN may be reconstructed as described in chapter 2, and explicitly portrayed as diagrams that connect up the network's global states according to their transitions - typically, the topology is branching trees rooted on attractor cycles, but the attractor basins in general

---

\* The term "network element" is used in this chapter instead of "cell" as in cellular automata to avoid confusion when discussing cell biology. In the context of genomic regulatory networks the network element is a gene, whereas in the neural network context the network element is a neuron.



cell	wiring	rule,	table
1	3,12,6	86,	01010110
2	7,11,4	4,	00000100
3	3,3,1	196,	11000100
4	11,3,9	52,	00110100
5	8,7,5	234,	11101010
6	1,8,1	100,	01100100
7	12,4,13	6,	00000110
8	8,6,8	100	01100100
9	9,2,6	6,	00000110
10	5,1,1	94,	01011110
11	2,7,1	74,	01001010
12	7,8,4	214,	11010110
13	1,4,7	188,	10111100

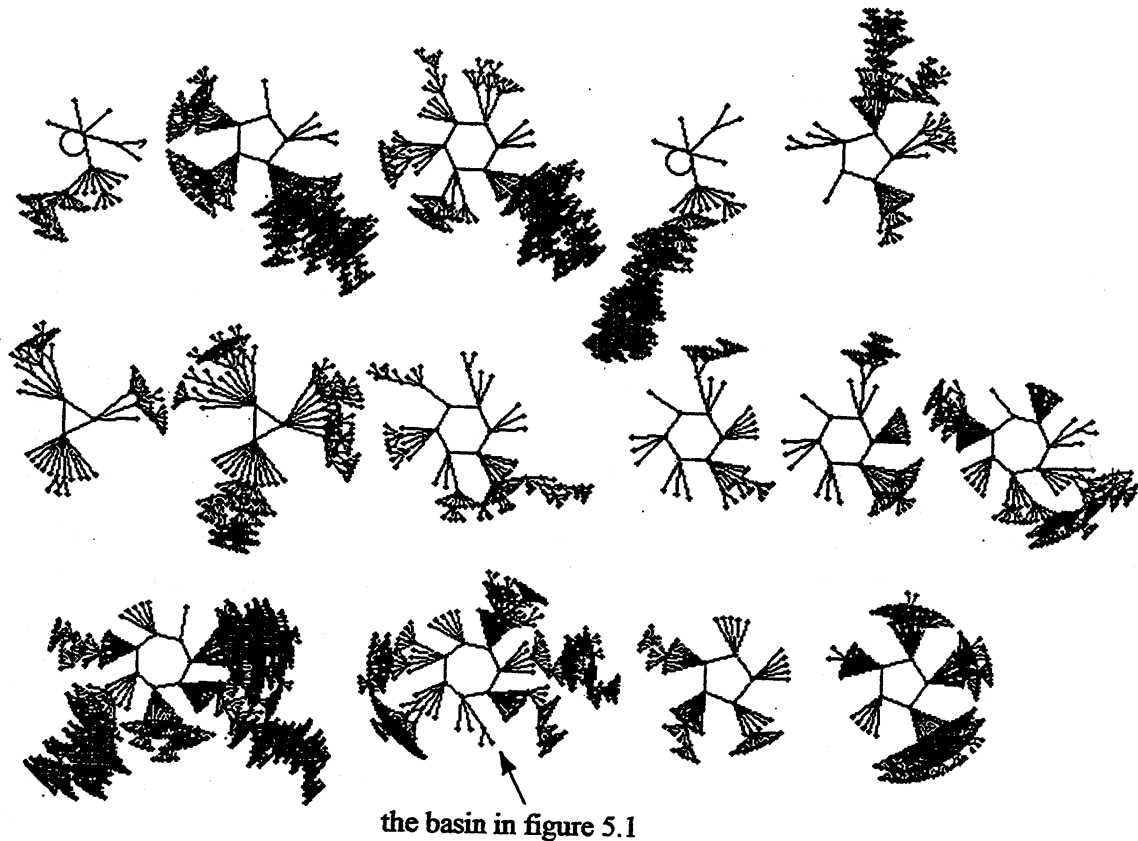
**Figure 5.1.**

*Above.* A basin of attraction of a random Boolean network ( $n=13$ ,  $k=3$ ). The basin links 604 states, of which 523 are garden of Eden states. The attractor has period 7. The direction of time is inwards from garden-of-Eden states to the attractor, then clock-wise. The basin is one of 15, and is indicated in the basin of attraction field in figure 5.2.

*Left:* The random Boolean network wiring/rule parameters. Wiring and rules were assigned at random, except that the neighbourhood  $000 \rightarrow 0$ .

lack the symmetries inherent in CA described in chapter 1.4. The diagrams are constructed with a RBN reverse algorithm (described in section 5.5 below) that directly computes a state's set of pre-images (if any).

In continuous deterministic dynamical systems, all possible time series are represented by families of tangents to the vector field, the field of flow imposed on phase space by the systems dynamical rule. This is represented by the system's phase portrait consisting of trajectories, attractors, separatrices etc. Attractors may be fixed points, limit cycles or chaotic (strange) attractors.



**Figure 5.2.** The basin of attraction field of a random Boolean network ( $n=13$ ,  $k=3$ ). The  $2^{13}=8192$  states in state space are organised into 15 basins, with attractor periods ranging from 1 to 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 5.1 shows the arrowed basin in more detail, and the network's wiring/rule scheme.

They "attract" various regions of phase space in the basin of attraction field. Analogous concepts apply to discrete dynamical systems, such as CA and RBN, which are noise free and update synchronously. An important difference, however, is that in these discrete deterministic systems transients merge outside the attractor, far from equilibrium, whereas in continuous systems they cannot (though the transients may come arbitrarily close).

In a highly influential paper by John Hopfield (1982), he presented a recurrent, asynchronous, neural network model that produced what he called a "general content addressable memory", where the system's dynamics is able to flow in a discrete state-space to a substantial number of locally stable points, attractors or energy minima; the systems equilibrium conditions. He characterised content addressable memory as the resulting categorisation of state-space, where an item of information (represented by a pattern of 0s and 1s across the network), the input, initiates a flow to a

particular stable point. The system is potentially capable of error-correction and generalisation if deviations from the input pattern end up at the same stable point. As well as stable points, Hopfield also found other types of equilibrium conditions in his model (which were not entirely welcome), namely cycles and "chaotic wanderings in a small region of state space". He further suggested analogies to real nervous systems.

I will argue that attractors in RBN can be viewed as constituting the networks "content addressable" memory in the way that Hopfield described, but with an added ingredient. In Hopfield's model state-space is categorised only at attractors, there is no notion of reliable convergence to transient states outside the attractors. In RBN, because transients merge deterministically, state-space is also categorised hierarchically along transients, by the root of every subtree. Reference to "memory" in this chapter should be understood in this context

Hopfield's model does not support deterministically merging transients because his updating method is randomly asynchronous, and thus non-deterministic. It is open to debate whether synchronous or asynchronous updating in a local network is more or less biologically plausible. However, synchronous random networks arguably have greater potential as content-addressable memory systems because of categorisation in subtrees as well as at attractors. Sub-categories of state-space converge to unique states along each transient. Categorisation occurs along a reliable time-series of states, creating a complex hierarchy of content addressable memory. Usually the majority of state-space is made up of garden-of-Eden states which have no predecessors. These states are the *leaves* of transient trees thus cannot form categories. They are analogous to points on the separatrices in the phase portrait of a continuous dynamical system. Other states belong on attractor cycles so categorise the whole basin of attraction. States that are neither garden-of-Eden nor attractor states form the roots of subtrees, so categorise all states in their particular subtree.

In biological networks such as neural networks in the brain or networks of genes regulating the differentiation and adaptive behaviour of cells, the topology of attractor basins and subtrees must be just right for effective categorisation in some larger context. The dynamics needs to be sufficiently versatile for adaptive behaviour but short of chaotic to ensure reliable behaviour, and this in turn implies a balance between order and chaos in the network. In RBN, as applied to genomic regulatory networks (Kauffman 1969), the notion of a phase transition between order and chaos is just this balance. Tuning various parameters in the network's wiring/rule setup, in particular the degree of "canalisation" at the level of rules and the extent of connectivity between network elements (Kauffman 1984, Harris *et al* 1997) moves behaviour across the transition. The range of topologies of basins of attraction of RBN, and by analogy of biological networks, and their potential for complex categorisation, meaningful in some particular context, suggests that attractor basins - *the ghost in the machine* - may emerge as the network's cognitive substrate (Wuensche 1992b).

A basic difficulty in explaining memory just by attractors in biological neural networks has been the probably astronomical transient lengths needed to reach an attractor in large networks, whereas reaction times in biology are extremely fast. The answer may lie in the notion of memory far from equilibrium along merging transients (Wuensche 1993c).

CA (of whatever dimension), with homogeneous connectivity and rules, may be regarded as a special sub-class of RBN. Evidence is presented that CA *local* architecture is necessary for the emergence of coherent space-time structures such as gliders, the expression of self-organisation in CA and the basis of CA models in artificial life. RBN architecture breaks the two basic CA architectural requirements, the wiring and the rule scheme may be arbitrary and different at each cell, though divergence from CA architecture may be made by degrees. An arbitrary wiring/rule scheme implies a vastly greater parameter space, and thus attractor basin configuration space than for CA. Though fully connected networks where  $k=n$  would be required to achieve any arbitrary basin of attraction field, a great many basin structures (perhaps the most useful) are achievable in networks where  $k \ll n$ . The stability of the field under small perturbations to parameters is noteworthy. In biological networks the process of adaptation through evolution and lifetime learning modifies a network's parameters, its wiring/rule scheme or size/connectivity, resulting in a modified basin of attraction field.

This chapter is organised into four parts introduced below. Firstly a detailed examination of RBN and their attractor basins (sections 5.2-5.8), secondly an examination of RBN learning algorithms by modifying the network or building it from scratch (section 5.9), thirdly a largely speculative discussion on the insights that may be derived from the notion of attractor basins in understanding memory in animal brains (section 5.10), and lastly a summery of how attractor basins are applied to provide insights into genomic regulatory networks. The ideas presented in this chapter represent work in progress developed to varying degrees, pointing to further lines of research. It is hoped that the distinction between conjecture and hard results will be clear.

Sections 5.2-5.8 describes RBN architecture, space-time patterns and basins of attraction. These are compared and contrasted with CA. The reverse algorithm for computing pre-images in RBN is explained.

Section 5.9 describes learning algorithms that automatically re-assign pre-images in a single step. New attractors can be created and transient trees and sub-trees transplanted, *sculpting* the basin of attraction field to approach a desired configuration. The effects and side effects of learning become immediately apparent by re-drawing the modified basin of attraction field, or some fragment of it. Such *visible learning* (Wuensche 1993b) may lead to useful applications as well as helping to clarify the process of memory and learning in a variety of artificial neural network architectures.

The idea of *sculpting* attractor basins has inspired some recent work, which will be briefly described, to find the RBN architecture that will most efficiently realise a particular basin of attraction field, or a partial set of predetermined transitions such as a transient. This has been called the *inverse problem*. Alternative methods for solving the inverse problem have recently been proposed independently by Manor Askenazi (1996) and John Myers (1996). Their approach adds support to the learning algorithms, which are methods for fine tuning a network rather than building it from scratch. Future work will aim to combine these methods. The methods may have potential in areas where artificial neural networks are applied such as pattern recognition, and also in understanding complex biological processes, for example in the extraction of genetic network architectures (Somogyi *et al* 1997).

**Section 5.10** suggests that RBN may provide a framework for understanding memory, and proposes a RBN as a neural network model. In the context of many semi-autonomous weakly coupled networks, the basin field/network relationship may provide a fruitful metaphor for the mind/brain.

**Section 5.11** goes on to discuss RBN as applied to model genomic regulatory networks, where cell types are explained as attractors or "frozen skeletons" in the dynamics of the network. DDLab is being used as a simulation platform for these models to unravel the dynamics in terms of attractor basins, and to extract various measures distinguishing dynamics between order and chaos and thus identify the phase transition. This is work in progress in collaboration with Stuart Kauffman and others. (Harris *et al* 1997, Somogyi and Sniegoski 1996a). Both *local* measures on space-time patterns and also *global* measures on the topology of the attractor basins and subtrees are being investigated. The Derrida plot (Derida and Stauffer 1986) is analogous to the Liapouov exponent in continuous dynamics and measures the divergence of trajectories based on Hamming distance. Further measures are available such as the frequency of sizes of damage spread resulting from random mutation, the percolation of "frozen" i.e. unchanging regions, input entropy of single genes taken on the frequency of input patterns over time. Measures on the topology of the attractor basins and subtrees such as their in-degree distribution can be made as for CA. The precise structure of attractor basins is also of interest as it indicates the stability of cell types to perturbation and allows methods of extracting genetic network architectures from biological data (Somogyi *et al* 1996), an application of the inverse problem. For larger RBN, methods are available for extracting statistical data on transient, frozen skeleton and attractor characteristics such as the number and size of different attractor basins or skeleton sub-trees, the length of attractor cycles and of attractor/skeleton transients.

Various network parameter settings may be tuned to move the RBN dynamics across the order/chaos phase transition as defined by these measures. The  $P$  parameter is equivalent to the  $\lambda$

parameter in CA. Another parameter which seems to be closer to observed cell biology is the degree of "canalisation" at the level of rules, though the characteristics of network connections also play an important role. Canalisation occurs when a particular input (0 or 1) on a connection determines a gene's behaviour irrespective of its other inputs. That connection is then said to be canalising. The canalisation setting for RBN corresponding to the phase transition is very far from random expectation. The data that is just beginning to be assembled on real genomic regulatory networks (Harris, *et al* 1997) seems to correspond to these settings, indicating that genomic regulatory networks in biological systems evolve to remain delicately posed at the "edge of chaos". The results and methods are described.

## 5.2. Random Boolean Networks

The idea of randomly connected multi-function networks as dynamical systems with a corresponding field containing all lines of behaviour can be traced back to Ross Ashby, in his book "Design for a Brain" (Ashby 1960). Random Boolean networks (also called Kauffman's model), have been investigated for a considerable time by Stuart Kauffman in theoretical biology and complex systems (Kauffman 1996,1984,1991), in particular to model genomic regulation underlying embryonic development. An overview of recent work in this area in collaboration with Kauffman and others is described later in this chapter (Harris *et al* 1997, Somogyi *et al* 1996,1997). Variations of random Boolean networks have been applied in the context of memory in the immune network, (e.g. Weisbuch *et al* 1990), and in complex systems in general (Walker and Ashby 1966, Walker 1971-1987). The focus of interest of these studies has often been to gain insights into global dynamics, the topology of basin of attraction fields in relation to a range of possible network parameters. Without the benefit of the methods for reconstructing attractor basins, the studies relied on statistical methods from data on trajectories from many forward runs, giving an idea of numbers of attractors, their size distribution, disclosure length and related information. Others have pursued these insights into the global dynamics of RBN by analytical methods, for example (Derrida and Pomeau, 1986).

Random Boolean network architecture is in many ways similar to that of weightless (or logical) neural networks (Alexander *et al* 1984), where standard memories (RAMs) hold each network element's look-up table. Classical neural network architectures use weighted connection and threshold functions. A random Boolean network may be regarded as a discrete generalisation of a sparsely connected artificial neural network. Connections with higher weights may simply be replaced by multiple couplings and the threshold function applied. However, a threshold function is a tiny sub-class of the  $2^{2^k}$  possible Boolean functions. Given a simple single layer artificial neural

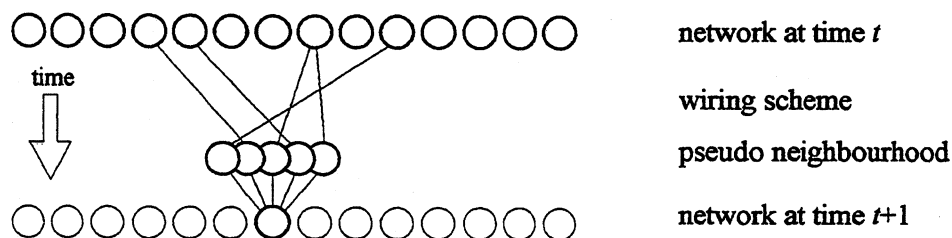
network based on threshold or sigmoid functions, the equivalent  $k$  input Boolean function for each network element can be found by filling in the lookup table based on all possible  $k$  inputs. Thus for such simple artificial neural networks the equivalent RBN can be found, and the methods described in this chapter can be applied to reconstruct their attractor basins.

### 5.3. Random Boolean Network Architecture

Random Boolean networks may be viewed as generalised (*disordered*) CA (Wuensche 1992b), breaking two basic premises of CA architecture by allowing arbitrary wiring and/or rules at each network element. The effect on behaviour of deviating from either or both of these premises by degrees will be discussed below (see figures 5.5 and 5.8). Not surprisingly, coherent space-time patterns and emergent complex structures such as *gliders*, characteristic of CA, are progressively degraded.

Random Boolean networks have a vastly greater parameter space, and thus behaviour space, than CA. The various symmetries and hierarchies that constrain CA dynamics previously described need no longer apply. There is no limit to the speed of propagation across the network. The notion of space and the 'speed of light' lose significance as the homogeneous wiring/rule scheme of CA is progressively scrambled, though this can occur by degrees.

It might be supposed that any arbitrary basin of attraction field configuration is possible given the right set of RBN parameters, however this is only true for fully connected networks, which are equivalent to a random map (see chapter 3). Where  $k \ll n$  the space of possible basin of attraction fields diminishes, though it is still very large. John Myers has produced expressions for this number as a function of  $n$  and  $k$ , including the case where  $k$  differs between network elements (Myers 1996).



**Figure 5.3.** RBN architecture. Each element in the network synchronously updates its value according to the values in a pseudo neighbourhood, set by single wire couplings to arbitrarily located network elements at the previous time-step. Each network element may have a different wiring/rule scheme and different  $k$ . The system is iterated.



Each element in an RBN may have a different *wiring* and/or *rule* scheme (but not necessarily), and also a different value of  $k$ . Most work on RBN has assumed networks with uniform  $k$ , but recent investigations of genomic regulatory networks require setting up networks with particular proportions of different  $k$  inputs to match data from cell biology. Mixed  $k$  networks are also required to check solutions to the inverse problem.

The software DDLab allows a network with any combination of wiring, rules, and  $k$  values to be set up. Random wiring can be restricted or biased in various ways for example within a local neighbourhood. Random rules can be biased to finely tune their fraction of canalising inputs.

A random Boolean network implies a value range of 2 (0 or 1), but in principle the arguments in this chapter could equally apply to a network where cells have more than two values. As in CA, the global state of a network of  $n$  elements is the pattern resulting from values assigned to each element, from a finite range of values  $v$  (usually  $v=2$ ). Each cell synchronously updates its value in discrete time steps. The value of a cell  $C_i$  at  $t+1$  depends on its particular Boolean function  $f_i$  (equivalent to a CA rule table), applied to a notional or *pseudo* neighbourhood, size  $k$ . Values in the neighbourhood are set according to single wire couplings to arbitrarily located elements in the network at time  $t$ . The system is iterated. The system's parameters are set by specifying the pseudo neighbourhood wiring and Boolean function (and  $k$  if necessary) for each of the  $n$  network elements. Once set, the networks wiring/rule scheme is usually fixed over time.

The  $i$ -th cell  $C_i$  has wiring connections  $w_{i1}, w_{i2}, \dots, w_{ik}$ . Connections are assigned to any of the  $n$  elements in the network, including  $C_i$  itself. Duplicate connections are allowed, giving  $n^k$  possible alternative wiring options. The rule for element  $i$  is  $f_i$  assigned from  $v^{n^k}$  alternatives in rule-space. The time evolution of the  $i$ -th element is given by

$$C_i^{t+1} = f_i(C_{w_{i1}}^t, C_{w_{i2}}^t, \dots, C_{w_{ik}}^t)$$

The number of all possible alternative wiring/rule schemes  $S_{n,k}$  that can be assigned to a given RBN of size  $n$  and homogeneous connectivity  $k$  is given by, (Wuensche 1992b)

$$S_{n,k} = (n^k)^n \times (2^{2^k})^n$$

This may be compared with the number of possible basin of attraction fields  $F_n$  for a network of size  $n$ . As was shown in chapter 2,  $F_n$  equals the number of alternative mapping from  $Q \rightarrow Q$ , for a set  $Q$  of size  $n$ , so

$$F_n = (2^n)^{(2^n)}$$

As  $S_{n,k} > F_n$  there must be considerable redundancy in wiring/rule schemes so that many equivalent schemes generate identical basin of attraction fields. This is indeed the case. For example, if the pseudo neighbourhood of a network element is re-ordered, there is an appropriately re-ordered rule table that gives equivalent behaviour. John Myers has analysed this problem and derived expressions for the number of effectively distinct maps, thus basin of attraction fields for RBN of size  $n$  and homogeneous connectivity  $k$ , and a more general expression for networks with mixed  $k$  (Myers 1996).

#### 5.4. Space-time patterns and basins of attraction

As in CA, it would be useful to relate the two aspects of RBN behaviour, local dynamics as seen in the space-time patterns of individual trajectories, and global dynamics as seen in attractor basins. This can be done to some extent, but because the space of possible RBN architectures is so vast it is difficult to characterise this space in general as for CA. Some relationships between local and global dynamics are apparent, especially as applied to genomic regulatory networks to be discussed later. The tools are in place but most of this territory remains unexplored.

For completeness we will briefly review network dynamics in terms of attractor basins (see also chapter 2). A space-time pattern represents a single determined trajectory through state-space. An initial pattern or *seed* assigned to the RBN lattice at time  $t_0$  sets off a succession of patterns at times  $t_1, t_2, t_3, \dots$  by the iteration of the RBN transition function. The layout of elements across the lattice (and its pattern) may be arbitrarily represented as 1d or 2d etc. If the wiring is fully random this is just a convenience. However, if the network has some degree of intermediate architecture, or if random wiring is confined within a radius around each network element, then the network geometry and boundary conditions are important parameters.

The future dynamics is determined yet unpredictable. As in CA there seems in general to be no short cut for knowing the future more efficiently than by performing the actual iterations themselves. The sequence of iterated states is a *trajectory*, and may be represented by a space-time pattern diagram. This may be shown in 1d as successive rows of elements as for in (see figure 4.2), or as is more usual in models of genomic regulatory networks, the network is represented in 2d. Network elements are coloured according to value, 0-white, 1-black.

An RBN state has just one successor, but may have an arbitrary number of predecessors (its pre-images). States with no pre-images are so called garden-of-Eden states. They cannot be reached by normal RBN evolution, but must be imposed from outside. Although a initial state determines a single future, each iteration (which cannot be a garden-of-Eden state) may have many possible past histories.

The state-space of a RBN with  $n$  cells is  $2^n$ . Any path inevitably encounters a repeat of a previous state, and must lead to a state cycle (the *attractor*). The attractor may have just one state, a stable point cycling to itself, or may have an arbitrarily long period. The set of all possible paths leading to the same attractor, including the attractor itself, make up a *basin of attraction*. State-space is typically divided into many basins, the *basin of attraction field*.

A trajectory is just one particular path within a basin of attraction. A *transient* is the portion of the trajectory outside the attractor cycle and usually merges with other transients to form a branching tree with garden-of-Eden states as the leaves. A *sub-tree* is a branch of the *transient tree*. Basins of attraction typically have a topology of trees rooted on cycles. Access to these objects opens up a new area of phenomenology, but this has not been explored to the same extent as for CA. For example, it is likely that measures of the relative length and "bushiness" of subtrees in RBN correlate with measures of order and chaos in space-time patterns such as the input-entropy of individual network elements (see chapter 4.11 and section 5.11.7), but systematic studies have not yet been made.

### 5.5. Computing RBN pre-images

Just as in CA, constructing a basin of attraction or subtree of an RBN poses the problem of finding the pre-images of each state. A possible method is to construct an exhaustive map resulting from network dynamics and to scan the map for pre-images (as described in 3.7) but this becomes computationally intractable as the network's size increases beyond modest limits. It is a useful method however for small networks with large  $k$ .

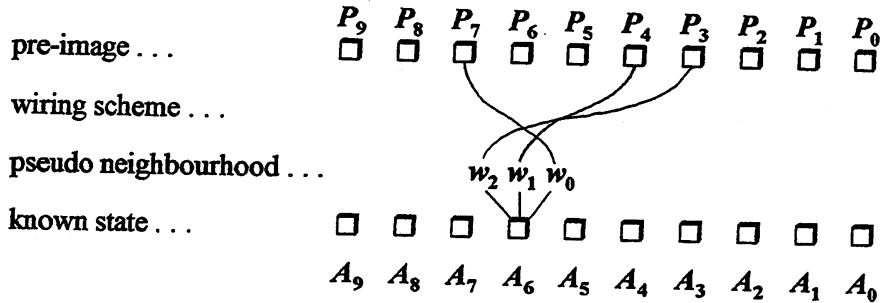
The *reverse algorithm* that directly computes pre-images for RBN, without the need for an exhaustive map, has been invented by the author (Wuensche 1992b, 1993a). Using this algorithm the network's dynamics can be run *backwards* in time. Backward trajectories will usually diverge. The DDLab source code of the function that implements the algorithm is shown in appendix 6.3. Appendix 7.2 describes network size limitations and time issues.

This algorithm also works for CA. The alternative methods for computing pre-images, for CA, RBN and random maps, provide a check on the correctness of the computed pre-images.

The RBN algorithm is described below. Consider a random Boolean network with  $n$  elements and a pseudo neighbourhood size  $k$ . The algorithm will be demonstrated for  $n=10$ ,  $k=3$ . Equivalent procedures apply for different  $n$  and  $k$ , or for a network with mixed  $k$ . For convenience, the system is represented as a 1d array,  $A_{n-1} A_{n-2} \dots A_1 A_0$ .

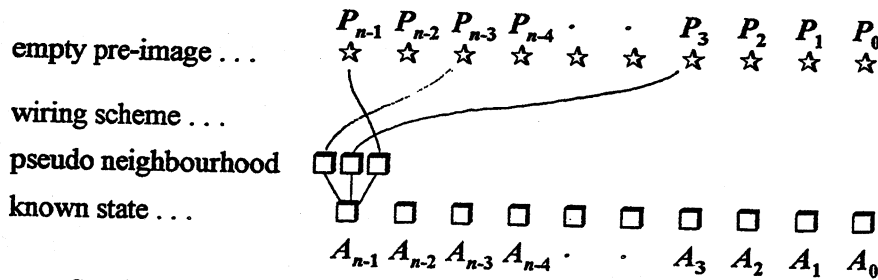
Each network element,  $A_i$  (value 0 or 1), has a pre-set wiring/rule scheme (possibly selected at random). The element's wiring scheme is given by,  $A_i(w_2 w_1 w_0)$  where  $w_j$  is a number between  $n-1$  and 0 signifying the position of the wire connections from  $j$ th branch of the pseudo neighbourhood,

and so on. The element's rule scheme is given by,  $A_i(T_7, T_6, \dots, T_0)$ , the  $k=3$  rule table. In the example below the wiring scheme for  $A_6$  is  $w_2=3$ ,  $w_1=4$  and  $w_0=7$ .



To derive the pre-images of an arbitrary global state, consider a candidate pre-image as an *empty* array, consisting of *empty* network elements. They are empty because their values are unknown, and unallocated as either 0 or 1. Empty elements are denoted by the wild card star symbol  $\star$ , known elements (with values established as 0 or 1) are denoted by the block symbol  $\square$ .

Consider a known network state,  $A_{n-1}A_{n-2}\dots A_0$  and the empty pre-image state  $P_{n-1}P_{n-2}\dots P_0$ .

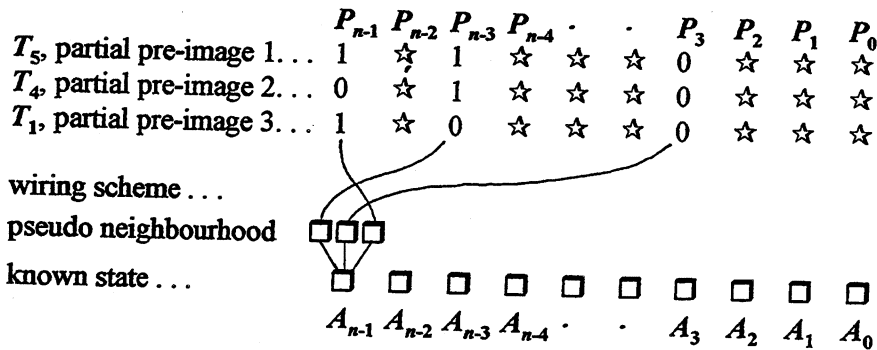


Starting with the first cell of the known state,  $A_{n-1}$ , the valid pseudo neighbourhood values, consistent with the value of  $A_{n-1}$  are assigned to separate copies of the empty pre-image according to the wiring scheme  $A_{n-1}(w_2, w_1, w_0)$ . As there will be a mix of 0s and 1s in the rule table, only some of the 8 possible pseudo neighbourhoods will be valid. If, say, 3 are valid, 3 *partial pre-images* (with some elements *known*, and some *empty*) will be generated. For example, given the  $k=3$  rule 50 at  $A_{n-1}$ , with a rule table as follows,

rule-table...	111	110	101	100	011	010	001	000	...neighbourhoods
	0	0	1	1	0	0	1	0	...outputs (0 or 1) rule 50
	$T_7$	$T_6$	$T_5$	$T_4$	$T_3$	$T_2$	$T_1$	$T_0$	

If  $A_{n-1}=1$ , then only 3 outputs match  $A_{n-1}$ ,  $T_5$ ,  $T_4$  and  $T_1$ , corresponding to the neighbourhoods 101, 100 and 001. These valid neighbourhoods are allocated to 3 empty arrays according to the

wiring scheme, say  $A_{n-1}(n-3,3,n-1)$ . Each of the 3 arrays now have some of their elements allocated as 0s or 1s, and are termed *partial pre-images*, as illustrated below.



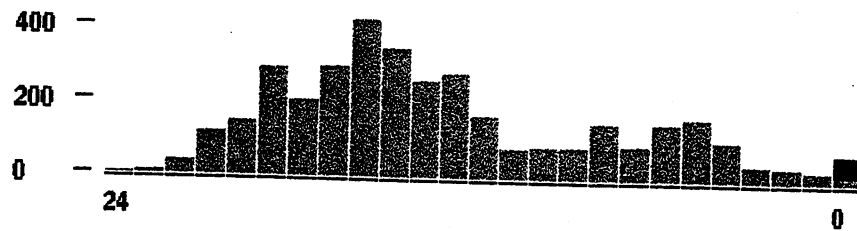
The procedure continues with the next element of the known state,  $A_{n-2}$  (though the order may be arbitrary). Say that the value of  $A_{n-2}$  has 5 (out of 8) valid pseudo neighbourhoods in its rule table,  $A_{n-2}(T_7 T_6 \dots T_0)$ . The pseudo-neighbourhoods are allocated to 5 copies of each of the partial pre-images that were generated at  $A_{n-1}$ , according to the wiring scheme  $A_{n-1}(w_2 w_1 w_0)$ .

If the allocation of a value to a given network element conflicts with the value already assigned to that element, then the partial pre-image is rejected. Otherwise, the partial pre-image is added to the partial pre-image stack. The allocation will be valid if it is made to an empty cell, or to a known cell with an equal value. 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,  $A_{n-3} A_{n-4} \dots A_0$ . At each successive element, more partial pre-images may be added to the stack, but also rejected. The changing size of the stack at successive elements can be displayed in DDLab. A typical example is shown in figure 5.4. If the stack size is reduced to zero at any stage then the known state  $A_{n-1} A_{n-2} \dots A_0$  has no pre-images; it is a garden-of-Eden state. In general, the vast majority of states in state space turn out to be garden-of-Eden states. Appendix 7.3 gives some indication of network size limitations and the time required to generate attractor basins using DDLab.

Note that the order in which network elements in  $A_{n-1} A_{n-2} \dots A_0$  are taken is entirely arbitrary. For the most efficient computation, that minimises the growth of the partial pre-image stack, the order should correspond to the greatest overlap of wiring schemes. Note also that if the wiring scheme for a given network element  $A_i(w_2 w_1 w_0)$  is ordered differently, there will be a different rule (an appropriately re-ordered rule table) that will give equivalent behaviour thus pre-images.

When the procedure is complete, the final pre-image stack may still have empty network elements, signifying that these elements are not sampled by any wiring couplings. Final stack arrays with empty elements are duplicated so that all possible configurations at empty element positions are



**Figure 5.4.**

Computing RBN pre-images. The changing size of a typical partial pre-images stack at successive elements, as displayed in DDLab.  $n=24$ ,  $k=3$ , random wiring. The black portion of the final stack represents arrays that ended up with empty elements.

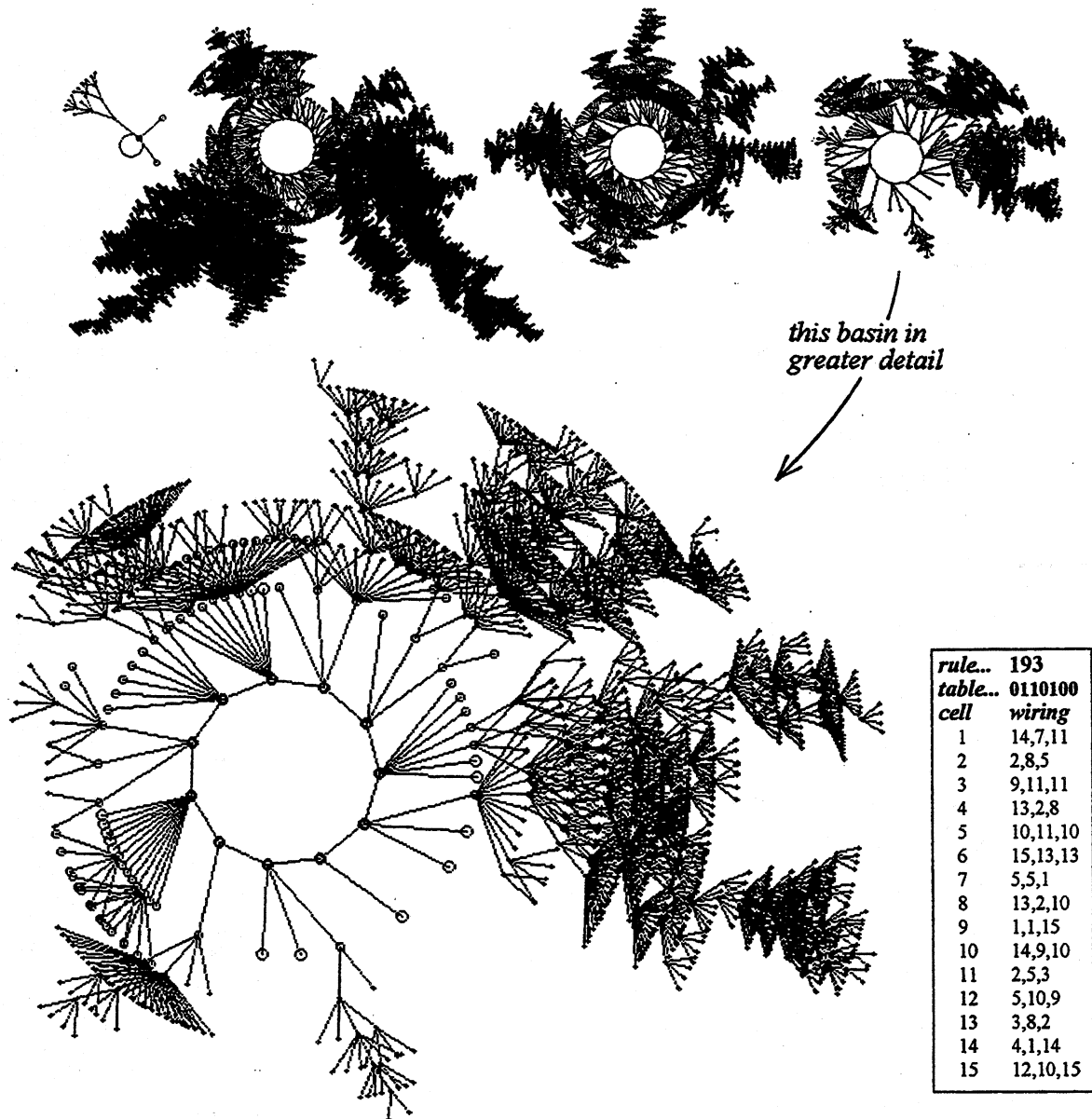
represented. The resulting pre-image stack is the complete set of pre-images of the given state, without duplication. An equivalent procedure is used for  $k$  values other than 3 or for mixed networks  $k$  networks.

This general reverse algorithm for RBN works for networks with any degree of intermediate architecture between RBN and CA, which of course includes CA of any dimension - the wiring scheme is simply set accordingly. Provided that  $k \ll n$  the algorithm is orders of magnitude faster than the exhaustive testing of state space.

### 5.6. Constructing and Portraying RBN Attractor Basins

As for CA, once the pre-images of a state are known, the information is applied to construct the pre-image fan, from the given state to its set of pre-images (if any). The pre-image fan for each pre-image is then computed, and so on, until only garden of Eden states remain. In this way transient trees (or sub-trees) are constructed. A basin of attraction (or the complete basin of attraction field) is constructed by first running the network forward to reveal the attractor cycle, then computing each transient tree in turn.

In the computer diagrams of attractor basins drawn in DDLab, representations of global states are linked by directed arcs. Each node will have zero or more incoming arcs from its pre-image nodes, but because the system is deterministic, exactly one outgoing arc (one *out-degree*). Nodes with no pre-images have no incoming arcs, and represent garden-of-Eden states. The number of incoming arcs to a node is its *in-degree*. The methods and the graphic conventions are explained more fully in chapter 2 and (Wuensche and Lesser 1992a). Figure 5.1 shows a typical basin of attraction of a random Boolean network (it is part of the basin of attraction field shown in figure 5.2). Note how the topology differs from CA attractor basins with their many symmetries (see chapter 1.4), for example as in figure 4.16. These symmetries generally can not occur in RBN.

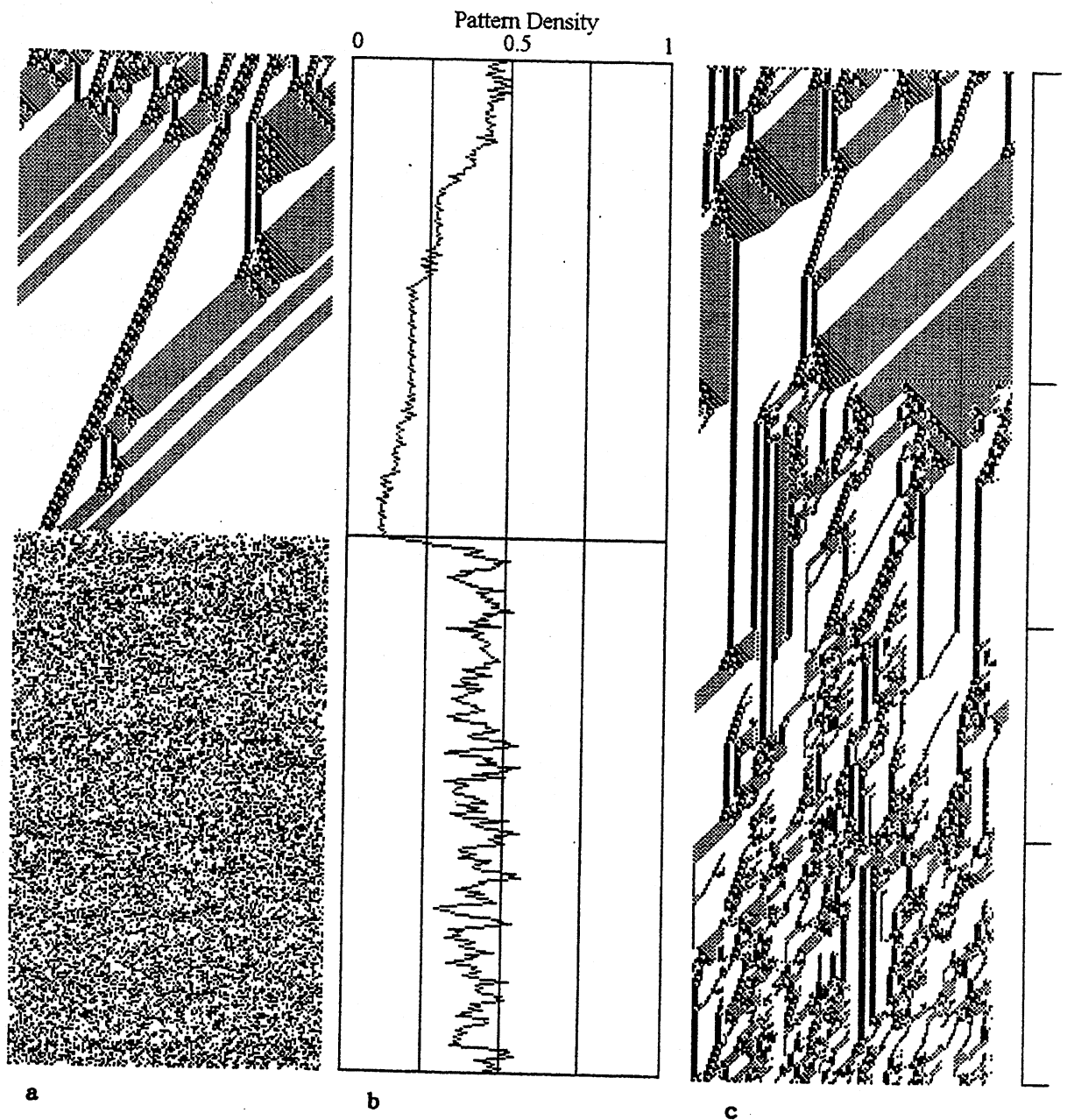


**Figure 5.5.**

*Top:* The basin of attraction field of a randomly wired, single rule network (non-local CA).  $n=15$ ,  $k=3$ , rule 193. The  $2^{15}=32768$  states in state space are organised into 4 basins of attraction. The total number of states in each basin is as follows (with attractor period in brackets): 24 (1), 26926 (22), 3498 (17), 2320 (11). The field has 27057 garden-of-Eden states,  $G$ -density = 0.823.

*Bottom left:* The last basin shown in greater detail (with the same  $G$ -density = 0.823).

*Bottom right:* The pseudo neighbourhood wiring scheme.



**Figure 5.6.** Single rule, local and random wiring, (non-local CA),  $n=150$ ,  $k=5$ , rule bc 82 66 d4  
 a) The space-time pattern of a complex rule with local wiring from a random initial state. After about 240 time steps, the wiring scheme has been totally randomised.  
 b) The pattern density (density of 1s) in successive global states analogous to an EEG.  
 c) The space-time pattern of the same rule from another random initial state. At the times indicated, 2% of the wires (15 out of 750) are cumulatively randomised.



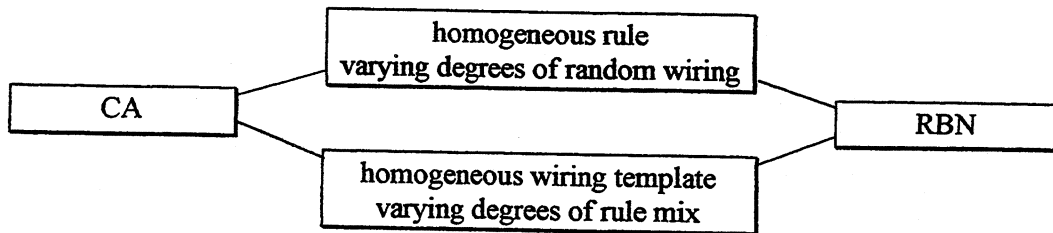


Figure 5.7. Intermediate architectures between CA and RBN.

### 5.7. Intermediate architecture between CA and RBN

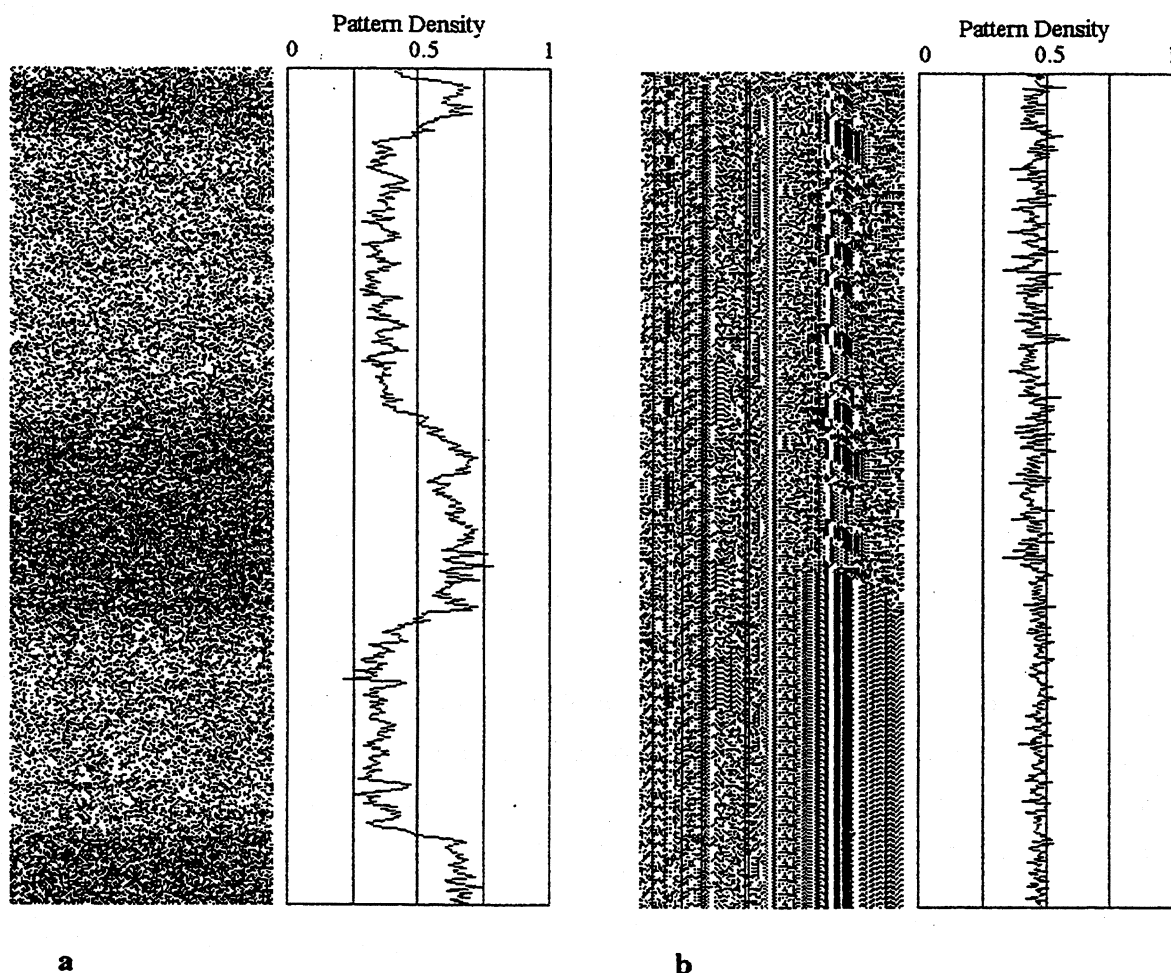
A spectrum of intermediate architectures between CA and RBN are indicated in figure 5.7. Just some aspects of CA wiring and/or rules might be altered. A network may have a homogeneous rule but random wiring, or homogeneous wiring but a random rule mix. There are many variations of intermediate architecture between CA and RBN, and also RBN where the random wiring or random rules are assigned with particular biases. Some varieties of intermediate architecture or biased RBN that have been studied are described below.

#### 5.7.1. Biased random Wiring

A variety of constraints can be imposed on random wiring. Hopfield's model (Hopfield 1982) mainly imposed only symmetric couplings between pairs of neurons, and a neuron did not connect to itself. Connections to a network element may be constrained to be distinct. Duplicate connections would correspond to unequal weights on single input lines (Kauffman 1984).

An example of the space-time patterns of a network with a homogeneous  $k=5$  rule but fully random wiring is given in the lower half of figure 5.6a. Space-time patterns appear random. This is not surprising since totally random wiring destroys the continuity of space. However the pattern density (the density of 1s) tends to settle and fluctuate at a characteristic level, though bi-stability is also possible as in figure 5.8a. The pattern density probably relates to the rule's internal homogeneity or  $\lambda$  parameter. Fluctuations are sometimes large with extended periods.

Networks with a homogeneous rule but partial or total random wiring (non-local CA) have been studied by Li (1991). He found that complex rules may result in a form of edge-of-chaos dynamics in these systems, by the emergence of co-operations among a cluster of components, which he calls *coherent clusters*. He also found evidence that the magnitude of density fluctuations becomes smaller for larger systems.



**Figure 5.8.** Examples of the space time patterns and pattern density plots in non-local CA (single rule/random wiring),  $n=150$ .

- a)  $k=5$  rule 7e e8 e8 81 with random wiring. This is a threshold rule set at 0.5 (majority rule) but with the end bits flipped, ie  $111 \rightarrow 0$  and  $000 \rightarrow 1$ . Note the bi-stable pattern density.
- b)  $k=3$  rule 193 with random wiring constrained to a 10 cell wide local periodic zone. Sub-attractors emerge.

Walker has studied basins of attraction by statistical methods for a  $k=3$  RBN where only the two outer wires were randomised, the central wire connects each cell to itself (Walker and Ashby 1966). The system thus retains some notion of space. A spectrum of constraints or biases on random wiring may be imposed in a similar way to retain a degree of spatial information in a network of given dimension and geometry. A varying proportion of wires from a CA-like neighbourhood could be set at random. Random wiring may be confined to a local periodic zone with a particular radius. This seems to produce sub-attractors as in figure 5.8b. Each network element could be connected to itself as in Walker's model, or alternatively self-wiring could be excluded.

To illustrate how coherent space-time structures are degraded by randomising *local* CA wiring, figure 5.6a shows the space-time pattern of a complex CA rule,  $k=5$ ,  $n=150$ . Gliders emerge from a random initial state. After approximately 240 time-steps, the wiring is totally randomised (but not the rule), All coherent space-time structure appears to be destroyed. Figure 5.6b shows the pattern density against time; irregular periodic fluctuations are apparent.

Randomising the wiring of the CA in stages will progressively transform a structured space-time pattern to a seemingly random pattern. Figure 5.6c shows the space-time pattern of the same complex CA from another initial random state. At the intervals indicated, 2% of all available wires (randomly selected) are cumulatively randomised (15 wires out of 750). Coherent structure is progressively eroded. Eventually, the space-time pattern will look like the lower half of figure 5.6a.

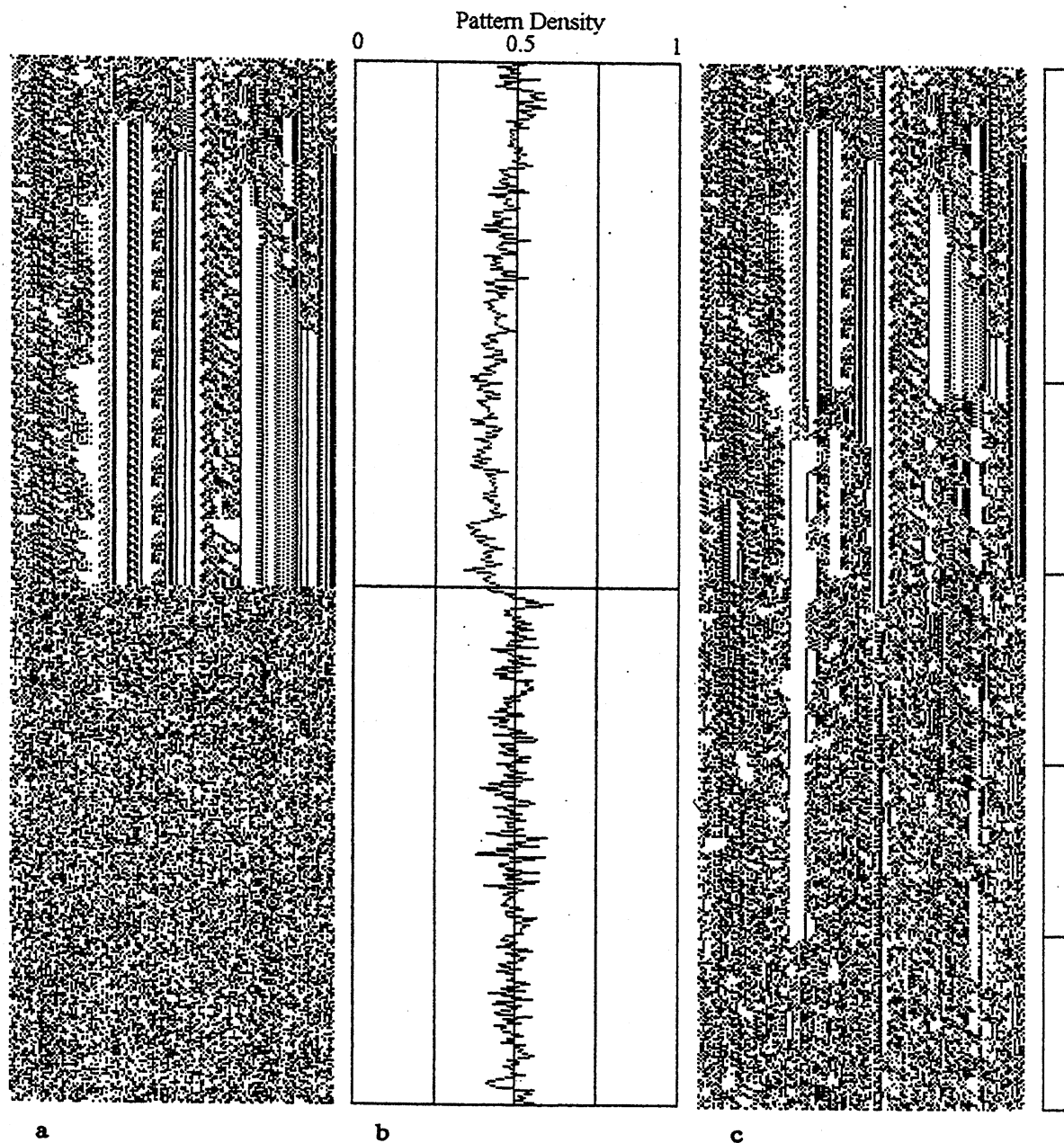
### 5.7.2 Biased random rule mix

A network's rule mix may be assigned at random, or the choice may be restricted to any combination of sub-categories of rules from rule-space, for instance rules with a particular setting of the  $Z$  parameter, or the  $\lambda$  (or equivalent  $P$ ) parameter. The rule mix can be restricted to only *additive* rules (Wolfram 1983), only threshold functions, or only Boolean functions that can be concisely expressed by the logical statements AND, OR and NOT. Setting the fraction of *canalising* inputs (Kauffman 1984) is a method of tuning a network between order and chaos, and also turns out to have biological significance. Order is characterised by the emergence and predominance (percolation) of "frozen islands" of unchanging elements in the network, chaos by the percolation of dynamic elements.

In networks with CA-like wiring but mixed rules, periodic structures confined within vertical bands will rapidly emerge within the space-time pattern. A random rule mix tends to compartmentalise space, depending on the degree of rule heterogeneity. The vertical features are local sub-attractors. In random networks (with both random wiring and mixed rules) local vertical features rapidly emerge for  $k=2$  rules due to the percolation of frozen elements (Kauffman 1993), but frozen elements become less dominant as  $k$  increases. At  $k=5$  space-time patterns appear chaotic, but with some residual vertical features (see figure 5.9a). This is due to the fact that Boolean functions with canalising inputs (and low  $P$ ) become increasingly unlikely to be set at random as  $k$  increases.

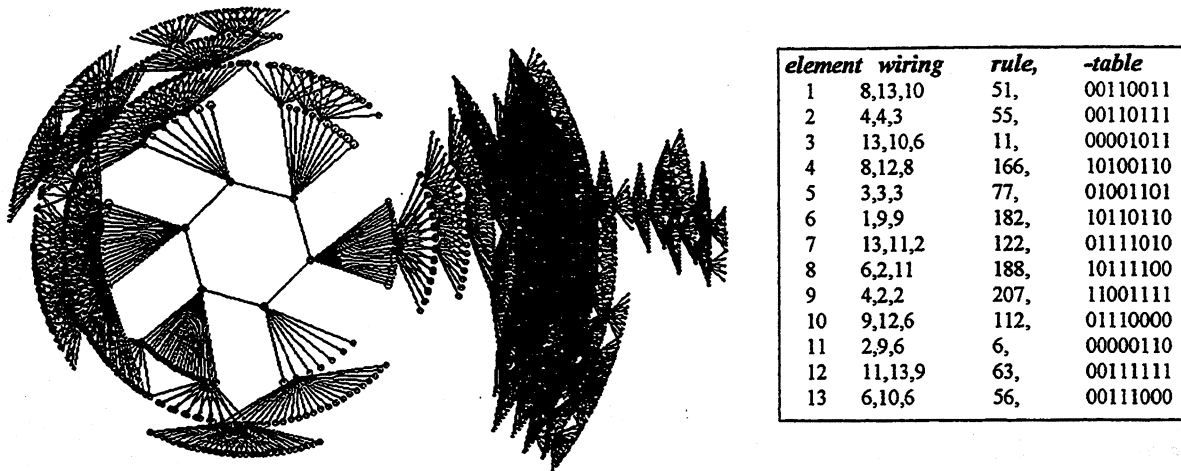
The pattern density probably relates to the *mean* internal homogeneity or  $\lambda$  parameter of all the rules in the network, corrected to allow for the proportion of output couplings from each cell.

Figure 5.9a shows the space-time pattern of a 1d network with CA-like wiring but mixed rules. Frozen islands and local sub-attractors emerge from a random initial state. After approximately 240



**Figure 5.9.**  $k=5$  mixed rule network with local and random wiring,  $n=150$

- a) The space-time pattern of with local wiring from a random initial state; sub-attractors (vertical features) emerge. After about 240 time steps, the wiring scheme is totally randomised. The space-time pattern becomes chaotic but some vertical features persist.
- b) Shows the pattern density (density of 1s) in successive global states analogous to an EEG.
- c) The space-time pattern of the same rule from the same random initial state. At the times indicated, 4% of the wires (30 out of 750) are cumulatively randomised.



**Figure 5.10.** The basin of attraction field of a random Boolean network with only one attractor, period 6.  $n=13$ ,  $k=3$ . The  $2^{13}=8192$  states in state space are all linked into one basin of attraction. The network's wiring/rule scheme is set out in the table.

time steps, the wiring is totally randomised (but not the rule scheme), resulting in the loss of all coherent structure, though some vertical features are evident. Figure 5.9b shows the pattern density. Figure 5.9c shows the space-time pattern of the same CA wiring/mixed rule network from the same initial random state. At the intervals indicated, 4% of all available wires (randomly selected) are cumulatively randomised (30 wires out of 750). Frozen island structure is progressively eroded. Eventually, the space-time pattern will look like the lower half of figure 5.9a.

### 5.8. Comparing CA and RBN.

Although there is much work to be done given the enormous behaviour space open to investigation, a few general observations can be made on the basin of attraction field topology of RBN, and intermediate architectures, on the basis of many computer runs to reconstruct attractor basins.

The various symmetries and hierarchies that dominate CA basin field topology (see chapter 1.4) are absent in RBN, though some symmetries are still evident in small randomly wired networks with a single homogeneous rule, or a limited rule-mix. This lack of constraint on the basin of attraction field topology, and the enormous parameter space, suggest that RBN (or a network of sparsely connected RBN) provide ideal mechanisms for flexible categorisation.

On the other hand, CA are discrete approximations of physical systems (Langton 1991), characterised by local connections. Just as the physical world has the potential for the spontaneous emergence of life, under certain edge-of-chaos conditions CA seem to support the spontaneous

emergence of analogous complex dynamical phenomena such as gliders. In turn, a key property of life is the emergence of complex networks with non-local connections. Biological systems are replete with non-local connections, from brains to economics.

