NEUTRAL EMERGENCE AND COARSE GRAINING CELLULAR AUTOMATA

Andrew Weeks

Submitted for the degree of Doctor of Philosophy

University of York

Department of Computer Science

March 2010

ABSTRACT

Emergent systems are often thought of as special, and are often linked to desirable properties like robustness, fault tolerance and adaptability. But, though not well understood, emergence is not a magical, unfathomable property.

We introduce neutral emergence as a new way to explore emergent phenomena, showing that being good enough, enough of the time may actually yield more robust solutions more quickly.

We then use cellular automata as a substrate to investigate emergence, and find they are capable of exhibiting emergent phenomena through coarse graining. Coarse graining shows us that emergence is a relative concept – while some models may be more useful, there is no correct emergent model – and that emergence is lossy, mapping the high level model to a subset of the low level behaviour.

We develop a method of quantifying the 'goodness' of a coarse graining (and the quality of the emergent model) and use this to find emergent models – and, later, the emergent models we want – automatically.

CONTENTS

FIGURES

11 Emergence and information

12 Subjective emergence

12.5 Towards directed coarse graining

ACKNOWLEDGEMENTS

I'm incredibly grateful to Susan Stepney and Fiona Polack, my supervisors, for their help throughout my PhD. They have always been there to give me guidance and support, and I would not have been able to complete this research without them. I'm also extremely grateful for their belief in this project and in me, even when I had doubts myself.

I'd like to thank John Clark for being my internal assessor and for providing me with useful feedback at the different milestones of my PhD.

I'm very grateful to Microsoft Research Cambridge for providing the funding that allowed me to undertake this research and to Fabien Petitcolas at Microsoft for his support and help.

Finally, I'd like to thank my parents, my brother Chris and everyone else who has been there for me. Thank you.

DECLARATION

I declare that all the work in this thesis is solely my own except where attributed and cited to another author. Some of the material has been previously published in the following papers

Neutral Emergence: a proposal [137] – An introduction to the concept of neutral emergence (in analogy to an information theoretic view of neutral evolution) and how it might be used to engineer robust systems.

Neutral Emergence and Coarse Graining [138] – Building on the concept of neutral emergence, we discuss engineering robust solutions and describe preliminary results from applying these ideas to coarse graining of cellular automata.

Investigating emergence by coarse graining Elementary Cellular Automata [139] – We extend coarse graining of cellular automata to investigate aspects of emergence. Starting with the total coarse graining approach introduced by Israeli and Goldenfeld [123], we introduce partial coarse graining. We also show the importance of the mapping between the lower and emergent levels in determining the quality of emergence.

1 INTRODUCTION

Emergent systems are often thought of as special. In the biological world at least, they are closely linked with desirable properties like robustness, fault tolerance and adaptability (§9.19). Unfortunately emergent systems are usually difficult to understand, and even more difficult to engineer.

But emergence is not a magical, unfathomable property. The principal difficulty in working with emergent systems stems from our approach to it: if we try to engineer them with a conventional development mindset and use conventional development techniques, we will face significant difficulties.

We claim we can develop emergent systems as easily as conventional systems. We claim we can even develop emergent systems automatically. And though we approach the problem in a new way, the tools and techniques we use are relatively conventional, drawing on relatively well understood evolutionary and other search methods.

Central to this is a new approach to finding answers: rather than trying to find a perfect solution, we suggest we should look for answers that are mostly correct, most of the time. In fact such solutions may be better than 'perfect' ones, not only because they are easier to find, but also because they can be made more robust than a perfect-but-brittle answer (§9.20).

1.1 Neutral emergence

We introduce neutral emergence, in analogy to neutral evolution, as a new way of looking at emergence. We aim to exploit the robustness and adaptability innate to emergence – and two of the strengths of neutral evolution – as a framework for thinking about what constitutes a good solution.

In an emergent system, it is not obvious how its high level properties arise from its low level behaviour. (In fact, we argue later that this is a requirement for emergence.) This is a big problem for conventional development, which builds a path of small steps between an abstract specification and concrete implementation. But many search and optimisation techniques work differently: we only need to be able to evaluate an answer, not know how to find it. This is a much easier task to solve, and one that only requires a mapping between the levels, not a path with intermediate steps. One class of search technique that works like this is evolutionary algorithms.

1.2 Evolutionary algorithms, landscapes and dynamics

Evolutionary algorithms (EAs) are good problem solvers: they are often efficient, particularly if the total thinking and computation time is considered; they are (usually) conceptually simple and appealing; and they are quite generally applicable – similar techniques can be used solve a wide range of problems.

But of course they are no panacea. Many problems can be answered more efficiently with other, often more specialist algorithms, including a substantial number of algorithms that predate the popularisation of EAs. There also seems to be a 'complexity ceiling' in many EAs that prevents them from solving problems beyond a certain difficulty, and unfortunately this can stop them moving from toy problems to the problems that concern science and industry today.

We suggest that there may be an inherent tension in these algorithms, a balance that must be struck between innovation and conservation which can limit their efficacy. But we believe we may be able to avoid this tension by taking advantage of the problem structure.

There is a story that genetic algorithms were created one afternoon, after someone sat down at a computer with a school biology textbook. In fact it took John Holland, along with his research students, three years to work out all of the details and perfect the technique [77]. The concepts are not difficult, nor is the abstraction obscure; most people would describe their operation as obvious. But DNA recombination in living organisms is an incredibly complex process that is still not fully understood today. Holland had the insight to abstract the right details from the process and apply them successfully to a new field.

Evolutionary systems, with few exceptions, only consider the final solution. They do not consider the structure of the search space, of the problem landscape, which may well be replete with useful information – of how to find a good solution, and of what a good solution is. It may prove possible to exploit these underlying features – to which most algorithms are blind – to find more effective solutions more effectively (at least to a subclass of interesting problems). This structure is believed to be important in two ways: to provide a guide, a constant stream of feedback of progress made; and perhaps more importantly, to constrain search (after all, if there is only one path to a solution, finding the answer is easy).

Landscapes and dynamics are key to understanding neutral evolution, to understanding emergence, and to understanding neutral emergence. They are also key to understanding why a mostly good, most of the time answer is often better than a theoretically perfect but brittle answer.

As we are trying to understand the fundamental properties of this topic, we choose to investigate these questions through cellular automata (CAs). The small search space we used meant it was usually efficient to explore the whole problem space exhaustively, leaving a lot of the reviewed literature on dynamics and landscapes in the further work section. However, we feel that this material is vital for taking this work further and also provides important background material and context for the work on CAs presented here.

1.3 Investigations with cellular automata

The mathematical transparency of CAs make them a good substrate for exploring these ideas without the uncertainty (of definition, of language, of boundaries, of relationships) that would almost

inevitably appear in more complex 'real world' models. Also the small number of some classes of CA, such as the elementary CAs (ECAs) we principally consider here, makes it practicable to explore the entire state space and test ideas against the whole gamut of possible model behaviour.

Despite their conceptual simplicity, CAs are capable of displaying very complex behaviour. For example, ECA rule 110 and Conway's Life (another CA) are both known to be Turing complete (§2.4, §2.7), and it is claimed that some ECAs can be used as pseudo-random number generators. The wide variety of interesting behaviour makes CAs more than just a trivial test case, and allows us to apply and extrapolate concepts found here to other models.

For CAs to be a useful model here, they must be capable of showing emergence. We explore the topic in greater depth in §8 and §9, but it is commonly accepted that emergent behaviour should be coherent (appearing, like a flock of birds, to act almost as a discrete object with behavioural rules of its own) and that there should be a non-obvious link between the emergent phenomenon and the low level actions that create it.

So we can demonstrate emergence in CAs by showing large-scale, high-level, coherent behaviour in CAs, where the relationship between this behaviour and the underlying CA rule is non-obvious. Specifically, we can do this by modelling a CA with another, coarser CA that maps onto the underlying CA.

We believe that emergence is a relative concept, and demonstrate this through CAs: while some models may be more useful to us than others, there is no 'correct' emergent model, and the emergent properties we see are dependent on our point of view. We also use CAs to show emergence as lossy, extracting a carefully chosen subset of the low level behaviour to produce a coherent high level model (often best interpreted in a different language).

CAs help us demonstrate why thinking of emergence as neutral emergence has real advantages when we introduce partial coarse graining. A partial coarse graining has an incomplete mapping between the low level and emergent models and can make mistakes, but this also gives it the freedom to capture low level behaviour that is impossible to model otherwise.

Through quantitative emergence, we show that mutual information is a measure of the goodness of an emergent CA model and use this to find such models automatically. Despite being a local measure, we discover its results closely correlate with our observations of good global CA behaviour.

To make useful emergent systems, we need a way of finding systems that model the emergent behaviour *we want to capture*, and not just systems that model emergent behaviour. Reynolds' Boids model (§8.2), for example, uses just three pieces of information out of the thousands (or millions) of possible choices, and works incredibly well. (In fact, we argue that this brutal restriction is one of the reasons it does work so well.) We develop a way of adding exceptions to CAs so they reflect the behaviour we want to capture. Combining this with the ideas already outlined, we show how we can direct a search to find the emergent models of CAs we want, and do so automatically.

2 CELLULAR AUTOMATA

As they will be used extensively in later chapters, cellular automata (CAs) are introduced, explaining how to update a CA and their typical dynamics. We describe two of the most popular CAs, Conway's Life and elementary cellular automata, and discuss the range of behaviour they produce.

2.1 Cellular automata grids

Cellular automata (CAs) are simple mathematical models that have been used to study many subjects, including complexity, computability theory, biology and physics. Though not required [64, 65], CAs are usually made from a regular grid of cells. The grid is most often one or two dimensional, though it is possible to construct a CA with any finite dimensionality.

Figure 2.1 A 2D cellular automaton grid. Each square is a cell.

A 2D grid is traditionally modelled as an infinite plane of cells (and a 1D grid as an infinite line of cells). Clearly modelling an infinite number of cells is challenging on a computer with finite memory and processing power, so the cells are usually arranged on a torus to give an infinite periodic grid: cells at the top of the grid come below the cells at the bottom, and those at the left edge fall just to the right of those on the right edge. Placing cells on a torus, rather than a rectangle, means we can avoid boundary conditions and also ensures that all locations on the grid are equal – though there is a limit to the size of the phenomena we can explore, our results will be identical no matter which location we use to start it.[1]

2.2 Updating a CA

Each cell has a state, which is set when the CA is initialised. The states are usually binary ('0' and '1', 'off' and 'on', 'dead' and 'alive' – we use □ and ■) though some CAs have more possible states.