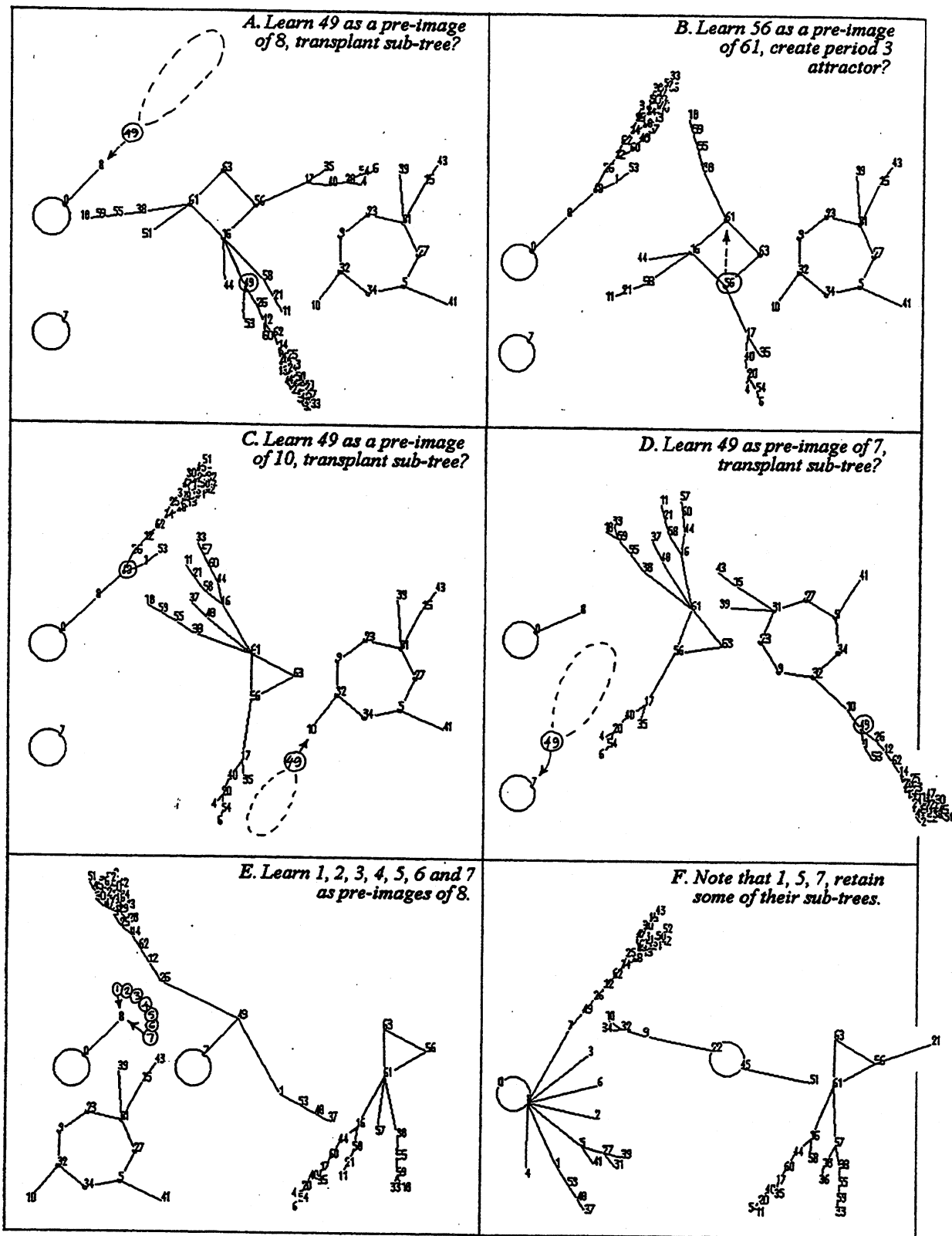
The basin structure of random Boolean networks is extremely varied, but for  $k < 4$  and other parameters set at random, the number of basins, attractor periods and transient lengths are generally very small and increase only slowly with system size, whereas in some categories of CA the growth may be exponential. Examples of just one attractor with a short period taking up the entire state space are not uncommon, as the example in figure 5.10 for a network with  $n=13$ ,  $k=3$ . This phenomena can explained by the significant proportion of rules with canalising inputs for  $k < 4$  (see section 5.11), and ties in with Kauffman's studies (Kauffman 1991) for large random Boolean networks (for  $n$  up to 10,000). He reported that for  $k=2$ , attractor cycles and the number of alternative attractors increased as  $\sqrt{n}$ , and at a increasing rate with greater  $k$ .

A systematic investigations of the topology and structure of basins of attraction for random Boolean networks and various intermediate architectures, using the tools now available, has yet to be done.

### 5.9. RBN Learning Algorithms.

As discussed in the introduction (section 5.1), attractor basins constitute the network's memory in the sense that separate basins in the basin of attraction field, and each node onto which dynamical flow converges, categorise state space. All the network's states other than garden-of-Eden states are content-addressable memories. Any external input will automatically initiate a dynamical flow along a unique chain of states. Each successive state categorises states in its transient sub-tree, far from equilibrium, forming a complex hierarchy of categorisation culminating at the attractor. The set of attractors and their branching trees constitute the network's collective memory.

This section sets out algorithms for directed learning that enable a random Boolean network to learn new transitions from experience (and also to forget) by changing the networks wiring and/or rules. The approach could be seen in the context of Kauffman's  $nk$  fitness landscape - "a walk in parameter space seeking good attractors" (Kauffman 1993), though here we are seeking good *basins* including the structure of their sub-trees, as well as attractors. These algorithms, implemented in DDLab, require a "teacher" to set the new transitions as pre-images to be learnt or forgotten, after which the procedure is automatic. Note that the side effect of learning, i.e. changes elsewhere in the basin of attraction field, can be severe if the new pre-images to be learnt require large changes to the network. Fewer side effects result from forgetting as will be made clear below. These learning algorithms may form the basis for fine tuning networks that are already close to some desired



**Figure 5.11.** A sequence of learning steps, A to F, in a 6 element network. Note the stability of basin structure as well as the side effects of learning at each step. Bit patterns are represented by decimal numbers.

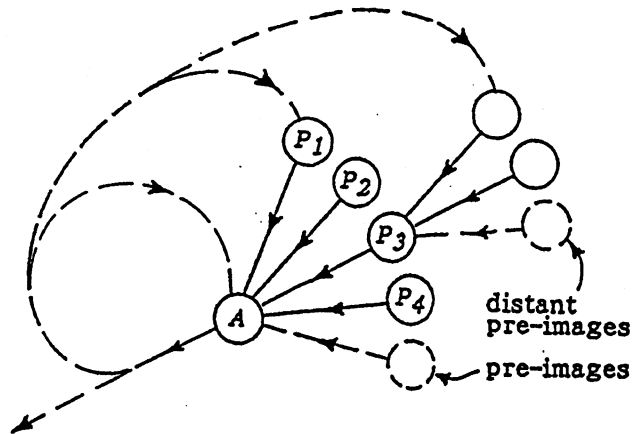


Figure 5.12. States  $P_1, P_2, P_3, \dots$  etc. may be learnt as pre-images of the state  $A$ . Distant pre-images of  $A$  may also be learnt, for instance as pre-images of  $P_3$ . Learning  $A$  as a pre-image of itself creates a point attractor. Learning  $A$  as a distant pre-image of itself creates a cyclic attractor. If  $A$  is learnt as the pre-image of some other state in the basin of attraction field, the states flowing to  $A$ , its transient sub-tree, may be fully or partially transplanted along with  $A$ .

behaviour, but are inappropriate for designing networks from scratch. This might be achieved with "reverse engineering" algorithms discussed section 5.9.3 below.

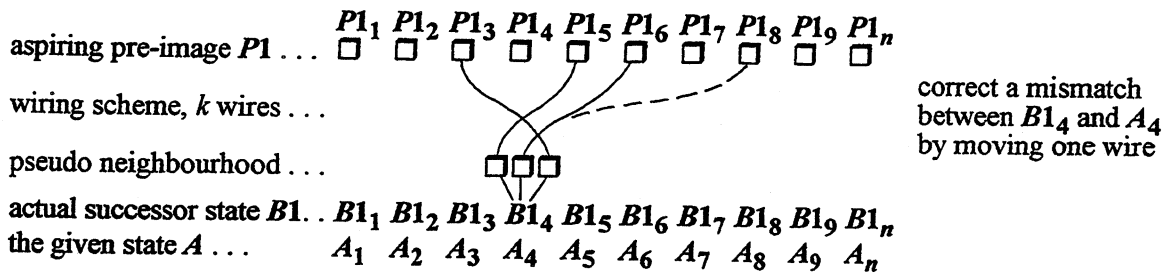
Suppose we want to make the state  $P_1$  the pre-image of state  $A$ . Any mismatches between element values of the actual successor state  $B_1$  (of the aspiring pre-image,  $P_1$ ) and state  $A$  can be corrected in one step by either of two methods, adjusting the network's wiring or rule scheme. The two methods have very different consequences.

Before learning starts, a network with a wiring/rule scheme must already be in place. If the relevant transitions in the basin of attraction field are already close to the desired behaviour, the side effects of learning will be minimised. The wiring/rule scheme may be selected from an *atlas* of basin of attraction fields, such as in (Wuensche and Lesser 1992a). It may be pre-evolved from a population of wiring/rule schemes using a genetic algorithm, or built from scratch by one of the methods for solving the inverse problem (Askenazi 1996, Myers 1966). In the worst case the wiring/rule scheme is assigned at random.

### 5.9.1. Learning by re-wiring

Consider the state  $P_1, (P_{1_1}, P_{1_2}, \dots, P_{1_n})$ , that the network is to *learn* as a pre-image of the given state  $A, (A_1, A_2, \dots, A_n)$ . When  $P_1$  (the aspiring pre-image) is evolved forward by one time-step according to its current  $k$ -neighbour wiring/rule scheme its actual successor state is  $B_1, (B_{1_1}, B_{1_2}, \dots, B_{1_n})$ . If the states  $P_1$  and  $A$  were selected at random, then  $B_1$  will probably have mismatches with  $A$  in about  $n/2$  locations.





Suppose that  $B1_4 \neq A_4$ . One wire, or more if necessary, of the  $B1_4$ 's wiring couplings is moved to a new position. We will limit this analysis to single wire moves only. Any move resulting in a pseudo neighbourhood with opposite output (according to that element's look up table) will correct the mismatch. This is a stochastic method as there are likely to be many alternative successful wire moves.

Assuming that  $P1$  has a roughly equal proportion 0s and 1s, there will be  $\approx n/2$  alternative positions where a wire move will pick up a changed value. Given  $k$  wires there will be  $\approx k \times n/2$  alternative re-wiring options that will change the pseudo neighbourhood. Assuming the rule at position 4 has  $\lambda \approx 0.5$ , (the proportion of 1s in the rule table),  $\approx 1/2$  of the changed pseudo neighbourhoods will change the value of  $B1_4$  so that  $B1_4 = A_4$ . The number of valid re-wiring options to correct each mismatch is therefore  $\approx k \times n/4$ . The choice for re-wiring may initially be selected at random from among the valid options to correct each mismatch making  $P1$  the pre-image of  $A$ .

Suppose that another aspiring pre-image,  $P2$  (with an actual successor  $B2$ ) is to be learnt as a pre-image of  $A$  by re-wiring, but without forgetting  $P1$ . If  $B2_4 \neq A_4$  there will be  $\approx k \times n/4$  valid re-wiring options to correct the mismatch. However  $\approx 1/4$  of the options would change the value of  $B1_4$ , forgetting  $P1$  ( $\approx 1/2$  the wire moves would change the pseudo neighbourhood of  $B1_4$ ,  $\approx 1/2$  of which will have outputs different to  $B1_4$ ).

If several pre-images are to be learnt in succession by single wire re-wiring, without forgetting any previously learnt pre-images, the space of valid re-wiring options for a particular element will be reduced by  $\approx 1/4$  for each mismatch correction. The shrinking of the valid re-wiring space is sensitive to the initial re-wiring choices, and the order of learning. Some re-wiring choices will be fitter than others, affecting the capacity of the network to learn that particular set of pre-images.

If the pre-images are close to each other in terms of Hamming distance, there will be fewer mismatches to correct thus greater learning capacity. Close pre-images are likely to be learnt by default so the network is able to generalise from examples.

What is the probability of *forgetting* pre-images on the same fan, or transitions elsewhere in the basin of attraction field, as a result of learning by re-wiring?

Consider an arbitrary transition  $X \rightarrow Y$  elsewhere in the basin. The probability of *forgetting* the transition by moving one wire is given by,

$$F_1 = 1/4$$

by moving two wires,  $F_2 = F_1 + F_1(1 - F_1)$

by moving  $n$  wires,  $F_n = \sum_{i=0}^{n-1} \left(1 - \frac{1}{4}\right)^i \times \frac{1}{4}$

The capacity of the network to learn more states as pre-images of a given state will thus depend on a number of factors: the original wiring/rule scheme, the similarity of the new pre-images, the order of learning, and the choice of re-wiring options. However, the network may have additional capacity to learn *distant* pre-images, further upstream in the transient tree as in figure 5.12. Note that if the network learns the given state itself as its own pre-image, this will result in a point attractor. If the state is learnt as a distant pre-image, this will result in a cyclic attractor with a period equal to the distance.

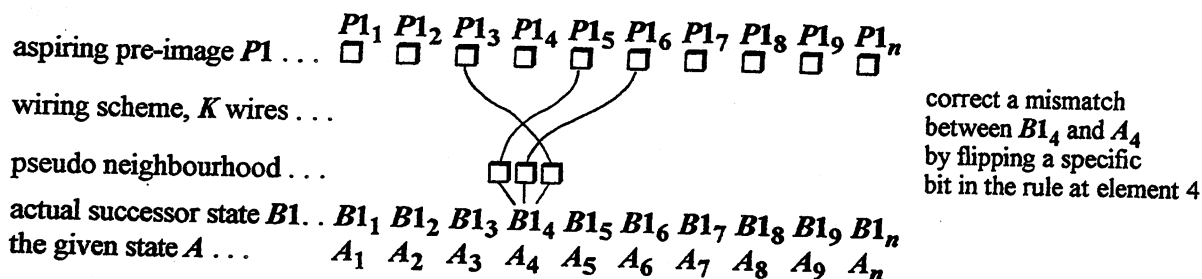
### 5.9.2. Learning by mutating the rule scheme

To correct a mismatch between an element in  $A$  and the corresponding element in the successor of an aspiring pre-image,  $P_1$ , by mutating the rule scheme, one specific bit in that element's rule table is flipped (changed from  $0 \rightarrow 1$  or  $1 \rightarrow 0$ ). There is only one option, certain to succeed. Adding another aspiring pre-image by the same method cannot result in  $P_1$  being forgotten. This is because any mismatch between a particular element in the successor state  $B_2$  (of the *aspiring* pre-image,  $P_2$ ) and  $A$  can not relate to the same rule table entry that was 'looked up' to determine  $P_1$ 's successor. Otherwise there would have been no mismatch. Any change to correct the mismatch must be to a different rule table entry;  $P_1$ 's successor cannot be affected. It turns out that there is no limit to the number of pre-images of a given state that can be learnt by this method, and no risk of forgetting previously learnt pre-images of the state, but of course there may be side effects elsewhere in the basin of attraction field.

As an extreme example, all states in state space can be made the pre-images of any arbitrary state  $A$ . The two trivial rules with rule tables consisting of only 0s or only 1s may be allocated to network elements in  $A$  according to whether an element equals 0 or 1. The result will be a basin of attraction field consisting of a single point attractor. All other states in state space will be garden-of-Eden pre-images of the point attractor.

Consider the state  $P1$ ,  $(P1_1, P1_2, \dots, P1_n)$ , that the network is to learn as a pre-image of the given state  $A$ ,  $(A_1, A_2, \dots, A_n)$ . When  $P1$  (the aspiring pre-image) is evolved forward by one time-step

according to its current  $k$  neighbour wiring/rule scheme its actual successor state is  $B1$ ,  $(B1_1, B1_2, \dots, B1_n)$ .



Suppose that  $B1_4 \neq A_4$ . To correct the mismatch, the rule at element 4 is mutated by flipping the bit in its rule table corresponding to the pseudo neighbourhood at  $B1_4$ . There is only one option. As the rule table has  $2^k$  bits, flipping one bit represents a change of  $1/2^k$  in the rule table, i.e. for 3-neighbour wiring  $1/8$ , for 5-neighbour wiring  $1/32$ . This is the probability of *forgetting* another transition elsewhere in the basin of attraction field, outside the immediate pre-image fan where the probability of forgetting = 0. For larger  $k$  the probability of a rule-table bit flip resulting in forgetting some arbitrary transition,  $X \rightarrow Y$  becomes smaller at an exponential rate as  $k$  increases. It was shown in (Wuensche and Lesser 1992a) that a 1 bit mutation (to each network element, i.e. flipping a total of  $n$  bits) in a 5-neighbour CA resulted generally in a small change in the basin of attraction field.

Consider an arbitrary transition  $X \rightarrow Y$  elsewhere in the basin. The probability of *forgetting* a transition by one bit-flip is given by,  $F_1 = 1/2^k$

by two bit-flips,  $F_2 = F_1 + F_1(1 - F_1)$

by  $n$  bit-flips,  $F_n = \sum_{i=0}^{n-1} \left(1 - \frac{1}{2^k}\right)^i \times \frac{1}{2^k}$

If a set of pre-images to be learnt are close to each other in terms of Hamming distance, there will be fewer side effects elsewhere in the basin of attraction field. Again, close pre-images are likely to be learnt by default; and the network is able to generalise from examples.

### 5.9.3. Sculpting the basin of attraction field and the inverse problem

Re-wiring has a much a greater effect on basin structure than mutating the rule scheme, but in either case the stability of basin structure is noteworthy. Using these methods, point attractors, cyclic attractors and transient sub-trees can be created. Transient sub-trees are sometimes transplanted along with the repositioned state (see figure 5.11), indicating how learnt behaviour can be re-applied

in a new context. Generalisation is present, because bit patterns in the same pre-image fan are likely to be close in Hamming distance to each other, and so may be learnt by default from examples. Forgetting involves *pruning* pre-images and transient sub-trees, and is achieved by the inverse of the method for learning. Since it is sufficient to create just one mismatch in order to forget, the side effects are minimal as compared with learning.

These learning methods are appropriate as a basis for fine tuning pre-existing networks, but are inappropriate for building networks from scratch because of side effects. If applied to sculpt a randomly constructed network toward a particular desired attractor basin structure they would be likely to fail because the network would be so far removed from the desired dynamics. What is required is a method of building a network to satisfy a total or partial set of transitions. This has been named the "inverse problem" or reverse engineering of Boolean networks. The learning algorithms and access to attractor basins of RBN with the software DDLab has inspired some recent work to solve the inverse problem, and two alternative methods have recently been proposed independently by Manor Askenazi (1996) and John Myers (1997).

Askenazi has approached the problem in the context of genomic regulatory networks in collaboration with Roland Somogyi and myself (Somogyi *et al* 1997). Given a set of trajectories (according to the "gene expression matrix" of cell differentiation) find the subset of wiring and rules that will satisfy it. Askenazi's method simply assigns minimal wiring and rules that are compatible with the transitions in the set, that is he starts with an unconnected "network" and builds it up element by element. There may be many networks that will satisfy the transitions, produced according to the sequence with which element are taken. The attractor basins of the various solutions are reconstructed with DDLab. Various methods are being developed to narrow the candidate solutions to those that might be the most biologically plausible.

Myers has approached the problem from an opposite direction. His motivation was to improve on the learning algorithm presented in (Wuensche 1993a), to find the rule wiring scheme that will satisfy a complete basin of attraction field for small networks, i.e. all transitions. His solution is to start with a fully wired network which is built directly from transitions, thus they are satisfied by definition. A fully connected RBN can satisfy any random mapping (see chapter 3). He then employs methods for pruning redundant connections (if any) to find the minimal random map (basin of attraction field). His solution is verified and reconstructed with DDLab.

Future work will aim to combine and develop these methods. The methods may have potential in areas where artificial neural networks are applied such as pattern recognition, in understanding genomic networks and perhaps in other areas as yet unpredictable.

### 5.10. The Emergence of Memory

This section is a discussion of the implications of attractor basins in the dynamics of discrete networks on understanding memory in animal brains. Here I allow myself a large measure of conjecture and speculation. The first conjecture, perhaps no longer particularly controversial, is that the brain is a vastly complex dynamical system\*. Cognitive functions such as memory, learning, volition, and consciousness seem to emerge as high level properties from the activity of billions of neurons sparsely connected into complex networks. What is it about networks that allows such emergence to happen?

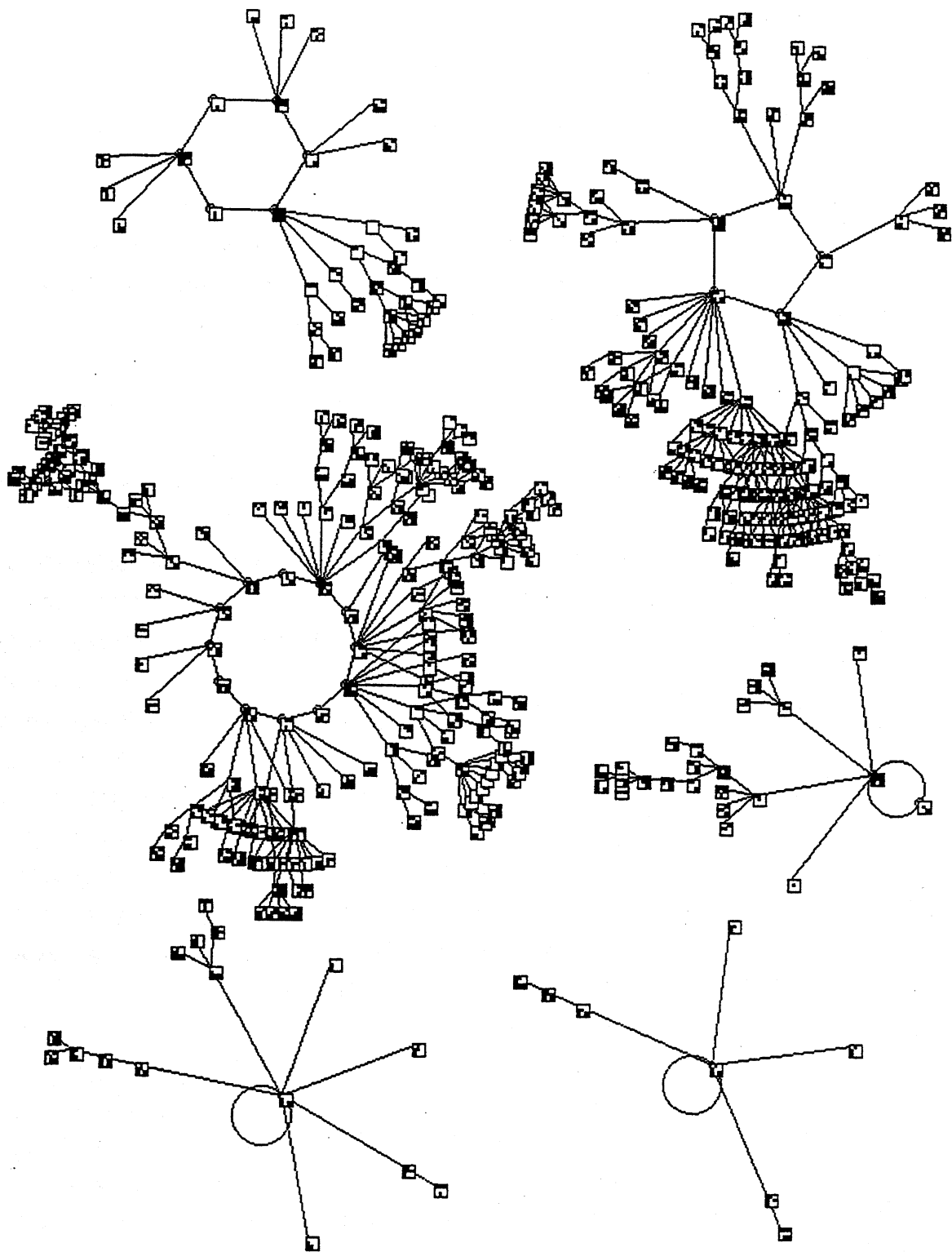
I will argue that the basic level of emergence is a network's ability to categorise its space of possible patterns of distributed activation. State space is not just categorised by attractors. Categorisation also occurs far from equilibrium, within the long transients leading to attractor cycles, giving a role to chaotic dynamics. The basin of attraction field into which a network self-organises its state-space is a space-time abstraction representing the network's memory. When networks link up with other networks, the system is able to use memory in individual fields to provide the components for the emergence of higher level cognitive states. RBN provide a simple yet powerful neural model for understanding attractor basins in this context.

Any artificial network model is necessarily a huge oversimplification compared to biological networks. However, RBN which allow Boolean functions, in contrast to just threshold functions as in classical artificial neural network models, arguably capture some essence of the logic implicit in the complex topology of each biological neuron's dendritic tree and synaptic microcircuitry, which is thought to play a vital role in brain function (e.g. Shepherd 1990). A Boolean function could be implemented in hardware by a combinatorial circuit, a sort of artificial dendritic tree.

If the emergence of memory can be demonstrated in such simple networks, perhaps insights may be gained by analogy to the vastly more powerful processes that occur in animal brains. Memory is a relative concept, meaning the creation of useful categories for an organism's adaptive behaviour out of the plethora of sensory inputs arriving at neural sub-networks, and the even greater internal traffic between sub-networks. The idea of networks of sub-networks in a sort of nested hierarchy is of course another necessary oversimplification, because the boundaries of sub-networks are unclear.

---

\* This paradigm contrasts with notions of the brain as a system doing computation like a Turing machine, or related notions of the brain as a system that manipulates symbols or "representations", approaches that I will not pursue.



**Figure 5.13.** The basin of attraction field of a RBN,  $n=9$ ,  $k=4$ . The  $2^9=512$  states in state space, shown as patterns on a  $3 \times 3$  grid, are organised into 6 basins, with attractor periods ranging from 1 to 12. The number of states in each basin is: 39, 191, 234, 24, 16, 8. The network's wiring/rule scheme is shown in figure 5.14.

To be useful for adaptive behaviour, memory-categories should fall naturally into hierarchies of sub-categories. They should be highly reliable yet easily changed to permit learning, and access should be extremely fast.

How discrete dynamical networks organise their state-space into attractor basins has already been described. Typically, the connection topology is branching trees, representing non-equilibrium states, rooted on attractor cycles representing the equilibrium states available to the network. State space is not just categorised by attractors. Subtrees within individual basins, formed by merging trajectories leading to attractor cycles, produce hierarchies of sub-categories. The basin topology is characterised by long transients, verging on dynamics analogous to chaos in continuous systems. The vast majority of state-space is typically *far from equilibrium*, being distant in time from equilibrium represented by attractors.

The idea that state-space is partitioned by attractors is the generally accepted paradigm for "content-addressable" memory in artificial neural networks, following Hopfield (1982) and others. However, RBN with unbiased Boolean functions and connectivity  $k \geq 5$  are in the "chaotic regime" (Kauffman 1993). The chaotic regime is characterised by long transients. In systems of even modest size (i.e.  $n=150$  as illustrated in figure 5.9) a very large, probably astronomical, number of steps through state-space would typically be required for the system to settle at the equilibrium of one of its attractors. Although genomic regulatory network seen to have Boolean functions biased towards the ordered regime (see section 5.11), biological neural networks may well operate in the chaotic regime, especially as the connectivity is extremely high (i.e.  $k \approx 10,000$ ). If this is the case, explaining memory just by attractors poses the difficulty of the long time needed to reach attractors in large networks, whereas reaction times in biology are extremely fast. This problem may be overcome by the realisation that categories occur in subtrees, far from equilibrium.

#### 5.10.1. Memory, far from equilibrium

The notion of memory far from equilibrium along merging transients may answer a basic difficulty in explaining memory by attractors in biological neural networks. A view of the brain as a complex dynamical system made up of many inter-linked specialised neural sub-networks is perhaps the most powerful paradigm currently available. Sub-networks may consist of further sub-categories of semi-autonomous networks, and so on, which contribute to re-setting or perturbing each other's dynamics.

A biological neural sub-network is nevertheless likely to be extremely large. As explained above, the time required to reach an attractor from some arbitrary global state will probably be astronomical. Even when an attractor is reached, it may well turn out to be a long cycle or a quasi-infinite chaotic attractor. The notion of memory simply as attractors seems to be inadequate to account for the extremely fast reaction times in biology.

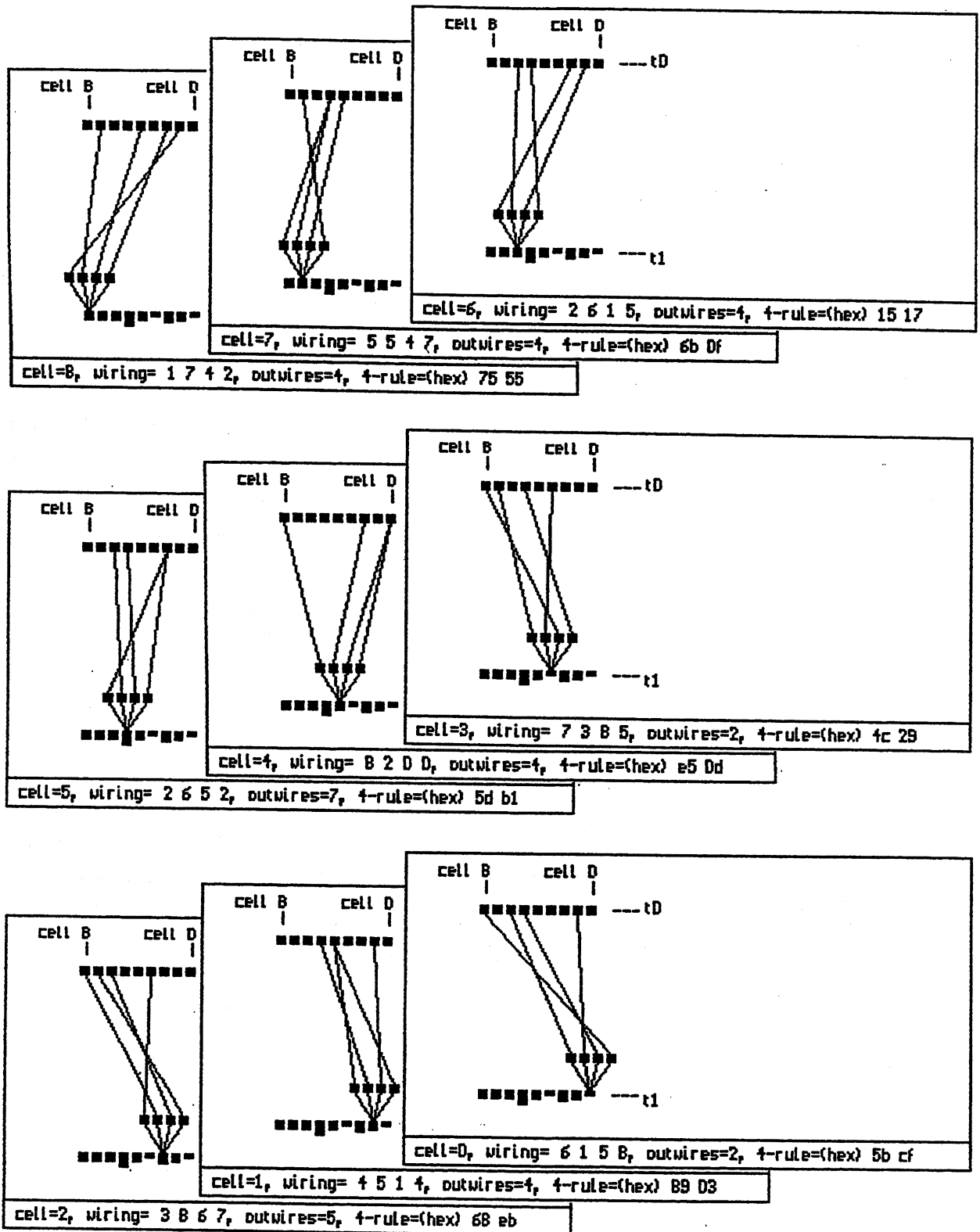


Figure 5.14. The wiring/rule scheme of the RBN shown in figure 5.13.  $n=9$ ,  $k=4$ . The wiring specifies couplings from the pseudo neighbourhood of each network element or cell (at time  $t_0$ ) to elements at the previous time-step,  $t_1$ . Different heights of elements at  $t_0$  indicate their number of output wires. Each element's rule is shown in hex. The parameters were chosen at random.



A discrete dynamical system with synchronous updating categorises its state space reliably along transient trees, far from equilibrium, as well as the attractors. A network that has evolved or learnt a particular global dynamics may be able to reach memory categories in a few steps, possibly just one. Moreover, the complex transient tree topology in the basin of attraction field, makes for a much richer substrate for memory than attractors alone, allowing hierarchies of memory sub-categories.

According to this approach, the detailed structure of a biological neural sub-network has found its form, and thus its basin of attraction field, by evolution, development and learning, where networks of sub-networks maintain each other far from equilibrium. Within this vastly complex dynamical system higher level cognitive properties based on lower level memory categories are able to emerge.

Deterministic transient tree topology, as described in this paper, requires synchronous updating in the network, though *asynchronous* updating might conceivably produce analogous behaviour provided that the updating was deterministic, and not random as in Hopfield's model (Hopfield 1982). Deterministic asynchronous updating could be implemented by adding an extra parameter to RBN architecture; the *length* of each wire coupling as proposed by (Harvey *et al* 1993), possibly in discrete intervals. A standard speed of signal transmission applied to a network with variable discrete length wires is equivalent to a network with just single length wires, but couplings to network states *before* the previous time step, for example a cell at time  $t$  might have connections to cells at  $t-2$ ,  $t-3$ ,... as well as to  $t-1$ .

A difficulty with this model is that a given network state at a particular instant may have multiple successors, and would be likely to occur many times in an enlarged state space. Alternatively, the states in the system's state space would need to consist of multiple time steps, not just one. Network architecture with "higher order in time" (Wolfram 1984a) or "historical time reference" (Wuensche and Lesser 1992a) may be worth pursuing in future work.

There is evidence that the firing of nearby biological neurons in the brain is strictly related in time. The classical view explaining "neural coding", how information is encoded by neural firing patterns, is based on the instantaneous nerve impulse *frequency* of individual neurons irrespective of the activity of their neighbours. However, various alternative temporal and synchrony code theories are gaining ground among neuroscientists, where the combined activity of subgroups of neurons encode information. For an excellent overview see (Stevens and Zador 1995). According to the synchrony code approach, information is encoded by the shifting synchronous pattern of activation over a population of neurons (e.g. Singer and Grey 1995). This requires mechanisms for synchronising activity. Although there is a "master clock" hypothesis (Hopfield 1995), synchronous activity could alternatively be explained as an emergent process.

Phase locking of spike discharges between neighbouring cells has been observed (Grey *et al* 1989, Singer 1991, Wasserman 1992), extending up to 7mm across the cortex . Synchronous firing

may be mediated by interneurons, which lack axons (Shepherd 1990), or mechanisms relying on close physical proximity between neurons (their dendrites, cell bodies and axons). Gap junction effects (physical connections between neurons made by large macro molecules), and ephaptic interactions (the local electrical field) serve to synchronise local neuronal activity (McCormick 1990). Further evidence for synchronisation is provided by Stevens and Zador (1995) who report,

"There are many examples of neuronal responses that are tightly locked to external stimuli. *In vitro*, spike transduction is essentially deterministic; current injected into a cortical neuron can reliably and repeatedly drive spikes with very little jitter\*. In the intact cochlea, firing is phase-locked to better than one millisecond precision. The firing of cortical neurons can also show precise stimulus locking, as in the often-overlooked case of the onset-transient that follows a light flash. These results show that the possibility of a code based on timing precision is not precluded by the biophysical substrate"

#### 5.10.2. RBN as a neural model

RBN may serve as a model of a semi-autonomous patch of neurons in the brain whose activity is synchronised. A network element's wiring scheme models that subset of neurons connected to a given neuron. Applying a Boolean function to an element's pseudo neighbourhood models the non-linear computation that a neuron is said to apply to these inputs to determine whether or not it will fire at the next time-step. This is far more complex than a threshold function (Shepherd 1990). The biological computation may depend on the precise topology of the dendritic tree, its microcircuitry of synaptic placements and intrinsic membrane properties. Networks *within* neurons based on the cytoskeleton of microtubules and associated protein polymers may be involved, suggested by Stuart Hameroff (1987) as the neuron's "internal nervous system". There appears to be no shortage of biological mechanisms that could perform the role of a Boolean function.

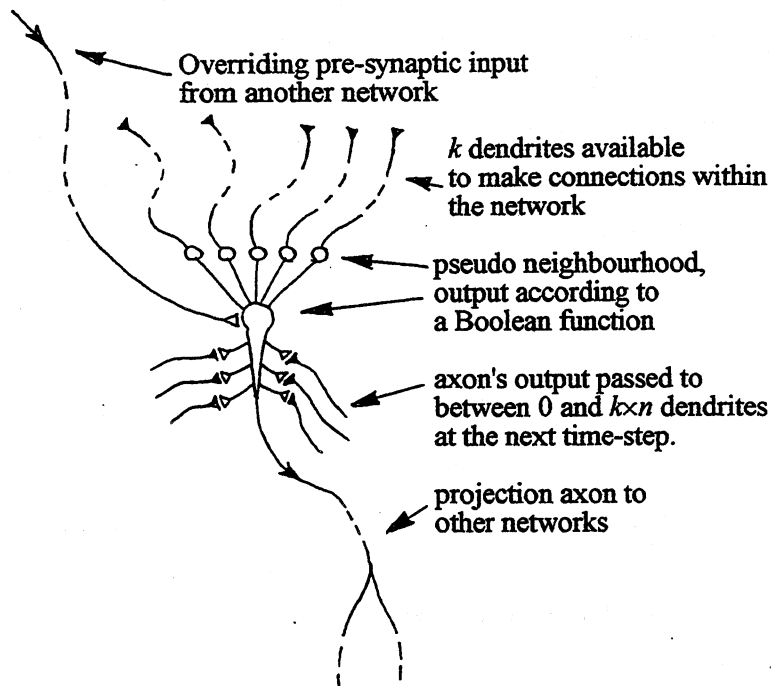
The elements of a RBN are arranged in an orderly array for convenience only, given fully random wiring their location may be arbitrary. One presumes that the actual location of neurons in the brain has been optimised through evolution to achieve high density by minimising the average length of connections that occupy space and consume resources.

To illustrate how a RBN could serve as a neural model, a hypothetical RBN model is shown in figure 5.16. A semi-autonomous population of 27 idealised neurons is distributed in 3 dimensions. Each neuron (figure 5.15) receives a post-synaptic excitatory (1) or inhibitory (0) signal from up to 5 neurons in the population (possibly including itself) via its 5 dendrites ( $k=5$ ), and computes a response signal to its axon according to its particular 32 bit Boolean function applied to its pseudo-

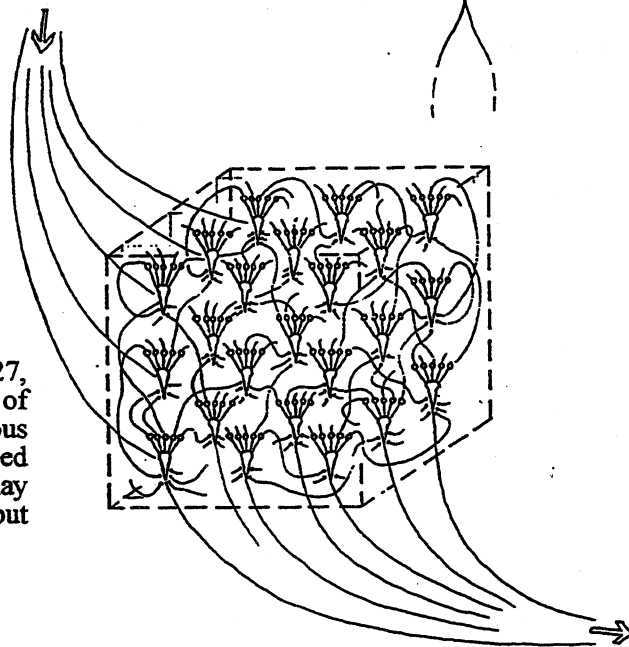
---

\* Mainen ZF, Sojnowski TJ: Reliability of spike timing in neocortical neurons, *Science* 1995, in press.

**Figure 5.15**  
A network element of a RBN represented as an idealised neuron with an overriding pre-synaptic contact from outside the system, and a projection axon to other networks.



**Figure 5.16**  
A RBN network ( $n=27$ ,  $k=5$ ) serving as a model of a semi-autonomous population of idealised neurons. Neurons may have input and output links to other networks.



neighbourhood. Let us suppose that the Boolean function would actually be encoded in the precise structure of the dendritic tree and synaptic microcircuitry which would provide the logic, equivalent to a combinatorial circuit with AND, OR and NOT gates. Thus all parameters driving the dynamics are physically present in wetware. The updating of axonic response is synchronous, and the process is iterated in discrete time steps.

The system is *semi*-autonomous because it must be capable of receiving input from outside to reset its global state, and also to communicate its current internal state to the outside via feed

forward channels. Thus each neuron in the model has overriding postsynaptic inputs from outside the population, and a projection axon to targets outside the population.

The model may be elaborated by weakly inter-connecting a number of such networks as in figure 5.17, so that the output of a particular network constituted the overriding input of another. Communication between networks may be *asynchronous*, and at a slower frequency with respect to a particular network's internal synchronous clock. Such an assembly of networks will have implicit in their particular pattern of connections at any instant, a vastly more complex but intangible web of interacting basin of attraction fields.

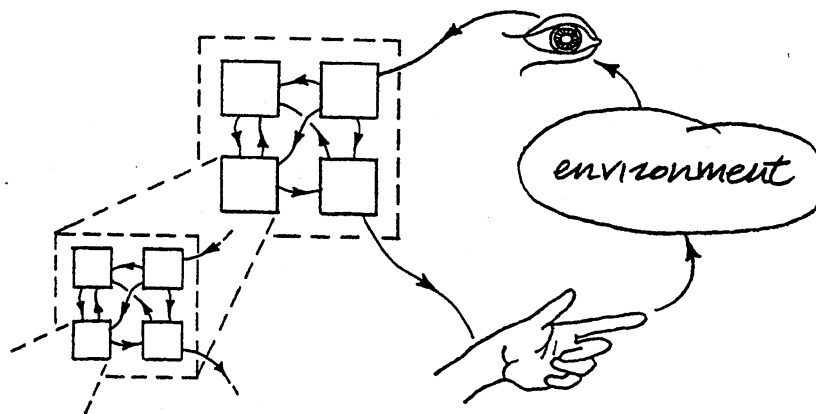
A given sub-network's basin of attraction field is implicit in its wiring rule/scheme. In a sub-network of biological neurons it would be implicit in the wetware. If sub-networks are linked as components in a super-network, indirect feedback could maintain each sub-network's dynamics in the outer branches of transient trees, far from equilibrium.

Categorisation of global states would be automatic and direct according to the particular deterministic trajectory (within the transient tree of the basin of attraction) that will inevitably follow. Recognition would involve other subnetworks receiving the networks axonal output.

Suppose a new state is imposed on sub-network  $n_1$  by another sub-network or sensory input. This would seed a determined trajectory in  $n_1$ , which opens up a sequence of categories along which the dynamics from the seed would flow. Recognition of these categories would involve another sub-network,  $n_2$ , activated by the axons of neurons in  $n_1$ . Recognition of categories in  $n_2$  would involve a third sub-network  $n_3$ .

After just a few steps,  $n_1$ 's dynamics might be reset to a new initial state, shifting the start point in its basin of attraction, by the axons of another sub-network belonging to this hypothetical super-network. Recognition in this system would be automatic and reliable because all trajectories in  $n_1$ ,  $n_2$ ,  $n_3$  etc. are inevitable given unchanged neuronal architecture. *Recall* in contrast to recognition would more difficult because a seed with the right association needs to be supplied. Folk psychology bears this out. I may know "Humphrey Bogart", but "its on the tip of my tongue". Recalling the name (dredging it out of my memory) in order to say it may be hard. But if I hear "Humphrey Bogart" mentioned, I will recognise it instantly, effortlessly and automatically.

Learning new behaviour implies self-amending the basin of attraction field by the self-adjusting of the wiring/rule scheme (by an unknown mechanism), thus some physical change to the wetware's neural architecture, and to dendritic tree and synaptic microcircuitry. Learnt behaviour often requires no mental effort; take driving a car. Such well rehearsed semi-conscious behaviour is delegated to a sort of "automatic pilot". The network's dynamics acts in the same way as for recognition, effortlessly and automatically. Learning to drive, on the other hand, requires mental concentration.



**Figure 5.17.** A neural model of weakly coupled semi-autonomous RBN linked to the environment. Each network may consist of a nested hierarchy of networks.

After all, physical changes in the brain must somehow be induced. By this argument, a conscious state coincides with the desire to learn or behave creatively, or the actual process of making the changes. How the appropriate change occurs in biological neural networks, and how the changes are triggered by the will to learn novel behaviour is unknown, probably one of the crucial unsolved problems in cognition. A sub-network may be able to alter the parameters of another. (Hameroff *et al* 1989,1993) suggest that cytoskeletal functions may provide retrograde signalling (analogous to back propagation in artificial neural networks) which may reconfigure intra-neuronal architecture

### 5.11. Genomic regulatory networks

Current work in collaboration with Stuart Kauffman<sup>1</sup> and Steve Harris<sup>2</sup> (Harris *et al* 1997), and with Roland Somogyi<sup>3</sup> (Somogyi and Sniegoski 1996, Somogyi *et al* 1996) applies RBN as models of parallel processing genomic regulatory networks and develops Kauffman's long established ideas in this area (e.g. Kauffman 1969,1984,1993).

The cells of living organisms differentiate within the developing embryo into the various cell types that form tissues by a process that is regulated at the molecular level by DNA sequences, encoding genes that produce proteins that regulate other genes. All eukaryotic cells in an organism carry an identical set of genes, some of which are expressed others not. A cell type is defined by the particular subset of genes that are expressed. The gene expression pattern of a cell type needs to be stable but also adaptable. What then is the mechanism that maintains the several hundred alternative stable patterns of gene expression of the various cell types making an organism?

Genes interact with each other in the genome, regulating each other's activity by coding for transcription factors (*trans* acting regulatory proteins), which may enhance or repress the expression

<sup>1</sup> Santa Fe Institute, Santa Fe, New Mexico

<sup>2</sup> University of Texas Health Science Centre at San Antonio, San Antonio, Texas

<sup>3</sup> Laboratory of Neurophysiology, NINDS, National Institutes of Health, Maryland

of other genes by binding at particular sites (at *cis* acting regulatory DNA sequences of genes). An inactive (or active) gene may require a combination of transcription factors (produced by other genes) to become and remain active (or inactive). The gene in question may produce transcription factors that effect the genes that effect *it*. Transcription factors may bind with each other as well as binding on *cis* acting elements within a gene. The result is a complex feedback web of genes turning each other on and off, acting at both long and short range relative to their sequence positions in the DNA. Additionally, genes are affected by signals from other cells. These tissue interactions drive development.

A particular gene directly regulates (i.e. has *outputs to*) just a small set of other genes, but many of those genes regulate other genes in turn. A gene can thus indirectly influence the activity of many genes *downstream*. Conversely, a particular gene is regulated directly by (i.e. has *inputs from*) just a small set of other genes. Those genes would have been previously regulated by other genes, so a gene is indirectly influenced by many other genes *upstream*. A gene may directly or indirectly contribute to regulating itself.

The system is a genomic regulatory network. To comprehend even small systems by following individual chains of cause and effect is a difficult task, let alone systems made up of thousands of genes (100,000 estimated in the human genome). The conceptual tools for understanding and simulating the behaviour of genomic networks are discrete dynamical networks as described in this thesis.

### 5.11.1 RBN as models of Genomic regulatory networks

There are many subtleties involved in the workings of a genomic regulatory network. Transcription is asynchronous and can be driven at different rates. However a first approximation is to model genomic networks as a dynamical system of interconnected on-off elements, the genes, updating in parallel according to the combinatorial logic of their inputs (Kauffman 1993). This is just a RBN defined by its wiring/rule scheme. There are various idealisations, for example updating a network in discrete time steps is an idealisation of many continuous (asynchronous) concurrent events. It still allows for variable protein concentrations because a gene may remain on for some fraction of a given time span. Signals affecting genes from neighbouring cells constantly perturb the network's dynamics. This can be modelled by a network of weakly connected RBN as discussed in the context of an RBN neural model (see figure 5.17).

When genomic regulatory networks are seen from a discrete dynamical network perspective, the problem of understanding how different stable cell types can exist with identical genes becomes more understandable. Cell types can be defined as the separate attractors or basins of attraction into which network dynamics will settle from various initial states. The trajectories can be defined as the

pathways of differentiation within a given cell. These genomic regulatory network models are the same in principle as memory by categorisation in RBN described earlier in this chapter. However, the quality of the dynamics, thus of attractor basin topology, and the quality of the evolved space-time patterns that might be appropriate for memory in neural networks is probably quite different from the dynamics required for effective cell types. In gene regulatory networks the dynamics seems to be biased towards the ordered regime, as will be discussed below, whereas in neural networks with their very high connectivity the dynamics is likely to be chaotic (see section 5.10).

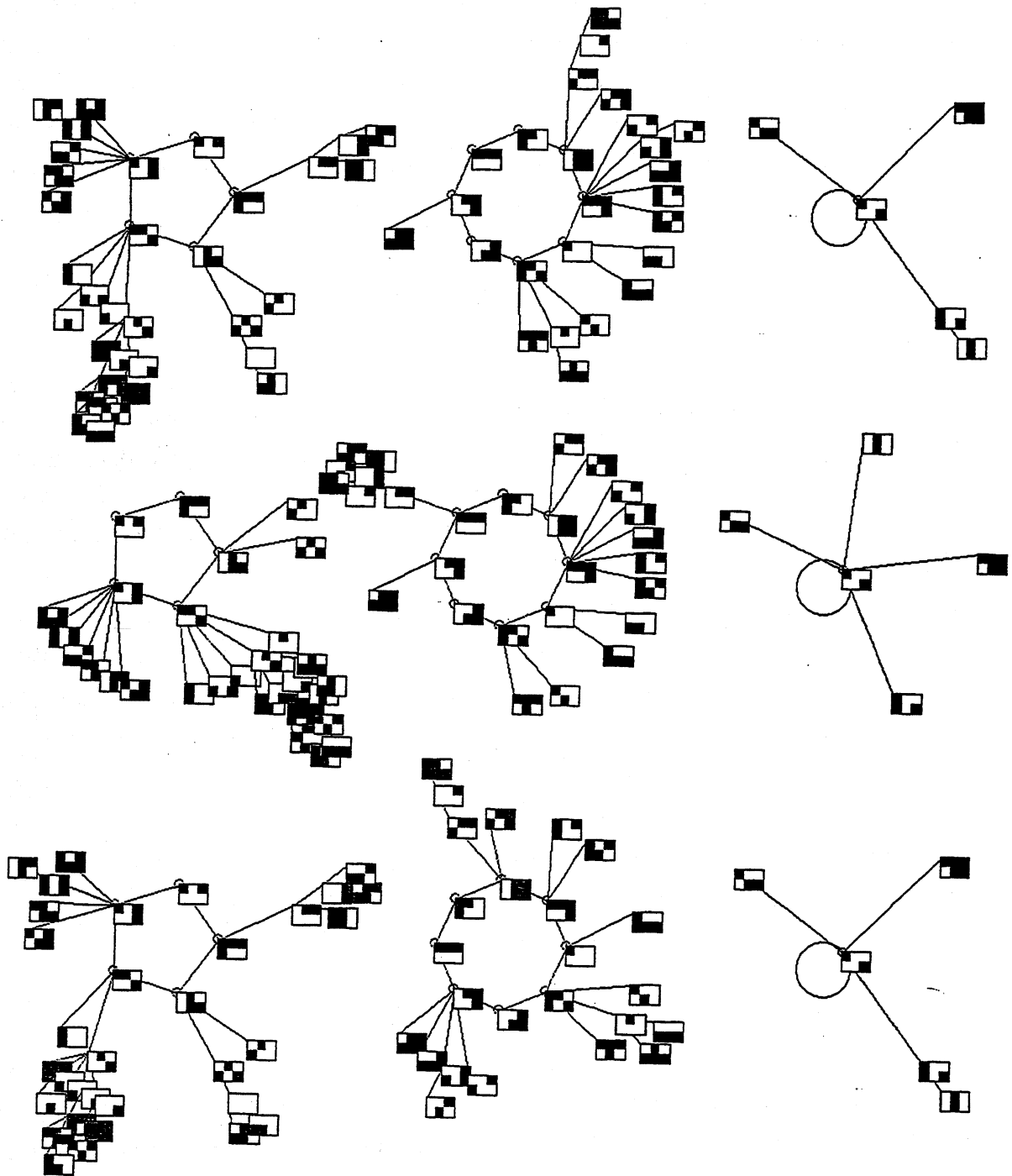
In a cell type's gene expression pattern over a span of time (i.e. its space-time pattern), a particular gene may, broadly speaking, be either on, off, or changing. If a large proportion of the genes are changing over time (i.e. chaotic dynamics), the cell will be unstable. On the other hand, a pattern where a large proportion of the genes are permanently on or off (frozen) may be too inflexible for adaptive behaviour. Cells of a particular type constantly need to adapt their gene expression pattern in response to a variety of hormone and growth/differentiation factors from nearby cells. Our definition of a cell type may be more correctly expressed as a set of closely related gene expression patterns allowing an essential measure of flexibility in behaviour. Too much flexibility might allow a perturbation to flip the dynamics into a different basin of attraction, a different cell type such as a cancer cell, or a bone cell to a fat cell. The appropriate dynamical regime turns out to be a delicate balance between stable on/off regions and dynamically changing regions in the space-time patterns. This can be visualised as a balance between order and chaos with its particular notion of a phase transition.

A cell type may achieve its identity on the attractor cycle, or on a "frozen skeleton", a transient outside the attractor but where the appropriate gene activation pattern has largely stabilised. This could be equivalent to the biological concept of specification or determination that proceeds overt differentiation.

### **5.11.2. Global and Local Measures.**

The DDLab software is used to model genomic regulatory networks, both their attractor basins and space-time patterns, and to fine tune various network parameters according to a variety of measures taken on local and global dynamics. The graphical display as well as the data itself provide insights into network dynamics.

Various order/chaos measures can characterise RBN dynamics and identify the order/chaos phase transition, both in general and in the context of genomic regulatory networks. These measures may be made on RBN represented either in 1d, but more usually in 2d with about  $100 \times 100$  network



<i>el.</i>	<i>wiring rule</i>
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 5.18**

The consequences of mutating a single bit in the rule table of just one element of a small RBN,  $n=6$ ,  $k=3$ . *Left*: Network parameters, rules shown in decimal. *Top*: The basin of attraction field of the RBN.

*Centre*: One bit mutation of the rule at element 2, changed to rule 4.

*Bottom*: One bit mutation of the rule at element 5, changed to rule 46.



elements, (10,000 genes) and up to a maximum of  $255 \times 255$ , approaching the size of the human genome (e.g. Harris *et al* 1997). This chapter does not set out to present the details of this work in progress, but merely to describe the measures themselves. A 2d RBN  $36 \times 36$ ,  $n=1296$ ,  $k=5$ , has been used where possible as a standard example, so that the various measures can be consistently related to each other for a range of rule (and wiring) parameters settings. Some results from (Harris *et al* 1997) are also presented. The methods described are available for a systematic investigation of RBN, and may also be applied to CA.

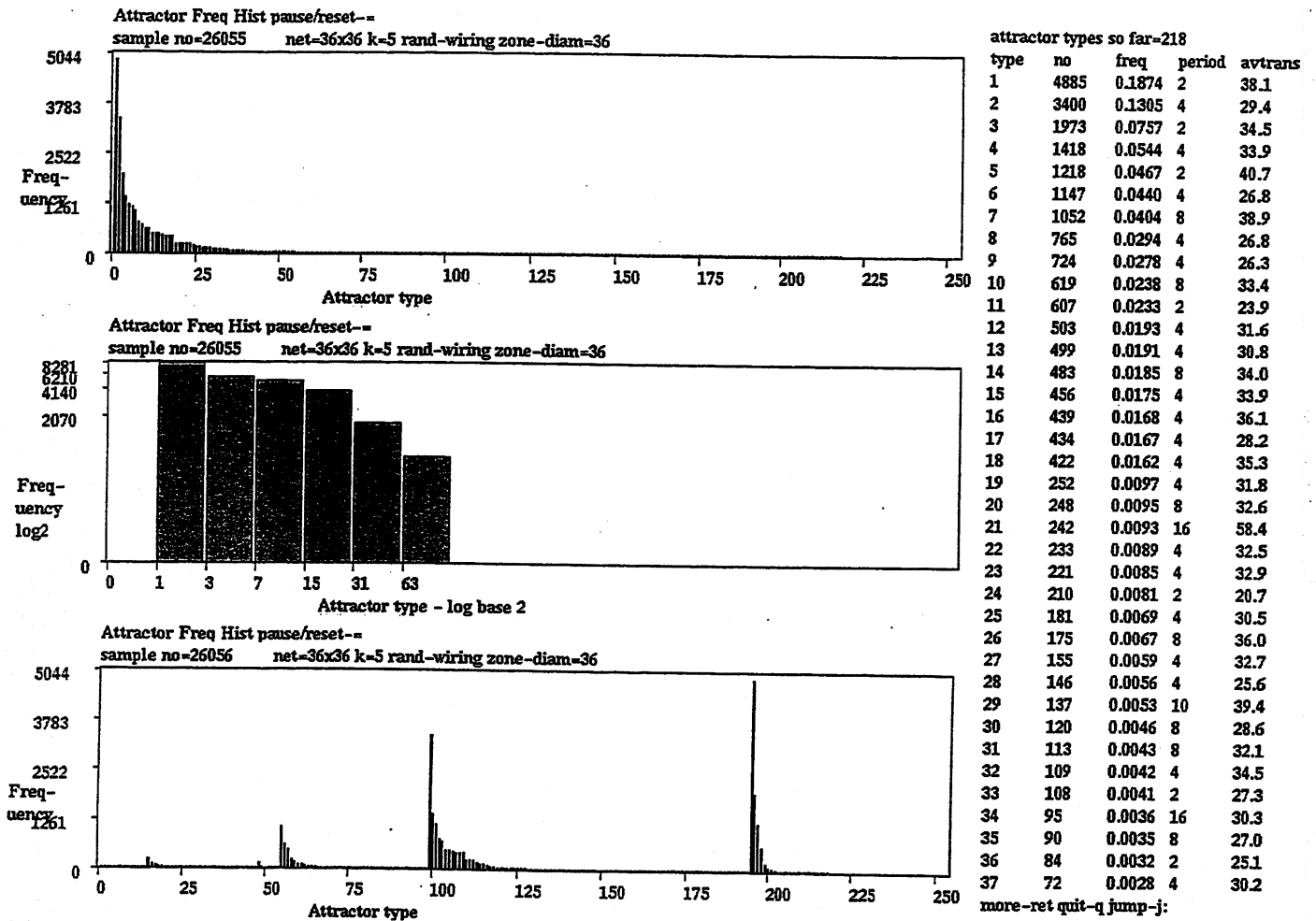
### 5.11.3 Attractor basins.

The attractor basins of RBN can be generated, as described earlier in this chapter, to model particular examples of genomic regulatory networks. Measures on the topology of the attractor basins and subtrees such as their in-degree distribution can be made as for CA (see chapter 4.15).

The precise structure of attractor basins is also of interest as it reflects the stability of cell types to perturbation, both to the current state of the network and to the network parameters themselves. Figure 5.18 shows the consequences of mutating a single bit in the rule table of just one element of a small RBN,  $n=6$ ,  $k=3$ . The basin of attraction fields are affected to varying degrees. Small networks were chosen so that the pattern at each node can be more clearly shown. Larger networks are affected in analogous ways, though the consequences of a one bit mutation are generally smaller with increasing network size. However a particular one bit mutation may cause drastic consequences whatever the size, such as breaking an attractor cycle. Note that the consequences of moving a connection (wire) is usually greater than a one bit mutation.

Portraying the precise structure of attractor basins also allows methods of extracting genetic network architecture from biological data (Somogyi *et al* 1997), an application of the *inverse problem* (see chapter 5.9.3).

Data on attractor basins can also be assembled for large networks by statistics on forward dynamics. The number and size of the different basins of attraction in the basin of attraction field can be inferred from a histogram of the frequency of reaching different attractor types from many random initial states. Other data such as attractor length and average run-in length are also available. An example is given in figure 5.19, which shows a histogram of attractor types against the frequency of arriving at each type from a large sample of initial states, sorted by frequency and also by attractor size. The RBN is 2d  $36 \times 36$  ( $n=1296$ ,  $k=5$ ). The wiring is fully random (not confined within a local radius, see section 5.11.9 below) and the rules scheme is randomly biased to give a fraction of canalising inputs of 52%. This tunes the network to achieve a balance between order and chaos as measured by the Derrida plot (see section 5.11.5 below).



**Figure 5.19.**

Data on attractor basins for a 2d RBN  $36 \times 36$  ( $n=1296$ ,  $k=5$ ) with fully random wiring and a fraction of canalising inputs  $C=52\%$ . The data was assembled by running the network forward from a sample of about 26,000 random initial states and recording the different attractors reached, which indicates the size of each basin of attraction, the number of states it contains.

*Top:* A histogram of attractor types against the frequency of each type sorted by frequency.

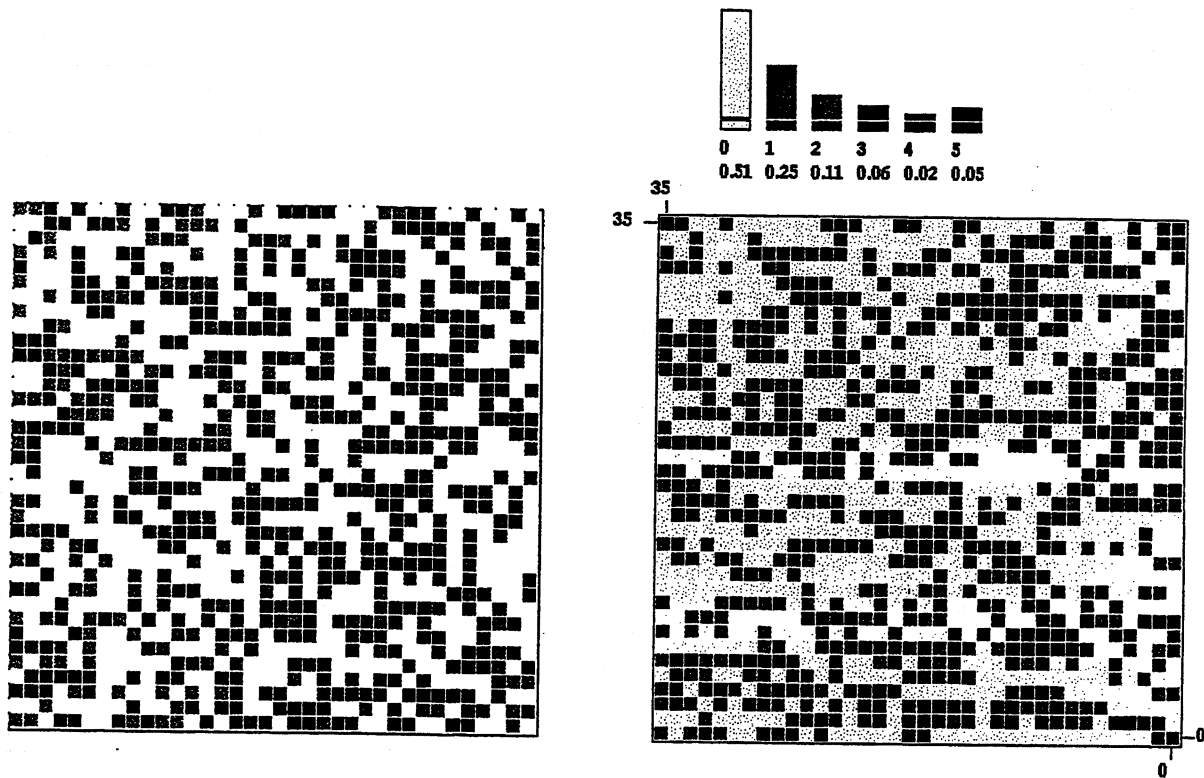
*Centre:* A log-log plot of the above.

*Right:* Data on the 37 most frequent attractors, including the average run-in (transient) length.

*Bottom:* The histogram sorted by attractor period (then by frequency for each period).

#### 5.11.4. Frozen Regions and Frozen Skeletons.

Genes that have stabilised or "frozen" as either 0 or 1 (that have remained unchanged for a preset number of previous time-steps) can be highlighted in the network. Measures are made of the fraction of "frozen" active and inactive genes, where the remainder are dynamic and continue to flicker, analogous to frustration in spin glasses. Figure 5.20 gives an examples for the fraction of canalising



**Figure 5.20.**

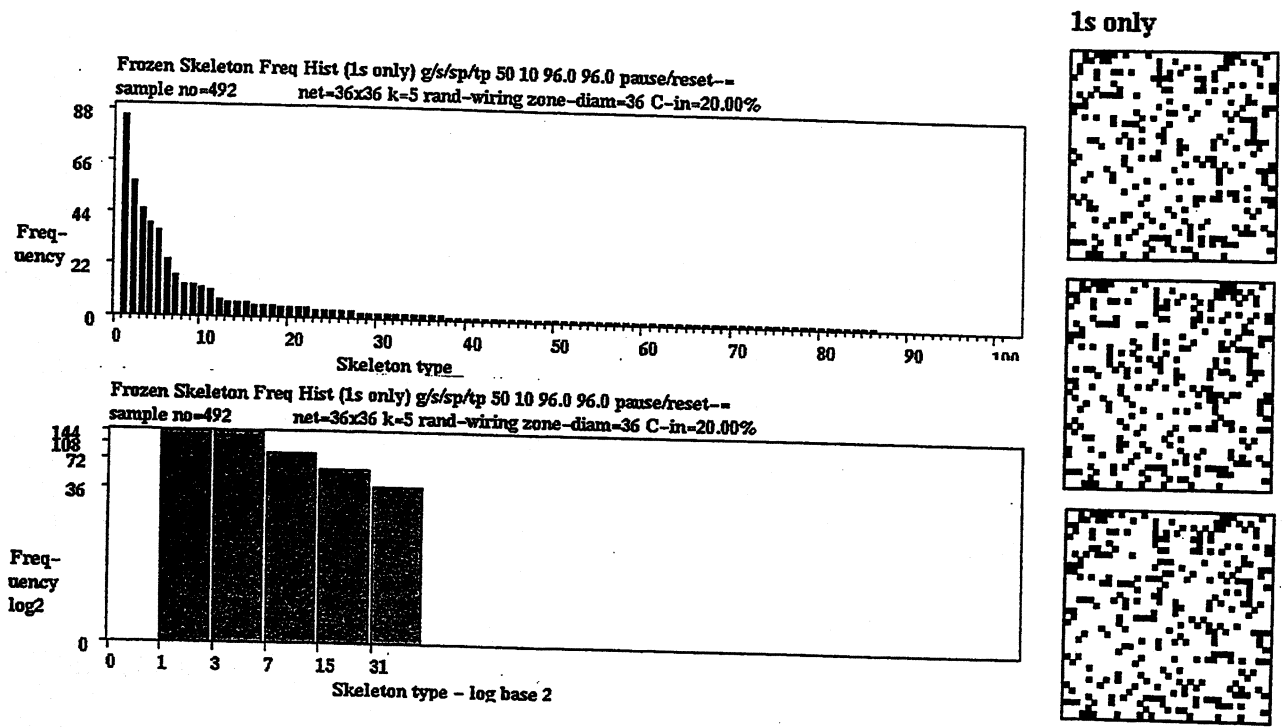
*Left:* Frozen regions in a  $36 \times 36$  RBN ( $n=1296$ ,  $k=5$ ) with fully random wiring. Frozen 1s are black, frozen 0s are grey, dynamic elements are white. To qualify as frozen a gene remained unchanged for 50 time steps.

*Right:* The pattern of canalising inputs across the network (see section 5.11.11). The overall fraction of canalising inputs,  $C=20\%$ . *Top right:* The fraction of different degrees of canalisation within  $C$ , from 0-5.

inputs,  $C=20\%$ . Further examples are given in figure 5.27 for  $C=10\%$ ,  $30\%$  and  $52\%$ . Order is indicated by the percolation or dominance of frozen regions, chaos by the percolation of dynamic regions. The behaviour is then related to network parameters where the frozen genes are interpreted as genes that are active and inactive in cell types.

Different characteristic frozen patterns are named the "frozen skeletons\*" of the network. Data on the skeleton types can be assembled in an analogous way to data on attractors described in section 5.11.3 above. These data are appropriate for networks where the run in length to reach the attractor would be prohibitively long, for example in very large networks or for networks in the chaotic regime, making the collection of data on attractors impractical. The fraction of states in state-space captured by different skeleton types can be inferred from the frequency of reaching the different skeleton types from many random initial states.

\* In a loose sense the "frozen skeleton" is analogous to the "topological skeleton" (Gutowitz Domain 1995) as applied to glider dynamics in CA, see chapter 4.7.



**Figure 5.21.**

Data on skeleton types of frozen 1s for a 2d RBN 36x36 ( $n=1296$ ,  $k=5$ ) with fully random wiring and a fraction of canalising inputs  $C=20\%$ .

*Top:* A histogram of skeleton types against the frequency of each type sorted by frequency.

*Bottom:* A log-log plot of the above.

*Right:* Typical skeleton types of frozen 1s.

The data was assembled by running the network forward from a sample of 492 random initial states and recording the different skeletons reached, which indicates the fraction of state-space captured by each skeleton type. The skeleton parameters (defined below) were as follows:

1: 1s only, 2: 50 time-steps, 3: 10 time-steps, 4: 96%, 5: 96%.

98 98

A number of parameters need to be set to define a skeleton type, as follows:

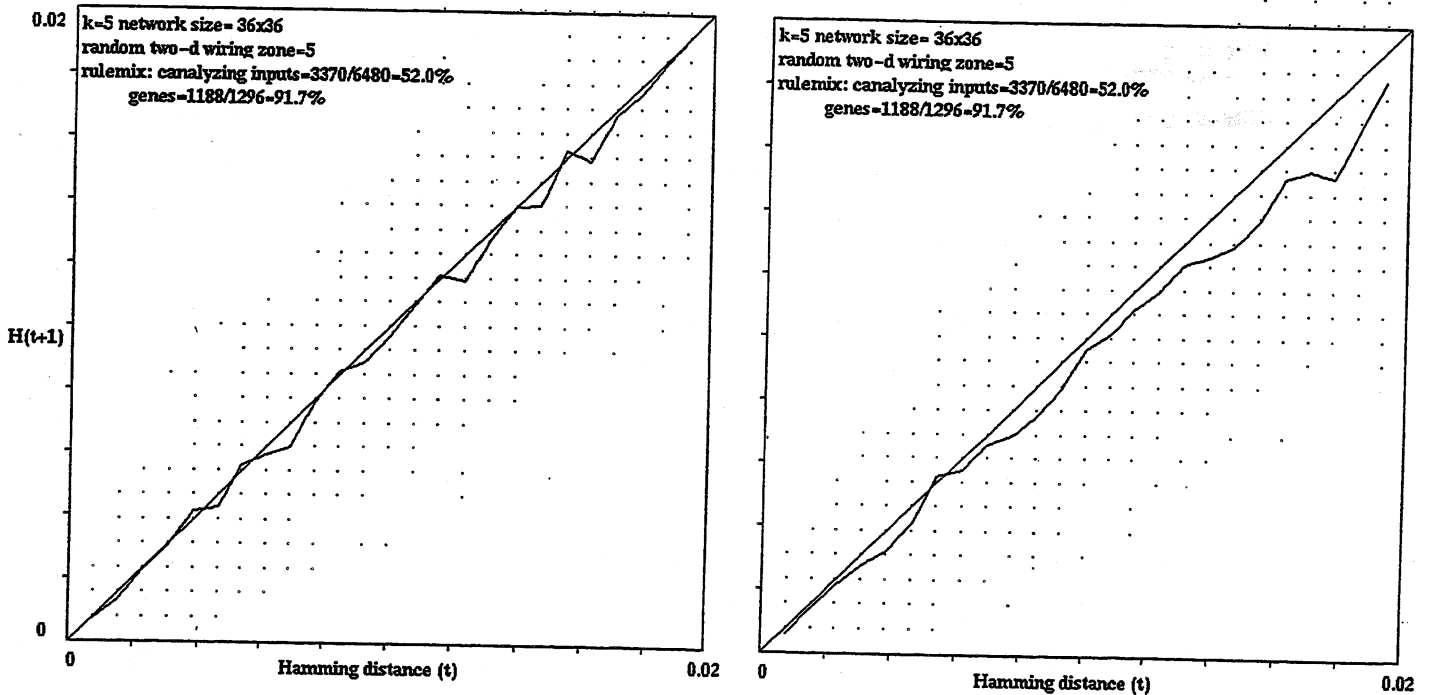
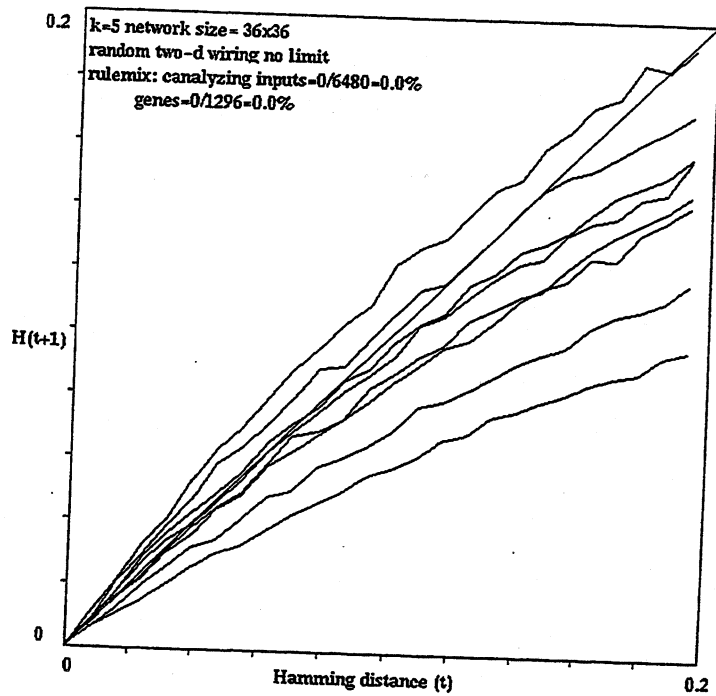
1. The frozen pattern under consideration, 0s only, 1s only, or both 0s and 1s.
2. The number of time-steps that a gene must remain unchanged to be considered frozen.
3. The number of time-steps the *pattern* of frozen genes must remain unchanged (within a set fractional Hamming distance) to be considered a frozen skeleton type.
4. The skeleton fractional Hamming distance limits in (3) above.
5. How close (within a fractional Hamming distance) must a skeleton type be to a pre-established skeleton type to qualify as that type.

An example is given in figure 5.21, which shows a histogram of skeleton types against the frequency of arriving at each type from a sample of random initial states, sorted by frequency, together with some typical skeleton types of frozen 1s.

**Figure 5.22.**

A set of Derrida plots for a 2d RBN  $36 \times 36$  ( $n=1296$ ,  $k=5$ ) with fully random wiring and a range of canalising input fractions  $C = 0\%$ ,  $10\%$ ,  $20\%$ ,  $30\%$ ,  $40\%$ ,  $52\%$ ,  $60\%$  and  $70\%$ .

The slopes of the Derrida plots decrease accordingly. The  $x=y$  diagonal is shown as a reference. The plots were made for Hamming distance interval=10 and sample size=25.



**Figure 5.23.**

*Left:* The Derrida plot for the same network as in figure 5.22 for  $C=52\%$ , shown in greater detail. The plot hugs the  $x=y$  diagonal, indicating a balanced between order and chaos. The Hamming distance interval=1, sample size=50 and the spread of the sample values are shown.

*Right:* The random wiring has been confined within a  $5 \times 5$  block centred on each element, otherwise the network parameters are the same as above. The plot shifts into the ordered regime below the  $x=y$  diagonal, indicating that the network's wiring scheme plays a significant role in behaviour, as well as its rule scheme.

Data on attractor basins and skeletons indicates what combination of network parameters might correspond to biologically plausible genomic regulatory networks, for example in terms of the numbers of attractors (or skeletons) in relation to the number of cell types in an organism.

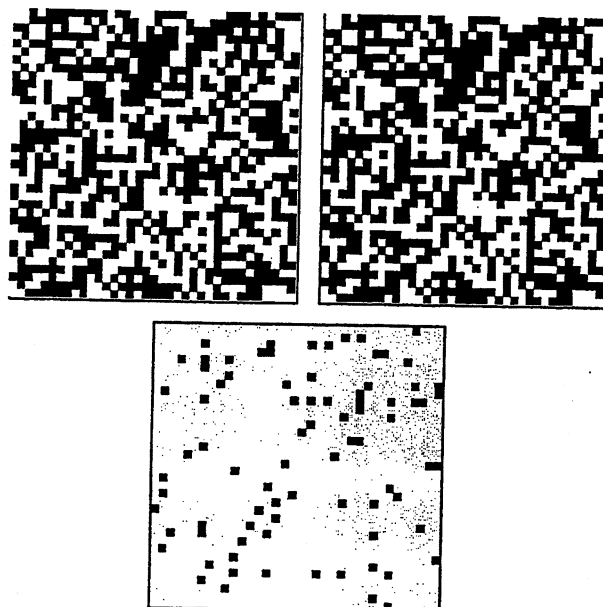
#### 5.11.5. The Derrida plot

The Derrida plot (Derrida and Stauffer 1986), is analogous to the Liapunov exponent in continuous dynamics and measures the divergence of trajectories based on Hamming distance (the number of bits that differ between two patterns).

Pairs of randomly chosen states, but separated by a given Hamming distance are independently iterated forward, usually by one step. The Hamming distance between their respective successors is plotted against the start Hamming distance. This is done for a sample of initial pairs of states and the average taken for the final plot. The procedure is repeated for increasing initial Hamming distance, where the intervals may be one or some greater interval. The chaotic regime is indicated by an initial curve that climbs above the main diagonal indicating that trajectories diverge. For large initial Hamming distance ( $>$  about 0.5) trajectories will inevitably reconverge. The ordered regime is indicated by an initial curve below the main diagonal where trajectories always converge. The phase transition is indicated by a curve hugging the main diagonal for a considerable interval (0 to about 0.2) before gradually falling below, indicating a balance between divergence and convergence for two similar patterns of gene activation.

The example in figure 5.22 shows a set of Derrida plots for a 2d RBN  $36 \times 36$  ( $n=1296$ ,  $k=5$ ) with fully random wiring, for a range of canalising input fractions,  $C = 0\%$ ,  $10\%$ ,  $20\%$ ,  $30\%$ ,  $40\%$ ,  $52\%$ ,  $60\%$  and  $70\%$ . The slopes of the Derrida plots decrease accordingly. The  $x=y$  diagonal is shown as a reference. The plots were made for Hamming distance intervals of 10 and a sample size of 25.

Figure 5.23 (*Left*) shows the Derrida plot for  $C=52\%$  in greater detail, with Hamming distance intervals of 1 and a sample size of 50. According to the Derrida approach, this network is well balanced between order and chaos as the curve closely hugs the  $x=y$  diagonal. In figure 5.23 (*Right*) the random wiring has been confined within a  $5 \times 5$  block centred on each element, otherwise the network parameters are the same. This causes the plot to shift slightly into the ordered regime below the  $x=y$  diagonal, indicating that the network's wiring scheme plays a significant role in behaviour, as well as its rule scheme.



**Figure 5.24.**

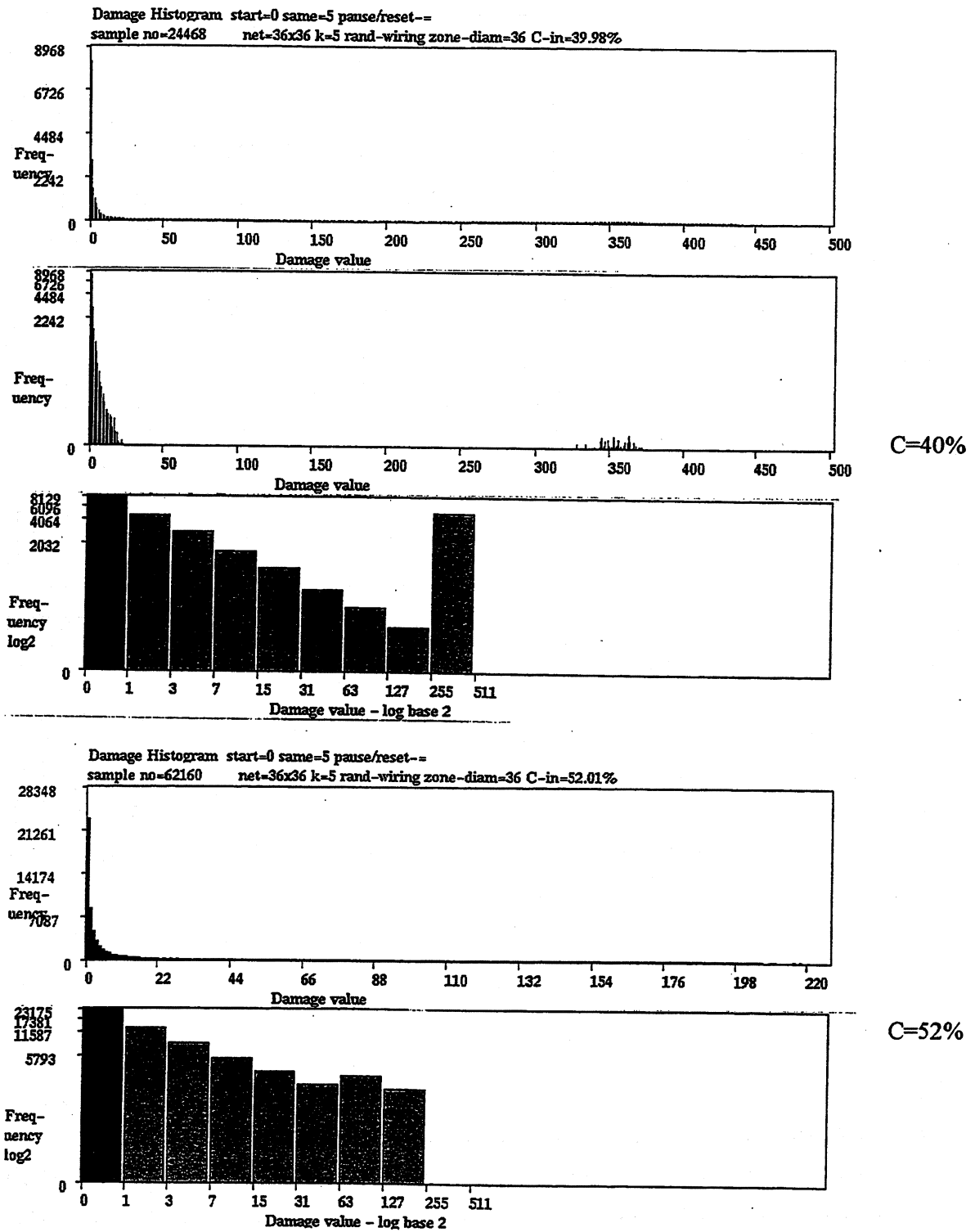
Damage spread resulting from a single bit change in a random pattern. The evolution of the pattern with the changed bit is compared with the evolution of the unchanged pattern (top two panels), and the cumulative difference between the two patterns is monitored (lower panel), i.e. damage is not repaired after the first time-step. The network is a 2d RBN  $36 \times 36$  ( $n=1296$ ,  $k=5$ ) with fully random wiring and a fraction of canalising inputs  $C=52\%$ .

#### 5.11.6. Damage spread

A related measure to the Derrida plot is the distribution of damage spread resulting from a single bit change at a random position in a random pattern of gene activation, for a sample of random patterns. An example is given in figure 5.24.

The evolution of each pattern with the changed bit is compared with the evolution of the unchanged pattern, and the cumulative difference between the two pattern is monitored, i.e. damage is not repaired after the first time-step. When the damage stops spreading (i.e. remains constant for a preset number of time-steps), its size is added to a frequency histogram of damage size distribution. A concentration of low damage spread indicates order, a bi-modal distribution with some low damage and some high indicates chaos. A distribution approximating a power law (but with a finite cut off due to the finite size of the network) indicates a balance between order and chaos.

Figure 5.25 (*top*) shows the damage spread histogram, presented in normal, log-normal (to highlight the cluster of high damage) and log-log form, for a 2d RBN  $36 \times 36$  ( $n=1296$ ,  $k=5$ ) with fully random wiring and a fraction of canalising inputs  $C=40\%$ . The bi-modal distribution indicates the network is in the chaotic regime. The damage was measured when it remained unchanged for 5 time-steps. Figure 5.25 (*bottom*) shows the same information for the same network but where  $C=52\%$ , in a balanced regime between order and chaos according to the Derrida plot in figure 5.23.



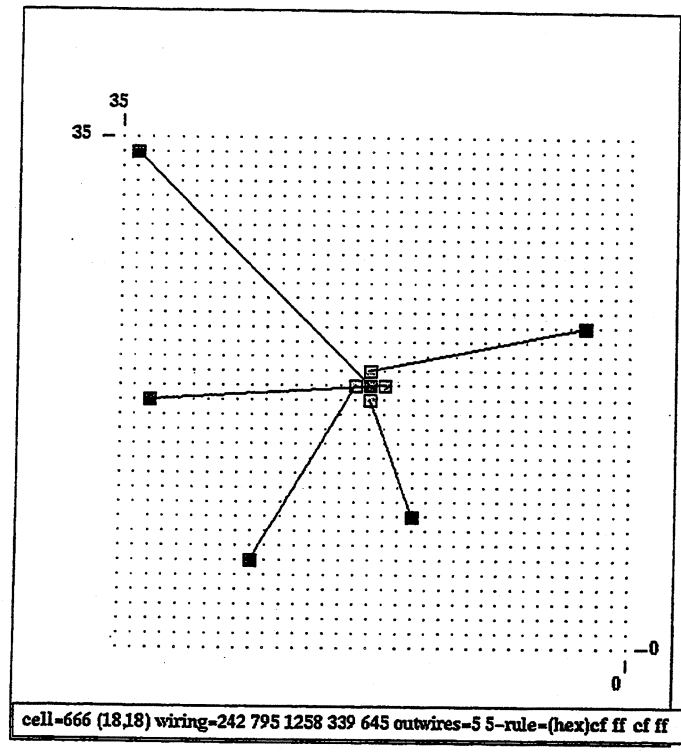
**Figure 5.25.** Damage spread distribution in a 2d RBN 36x36 ( $n=1296$ ,  $k=5$ ) with fully random wiring. The damage was measured when it remained unchanged for 5 time-steps.  
*Top:* C=40%. The damage spread histogram, presented in normal, log-normal and log-log form. The bi-modal distribution indicates chaos.  
*Bottom:* C=52%. The damage spread histogram in normal and log-log form. The network is balanced between order and chaos according to the Derrida plot in figure 5.23.



**Figure 5.26.**

The random wiring of a gene in the standard example 2d RBN,  $36 \times 36$  ( $n=1296$ ,  $k=5$ ) with fully random wiring. The gene's 5 couplings (wires) were assigned randomly to any position in the network, and are shown linking these positions to the gene's pseudo-neighbourhood (see section 5.3).

The gene's wiring could alternatively be assigned randomly within a smaller area centred on the gene.



### 5.11.7. Input entropy for single genes

The input entropy for individual genes (elements) in a network can be measured in a way analogous to the input entropy for the whole network as described in chapter 4.9. The frequency with which each of the neighbourhoods in the rule-table (for a single network element) is "looked up" over a large moving window of time-steps can be represented by a histogram and the entropy calculated. Simulations on various RBN (including the standard example 2d RBN  $36 \times 36$ ,  $k=5$ ) show high single gene entropy for  $C=0$ , becoming consistently lower as  $C$  is increased, giving further indication of order and chaos in the network.

### 5.11.8. RBN parameters as applied to genomic regulatory networks

To produce networks with the various order/chaos and phase transition characteristics and measures described above (5.11.2-5.11.7), the parameters of the network in relation to its wiring and/or rules, must be tuned to the right settings. Various network parameter may be independently or collectively adjusted. Parameter settings relative to rules are the  $P$  parameter, equivalent to the  $\lambda$  parameter and  $\lambda_{\text{ratio}}$  as applied in CA, and the degree of canalisation  $C$ .  $C$  turns out to be the most significant parameter in relation to cell biology. The characteristics of network connections also play an important role, i.e.  $k$  or the  $k$  mix and the biases in the random wiring. These parameters are described below.

### 5.11.9. Network wiring $k$ , and size $n$

The principal wiring parameters are  $k$ , the number of inputs per gene, or the relative mix of  $k$  values in the network, and the connection bias, whether the wiring is fully random, confined within a local radius (as described in chapter 5.1.7) or with some other wiring biases, such as whether or not genes are wired to themselves. The wiring of a particular gene in the standard example 2d RBN,  $36 \times 36$ ,  $k=5$ , with fully random wiring is shown in figure 5.26, where the gene's 5 couplings (wires) were assigned randomly to any position in the network. The genes wiring could alternatively be assigned randomly within a smaller area centred on the gene. Confining random wiring in this way for all genes where the area size is  $5 \times 5$  shifts behaviour towards order as indicated in the Derrida plots in figure 5.23, but further work is required to fully characterise this shift, and the consequences of other wiring biases.

For unbiased rule allocation to the network, where  $k \leq 2$  there is a strong bias towards order. For  $k \geq 3$  there is a bias towards chaos increasing rapidly with larger  $k$  (Kauffman 1993). This occurs because higher  $k$  Boolean functions have an exponentially decreasing fraction of high  $P$  and canalising rules in their rule space. These rule biases play a critical role in driving behaviour towards order, as will be explained below.

The size of the network itself does not seem to significantly alter behaviour according to the measures except for normal statistical scaling effects.

### 5.11.10. The $P$ parameter

The  $P$  parameter is the fraction of 0s or 1s in lookup table, whichever is more. Tuning the average  $P$  value of all the rules in a RBN across its range of  $1 - 0.5$  moves behaviour from order to chaos according to the measures described in 5.11.2-7 above. There are at least\* three equivalent parameters encountered in discrete dynamical networks which measure the same thing in slightly different ways, the proportion of 0s and 1s in the Boolean lookup table on  $k$  inputs ( $B_k$ ).  $\lambda$  distinguishes between 0s and 1s, whereas  $\lambda_{\text{ratio}}$  and the  $P$  parameter treat 0s and 1s equivalently.

$\lambda$ ,  $\lambda_{\text{ratio}}$  and  $P$  are defined and compared below,

$\lambda = c/2^k$  where  $c$  = the count of 1s in a Boolean lookup table on  $k$  inputs ( $B_k$ )

$\lambda$  varies as follows with expected behaviour in brackets      0(order) - 0.5(chaos) - 1(order)

---

\* Another equivalent parameter is *internal homogeneity* introduced earlier by Walker (1966).

$$\lambda_{\text{ratio}} = \frac{2 \times c_{\text{min}}}{2^k} \quad \text{where } c_{\text{min}} = \text{the count of 0s or 1s in } (B_k) \text{ whichever is less}$$

$\lambda_{\text{ratio}}$  varies as follows with expected behaviour in brackets      0(order) - 1(chaos)

$$P = \frac{c_{\text{max}}}{2^k} \quad \text{where } c_{\text{max}} = \text{the count of 0s or 1s in } (B_k) \text{ whichever is more}$$

$P \geq 0.5$  and varies as follows with expected behaviour in brackets      0.5(chaos) - 1(order)

$P$  relates to  $\lambda_{\text{ratio}}$  as follows,      
$$P = 1 - \frac{\lambda_{\text{ratio}}}{2}$$

### 5.11.11. Canalising inputs

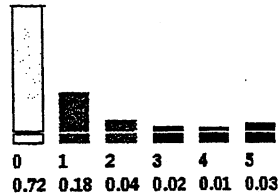
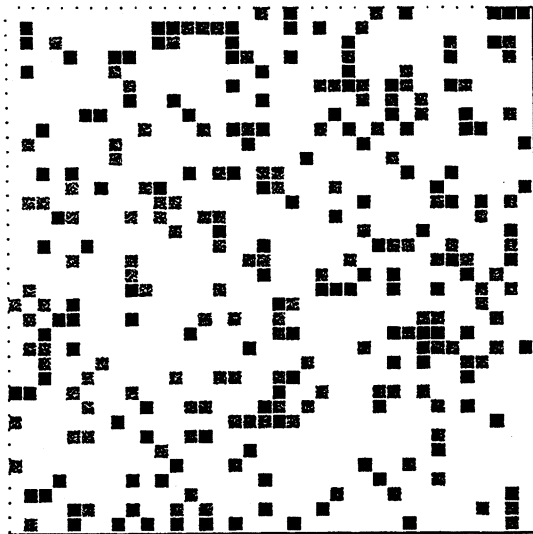
An input connection to an RBN element is "canalising" when one value (0 or 1) on the connection determines the element's state irrespective of its other inputs. The OR rule is an example. A Boolean function or rule on  $k$  inputs can have anywhere from 0 to  $k$  canalising inputs. The occurrence of rules with canalising inputs in rule-space falls off very sharply as  $k$  increases.

The fraction of canalising inputs,  $C$ , across the whole network can be randomly adjusted in DDLab to achieve any arbitrary fraction. This is done by randomly targeting genes and their inputs to raise or lower  $C$  in an unbiased fashion until the required fraction of canalising inputs is realised. For mixed  $k$  networks each subset of elements with a given  $k$  can be tuned separately. Tuning  $C$  moves behaviour between order and chaos according to the measures described in sections 5.11.2-7.

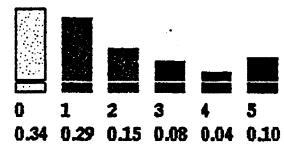
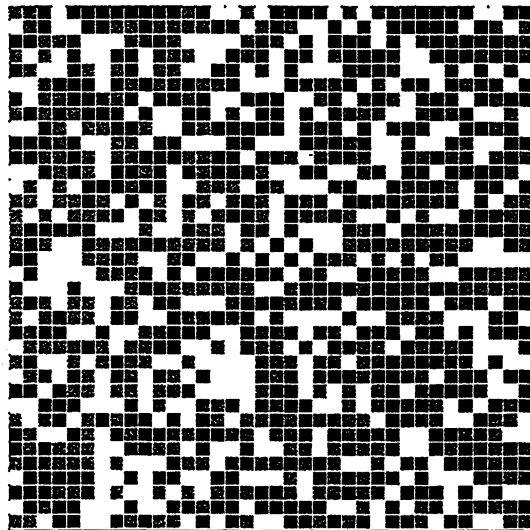
For example, figure 5.20 shows the pattern of canalising inputs across a 2d RBN 36×36 ( $n=1296$ ,  $k=5$ ) with fully random wiring where  $C=20\%$ , and gives an example of the frozen regions that emerge. Figure 5.27 shows a further sequence of typical frozen regions for the same network with  $C=10\%$ , 30%, and 52%.

### 5.11.12. Preliminary results

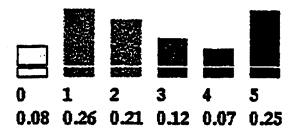
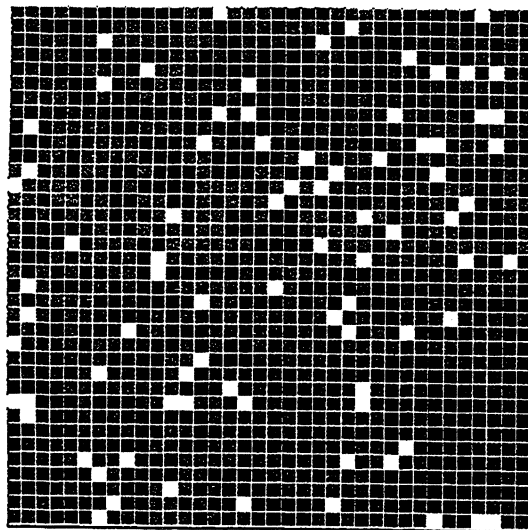
Data is available (produced by DDLab) for the distribution of various  $P$  and canalising values for rules in rule-space for a range of  $k$ , and how these distributions relate to each other. This can be compared with the distribution of  $P$  and canalising values in transcription systems, representing real genomic regulatory networks, from data assembled by Steve Harris. The comparison shows that the real genomic networks exhibit a strong bias towards a high degree of canalising in their Boolean functions. A careful argument distinguished this bias from a possible bias towards high  $P$ , which proved to be less significant.



C=10%



C=30%



C=52%

**Figure 5.27.** Frozen regions in a  $36 \times 36$  RBN ( $n=1296$ ,  $k=5$ ) with fully random wiring for a series of networks,  $C=10\%$ ,  $30\%$  and  $52\%$  from the top. Frozen 1s are black, frozen 0s are grey, dynamic elements are white. To qualify as frozen a gene remained unchanged for 50 time steps. The fraction of different degrees of canalisation within  $C$ , from 0-5 is shown. See also figure 5.20.

The table below sets out the fractions of canalising inputs from real data on transcription systems (sample size shown), compared to the probability of finding canalising inputs when Boolean functions are randomly picked from rule-space, for  $k=3, 4$  and  $5$ .

	<u>sample size</u>	<u>real</u>	<u>random</u>
$k=3$	51	0.78	0.23
$k=4$	35	0.68	0.034
$k=5$	11	0.54	0.0031

The fractions of canalising inputs from the real transcription data was used to simulate large RBN using DDLab. According to the various measures listed in sections 5.11.2 - 5.11.7, the dynamics of the networks turned out to be just in the ordered regime but very close to the order/chaos phase transition.

These and more recent results will be published in (Harris *et al* 1997).

## 5.12. Memory in RBN - Conclusion

Whereas CA, embedded as they are in a regular space with homogeneous laws, provide models of processes in physics, including the emergence of self-organisation and simple life-like phenomena, RBN may be appropriate as models for the *non-local* interactions between living sub-systems typical of biology. Non-local interactions transcend inanimate physics and are perhaps a key property distinguishing living from non-living matter. Complex non-local networks with emergent properties occur hierarchically at all scales in biology, from networks of genes regulating one another within a cell, to neural networks in the brain, to networks regulating entire ecosystems.

Non-local interactions, shaped by evolution, development and learning, allow the emergence of unconstrained *categorisation* in a system's repertoire of dynamical behaviour - its content addressable memory. Understanding the dynamics of abstract random networks may provide insights into biological networks in general, where concepts analogous to memory may be ubiquitous, take for instance memory in the immune system, or the cell type as a kind of memory in genomic regulatory networks.

The emergence of novel structures by the interaction of many low level components underlies complex adaptive systems, ultimately the emergence of life by the self-organisation of matter. Can this approach extend to cognition and consciousness, and if so what are the low level components from which cognition emerges? I have suggested that these components are the hierarchical categories implicit in the dynamics of neural sub-networks, and that memory emerges when

sub-networks combine and maintain each other far from equilibrium. Memory may in turn provide the substrate for unconstrained emergence producing higher level cognitive states.

The attractor basin diagrams of random Boolean networks capture the network's memory. The diagrams demonstrate that a complex hierarchy of categorisation exists within transient trees, far from equilibrium, providing a vastly richer substrate for memory than attractors alone. In the context of many semi-autonomous weakly coupled networks, the basin field/network relationship may provide a fruitful metaphor for the mind/brain.

## Chapter 6

### Conclusions and open questions

This thesis has investigated the new perspective on discrete dynamical networks provided by attractor basins, using computer tools developed by the author. The work has continued on from ideas first presented in the authors book "The Global Dynamics of Cellular Automata" (Wuensche and Lesser 1992a), with new methods for reconstructing the attractor basins of random Boolean networks (RBN) in addition to cellular automata (CA). The investigation has moved into new areas where studies of attractor basins combined with studies of local dynamics can shed light on self-organisation and memory.

Attractor basins show how state-space is linked according the network's dynamics, analogous to the "phase portrait" which proved such a powerful concept in continuous dynamical systems theory. General principles underlying the dynamics of discrete networks can be found by comparing the *local* behaviour of the unfolding patterns in local trajectories with the *global* characteristics of attractor basins. Concepts of order, complexity, chaos, and phase transitions in local dynamics can be tied into the degree of convergence found attractor basin topology.

Contrasting notions of emergence and self-organisation between CA and RBN have been highlighted. CA are systems having a regular space and universal laws providing models in physics on the basis of self-organisation within their spatial pattern, whereas RBN provide models in biology because of their non-local connections and heterogeneous rules. The hierarchical categorisation implicit in their attractor basins, unconstrained by the symmetries found in CA, provide insights into memory in neural and genomic networks.

Some preliminary ideas have been presented on forging a link between discrete dynamical networks and random graph theory that may provide a powerful mathematical framework.

New measures and results on attractor basins and how they relate to measures on the local dynamics of CA have been presented, in particular in the context of the complex dynamics supporting gliders, which are of prime interest as an example of self-organisation and emergence in simple systems.

A theory explaining memory based on the attractor basins of RBN has been proposed, demonstrating that memory is not only based on attractors but also on the complex hierarchy of categorisation that exists within transient trees, far from equilibrium. Learning algorithms and a neural model based on RBN have been described. Applications of RBN to model genomic regulatory networks have been described, where cell types are explained as basins of attraction in the dynamics of the network.

Nearly all the work described in this thesis is open ended and points to many uncharted areas providing many avenues for future research. The work relies on a cycle of theory and experiment where the experimental apparatus is computer simulations. The software "Discrete Dynamics Lab" has the ability to perform experiments and simulations that have not yet been fully explored, for example to simulate networks of semi-autonomous CA and RBN. Other users of DDLab from among the growing user community will, it is hoped, initiate their own lines of enquiry. This is already happening, especially in theoretical biology. DDLab continues to be developed and new software capabilities will provide new insights.

Some future avenues of enquiry are as follows:

- Pursue the link between discrete dynamical systems and random graph theory.
- Extend the methods for reconstructing attractor basins, and the  $Z$  parameter, to include networks element with more than binary values.
- Investigate how the statistics on subtrees depend on the location of the subtree root.
- Complete a systematic evaluation of CA rule space with the tools and methods already in place such as the in-degree histogram, in particular on the basis of automatic samples according to input-entropy.
- Relate the methods to artificial neural networks to implement techniques for reconstructing their attractor basins.
- Investigate networks of interconnected sub-networks. Apply these systems as higher order neural and genomic network models, and to models of microbial communities.
- Develop the methods for modelling genomic regulatory networks in response to the requirements of molecular biologists.
- Combine the learning algorithms with the new "inverse problem" algorithms to create applications in areas such as pattern recognition, and also to reverse engineer genomic regulatory networks.
- Systematically investigate of the topology and structure of basins of attraction of random Boolean networks and various intermediate architectures, using the tools now available. This area has only been touched on because it is such a vast space.
- Extend the methods to asynchronous networks, for example networks with discrete time delays.
- Investigate probabilistic and noisy networks, and the consequences of perturbing network parameters.



## References.

- Adamatzki, A., (1994) "*Identification of Cellular Automata*" Francis and Taylor, London.
- Alexander, I., W. Thomas and P. Bowden, (1984) "*WISARD, a radical new step forward in image recognition*", *Sensor Review*, 120-4.
- Aizawa, Y., I. Nishikawa and K. Kaneko, (1990) "Soliton Turbulence in One-Dimensional Cellular Automata", *Physica D* 45, 307-327.
- Amit, D.J. (1989) "*Modeling Brain Function, The world of attractor neural networks*", Cambridge University Press.
- Ashby, W.R., (1956) "*An Introduction to Cybernetics*", Chapman & Hall.
- Ashby, W.R., (1960) "*Design for a Brain*", Chapman & Hall.
- Askenazi, M., (1996) "GeneTool" software, manor@santafe.edu
- Bates, J.E., and H.K. Shepard, (1993) "Measuring complexity using information fluctuations", *Physics Letters A* 172, 416-425.
- Bennet, C.H., (1986) "On the Nature and Origin of Complexity in Discrete, Homogeneous, Locally-Interacting Systems", *Foundations of Physics* 16, 585-592.
- Boccaro, N., J. Nasser and M. Roger, (1991) "Particlelike structures and their interactions in spatiotemporal patterns generated by one-dimensional deterministic cellular automata rules", *Physical Review A*, Vol.44, No.2.
- Bollobás, B. (1985) "*Random Graphs*", Academic Press, London.
- Burks, A. W. (1970) "*Essays on Cellular Automata*", Univ. of Illinois Press, Urbana.
- Codd, E.F., (1986) "*Cellular Automata*", Academic Press, New York.
- 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.
- Coste, J., and H. Herron, (1986), in "*Disordered Systems and Biological Organisation*" eds. E. Bienenstock, F. Fogelman-Soulie, and G. Wiesbuch, Series F: *Computer Systems Sciences*, vol. 20, Springer, New York.
- Crutchfield, J.P., and K. Young, (1989) "Inferring statistical complexity", *Phy.Rev.Lett.* 63, 105 .
- Crutchfield, J.P., and J.E. Hanson, (1993) "Turbulent Pattern Bases for Cellular Automata", Santa Fe Institute Technical Report SFI 93-03-010.
- Delamare, T., (1994) "Exploration et visualisation d'espaces du calcul cellulaire", Mémoire de DEA d'intelligence artificielle, Université de Paris-8.
- Derrida, B., and D. Bessis (1988) "Statistical properties of valleys in the annealed random map model", *J.Phys.A.Lett.* 21:L509.
- Derrida, B., and H. Flyvbjerg, (1987) "The Random Map Model: a disordered model with deterministic dynamics", *J.Physique* 48:971.
- Derrida, B., and Y. Pomeau, (1987) "Random Network Automata: A Simplified Annealed Approximation", *Europhys.Lett.* 1, 45.
- Derrida, B., and D. Stauffer, (1986) "Phase transitions in Two-Dimensional Kauffman Cellular Automata Random Network Automata", *Europhys.Lett.* 2, 739.
- Erdos, P., and A. Renyi, (1960) "*On the Evolution of Random Graphs*", Institute of Mathematics, Hungarian Academy of Sciences, publication no.5.
- Frisch, U., B. Hasslacher and Y. Pomeau, (1986) "Lattice gas automata for the Navier-Stokes equation", *Phys.Rev.Lett.*, 56:1505.
- Gelfant, A.E., and C.C. Walker, (1984) "*Ensemble Modeling*", Marcel Dekker, New York.
- Goodwin, B., (1994) "*How the Leopard Changed its Spots; The Evolution of Complexity*", Orion.

- Grassberger,P., (1986) "Towards a Quantitative Theory of Self-Generating Complexity", *Int.J.Theor.Phys.*25, 907.
- Grey C.M, W.P.König, A.K.Engel, W.Singer, (1989) "Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties", *Nature* 388, 334- 337.
- Gutowitz,H.A., (1991) "Transients, cycles, and complexity in cellular automata", *Physical Review A*, Vol.44, No.12.
- Gutowitz,H.A., ed., (1991) "*Cellular Automata, Theory and Experiment*", MIT press.
- Gutowitz,H.A., and C.G.Langton., (1995) "Mean Field Theory of the Edge of Chaos", Proceedings of ECAL.
- Gutowitz,H.A. and C.Domain, (1995) "The topological skeleton of cellular automaton dynamics", to appear in *Physica D*.
- Hameroff,S.R. 1987. "*Ultimate Computing: Biomolecular Consciousness and NanoTechnology*", North Holland, Amsterdam.
- Hameroff,S., S.Rasmussen and B.Mansson, (1989) "Molecular Automata in Microtubules: Basic Computational Logic of the Living State". in *Artificial Life*, C.Langton (ed) Santa Fe Institute Studies in the Sciences of Complexity, Proceedings Vol. VI, Addison-Wesley 521-553.
- Hameroff,S.R., J.E.Dayhoff, R.Lahoz-Beltra, S.Rasmussen, E.M.Insinna and D. Koruga (1993) "Nanoneurology and the Cytoskeleton: Quantum Signaling and Protein Conformational Dynamics as Cognitive Substrate", in *Behavioral Neurodynamics*, K.Pribram and H Szu, eds., Pergamon Press
- Hanson,J.E., and J.P.Cruchfield (1995) "Computational Mechanics of Cellular Automata: An Example", Santa Fe Institute Working Paper 95-10-095.
- Harris,S., A.Wuensche, S.Kauffman and B.Sawhill, (1997, paper in progress, provisional title) "Are Eukaryotic Cells at the Edge of Chaos".
- Harvey,I., P.Husbands, D.Cliff, (1993) "Issues in Evolutionary Robotics", in *From Animals to Animals 2*, Proceedings of the 2nd International Conf. on Simulation of Adaptive Behaviour, ed. Mayer, Roitblat and Wilson, MIT Press, 364-373.
- Hasslacher, B., (1987) "Discrete Fluids, Part 1, Background for Lattice Gas Automata", Los Alamos Science Special Issue.
- Hopfield,J.J., (1982) "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences 79:2554-2558.
- Hopfield,J.J., (1995) "Pattern recognition computation using action potential timing for stimulus representation", *Nature*, 376:33-36.
- Kauffman,S.A., (1969) "Metabolic stability and epigenesis in randomly constructed genetic nets", *Journal of Theoretical Biology*, 22, 437-467.
- Kauffman,S.A., (1971) "Gene regulation networks: A theory for their global structure and behaviour". *Current Topics in Dev. Biol.* 6:145.
- Kauffman,S.A., (1984) "Emergent properties in random complex systems", *Physica 10D*, 146-156.
- Kauffman,S.A., (1991) "Requirements of evolvability in complex systems; Orderly dynamics and frozen components", in *Complexity, Entropy and the physics of information*, ed. W.H.Zurek. Santa Fe Institute Studies in the Sciences of Complexity, Proceedings Vol.VIII, Addison-Wesley, 151-192.
- Kauffman,S.A., (1993) "*The Origins of Order, Self-Organization and Selection in Evolution*" Oxford University Press.
- Langton C.G., (1984) "Self-Reproduction in Cellular Automata", *Physica 10D*, 135-144.

- Langton, C.G., (1986) "Studying Artificial Life with Cellular Automata", *Physica* 22D, 120-149.
- Langton, C.G., eds. (1989) "Artificial Life", Santa Fe Institute Studies in the Sciences of Complexity, Proceedings Vol Vol.X, Addison-Wesley
- Langton, C.G., (1990) "Computation at the Edge of Chaos: Phase Transitions and Emergent Computation", *Physica D* 42, 12-37.
- Langton, C.G., C.Taylor, J.D.Farmer, S.Rasmussen, eds. (1991) *Artificial Life II*, proceedings vol.X, Santa Fe Institute Studies in the Sciences of Complexity, Proceedings Vol Vol.X, Addison-Wesley
- Langton, C.G., (1991) "Life at the Edge of Chaos", in *Artificial Life II* eds C.Langton *et al*, Santa Fe Institute Studies in the Sciences of Complexity, Proceedings Vol Vol.X, Addison-Wesley
- Lewis, E.J, and Glass, L., (1992) "Non-Linear Dynamics and Symbolic Dynamics of Neural Networks", *Neural Computation* 4, 621-642.
- Li, W., (1989) "Complex Patterns Generated by Next Nearest Neighbors Cellular Automata". *Comput. & Graphics* Vol.13, No.4, 531-537.
- Li, W., (1990a) "Mutual information functions versus correlation functions", *Journal of Statistical Physics* 60, 823-837.
- Li, W., (1990b) "Phenomenology of Non-Local Cellular Automata", Santa Fe Institute Working Paper 91- 01-001.
- Li, W., N.H.Packard and C.G.Langton, (1990) "Transition Phenomena in Cellular Automata Rulespace", *Physica D* 45, 77-94.
- Li, W., and N.H.Packard, (1990) "The Structure of Elementary Cellular Automata Rule Space", *Complex Systems* 4, 281-297.
- Li, W., and M.G.Nordahl, (1992) "Transient Behaviour of Cellular Automata Rule 110". Santa Fe Institute Working Paper 92-03-016.
- Lindgren, K., and M.G.Nordahl, (1990) "Universal Computation in a Simple One-Dimensional Cellular Automata", *Complex Systems* 4, 299-318.
- Lindgren, K., and M.G.Nordahl, (1988) *Complex Systems* 2, 409.
- McCormick, D.A., (1990) "Membrane properties and neurotransmitter actions", in "The Synaptic Organization of the Brain", Shepherd, G.M., ed 3rd edition, Oxford University Press, 32-66.
- Mange, D., (1992) "Microprogrammed Systems, An introduction to firmware theory" Chapman Hall.
- Martin, O., A.M.Odlyzko, and S.Wolfram, (1984), "Algebraic Properties of Cellular Automata", *Comm. Math. Phys.* 93, 219-258.
- McIntosh, H.V., (1990) "Wolfram's Class IV Automata and a Good Life", *Physica D* 45, 105-121.
- McIntosh, H.V., (1993) "Ancestors: Commentries on The Global Dynamics of Cellular Automata by Andrew Wuensche and Mike Lesser", Departamento de Aplicacion de Microcomputadoras, Instituto de Ciencias, Universidad de Autonoma de Puebla.
- Moore, E. (1962) *Pros.Symp.Appl.Math.* 14 17-33.
- Myers, J.E., (1997) "Random Boolean Networks - Three Recent Results", to be appear in *Complexity*.
- von Neuman, J., (1966). "Theory of Self Reproducing Automata", edited and completed by A.W.Burks, University of Illinois Press.
- Pitsianis, S., G.L.Bleris, Ph.Tsalides, A.Thanailakis, H.C.Card, (1989) "Algebraic theory of bounded one-dimensional cellular automata", *Complex Systems* 3, 209-227.
- Prigogine, I., and I.Stengers, (1984) "Order out of Chaos", Heinemann.

- Reidys,C., (1996), "Mapping in random structures", to appear in SIAM Journal of Discrete Mathematics.
- Reidys,C., and A.Wuensche, (1997, paper in progress, provisional title) "Attractor Basins of Random Directed Maps".
- Shepherd,G.M., ed, (1990) "*The Synaptic Organization of the Brain*", 3rd edition, Oxford University Press.
- Sherrington, D., (1990) "Complexity Due to Disorder and Frustration", in *1989 Lectures on Complex Systems*, E.Jen (ed.), Addison-Wesley.
- Singer,W., (1991) "Response synchronization of cortical neurons; An epiphenomenon or a solution to the binding problem?" IBRO News 19,6-7.
- Singer W., and Grey C.M., (1995) "Visual feature integration and the temporal correlation hypothesis", *Annu Rev Neurosci* 18, 555-586.
- Smith III, A.R., (1971) "Simple Computation -Universal Cellular Spaces", *JACM*, vol 2, no. 3, 339-353.
- Somogyi,R., and C.Sniegoski, (1996) "Modeling the Complexity of Genetic Networks: Understanding Multigenetic and Pleiotropic Regulation", *Complexity*, Vol.1/No.6,45-63.
- Somogyi,R, S.Fuhrman, M.Askenzi, A.Wuensche., (1997) "The Gene Expression Matrix:Towards the Extraction of Genetic Networks Architectures" in *Proceedings of the World Congress of Non-Linear Analysts*, in press.
- Toffoli,T and N.Margolus, (1987) "*Cellular Automata Machines, A new environment for modeling*", MIT Press.
- Vichniac,G.Y., (1990) "Boolean derivatives on Cellular Automata", *Physica D* 45, 63-74.
- Walker,C.C., and W.R.Ashby, (1966) "On the temporal characteristics of behavior in certain complex systems", *Kybernetik* 3, 100-108
- Walker,C.C., (1971) "Behavior of a class of complex systems: the effect of system size on properties of terminal cycle", *Cybernetics*, 55-67
- Walker,C.C., and A.A.Aadryan., (1971) "Amount of computation preceding externally detectable steady-state behavior in a class of complex systems", *Bio-Medical Computing*, vol.2, 85-94
- Walker,C.C., (1987) "Stability of equilibrium states and limit cycles in sparsely connected, structurally complex Boolean nets", *Complex Systems* 1, 1063-1086
- Wasserman,G.S.,(1992) "Isomorphism, Task Dependence, and the Multiple Meaning Theory of Neural Coding", *Biological Signals*.
- Weisbuch,G., R.DeBoer and A.S.Perelson, (1990) "Localized memories in idiotypic networks". *J.Theoret. Biol.* 146,483-499.
- Wolfram,S., (1983) "Statistical mechanics of cellular automata", *Reviews of Modern Physics*. vol 55, no.3, 601-64.
- Wolfram,S., (1984a) "Universality and complexity in cellular automata", *Physica D*, vol 10D, 1-35.
- Wolfram,S., (1984b) "Computational Theory of Cellular Automata", *Commun.Maths.Phys.*96, 15-57.
- Wolfram,S., (1985) "Twenty problems in the theory of cellular automata", *Physica Scripta*. Vol.T9, 170-183.
- Wolfram,S., ed. (1986a) "*Theory and Application of Cellular Automata*", World Scientific.
- Wolfram,S., ed. (1986b) "Random sequence generation by cellular automata", *Adv. Applied Math.* 7, 123-169.42.

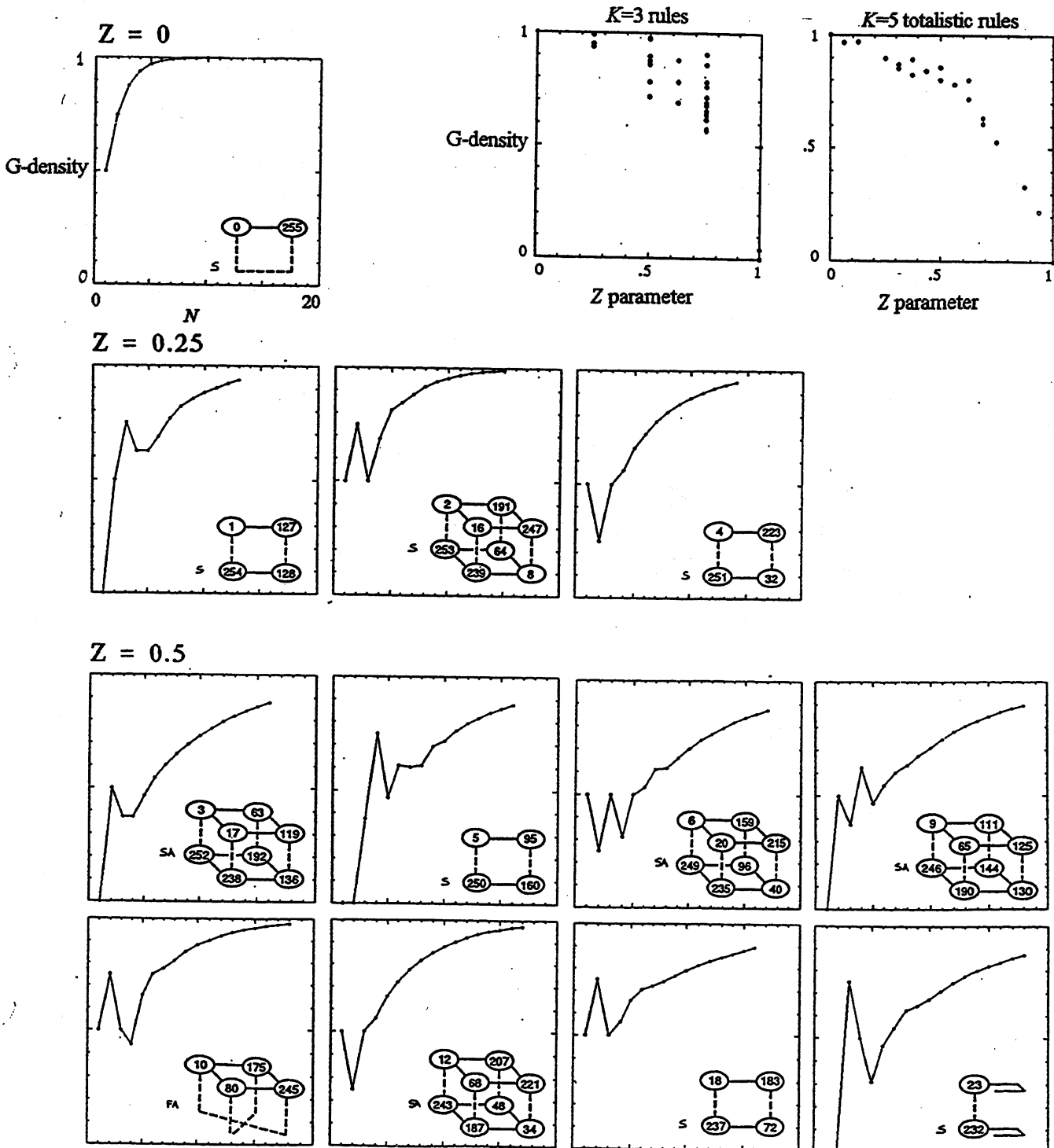
- Wolfram,S. (1986c) "Approaches to Complexity Engineering", in *Theory and Application of Cellular Automata*, S.Wolfram, ed. World Scientific, 400-413.
- Wolfram,S and H.Peck, (1986) *Table 13*, "State transition diagrams for cellular automata on finite size lattices", in *Theory and Application of Cellular Automata*, S.Wolfram, ed. World Scientific, 531-536.
- Wooters,W.K., and C.Langton, (1990) "Is there a sharp Phase transition for deterministic cellular automata?", *Physica D* 45, 95-104.
- Wuensche,A., and M.J.Lesser. (1992a) "*The Global Dynamics of Cellular Automata; An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*", Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.
- Wuensche,A. (1992b) "The Ghost in the Machine; Basin of Attraction Fields of Disordered Cellular Automata Networks", Santa Fe Institute Working Paper 92-04- 017.
- Wuensche,A. (1992c) "Basins of Attraction in Disordered Networks", in *Artificial Neural Networks*, 2, eds. I.Alexander and J.Taylor, Elsevier.
- Wuensche,A. (1993a) "The Ghost in the Machine; Basin of Attraction Fields of Random Boolean Networks", CSRP 281, University of Sussex, published in *Artificial Life III* (1994), ed C.G.Langton, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.
- Wuensche,A (1993b)., "Visible Learning; Sculpting the Basin of Attraction Fields of Random Boolean Networks", in *Computing with Logical Neurons*, ed E.M.Allison, Dept. of Electronics, University of York.
- Wuensche,A. (1993c), "Memory Far from Equilibrium", in *Proceedings of the Second European Conference on Artificial Life*, eds J.L.Deneubourg, S,Goss, G,Nicolis, H.Bersini, R.Dagonnier, Centre for Non-Linear Phenomena and Complex Systems, Université Libre de Bruxelles.
- Wuensche,A., (1994a) "Complexity in One-D Cellular Automata; Gliders, Basins of Attraction and the Z Parameter", Cognitive Science Research Paper 321, Univ. of Sussex.
- Wuensche,A., (1994b) "The Emergence of Memory", Cognitive Science Research Paper 346, Univ. of Sussex, published in *Towards a Science of Consciousness*, (1996), eds. S.R.Hameroff, A.W.Kaszniak, A.C.Scott, MIT Press.
- Wuensche,A.,(1996) "Discrete Dynamics Lab (DDLab)" , interactive graphics software for investigating discrete dynamical networks including their attractor basins. The software and documentation is published at *MIT Press Alife Online*, at the Santa fe Institute.
- Download instructions:  
 DDLab home page: <http://www.santafe.edu/~wuensch/ddlab.html>,  
 MIT Press Alife Online: <http://alife.santafe.edu/alife/software/ddlab.html>  
 ftp: <ftp://alife.santafe.edu/pub/SOFTWARE/ddlab>
- Zurek,W.H., ed. (1990) "*Complexity, Entropy and the Physics of Information*", ed. W.H.Zurek, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.
- Zwick,M., and H.Shu, (1995) "Set-Theoretic Reconstructability of Elementary Cellular Automata", *Advances in Systems Science and Applications*, Special Issue I, 31-36.