The CA has an update rule to calculate the next state for each cell. Though asynchronous CAs exist, the next states are usually calculated for all cells at once in a series of discrete *timesteps* (or *generations*). The cells around each cell, called its *neighbourhood*, are used to determine the next state of the cell. On a 2D grid this is normally the cell itself and the eight surrounding cells. (This is the *Moore*

1 Of course there may be circumstances where a rectangular sheet of cells is more appropriate, perhaps to prevent expelled spaceships from wrapping round and interfering with the pattern being modelled.

neighbourhood; an alternative, diamond-shaped neighbourhood without the diagonally adjacent cells is called the *von Neumann neighbourhood*.) For a 1D CA, the cell itself and the two immediately adjacent cells usually form the neighbourhood.

Moore neighbourhood *von Neumann neighbourhood*

Figure 2.2 The Moore neighbourhood for a regular 2D CA (left); the von Neumann neighbourhood for a regular 2D CA (right).

Some update rules differentiate between cells and give different results depending on which cells are active, whereas others are *totalistic* and only count the number of cells in each state.

2.3 CA dynamics

CAs are discrete dynamical systems. Many CAs follow irreversible, dissipative paths – the trajectories followed from different initial conditions merge over time and end up concentrated on a smaller number of attractors. There must be some initial conditions for irreversible CAs for which there are no previous states. Such patterns are known as *Garden of Eden patterns*.

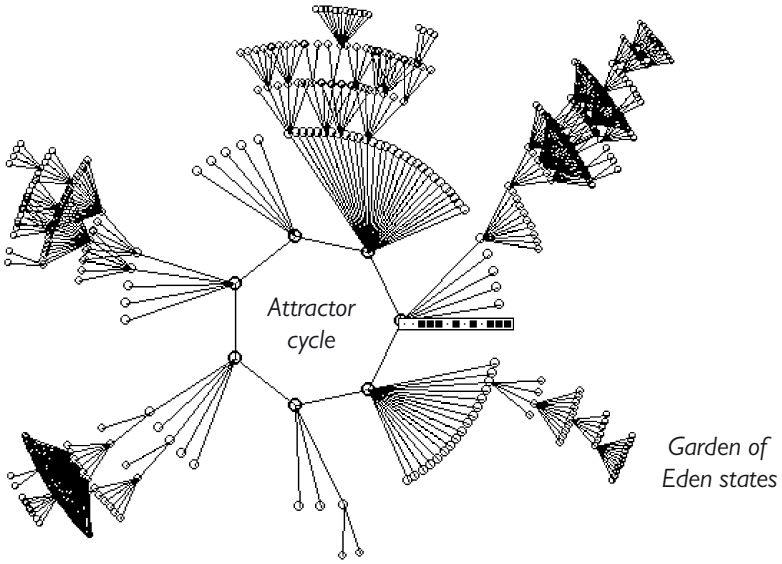


Figure 2.3 The progression over time from all possible states. (Image from [48]) Garden of Eden states are shown on the edges of the diagram, while the attractor appears at the centre. (The diagram shows just one attractor for the system.)

The neighbourhood reach determines how quickly information can spread through a CA: if the neighbourhood only extends for one cell in each direction, information cannot spread faster than one cell per timestep. This is known as the *speed of light* for the CA.

2.4 Conway's Game of Life

The most famous example of a cellular automaton is Conway's Game of Life [66]. Life was created by Conway to simplify von Neumann's self reproducing automata [67], a loop and 'tape' of many cells with 29 different states that is capable of making copies of itself. Life operates on binary, 2D, regular grid with a Moore neighbourhood, and uses the following update rules

- A live cell with fewer than two live neighbours dies of starvation.
- A live cell with more than three live neighbours dies of overcrowding.
- A live cell with two or three live neighbours survives.
- A dead cell with exactly three live neighbours comes to life.

Despite apparent simplicity of these rules, Life can display complex behaviour and many interesting patterns have been discovered. Classes of patterns found include

- *Still lives* – patterns that don't change between generations.
- *Oscillators* – patterns that repeat after a number of generations. The vast majority are period two oscillators [68].
- *Spaceships* – patterns that move across the grid as they repeat. The most famous spaceship is the glider, which moves one square down and to the right in four generations.

Figure 2.4 Patterns in Life

Life is a dissipative CA, and a random initial state usually changes rapidly over a few tens of generations from a mêlée to settle into an ash of simple still lives, oscillators and a few spaceships. Though with careful setup, Life is capable of simulating longer lived patterns. The first one discovered by

Conway was *r-pentomino*, which takes 1103 generations to stabilise and emits six gliders during the run. It is also possible to construct *glider guns* that emit an infinite stream of gliders at regular intervals.

Significantly more complex simulations are also possible with Life. From very specific initial states, it is possible to use gliders, glider guns and other Life patterns to emulate logic gates, timers, memory and the other behaviour needed to create a universal computer. Though very slow to run on a computer, Conway's Life has been shown to have equivalent power of a universal Turing machine [36, 69].

2.5 Elementary cellular automata

Elementary cellular automata (ECAs) are amongst simplest CAs possible. They are 1D binary CAs with neighbourhood of one, so just three cells determine next state of each cell. This limits the number of elementary CAs to just $2^{2^3} = 256$, of which just 88 are distinct.[2] (Compare this to a possible $2^{2^9} = 10^{154}$ 2D CAs like Life.) Having so few rules makes studying the entire rule space practicable and wouldn't leave us reliant on sampling a small (and possibly unrepresentative) corner of the rule space.

Consider rule 58. The rule number is calculated by summing the binary values of the rule's output states, as shown in Figure 2.5.

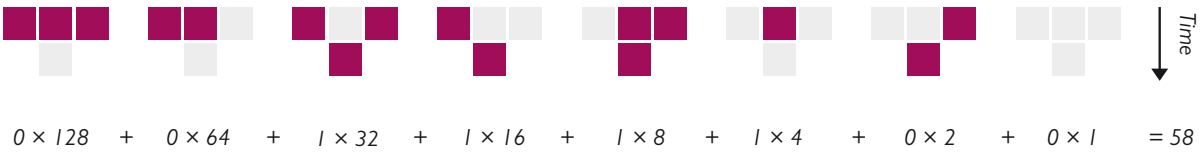


Figure 2.5 Elementary CA rule 58

As ECAs are one dimensional, we can efficiently show progress of the CA over time by placing each subsequent generation underneath the last.

2.6 Simple ECA behaviour

Many rules have relatively uninteresting behaviour, quickly dying out or ending up in unchanging states. One such rule is rule 128. It dies out after a few generations, drawing tapering triangles over time until finally disappearing.

2 The others are either reflections or inversions. If a rule produces diagonal lines that move to the left over time, the reflection would produce a mirror image with lines that moved to the right. An inverse rule substitutes □ for ■ in all cases.

Figure 2.6 The result of running rule 128 for twelve steps, starting with the initial condition in the top line. Each subsequent generation is shown directly underneath the previous one. ■ cells are represented by red squares and □ cells by white squares.

We can see why rule 128 draws triangles by examining the rule in detail. The only case when it outputs a ■ is when the input triple is ■■■. So wide blocks of ■s will remain as ■s in the next generation, but cells on the edge of such a block (with inputs □■■ or ■■□) will change to □. (And any other state will change to or stay as □ as well.) This behaviour repeats, gradually tapering in the edges of wide blocks of ■s, until all of the ■ cells have disappeared.

Figure 2.7 Rule 128

Suppose we change rule 128 to rule 138 so the triples □■■ and □□■ now also output ■. We still lose cells from the right side of wide segments of ■s (state ■■□), but the other side now outputs ■s. The triple □□■ now adds ■s to the CA when there is a single ■ to the right of the cell. Taken together, the rule now draws diagonal lines over time: if we start with a single ■, we end up with single width diagonal line; if we start with five contiguous ■s, we get a diagonal line five cells thick.

Figure 2.8 A run of rule 138. The cell space wraps round horizontally, so the diagonal lines on the right side of the run continue on the left.

2.7 More complex ECA behaviour

Elementary CAs can display complex patterns too. Rule 102, for example, produces this pattern

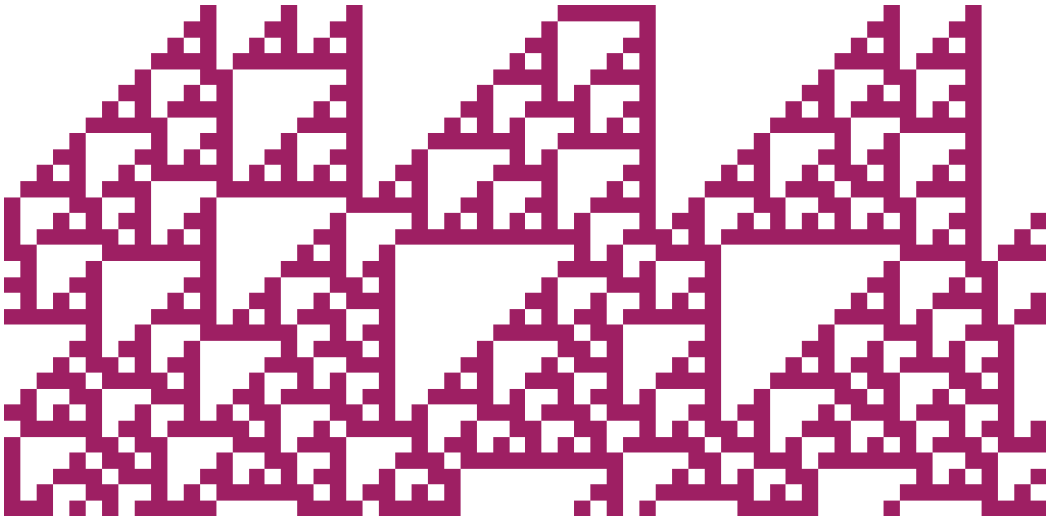


Figure 2.9 Rule 102

Much of the early work into ECAs was carried out by Wolfram [37]. As part of his initial investigations, he classified (by inspection) the elementary CAs into four types of behaviour

- *Class 1* – rules that rapidly converge to a single state (e.g. rule 128).
- *Class 2* – rules that converge to a repetitive or stable state (e.g. rule 138).
- *Class 3* – rules that appear to show chaotic or random behaviour (e.g. rule 102).
- *Class 4* – rules that appear to show complex behaviour, with areas or periods of repetition or stability, but that also show other complicated (and often long term) interactions.

Two rules in Class 4 are 54 and 110. Rule 110 is particularly interesting as it has been shown to be Turing complete by Cook, who proved that it was possible to emulate a cyclic tag system (which is known to be universal) by using spaceships to construct stationary data strings, production rules and clock pulses [70]. As with Life, emulating a Turing machine with rule 110 requires a specific initial condition. The run of 110 in Figure 2.10 is far too small to emulate a Turing machine, but still shows the mix of regular structure and irregular patterns that is characteristic of Class 4 rules.

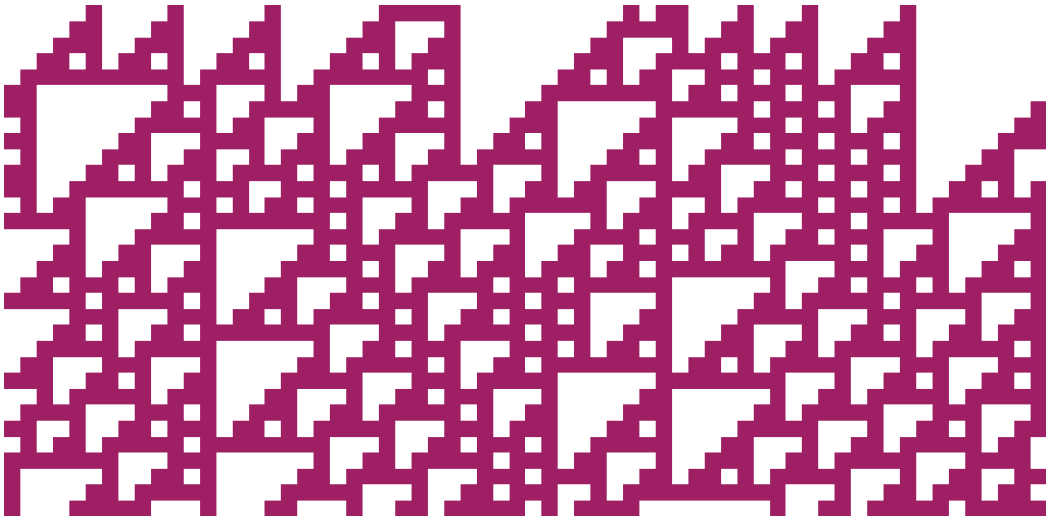


Figure 2.10 Rule 110

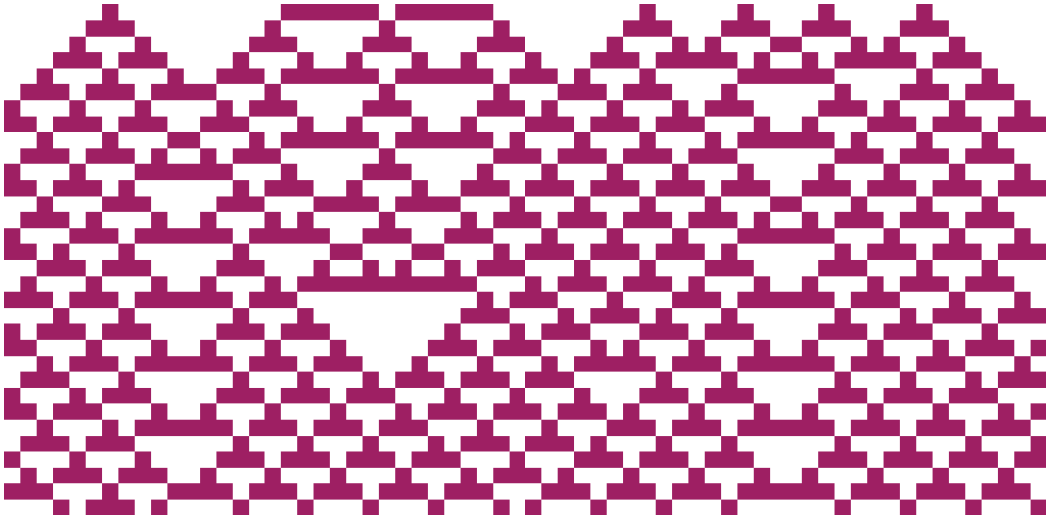


Figure 2.11 Rule 54

2.8 Key points

- Cellular automata are simple mathematical models consisting of many cells and usually arranged on a 1D or 2D grid.
- Conway's Game of Life has equivalent power as a Turing machine and can show many interesting and complex patterns.
- Different elementary CAs show a wide range of behaviour, from simple patterns to chaotic and complex behaviour. Elementary CA rule 110 has also been shown to have equivalent power as a Turing machine.

3 EVOLUTIONARY ALGORITHMS

This section describes the main characteristics of evolutionary algorithms and why they are successful as a search tool. We then look at genetic algorithms in more depth and explore a possible tension in evolutionary algorithms.

Evolutionary algorithms (EAs) are "powerful search and optimisation" [1] techniques that are increasingly used and accepted as part of the computer science mainstream. Usually inspired by the biological processes of genetics and natural selection, evolutionary algorithms are principally designed to meet the challenge of automatic problem solving. Whether the objective is to program a computer to do "useful things" [2], solve mathematical equations or design a bridge, the aim of automatic problem solving is to do this without needing to specify, step by step, how to do it.

EAs usually have a number of common features (from Banzhaf et al. [3])

- A population of solutions
- Innovation operations
- Conservation operations
- Quality differentials
- Selection

Choosing a good *solution representation* is crucial to the success of the algorithm. It must be possible to examine candidate solutions and ultimately to evolve them into better solutions. Most representations are inspired either by nature or computer science: genomes, neural networks, numbers and bit strings are all common.

A measure, or *fitness function,* must be defined against which to judge the quality of solutions. Often an automatic evaluation function is used, but the quality measure could equally be external to the system and subjective, such as aesthetic appeal [4]. It is vital that the fitness function provides a gradation of results, that it can say (approximately) how good a solution is. Otherwise the algorithm degenerates into a "multi-membered blind search" [3].

Rather than just trying one candidate at a time, EAs group a large number of solutions into a *population*. There will inevitably be variation in the quality, or *fitness,* of individuals in the population. EAs improve the overall fitness of the population by preferentially selecting the fitter individuals to produce the next generation of solutions. This means that the population will move towards one comprising high quality solutions, without requiring the EA to understand how to make solutions fitter.

The initial population usually only comprises a tiny fraction of all possible solutions, so it is unlikely that it will contain the best answer. EAs evolve new solutions by applying *innovation operators* to the population. Innovation operators aim to combine the best parts of existing individuals with a novel

element to create fitter solutions. The most common innovation operator is *mutation*, which typically replaces part of a solution with new values or alters a parameter within given bounds. There is no guarantee that the new solution will be better than its parent – if the parent is fit, it is quite likely to be worse.

Conservation operators aim to consolidate the progress already made by members of the population; the most common operators are *reproduction* and *crossover*. Reproduction simply preserves existing material through copying, but never finds innovative new solutions. Like mutation, crossover alters solutions, but instead of replacing material with (usually) random values, information is exchanged between the parents. The objective is to combine the best material from both parents in their progeny, which is often more effective than mutation. Obviously it has the downside of not introducing any new variations that are not permutations of their parents into the population.

Getting the balance right between innovation and conservation is often crucial to the success of an EA. An excess of novelty will tend to destroy good solutions that already exist, but too much conservation will hardly make any progress at all.

Most evolutionary algorithms, including those discussed next, use stochastic decision making. Probabilistic processes are used when selecting individuals for conservation or innovation in an attempt to mimic the apparent randomness of nature.

Individuals in most EAs do not represent the sought answer directly; rather they use an interpretation step to transform the individual into a real world solution. Biological nomenclature for this – genotype and phenotype (respectively) – is usually adopted.

3.1 Genetic algorithms

Genetic algorithms (GAs) [5] are by far the most famous and widely used examples of EAs. Based on a simplified model of genetic recombination, one of their strengths – and reasons for their popularity – is their broad applicability to a wide range of problems.

GAs use a simplified model of a DNA strand to represent each member of the population. Each individual is formulated as a fixed-size *chromosome* divided into a number of segments, or *genes* (the values a gene can take are called *alleles*). The fitness of the final solution is calculated by interpreting the genes in the individual. For example, the gene sequence might reflect the order in which the canonical salesman visits cities on his itinerary.

As with most EAs, the initial population is usually generated randomly. The population size varies substantially, depending on the problem and implementation, but is typically hundreds, covering only a fraction of the total search space. A GA run proceeds as follows

- Each member of the population is evaluated to see how fit it is. It is extremely unlikely that the initial generation will contain an individual that is fit enough, but if one is found then this is returned and the run ends.
- The next generation is produced from the current one through innovation and conservation operators. In GAs, crossover, mutation and reproduction are used. Individuals are selected to be parents based on their fitness.
- The new generation is evaluated and, if no solution is found, the cycle begins again. A maximum number of generations is specified, which may be tens, hundreds or thousands, depending on the problem.

3.1.1 Representation

GA chromosomes are usually modelled as a sequence of bits, though other gene encodings have been used as well, such as real numbers. The use of bit genomes is largely influenced by Holland [5], who pioneered much early work in the field; in particular Holland's Schema Theorem (§3.1.4, [5]) initially provided much of their theoretical underpinnings. In almost all cases, the mapping from genotype to phenotype is a simple one.

The representation – the translation between problem and genome – is domain specific and can be crucial to the success of a GA run. The encoding must be comprehensive enough to model the problem adequately, but sufficiently restrictive to limit the search space to a manageable size. Finding good encodings is "still an art" [3] and often depends on prior knowledge of the problem domain. Unlike most evolutionary algorithms, GAs are domain independent, so for a wide variety of problems "the genetic algorithm carries out its search by performing the same amazingly simple operations" [6].

3.1.2 Fitness and selection

Each individual in the population is evaluated using a fitness function. This score determines the chance each has of producing the next generation of solutions. Fitness-proportional (or roulette) selection (again advocated by Holland [5]) is probably the most frequently used scheme for choosing which candidates should become parents. The chance of an individual being selected is

$$p_i = f_i / \sum f_i$$

p = probability, f = fitness, i = individual

A number of other selection methods have been designed, aiming to improve the fitness or variety of the next generation of solutions. Three of the most commonly used are *rank* and *tournament selection* and *elitism.*

- In rank selection, the population is sorted by fitness, with the least fit individual given a score of 1, the next a score of 2 and so on up to n. This score determines the each individual's chance of selection (from a total score of $n \times (n + 1) / 2$), so fitter individuals are probabilistically selected more frequently to reproduce.
- Tournament selection attempts to imitate rivalry between individuals in nature for the right to mate. Individuals are chosen randomly to compete against each other and each candidate is assigned a probability of winning based on its fitness. Selection pressure can be adjusted to give the fittest individuals a greater or smaller chance of winning each tournament.
- An elitist scheme always copies the best individuals in the current generation into the new population. This ensures that the fittest individual in the new generation is at least as good as that in the previous one.