### Appendix 1 - Garden-of-Eden density/ System size graphs

#### K=3 rules. (K=5 totalistic rules, page 1.4)

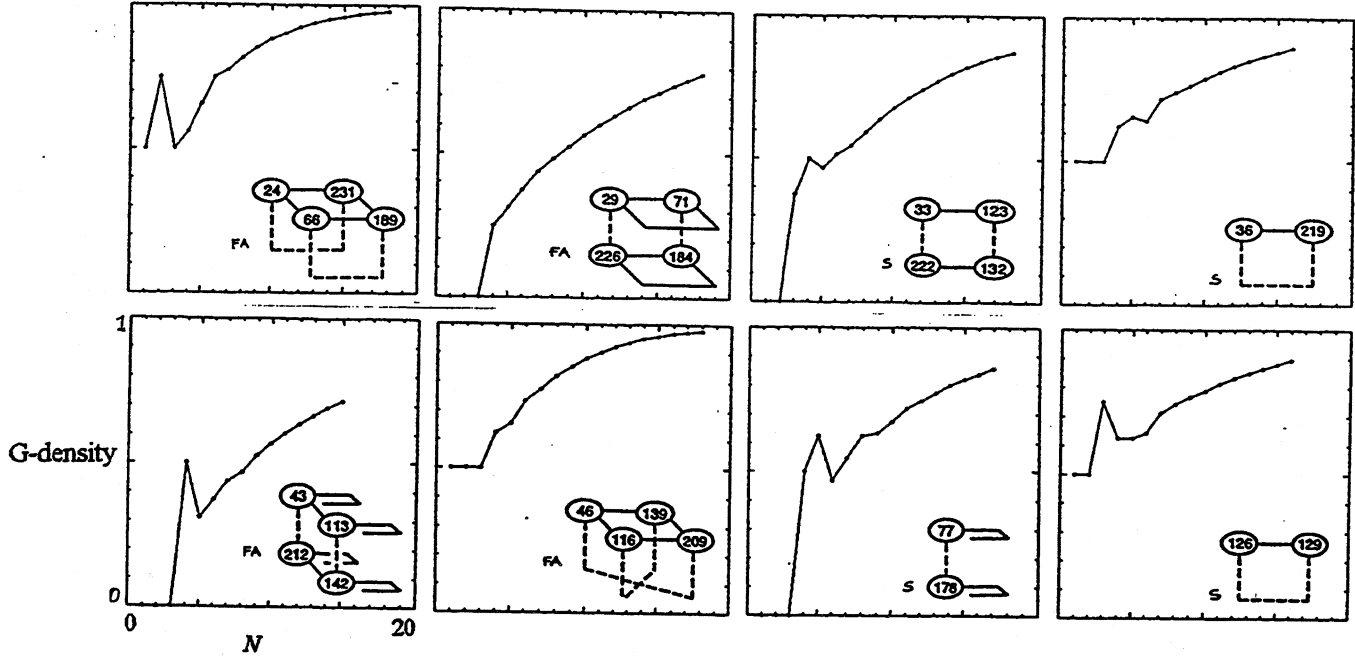
The garden of Eden density (G-density) is plotted against system size,  $N$ , as  $N$  is increased from 1 to 18. The rules in a rule cluster<sup>21</sup> have equivalent G-density, so the graphs have therefore been plotted for 48 K3 rule clusters. The clusters are ordered by the  $Z$  parameter (which is indicated) and secondly by the lowest rule number, in the top left hand corner of each cluster. The plots are based on the complete basin of attraction field for each value of  $N$ .

Below right are graphs of G-density against the  $Z$  parameter for all K3 rules and K5 totalistic rules for  $N=18$ . The rules in a rule clusters have equivalent  $Z$ .

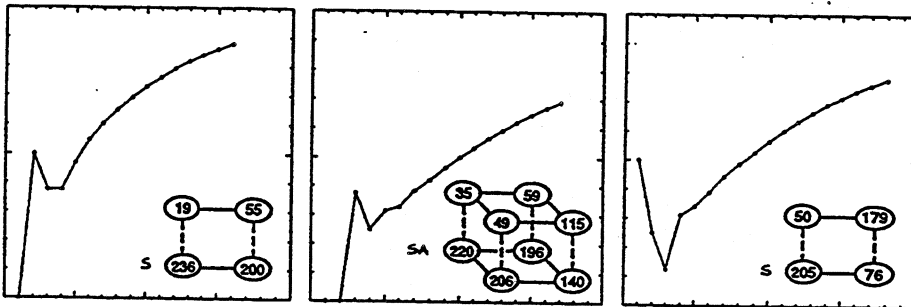


$K=3$  rules. G-density plotted against system size,  $N$ , for  $N=1$  to 18 continued...

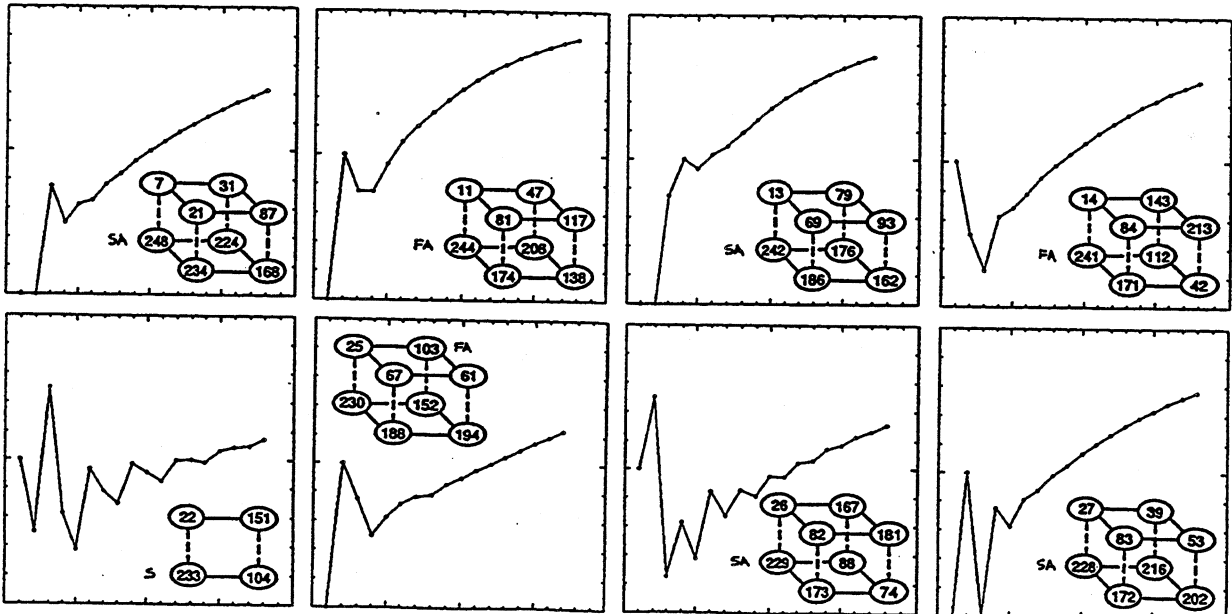
$Z = 0.5$  continued



$Z = 0.625$

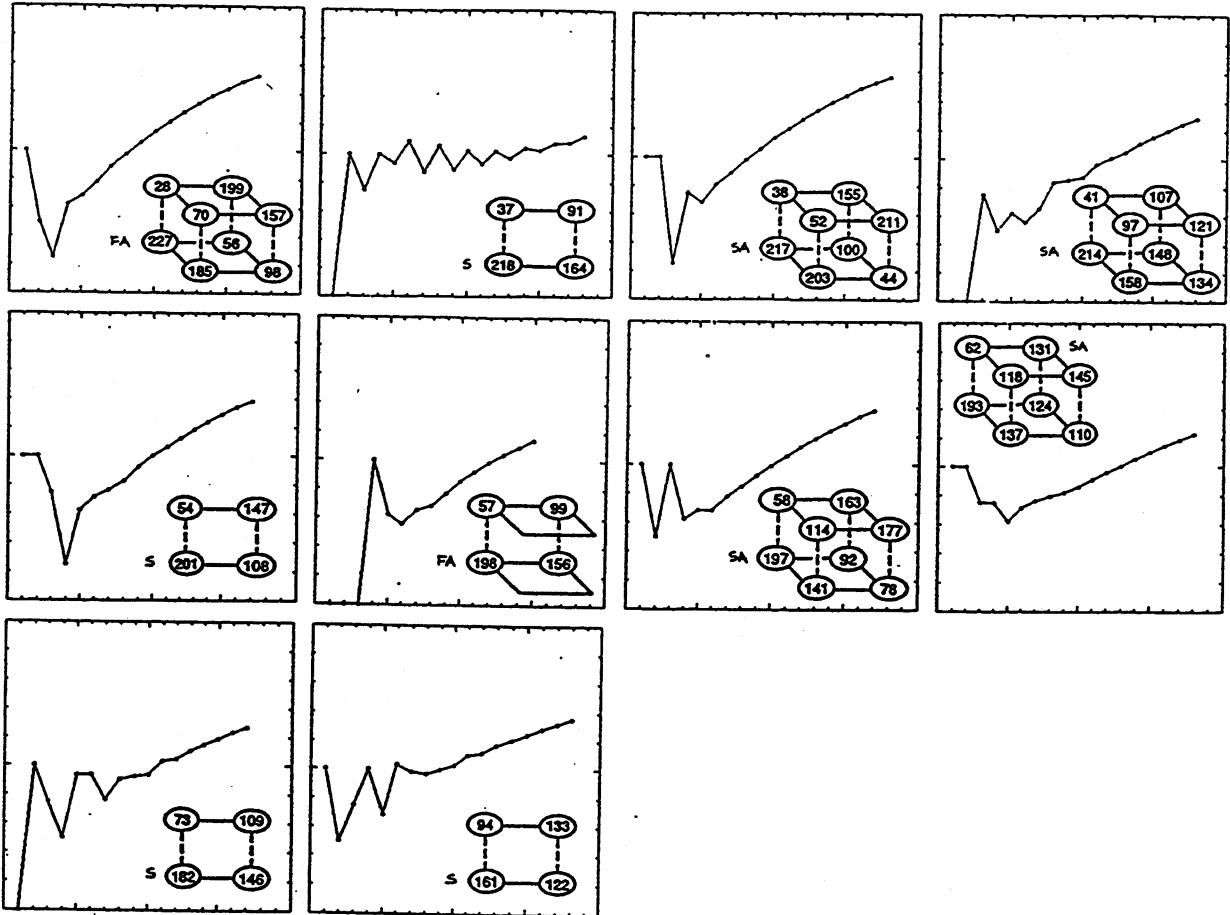


$Z = 0.75$

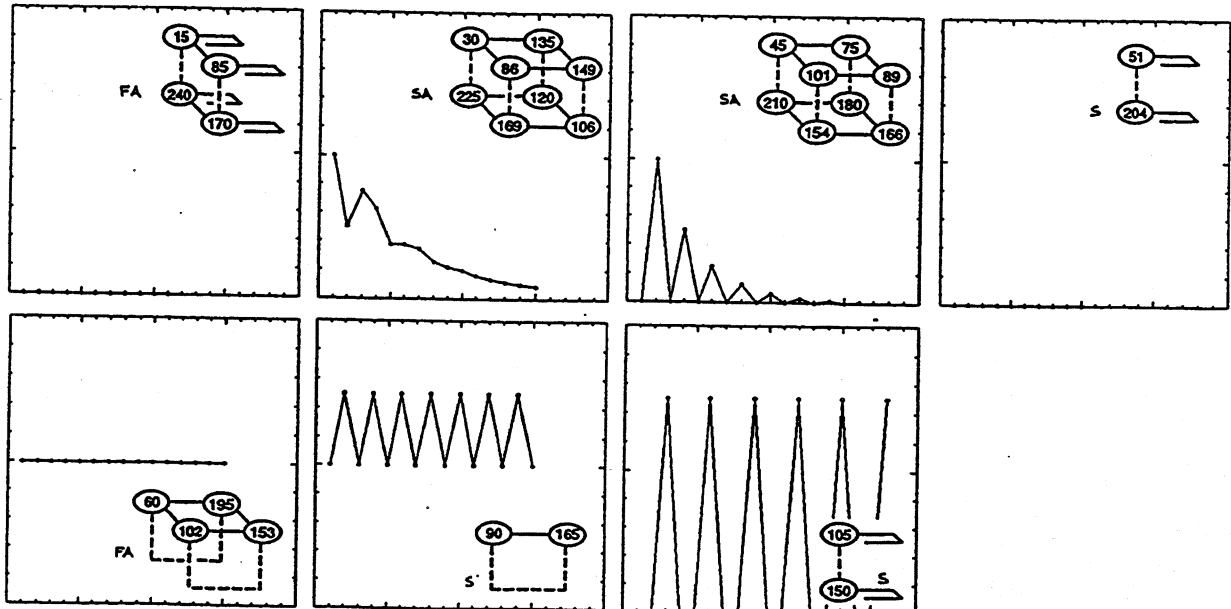


G-density plotted against system size,  $N$ , for  $N=1$  to 18 continued...

$Z = 0.75$  continued



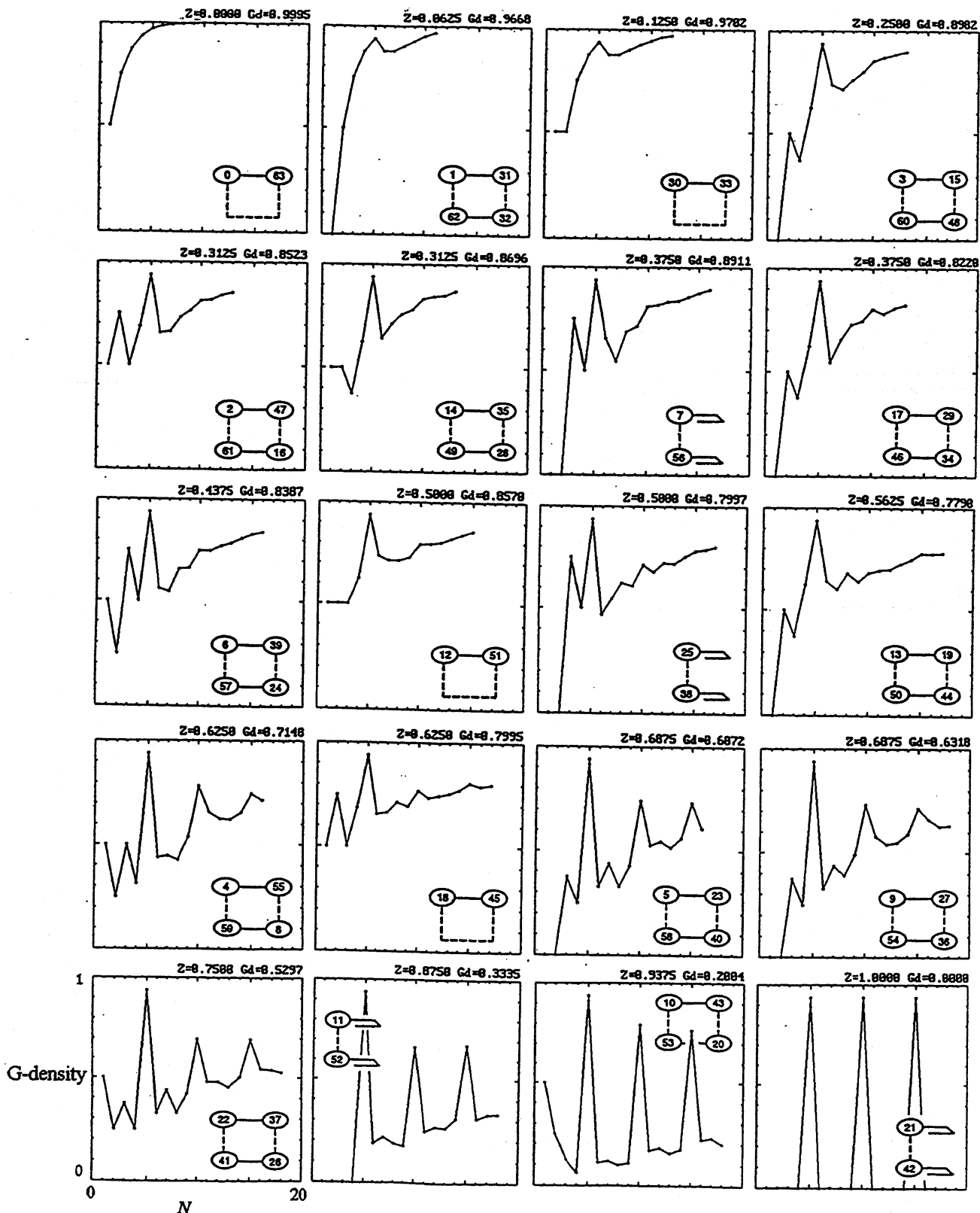
$Z = 1$





**K=5 totalistic rules.**

The garden of Eden density (G-density) is plotted against system size,  $N$ , as  $N$  is increased from 1 to 18. The rules in a rule cluster<sup>21</sup> have equivalent G-density, so the graphs have therefore been plotted for 20 K5 totalistic rule clusters. Totalistic rules are identified by their totalistic code<sup>16,21</sup>. The clusters are ordered by the Z parameter (which is indicated) and secondly by the lowest code number, in the top left hand corner of each cluster. The plots are based on the complete basin of attraction field for each value of  $N$ .

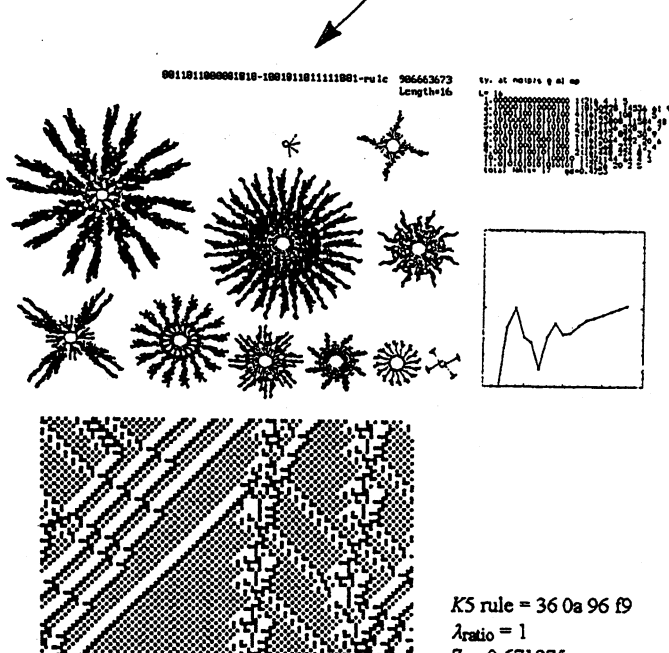
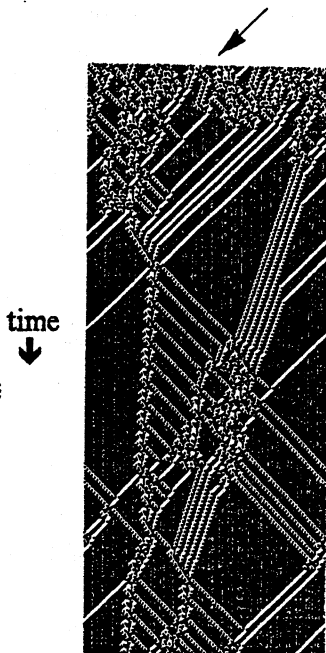


### Appendix 2 - Sample of complex rules, $K=5$ , Space-time patterns and the basin of attraction field.

The following characteristics of behaviour are illustrated in appendix 2 for each rule in the sample. A typical space-time pattern, a detail showing glider interactions, the basin of attraction field, data relating to the field, and a graph of G-density against increasing system size. A typical layout from appendix 2 is annotated below.

1. A typical space-time pattern form a random seed, system size 200 with periodic boundary conditions, 480 time-steps from the top down.

2. The basin of attraction field<sup>21</sup> for system size 16. Only one example of each equivalent basin is shown.



3. Data on the basin of attraction field<sup>21</sup> Key to data below.

4. Graph of G-density (y-axis, 0-1) against system size from 1 to 18. The x-axis is scaled 0-20.

5. The rule number in hex, the  $\lambda_{ratio}$  and Z parameter.

6. An evolved space-time pattern, (3 times the scale of 1) showing a detail of typical glider interactions. System size 100, periodic boundary conditions.

#### Key to data

system size, 16 cells → L= 16

ty.	at	no(p)	s	g	ml	mp
1.	0000000000000000	1(2)	6	4	1	5
2.	1000011010000110	1(8)	30928	14536	61	9
3.	0100110010110100	4(16)	220	108	11	5
4.	1010000110101010	1(16)	23808	11584	48	9
5.	0101010010110101	2(16)	1136	528	16	7
6.	0010101000101010	2(8)	1572	892	30	9
7.	0101101010001010	1(16)	2064	992	20	6
8.	1001100010011000	1(16)	568	232	12	4
9.	0010101001011010	2(16)	448	224	8	7
10.	011110101100010	1(32)	144	64	4	3
11.	0101010101010101	1(2)	26	20	2	5
total NATs=		17	gd=0.4753			

reminder of data order relating to each equivalent basin of attraction type.

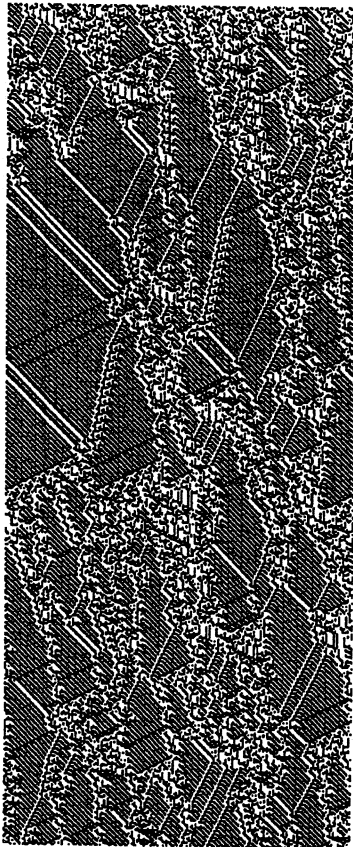
- ty. - equivalent basin type.
- at. - an attractor state.
- no. - number of equivalent basins.
- (p) - attractor period
- s - total states in the basin
- g - total garden-of Eden states in the basin.
- ml - maximum length of longest transient.
- mp - maximum pre-imaging.

total number of basins in the basin of attraction field.

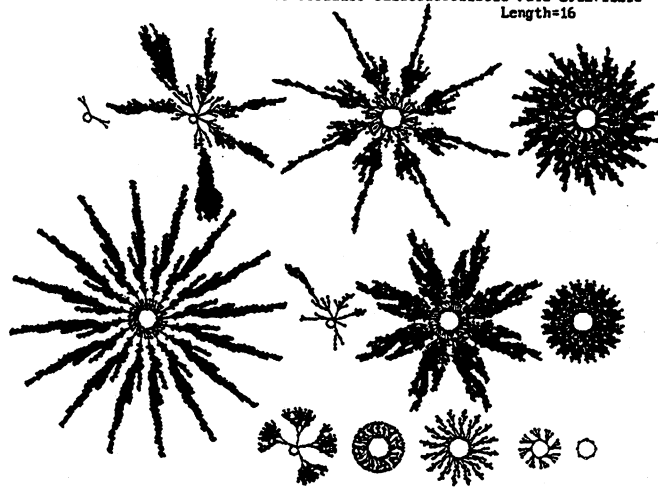
gd - garden-of Eden density of the basin of attraction field.

Appendix 2.2

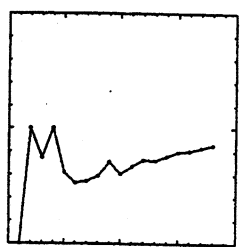
Andrew Wuensche



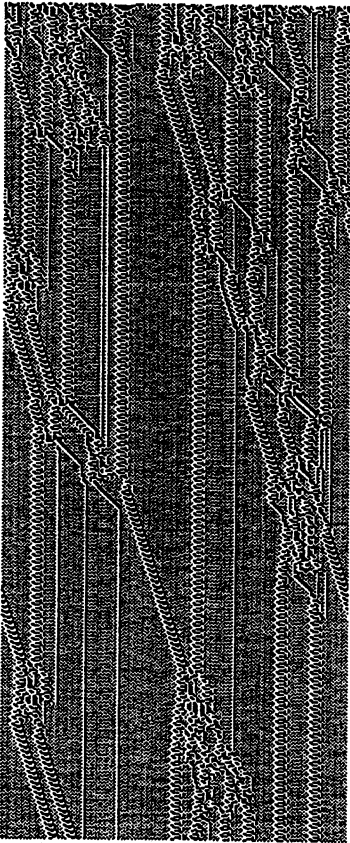
1818118118811188-8111881888118818-rule 2912711218  
Length=16



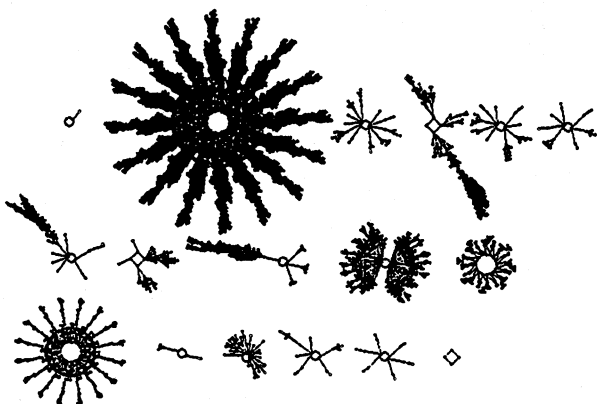
ty. at no(p)s g al ap  
L: 16  
1: 0000000000000000 1117 4 2 3  
2: 0000000000000000 1117 4 2 3  
3: 0000000000000000 1117 4 2 3  
4: 0000000000000000 1117 4 2 3  
5: 0000000000000000 1117 4 2 3  
6: 0000000000000000 1117 4 2 3  
7: 0000000000000000 1117 4 2 3  
8: 0000000000000000 1117 4 2 3  
9: 0000000000000000 1117 4 2 3  
10: 0000000000000000 1117 4 2 3  
11: 0000000000000000 1117 4 2 3  
12: 0000000000000000 1117 4 2 3  
13: 0000000000000000 1117 4 2 3  
14: 0000000000000000 1117 4 2 3  
15: 0000000000000000 1117 4 2 3  
16: 0000000000000000 1117 4 2 3  
total Nbits= 34 qd= .4031



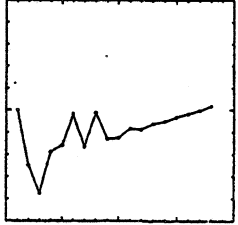
K5 rule = ad 9c 72 32  
 $\lambda_{ratio} = 1$   
Z = 0.75



1818811111888818-8181811888111111-rule 2814531135  
Length=16



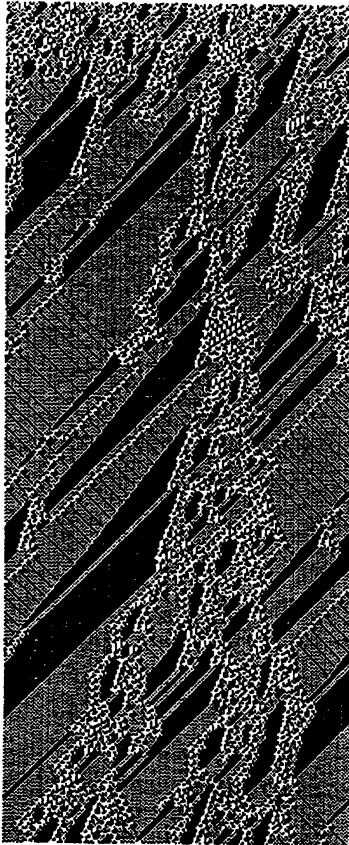
ty. at no(p)s g al ap  
L: 16  
1: 0101001001001001 112 1 1 2 3008 48 12  
2: 0101001001001001 112 1 1 2 3008 48 12  
3: 0101001001001001 112 1 1 2 3008 48 12  
4: 0101001001001001 112 1 1 2 3008 48 12  
5: 0101001001001001 112 1 1 2 3008 48 12  
6: 0101001001001001 112 1 1 2 3008 48 12  
7: 0101001001001001 112 1 1 2 3008 48 12  
8: 0101001001001001 112 1 1 2 3008 48 12  
9: 0101001001001001 112 1 1 2 3008 48 12  
10: 0101001001001001 112 1 1 2 3008 48 12  
11: 0101001001001001 112 1 1 2 3008 48 12  
12: 0101001001001001 112 1 1 2 3008 48 12  
13: 0101001001001001 112 1 1 2 3008 48 12  
14: 0101001001001001 112 1 1 2 3008 48 12  
15: 0101001001001001 112 1 1 2 3008 48 12  
16: 0101001001001001 112 1 1 2 3008 48 12  
total Nbits= 158 qd=0.4838



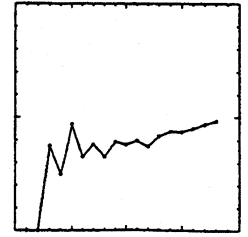
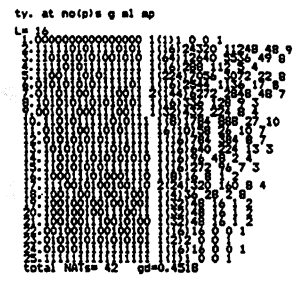
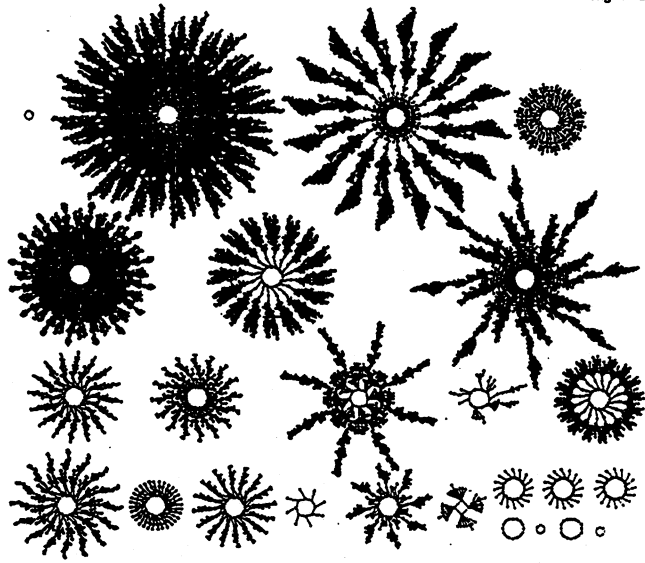
K5 rule = a7 c2 56 3f  
 $\lambda_{ratio} = 0.875$   
Z = 0.75

Complexity in One-D Cellular Automata

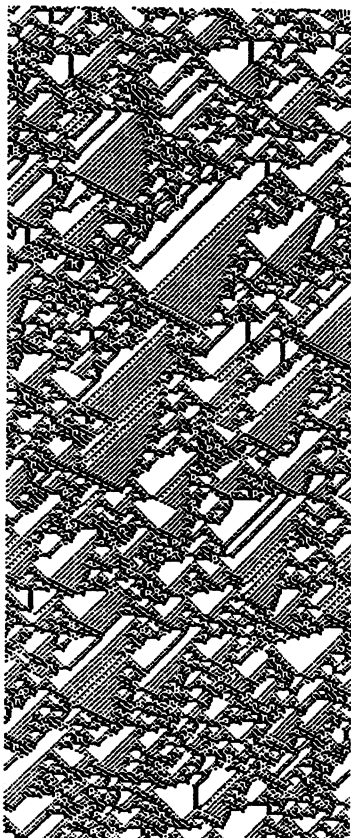
Appendix 2.3



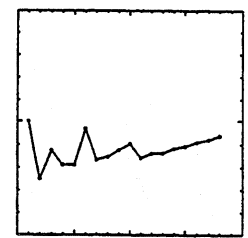
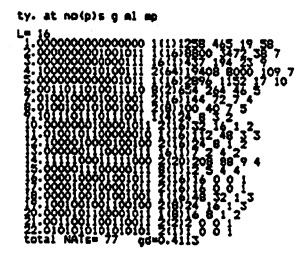
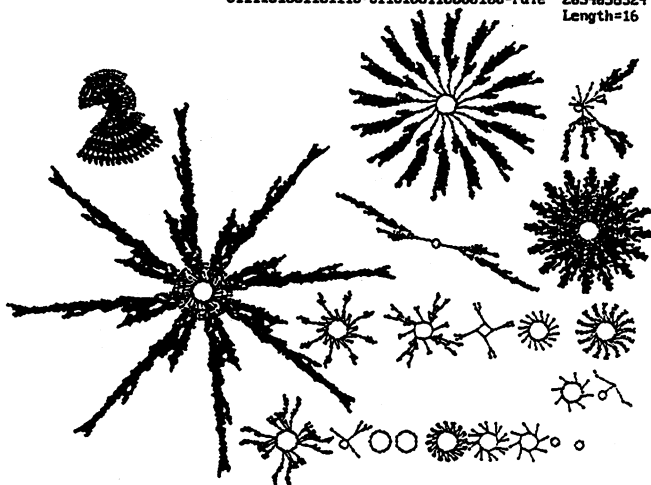
1101010110011000-1010110101111010-rule 3583552898  
Length=16



K5 rule = d5 98 ad 7a  
 $\lambda_{ratio} = 0.875$   
 $Z = 0.75$



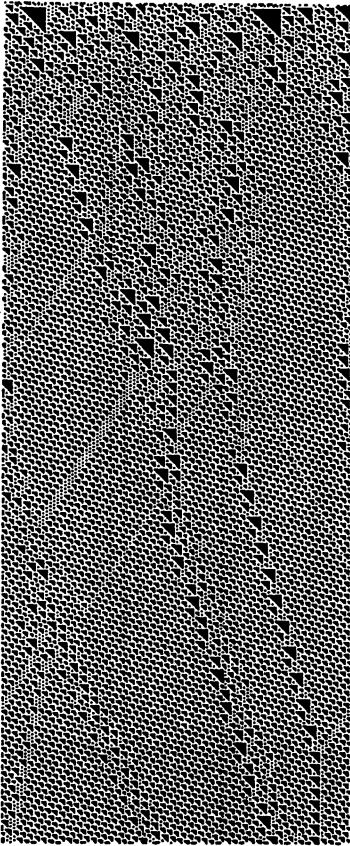
011101001101110-0110100110000100-rule 2054056324  
Length=16



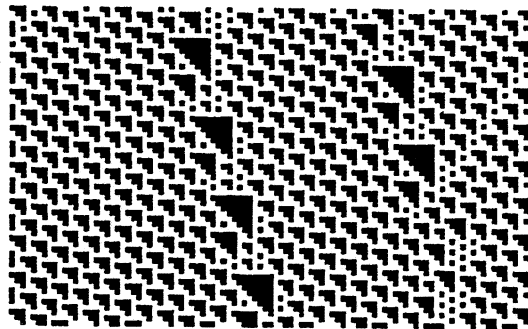
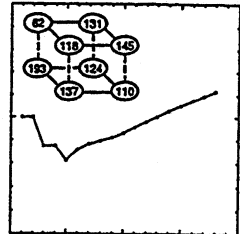
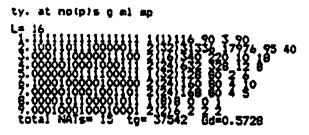
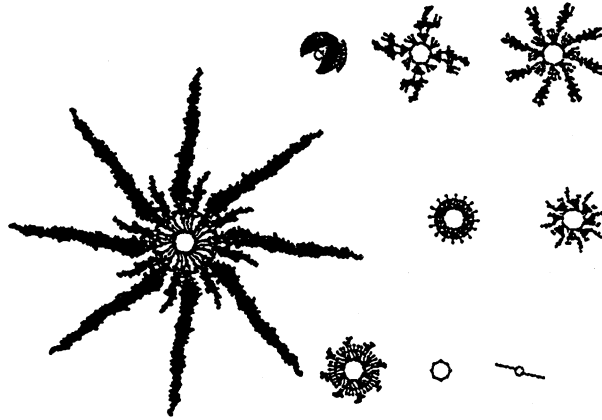
K5 rule = 7a 6e 69 84  
 $\lambda_{ratio} = 1$   
 $Z = 0.75$

Appendix 2.4

Andrew Wuensche

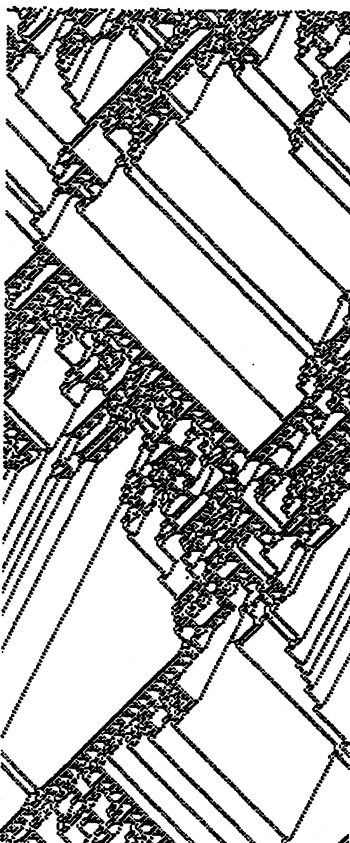


1111000000000011-1111000000000011-rule 4826789891  
=3-rule 193 -11000001 Length=16

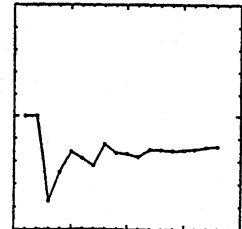
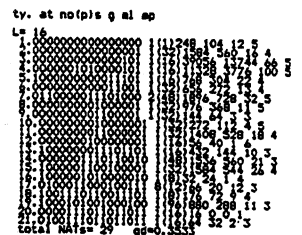
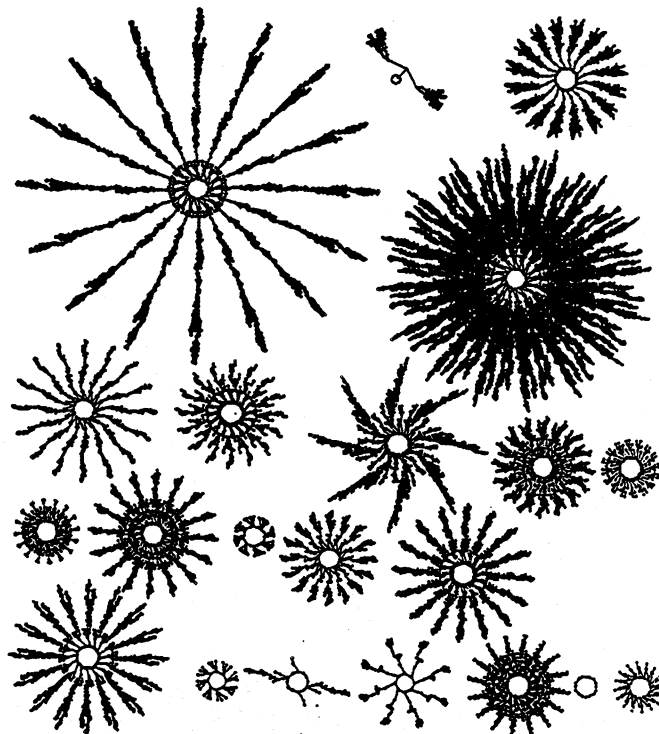


Acknowledgements to  
Wolfram <sup>17</sup>

K3 rule = c1 (dec 193)  
 $\lambda_{ratio} = 0.75$   
 $Z = 0.75$



010110001101010-0100110110011000-rule 1558478552  
Length=16

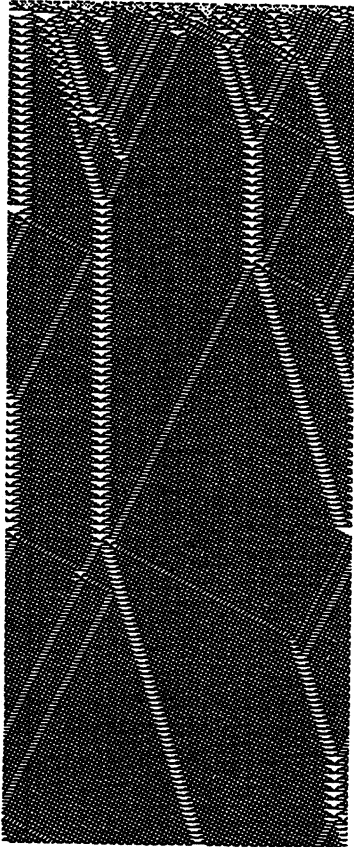


for detail of space-time  
pattern see figure .

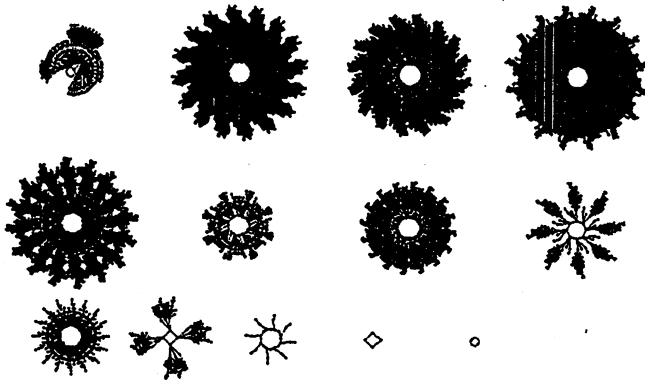
K5 rule = 5c 6a 4d 98  
 $\lambda_{ratio} = 0.9375,$   
 $Z = 0.7265625$

Complexity in One-D Cellular Automata

Appendix 2.5

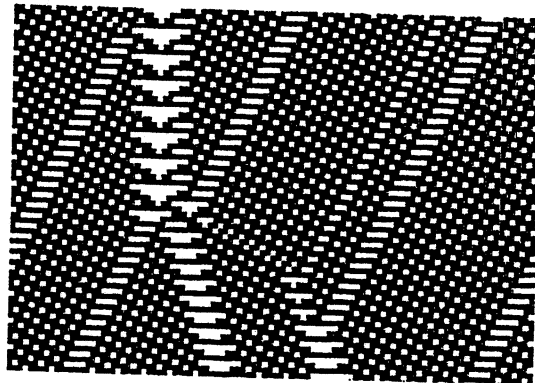
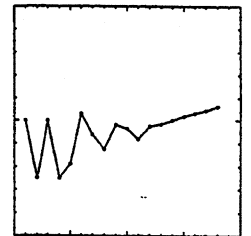


8111181111881881-81811181181818-rule 2876794282  
Length=16

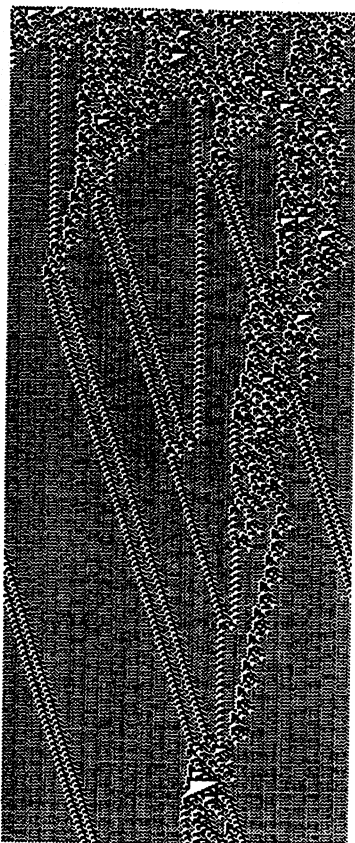


ty. at no(p)l s g al ap

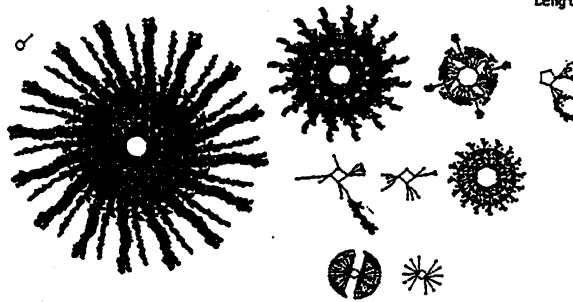
L	16	0000000000000000	1111778	416	9	60
1	00000	0101010101010101	1111111	11111	11111	11111
2	00000	0101010101010101	1111111	11111	11111	11111
3	01010	0101010101010101	1111111	11111	11111	11111
4	01010	0101010101010101	1111111	11111	11111	11111
5	01010	0101010101010101	1111111	11111	11111	11111
6	01010	0101010101010101	1111111	11111	11111	11111
7	01010	0101010101010101	1111111	11111	11111	11111
8	01010	0101010101010101	1111111	11111	11111	11111
9	01010	0101010101010101	1111111	11111	11111	11111
10	01010	0101010101010101	1111111	11111	11111	11111
11	01010	0101010101010101	1111111	11111	11111	11111
12	01010	0101010101010101	1111111	11111	11111	11111
13	01010	0101010101010101	1111111	11111	11111	11111
14	01010	0101010101010101	1111111	11111	11111	11111
15	01010	0101010101010101	1111111	11111	11111	11111
total	NAI	16	qs	0.532		



K5 rule = 7b c9 5d aa  
 $\lambda_{ratio} = 0.8125$   
 $Z = 0.7265625$

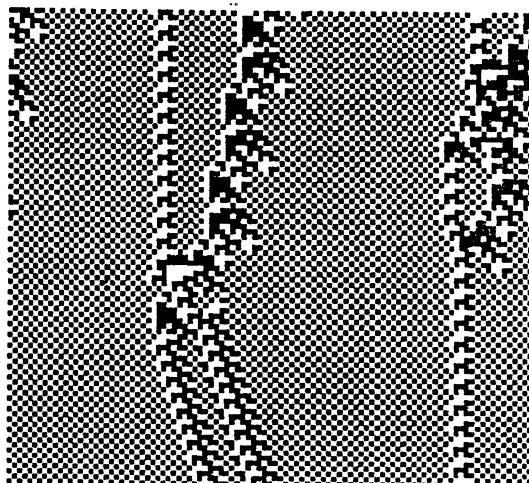
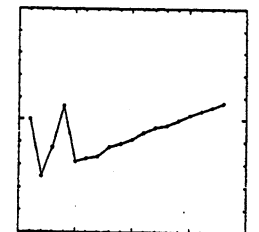


8111111818888118-188181181181118-rule 2122749662  
Length=16



ty. at no(p)l s g al ap

L	16	0000000000000000	11115	540	28200	61 12
1	01010	0101010101010101	1111111	11111	11111	11111
2	01010	0101010101010101	1111111	11111	11111	11111
3	01010	0101010101010101	1111111	11111	11111	11111
4	01010	0101010101010101	1111111	11111	11111	11111
5	01010	0101010101010101	1111111	11111	11111	11111
6	01010	0101010101010101	1111111	11111	11111	11111
7	01010	0101010101010101	1111111	11111	11111	11111
8	01010	0101010101010101	1111111	11111	11111	11111
9	01010	0101010101010101	1111111	11111	11111	11111
10	01010	0101010101010101	1111111	11111	11111	11111
11	01010	0101010101010101	1111111	11111	11111	11111
12	01010	0101010101010101	1111111	11111	11111	11111
13	01010	0101010101010101	1111111	11111	11111	11111
14	01010	0101010101010101	1111111	11111	11111	11111
15	01010	0101010101010101	1111111	11111	11111	11111
total	NAI	16	qs	0.532		

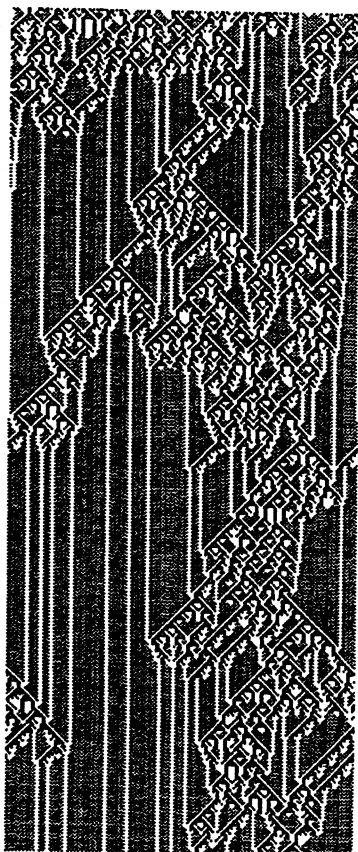


K5 rule = 7e 86 96 de  
 $\lambda_{ratio} = 0.8125$   
 $Z = 0.6875$

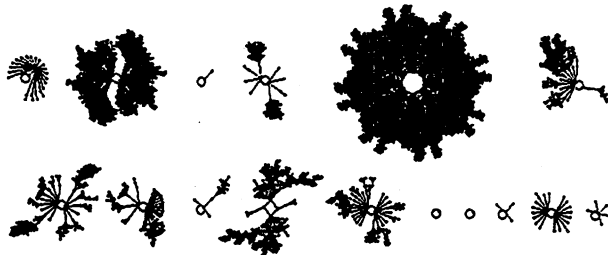


Appendix 2.6

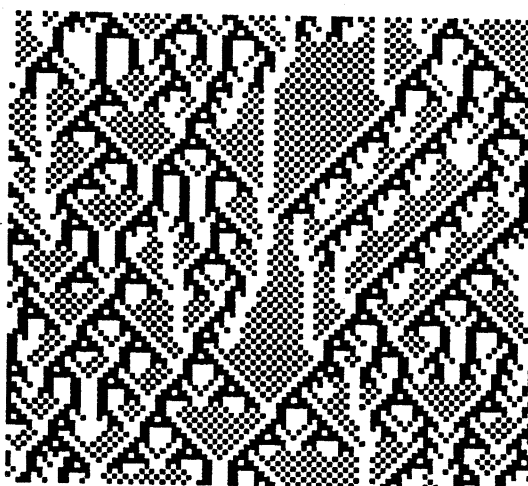
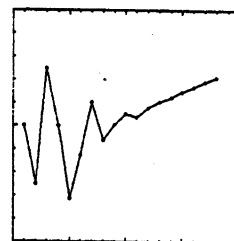
Andrew Wuensche



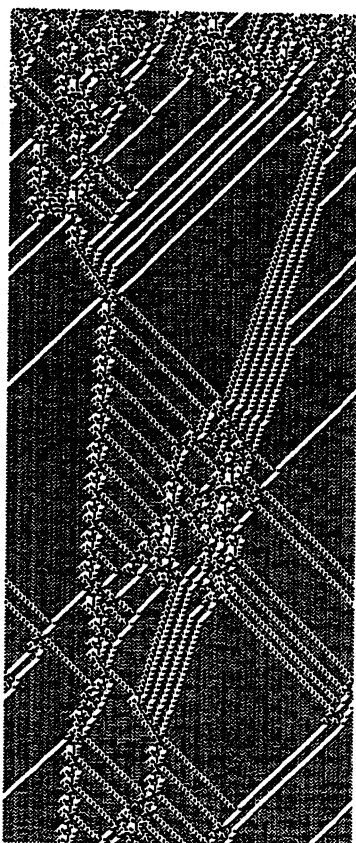
0811181881881888-1811818111888188-rule 97764365Z  
Length=16



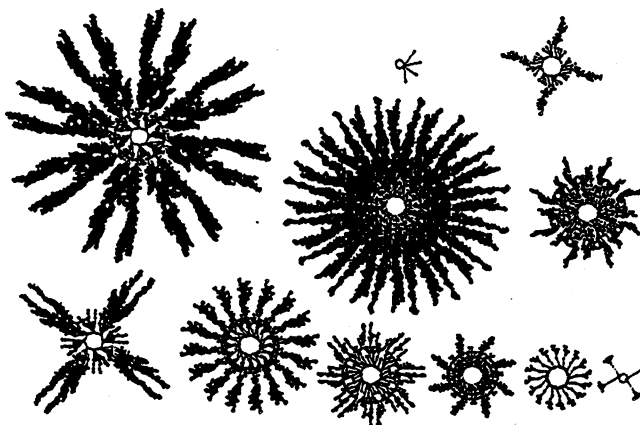
ty. at notpis g al ap  
L: 18  
1 10 20 30 40 47  
2 10 20 30 40 47  
3 10 20 30 40 47  
4 10 20 30 40 47  
5 10 20 30 40 47  
6 10 20 30 40 47  
7 10 20 30 40 47  
8 10 20 30 40 47  
9 10 20 30 40 47  
10 10 20 30 40 47  
11 10 20 30 40 47  
12 10 20 30 40 47  
13 10 20 30 40 47  
14 10 20 30 40 47  
15 10 20 30 40 47  
16 10 20 30 40 47  
total Nbits=175 qd=0.6670



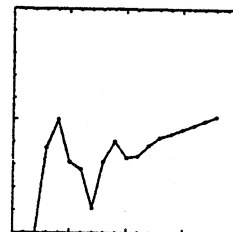
K5 rule = 3a 48 b5 c4  
 $\lambda_{ratio} = 0.875$   
Z = 0.6875



0811811888881818-188181181111881-rule 986663673  
Length=16



ty. at notpis g al ap  
L: 18  
1 10 20 30 40 47  
2 10 20 30 40 47  
3 10 20 30 40 47  
4 10 20 30 40 47  
5 10 20 30 40 47  
6 10 20 30 40 47  
7 10 20 30 40 47  
8 10 20 30 40 47  
9 10 20 30 40 47  
10 10 20 30 40 47  
11 10 20 30 40 47  
12 10 20 30 40 47  
13 10 20 30 40 47  
14 10 20 30 40 47  
15 10 20 30 40 47  
16 10 20 30 40 47  
total Nbits=175 qd=0.6725



K5 rule = 36 0a 96 f9  
 $\lambda_{ratio} = 1$   
Z = 0.671875

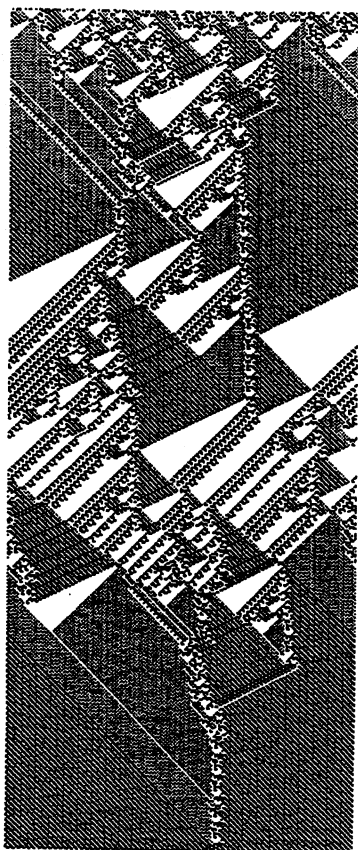




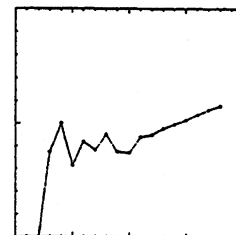
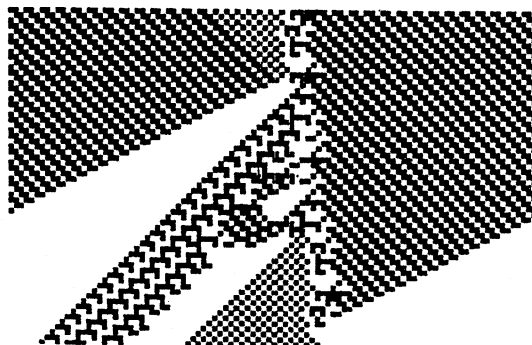
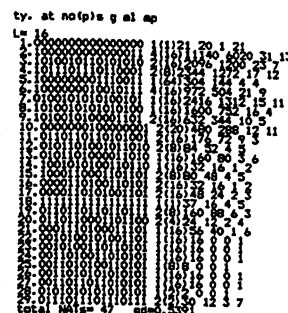
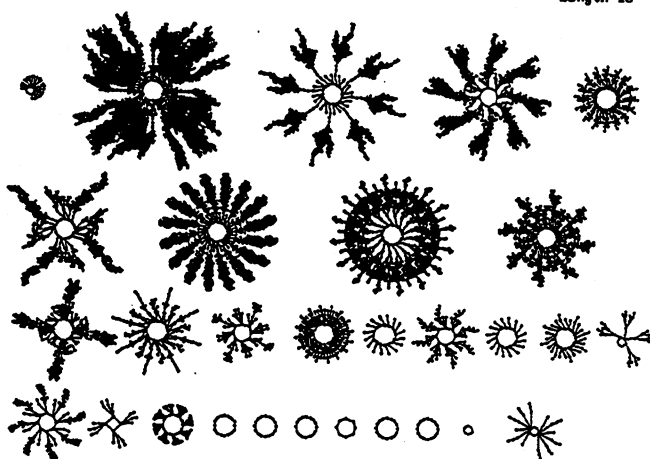


Complexity in One-D Cellular Automata

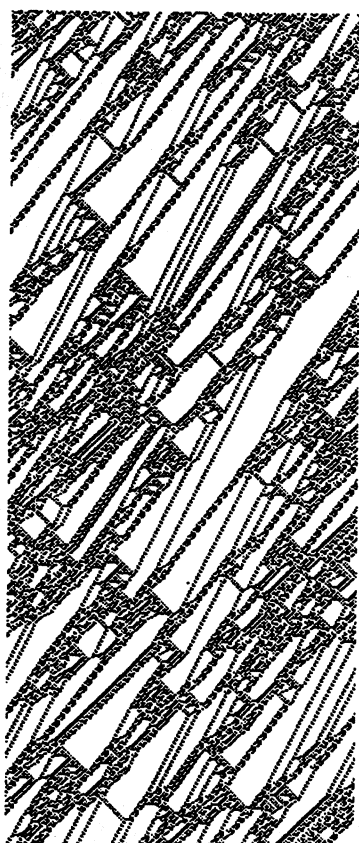
Appendix 2.9



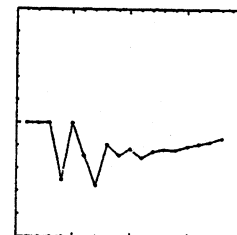
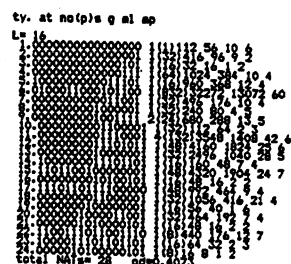
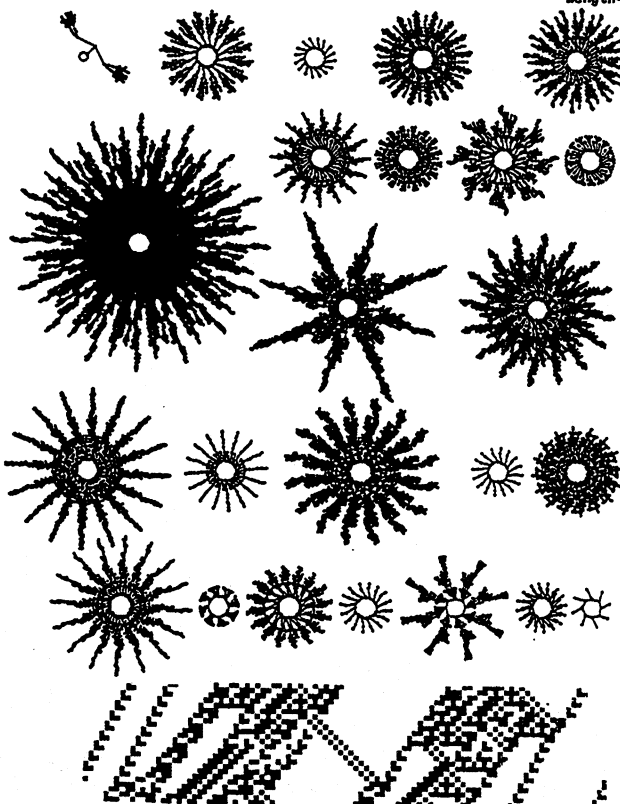
1811118811888818-8818811188811108-rule 3162646388  
Length=16



K5 rule = bc 82 27 1c  
 $\lambda_{ratio} = 0.875$   
 $Z = 0.671875$



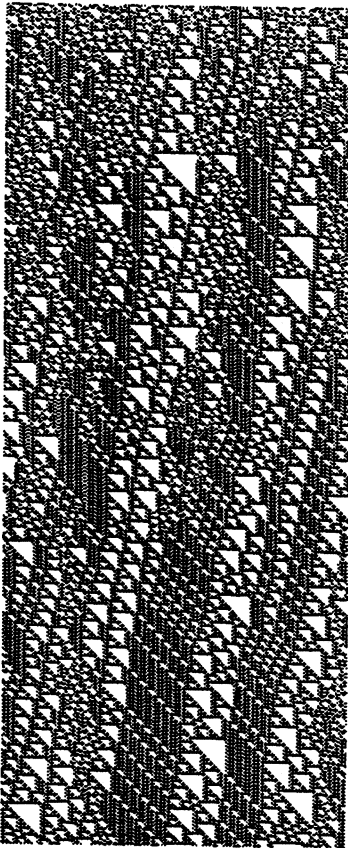
8181118881181818-8188111188811888-rule 1558471864  
Length=16



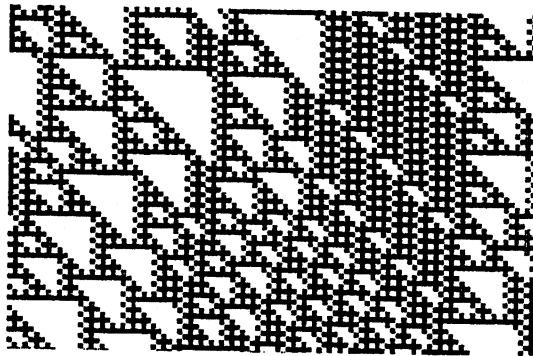
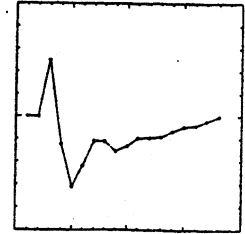
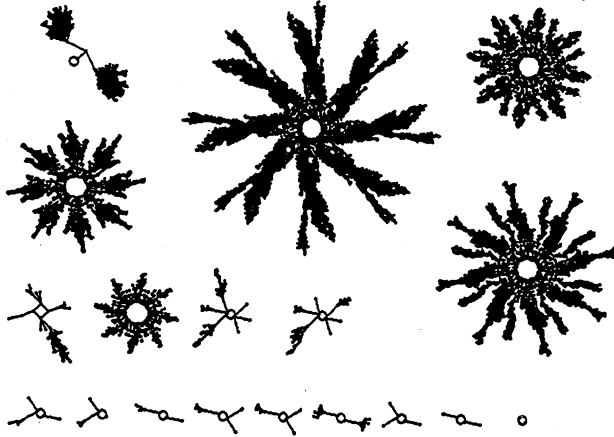
K5 rule = 5c 6a 4f 98  
 $\lambda_{ratio} = 1$   
 $Z = 0.671875$