Both rank and tournament selection are relative methods. Since they do not rely on the absolute fitness of individuals, they reduce the chance of one very fit individual coming to dominate the population. These techniques can also help the population converge faster if the whole population has roughly the same fitness.

3.1.3 Conservation and innovation

As with many EAs, conservation (reproduction) simply copies individuals from one generation to the next. Not all GAs use reproduction, as solutions are usually not changed too radically by the most commonly used innovation operators, crossover and mutation.

3.1.3.1 Crossover

Crossover mimics the sexual reproduction (recombination) of genetic material. The most frequently used version is single point crossover.

- Two chromosomes are chosen as parents using the selection policy
- One gene is chosen at random to be the crossover point
- The tails of the parents are switched after the crossover point, producing two new children

Multi-point (n-point) crossover works similarly, but here several genes along the length of both candidates are selected as crossover points.

3.1.3.2 Mutation

Mutation asexually changes one individual only.

- A chromosome is selected for mutation
- A gene is chosen at random as the mutation start point
- The tail of the individual is replaced with randomly generated new genes

Point mutation just mutates the selected gene, leaving the tail untouched.

3.1.4 Schemata

In almost any population there will be individuals with good characteristics, but it is unlikely that these characteristics will all be found in the same chromosome. The aim of a GA run is to combine these elements into one optimal individual. We now examine the theory behind how a GA accomplishes this often difficult task.

Suppose a pet food manufacturer has persuaded a cat to evaluate some new flavours of kitty chews. The unique selling point for the new range is that it will comprise a blend of three different flavours, but the company needs to find out which combinations taste the best. The options have already been narrowed down by previous research, so the cat is to be presented with the following choices

Salmon	0	Egg custard	1
Tuna fillet	0	Vine tomato	1
Beef stroganoff	0	Jalapeño pepper	1

Figure 3.1 Kitty chew flavours.

Since the company will use a genetic algorithm to formulate new varieties, potential flavours have been paired off into three pairs of alleles (possible values for that gene) and assigned a binary value. So a kitty chew combining egg custard, tuna fillet and beef stroganoff would be denoted as 100.

The first four samples are presented to the cat for an objective succulence evaluation. These results are then fed into the GA to formulate new flavour combinations for the next taste test.

Variety	**Genome**	**Score**
Salmon, vine tomato and jalapeño pepper	011	3
Salmon, tuna fillet and jalapeño pepper	001	5
Egg custard, tuna fillet and beef stroganoff	100	7
Egg custard, vine tomato and jalapeño pepper	111	1

Figure 3.2 Test kitty chew flavour combinations.

The most popular kitty chew is the third one, but no variety has scored a ten (and in general the optimum value may not be known). The kitty chew compositions and ratings do not provide any information about which of the ingredients were good or bad, but the population as a whole contains additional hidden information that can be exploited.

One conjecture might be that egg custard is crucial to a kitty chew's popularity. This hypothesis, or *schema*, covers all chromosomes containing custard and is represented as 1** (* indicates don't care and matches 0 or 1). Similar schemata can be created for all other possible suppositions; these schemata can be of differing length (*order*), from salmon, tomatoes and beef matter (010, order 3) to nothing matters (***, order 0).

It is possible to assign a fitness to each schema by averaging over its instances (individuals that use the schema) in the current population

$$\hat{f}(S,t) = \sum_{x \in S} f(x) / n(S,t)$$

$\hat{f}(S,t)$= schema fitness, $f(x)$ = fitness of string x, $n(S, t)$ = instances of schema S at time t

These schemata can be seen as competing with each other, their potential validity as expositions being proportional to their fitness. Holland's schema theorem [5] states that (for roulette selection) the expected fecundity of a schema S at time $t + 1$ will be proportional to its relative fitness (compared to the population's mean fitness)

$$E(n(S,t+1)) = n(S,t)\frac{\hat{f}(S,t)}{\bar{f}(t)}$$

$E(n(S, t + 1))$ = expected instances of S at $t + 1$, $\bar{f}(t)$ = mean fitness of population at t

Note that, since the expected instances of a schema S at time $t + 1$ is proportional to the number at time t, schemata experience (approximately) exponential growth.[1]

The schema theorem suggests that GAs work by manipulating and combining these schemata as whole units; in fact, the schema theorem assumes that there is no disruption to them at all during a GA run. Crossover and mutation will inevitably destroy some schemata, but the impact can be minimised if the mapping between the genotype and phenotype is carefully considered. By grouping the information within the chromosome pertaining to each phenotypic feature, there is a much greater chance of this being passed on whole to the individual's children.

3.2 A tension in evolution

Crossover is by far the dominant operator in GAs; typically "95% of operators [applied] are either reproduction … or crossover" [3]. (Though it should be noted that some evolutionary algorithms, such as evolutionary strategies [1, 7], often rely exclusively on mutation.) Mutation is essential to explore novel areas of the search space and limit premature convergence, but it can be seriously disruptive to the population as good adaptations tend to be lost in the noise. We have argued that "[t]here is a tension between the need for innovation and conservation in GAs" [8].

Genetic programming (GP) [6] adopts many of the principles behind GAs and applies them to tree structures. Unlike (most [9]) GAs, GP genomes can be of variable length. In addition to encoding

1 Spall [157] says that claims of exponential growth are "not fully justified", noting that changes in the mean fitness can affect the growth rate and that the Taylor series approximation associated with the mutation probability makes it only an approximation (please see [5] for details).

programs, this property has made GP useful in many other fields including modelling antennae [10] and circuit design [11].

The problem of poor quality offspring is substantially accentuated in GP, since the progeny of non-uniform crossover are rarely the same as their parents – even incestuous (self) reproduction usually produces novel offspring – which tends to infuse a lot more variety in the population. In standard GP, crossover has an "overwhelmingly negative" [2] impact on the fitness of offspring. Experiments investigating crossover in different GP systems by Nordin and Banzhaf [12] and Teller [13] suggest that 75% of children are less than half as fit as their parents. Further, only 10% of progeny were fitter.[2]

3.3 Recombination as conservation

Nature goes to great lengths to prevent changes to the genetic code. There are more sequences that code for the most abundant amino acids; this redundancy means that most transcription errors will be neutral [14]. Much of the length of a DNA strand does not appear to have any function; it has been suggested that this junk DNA exists to act as a buffer to prevent crucial information being overwritten by mistake [3, 15].[3]

Almost all genetic exchange is homologous [3]. In homologous exchange, the locations where genetic material is swapped are tightly controlled: exchange can only occur between nearly identical DNA segments and only if the swap point means both are functionally identical too. The reason life goes to these lengths is to preserve the function of genes – if a species relies on a particular protein, it is critical that the ability to produce it is passed on to its progeny; an essentially random genetic exchange is unlikely to preserve functionality. When large-scale changes (mutations) do occur they are almost always lethal.

This is why disparate species such as the horse and finch should not mate [3], and nature deploys a substantial arsenal of isolating barriers to prevent this happening. Characteristics evolved to promote isolation include habitat, seasonal or temporal avoidance, lack of sexual attraction, and hybrid inviability or sterility [15].

Although genetic exchange is a source of variability, Watson et al. [16] argue that "recombination's most vital function is probably the repair of damaged DNA." Indeed the neutral theory of evolution suggests that most adaptation (in large populations) happens through genetic drift, not

2 Nordin and Banzhaf looked at was symbolic regression using a polynomial, for which they attempted to evolve register machine programs. Teller introduces PADO, a learning architecture for signal understanding whose "learning core is Genetic Programming" [13]. An evolved PADO program is a directed graph "reminiscent of Turing machines and Finite State Automata" [13].

3 There is some evidence to support this. Pseudogenes (close analogues of genes) have been found in junk DNA [15]. Introns can also build up dramatically towards the end of a GP run. It has been suggested that these structures evolve to reduce the chance of good schemata being disrupted by crossover.

selection [15]. The "highly influential" [15] Fisher model of adaptive evolution argues that species develop through a series of small steps, gradually edging towards their adaptive optimum.

3.4 Recombination as innovation

The design of genetic algorithms is largely inspired by the structure and reproductive processes of genetic material in eukaryotes: they have a linear genome, rely principally on crossover and allow some mutation to introduce novelty. However, the makeup of a solution is not known at the start of a GA run, and the fittest individual found is likely to be radically different from the (often randomly generated) initial population.

So to give results in a reasonable timeframe, GA individuals must be able to evolve much more quickly than nature does. There is a potential for conflict between reproduction in nature, which is geared very much towards genetic stability, and the need for progress in a GA search. In EAs, there is a conflict – or at least a trade off – between information retention and progress. Even within GAs, there is tension between the need for mutation and crossover within GAs and their disruptive effects on individuals.

There are obviously many caveats to this argument

- A GA is a simple abstraction of natural processes. In particular, much of the machinery and structure that preserves the genome is discarded (diploidity (or higher), redundant coding, error checking mechanisms, etc.). Thus arguments for constraints on genome evolution may not apply.
- A GA population need not be viable at all points during the search - an individual needs only to be relatively fit within its generation, whereas life in the natural world must not only compete with its peers but also survive in its environment.
- GAs are a valid and often effective search technique that can be successfully applied to a wide range of problems.

It has been argued that the balance between innovation and conservation is not a trade off at all, but rather an optimum value [17]. Certainly life would want to maintain a degree of change (assuming it were possible to eliminate this entirely) and introducing a small degree of mutation to GAs is usually seen as beneficial, but we believe it is fair to argue this 'optimum' is merely the best compromise available within the considered scope. It is surely the case that increasing innovation will help the search progress more quickly; unfortunately, the search will tend to progress in the wrong direction as good information is lost. Similar arguments can be made about conservation. A new 'oracle' search technique that innovated quickly and correctly would outperform it consistently.

While trying to find such a technique would obviously prove as fruitless as the oracle would be bountiful, it does raise the question as to whether a more efficient search technique – over a subset

of interesting problems, free lunch caveats noted – could be developed using different, perhaps developmental, paradigms. In the next few chapters, we explore some of the factors that may be important in forming and constraining such ideas.

3.5 Key points

- Evolutionary algorithms are popular search and optimisation techniques usually inspired by the biological processes of genetics and natural selection.
- Genetic algorithms, the most widely used EA, use a simplified model of genetic recombination to find solutions. They are quite general and can be applied to many problems.
- There may be a tension in evolution between recombination as innovation and recombination as conservation in evolution-inspired search techniques.

4 NONLINEAR DYNAMICS

Dynamical systems are a key component of the questions we are trying to answer here. The next chapter focuses on Chris Langton [18], who believes that computation and the dynamical behaviour of a system are one and the same. In the following chapter, Stuart Kauffman [17, 19] takes these ideas further, suggesting that dynamical behaviour dictates a system's capacity to adapt to its environment, and its ability to adapt to itself (systems evolve to evolve). Finally, dynamical systems form a key part of neutral emergence (§9.9), and the robust systems advocated alongside.

Chaos and fractals are perhaps one of the best known (though least understood) branches of mathematics: James Gleick wrote a bestselling book [20] describing their discovery and development; they are well established in the lexicon of newspapers, fiction and popular science; computer programs capable of rendering beautiful images are widely available. What is perhaps less well known is how intimately these ideas related to almost every aspect of the world around us, including life itself [17, 44, 45, 145, 146].

Chaos forms part of the field of dynamics, specifically nonlinear dynamics. Linear dynamical systems have long been used by physicists and others for modelling the world. But to create these linear models it is often necessary to approximate and restrict, so calculations are only valid over a short timespan or only account for the most common mode of behaviour. Nonlinear systems do not intrinsically suffer from this deficiency. On the other hand, even complicated linear systems can be solved; nonlinear systems, as a rule, cannot.

The adequacy of linear approximations varies dramatically with the application: both the weather and the motion of planets in our solar system contain nonlinear components, but only in the former do they dominate. Thus the position of Venus in six months' time can be calculated with some accuracy using equations as simple as Kepler's laws [21], whereas predicting the weather even six days ahead requires something analogous to a brute force attack by supercomputer. And Kepler would have been dismissed long ago if his laws had as poor a record as that of weather forecasters.

The difficulty of applying traditional mathematical techniques to nonlinear problems has led to an emphasis on modelling, iteration and geometric methods, producing some very interesting and important results that – although not proved – have appreciably shifted perceptions in a number of important areas [22].

4.1 Pendulum

One of best known examples of nonlinear system is the swinging pendulum. As it swings, the pendulum charts a path through space. Since the rod is of fixed length, the weight's motion can be modelled one dimensionally. The (undamped) pendulum thus charts a path through position / velocity space over time, governed by

$$\ddot{\theta} + \frac{g}{L}\sin\theta = 0$$

$\ddot{\theta}$ = acceleration, θ = angle from (downward) vertical, g = acceleration due to gravity, L = length of pendulum

Even simple dynamical systems such as the pendulum can exhibit qualitatively different – and sometimes surprising – behaviour. One normally thinks of a pendulum as swinging back and forth, but (by applying a bit more force) it is possible to get the weight to whirl over the top, or stop it completely at the bottom of its swing. In theory (though not in practice) the pendulum can also sit still vertically with the weight at the top.

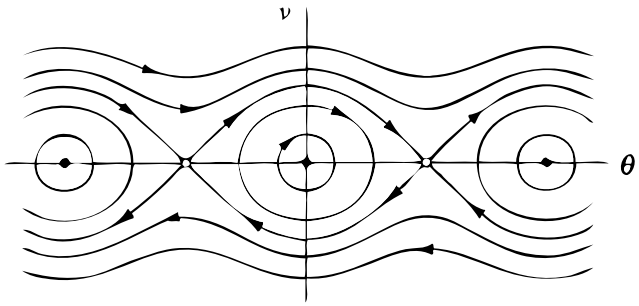


Figure 4.1 Pendulum phase portrait plotting velocity against angle (from [22])

These different states correspond to the pertinent features shown in Figure 4.1. The pendulum is oscillating when it flows around the circles (larger circles for bigger swings). When the pendulum flies over the top it follows the wavy trajectories at the top and bottom of the picture. The solid dots correspond to the weight hanging still at the bottom (all are actually the same point as the diagram 'wraps') – these are *stable fixed points*. The white dots are analogous to the pendulum standing vertically upwards; they are *unstable fixed points*.

Figure 4.1 shows a non-dissipative system – no friction or other force acts on the weight to slow it down, so the pendulum follows the same trajectory for eternity. Most real systems are of course dissipative: the pendulum will eventually come to rest as energy is lost to the environment. In the dissipative system Figure 4.2, the stable fixed points (point) have now become attractors with trajectories spiralling into them from their *basins of attraction*. Basins of attraction surround stable fixed points (and other attractors), inexorably drawing in any captured trajectories. Unstable fixed points do not have a basin around them, so they can only be reached by following one specific trajectory (out of the infinite possible ones).

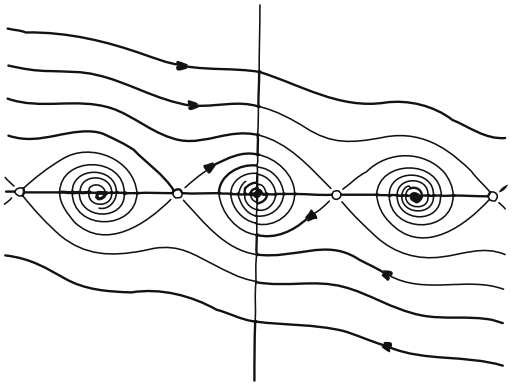


Figure 4.2 Dissipative pendulum phase portrait (from [22])

4.2 Bifurcations

One of the reasons – perhaps the main reason – that dynamical systems are so interesting (not to mention complicated, obscure and, in some cases, dangerous) is their propensity to change their behaviour qualitatively. Consider placing a light weight on top of a beam. The beam will resist the weight and remain straight. But if the small weight is replaced by a large one, the beam may buckle to one side or the other. Here, by increasing the mass (by changing the system's parameters), a single stable situation has been replaced by two alternate stable states, and the original has become unstable.

In general, fixed points can be created or destroyed, and their stability can change. These changes, or *bifurcations*, come about as the system's parameters are varied.[1] Bifurcations come in a number of forms with substantially different qualitative behaviour. The most common are summarised next.[2]

4.3 Saddle node bifurcation

A *saddle node bifurcation* (Figure 4.3 and Figure 4.4) is the "basic mechanism" [14] through which fixed points are created or destroyed. As a parameter is varied, two points move towards each other, collide and mutually annihilate. The effects of the node are apparent even after the saddle node has vanished in the form of a *ghost*, a bottleneck region that develops around the former node's location, much like a basin surrounding a node (Figure 4.5). The ghost sucks in trajectories and delays them before they proceed out the other side.

1 Though bifurcations represent a qualitative shift in system behaviour, they can be seen as part of a continuum within a higher dimensional model.

2 Most of the diagrams in this section have been taken from [22]. Some have axes labelled as distance (x or θ) against velocity; others use a more abstract x and y. These labels may be useful to interpreting the graphs but are not central to the points made here – it is their qualitative shape that is important. A number show the effect of varying a system parameter r or μ which does alter the graph qualitatively. This is remarked upon where appropriate.

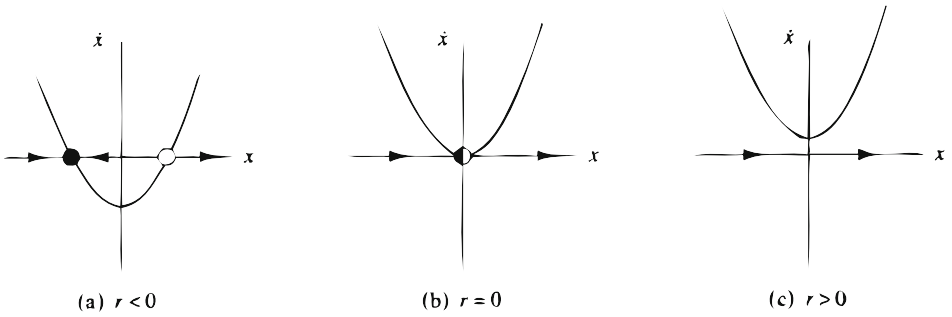


Figure 4.3 Saddle node bifurcation. Bifurcation occurs as system parameter r is varied. When $r = 0$, the fixed point is half-stable: it attracts from one side and repels from the other (from [22]).

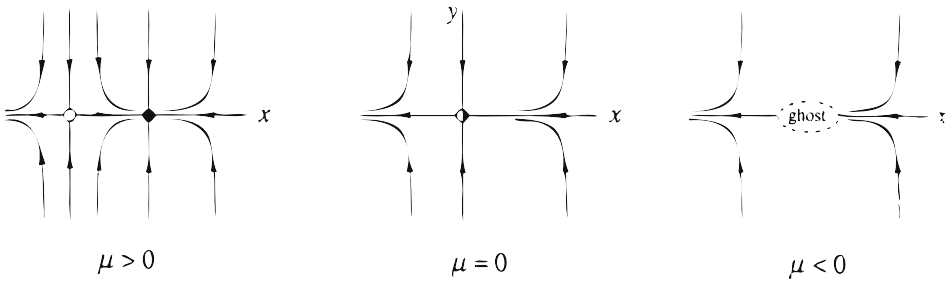


Figure 4.4 Phase portrait of a saddle node bifurcation (including ghost) as parameter μ is changed (from [22]).

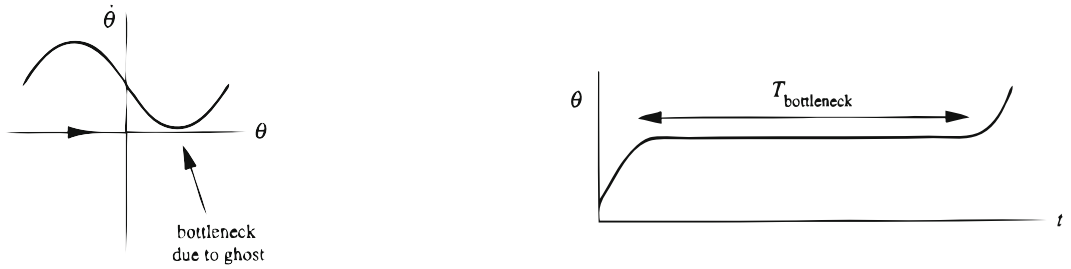


Figure 4.5 Graphs show velocity against position and position against time. The ghost (slowing speed $\dot{\theta}$ almost to 0) holds the system in one location θ for a substantial period of time (from [22]).

4.4 Transcritical bifurcation

In a *transcritical bifurcation* a fixed point exists for all parameter values, but the point may change its stability. (Consider a simple population growth model, where 0 must always be a fixed point.) In Figure 4.6 the unstable fixed point approaches the origin, coalesces with it forming a saddle node, and finally emerges from the origin as a stable node. There has been an exchange of stabilities between the two nodes.