Appendix 2.10

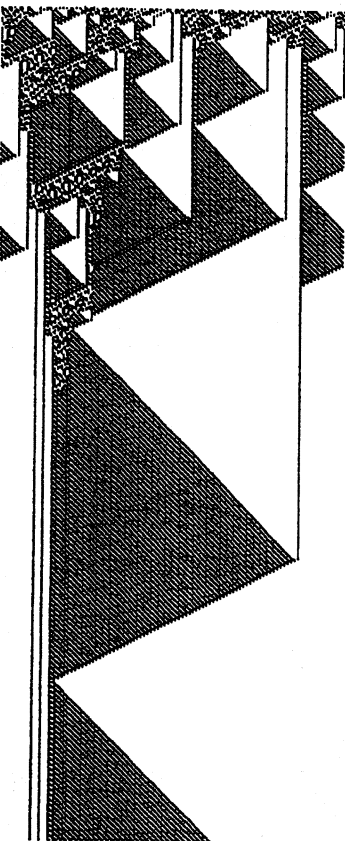
Andrew Wuensche



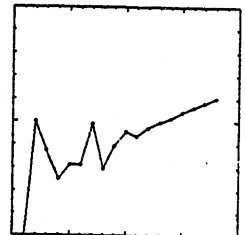
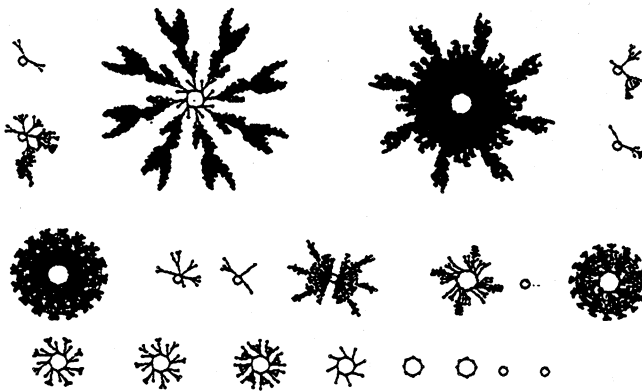
81888118811188-11881118881888-rule 1195167512  
Length=16



K5 rule = 47 3c cf 18  
 $\lambda_{ratio} = 1$   
Z = 0.625



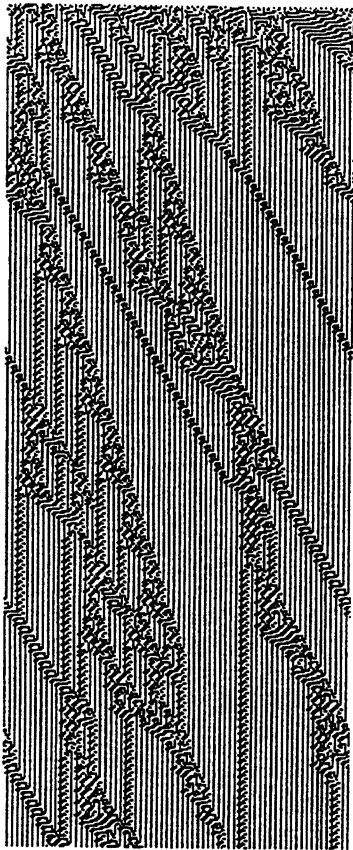
181188118888118-881181811118888-rule 3112581872  
Length=16



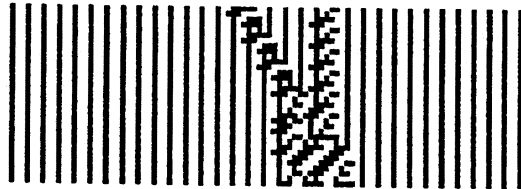
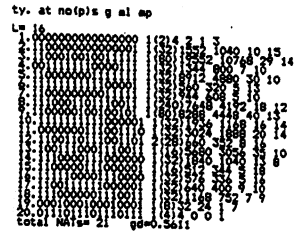
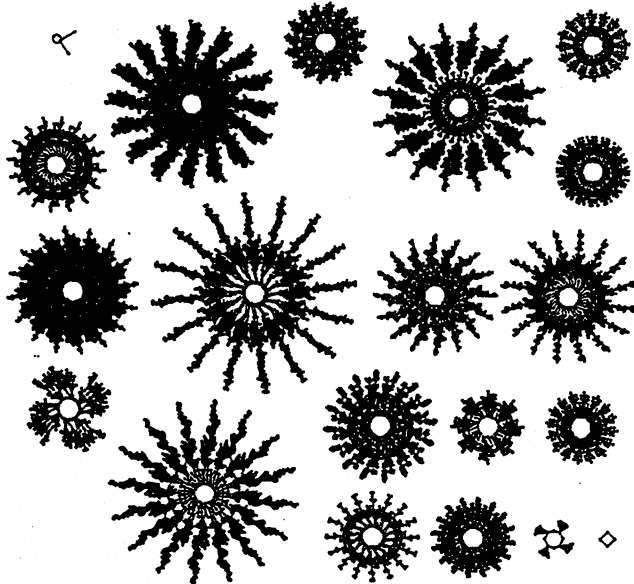
K5 rule = b9 86 3a f0  
 $\lambda_{ratio} = 1$   
Z = 0.671875

Complexity in One-D Cellular Automata

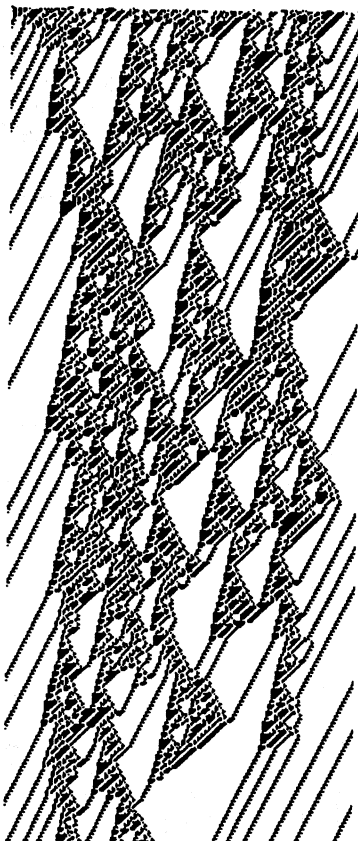
Appendix 2.11



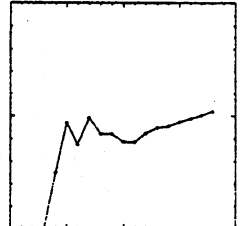
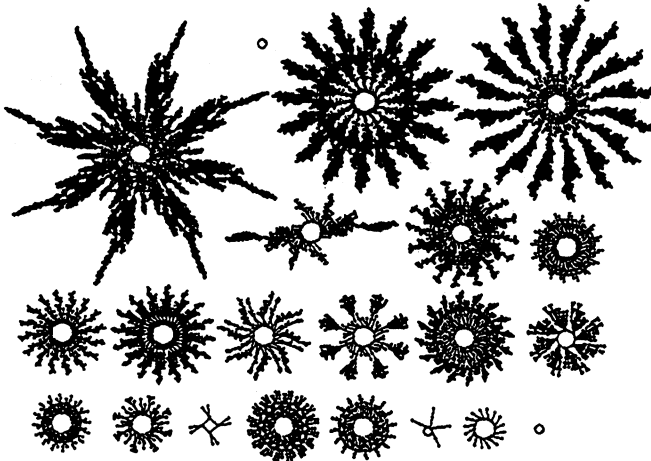
0010100000111001-1100010110111011-rule 674874811  
Length=16



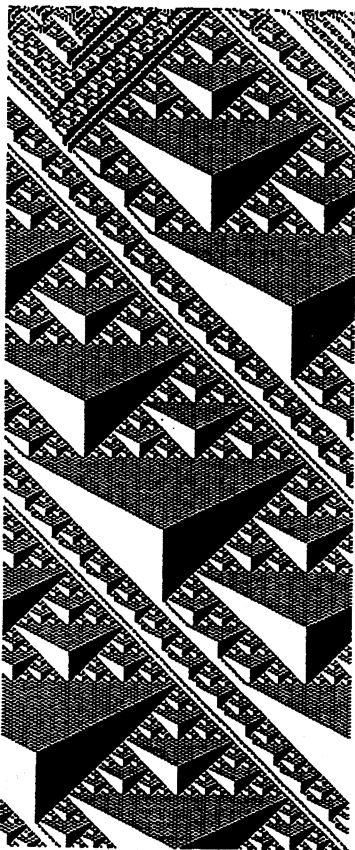
K5 rule = 28 39 c5 bb  
 $\lambda_{ratio} = 1$   
 $Z = 0.625$



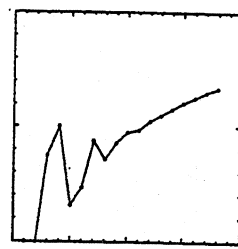
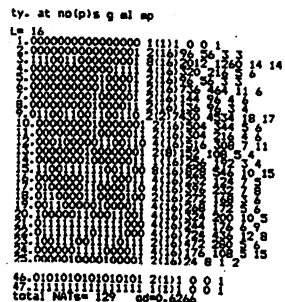
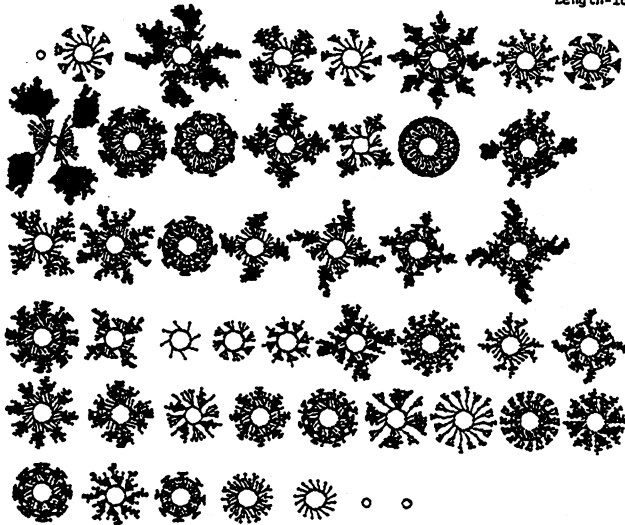
111000001001010-1010110111110100-rule 376298588  
Length=16



K5 rule = e0 4a ad f4  
 $\lambda_{ratio} = 1$   
 $Z = 0.625$

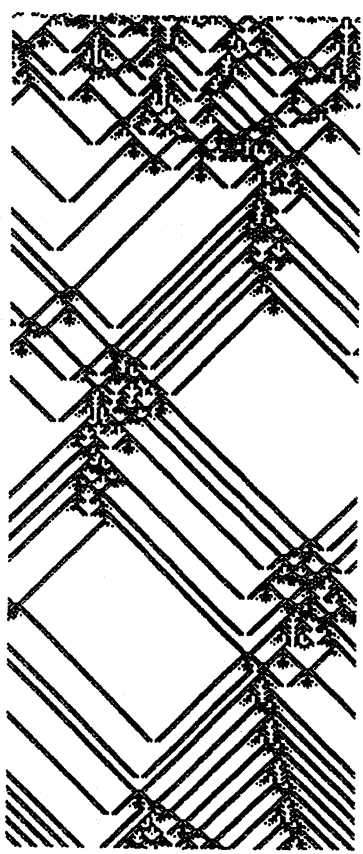
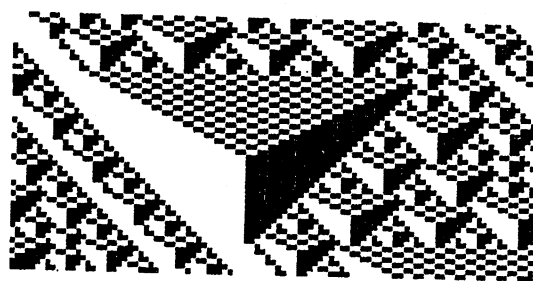


118080111811180-1118001118018080-rule 3283936144  
Length=16

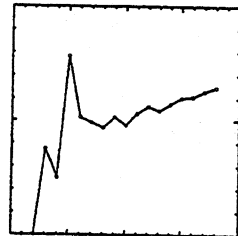
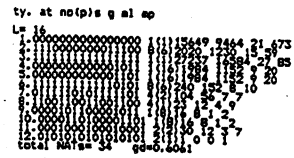
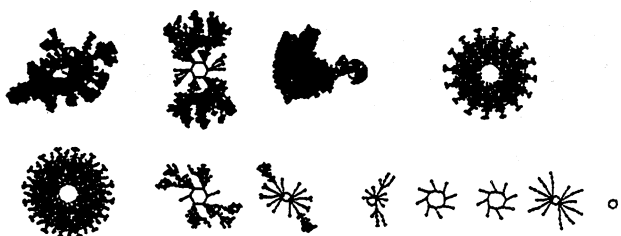


Acknowledgements to  
Wentian Li<sup>10</sup>

K5 rule = c3 bc e3 90  
 $\lambda_{ratio} = 1$   
Z = 0.625



181118118180818-818180811801808-rule 3181533384  
Length=16



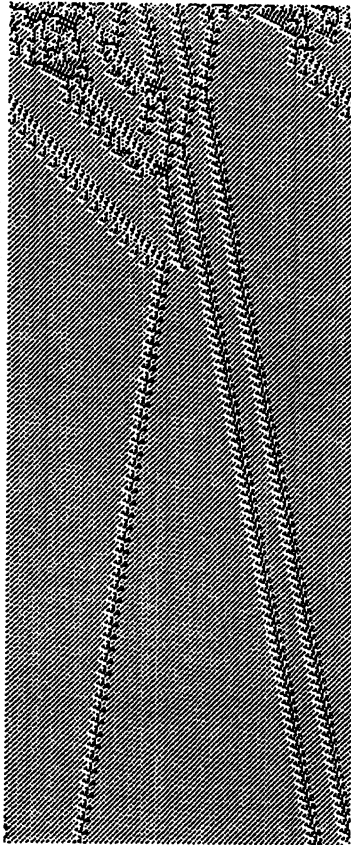
Acknowledgements to  
Aizawa *et al*<sup>1</sup>

K5 rule = bd a2 58 c8  
 $\lambda_{ratio} = 0.9375$   
Z = 0.6171875

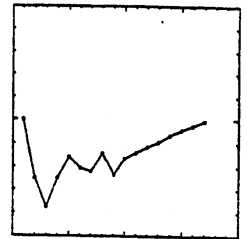
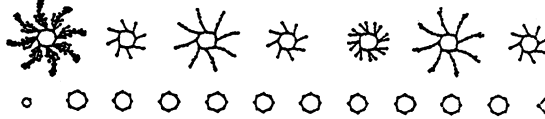
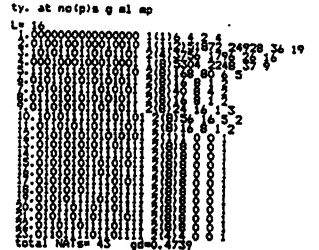


Complexity in One-D Cellular Automata

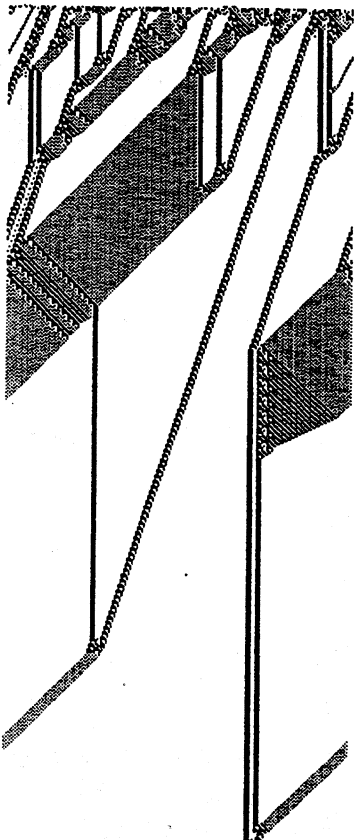
Appendix 2.13



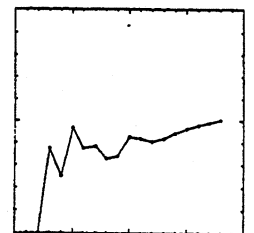
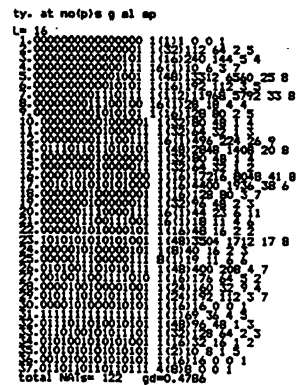
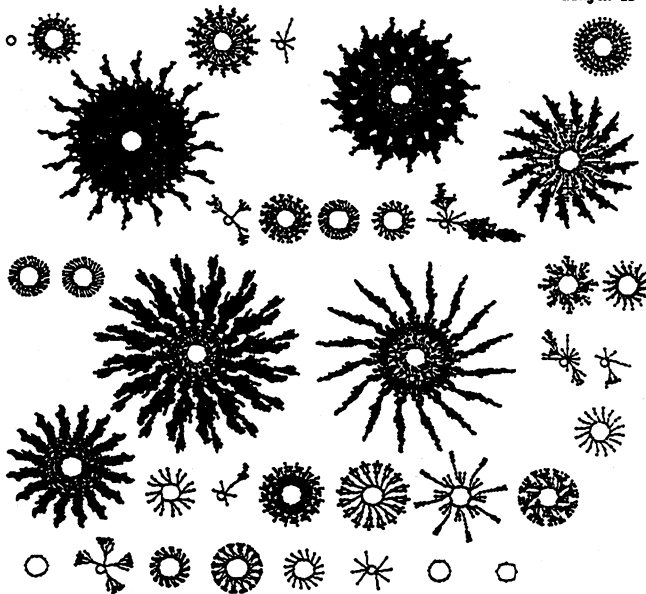
0110110011100111-0101000010100100-rule 1027098700  
Length=16



K5 rule = 6c e7 50 a4  
 $\lambda_{ratio} = 0.9375$   
 $Z = 0.6171875$



101110010000010-01100110110100-rule 3162662612  
Length=16

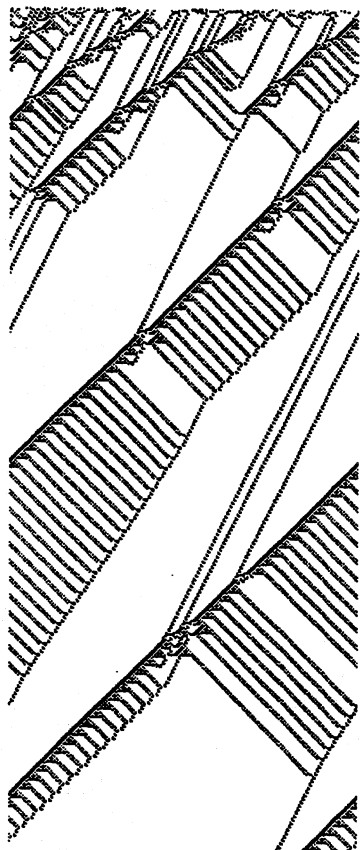


K5 rule = bc 82 66 d4  
 $\lambda_{ratio} = 0.9375$   
 $Z = 0.6171875$

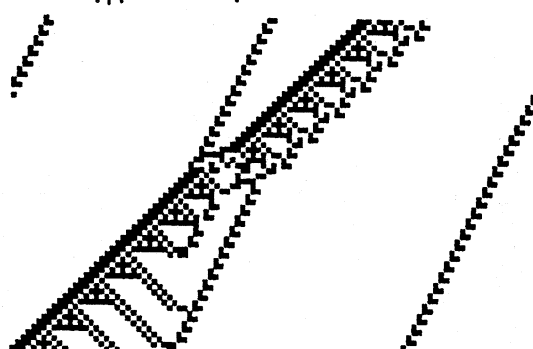
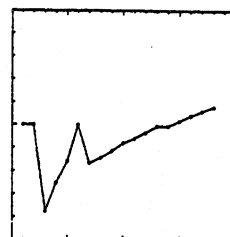
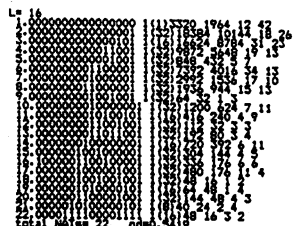
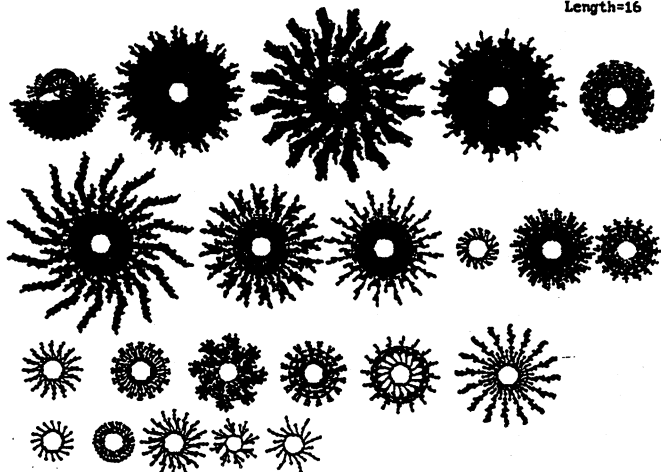


Appendix 2.14

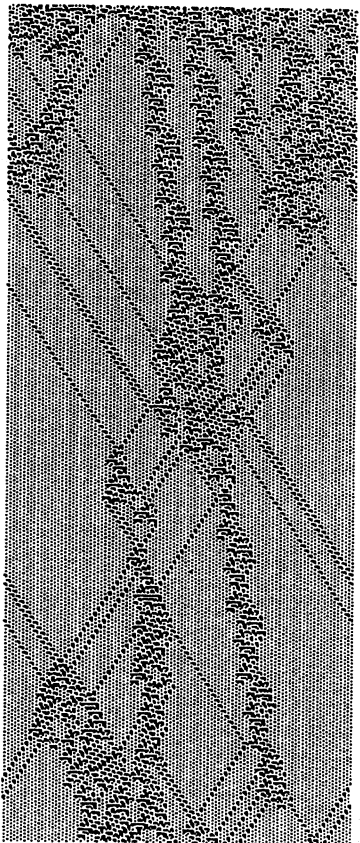
Andrew Wuensche



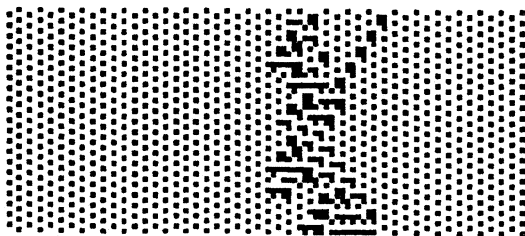
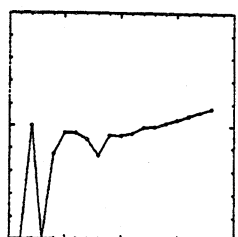
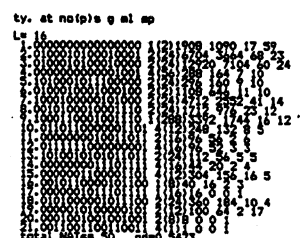
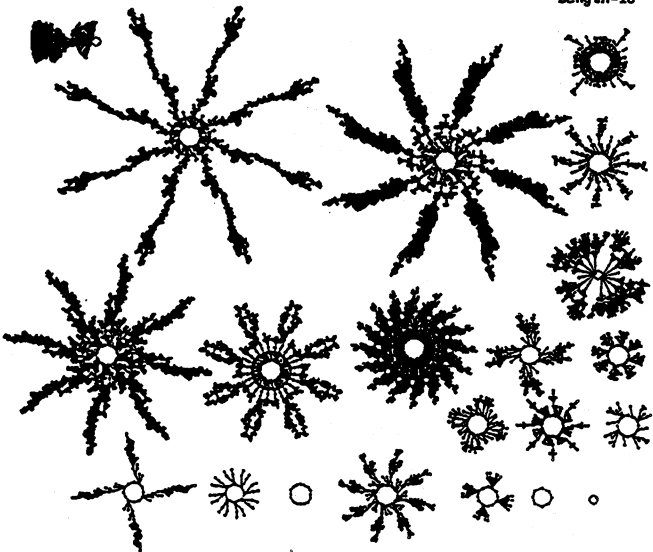
000110000101010-010001110011000-rule 472532888  
Length=16



K5 rule = 1c 2a 47 98  
 $\lambda_{ratio} = 0.8125$   
Z = 0.6171875



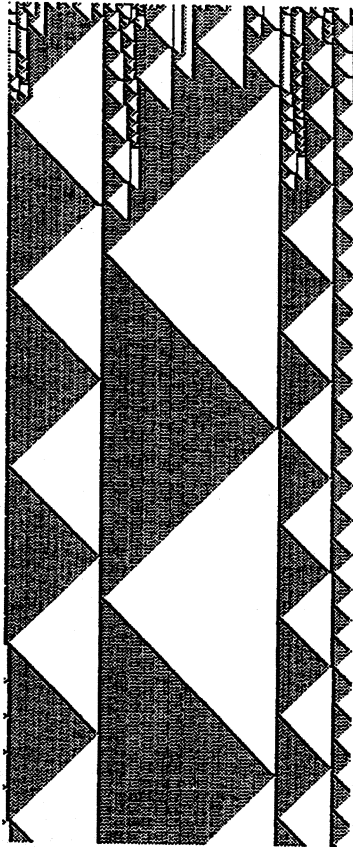
010110000000111-101100000100011-rule 147690883  
Length=16



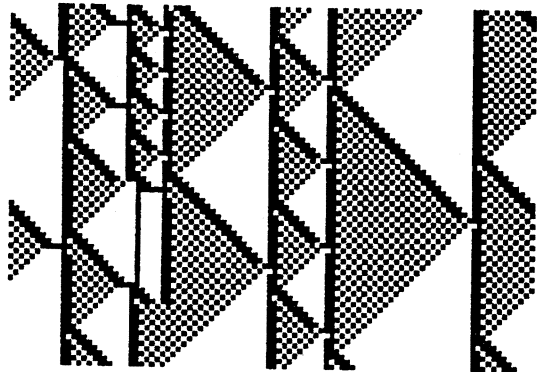
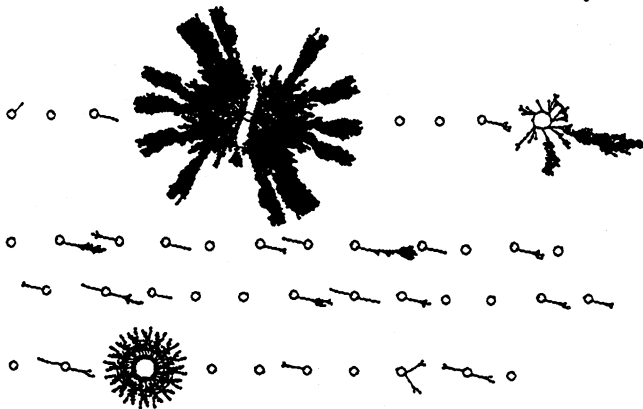
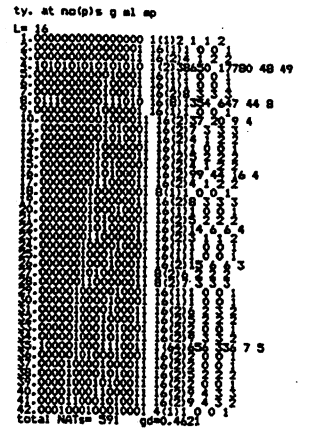
K5 rule = 58 07 b0 43  
 $\lambda_{ratio} = 0.75$   
Z = 0.609375

Complexity in One-D Cellular Automata

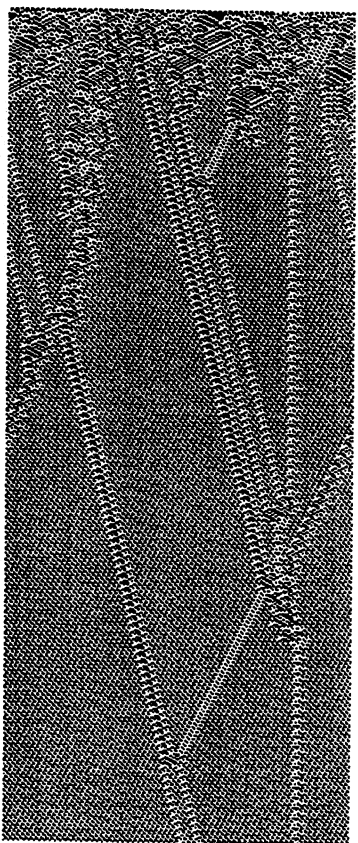
Appendix 2.15



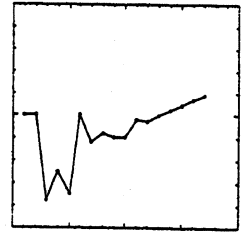
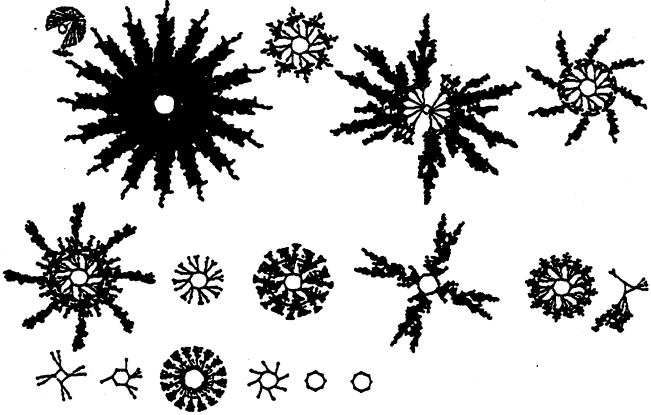
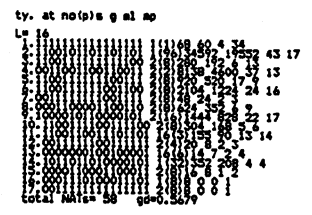
010111101000100-0110110011110000-rule 1598319856  
Length=16



K5 rule = 5f 44 6c f0  
 $\lambda_{ratio} = 1$   
Z = 0.58984375



1101101000001010-010000100011011-rule 3650105627  
Length=16

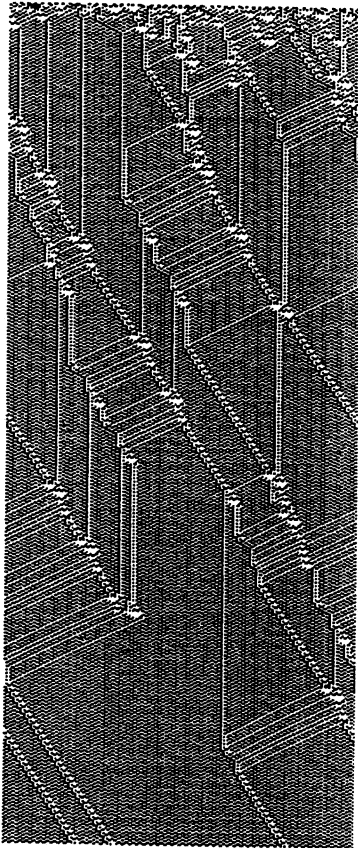


K5 rule = da 0a 43 1b  
 $\lambda_{ratio} = 0.875$   
Z = 0.58984375

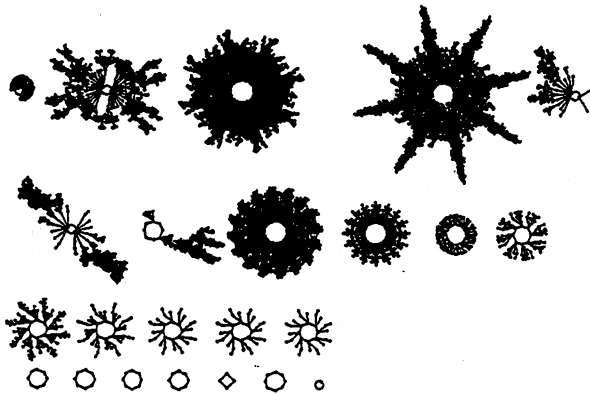


Appendix 2.16

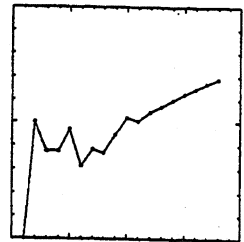
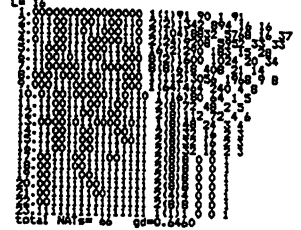
Andrew Wuensche



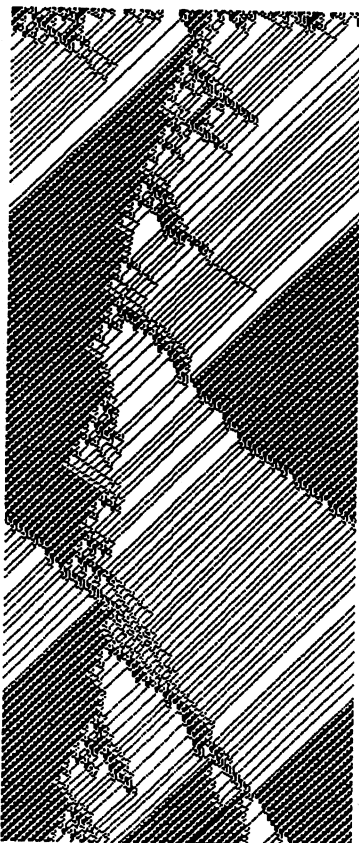
1101001010001111-0000001000101100-rule 3532587564  
Length=16



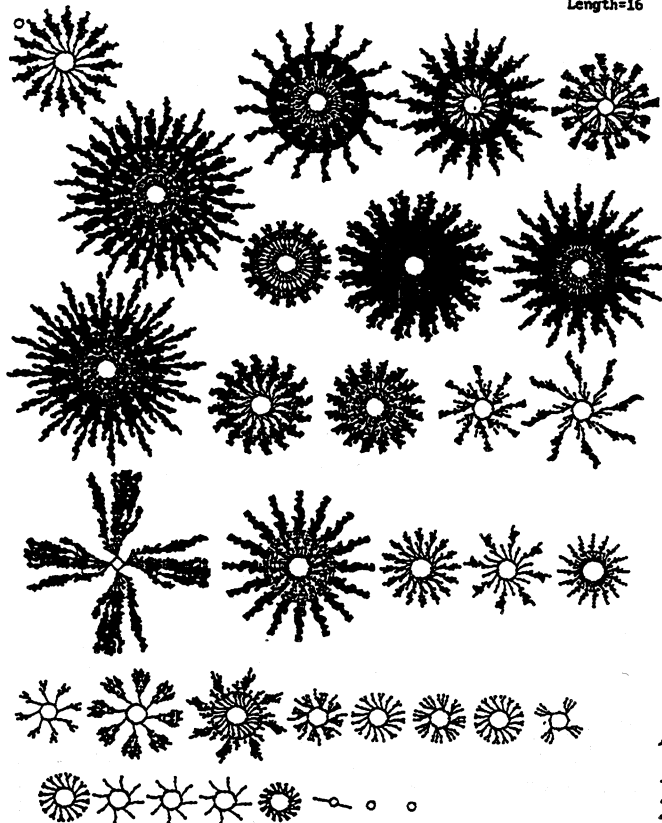
ty. at noip's g al ap



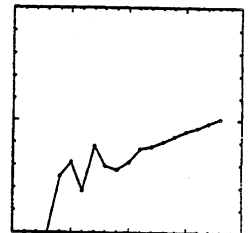
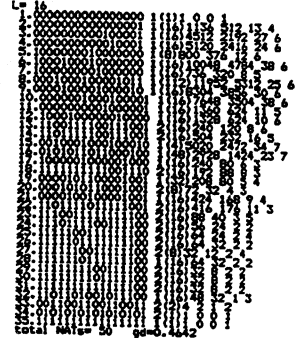
K5 rule = d2 8f 02 2c  
 $\lambda_{ratio} = 0.8125$   
Z = 0.578125



1100001110111100-1110001010000100-rule 3283935876  
Length=16



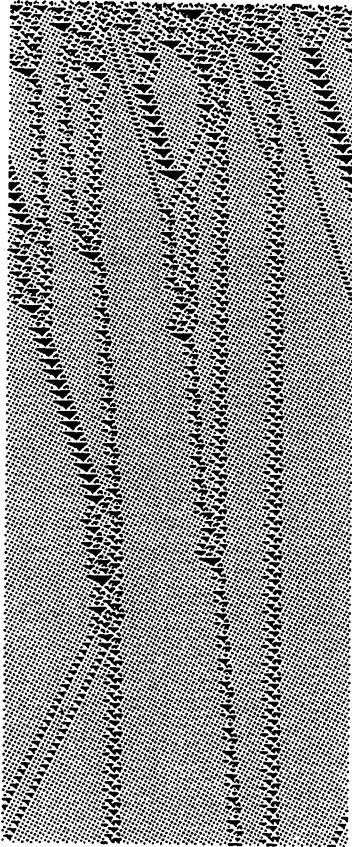
ty. at noip's g al ap



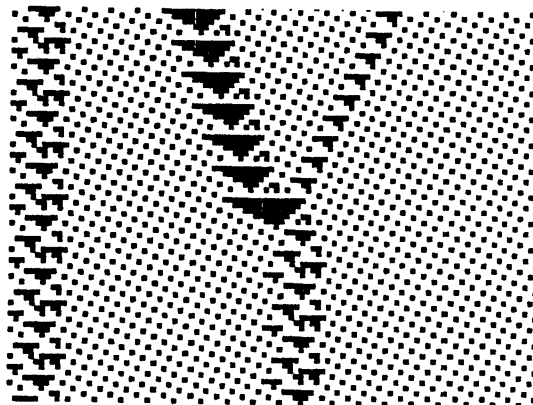
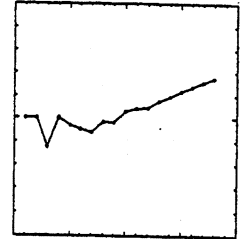
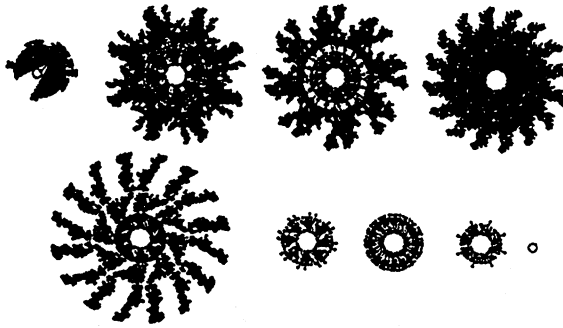
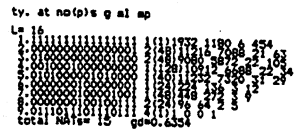
K5 rule = c3 bc e2 84  
 $\lambda_{ratio} = 0.9375$   
Z = 0.5703125

Complexity in One-D Cellular Automata

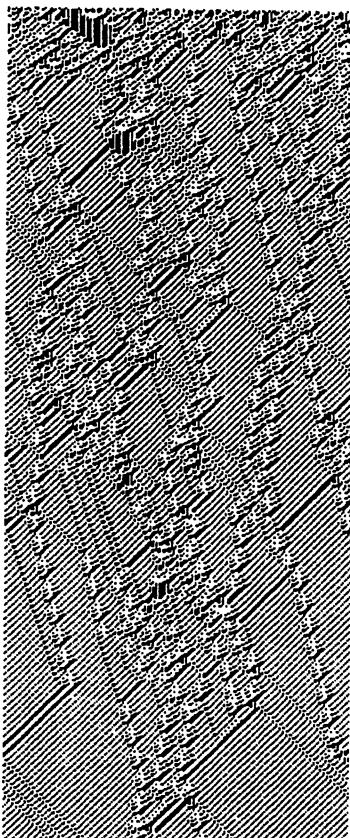
Appendix 2.17



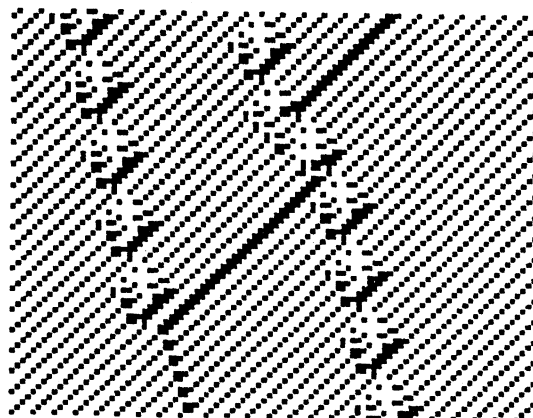
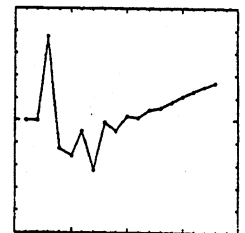
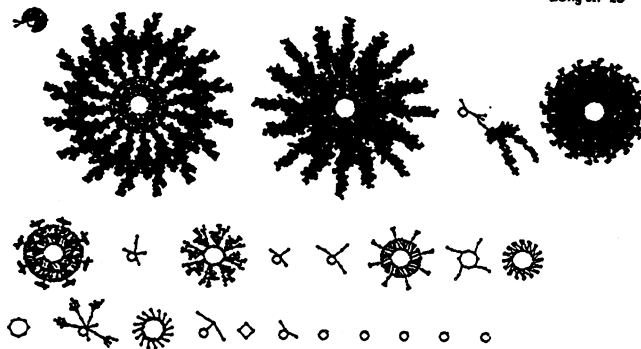
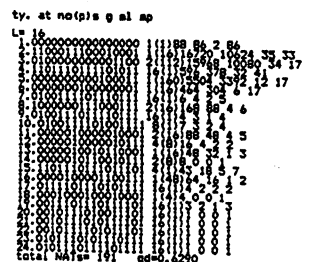
1118000018881001-0111180000000001-rule 3767105537  
Length=16



K5 rule = e0 89 78 01  
 $\lambda_{ratio} = 0.6875$   
 $Z = 0.5703125$



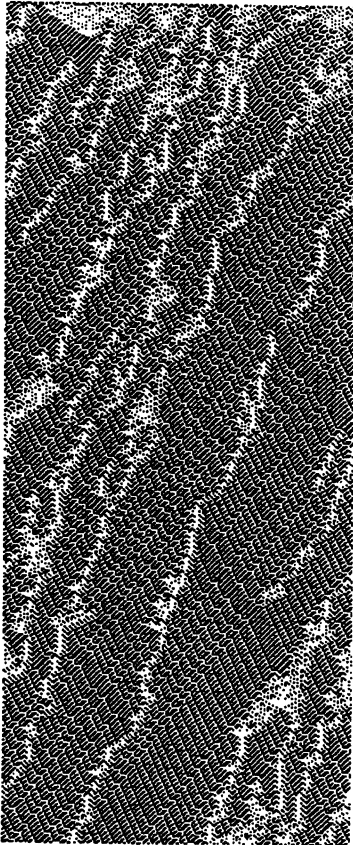
0118000010110001-1100010010101110-rule 1622262958  
Length=16



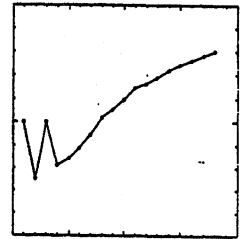
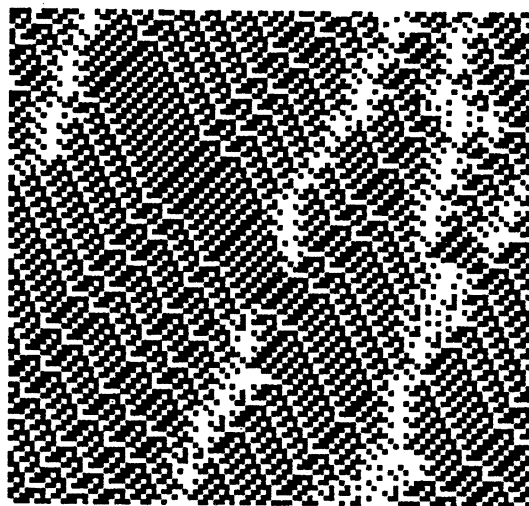
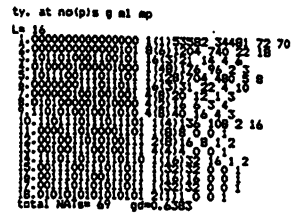
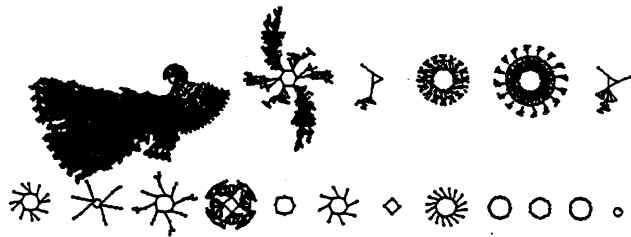
K5 rule = 60 b1 c4 ac  
 $\lambda_{ratio} = 0.875$   
 $Z = 0.5625$

Appendix 2.18

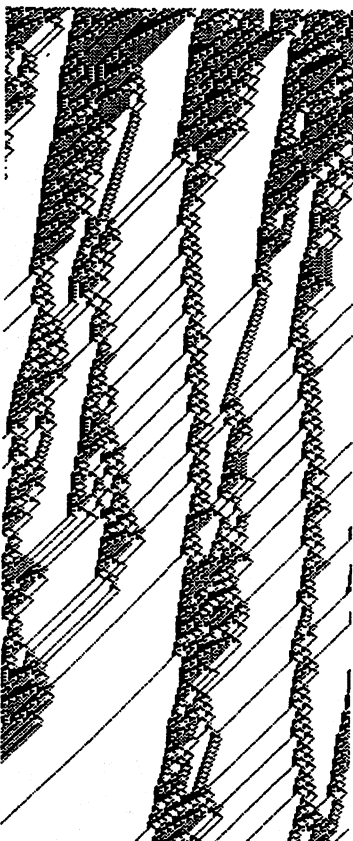
Andrew Wuensche



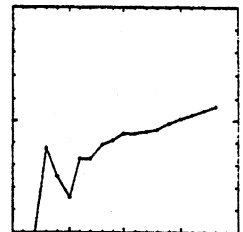
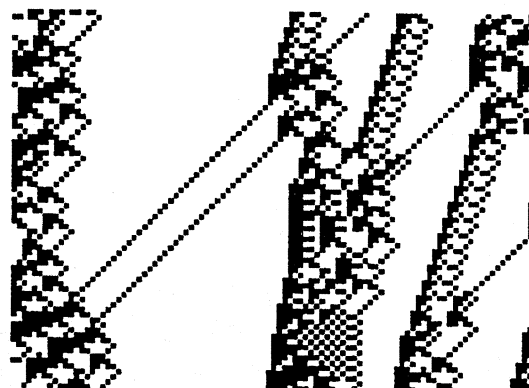
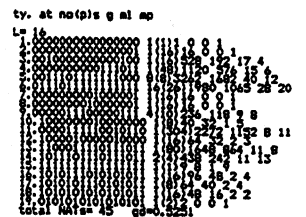
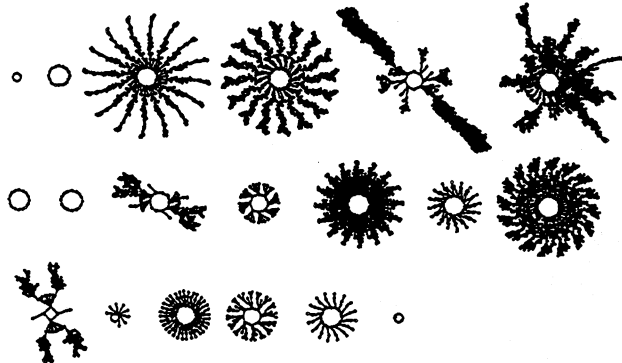
000010111101001-0100101001000010-rule 199838274  
Length=16



K5 rule = 0b e9 4a 42  
 $\lambda_{ratio} = 0.8125$   
 $Z = 0.5625$



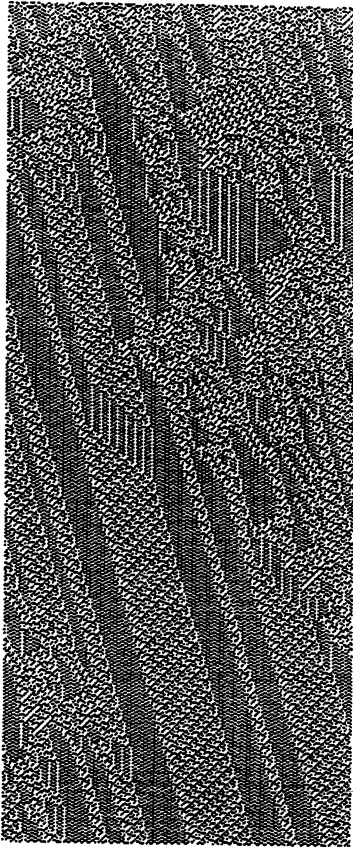
1001011110001110-1100111011100100-rule 2542710692  
Length=16



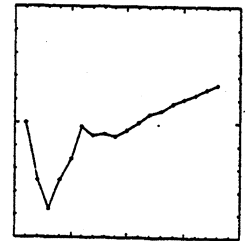
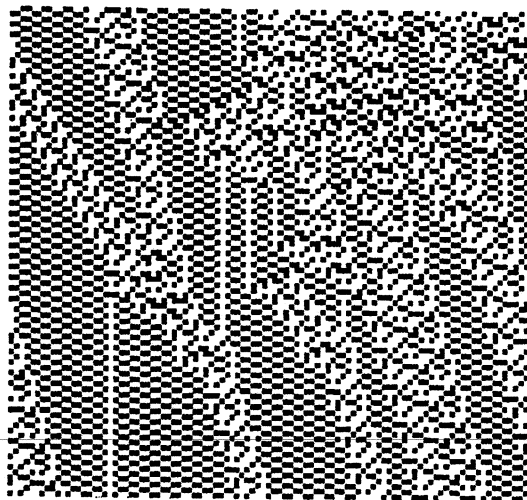
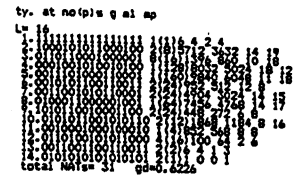
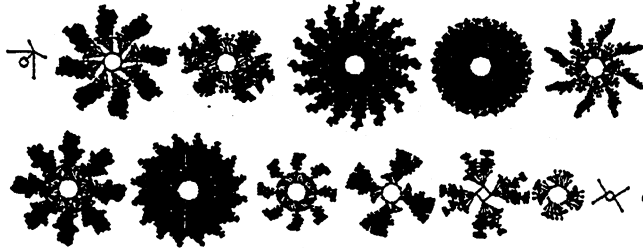
K5 rule = 97 8e ce e4  
 $\lambda_{ratio} = 0.875$   
 $Z = 0.5625$

Complexity in One-D Cellular Automata

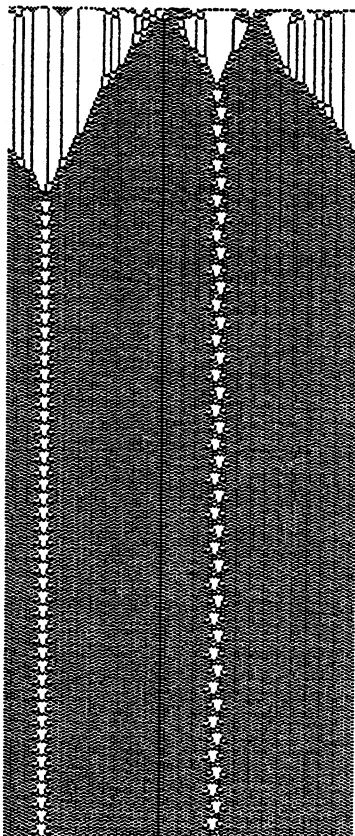
Appendix 2.19



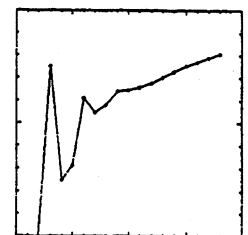
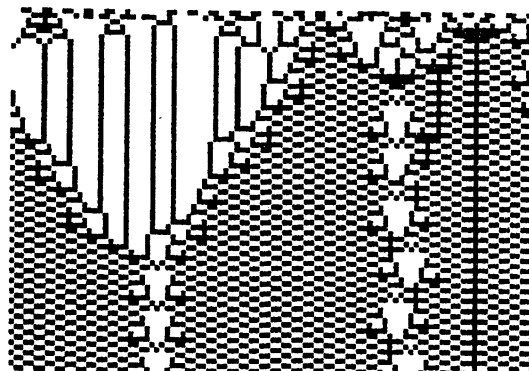
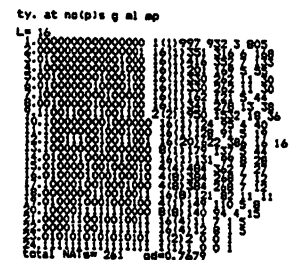
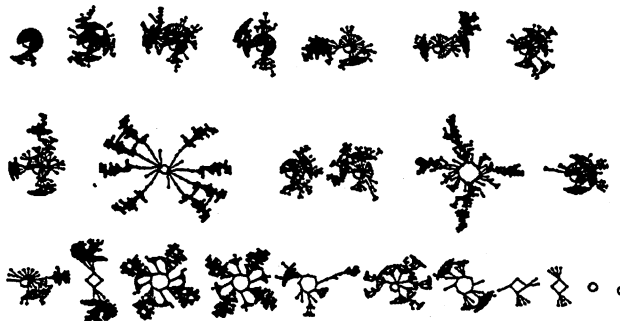
1888811188181181-181888188881181-rule 2267914765  
Length=16



K5 rule = 97 2d a2 0d  
 $\lambda_{ratio} = 0.875$   
 $Z = 0.53125$



18888118881188-88881188811888-rule 2198683288  
Length=16



K5 rule = 83 0c 06 18  
 $\lambda_{ratio} = 0.5625$   
 $Z = 0.484375$

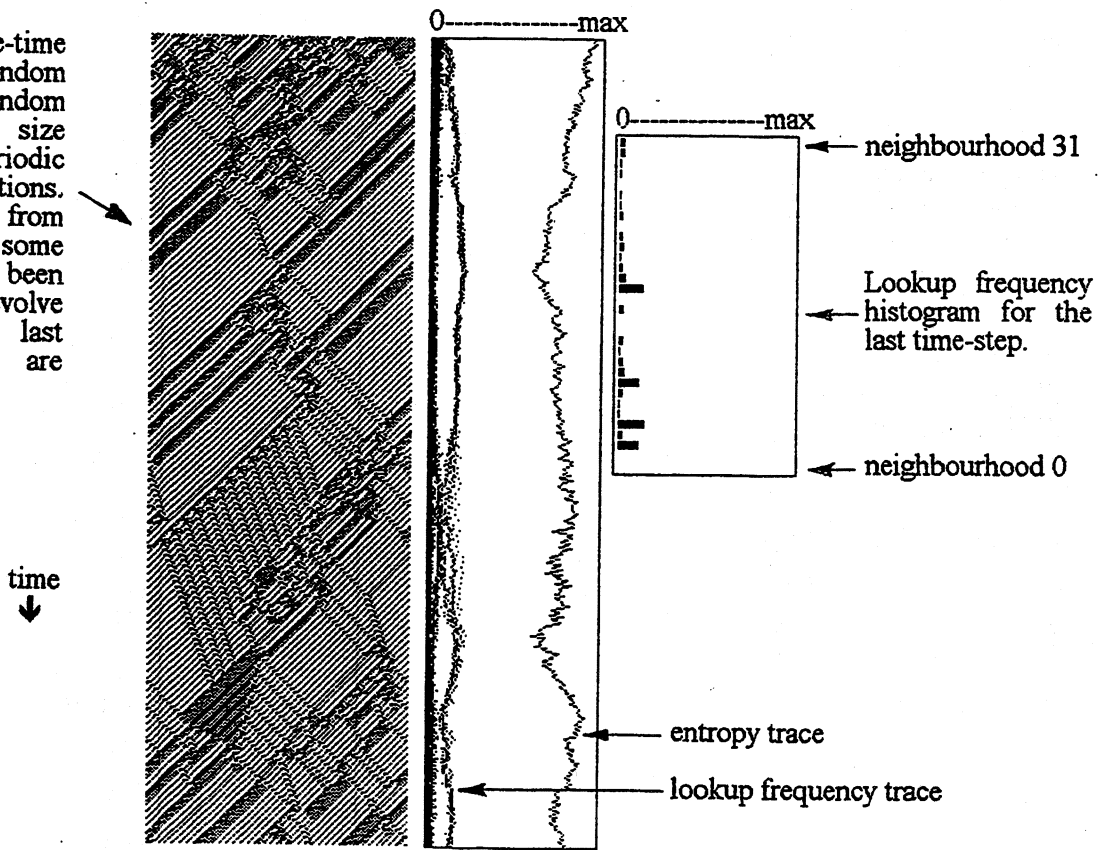
### Appendix 3 - Sample of complex rules, $K=5,6,7$ .

#### Space-time patterns, neighbourhood lookup frequency and entropy

The frequency that each of the  $K$  neighbourhoods in the rule-table is "looked up" at a given time-step can be represented by a histogram, or lookup spectrum. A typical lookup histogram (of the very last time-step) is shown, which distributes the total of  $N$  lookups among the  $2^K$  neighbourhoods, (where  $N$ =system size). For each rule in the sample, appendix 3 shows a typical space-time pattern, with a *superimposed* frequency spectrum ( $2^K$  points corresponding to the histogram values) plotted alongside each time-step. The entropy of the lookup frequency spectrum is also shown for each time step on the same plot.

A typical layout from appendix 3 is annotated below.

A typical space-time pattern from a random seed (or random block), system size 150 with periodic boundary conditions, 460 time-steps from the top down. In some cases the CA has been allowed to evolve further and the last 460 time-steps are shown.



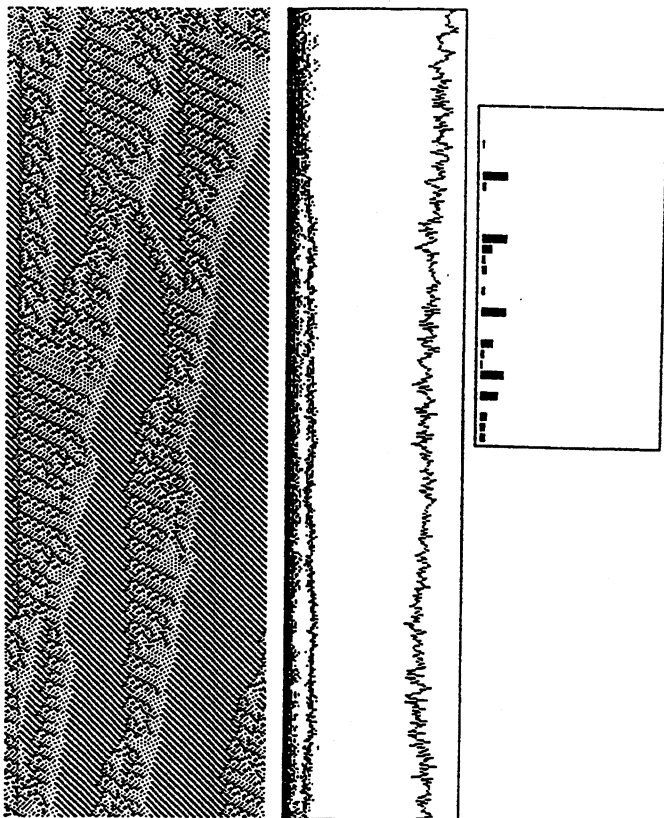
K5 rule = 46 c5 9c af,  $\lambda$  ratio = 0.938,  $Z = 0.671875$

The rule number in hex, the  $\lambda$  ratio and the  $Z$  parameter.

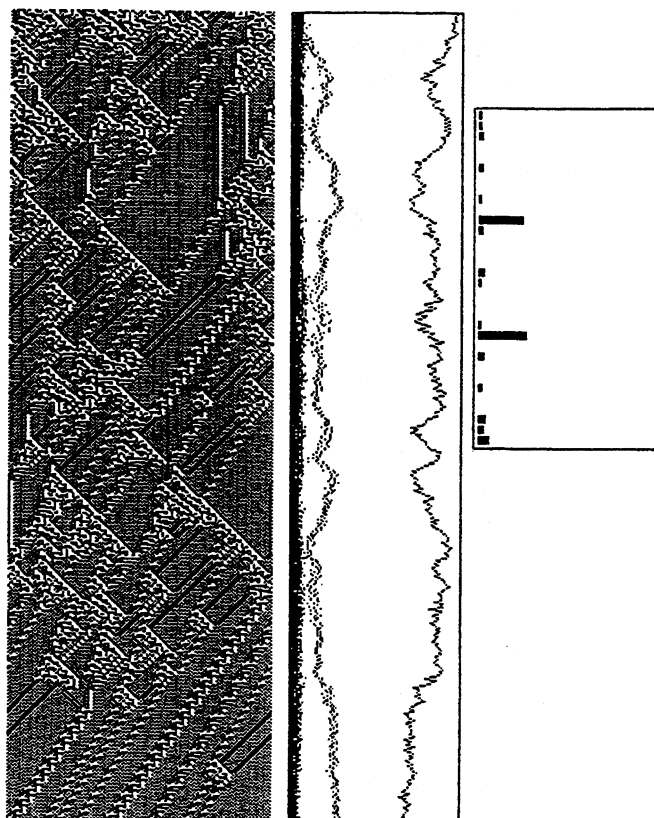


Appendix 3.2

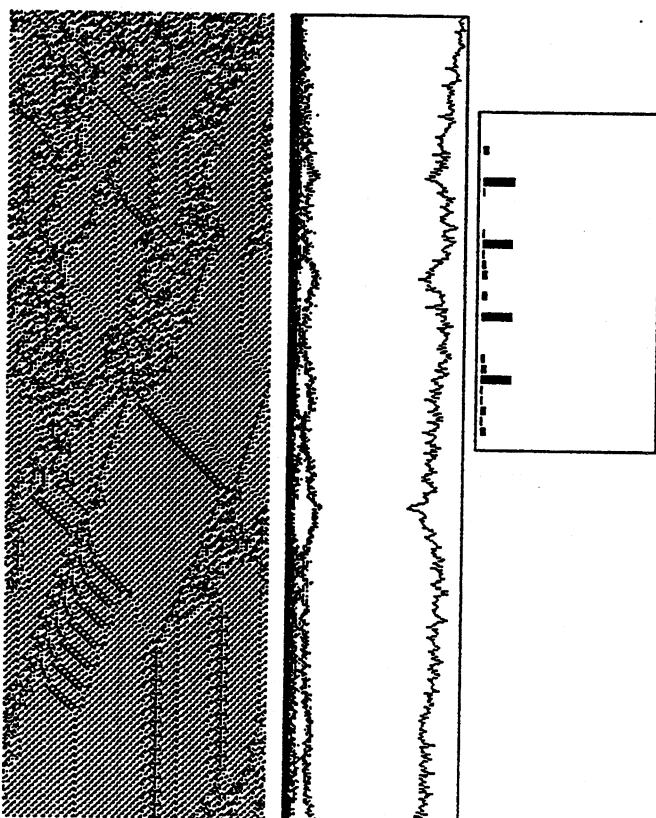
Andrew Wuensche



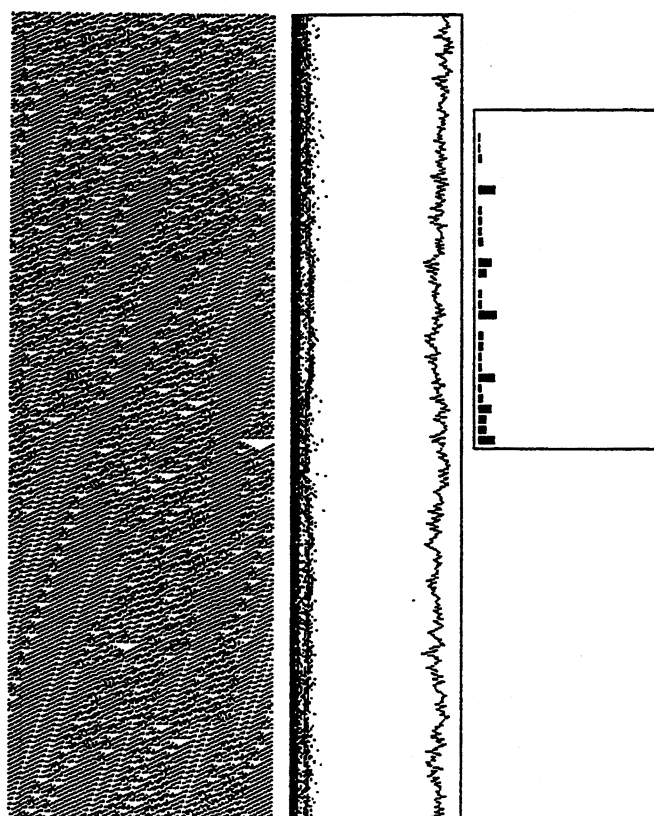
K5 rule = 82 26 dc 23,  $\lambda$  ratio = 0.812, Z = 0.578125



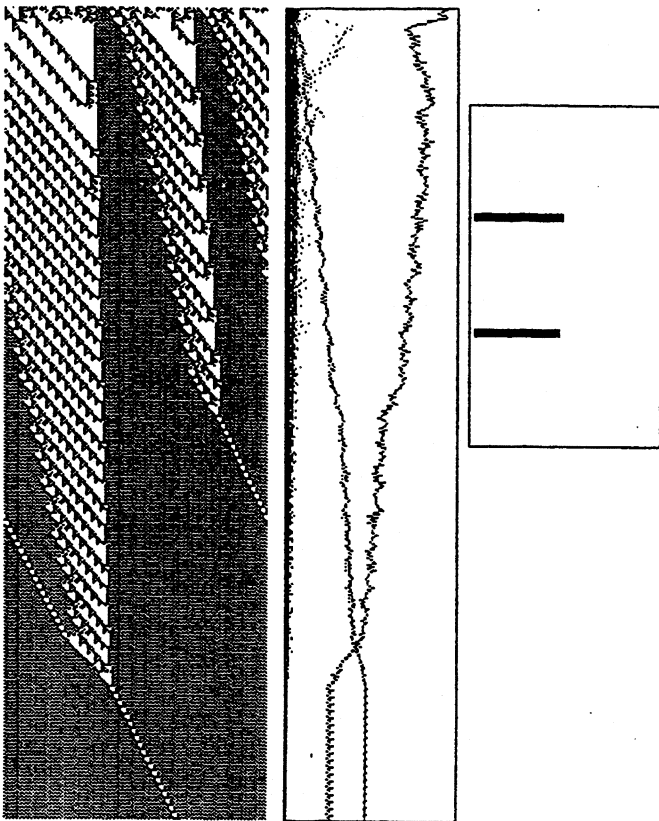
K5 rule = 5d 91 ac 17,  $\lambda$  ratio = 1, Z = 0.671875



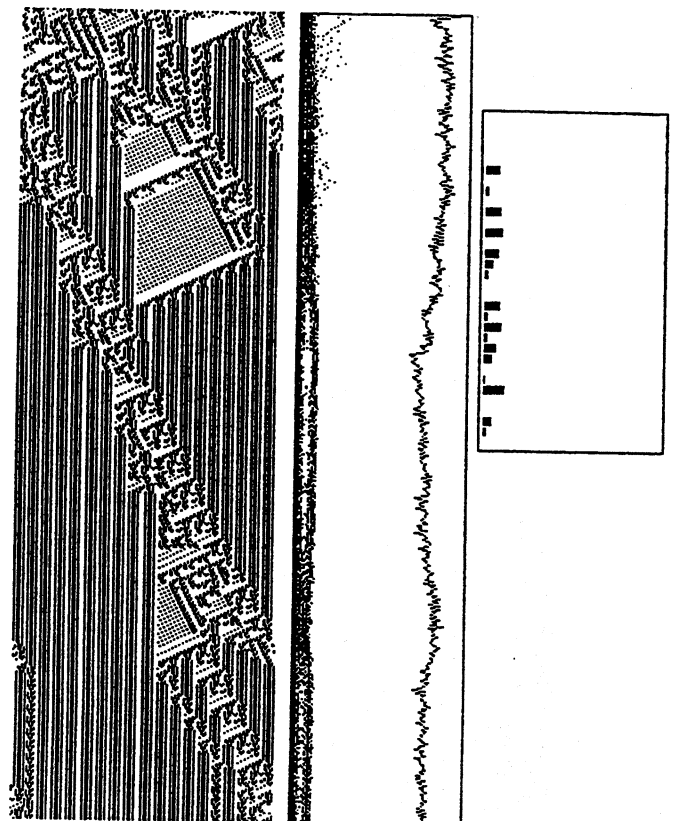
K5 rule = 98 8d 66 3c,  $\lambda$  ratio = 0.938, Z = 0.6875



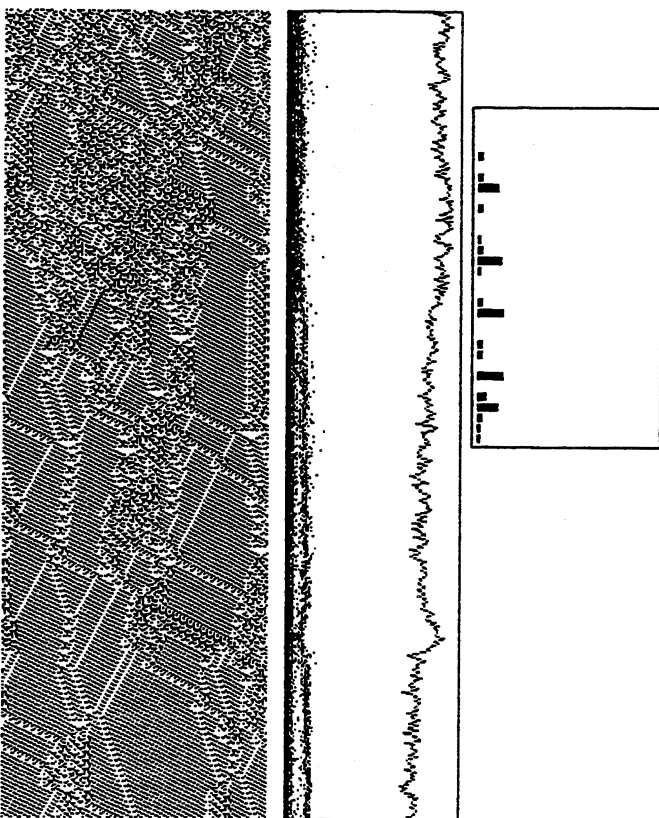
K5 rule = 44 2b 8a 3e,  $\lambda$  ratio = 0.875, Z = 0.5625



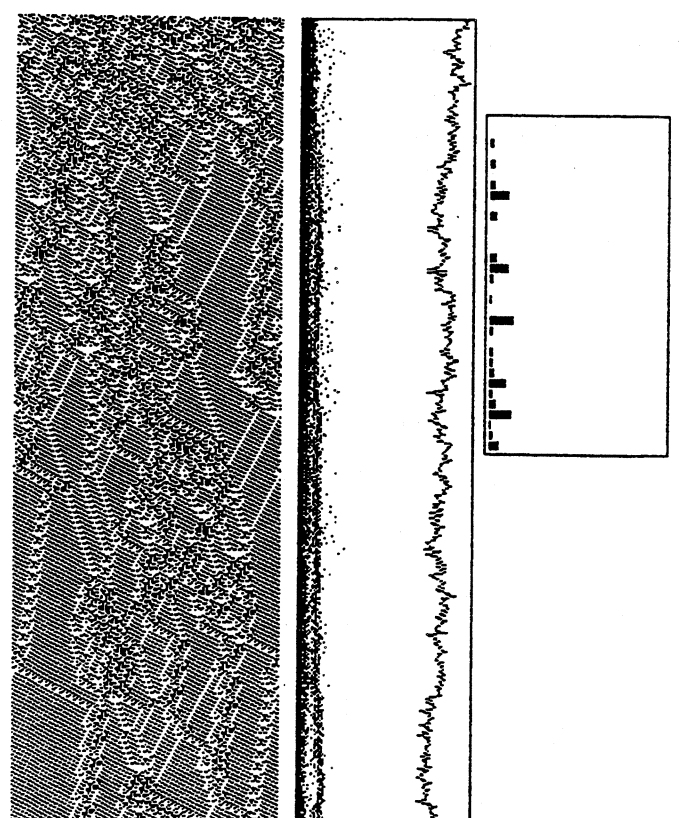
K5 rule = 07 1c dc e0,  $\lambda$  ratio = 0.875, Z = 0.75



K5 rule = c1 d0 64 22,  $\lambda$  ratio = 0.688, Z = 0.5625



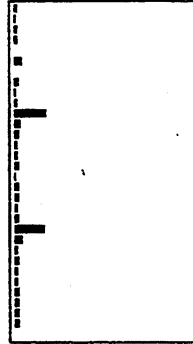
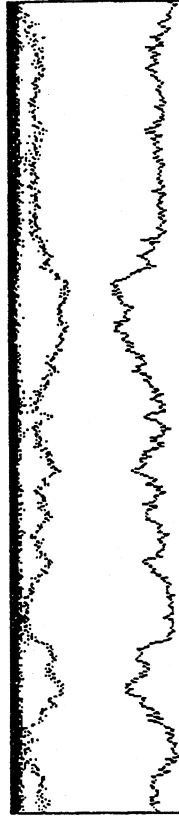
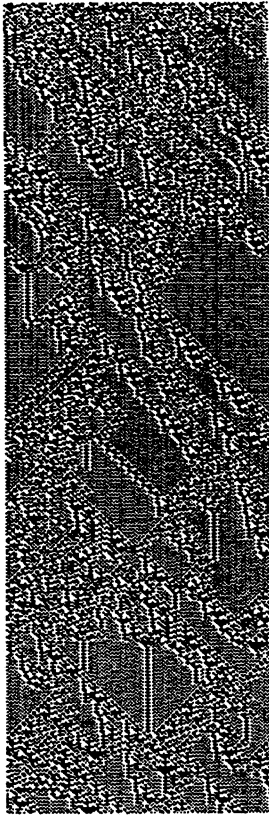
K5 rule = a1 1b c1 26,  $\lambda$  ratio = 0.812, Z = 0.617188



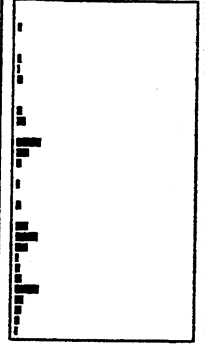
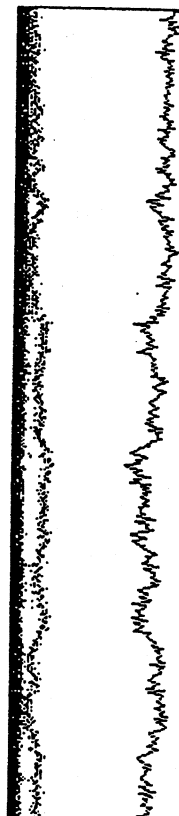
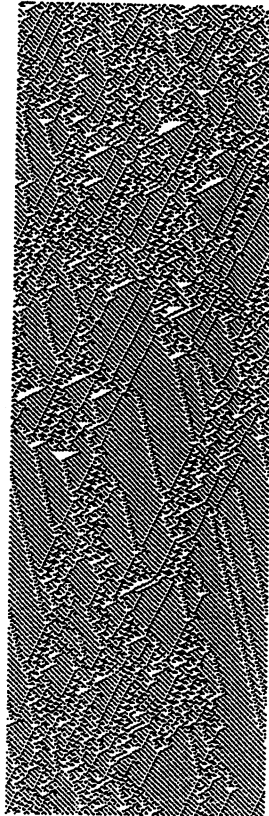
K5 rule = a1 9b c1 26,  $\lambda$  ratio = 0.875, Z = 0.671675

Appendix 3.4

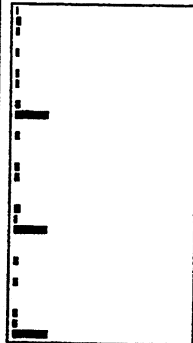
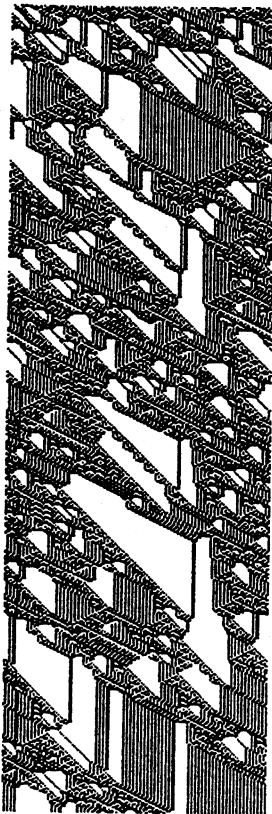
Andrew Wuensche



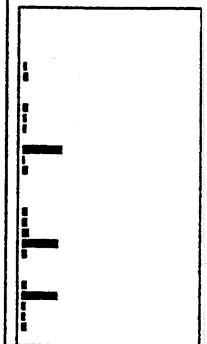
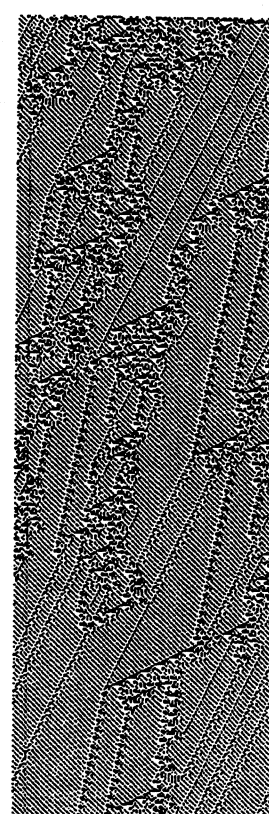
K5 rule = 9a 4c f7 17,  $\lambda$  ratio = 0.875,  $Z = 0.671875$



K5 rule = 0e d0 83 0e,  $\lambda$  ratio = 0.75,  $Z = 0.625$

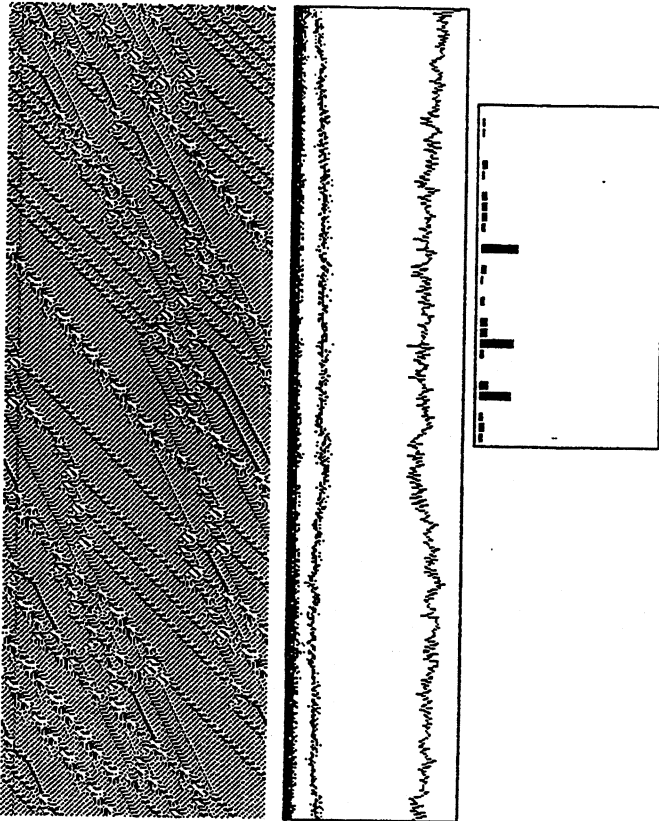


K5 rule = 6e ee 93 60,  $\lambda$  ratio = 0.938,  $Z = 0.6875$

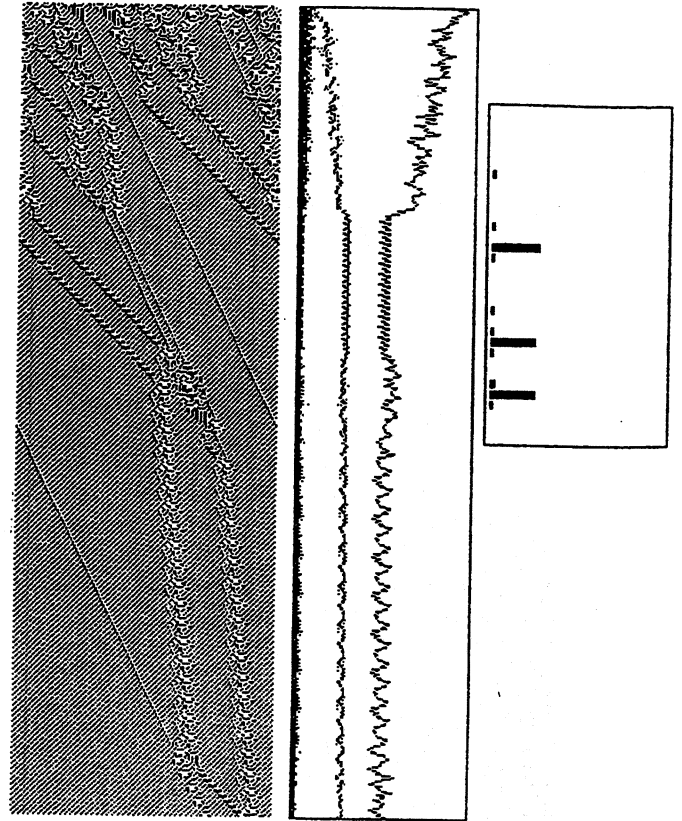


K5 rule = 0f 68 63 46,  $\lambda$  ratio = 0.875,  $Z = 0.671875$

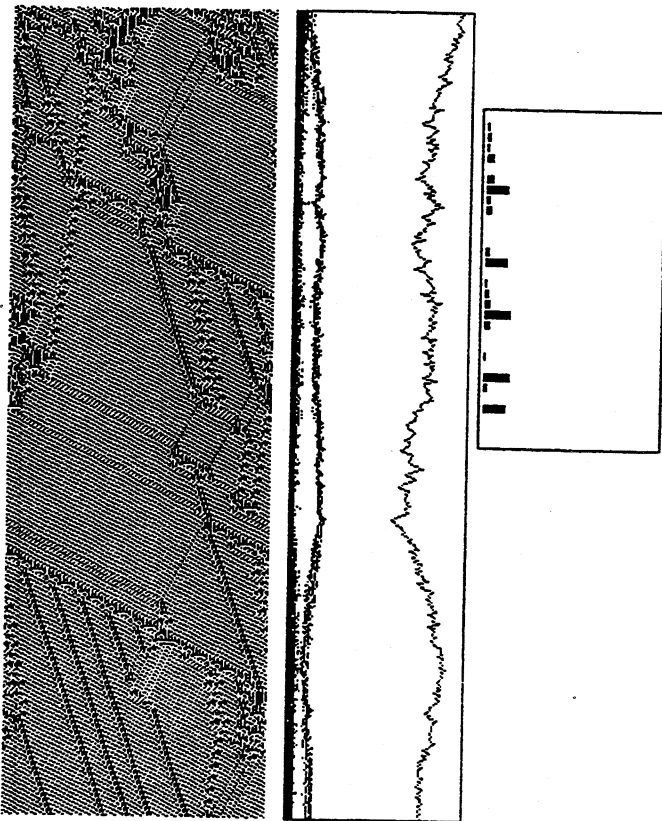




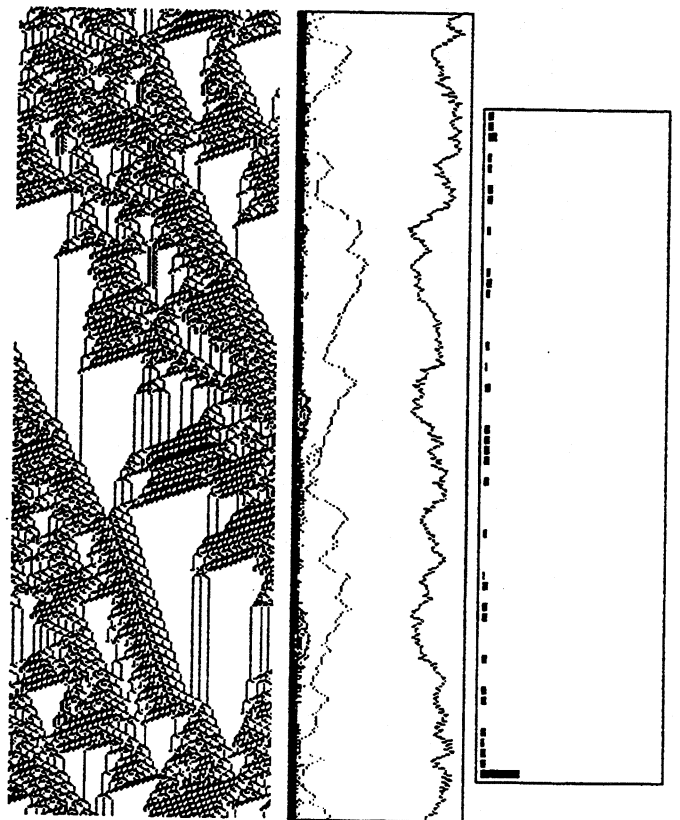
K5 rule = 89 ed 71 06,  $\lambda$  ratio = 0.938,  $Z = 0.726562$



K5 rule = 88 6d 71 06,  $\lambda$  ratio = 0.812,  $Z = 0.726562$



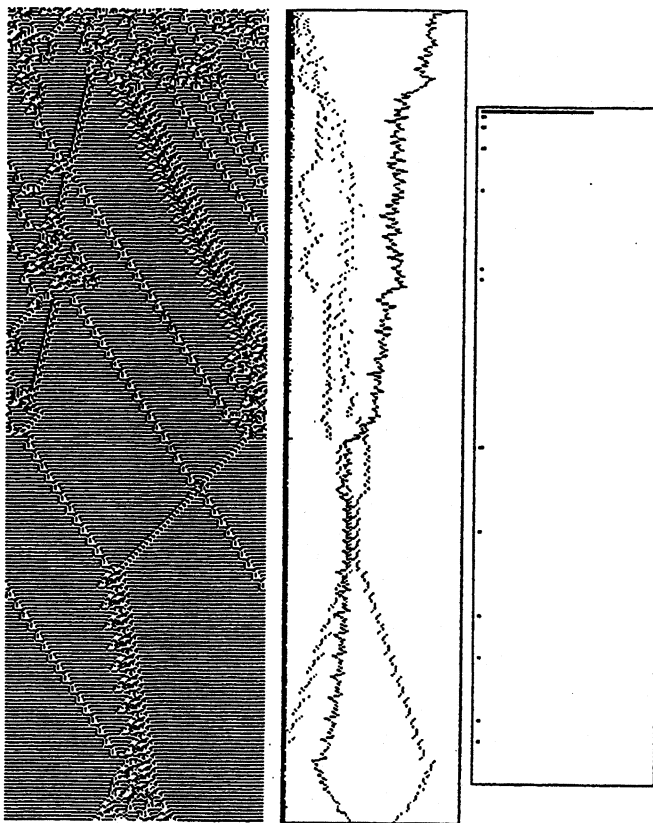
K5 rule = 63 be a5 83,  $\lambda$  ratio = 0.938,  $Z = 0.671875$



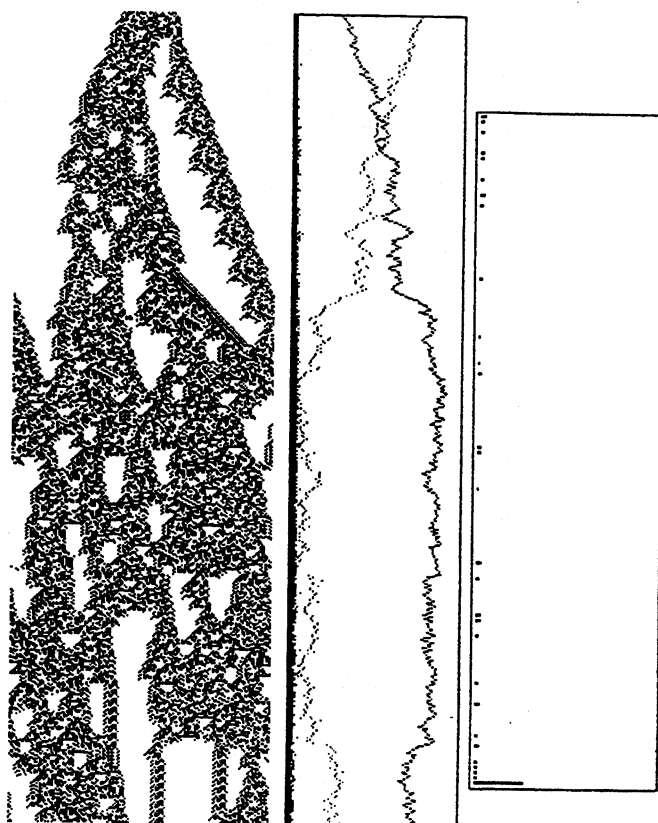
K6 rule = b3 87 b6 b8 8c d4 af a0  
 $\lambda$  ratio = 0.938,  $Z = 0.726562$

Appendix 3.6

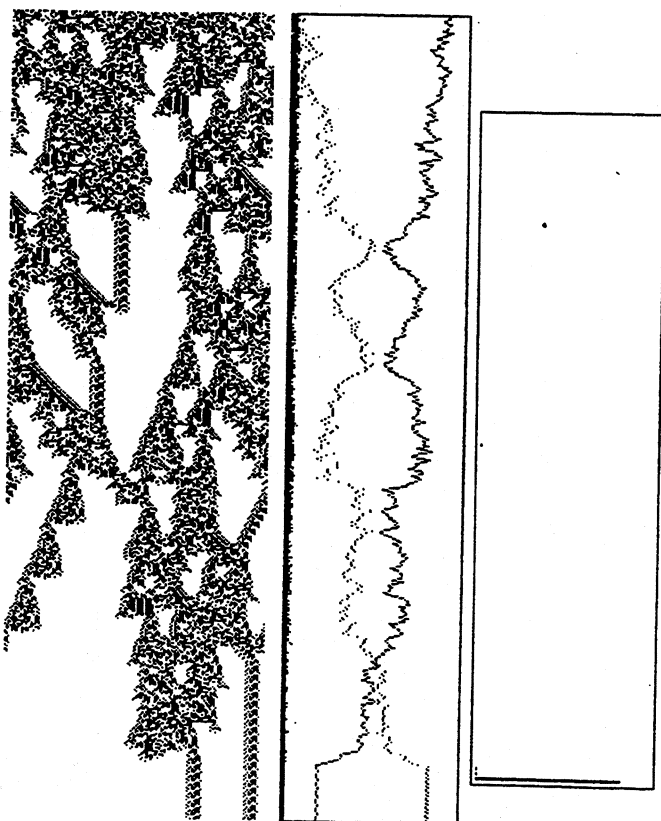
Andrew Wuensche



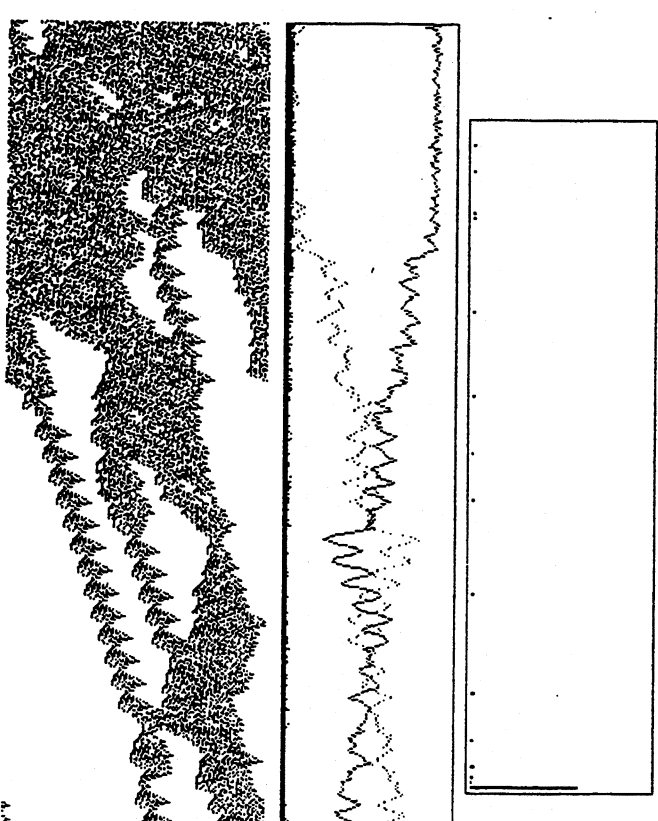
K7 rule = 16 42 44 f5 2d 01 de c7 0e de 02 4b 76 0b e4 ff  
 $\lambda$  ratio = 0.984, Z = 0.521484



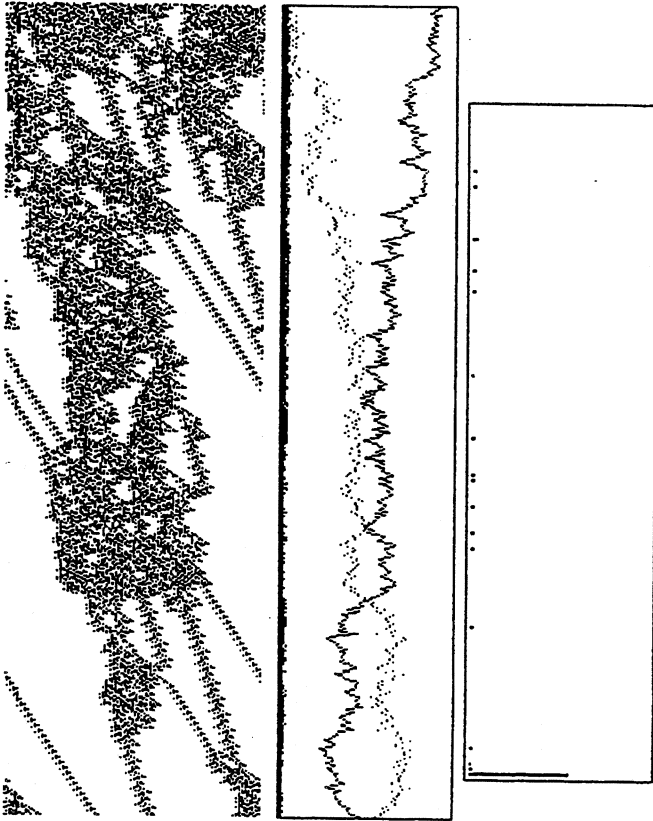
K7 rule = 3b 46 9c 0c e4 f7 fa 96 b9 3b 4d 3a b8 9e c0 e0  
 $\lambda$  ratio = 0.958, Z = 0.591309



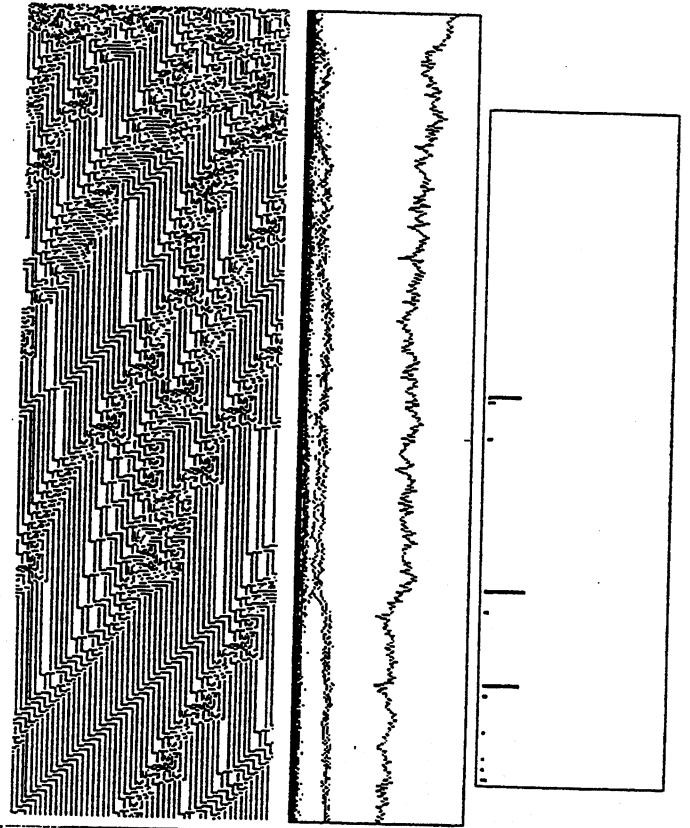
K7 rule = 3b 46 9c 0e e4 f7 fa 96 f9 3b 4d 32 b0 9e d0 e0  
 $\lambda$  ratio = 0.938, Z = 0.578125



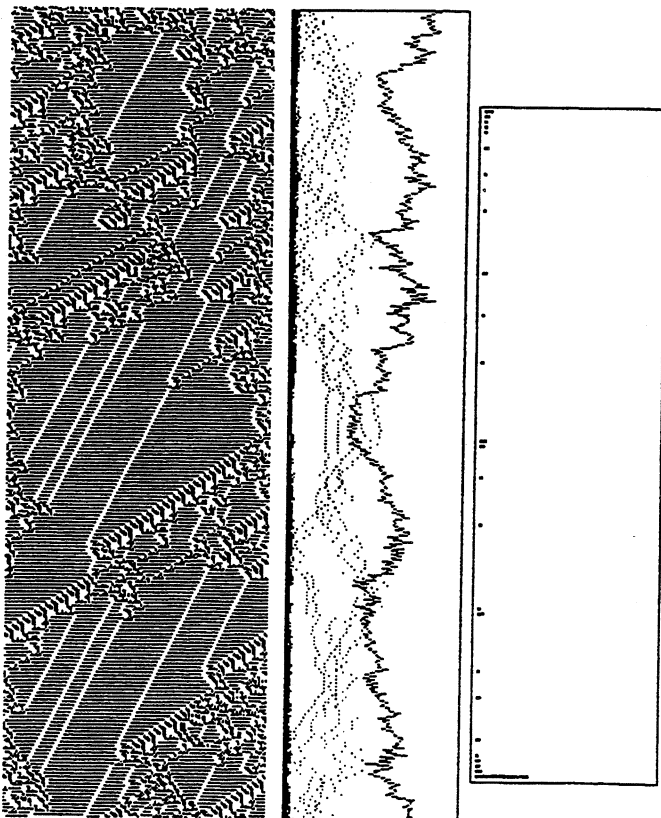
K7 rule = ef 80 af 6e 25 32 62 a6 2a 9f 98 b1 88 89 fb 40  
 $\lambda$  ratio = 0.958, Z = 0.663086



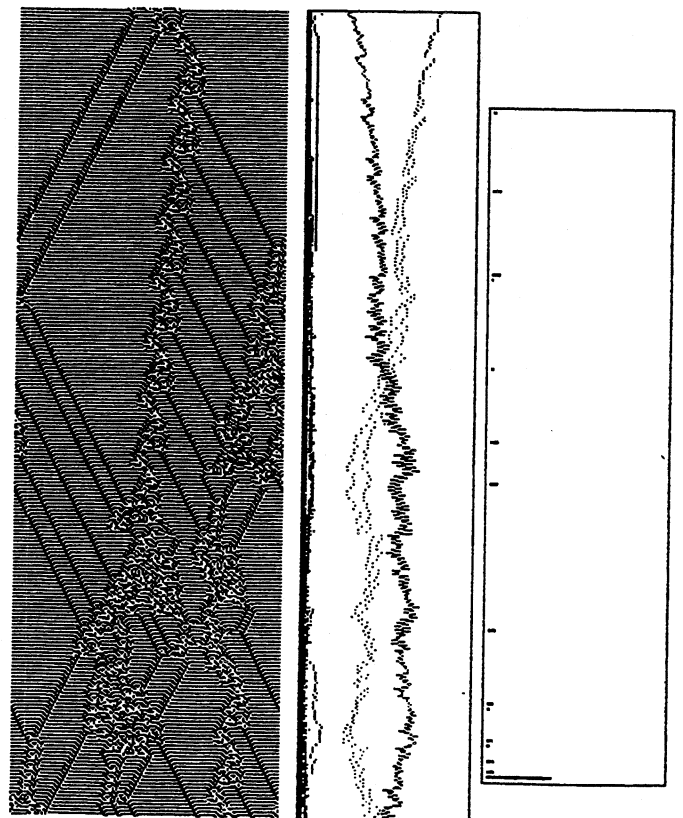
*K7* rule = ed 88 27 ee 25 32 22 a6 2a 1f 9c 53 8c 89 fb 48  
 $\lambda$  ratio = 0.953,  $Z = 0.682129$



*K7* rule = 2e fb df e2 2d aa 67 f0 45 c1 4b 08 24 a2 2e 2b  
 $\lambda$  ratio = 0.906,  $Z = 0.517517$



*K7* rule = 00 1e 17 2d 67 1e 5f 21 3f 10 07 ff 37 be 07 21  
 $\lambda$  ratio = 0.938,  $Z = 0.384766$

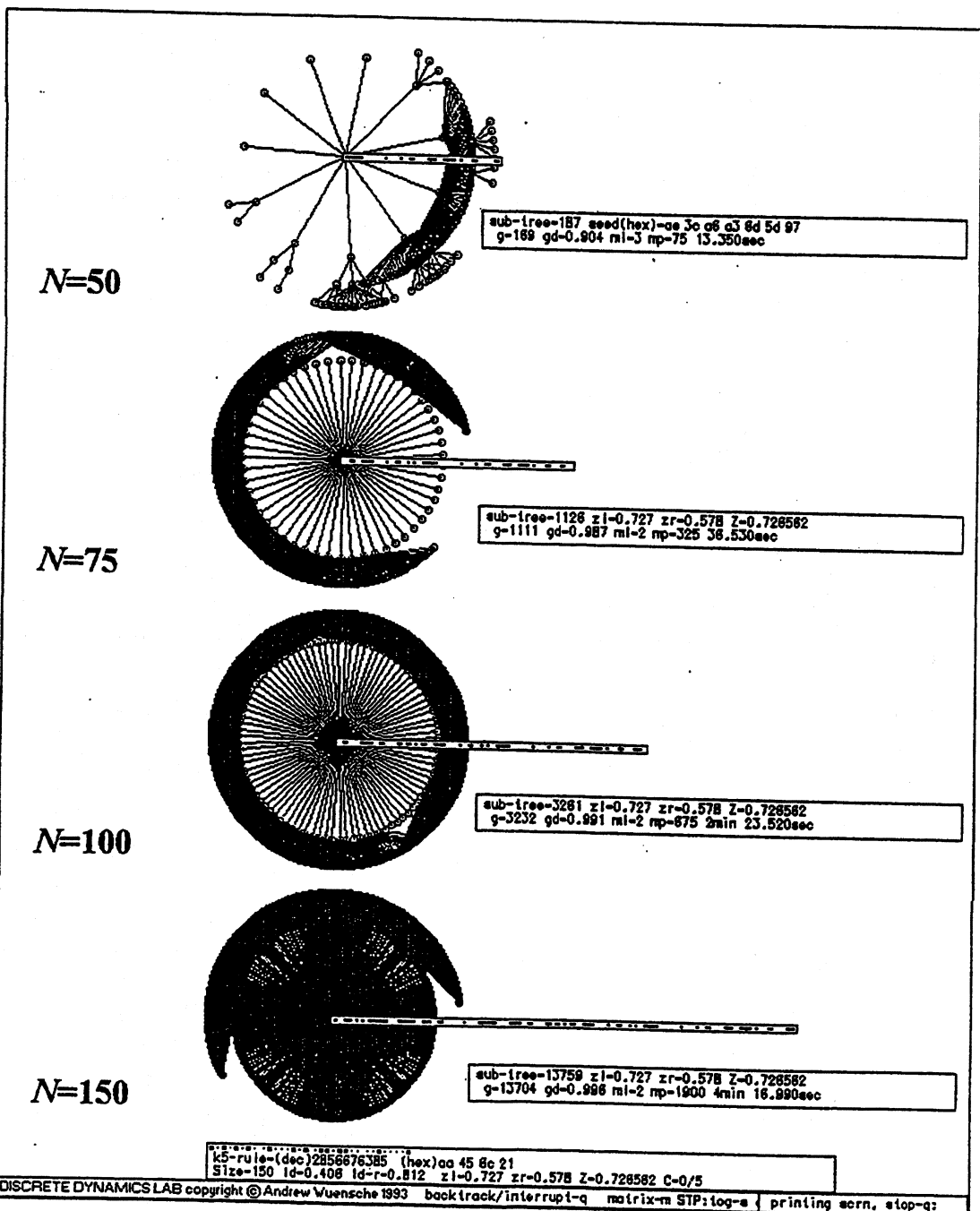


*K7* rule = 0c f7 8a cf 30 4b 06 4f 24 fb 1e 8f bf 7d 4f df  
 $\lambda$  ratio = 0.844,  $Z = 0.384766$

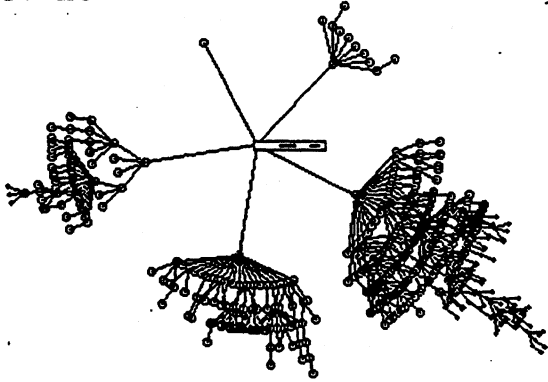
**Appendix 4 - Subtrees for large lattice sizes,  $N=20$  to 150** appendix 4.1

Subtrees for the glider  $K=5$  rule *aa 45 6c 2l* are shown, together with data on G-density (gd), maximum levels (ml) and maximum in-degree (mp). The time to generate the subtree is also shown. The rule is the same as shown in figures 1d and 18, and is also the negative equivalent of rule *7b c9 5d aa* in appendix 2.5. The same seed, or seed segment, was iterated forward by a varying number of steps, and the subtree was generated from the resulting root state, which is shown graphically in each subtree. This is necessary because a random state is very likely to be a garden-of-Eden state, having no subtree.

This page shows subtrees for  $N = 50, 75, 100,$  and  $150$ . The CA was iterated forward by 2 steps before running backward. The following page shows subtrees for  $N = 20, 25$  and  $30$  (iterated 4 steps forward), and  $N = 35, 40$  and  $45$  (iterated 3 steps forward).

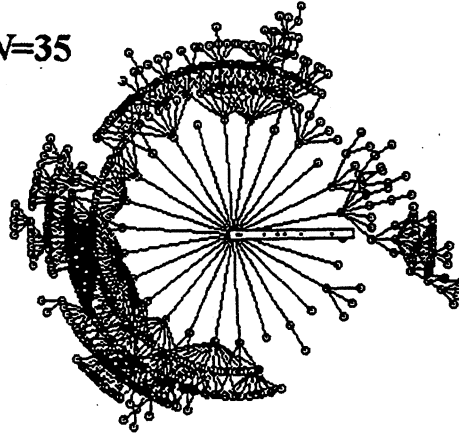


N=20



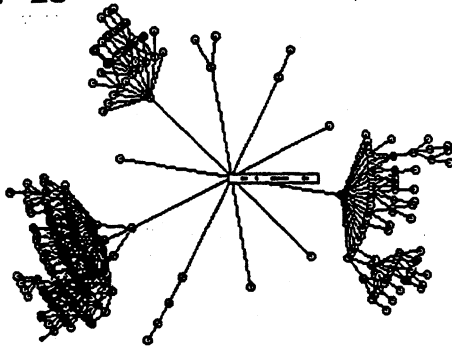
sub-tree-485 seed(hex)=6d 5d 97  
 g-301 gd-0.821 mi-18 np-19 5.020sec

N=35



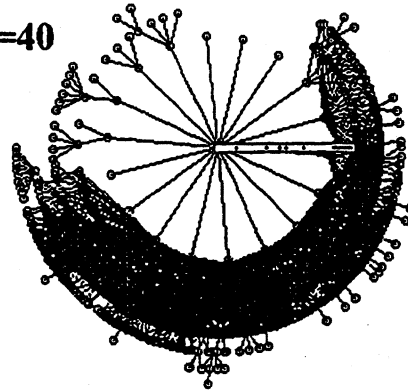
sub-tree-772 seed(hex)=a8 a3 6d 5d 97  
 g-587 gd-0.734 mi-7 np-30 19.830sec

N=25



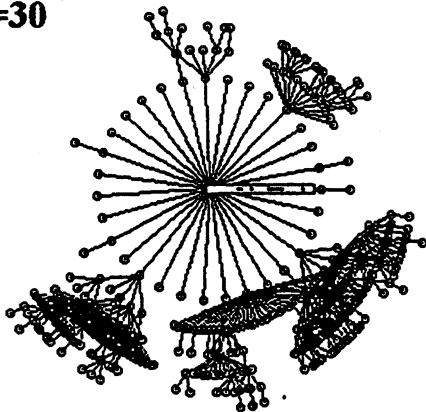
sub-tree-373 seed(hex)=a3 6d 5d 97  
 g-239 gd-0.641 mi-8 np-21 7.470sec

N=40



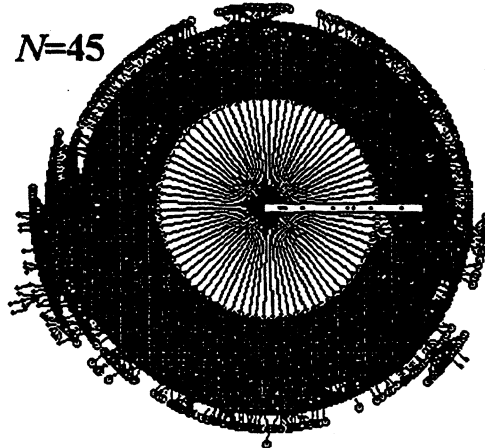
sub-tree-2851 seed(hex)=a8 a3 6d 5d 97  
 g-2237 gd-0.785 mi-7 np-102 42.220sec

N=30



sub-tree-591 seed(hex)=a3 6d 5d 97  
 g-432 gd-0.731 mi-7 np-35 14.340sec

N=45



sub-tree-16510 seed(hex)=3c a8 a3 6d 5d 97  
 g-13721 gd-0.831 mi-10 np-100 10min 41.100sec

ks-rule=(dec)2856678385 (hex)aa 45 6c 21  
 Size-45 ld-0.406 ld-r-0.812 z1-0.727 zr-0.578 z-0.728582 c-0/5

## Appendix 5

### Automatic Rule Sample

Appendix 5 gives some supplementary information on the automatic input-entropy rule samples described in chapter 4.11.

**Appendix 5.1.**        *k=5* rules.

**Appendix 5.2.**        *k=6* rules.

**Appendix 5.3.**        *k=7* rules.

- 5.(1-3).1. Examples of the space-time patterns of complex, ordered and chaotic rules from the automatic sample. This supplements the examples of complex rules in figures 4.6 - 4.8.
- 5.(1-3).2. The rule sample shown as a scatter plot of mean input-entropy against standard deviation, and as a 2d frequency histogram in normal and log form.
- 5.(1-3).3. The rule sample shown as a scatter plot of the  $Z$  parameter against standard deviation, and as a 2d frequency histogram in normal and log form.
- 5.(1-3).4. The rule sample shown as a scatter plot of  $\lambda_{ratio}$  against standard deviation, and as a 2d frequency histogram in normal and log form.
- 5.(1-3).5. Examples of complex, chaotic and ordered rules from the sorted rule sample list.

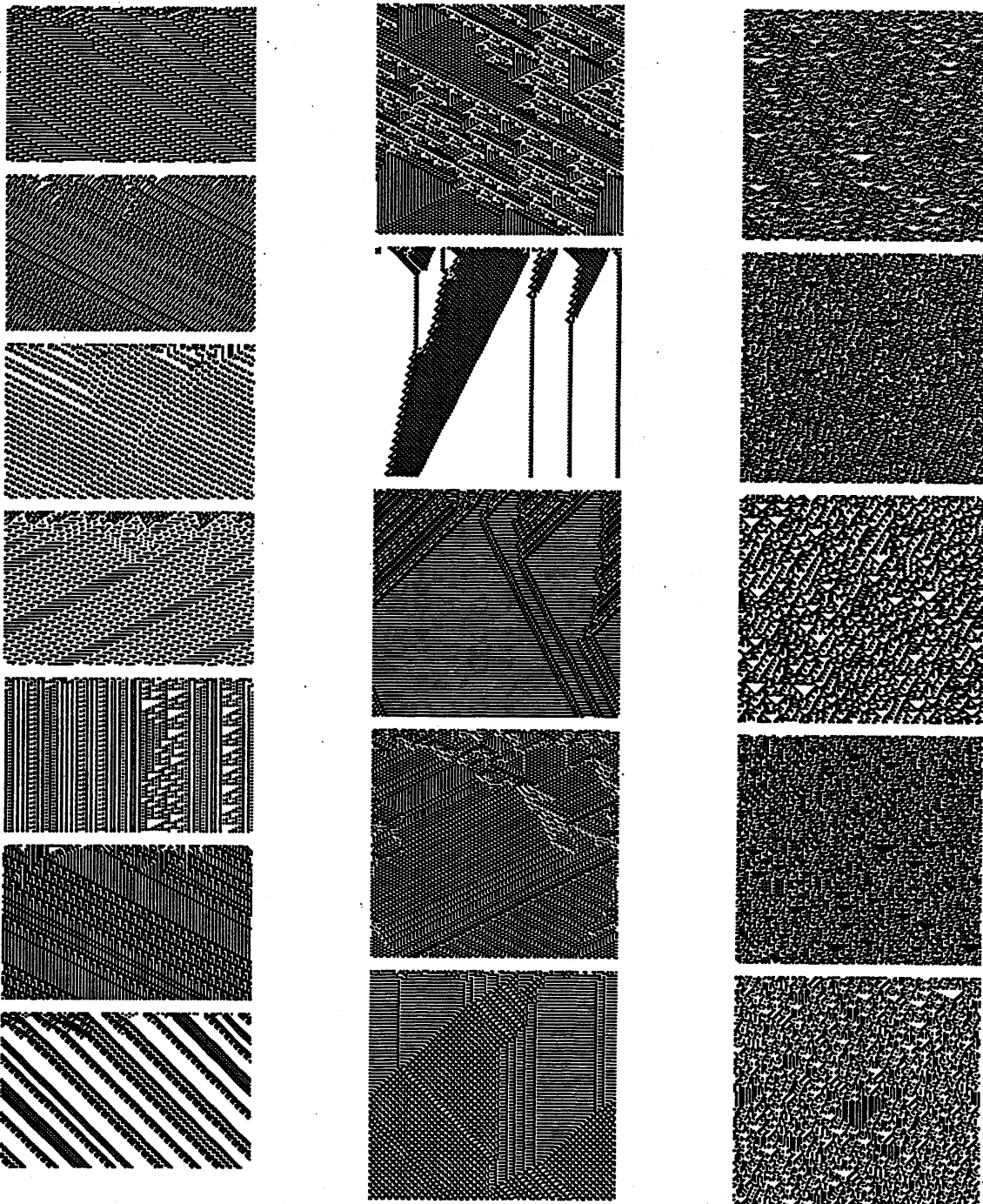
### Appendix 5.1.1. ( $k=5$ )

Examples of the space-time patterns of ordered, complex and chaotic rules from the automatic sample. This supplements the examples of complex rules in figures 4.4.  $n=150$ . The rule numbers can be found by their rule index, they are listed in this appendix 5.1.5.

*left:* ordered rules, 100 time steps from a random initial state, index 15001-15007.

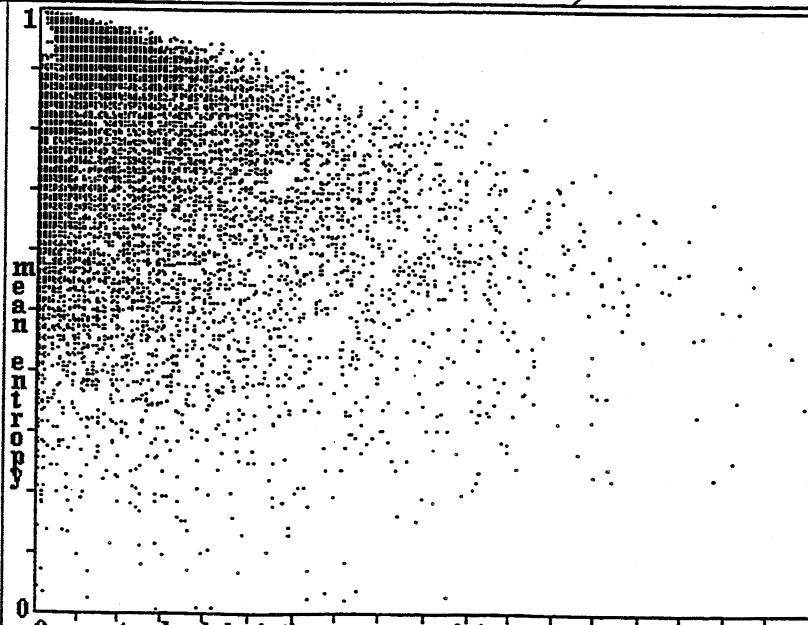
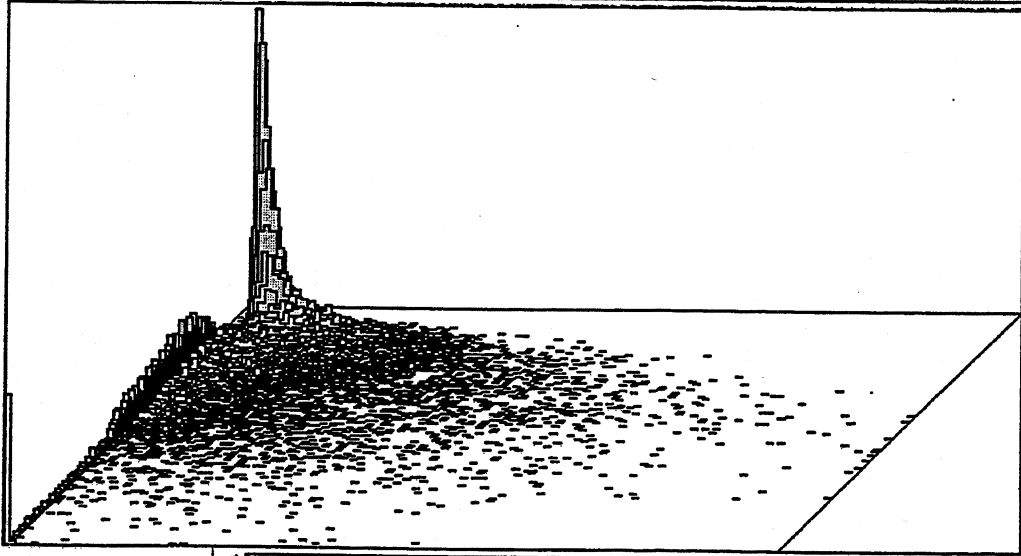
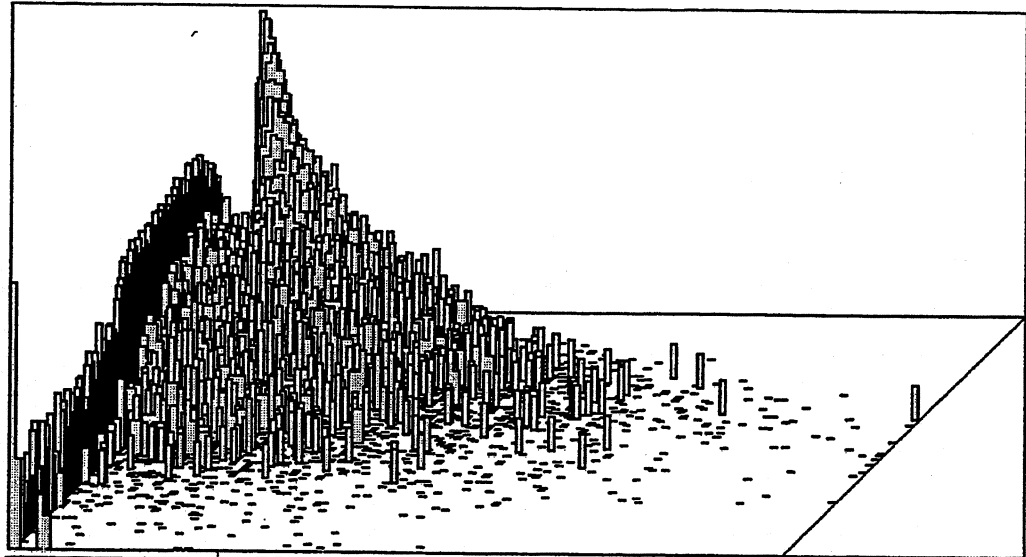
*centre:* complex rules, 140 time steps from a random initial state, index 22-26.

*right:* chaotic rules, 140 time steps from a random initial state, index 12001-12005.



**Appendix 5.1.2. ( $k=5$ )**

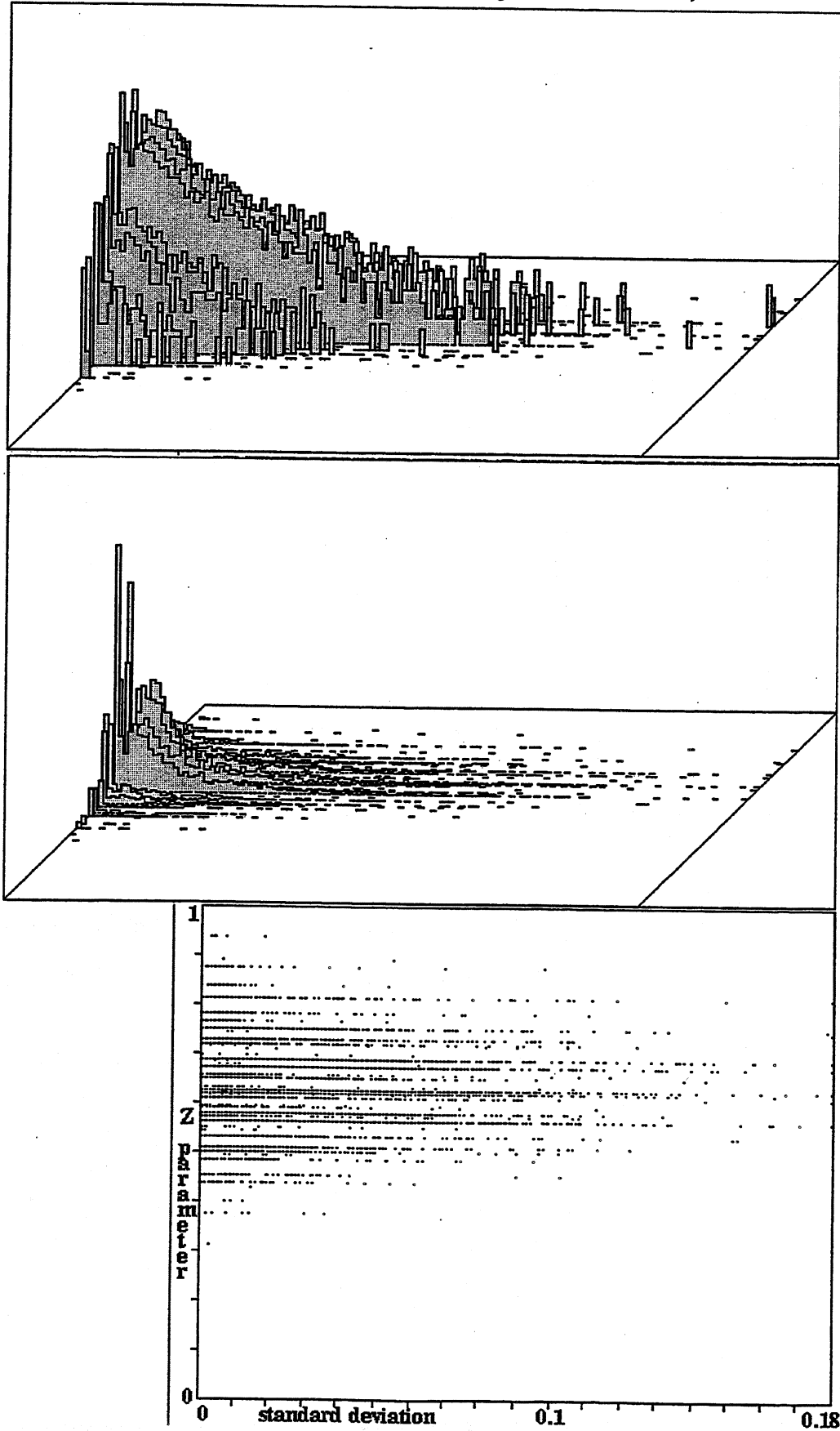
*Below:* Mean entropy ( $y$ -axis) plotted against standard deviation for a random sample of 17680  $k=5$  rules,  $n=150$  (see section 4.11, and figure 4.23.). *Centre:* The frequency distribution of points falling within a  $128 \times 128$  grid. *Top:* the frequency shown in log form.