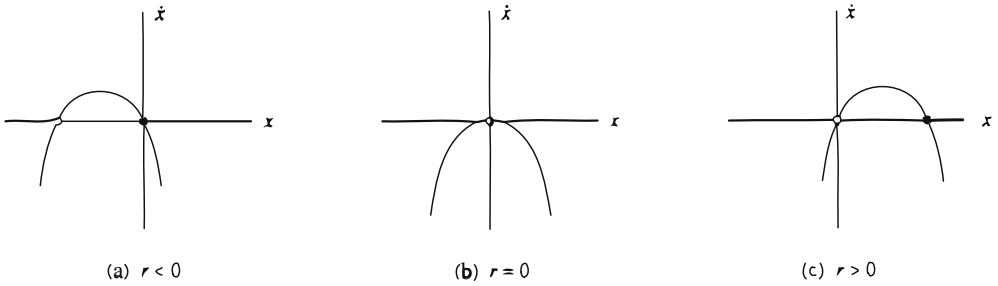


Figure 4.6 Transcritical bifurcation as parameter *r* is varied (from [22])

4.5 Pitchfork bifurcation

Pitchfork bifurcations are common in systems with symmetry (Figure 4.7). (For example, the beam above buckled to the left or right.) There are two types of pitchfork bifurcation: *subcritical* and *supercritical*. The bifurcation diagram for the supercritical case in Figure 4.8 shows a stable fixed point becoming unstable and, at the same time, two new stable fixed points emerging from the origin (see also Figure 4.9). In a subcritical system, the pitchfork is inverted and now unstable.

Note from Figure 4.8 (right) that no stable fixed points exist beyond the origin. In this case, higher order terms in the system's equations provide stability further out by turning around unstable branches, in the process adding interesting dynamics, as jumps and hysteresis are now possible. The system is no longer guaranteed to be reversible as the parameter is varied.

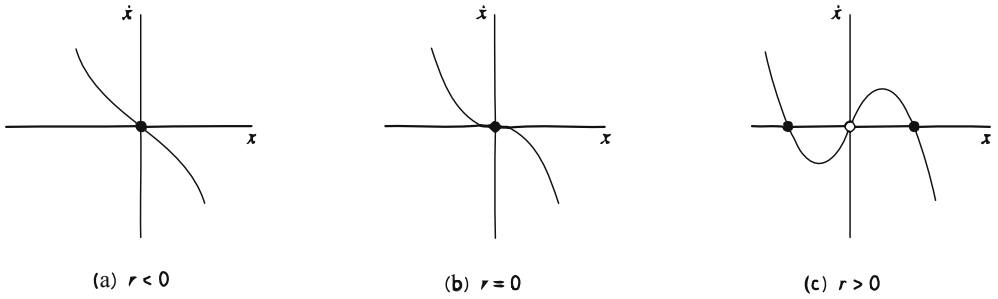


Figure 4.7 Pitchfork bifurcation as *r* is altered (from [22]).

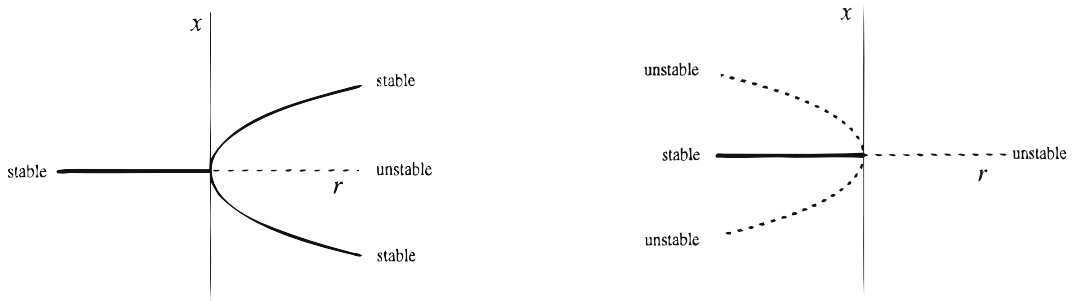


Figure 4.8 Supercritical (left) and subcritical (right) pitchfork bifurcations. Graphs show fixed points in *x* as *r* is varied (from [22]).

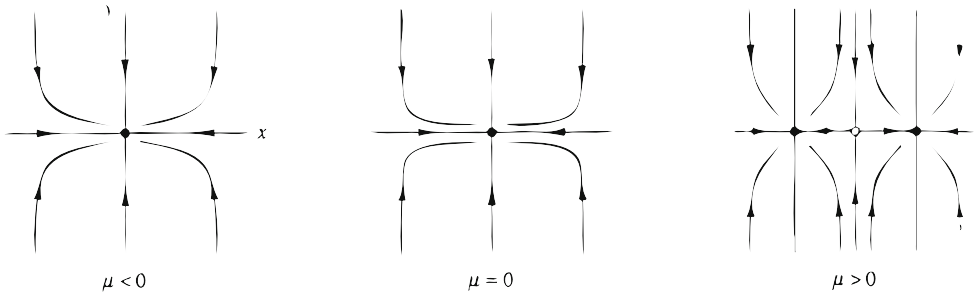


Figure 4.9 Phase portrait of pitchfork bifurcation with changing μ (from [22]).

4.6 Hopf bifurcation

Unlike the other bifurcations seen so far, Hopf bifurcations exist only in two- and higher-dimensional systems. As with the pitchfork bifurcation, there are two types of Hopf bifurcation: supercritical and subcritical. Suppose a system relaxes to equilibrium through damped oscillations and that the rate of decay is controlled by a parameter μ. As μ is increased, decay becomes slower and slower until it passes through a critical value μ_c, at which point the decay changes to growth and the equilibrium loses stability. The system has undergone a supercritical Hopf bifurcation (Figure 4.11). Often systems switch to a small-amplitude sinusoidal limit cycle, shown in Figure 4.10.

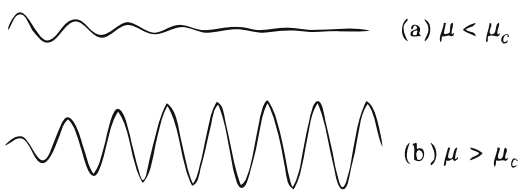


Figure 4.10 System behaviour switches from decay to growth after a supercritical Hopf bifurcation (from [22]).

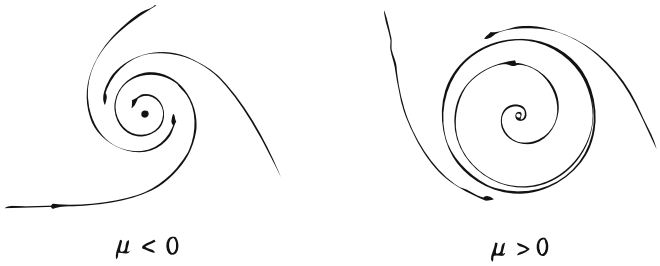


Figure 4.11 Supercritical Hopf bifurcation at $\mu = 0$ (from [22])

Subcritical Hopf bifurcations are "always much more dramatic, and potentially dangerous in engineering applications" [22]. The bifurcation occurs when an unstable cycle tightens around a stable fixed point, causing it to become unstable. Trajectories within its area of influence must then jump to a distant attractor (Figure 4.12). Note that the system experiences hysteresis: once trajectories have moved to distant attractors they will not return when μ is reduced.

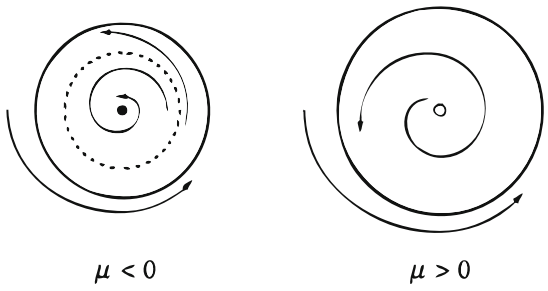


Figure 4.12 Subcritical Hopf bifurcation (from [22])

4.7 Linear analysis

Though they generally offer a poor facsimile of a system's true performance, linear approximations can usefully be applied to small windows of behaviour or to discern salient system features. These approximations then allow powerful linear techniques to be applied to the model, for example constructing a phase portrait of the system from its eigenvectors and eigenvalues. (This summary is adapted from [22]. For a more detailed explanation, see e.g. [23].)

The eigenvalues of a matrix A are given by the characteristic equation $\det(A - \lambda I) = 0$ (I is the identity matrix). For a 2×2 matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The characteristic equation is

$$\det\begin{bmatrix} a-\lambda & b \\ c & d-\lambda \end{bmatrix} = 0$$

The determinant is therefore

$$\lambda^2 - \tau\lambda + \Delta$$

where

$$\tau = \text{trace}(A) = a + d$$

$$\Delta = \det(A) = ad - bc$$

The qualitative structure of fixed points can be classified through the *trace* τ and *determinant* Δ.

- If $\Delta < 0$, the eigenvalues are real and have opposite signs; the fixed point is a saddle. Trajectories approach the node at a tangent to the slow eigendirection (direction of eigenvector with the smaller eigenvalue; see Figure 4.13).

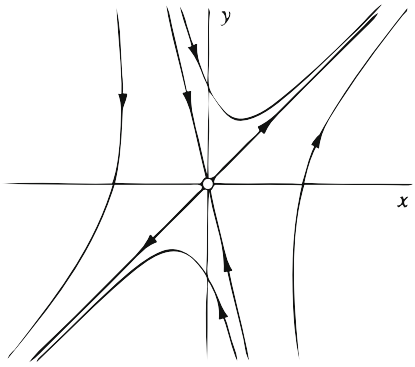


Figure 4.13 Saddle node (from [22])

- If $\Delta > 0$ and $\tau^2 - 4\Delta > 0$ the point will be a node. Alternatively, if $\Delta > 0$ and $\tau^2 - 4\Delta < 0$, it will be a spiral or centre (Figure 4.14). Stability of the nodes and spirals is determined by τ. When $\tau < 0$ the fixed point is stable; otherwise when $\tau > 0$ it is unstable (if $\tau = 0$ the point is a star). Stars and centres are comparatively rare borderline cases,[3] though centres are commonly found in non-dissipative mechanical systems.

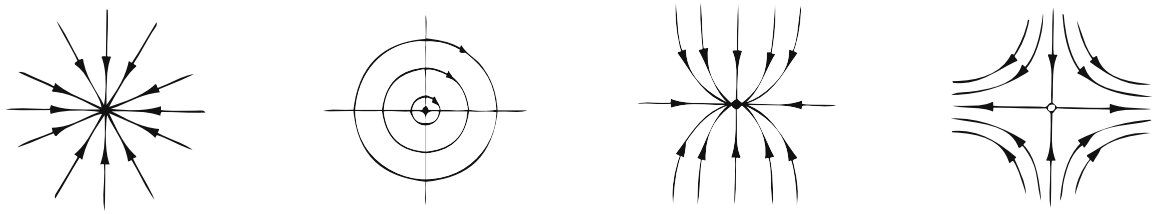


Figure 4.14 Star, centre and two other nodes for comparison (from [22])

In nonlinear systems it is also possible to have limit cycles. Similarly to fixed points, limit cycles can be stable (attracting), unstable and – in rare cases – half-stable (attracting from one side and repelling from the other; see Figure 4.15). Unlike a periodic solution, a limit cycle is isolated: neighbouring trajectories aren't closed. Limit cycles pervade the world around us (and indeed within us): the beating of a heart; the hormone levels in the body; yeast metabolic cycles [24]; or the (dangerous) self-excited oscillations of bridges [25, 26]. These systems have a preferred period, waveform and amplitude, and when perturbed slightly they will always resume their original cycle.

3 Such instances can be difficult to discern through linear analysis as the approximation often distorts the phase space, turning, for example, a centre into a spiral.

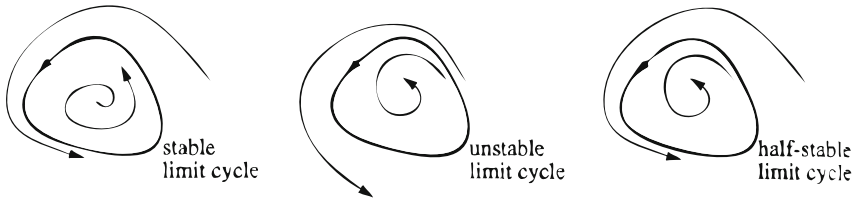


Figure 4.15 Limit cycles (from [22])

4.8 Phase portrait stability

Two systems are topologically equivalent if one is a distorted version of the other: bending and warping are allowed, but ripping is not (e.g. so closed orbits remain closed). A phase portrait is structurally stable if its topology cannot be changed by an arbitrarily small perturbation to its vector field. Stars and centres do not have stable phase portraits: they are perched on the edge between stability and instability. Applying numerical techniques or linearisations to them will distort the phase space and can give misleading results.

4.9 Nullclines

Though it is obviously impossible to determine the whole phase portrait of a nonlinear system from a handful of eigenvectors, linearising around the fixed points often reveals a large amount of local structural information that allows much of the system's flow pattern to be sketched by 'joining the dots', particularly if combined with nullcline information.

Nullclines are defined as the curves where $\dot{x} = 0$ or $\dot{y} = 0$ – they demark lines where the flow is purely horizontal or vertical and partition the plane into regions where $\dot{x}$ and $\dot{y}$ have different signs. Figure 4.16 shows diagram with nullclines drawn and some sample flow vectors; next to this is a computer generated image of the system.

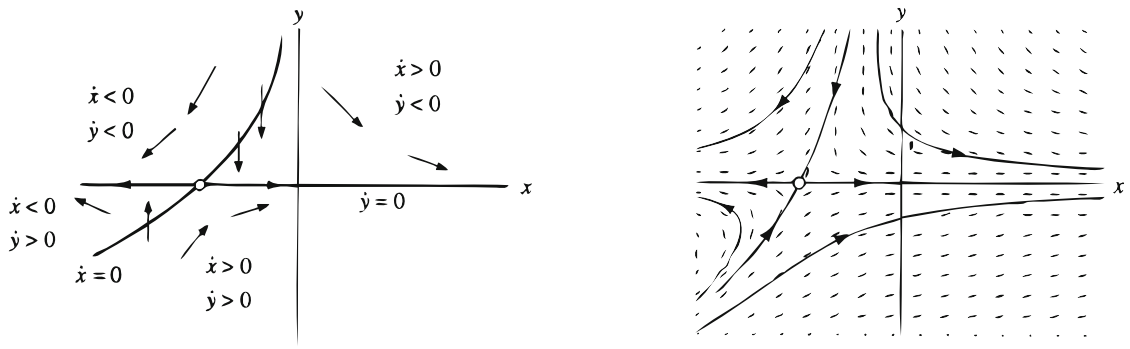


Figure 4.16 Nullclines and a few flow arrows can often give a good indication of the phase portrait (from [22]).

4.10 Higher dimensional systems

The models considered so far have all been one- or two-dimensional. In three dimensions, systems suddenly become capable of exhibiting complex and unpredictable dynamics that are qualitatively different from those seen so far. The first – and arguably most famous – example of a higher dimensional system is due to Lorenz [27]. Searching for an abstract model of weather, Lorenz came up with three simple, interlinked equations that satisfied his brief; these are more commonly presented today through an equivalent model known as the Lorenz waterwheel [28].

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = rx - y - xz$$
$$\dot{z} = xy - bz$$

In the Lorenz equations σ, r, b are parameters. σ is the Prandtl number, r the Rayleigh number and b has no name. Lorenz originally derived his equations from fluid dynamics, hence the constants, though their genesis is not important to the model, particularly when abstracted away from its meteorological origins.

4.10.1 Lorenz waterwheel

The waterwheel has a number of leaky cups on its rim. Water is poured in from above at a steady rate. When the flow is slow, the top cups never fill up enough to overcome friction and the wheel remains stationary. If the inflow is increased to a certain point, the additional weight of the top cups starts the wheel turning, and the system settles into a steady rotation in one direction (depending on initial conditions). But if the rate of flow is increased further, this steady rotation is disrupted and the wheel's motion becomes chaotic: it first rotates one way for a few iterations, then some of the cups get too full and it swings pendulously back in the other direction. The wheel keeps on changing direction erratically, exhibiting an irregular sequence of reversals such as in Figure 4.17.

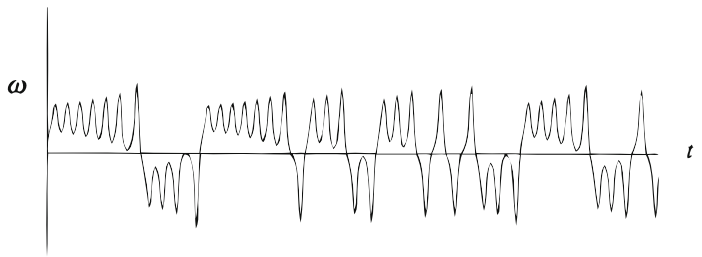


Figure 4.17 Velocity of chaotic waterwheel over time. Note irregular switchbacks (from [22]).

The Lorenz equations are dissipative (in fact they are highly dissipative [22]). For low rates of water flow (low r in Figure 4.18), trajectories end at a fixed point (the stationary wheel). As the rate is adjusted, the fixed point undergoes supercritical pitchfork bifurcation at $r = 1$ to a limit cycle (steady rotation, Lorenz called these states $C+$ and $C-$, shown in Figure 4.18). When flow is increased fur-

ther, each branch of the pitchfork experiences a further change, but this time the pitchfork bifurcations are subcritical – there are no stable points beyond this transition.

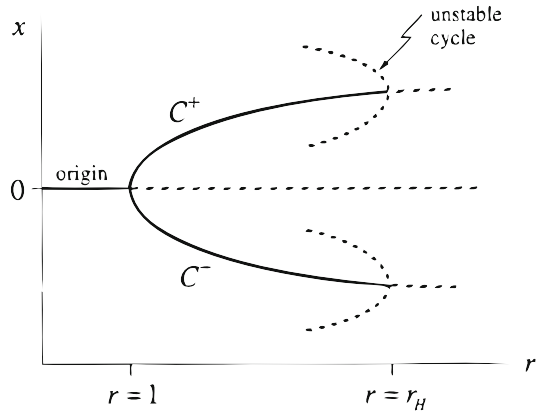


Figure 4.18 Bifurcation diagram of Lorenz waterwheel as flow rate is increased (from [22]).

4.10.2 Lorenz attractor

Since the equations are dissipative, in the limit all trajectories must be confined to a bounded set of zero volume. Yet despite this they manage to move forever without intersecting or approaching a point attractor or limit cycle. This apparently contradictory state of affairs is possible because the attractor exists in three dimensions. A trajectory on the attractor can coil in until it is close to the centre of the spiral and then 'hop' out again to the edge to begin the process again. This is perhaps most clearly shown by the Rössler attractor (Figure 4.19).

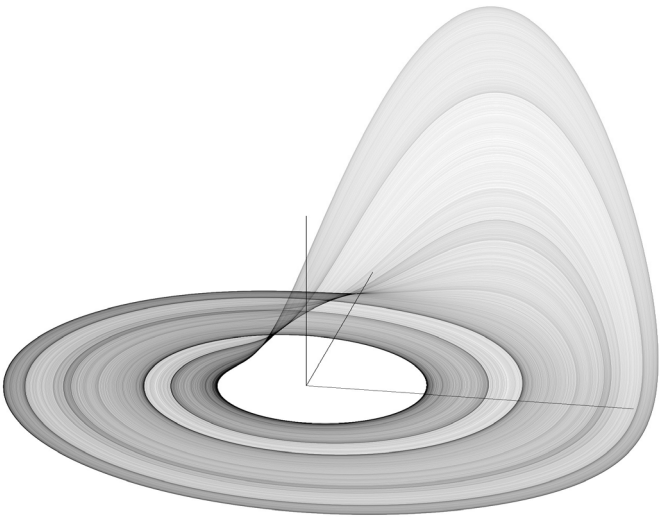


Figure 4.19 Rössler attractor (from [29])

When Lorenz plotted his equations against each other he found the infamous butterfly shape in Figure 4.20; here trajectories spiral outwards on one cycle before jumping to the other (equivalent

to reversing the waterwheel's oscillations). The lines in the diagram only appear to cross because of the 2D depiction here; the attractor's dimensionality has been estimated at about 2.05. Though the Lorenz attractor has zero volume, due to its fractal structure it has an infinite surface area [22, 30].