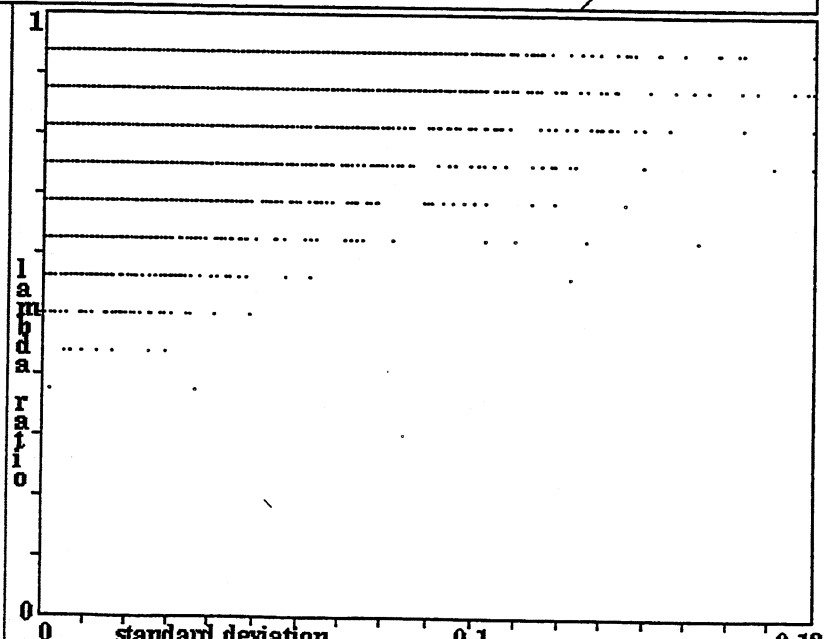
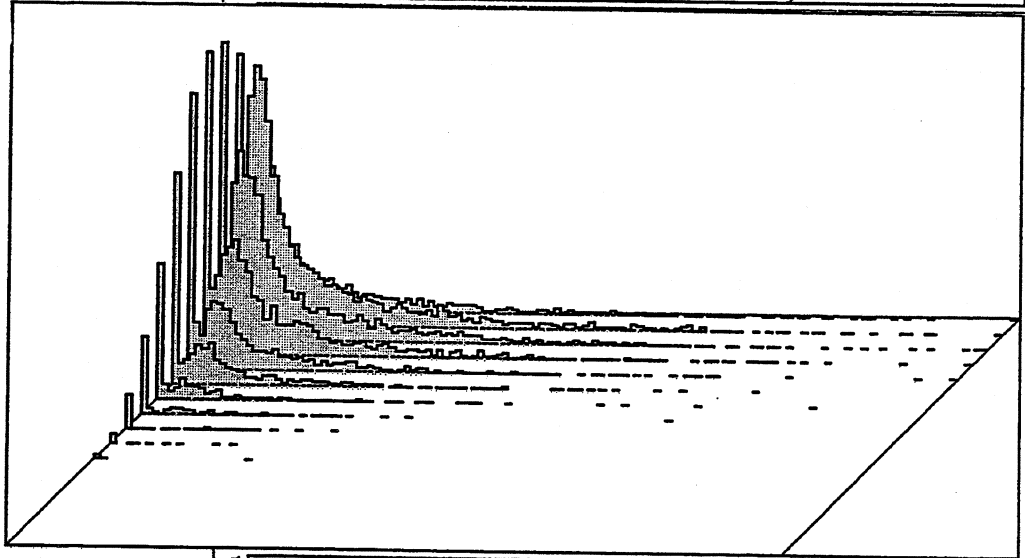
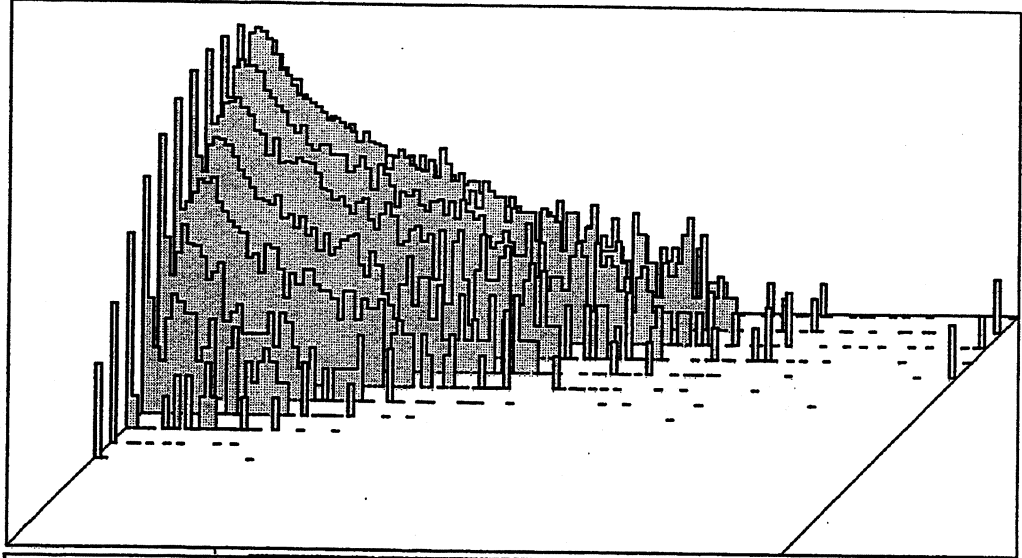
**Appendix 5.1.3. ( $k=5$ )**

*Below:* The  $Z$  parameter ( $y$ -axis) plotted against standard deviation for a random sample of 17680  $k=5$  rules,  $n=150$  (see section 4.12). *Centre:* The frequency distribution of points falling within a  $128 \times 128$  grid. *Top:* the frequency shown in log form. High  $Z$  indicates chaos, low  $Z$  order.



### Appendix 5.1.4. ( $k=5$ )

*Below:*  $\lambda_{\text{ratio}}$  ( $y$ -axis) plotted against standard deviation for a random sample of 17680  $k=5$  rules,  $n=150$  (see section 4.14). *Centre:* The frequency distribution of points falling within a  $128 \times 128$  grid. *Top:* the frequency shown in log form.



### Appendix 5.1.5. (k=5)

Examples of complex, chaotic and ordered rules from the sorted sample of 17680. The sample is sorted from highest to lowest standard deviation, and by highest to lowest mean entropy for each value of standard deviation. Note that these measures are saved as 8 bits so the resolution is 1/256. *Top*: complex rules, index 1-132. *Bottom left*: chaotic rules, index 12000-12043. *Bottom right*: ordered rules, index 15000-15043.

17680 k=5 rules m-ent s-dev	17680 k=5 rules m-entropy stan-dev	17680 k=5 rules m-entropy stan-dev
1. dd 25 d8 e0 0.573 0.1800	45. 1c 83 bc 3e 0.510 0.1341	89. 1a 03 36 33 0.631 0.1172
2. 5e 99 ad 25 0.573 0.1800	46. fe 36 69 b8 0.239 0.1334	90. 2c 04 31 b0 0.412 0.1158
3. d8 41 46 1e 0.553 0.1800	47. fc 7a 63 94 0.592 0.1334	91. 19 88 ed 95 0.420 0.1158
4. 27 33 78 28 0.478 0.1800	48. 55 a5 27 d7 0.702 0.1334	92. dc 1a 4f 74 0.349 0.1158
5. 19 39 5a 20 0.435 0.1800	49. d4 4e 94 f9 0.588 0.1334	93. 38 76 a0 59 0.439 0.1151
6. 15 a4 dc a8 0.427 0.1800	50. fc 64 ba 95 0.357 0.1327	94. e5 e6 d2 45 0.686 0.1151
7. 9e f6 e1 e0 0.408 0.1800	51. 12 38 47 2b 0.514 0.1320	95. 03 26 6c e4 0.439 0.1151
8. 2c 0a 27 68 0.388 0.1800	52. 21 b2 69 62 0.545 0.1313	96. 4d d4 0c be 0.675 0.1144
9. 4d a2 5e b0 0.357 0.1800	53. 58 c8 51 9c 0.549 0.1313	97. e2 5f a7 a1 0.392 0.1144
10. ff aa 1f 14 0.227 0.1800	54. fc 64 ba 95 0.357 0.1306	98. 5b 5b 08 c5 0.522 0.1144
11. d0 45 af 9f 0.341 0.1786	55. 23 69 6c 0a 0.729 0.1299	99. 3f f6 96 21 0.576 0.1144
12. ab ed a0 b5 0.427 0.1758	56. e5 84 97 8d 0.227 0.1299	100. b2 8d 1d 9d 0.365 0.1136
13. e9 cf d3 17 0.451 0.1708	57. dd 5f 85 00 0.569 0.1292	101. 3b 74 b8 14 0.667 0.1136
14. 52 6b b2 7d 0.545 0.1666	58. d9 fa 91 4c 0.427 0.1292	102. ad 99 7f 67 0.737 0.1136
15. 47 85 2d 7e 0.498 0.1638	59. ed 5d 91 9a 0.318 0.1292	103. 3e c8 b7 b7 0.604 0.1136
16. 3e 87 44 09 0.576 0.1638	60. e8 ea de 78 0.365 0.1292	104. 71 87 2f 51 0.361 0.1129
17. fa 06 bf 98 0.353 0.1631	61. 21 b2 49 62 0.494 0.1285	105. f2 1d 8c 86 0.737 0.1129
18. fe 1a 90 a2 0.251 0.1624	62. c2 b9 ab 38 0.400 0.1285	106. 36 e4 b7 0f 0.541 0.1129
19. ce 64 b1 ce 0.502 0.1595	63. 79 ed 71 8b 0.635 0.1271	107. 9a a4 78 2a 0.600 0.1122
20. db 93 a8 78 0.224 0.1581	64. f8 00 77 d4 0.612 0.1271	108. 26 d7 5b 8b 0.584 0.1122
21. ea de c1 01 0.678 0.1574	65. dd 71 c5 aa 0.263 0.1264	109. 59 d8 54 6b 0.553 0.1122
22. fa ed 23 31 0.459 0.1553	66. e6 5f 9c df 0.647 0.1264	110. 09 ea fe 14 0.435 0.1115
23. be 2a f8 14 0.325 0.1539	67. 24 aa ba e8 0.620 0.1264	111. b6 d0 6b 48 0.239 0.1115
24. 5d bf d4 e7 0.455 0.1532	68. 5b b0 19 0d 0.580 0.1249	112. 71 ee 30 99 0.467 0.1108
25. 8b 6c 63 8b 0.565 0.1518	69. a9 86 7c e9 0.718 0.1242	113. c9 66 ba a4 0.325 0.1108
26. 32 2d 77 37 0.620 0.1518	70. d2 5d 8d 5d 0.573 0.1242	114. c8 0c 80 9a 0.267 0.1094
27. 9a e5 e2 aa 0.561 0.1496	71. db 5e e9 b3 0.604 0.1235	115. dd 5f 85 00 0.557 0.1094
28. 6a da 1c 14 0.533 0.1475	72. 45 1a 7f de 0.533 0.1235	116. 59 b2 a8 2b 0.455 0.1094
29. 5a 4e cd bd 0.580 0.1461	73. 7f 8c bf d7 0.600 0.1228	117. dd 4d d4 00 0.278 0.1094
30. 27 f1 88 b6 0.537 0.1454	74. 3c 13 e7 ef 0.612 0.1221	118. 3b 7d 58 28 0.812 0.1087
31. cb 3a d2 34 0.651 0.1440	75. 13 15 b8 fb 0.678 0.1221	119. 67 41 35 9b 0.706 0.1087
32. 6a 92 3e d8 0.580 0.1440	76. d2 71 18 a2 0.702 0.1214	120. ad f8 c0 b5 0.702 0.1087
33. 2e 2e 67 a5 0.655 0.1433	77. d0 98 8f 4f 0.698 0.1214	121. ae c1 23 d3 0.659 0.1087
34. 92 79 e4 74 0.616 0.1412	78. dd 88 e8 74 0.314 0.1214	122. ca b4 bb 94 0.608 0.1087
35. be 56 be 14 0.451 0.1412	79. 1a 12 5c 57 0.706 0.1207	123. 6b 22 78 de 0.596 0.1087
36. 14 65 4c 62 0.663 0.1398	80. 64 6e e1 14 0.561 0.1200	124. b1 6a 12 f8 0.482 0.1087
37. df 7a 60 b3 0.549 0.1398	81. 4e 22 74 be 0.529 0.1200	125. e5 15 dd 85 0.384 0.1087
38. 21 b2 69 62 0.514 0.1376	82. ad 99 7f 67 0.682 0.1186	126. 35 c8 56 d5 0.733 0.1080
39. 0a 76 e8 d4 0.498 0.1376	83. 22 c4 9d 0b 0.655 0.1186	127. ba c2 c3 4c 0.600 0.1080
40. 1f b1 a1 6b 0.537 0.1362	84. 51 0b d6 14 0.643 0.1186	128. 6c 03 0e 7e 0.467 0.1080
41. 7a 16 bb 48 0.471 0.1362	85. 86 17 4e 49 0.565 0.1186	129. da dc ec c4 0.282 0.1080
42. 12 a9 02 55 0.651 0.1355	86. 70 08 88 5f 0.663 0.1186	130. 85 79 c2 b9 0.702 0.1073
43. dd 5f 85 00 0.557 0.1355	87. e9 e0 b1 89 0.459 0.1179	131. e4 72 31 93 0.694 0.1073
44. bb e2 c2 64 0.220 0.1341	88. 3d f1 97 6b 0.820 0.1179	132. 30 2a 7b df 0.631 0.1073
17680 k=5 rules m-entropy stan-dev		17680 k=5 rules m-entropy stan-dev
12000. ca cb 13 12 0.965 0.0049		15000. e3 4a e9 e9 0.706 0.0007
12001. 4b c7 b0 6e 0.965 0.0049		15001. 1e 3c 62 bf 0.706 0.0007
12002. 85 da 1b d5 0.965 0.0049		15002. 45 5c 50 2e 0.702 0.0007
12003. 91 42 69 ec 0.965 0.0049		15003. 50 35 46 90 0.702 0.0007
12004. 98 37 d5 1e 0.965 0.0049		15004. 64 ac 6a 61 0.702 0.0007
12005. 22 e1 41 dc 0.965 0.0049		15005. 96 12 20 b2 0.702 0.0007
12006. 07 45 d3 7a 0.965 0.0049		15006. 66 e0 b3 b6 0.702 0.0007
12007. b2 8e 75 78 0.965 0.0049		15007. f7 4a 58 c0 0.702 0.0007
12008. 49 ad 37 c4 0.965 0.0049		15008. ad 4b ed 97 0.702 0.0007
12009. a1 04 c4 3d 0.965 0.0049		15009. 02 a1 89 88 0.698 0.0007
12010. 5d 33 d2 b4 0.965 0.0049		15010. b6 76 10 b6 0.698 0.0007
12011. 63 d6 3e b2 0.965 0.0049		15011. f0 00 17 10 0.698 0.0007
12012. f6 d9 1a 38 0.961 0.0049		15012. f4 e9 65 ac 0.694 0.0007
12013. ed 27 7c 73 0.961 0.0049		15013. 9e db b2 88 0.694 0.0007
12014. c6 7b da 26 0.961 0.0049		15014. 02 a1 89 88 0.694 0.0007
12015. 41 83 ed 13 0.961 0.0049		15015. 7d ce 04 91 0.694 0.0007
12016. 25 87 12 09 0.961 0.0049		15016. f7 5e 0c c3 0.694 0.0007
12017. 80 3b 3c d6 0.961 0.0049		15017. 84 38 78 72 0.690 0.0007
12018. 99 77 aa cc 0.961 0.0049		15018. 5b fd 17 5f 0.690 0.0007
12019. e1 94 31 c9 0.961 0.0049		15019. a9 70 fa 2a 0.690 0.0007
12020. 08 af d7 f2 0.961 0.0049		15020. 6a 02 ce 91 0.690 0.0007
12021. 0e 5f 3e 96 0.961 0.0049		15021. b3 50 79 4e 0.690 0.0007
12022. d2 16 33 75 0.961 0.0049		15022. 92 09 53 80 0.690 0.0007
12023. 20 09 66 96 0.961 0.0049		15023. b5 d7 f5 a6 0.690 0.0007
12024. 6c 93 6f d8 0.961 0.0049		15024. 05 80 d8 32 0.686 0.0007
12025. 02 df b3 68 0.961 0.0049		15025. 0f 0c 6d d5 0.686 0.0007
12026. 91 01 dc 29 0.961 0.0049		15026. 34 b5 21 5f 0.686 0.0007
12027. 22 16 d4 19 0.961 0.0049		15027. fb 67 7d dc 0.686 0.0007
12028. b7 9d b1 b6 0.961 0.0049		15028. 07 19 c8 18 0.686 0.0007
12029. a7 35 47 0b 0.961 0.0049		15029. 6c ea c9 ce 0.686 0.0007
12030. e8 65 3e 79 0.961 0.0049		15030. aa d9 c7 d7 0.686 0.0007
12031. 3f 92 6a 11 0.961 0.0049		15031. b0 9b d1 d7 0.682 0.0007
12032. 4c 61 21 2c 0.961 0.0049		15032. 00 29 cf af 0.682 0.0007
12033. 57 a7 fc 78 0.961 0.0049		15033. f3 23 ba 08 0.682 0.0007
12034. b9 34 b2 93 0.961 0.0049		15034. cf db 7b 21 0.682 0.0007
12035. 2d a4 73 d4 0.961 0.0049		15035. ed 60 82 62 0.678 0.0007
12036. ed 27 7c 73 0.961 0.0049		15036. 21 e3 4a cb 0.678 0.0007
12037. 06 41 a9 9c 0.961 0.0049		15037. 7f 59 34 45 0.678 0.0007
12038. 27 92 81 32 0.961 0.0049		15038. 40 3e c4 9c 0.678 0.0007
12039. a2 8f 1e 8a 0.961 0.0049		15039. 41 68 f0 d0 0.678 0.0007
12040. 47 fb 57 ca 0.961 0.0049		15040. 04 e5 f6 62 0.678 0.0007
12041. 56 f7 98 cb 0.961 0.0049		15041. d5 f5 d6 b7 0.675 0.0007
12042. 4a 2c 25 0b 0.957 0.0049		15042. 58 43 2c 02 0.675 0.0007
12043. 9d 88 3d 7c 0.957 0.0049		15043. 79 f8 f5 58 0.675 0.0007

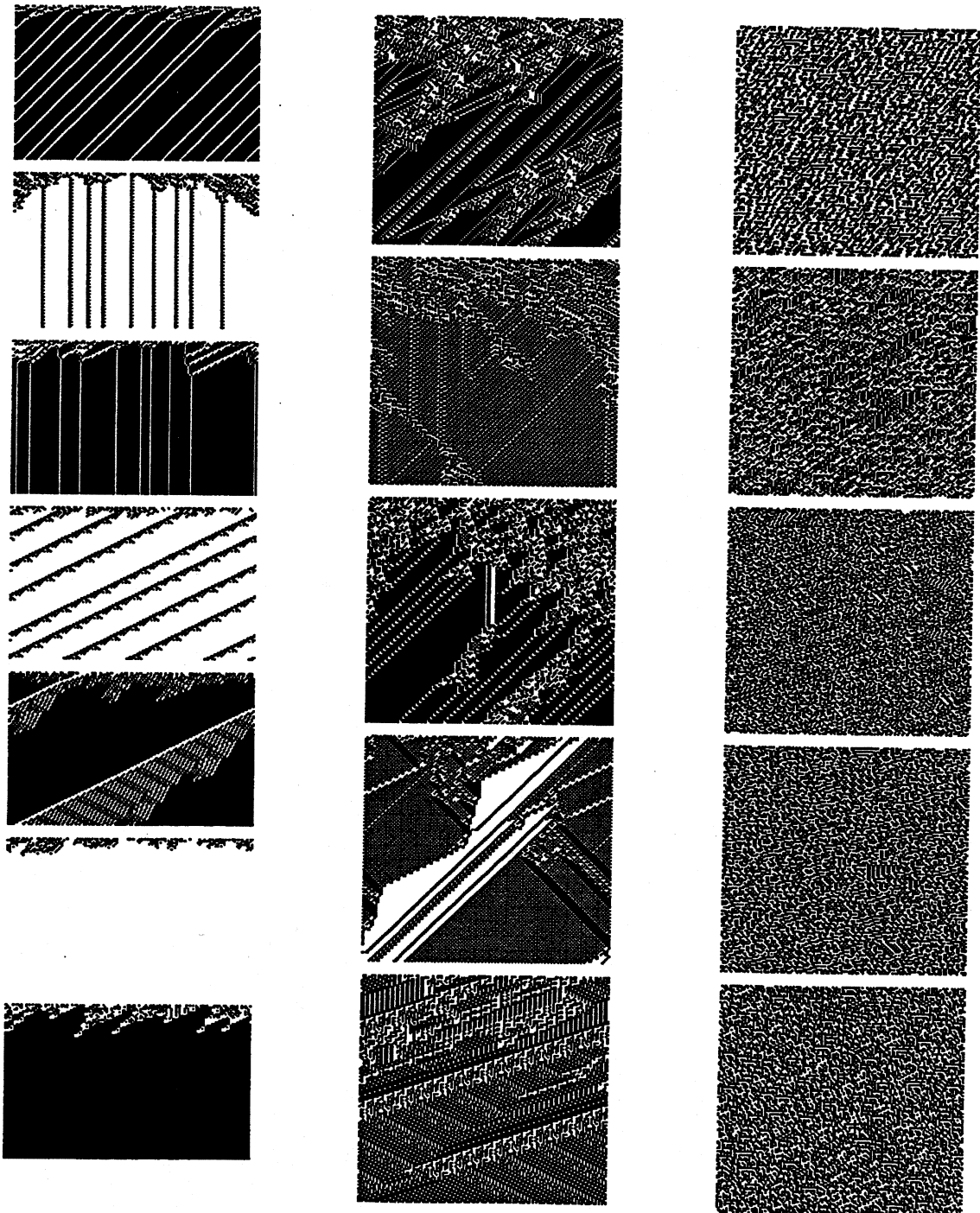
### Appendix 5.2.1. ( $k=6$ )

Examples of the space-time patterns of ordered, complex and chaotic rules from the automatic sample. This supplements the examples of complex rules in figures 4.5.  $n=150$ . The rule numbers can be found by the rule index, they are listed in this appendix 5.2.5.

*left:* ordered rules, 100 time steps from a random initial state, index 15001-15007.

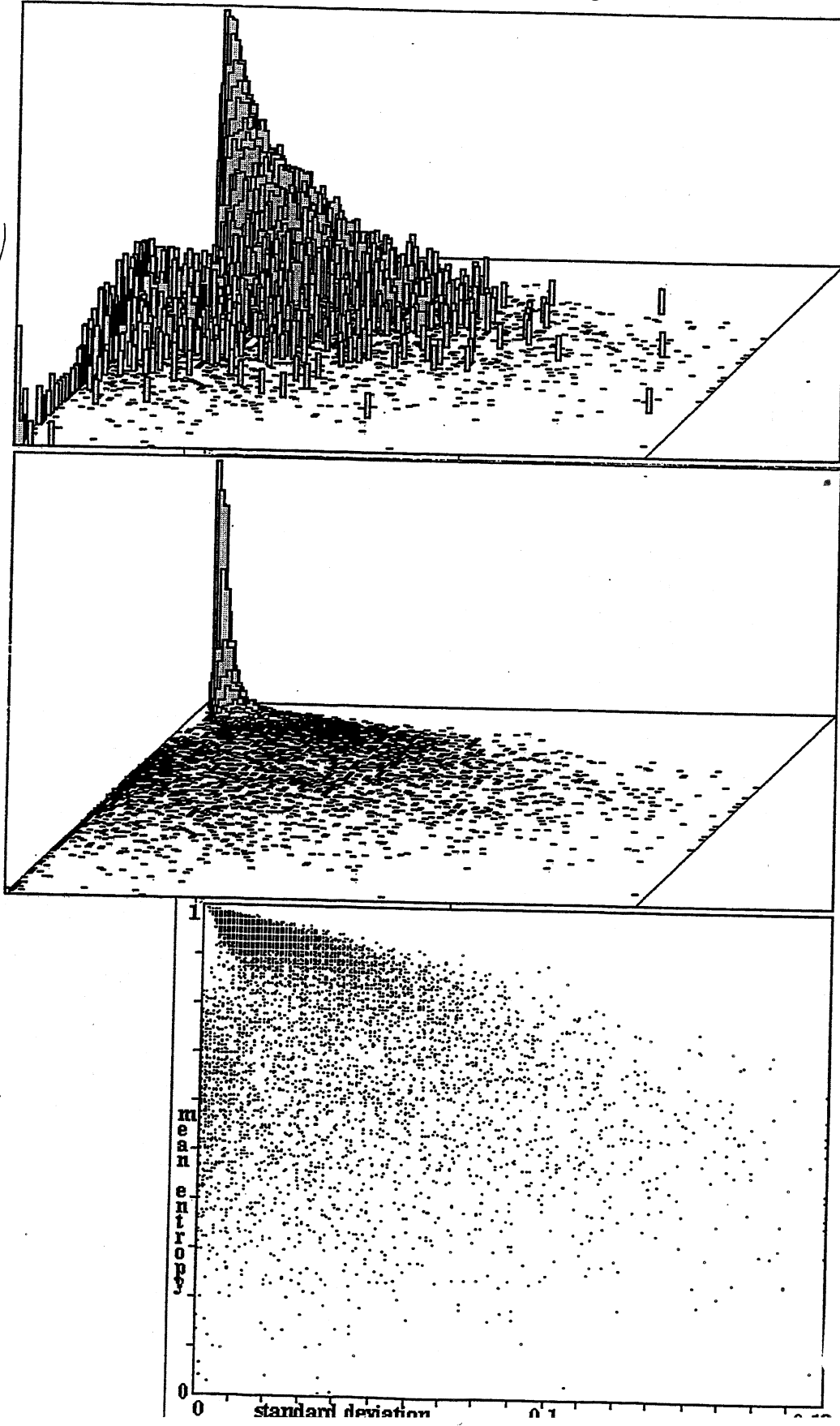
*centre:* complex rules, 140 time steps from a random initial state, index 22-26.

*right:* chaotic rules, 140 time steps from a random initial state, index 12001-12005.



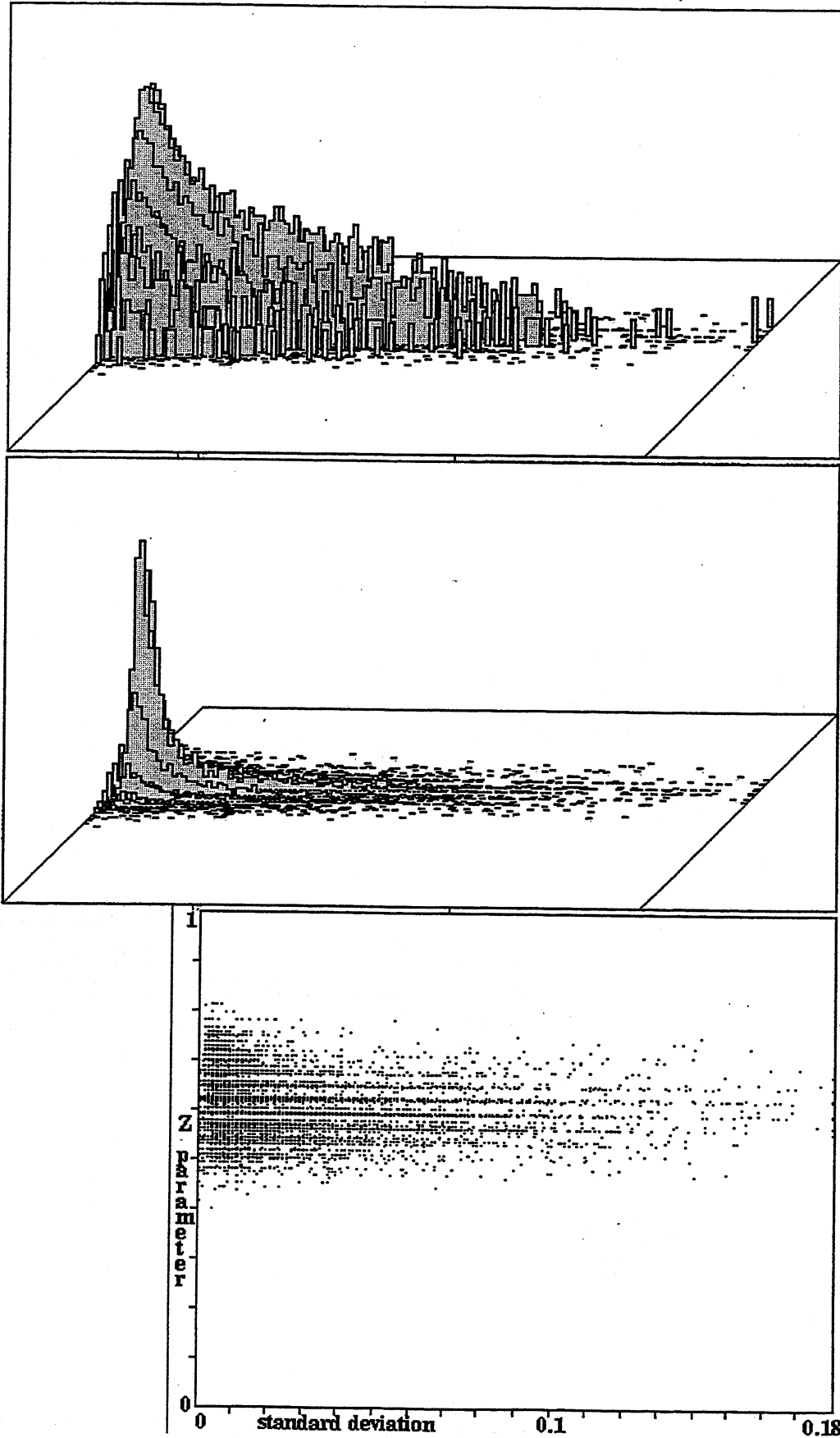
**Appendix 5.2.2. ( $k=6$ )**

*Below:* Mean entropy ( $y$ -axis) plotted against standard deviation for a random sample of 15425  $k=6$  rules,  $n=150$  (see section 4.11, and figure 4.24.). *Centre:* The frequency distribution of points falling within a  $128 \times 128$  grid. *Top:* the frequency shown in log form.



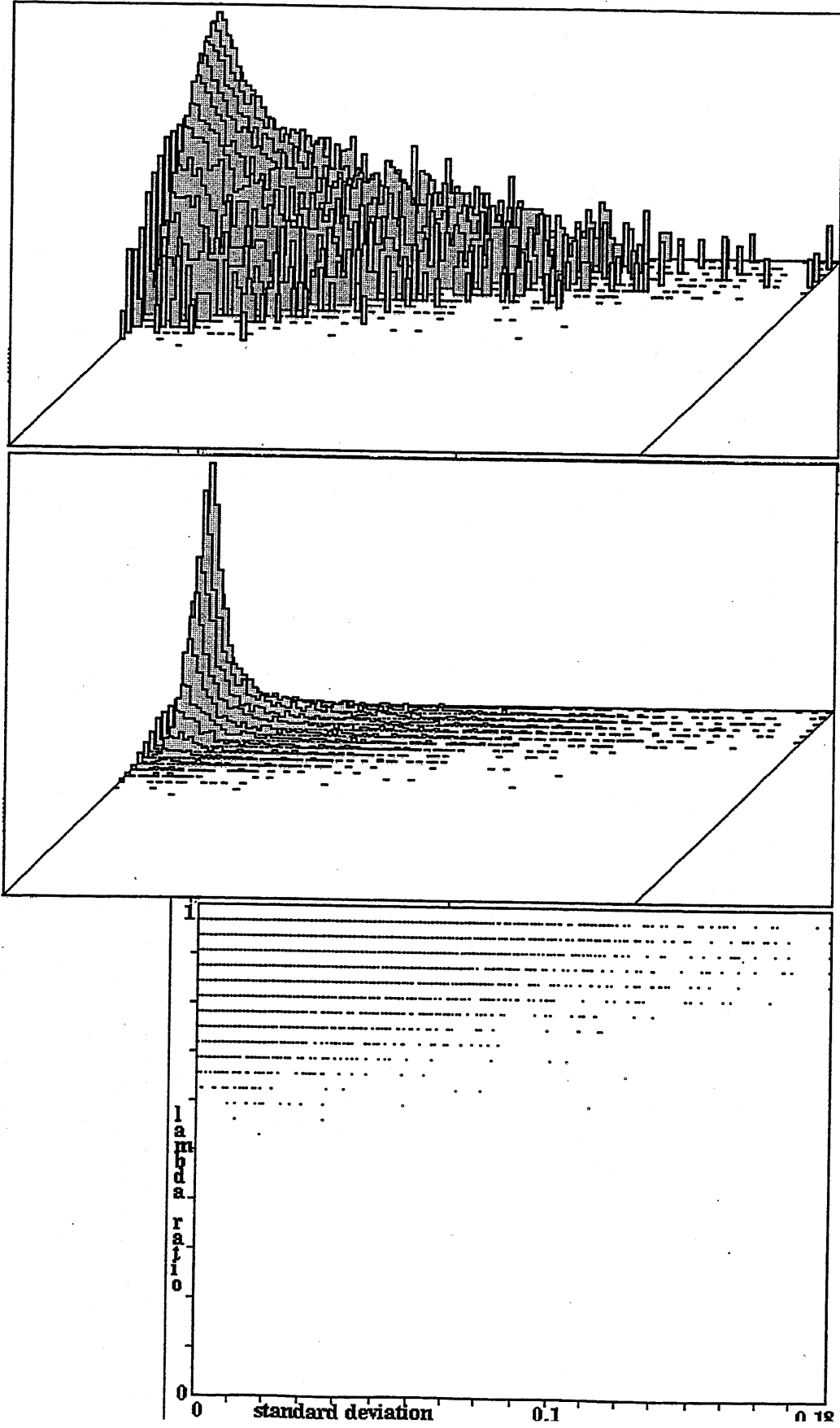
**Appendix 5.2.3. ( $k=6$ )**

*Below:* The  $Z$  parameter ( $y$ -axis) plotted against standard deviation for a random sample of 15425  $k=6$  rules,  $n=150$  (see section 4.12). *Centre:* The frequency distribution of points falling within a  $128 \times 128$  grid. *Top:* the frequency shown in log form. High  $Z$  indicates chaos, low  $Z$  order.



### Appendix 5.2.4. ( $k=6$ )

*Below:*  $\lambda_{\text{ratio}}$  ( $y$ -axis) plotted against standard deviation for a random sample of 15425  $k=6$  rules,  $n=150$  (see section 4.14). *Centre:* The frequency distribution of points falling within a  $128 \times 128$  grid. *Top:* the frequency shown in log form.







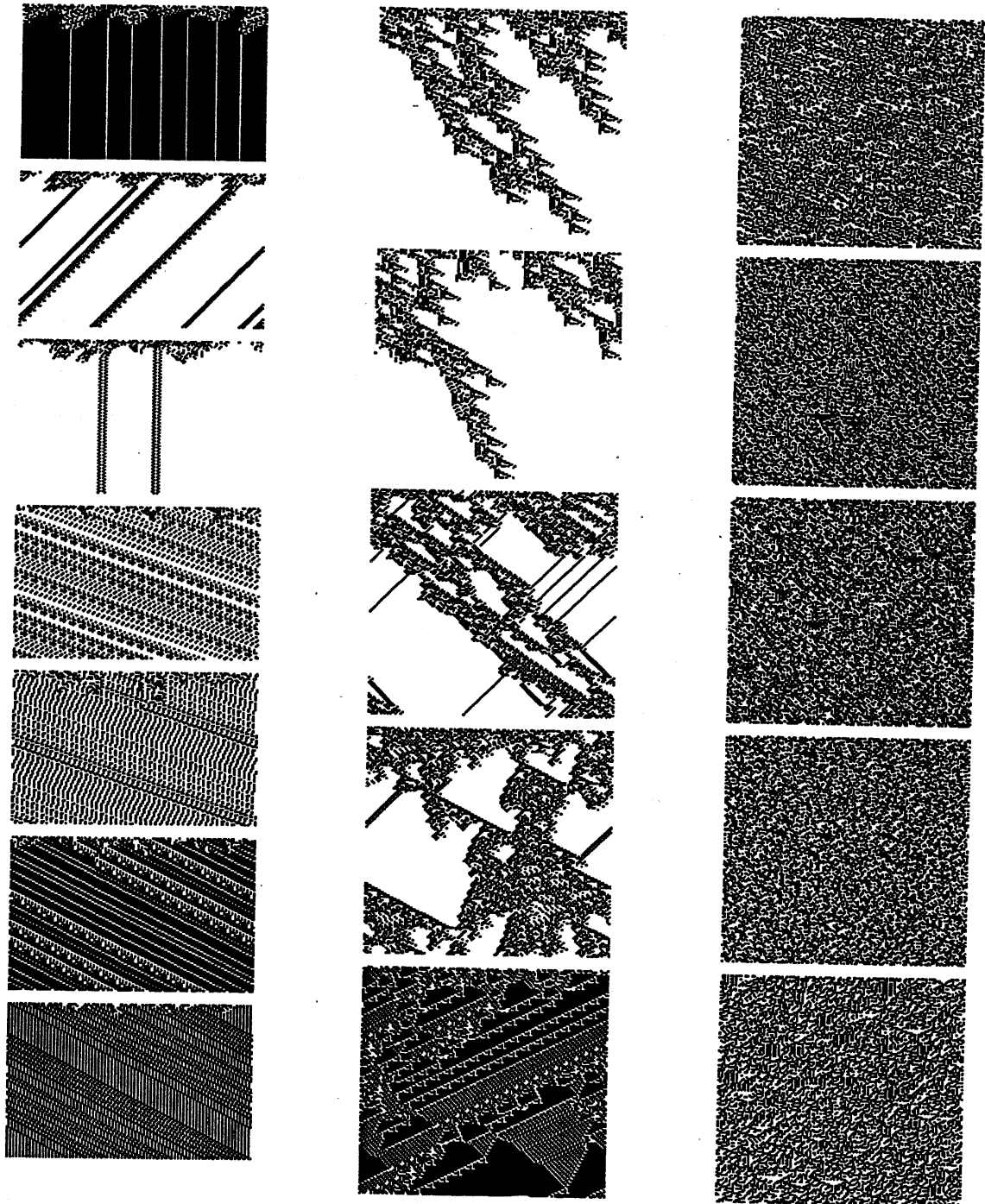
### Appendix 5.3.1. ( $k=7$ )

Examples of the space-time patterns of ordered, complex and chaotic rules from the automatic sample. This supplements the examples of complex rules in figures 4.6.  $n=150$ . The rule numbers can be found by the rule index, they are listed in this appendix 5.3.5.

*left:* ordered rules, 100 time steps from a random initial state, index 14189-14195.

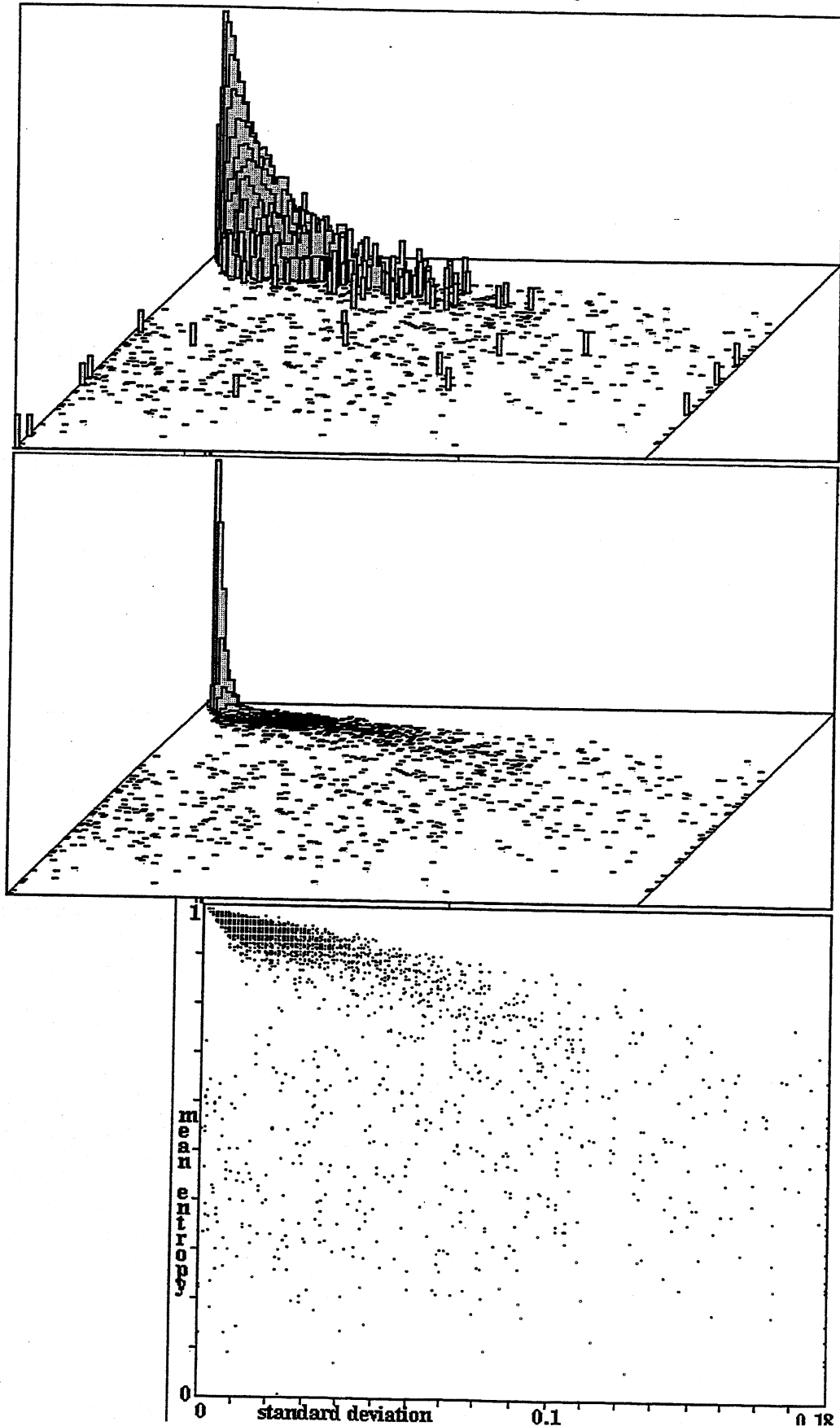
*centre:* complex rules, 140 time steps from a random initial state, index 22-26.

*right:* chaotic rules, 140 time steps from a random initial state, index 12001-12005.



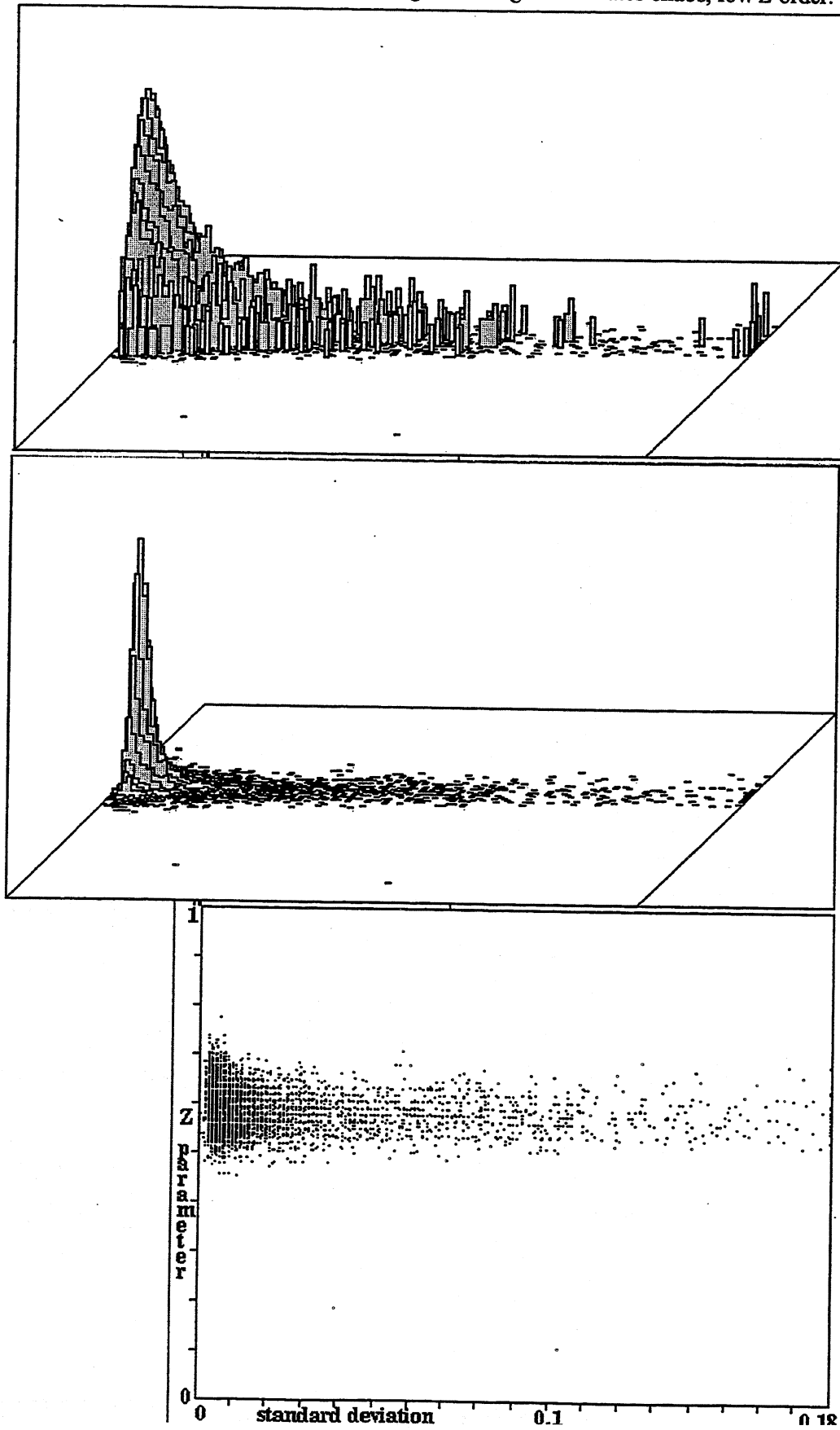
**Appendix 5.3.2. ( $k=7$ )**

*Below:* Mean entropy ( $y$ -axis) plotted against standard deviation for a random sample of 14221  $k=7$  rules,  $n=150$  (see section 4.11, and figure 4.25). *Centre:* The frequency distribution of points falling within a  $128 \times 128$  grid. *Top:* the frequency shown in log form.



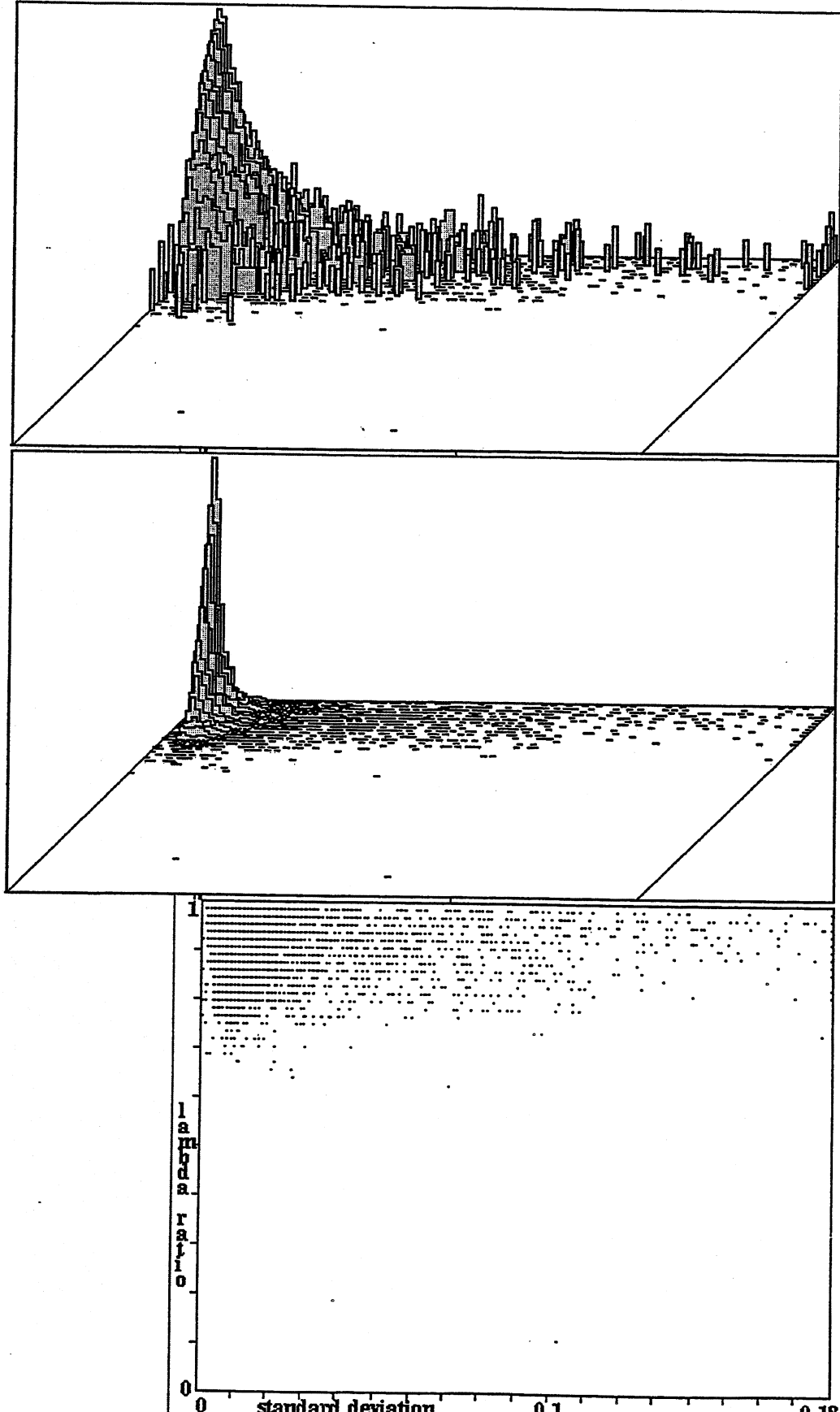
### Appendix 5.3.3. ( $k=7$ )

*Below:* The  $Z$  parameter ( $y$ -axis) plotted against standard deviation for a random sample of 14221  $k=7$  rules,  $n=150$  (see section 4.12). *Centre:* The frequency distribution of points falling within a  $128 \times 128$  grid. *Top:* the frequency shown in log form. High  $Z$  indicates chaos, low  $Z$  order.



### Appendix 5.3.4. ( $k=7$ )

*Below:*  $\lambda_{ratio}$  ( $y$ -axis) plotted against standard deviation for a random sample of 14221  $k=7$  rules,  $n=150$  (see section 4.14). *Centre:* The frequency distribution of points falling within a  $128 \times 128$  grid. *Top:* the frequency shown in log form.



Appendix 5.3.5. (k=7)

Examples of complex, chaotic and ordered rules from the sorted sample of 14221. The sample is sorted from highest to lowest standard deviation, and by highest to lowest mean entropy for each value of standard deviation. Note that these measures are saved as 8 bits so the resolution is 1/256. Top: complex rules, index 1-132. Bottom left: chaotic rules, index 12000-12043. Bottom right: ordered rules, index 14175-14218.

Table with 3 columns: 14221 k=7 rules m-ent s-dev, 14221 k=7 rules m-entropy stan-dev, and 14221 k=7 rules m-entropy stan-dev. Each column contains a list of rule numbers and their corresponding entropy and standard deviation values.

Table with 2 columns: 14221 k=7 rules m-entropy stan-dev and 14221 k=7 rules m-entropy stan-dev. This section contains a list of chaotic rules with their entropy and standard deviation values.

Table with 2 columns: 14221 k=7 rules m-entropy stan-dev and 14221 k=7 rules m-entropy stan-dev. This section contains a list of ordered rules with their entropy and standard deviation values.

## Appendix 6

### Reverse Algorithm and Z parameter Code

Appendix 5 gives the C code functions used in Discrete Dynamics Lab (DDLab) for three alternative algorithms for computing pre-images, and also for computing the Z parameter for a rule-table. DDLab is interactive graphics software for investigating discrete dynamical networks including their attractor basin (Wuensche 1996).

#### Appendix 6.1.

##### **Random Maps.**

The function for computing pre-images for random maps. This function can also be applied to CA and RBN as well as random maps. For context refer to chapter 2.2, 3.6.

#### Appendix 6.2.

##### **Cellular Automata.**

The function for computing pre-images for CA (or CA with mixed rules), for any homogenous neighbourhood from 1-9. For context refer to chapters 2.2 and 4.4.

#### Appendix 6.3.

##### **Random Boolean Networks.**

The function for computing the pre-images for RBN (mixed neighbourhoods 1-9, mixed rules, arbitrary wiring. This function can also be applied to CA. and intermediate architecture between CA and RBN. For context refer to chapters 2.2 and 5.5.

#### Appendix 6.4.

##### **Z parameter.**

The function for computing the Z parameter. For context refer to chapter 4.12-13.

## Appendix 6.1.

The function in DDLab (C code) for computing pre-images for random maps. This function can also be applied to CA and RBN as well as random maps.

For context refer to Chapter 2.2, 3.6.

```

***** compute pre-images by exhaustive testing *****
void pre_exhaustive(void){

    unsigned long state_space,prestate;

    attrepeat=0;
    state_space=(unsigned long)(pow(2,(double)lineL));

    for (prestate=0;prestate<=state_space-1;prestate++){
        if (get_bitlong(tickoff,prestate)) continue;
        if (memcmp(A,exh_pairs+prestate*bytesL,bytesL)==0){
            /* pre-image found */
            dec_to_bit_line(F,prestate,lineL);
            /* check for cycle repeat, rotation (from all 0s or 1s,
            if ok add, print */
            if (check_pre_image(F)==0){
                accept_pre_image(F);
                /* if interrupt */
                interrupt_pre_fan();
                if (is[0]=='q' || is[0]=='n') return;
            }
        }
    }
}

```

## Appendix 6.2.

The function in DDLab (C code) for computing pre-images for CA (or CA with mixed rules), for any homogenous neighbourhood from 1-9.

For context refer to Chapters 2.2 and 4.4.

```

/***** compute pre-images 1D local, 1 rule or mixed *****/
void pre_images(void){
    signed short
        ppindex,          /* continue pp from ppindex */
        *ppend,          /* holds index reached by pp */
        z,bytereached,   /* pos of index in countdown of bytes in B */
        n,m,
        u,ul,ur,i,       /* n'hood value */
        tablebytes;      /* no of bytes in rule table */
    char **pp,           /* array of pointers
                        to partial pre-images pointers */
        trypp=0, failpcheck=0,
        nrad,oddrad=0,ppcomplete,tul;
    unsigned long ppcount=0, /* count of partial pre-images in queue */
        maxxpcount=0;

    attrepeat=0;        /* 1 if pre_image on attractor cycle */
    if ((pp=(char**)calloc(1,sizeof(char*)))==NULL) outof_memory("pp");
    if ((ppend=(signed short*)calloc(1,sizeof(signed short)))
        ==NULL) outof_memory("ppend");
    /*      nhood=3      nhood=4      etc. asymmetric nhood is
       o o *          o o o *      wider on the right
       \ | /          \ | / /
        T              T          */
    nrad=nhood/2;
    if (nhood%2==0){
        nrad-=1;
        oddrad=1;
    }
    /* bitstrings A,B and pp:
           L          n          0
           |          |          |
    -----|-----|-----| .....A
           2          1          0

           L+8      m m-nrad      8      0
           | |      | |          |          |
    -----|-----|-----|-----| .....B
           4          3          2          1          0

           set to 1  0
                   |  |
    -----|-----|-----| .....PP
           2          1          0          */
    memset(B,0,bytesL+2); /* set all bits in B to 0 */

```



```

for (z=nhood-2+oddrad;z>=0;z--)
    set_bit(B,8 +L +z-(nrad+oddrad-1),p[z]);

/*      L(+8)
      |
=1      -----
=2      x-----
=3      xx-----
=4      xxx-----
=5      xxxx-----
=6      xxxxx-----
=7      xxxxxx-----
=8      xxxxxxx-----
=9      xxxxxxxx----- */

ppindex=L;
tablebytes=(signed short)ceil(pow(2,(double)nhood)/8);
/*homogenous rule*/
if (!rulemix){
    memcpy(arule,carule[0],tablebytes);
    memcpy(arule1,carule[0],tablebytes);
    memcpy(arule2,carule[0],tablebytes);
}
prel:
for ( n=ppindex;n>=0;n--){      /* from right to left -----> */
    if (rulemix){
        memcpy(arule,carule[n],tablebytes);          /* rule at n*/
        memcpy(arule1,carule[(lineL+n-1)%lineL],tablebytes);/* ..at n-1*/
        memcpy(arule2,carule[(lineL+n+1)%lineL],tablebytes);/* ..at n+1*/
    }
    /*backwards logic*/
    m=n+8;
    /*excluded permutation left->right, 21*->T,321*->T etc.,
    nhood ending with 0*/
    u=0; i=nhood;
    for (z=nrad;z>=1-nrad-oddrad;z--){
        i--;
        u += get_bit(B, m+z) * (signed short)pow(2,(double)i);
    }
    if (get_bit(arule,u)==get_bit(arule,u+1) &&
        get_bit(A,n)!=get_bit(arule,u)){
        trypp=1;
        break;
    }
    /*excluded permutation right->left,T<-*10, T<-*321 etc.,
    nhood starting with 0*/
    ur=0;i=nhood-1;

    for (z=nrad;z>=1-nrad-oddrad;z--){
        i--;
        ur = ur + get_bit(B, m+z) * (signed short)pow(2,(double)i);
    }
    if (get_bit(arule2,ur)==
        get_bit(arule2,ur+(signed short)pow(2,(double)nhood)/2)
        && get_bit(A,(lineL+n+1)%lineL)!=get_bit(arule2,ur)){

```

```

    trypp=1;
    break;
}
/* deterministic permutation left->right */
if (get_bit(arule,u)!=get_bit(arule,u+1)){
    if (get_bit(A,n)==get_bit(arule,u))
        set_bit0(B,m-nrad-oddrad); /* right end of nhood=0 */
    else set_bit1(B,m-nrad-oddrad); /* right end of nhood=1 */

    /* check boundary conditions */
    if (n<=nhood-2 && p[n]!=get_bit(B,m-nrad-oddrad)){
        trypp=1;
        break;
    }
    continue;
}
/* else if ambiguous permutation */
set_bit0(B,m-nrad-oddrad); /* next cell could be both 1 and 0,
                               assume 0 */
/* shift right nhood ending with 0 */
ul=0;i=nhood;
for (z=nrad-1;z>=-nrad-oddrad;z--){
    i--;
    ul += get_bit(B, m+z) * pow(2,(double)i);
}
if (get_bit(arule1,ul)==get_bit(arule1,ul+1)){
    /* check for deterministic permutation 2-tuples pairs unequal
        A * B *
        |   |
        -2  2   */
    if (ul%4==0) tul=2;
    else tul=-2;
    if (get_bit(arule1,ul)!=get_bit(arule1,ul+tul) &&
        get_bit(arule1,ul+tul)==get_bit(arule1,ul+tul+1)){
        if (get_bit(A,(lineL+n-1)%lineL)==get_bit(arule1,ul)){
            if (tul==2) set_bit0(B,m-nrad-oddrad);
            else set_bit1(B,m-nrad-oddrad);
        }
    }
    else{
        if (tul==2) set_bit1(B,m-nrad-oddrad);
        else set_bit0(B,m-nrad-oddrad);
    }
    /* check boundary conditions */
    if (n<=nhood-2 && p[n]!=get_bit(B,m-nrad-oddrad)){
        trypp=1;
        break;
    }
    continue;
}
/* shift ppindex->right,
check for excluded permutation 21*->T, 4321*->T, etc.
if excluded change assumed next cell from 0 to 1 */
if (get_bit(A,(lineL+n-1)%lineL) !=get_bit(arule1,ul)){
    set_bit1(B,m-nrad-oddrad);
}

```

```

    /* check if 1 is compatible with boundary conditions */
    if (n<=nhood-2 && p[n]!=1){
        trypp=1;
        break;
    }
    continue;
}
}
/* end backwards logic */
if (n<=nhood-2 && p[n]!=1){ /* check boundary for */
    continue; /* alternative ending in 1 */
} /* to be added to pp stack */
ppcount ++; /* ppcount=partial preimage no */
if (maxppcount<ppcount){
    maxppcount=ppcount;
    if( (pp=(char**) realloc(pp,sizeof(char)*(ppcount+1)))
        ==NULL) outof_memory("pp");
    if((ppend=(signed short*)realloc
        (ppend,sizeof(signed short)*(ppcount+1)))
        ==NULL) outof_memory("ppend");
}
/* increace bar showing pp stack*/
if (see_ppstack && ppcount<150){
    setcolor(magenta);
#       if __BORC__ || __UNIX__
        line(vgax(125)+vgax(ppcount),Maxy-vgay(40),
            vgax(125)+vgax(ppcount),Maxy-vgay(34));
#       elif __WATCOM__
        line(vgax(140)+vgax(ppcount),Maxy-vgay(40),
            vgax(140)+vgax(ppcount),Maxy-vgay(34));
#       endif

    setcolor(white);
}
/* in B */
bytereached=(m-nrad-oddrad)/8; /* countdown */
/* alloc required bytes */
sprintf(textb[0],"pp[%lu]",ppcount);
if ((pp[ppcount]=(char *)malloc(bytesL+2-bytereached))
    ==NULL) outof_memory(textb[0]);
memset(pp[ppcount], 0, bytesL+2-bytereached);
/* signed short array holding ppend index -
   to be continued from this */
ppend[ppcount]=n;
memcpy(pp[ppcount],B+bytereached,bytesL+2-bytereached);
set_bit1(pp[ppcount],(m-nrad-oddrad)%8); /* set last bit to 1 */
} /*** next n ***/
if (trypp==1){
    trypp=0;
    goto pre3;
}
pre2:
failpcheck=0; /*periodic boundary check */

```

```

for (z=nhood-2;z>=0;z--){
    if(p[z]!=get_bit(B,8+z-nrad-oddrad)){
        failpcheck=1;
        break;
    }
}
if (failpcheck==1) goto pre3;
/*pp line is now complete and valid*/
/*set_bits in B beyond L to 0, allows bytes to be compared*/
for (z=(bytesL+1)*8-1;z>=L+1;z--) set_bit0(B,z+8);

/* check for cycle repeat, rotation (from all 0s or 1s),
    if ok add, print */
if (check_pre_image(B+1)==1) goto pre3;
accept_pre_image(B+1);
/* if interrupt */
interrupt_pre_fan();
if (is[0]=='q' || is[0]=='n'){
    for (z=1;z<=ppcount;z++) free(pp[z]);
    free(pp);
    free(ppend);
    ppcount=0;
    return;
}
pre3:
if (ppcount==0){
    free(pp);
    free(ppend);
    return; /*preim stack empty*/
}
ppcomplete=0;
if (ppend[ppcount] == 0) ppcomplete=1;
memset(B,0,bytesL+2); /*set all bits in B to 0*/
bytereached=(8+ppend[ppcount]-nrad-oddrad)/8; /*countdown*/
memcpy(B+bytereached,pp[ppcount],bytesL+2-bytereached);
ppindex=ppend[ppcount]-1;
free(pp[ppcount]);
ppcount--;
if (see_ppstack && ppcount+1<150){ /*decrease bar showing pp stack*/
    setcolor(black);
#   if __BORC__ || __UNIX__
        line(vgax(125)+vgax(ppcount+1),Maxy-vgay(40),
            vgax(125)+vgax(ppcount+1),Maxy-vgay(34));
#   elif __WATCOM__
        line(vgax(140)+vgax(ppcount+1),Maxy-vgay(40),
            vgax(140)+vgax(ppcount+1),Maxy-vgay(34));
#   endif

    setcolor(white);
}
if (ppcomplete==1) goto pre2;
goto pre1;
}

```

### Appendix 6.3.

The function in DDLab (C code) for computing pre-images for RBN (mixed neighbourhoods 1-9, mixed rules, arbitrary wiring. This function can also be applied to CA. and intermediate architecture between CA and RBN.

For context refer to Chapters 2.2 and 5.?

```

/***** compute pre-images, - random Boolean networks *****/
void pre_images_rand(void){
  char tog,      /* toggle id of two latest partial pre-image stacks */
  newbit,       /* new bit from pseudo-nhood */
  PSbit,        /* bit at wire coupling in partial pre-image */
  existbit,     /* bit flags if above has been set=1, or is empty=0 */
  conflict=0;
  /* global variables: */
  **nhoodE,     reduced effective nhood
  **skiptable; skip rule table entry because nhood reduced
  */
  unsigned short zu,
  ppstack,     /* partial pre-image index */
  wildpre;    /* to deal with pre-image wildcards */
  /* global variable:
  *PScount,    count of partial pre-images in toggled stacks */
  signed short
  Tcell,      /* target cell index */
  z,zz,
  Rtable,     /* rule table index */
  tablemax,   /* highest index in rule table */
  tablebytes, /* no of bytes " " " */
  byteno,
  tempx, tempy,
  possofw,   /* to deal pre-image with wildcards */
  pp_temp;

  attrepeat=0; /* 1 if pre_image on attractor cycle */
  sprintf(textb[0],"PS.& PSflag.");
  if ((PS[1][0]=(char *)calloc(1,sizeof(char)))
      ==NULL) outof_memory(textb[0]);
  if ((PSflag[1][0]=(char *)calloc(1,sizeof(char)))
      ==NULL) outof_memory(textb[0]);
  for (z=0;z<=1;z++){
    if (z==1) byteno=bytesL;
    else byteno=1;
    if ((PS[0][z]=(char *)calloc(byteno,sizeof(char)))
        ==NULL) outof_memory(textb[0]);
    if ((PSflag[0][z]=(char *)calloc(byteno,sizeof(char)))
        ==NULL) outof_memory(textb[0]);
  }
}

```

```

tog=0;
tablemax=(signed short)pow(2,(double)nhood)-1;
tablebytes=(signed short)ceil(((double)tablemax+1)/8);
PScount[0]=1; /*notional initial pp stack=1*/
memset(PS[0][1],0,bytesL);
memset(PSflag[0][1],0,bytesL);
PScount[1]=0;
if (!rulemix) memcpy(arule,carule[0],tablebytes);

if (see_ppstack){
  pp_histcount+=1; /* to find average histogram */
  /* rescale y axis smaller */
  if (ppL==lineL && pp_hist_max < pp_ymax/6){
    pp_ymax=pp_ymax*.75;
    if (pp_ymax<10) pp_ymax=10;
    ppL=0;
  }
  pp_hist_max=0;
  setup_ppstack_hist();
}

for (Tcell=lineL-1;Tcell>=0;Tcell--){ /* for each target cell */
  if (nhood_mixed){
    tablemax=(signed short)pow(2,(double)nhoodm[Tcell])-1;
    tablebytes=(signed short)ceil(((double)tablemax+1)/8);
  }
  if (rulemix) memcpy(arule,carule[Tcell],tablebytes);
  tog=abs(tog-1); /* toggle 0-1 for even-odd, first tog=1 */
  /* de-allocate mem in current stack */
  if (PScount[tog]>0){
    for (zu=1;zu<=PScount[tog];zu++){
      free(PS[tog][zu]);
      free(PSflag[tog][zu]);
    }
    /* realloc mem for new pp stack stack PS[tog][set count to 0] */
    sprintf(textb[0],"PS..PSflag..[%d]",tog);
    if ((PS[tog]=(char**)realloc(PS[tog],sizeof(char*)))==NULL)
outof_memory(textb[0]);
    if ((PSflag[tog]=(char**)realloc(PSflag[tog],sizeof(char*)))
        ==NULL) outof_memory(textb[0]);
  }
  PScount[tog]=0; /* set new pre-image count to 0 */
  /* for each item in previous partial pre-image stack */
  for (ppstack=1;ppstack<= PScount[abs(tog-1)];ppstack++){
    /* each rule table entry */
    for (Rtable=tablemax;Rtable>=0;Rtable--){
      /* reject rule table entries that dont match target cell */
      if (get_bit(A,Tcell)!=get_bit(arule,Rtable)) continue;
      /* skip redundant rule table entries */
      if (get_bit(skiptable[Tcell],Rtable)==1) continue;
      /* copy partial pre-image from stack to temp bitstring */
      memcpy(PStemp,PS[abs(tog-1)][ppstack],bytesL);
      /* copy parrallel flag array from flag stack to temp bitstring */
      memcpy(flagtemp,PSflag[abs(tog-1)][ppstack],bytesL);
    }
  }
}

```

```

for (z=nhoodm[Tcell]-1;z>=nhoodE[Tcell][Rtable];z--){
    /*get new bit from pseudo-nhood */
    newbit = get_bit(nm[z],Rtable);
    /*get value at wire coupling, 0 or 1 */
    PSbit = get_bit(PStemp,wiring[Tcell][z]);
    existbit=get_bit(flagtemp,wiring[Tcell][z]);
    /*ERROR - if bits conflict and bit set reject psuedo-nhood */
    /* if bits conflict and bit set reject psedo-nhood */
    if (newbit!=PSbit && existbit==1){
        conflict=1;
        break;
    }
    /* if bit empty insert a new bit, set flag */
    if (newbit!=PSbit) set_bit(PStemp,wiring[Tcell][z],newbit);
    /* flag the new bit as existing */
    if (existbit!=1) set_bit1(flagtemp,wiring[Tcell][z]);
}
if (conflict==1){
    conflict=0;
    continue;
}
PScount[tog]+=1;
/* realloc mem for new pp stack stack PS[tog][PScount[tog]] */
sprintf(textb[0],"bigger pp stack=%d",PScount[tog]);
if ((PS[tog]=(char**)realloc(PS[tog],(PScount[tog]+1)*
    sizeof(char*)))==NULL) outof_memory(textb[0]);
if ((PSflag[tog]=(char**)realloc(PSflag[tog],(PScount[tog]+1)*
    sizeof(char*)))==NULL) outof_memory(textb[0]);
if ((PS[tog][PScount[tog]]=(char*)calloc(bytesL,sizeof(char)))
    ==NULL) outof_memory(textb[0]);
if((PSflag[tog][PScount[tog]]=(
    char*)calloc(bytesL,sizeof(char)))
    ==NULL) outof_memory(textb[0]);
/*copy partial pre-image to next pp stack */
memcpy(PS[tog][PScount[tog]],PStemp,bytesL);
/*copy parrallel flag array to next flag stack */
memcpy(PSflag[tog][PScount[tog]],flagtemp,bytesL);
}/* next Rtable */
}/* next ppstack */

if (PScount[tog]==0) break;
/* show partial pre-image histogram */
if (see_ppstack){
    pp_hist_total[Tcell]+=PScount[tog];/*to find average histogram */
    pp_hist[Tcell]=PScount[tog];
    /* rescale y axis bigger */
    if (PScount[tog]>pp_ymax){
        pp_ymax=PScount[tog]*1.2;
        ppL=0;
        setup_ppstack_hist();
        for (z=lineL-1;z>=Tcell;z--){
            pp_heighty=
                (signedshort)(vgax(120)*(float)pp_hist[z]/pp_ymax);

```

```

        bar(pp_startx,pp_starty-pp_heighty,
            pp_startx+pp_barwidth,pp_starty);
        pp_startx+=pp_bartotw;
    }
}
else{
    pp_heighty=(signed short)(vgax(120)*(float)PScount[tog]/pp_ymax);
    bar(pp_startx,pp_starty-pp_heighty,
        pp_startx+pp_barwidth,pp_starty);
    pp_startx+=pp_bartotw;
}
}
if (pp_hist_max < pp_hist[Tcell]) pp_hist_max=pp_hist[Tcell];
}
if (PScount[tog]>0){
    wildpre=0;
    while (wildpre<PScount[tog]){
        wildpre+=1;
        while(1){
            possofw=-1;
            for (z=lineL-1;z>=0;z--){
                if (get_bit(PSflag[tog][wildpre],z)==0){
                    possofw=z;
                    break;
                }
            }
            /* wildcard not found */
            if (possofw==-1) break;
            /* wildcard found - undo wildcard in PSflag */
            set_bit1(PSflag[tog][wildpre],possofw);
            /* copy pre-image to PStemp */
            memcpy(PStemp,PS[tog][wildpre],bytesL);
            /* copy parrallel flag array to flagtemp */
            memcpy(flagtemp,PSflag[tog][wildpre],bytesL);
            /* set 1 at wildcard poss in temp array */
            set_bit1(PStemp,possofw);
            PScount[tog]+=1;
            /* realloc mem for new pp stack stack PS[tog][PScount[tog]] */
            sprintf(textb[0],"wildpre, pp stack=PScount[tog]");
            if ((PS[tog]=(char**)realloc(PS[tog],(PScount[tog]+1)*
                sizeof(char*)))==NULL) outof_memory(textb[0]);
            if ((PSflag[tog]=(char**)realloc(PSflag[tog],(PScount[tog]+1)*
                sizeof(char*)))==NULL) outof_memory(textb[0]);
            if ((PS[tog][PScount[tog]]=(char*)calloc(bytesL,sizeof(char)))
                ==NULL) outof_memory(textb[0]);
            if ((PSflag[tog][PScount[tog]]=(char*)calloc(bytesL,
                sizeof(char)))==NULL) outof_memory(textb[0]);
            /* copy temp arrays to top of pre-image stack */
            memcpy(PS[tog][PScount[tog]],PStemp,bytesL);
            memcpy(PSflag[tog][PScount[tog]],flagtemp,bytesL);
        }
    }
}
/* count pre-images due to wildcards */
if (see_ppstack && PScount[tog]-pp_hist[0]>0){

```



```

pp_wild=PScount[tog]-pp_hist[0];
pp_wild_tot+=pp_wild;
pp_startx--pp_bartotw;
setfillstyle(INTERLEAVE_FILL,red);
pp_temp=pp_starty-pp_heighty-1
-(signed short)vgax(120)*(float)pp_wild/pp_ymax;
if (pp_temp < pp_starty0-pp_depthy+1)
    pp_temp=pp_starty0-pp_depthy+1;
bar(pp_startx,pp_temp,
    pp_startx+pp_barwidth,pp_starty-pp_heighty-1);
}
if (see_ppstack==2){
tempx=getx;
tempy=gety;
sprintf(textb[0],"abandon pause-p, see next-default:");
draw_prompt_box(1);
myouttext(textb[0]);
input_string(1,0);
if (is[0]=='p'){
    sprintf(textb[0],"partial pre-image histogram: start pause-p");
    draw_prompt_box(1);
    outtext(textb[0]);
    see_ppstack=1;
}
moveto(tempx,tempy);
}
for (zu=1;zu<=PScount[tog];zu++){
    /* check for cycle repeat, rotation (from all 0s or 1s),
       if ok add, prin */
    if (check_pre_image(PS[tog][zu])==1) continue;
    accept_pre_image(PS[tog][zu]);
}
}
for (zz=0;zz<=1;zz++){
    for (zu=0;zu<=PScount[zz];zu++){
        free(PS[zz][zu]);
        free(PSflag[zz][zu]);
    }
}
return;
}

```

## Appendix 6.4.

The function in DDLab (C code) for computing the Z parameter from a rule table (also the  $\lambda$  parameter and  $\lambda_{\text{ratio}}$ ). For context refer to chapter 4.12-13.

```

/***** compute Z parameter *****/
void z_parameter_all(char nhoodz, char showz, char *thisrule){
    float *ln,*notln,
        temp, onecount;
    signed short *ncount,
        tuple, tup, not, tablemax, tablebytes, i, x, y, z,
        revtable, q, sod, tempx, tempy;
    char *temprule,
        *revrule; /* for re-arranging bits from right to left <--*/

    tempx=getx;
    tempy=gety;
    tablemax=(signed short)pow(2,(double)nhoodz)-1;
    tablebytes=(signed short)ceil(((double)tablemax+1)/8);

    if ((ln=(float*)calloc(nhoodz+1,sizeof(float)))==NULL)
        outof_memory("ln");
    if ((notln=(float*)calloc(nhoodz+2,sizeof(float)))==NULL)
        outof_memory("notln");
    if ((ncount=(signed short*)calloc
        (nhoodz+1,sizeof(signed short)))==NULL)
        outof_memory("ncount");
    /* to hold rule for standard left-> right algorithm */
    if ((temprule=(char *)calloc
        (tablebytes,sizeof(char)))==NULL) outof_memory("temprule");
    if ((revrule=(char *)calloc
        (tablebytes,sizeof(char)))==NULL) outof_memory("revrule");

    /* re-arrange rule table for zparam right to left <--
    by positioning acc to reflected nhoods, this allows calc of
    Z-right by same method as Z-left, ie take the nhood=3 rule table...

    left->right
    7 6 5 4 3 2 1 0 rule table index= dec nhoods
    : : : : : : :
    1 1 1 1 0 0 0 0 ..2 nhood index
    1 1 0 0 1 1 0 0 ..1
    1 0 1 0 1 0 1 0 ..0
    * * * * * * * * ..outputs

    \_/_ \_/_ \_/_ \_/_ ..deterministic permutation templates
    \_/_ \_/_ ..ignore 0
    \_/_ \_/_ ..ignore 0,1

```

```

left<-right
7 6 5 4 3 2 1 0  same rule table index
7 3 5 1 6 2 4 0  reflected nhoods
: : : : : : : :
1 0 1 0 1 0 1 0  ..2  nhood index
1 1 0 1 1 1 0 0  ..1
1 1 1 1 0 0 0 0  ..0
* * * * * * * *  ..outputs

┌───┐ ┌───┐ ┌───┐ ┌───┐  ..deterministic permutation templates
└───┘ └───┘ └───┘ └───┘

┌───┐ ┌───┐  ..ignore 2
└───┘ └───┘

┌───┐ ┌───┐
└───┘ └───┘  ..ignore 2,1 */

for (i=tablemax;i>=0;i--){
  revtable=0;
  sod=i;
  for (q=nhoodz-1;q>=0;q--){
    if (sod/(signed short)pow(2,(double)q)==1){
      revtable+=(signed short)pow(2,(double)(nhoodz-1-q));
      sod-=(signed short)pow(2,(double)q);
    }
  }
  set_bit(revrule,i,get_bit(thisrule,revtable));
}
setcolor(lightred);
notln[nhoodz+1]=1;
/* z=1..left->right, z=2 equivalent to left<-right */
for (z=1;z<=2;z++){
  if (z==1) memcpy(temprule,thisrule,tablebytes);
  if (z==2) memcpy(temprule,revrule,tablebytes);
  tuple=2; /*size of template 2,4,8,16,...*/
  /* count template matches */
  for (tup=1;tup<=(tablemax+1)/2;tup++){
    tuple=(signed short)pow(2,(double)tup);
    /* for template size 4,8,16,... */
    for (x=0;x<=tablemax+1-tuple;x+=tuple) {
      if (tup>1 && nhoodz>2){
        for (y=1;y<=(tuple/2)-1;y++){
          if (get_bit(temprule,x)!=get_bit(temprule,x+y)) goto skip1;
          if (get_bit(temprule,x+tuple/2)!=
              get_bit(temprule,x+(tuple/2)+y)) goto skip1;
        }
      }
      /* for template size 2 */
      if (get_bit(temprule,x)==get_bit(temprule,x+tuple/2))
        goto skip1;
      /* count of template matches */
      ncount[nhoodz-(tup-1)]+=1;
      skip1:;
    }
  }
}

```

```

/* prob det = (count/max count) * not prob before */
ln[nhoodz-(tup-1)]=((float)ncount[nhoodz-(tup-1)]
                /((tablemax+1)/tuple))*notln[nhoodz-(tup-2)];
/* calc not prob */
temp=0;
for (not=1;not<=tup;not++){
    temp+=ln[nhoodz-(not-1)];
}
notln[nhoodz-(tup-1)]=1-temp;
}
zparam[z]=0;
for (i=nhoodz;i>=1;i--) zparam[z]+=ln[i];
for (i=nhoodz;i>=1;i--){
    ncount[i]=0;
    ln[i]=0;
    notln[i]=0;
}
}
/* print details */
if (zparam[1]>zparam[2]) zparam[0]=zparam[1];
else zparam[0]=zparam[2];
onecount=0;
for (x=0; x<=tablemax; x++) {
    if (get_bit(thisrule,x)==1) onecount=onecount+1;
}
if (onecount<=(tablemax+1)/2) lratio = onecount/((tablemax+1)/2);
else lratio=(tablemax+1-onecount)/((tablemax+1)/2);
lambda=onecount/(tablemax+1);
/*calculate canalizing inputs*/
canalyzing(nhoodz,0,thisrule);

if (showz){
    draw_box(-1,Maxx/2,vgay(27),Maxx-2,vgay(55),1,0);
    moveto(Maxx/2+vgax(3),vgay(29));
    /* show canalizing inputs */
    sprintf(textb[0],"C=%d/%d",canaltotal,nhoodz);
    myouttext(textb[0]);
    if (canaltotal && nhoodz>1){
        myouttext("=");
        for (z=nhoodz-1;z>=0;z--){
            if (get_bit(canal,z)==1){
                sprintf(textb[0],"%d",z);
                myouttext(textb[0]);
            }
            else myouttext("*");
        }
    }
}
myouttext(" ");

/* show Z and lambda */
sprintf(textb[0],"z->=%g z<--=%g",zparam[1],zparam[2]);
sprintf(textb[1],"lda=%g",lambda);
sprintf(textb[2],"lda-ratio=%g P=%g z-param=%g",
        lratio,.5+(1.0-lratio)/2,zparam[0]);

```

```
    outtext(textb[0]);
    moveto(Maxx/2+vgax(3), gety+CHeight);
    outtext(textb[1]);
    outtext(textb[2]);
}
setcolor(white);
free(ln);
free(notln);
free(ncount);
free(temprule);
free(revrule);
moveto(tempx, tempy);
}
```

## Appendix 7

### Discrete Dynamics Lab

The software that was used for the examples, results, data and figures in this thesis was written by the author. The software and documentation is available on the Internet from the DDLab home page at <http://www.santafe.edu/~wuensch/ddlab.html> (Wuensche 1996).

A detailed description of DDLab is given in Appendix 7.1, which contains the introductory sections of the DDLab manual. The main body of the manual, including the Program Reference, are not included in the appendix for reasons of length, except for the section on network size limitations and time issues in appendix 7.2.

#### Appendix 7.1.

#### DDlab Manual

##### Index

##### Overview

- A1. Introduction
- A3. Presentation options
- A4. Filing and Printing
- A5. Mutations
- A6. Statistical measures and data
- A7. Reverse algorithms
- A8. Sculpting attractor basins
- A9. Hardware and software requirements
- A10. References

##### Quick Start Instructions

1. Downloading, unzipping and running  
(*sections below are not include in the appendix*)
2. The user interface.
3. Quick start examples

##### Program Reference

#### Appendix 7.2

#### Network size limitations (and time issues)

(chapter 6.3 of the DDLab operating manual)

For context refer to chapter 4.4 (CA)

and chapter 5.5 (RBN).

**Appendix 7.1 Discrete Dynamics Lab - manual**  
*(Index, Overview and section 1 only)*

# ***DDLab manual***

## **Discrete Dynamics Lab Cellular Automata - Random Boolean Networks.**

March 1996 (copyright © Andrew Wuensche 1993-96)

Santa Fe Institute  
 and The University of Sussex (COGS)  
 wuensch@santafe.edu

## **Index.**

---

### **Overview**

- A1. Introduction**
  - A3. Presentation options**
  - A4. Filing and Printing**
  - A5. Mutations**
  - A6. Statistical measures and data**
  - A7. Reverse algorithms**
  - A8. Sculpting attractor basins**
  - A9. Hardware and software requirements**
    - A9.1. DOS Platform.
    - A9.2. UNIX/XWindows platform.
    - A9.3. Mac platform.
  - A10. References**
- 

### **Quick Start Instructions**

---

#### **1. Downloading, unzipping and running.**

---

#### **2. The user interface.**

- 2.1. Input.
  - 2.2. Backtrack.
  - 2.3. Changing the graphics set-up.
- 

#### **3. Quick start examples**

- 3.1 Basin of attraction fields.
- 3.2. "Backwards" space-time patterns, and state-space matrix
- 3.3. Basins of attraction fields for a range of network sizes
- 3.4 A single basin of attraction.
- 3.6. A subtree.
- 3.7. Space-time pattern only, 1d
- 3.8. Space-time pattern only, 2d (game of life).
- 3.9. "Screen-saver" demo.

---

## **Program Reference**

---

### **4. Starting DDLab, memory, and the user interface.**

- 4.1. Title bar.
- 4.2. DDLab version, graphics and mouse status.
- 4.3. Memory available.
- 4.4. Virtual Memory.
- 4.5. Prompts.

---

### **5. The first prompt (and graphics setup).**

- 5.1. Field, or a run requiring a seed.
- 5.2. Graphics set-up.
  - 5.2.1. VGA and SVGA.
  - 5.2.2. Black or white background.
- 5.3. Options.
- 5.4. Random number seed.
- 5.4. Exit DDLab.

---

### **6. Network size, 1d.**

- 6.1. Setting range of sizes.
- 6.2. Setting one network size.
- 6.3. Network size limitations.

---

### **7. Cell scale.**

---

### **8. Neighbourhood.**

- 8.1. Setting neighbourhood size or mix.
- 8.2. Setting the neighbourhood mix.
- 8.3. Loading a neighbourhood mix file.
- 8.4. Setting the neighbourhood mix by hand.
- 8.5. Setting the neighbourhood mix at random.
- 8.6. Reviewing the neighbourhood mix.
  - 8.6.1. Reviewing the neighbourhood mix in large networks.
  - 8.6.2. Jumping to a new cell index.
  - 8.6.3. Saving the neighbourhood mix.
  - 8.6.4. Other options to review and alter the neighbourhood mix.

---

### **9. Wiring.**

- 9.1. Network geometry
- 9.2. The pseudo-neighbourhood.
- 9.3. Setting the wiring.

---

### **10. Quick wiring settings.**

- 10.1. Loading the wiring scheme.
- 10.2. Regular 1d wiring.
- 10.3. Regular 2d wiring
  - 10.3.1. Conserving wiring memory in regular 2d networks
  - 10.3.2. Setting 2d network size.
- 10.4. Random 1d wiring.



---

## 11. Special wiring.

- 11.1. Set network geometry.
- 11.2. Set special wiring.
- 11.3. Special wiring, regular 1d (treat as random).
- 11.4. Special wiring, regular 2d.
- 11.5. Special wiring, random.
  - 11.5.1. Confining random wiring to a set zone.
  - 11.5.2. Self wiring.
  - 11.5.3. Distinct wiring.
- 11.6. Wiring by hand.
  - 11.6.1. Wiring by hand in large networks.

---

## 12. Reviewing Wiring.

- 12.1. The wiring matrix, and 1d or 2d wiring graphic.
- 12.2. The wiring matrix.
  - 12.2.2 Revise wiring, wiring matrix.
  - 12.2.3 Revise by hand.
  - 12.2.4 Randomly rewire.
  - 12.2.5 Reset wiring options.

---

## 13. Rules.

- 13.1. The rule-table.
  - 13.1.2. The totalistic code table
  - 13.1.3. Totalistic rules.
- 13.2. Rule table matrix.
- 13.3. Single rule or rule mix.
  - 13.3.1. A rule mix with just one rule.
- 13.4. Rule mix.
- 13.5. Rule mix options.
  - 13.5.1. Rule mix with no limit.
  - 13.5.2. By hand.
  - 13.5.3. Majority.
  - 13.5.4. Majority + end bits flipped.
  - 13.5.5. Random.
  - 13.5.6. Random + canalising.
  - 13.5.7. Setting a limited sub-set of rules for random selection.
- 13.6. Setting the selection method for a single rule (by hand).
  - 13.6.1. Methods for setting a rule.
  - 13.6.2. The rule window.
- 13.7. Setting the majority rule.
- 13.8. Setting the "game of life" rule.
- 13.9. Setting the rule in decimal.
- 13.10. Setting the rule at Random.
- 13.11. Setting the rule as bits.
  - 13.11.1. Setting bits with the mouse.
  - 13.11.2. Setting bits from the keyboard.
- 13.12. Setting the rule in hex.
- 13.13. Repeating the last rule.
- 13.14. Loading Rules.
  - 13.14.1. Single rule file encoding.
  - 13.14.2. Conflicts in neighbourhood size between network and file.
- 13.15. Rule mix selection by hand.
  - 13.15.1. Changing the cell index or selection method.

- 13.15.2. Changing the cell index.
- 13.15.3. Changing the rule selection method..
- 13.16. Automatic saving of last rule.
- 13.17. Transforming the rule, and setting canalising inputs.
  - 13.17.1. *Complimentary* transformation.
  - 13.17.2. Equivalent rule by *negative* transformation.
  - 13.17.3. Equivalent rule by *reflection* transformation.
  - 13.17.4. Equivalent rules with a greater neighbourhood, size, *k*.
  - 13.17.5. Setting canalising inputs.
  - 13.17.6. Saving the transformed rule.

#### 14. Reviewing wiring/rules.

- 14.1. 1d or 2d wiring graphic and rule scheme.
- 14.2. Wiring graphic, 1d.
  - 14.2.1. Moving or jumping between cells, 1d wiring graphic.
- 14.3. Wiring graphic, 2d.
  - 14.3.1. Moving or jumping between cells, 2d wiring graphic.
- 14.4. Options, 1d and 2d wiring graphic.
  - 14.4.1. Revising the wiring.
  - 14.4.2. Revising the rule.
  - 14.4.3. Computing the (weighted) average  $\lambda$  and Z parameters.
  - 14.4.4. Learning pre-images without attractor basins.
  - 14.4.5. Printing the wiring graphic window.
  - 14.4.5. Options.
- 14.5. Revise wiring with the 1d or 2d wiring graphic.
  - 14.5.1 Changing the neighbourhood size.
  - 14.5.2 Hand rewiring.
  - 14.5.3. Random wiring.
  - 14.5.4. Random special wiring.
  - 14.5.5. Regular 1d wiring.
  - 14.5.6. Regular 2d wiring.
- 14.6. Revise rules with the 1d or 2d wiring graphic.
- 14.7. Rule/wiring scheme: saving, loading, or printing (to a printer or file).
  - 14.7.1. Save/load/print selection.
  - 14.7.2. Mixed rules and non-local wiring.
  - 14.7.3. One rule and non-local wiring.
  - 14.7.4. Mixed rules and local wiring.
  - 14.7.5. Printing to a printer or file.
  - 14.7.6. Wiring/rulemix file names.
  - 14.7.7. Wiring/rulemix encoding..
  - 14.7.8. Wiring/rulemix file loading conventions.
  - 14.7.9. Neighbourhood mix loading conventions.

#### 15. Seed.

- 15.1. Setting the selection method for the seed.
- 15.2. Seed bit pattern display.
- 15.3. Rotating the seed.
- 15.4. Setting the seed in decimal.
- 15.5. Setting the seed at random.
- 15.6. Setting the seed in hex.
- 15.7. Setting the seed set as bits, drawing with the mouse.
  - 15.7.1. 1d networks (shown as 1d).
  - 15.7.2. 1d networks (shown as 2d).
  - 15.7.3. 2d networks.

- 15.7.4. Setting/deleting bits with the mouse or keyboard.
- 15.8. Repeating the seed..
- 15.9. Loading a seed.
  - 15.9.1. Seed file encoding and loading conventions.
  - 15.9.2. Seed file, 1d.
  - 15.9.3. Seed file, 2d.
- 15.10. Saving a seed.

## 16. Output parameters

- 16.1. The first output parameter prompt.
  - 16.1.1. Accept all output parameter defaults.
  - 16.1.2. Space-time patters only.
  - 16.1.3. Restore all output parameter defaults, or just layout defaults.
  - 16.1.4. Jump to layout of attractor basins.
  - 16.1.5. Jump to display of attractor basins, and data
  - 16.1.6. Jump to mutation.

## 17. Set various output parameter options.

- 17.1. State-space matrix scatter-plot.
  - 17.1.1. Display all states.
  - 17.1.2. Display attractor states only.
  - 17.1.3. Change matrix size.
  - 17.1.4. Change start colour.
- 17.2. In-degree frequency histogram.
  - 17.2.1. In-degree frequency cut-off.
  - 17.2.2. In-degree frequency window.
  - 17.2.3. Data shown.
  - 17.2.4. Options.
  - 17.2.5. Re-scaling the x-axis.
  - 17.2.6. Re-scaling the y-axis.
- 17.3. States with majority of 1s or 0s.
- 17.4. "Screen save" demo.
- 17.5. Graphs: G-density, Z and  $\lambda$ ratio.
  - 17.5.1. G-density against Z and/or  $\lambda$ ratio.
  - 17.5.2. Z against  $\lambda$ ratio.
- 17.6. "Backwards" space-time patterns.
- 17.7. Scroll space-time patterns.

## 18. Graphic conventions for attractor basins.

- 18.1. Network states, nodes.
- 18.2. Attractor cycles.
- 18.3. Transient trees.
- 18.4. Transient tree colors.
- 18.5. Transient trees for the states all 0s or all 1s.
- 18.6. Subtree only.

## 19. Layout of attractor basins.

- 19.1. The layout preview.
- 19.2. The first layout prompt.
  - 19.2.1 Reset all layout defaults.
  - 19.2.2 Reset layout for learning.
  - 19.2.3. Basin scale, attractor radius.
- 19.3. Basin start position.

- 19.4. Show the field as successive basins.
- 19.5. Basin spacing for fields.
- 19.6. Select minimum right border width.
- 19.7. Amend the layout after each basin or tree.
  - 19.7.1. Amend the spacing and right border after each basin.
  - 19.7.2. Amending the next position (and spacing) after each basin.
- 19.8. Amend the spacing increase for successive fields.
- 19.10. Revise layout parameters.

---

## 20. Display of attractor basins, and data.

- 20.1. Compression of equivalent CA dynamics.
  - 20.1.1. Suppress compression.
  - 20.1.2. Suppress copies of trees (and subtrees).
- 20.2. Node display.
  - 20.2.1. Show bit pattern nodes for 1d networks in 2d.
  - 20.2.2. Alter node label size, decimal or hex.
  - 20.2.3. Alter node size, bits.
- 20.3. Highlight one state in each attractor.
- 20.4. Pause, and data.
  - 20.4.1. Data on sub-trees.
  - 20.4.2. Data on a subtree from a uniform state.
  - 20.4.3. Data on trees.
  - 20.4.4. Data on basins.
  - 20.4.5. Pause after each field:
  - 20.4.6. Pause after each basin or tree:
- 20.5. Print data to printer.
- 20.6. Save data to file.
- 20.7. Format of printed or saved data.
  - 20.7.1. Network data.
  - 20.7.2. Basin field data (no tree data).
  - 20.7.3. Basin field data, including tree and subtree data.
  - 20.7.4. Single basin data file.
  - 20.7.5. Subtree data file.

---

## 21. Mutation.

- 21.1. The first mutation prompt.
- 21.2. Flip bits.
  - 21.2.1. Alter the number of bits to flip.
  - 21.2.2. Alter the percentage of bits to flip.
- 21.3. Special rule mutation.
  - 21.3.1. Mutate by the rule or code number.
  - 21.3.2. Assign a random rule or code number.
  - 21.3.3. Mutate by bitflip.
- 21.4. Mutate wiring.
  - 21.4.1. Alter the number of wires to move.
  - 21.4.2. Alter the percentage of wires to move.
- 21.5. No pause before next mutant.

---

## 22. Run backwards for subtree, or forwards for a single basin.

- 22.1. Subtree: Run forwards before running backwards.
  - 22.2. Single basin repeat check limit.
    - 22.2.1. Alter repeat check limit, or forwards only.
-

---

### 23. Final options: subtree, single basin or basin of attraction field.

- 23.1. Local wiring.
- 23.2. Non-local wiring.
- 23.3. Partial pre-image stack, local wiring.
- 23.4. Partial pre-image histogram, non-local wiring.
- 23.5. The exhaustive testing reverse algorithm.
  - 23.5.1. Load exhaustive pairs.
  - 23.5.2. Computing exhaustive pairs.
  - 23.5.3. Saving exhaustive pairs.

---

### 24. Final options for space-time patterns (forwards only).

- 24.1. Color cells by value or neighbourhood.
- 24.2. Pause when screen full.
- 24.3. Frozen generation size.
- 24.4. Analysis.
  - 24.4.1. Pattern density.
  - 24.4.2. Lookup frequency and entropy.
- 24.5. Analysis generation size.
- 24.6. State-space matrix scatter-plot (forwards only).

---

### 25. Changing settings "on the fly" (attractor basins).

- 25.1. Interrupting attractor basins.
  - 25.1.1. Backtrack through prompts.
  - 25.1.2. Abandoning a tree and continuing with the next tree.
  - 25.1.3. Other options.
- 25.2. Key hit attractor basin options.
- 25.3. Attractor basin complete options.
- 25.4. Further attractor basin complete options.
  - 25.4.1. Rule options.
  - 25.4.2. Seed options.

---

### 26. Changing settings "on the fly" (space-time patterns only).

- 26.1. The "Interrupt Key Index" (space-time patterns only).
  - 26.2.1. *change rule table*
  - 26.1.2. *change wiring*
  - 26.1.3. *change seed/size*
  - 26.1.4. *presentation*
  - 26.1.5. *2d space-time pattern*
  - 26.1.6. *highlight frozen cells*
  - 26.1.7. *analysis*
  - 26.1.8. *state-space matrix*
  - 26.1.9. *index, pause, quit*
- 26.2. The "Interrupt Key Index", some further details
  - 26.2.1. *change rule table*
  - 26.2.2. *change wiring*
  - 26.2.3. *2d space-time pattern*
  - 26.2.4. *highlight frozen cells*
  - 26.2.5. *analysis*
- 26.3. Pause space-time pattern.
- 26.4. Rule details (space-time patterns only).
- 26.5. Further space-time pattern options .
  - 26.5.1. Screen options.

- 26.5.2. Rule options.
- 26.5.3. Seed options.
- 26.5.4. Other options.
- 26.5.5. Further options, first backtrack.

## **27. Learning, forgetting, and highlighting.**

- 27.1. Highlighting states.
- 27.2. Default attractor basin layout for learning.
- 27.3. Selecting the wiring graphic.
- 27.4. Selecting the learn/highlight window.
- 27.5. Select the "target" state.
- 27.6. Select a set of states as pre-images.
  - 27.6.1. An arbitrary list of aspiring pre-images.
  - 27.6.2. Pre-images according to Hamming distance.
  - 27.6.4. Odd or even parity.
  - 27.6.5. Repeat previous selection.
- 27.7. Review target state and aspiring pre-images.
- 27.8. Learn, forget, or highlight only.
- 27.9. Learning/forgetting by wire moves or bit-flips.
  - 27.9.1. Learning/forgetting by bit-flips.
  - 27.9.2. Learning/forgetting by wire-moves.
- 27.10. Highlighting options.
- 27.11. Learning complete.

## **28. Filing**

- 28.2. Saving and Loading
- 28.3. Changing the directory
- 28.4. List files
- 28.5. File encoding
  - 28.5.1. SCREEN (image)
  - 28.5.2. NHOOD MIX
  - 28.5.3. WIRING ONLY, RULE SCHEME ONLY, WIRING-RULE SCHEME
  - 28.5.4. SINGLE RULE
  - 28.5.5. SEED
  - 28.5.6. DATA
  - 28.5.7. EXHAUSTIVE (pairs)

## **Overview.**

### **A1. Introduction**

DDLab is an interactive graphics program for research into the dynamics of finite binary networks, from Cellular Automata (CA) to random Boolean networks, including their attractor basins. The program is relevant to the study of complexity, emergent phenomena, neural computation and aspects of theoretical biology such as modelling gene regulatory networks. The results presented in (Wuensche 1992-1995) may be implemented with this program.

Versions of the program run on the following platforms: DOS-PC (vga and svga), UNIX/XWindows-Sun-Spark, and the Mac. This manual covers all three versions. The various versions function in essentially the same way, though aspects of the graphics presentation may be slightly different.

Using a flexible user interface, a network can be set up with any architecture ranging from regular CA (1d or 2d with periodic boundary conditions) on the one hand, and random Boolean networks (disordered CA) on the other. The latter have arbitrary connections, and rules which may be different at each site. The neighbourhood (or "pseudo-neighbourhood") size may be set from 1 to 9, and the network may have a mix of neighbourhood sizes.

The program is able to compute the global dynamics of networks as well as run the usual forward dynamics to show space-time patterns. For global dynamics the network is run "backwards" to generate a pattern's (or state's) predecessors and reconstruct its branching sub-tree of ancestor patterns. All "garden of Eden" states, the leaves of the sub-tree, may be disclosed. Sub-trees, basins of attraction or the entire basin of attraction field (referred to collectively as "attractor basins") can be displayed as a directed graph or set of graphs in real time, with many presentation options.

It can be argued that attractor basins represent the network's "memory" by the hierarchical categorisation of state-space. Each basin is categorised by its attractor and each sub-tree by its root. Learning/forgetting algorithms allow attaching/detaching sets of states as predecessors of a given state by automatically mutating rules or changing connections. This allows sculpting the basin of attraction field to approach a desired scheme of hierarchical categorisation. The attractor basins of "random mappings" (with or without various biases) can also be generated.

Whereas large systems sizes may be run forward to look at space-time patterns, or backwards to look at subtrees, much smaller sizes (say less than 18) are appropriate for the entire basin of attraction field given that state-space grows exponentially with system size. The program's upper limit for basin of attraction fields is 32; there is no limit for single basins or sub-trees. Larger sizes may be tried, but may impose unacceptable time, memory or display constraints. Sub-trees may be generated for much larger systems for rules having a low in-degree.

The network's parameters, and the graphics display and presentation options, can be very flexibly set, reviewed and altered. Changes can be made "on the fly", including mutations to rules, connections or current state. 2d networks (including the "game of life" (Conway 1982) or any mutation thereof) can be displayed as a space-time pattern in a 3d isometric projection.

Various quantitative, statistical and analytical measures and data on both forward dynamics and attractor basin topology are made available in DDLab, as well as various global parameters of rules and network architecture. These measures and data (mostly presented graphically) include the following:

- The  $\lambda$  (or  $P^*$ ) parameter and the  $Z$  parameter.
- The frequency of canalising "genes" and inputs. This can be set to any arbitrary level.
- Various measures on forward dynamics such as pattern density, frozen islands, pattern difference between two networks\*, the Derrida plot\*, rule-table lookup frequency and entropy, and the variance of the entropy\*, which allows ordered, complex and chaotic rules to be discriminated automatically\*.
- Various global measures on the topology of attractor basins including garden-of-Eden density and a histogram of in-degree frequency.
- A scatter-plot of state-space.
- (items marked \* are not yet covered in the program reference #4)

The network architecture, states, data, and the screen image can be saved and loaded in a variety of tailor-made file formats.

#A1-10 gives an overview of the program. #1-3 give some "quick start" examples. These should be tried initially to get the flavour of the DDLab. #4-28 is a detailed reference manual

relating to the DOS version of DDLab, and omitting a number of new functions - however this should provide an adequate guide until the manual is updated.

For further background on the attractor basins of CA and random Boolean networks, and their implications, refer to Wuensche (1992-1995), Kauffman (1993) and other references listed in #A10. DDLab is in the process of continual development. New features are being added in response to various research needs. Some aspects of this manual may be out of date or not applicable to particular platforms.

## A2. Network parameters

DDLab's user interface allows setting, viewing and amending network parameters by responding to prompts or accepting defaults. The prompts present themselves in a main sequence and also in a number of context dependent prompt windows. It is possible to backtrack to previous prompts in the sequence, and in some cases to skip forward. A flashing cursor indicates the prompt. "Return" (or the left mouse button) steps forward through the prompts, "q" (or the right mouse button) backtracks, or interrupts a run.

CA rules (or totalistic codes) are set in decimal, hex, as a lookup table bit pattern (using the mouse), at random or loaded from a file. The "game of life" and the "majority" rule can also be selected. Rules may be changed into their equivalents (by reflection and negative transformations), and transformed into equivalent rules with larger neighbourhoods. The latter is useful to achieve finer mutations of interesting rules. The frequency of canalising "genes" or inputs in a network can be set to any arbitrary level, useful in modelling gene regulatory networks.

Wiring is set as regular 1d or 2d for the given neighbourhood size, at random, with a "spread sheet" to set or alter individual couplings by hand, or a wiring scheme may be loaded from a file. In most cases regular 2d wiring defines a square grid on the torus, and includes the von Neumann and Moore neighbourhoods of 5 and 9 cells. However the 7 cell regular 2d neighbourhood is wired to define a triangular grid. Non standard wiring can be constrained in various ways, including confining it to a local patch of cells with a set diameter.

The network parameters can be scrutinised and amended in a 1d or 2d graphic display format, or a "spread sheet" format, showing each cell's particular wiring and rule, using the arrow keys to move between cells.

An initial network state is required as a seed for a forwards or backward run (to seed a subtree or single basin). A basin of attraction field does not require a seed. The seed is set in decimal or hex, as a bit

pattern in 1d or 2d (using the mouse), at random, or loaded from a file. The bit pattern method is a mini drawing program, using the mouse (or keyboard) to set cell values (0,1). It is particularly useful for 2d patterns.

It is possible to create sets of independent sub-networks within the total network\*. This is done by saving smaller networks parameters to a file, then loading them at appropriate positions in the total network, tiling a network with sub-networks in the case of 2d. Alternatively a network can be automatically duplicated to create a total network made up of two identical sub-networks\* (repeating this results in 4,8,etc). This is useful to see the difference pattern, or damage spread, between two networks from similar initial states. Sub-networks created by either method may be re-wired to influence each other, for instance to create a network of (sparsely connected) sub-networks.

## A3. Presentation options

Many options are provided for the presentation of attractor basins and space-time patterns. Again, many of these settings can be changed "on the fly", including the scale and colour of space-time patterns. Cells in space-time patterns are coloured according to their value (0,1) or alternatively



according to their neighbourhood at the previous time step (the entry in the look-up table that determined the cell's value). A key press will toggle between the two. Alternatively, the colour presentation can be set to highlight cells that have not changed in the previous  $x$  generations, where  $x$  can be set to any value. The development of such *frozen clusters*, which may link up into *percolating walls*, is associated with canalising "genes", and underlies Kauffman's "random Boolean network" model of gene regulatory networks (Kauffman 1993).

A 1d space-time pattern is presented in successive vertical sweeps, or may be scrolled. 2d networks (including the "game of life" or any mutation thereof) can be displayed as a space-time pattern in a 3d isometric projection (as if looking up at a glass lift shaft). The projection can be toggled between 3d, 2d and 1d (which shows a series of slices).

For attractor basins, options allow the selection of the basin of attraction field, a single basin (from a selected seed), or a sub-tree (also from a seed). A random seed is likely to be a garden of Eden state. For sub-trees, an option is therefore offered to run the network forward a given number of steps to a new seed before running backward. This guarantees a sub-tree with at least that number of levels.

Options (and defaults) are provided for the layout of attractor basins, their size, position, spacing, and type of node display (as a spot, in decimal, hex or a (1d or 2d) bit pattern). Regular 1d and 2d CA produce attractor basins where sub-trees and basins are equivalent by rotational symmetry. This allows "compression" of basins (by default) into non-equivalent prototypes, though compression can be suppressed. Attractor basins can be generated for a given system size, or automatically for a range of sizes. As attractor basins are generating, the reverse space-time pattern can be displayed either by sweeps or scrolled. The 1s in each state are shown in black, 1s in predecessors (if any) in red.

An attractor basin run can be set to pause to see data on each transient tree, each basin, or each field. Normally the run will pause before the next "mutant" attractor basin, but this pause may be turned off to create a continuous demo of new attractor basins. A "screensave" demo option shows new basins continually growing in random position. At any time, a space-time pattern or attractor basin run can be interrupted to pause, save or print the screen image, or backtrack through options.

In the DOS version the graphics will start up in VGA (640x480) unless the program parameter "-h" is set (for high resolution SVGA (1024x768). The resolution can subsequently be toggled between VGA and SVGA (SVGA requires an appropriate monitor, graphics card and driver, and the mouse is not visible though mouse functions work). In the XWindows version the DDLab window starts up at 1024x768 but can be resized, moved and iconised in the usual way.

The background color is by default white, but can be reset to black either from within the program or by setting the program parameter "-b", i.e. for a black background start DDLab by entering "ddlab -b".

---

## A4. Filing and Printing

Network parameters and states can be saved and loaded with default file extensions (and default file names) for the following: rules, rule-schemes, wiring-schemes, wiring/rule schemes, neighbourhood mix, and network state.

The screen image is saved and loaded using a home made compressed format. In DOS, to save the image in a standard format (and print to any printer), use a "stay resident screen grabber". A number a specialist screen grabbers are available, and others are part of "paint" programs. Alternatively run DDLab as a DOS application in Microsoft Windows and use their "paint" screen grabber. The screen image can be printed at any time directly from DDLab (in DOS reliably only to the Epson MX-82 dot matrix printer, or to the Cannon BJC 4000 bubble jet - printing to other printers is worth a try but is unpredictable). In the XWindows versions the screen can be printed from options within DDLab, or alternatively use XView to grab the image and print.

Detailed data on attractor basins can be automatically saved (also printed). A file of "exhaustive pairs", made up of each state and its successor, can be created for very fast generation of the basin of attraction field of a random Boolean network. Various data on space-time patterns such as the entropy variance for different CA rules can be automatically saved and sorted to create a data base of complex rules, those featuring "gliders" or other large scale emergent structure\*.

---

## A5. Mutations

Network parameters (as well as graphics presentation) can be altered "on the fly". For example during forward iteration various key presses allow mutations to rules, wiring or current state; the network size can be increased or decreased, and the graphical display of quantitative and analytical data can be changed. A number of 1d "complex" rules (with glider interactions) for 5-7-neighbours are provided as files and can be set with a button press. Examples of some of these are presented in (Wuensche 1994b,1995).

When attractor basins are complete, a key press will regenerate the attractor basin of a mutant network. Various mutation options can be pre-set including random bit flips in rule tables, or random rewiring of a given number of wires. Sets of states can be specified and highlighted in the attractor basin to see how mutations affect their distribution.

---

## A6. Quantitative, statistical and analytical measures

Various measures and data (also presented graphically) include the following:

### Parameters on rules and network architecture:

- The  $\lambda$  (or P\*) parameter and the Z parameter measured on rule look-up tables.
- The frequency of canalising "genes" and inputs. This can be set to any arbitrary level.

### Various measures on forward dynamics (these may be spread over a number of generations):

- A rule-table lookup frequency histogram, which can be toggled between 2d and 3d to include a time dimension.
- The entropy of the lookup frequency over time.
- The variance of the entropy, which allows ordered, complex and chaotic rules to be discriminated automatically\*.
- A plot of mean entropy against the variance of the entropy for a set range of time steps (and initial conditions), for an arbitrarily large sample of CA rules. This plot can be redrawn as a 3d frequency histogram. Ordered, complex and chaotic dynamics are located in different regions allowing a statistical measure of their frequency. In addition the rules can be sorted by entropy variance allowing complex rules to be found automatically\*.
- The pattern density over time.
- Frozen islands, the fraction of "genes" that have not changed over the last n generations (and the fraction of frozen 0s and 1s\*).
- Pattern difference fraction between two networks, or damage spreading\*.
- The "Derrida plot" showing how the Hamming distance of network trajectories from sets of state pairs changes. This indicates if a network is in the ordered or chaotic regime\*.
- A plot of the Hamming distance between successive iterations\*.

### Various measures on global dynamics:

- Garden-of-Eden density and more generally the in-degree frequency of attractor basins (or fragments).

- A scatter-plot of state space (plotting the left half against the right half of each state bit string), using colour to identify different basins, or attractor cycle states.
- (items marked \* are not yet covered in the program reference #4)

Most of these ideas are explained in (Wuensche 1994b,1995).

## A7. Reverse algorithms

There are three different reverse algorithms for generating the pre-images of a network state. The first is specifically for "local" 1d wiring with a homogeneous neighbourhood, such as 1d CA, though the rules may be heterogeneous. This is the most efficient thus fastest algorithm, described in (Wuensche 1992). Furthermore, compression of 1d CA attractor basins by rotation symmetry (see #20.1) speeds up the process.

Any other network architecture, with non-local wiring, will be dealt with by a slower *general* reverse algorithm described in (Wuensche 1994a). A histogram revealing the inner workings of this algorithm can be displayed. Regular 2d CA will use this general reverse algorithm; however compression algorithms will come into play to take advantage of the many rotation symmetries on the torus (unless suppressed, and not for a neighbourhood of 7, the symmetries of a triangular grid on the torus have yet to be resolved).

Both these reverse algorithms generate the pre-images of a state directly; their speed is sensitive to neighbourhood size as well as system size. A neighbourhood of 9, such as the game of life, is slowest. A third reverse algorithm first sets up a list of "exhaustive pairs" of each state in state-space and its successor (this can be saved). The pre-images of states are generated by reference to this list. This method is efficient for basin of attraction fields, or large basins that use up a good proportion of state-space, and additionally is not sensitive to neighbourhood size (the basin of attraction field of the 4x4 "game of life" by this method takes about 30 minutes on my 66MHz 486).

The method used to generate pre-images will be chosen by default, which can be overridden. For example, a regular 1d CA can be made to use either of the two other algorithms for benchmark purposes. The time taken to generate attractor basins is displayed, and for the basin of attraction field, a progress bar indicates the proportion of states in state-space used up so far.

Finally, a "random mapping" can be set up which assigns a successor state at random to each state in state space (rules and wiring previously set are ignored)\*. The attractor basins are reconstructed by reference to this random mapping. The space of such random mappings corresponds to the space of all possible basin of attraction fields and is the super-set of all other deterministic discrete dynamical systems.

## A8. Sculpting attractor basins

Learning and forgetting algorithms allow attaching and detaching sets of states as predecessors of a given state by automatically mutating rules or wiring couplings. This allows "sculpting" the attractor basin to approach a desired scheme of hierarchical categorisation. When an attractor basin run is complete, various "learning" windows allow a "target" state to be set, together with a number of "aspiring pre-images" (predecessors). These may be selected in various ways including by Hamming distance. Learning/forgetting algorithms will attempt to attach the aspiring pre-images to the target state, and can be set for either rule-table bit-flips or wire moves. In fact the bit-flip method cannot fail. In this way, new point and cyclic attractors can be created and subtrees transplanted. The result of learning/forgetting, including side effects, will be apparent in the new attractor basins. The algorithms, and their implications are described in (Wuensche 1994a).

---

## A9. Hardware and software requirements

### A9.1. DOS Platform.

PC-DOS 386 or higher, with VGA or SVGA graphics; maths co-processor, mouse (optional). (ideally a fast 486 with 8MB ram). The code is written in C and compiled with the Watcom C32 compiler (and the Rational Systems DOS extender) giving access to extended memory. The supplied Watcom file "dos4gw.exe" must be in the same directory as "ddlab.exe".

### A9.2. UNIX/XWindows platform.

Compiled for the SUN OS 4.1.

### A9.3. Mac platform.

This is a preliminary version of DDLab ported to the Mac by Simon Fraser. To obtain the Mac version contact Simon at [smfr@santafe.edu](mailto:smfr@santafe.edu).

---

## A10. References

- 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.
- Gutowitz, H.A., ed., (1991) *Cellular Automata, Theory and Experiment*, MIT press.
- Hopfield, J.J. (1982) "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences* 79:2554-2558.
- Kauffman, S.A., (1993) *The Origins of Order, Self-Organization and Selection in Evolution*, Oxford University Press.
- Langton, C.G., (1986) "Studying Artificial Life with Cellular Automata", *Physica D*, 22, 120-149.
- Langton, C.G., (1990) "Computation at the Edge of Chaos: Phase Transitions and Emergent Computation", *Physica D*, 42, 12-37.
- Wolfram, S., ed. (1986) *Theory and Application of Cellular Automata*, World Scientific, 1986.
- Wuensche, A., and M.J. Lesser. (1992) *The Global Dynamics of Cellular Automata; An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.
- Wuensche, A., (1994a) "The Ghost in the Machine; Basin of Attraction Fields of Random Boolean Networks", in *Artificial Life III*, ed C.G. Langton, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.
- Wuensche, A., (1994b) "Complexity in One-D Cellular Automata; Gliders, Basins of Attraction and the Z [Parameter]", Santa Fe Institute Working Paper 94-04-025.
- Wuensche, A., (1994c) "The Emergence of Memory", Cognitive Science Research Paper 346, Univ. of Sussex, to appear in *Towards a Scientific Basis for Consciousness*, eds. S.R. Hameroff, A.W. Kaszniak, A.C. Scott, MIT Press.
- 

## Quick Start Instructions.

---

### 1. Downloading, unzipping and running.

#### Updates to DDLab.

The latest version of DDLab is available from the World Wide Web at "<http://alife.santafe.edu/alife/software/ddlab.html>" the DDLab page on ALife online, or directly by "<ftp://alife.santafe.edu/pub/SOFTWARE/ddlab>".

#### Files to download

ddlab.txt	a brief uncompressed text file about DDLab
ddman_20.zip	The DDLab reference manual. At present the manual is only available as an unillustrated Word for Windows file.
ddlab_20.zip	the DDLab software for DOS.
ddlabx_20.tar.gz	the DDLab software for Unix/XWindows.

**FTP instructions**

If you wish to download these files using FTP, follow these instructions:

```
%          ftp alife.santafe.edu
name:      anonymous
password:  your complete email address
ftp>      cd pub/SOFTWARE/ddlab      (note "SOFTWARE" is upper case)
ftp>      bin                        (or "ftp>asc" to download "ddlab.txt", then get ddlab.txt)
ftp>      get ddlab_20.zip           (for DOS)
          or ftp>                  get ddlab_20.tar.gz       (for UNIX/XWindows)
          or ftp>                  get ddman_20.zip       (the manual in Word for Windows format)
ftp>      quit
```

**DDLab documentation.**

"ddman\_20.zip" will unzip to give the following file:

"ddman.doc" a Word for Windows file with the (un-illustrated) reference manual).

"ddlab.txt" is a text file with a brief introduction to DDLab

**DOS program files.**

"ddlab\_20.zip" will unzip (using pkunzip) to give the following files:

"ddlab.exe" the program  
 "dos4gw.exe" the DOS extender, giving access to extended memory.  
 "dd\_bh.map, dd\_bl.map, dd\_wh.map, dd\_wl.map" 4 image files for titles.  
 some extra files listed below.

Keep these files in the same directory. To run the program in DOS, from this directory enter "ddlab" plus the optional program parameters -b (black background) and -h (svga resolution), i.e. for a black background in svga enter "ddlab -b -h". The "DOS extender" banner will briefly appear, and the program will start.

**UNIX/XWindows program files.**

To uncompress "ddlabx\_20.tar.gz" into a tar archive, type: gunzip "ddlabx\_20.tar.gz". Now, to unpack the tar archive, type: "tar -xf ddlabx\_20.tar". This will create a new directory called "ddlabx\_20", and place the following files into it:

"ddlabx" the program  
 some extra files listed below.

Keep these files in the same directory. To run the program in UNIX/X Windows, enter "ddlabx" from this directory plus the optional program parameters -b (black background).

**Extra program files common to both DOS and UNIX**

Files with "complex" 1d CA rules which feature interacting gliders.

"glider5.r_s"	about 60 complex 5-neighbour rules.
"glider6.r_s"	about 5 complex 6-neighbour rules.
"glider7.r_s"	about 10 complex 7-neighbour rules.
"pento.eed"	the "rpentomeno" 2d pattern, which seeds interesting "game-of-life" dynamics.

Section 2 and 3 give brief "quick start" examples. You are encouraged to try these examples first to get the flavour of DDLab. Section 4 onwards is the detailed program reference, listing all the various options and prompts with explanatory notes.

Note that DDLab may also be set up and run as a non-Windows application under Microsoft Windows.

---

*further sections of the manual not included*  
*except section 6.3. Network size limitations in Appendix 7.2 below*

## Appendix 7.2 Network size limitations.

(section 6.3 of the DDLab operating manual)

For context refer to chapter 4.4 (CA) and chapter 5.5 (RBN).

### 6.3. Network size limitations.

For attractor basins it is best to try out small sizes first, especially for networks other than regular 1d CA. Much larger systems sizes may be run "forward only" for just space-time patterns, or "backwards" to generate a subtree. The maximum sizes DDLab can support are 9999 for 1d, and  $255 \times 255$  for 2d, but in practice smaller sizes are preferable. A good size for 1d space-time patterns for a clear view of other features on the screen is 150-200.

A small network size  $n$  is appropriate for attractor basins as these exist in a state-space that grows exponentially (though the program's upper limit for single basins or subtrees is as above, and  $n=32$  for basin of attraction fields).

As an example, the time to generate the basin of attraction field of some 1d CA rules for a range of neighbourhood,  $k$ , and system size,  $n$ , is noted below. With some notable exceptions, the table shows the time doubling with increasing  $n$ , and by a factor of about 1.6 with increasing  $k$  (on my 66MHz 486 PC),

$n=$	12	13	14	15	16	17	18	
time=	2s	4.9s	8s	16s	32s	1m4s	2m13s	$k=3$ rule (dec) 193
time=	3.8s	7s	13s	27s	50s	1m47s	3m37s	$k=4$ rule (hex) e9 24
time=	5.4s	9.8s	30s	37s	1m18s	2m36s	8m46s	$k=5$ rule (hex) b7 55 d3 d9

A similar exercise for random Boolean networks (wiring and rules selected at random) gave the following more erratic timings, especially sensitive to  $k$ .

$n=$	12	13	14	15	16	
time=	12s	21s	40s	1m13s	3m21s	$k=2$
time=	18s	50s	1m8s	2m43s	8m39ss	$k=3$
time=	60s	2m8s	4m33s	34m32s	87m20s	$k=4$

The timing depends on  $n$ ,  $k$ , the rule, and the resulting topology of attractor basins (the greater the characteristic in-degree, the longer the time). In the case of random Boolean networks, the timing also depends on the particular wiring and rule mix.

The system size that is appropriate for generating an attractor basin in a reasonable time can be inferred from the above. Large sizes may be tried, but may impose unacceptable time, memory and display constraints. Single basins, and especially sub-trees, may be generated for relatively much larger systems, especially for rules having a low in-degree.