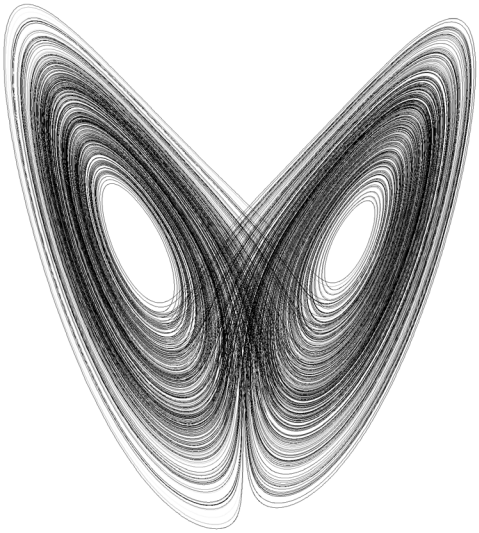


Figure 4.20 Lorenz attractor (from [31])

4.10.3 Lorenz map

Though it seems impossible to guess the state of the system far into the future, the short term behaviour of the attractor looks more predictable. The trajectory appears to have to reach a certain distance out before it jumps to the other spiral. Also, if the trajectory spirals right out to the edge of one loop, it will end up near the centre of the other and spend longer cycling round before jumping again. Lorenz noticed these points too, and hypothesised that z_n (the n^{th} value of the variable z) should predict $z_n + 1$. He plotted the values of z_n against $z_n + 1$ and found that they fell neatly on a curve, with almost no thickness to the graph (Figure 4.21). In this way, Lorenz was able to extract substantial order from chaos.

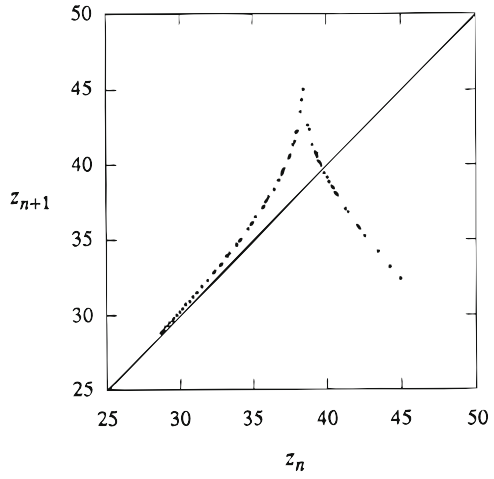


Figure 4.21 Lorenz map (from [22])

4.10.4 Divergence of trajectories

Lorenz found that motion on the attractor shows a sensitive dependence on initial conditions. This means that trajectories started arbitrarily close to each other will rapidly diverge and eventually spread over the entire attractor's surface. Numerical studies have shown that

$$|\delta(t)| \sim |\delta_0| e^{\lambda t}$$

δ is the initial separation of the trajectories. λ is the *Lyapunov exponent* of the system, the slope of the graph of $\ln|\delta|$ against t in Figure 4.22. In this case, λ is calculated to be 0.9.

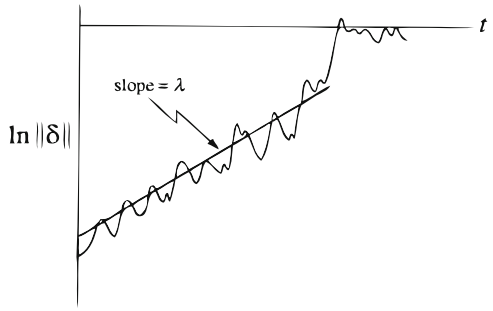


Figure 4.22 Estimating the Lyapunov exponent (from [22])

The positive Lyapunov exponent means that neighbouring trajectories separate exponentially quickly. The implications of this are substantial; in particular it is impossible to predict the long term behaviour of a system that contains a strange attractor (an attractor that shows sensitive dependence on initial conditions) such as Lorenz's [22, 32]. Any measurement will inevitably be subject to some error, and because of the exponential rate of divergence, this error – however small – will prevent long term forecasting with any degree of accuracy. Strogatz [22] notes that a millionfold improvement in measurement quality only affords prediction for 2.5 times longer.

4.11 Chaos in discrete systems

Perhaps surprisingly, chaotic behaviour can also be found in some discrete dynamical systems. By far the most commonly studied examples are iterated maps, such as $x_{n+1} = \cos x_n$. The discreteness in the model comes from employing fixed timesteps; the function values still range over the (continuous) real numbers. In an iterated map, the sequence of points $x_0, x_1, x_2 \ldots$ describes the orbit starting from x_0. These maps have proved useful in analysing differential equations; modelling phenomena including digital electronics, parts of economics and finance theory; and modelling populations in which generations don't overlap.

As with their continuous counterparts, discrete systems have fixed points. Again these can be stable or unstable: their stability is determined by considering a point close by and seeing if it is repelled. The map of $\cos x_n$ has a single stable fixed point at 0.739… – no matter what initial value is used, the plot approaches this value over time. Maps are often illustrated using cobweb diagrams, such as Figure 4.23, which show the path taken over successive iterations of the plot.

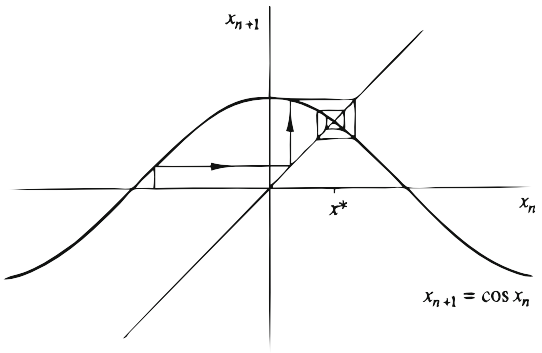


Figure 4.23 A cobweb diagram for cos x_n (from [22])

4.12 Bifurcation diagrams

For a rather more interesting plot, consider the cobweb diagram for the logistic map $x_{n+1} = rx_n(1\ x_n)$. This is often used as a simple model of population growth. When $r < 1$, the plot always goes to 0; in the range $1 < r < 3$, the graph settles to a steady value. For larger values of r, say $r = 3.3$, the population builds up as before but then oscillates alternately between two population sizes. If r is increased still further (e.g. $r = 3.5$), the cycle bifurcates again into one that repeats every four generations. These period doublings continue, arriving faster and faster until, at about $r = 3.569946$, the map becomes chaotic. Figure 4.24 shows the progress from a single fixed point to chaos visually. Interestingly, the behaviour doesn't remain chaotic forever: a stable period 3 cycle begins at around $r = 3.83$ (which itself bifurcates and, amazingly, contains a copy of the entire diagram in miniature).

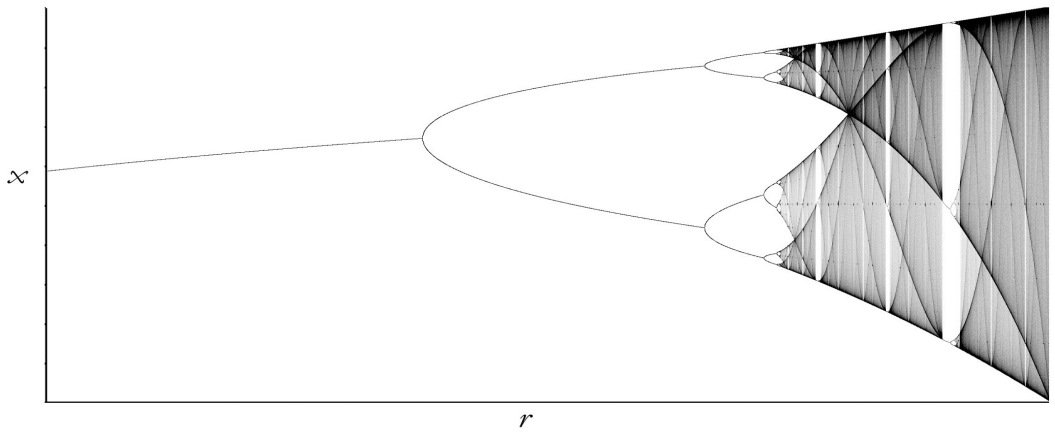


Figure 4.24 Typical iterated map bifurcation diagram showing value(s) taken by x after converging for increasing r (adapted from [33]).

Plotting the bifurcation diagrams of a number of iterated maps shows a surprising degree of correspondence: there are quantitative differences, but qualitatively the graphs are very similar. In fact, a number of universal characteristics have been found in these maps

- A unique series, the *U-sequence*, dictates the order in which stable periodic solutions will appear, independent of the map being iterated. "This amazing result implies that the algebraic form of $f(x)$ is irrelevant; only its overall shape matters" [22].
- Feigenbaum [34] discovered a relationship between consecutive values of r_n, the value at which a period $2n$ cycle first occurs. He noticed that r_n converged between successive transitions, shrinking by a constant factor of about 4.669. The same convergence rate appears no matter what unimodal map is iterated. This value δ is a new mathematical constant, "as basic to period-doubling as π is to circles" [22]. Results from fluid convection and nonlinear electronics appear to support Feigenbaum's value for δ [35].

4.13 Key points

- Nonlinear dynamics is a powerful paradigm for modelling physical systems. Subsequent sections will show it to be equally potent at describing computation, chemical systems and life.
- With few exceptions, nonlinear systems cannot be solved analytically, so simulation is often used. That said, linear analysis can be applied to discover key elements of a system's structure.
- Systems can experience bifurcations, qualitative shifts in system behaviour.
- The system dimension determines the types of behaviour it can exhibit. Systems in three or more dimensions can have chaotic dynamics. The most famous example of this is the Lorenz attractor.
- Discrete systems are also capable of displaying chaotic dynamics. The bifurcation diagram for all such systems is qualitatively the same.
- Chaotic systems have a sensitivity to initial conditions, which makes it impossible to predict their behaviour in the long term.

5 COMPUTATION AS A DYNAMICAL SYSTEM

Nature has shown great resourcefulness in solving apparently intractable problems: photosynthesis, flocking, sex and the genetic code are great examples. We have shown a similarly great aptitude in copying (or trying to copy) nature's ingenuity to solve our own problems. But the emphasis is almost invariably on understanding each specific case, not the general process. Langton's work on cellular automata, reviewed in this chapter, provides a crucial bridge connecting computation and the real world. He suggests that there is a strong analogy between computation and the dynamical behaviour of a system, and that ideas and analyses from each can be applied to the other. Among his most important findings are a number of dynamical properties that a system must have in order to support computation; the next chapter describes how Kauffman extends these to evolution.

One body of work that has shaped the field of artificial life like few others is due to Chris Langton [18]. He posed the question: "Where does computation – especially universal computation – fit in the spectrum of behaviours exhibited by dynamical systems?" A single computation maps a machine from its current state to its next state in the following timestep. Following a sequence of steps, the system charts a path through space and time – this is the dynamical behaviour of system.

5.1 The λ parameter

Cellular automata (CAs) are known to support computation; indeed a few examples (most notably Conway's Life ([36], cited in [18]) have been shown to have equivalent power to a Turing machine. Wolfram [37] has claimed that CA behaviour falls into four broad classes: homogeneous fixed point; heterogeneous fixed point or short period; chaotic; and complex, often with long transients. (See §2.5-§2.7 for more on these classes.)

Langton defined a number of parameters to quantify CA behaviour and exposed "what appears to be the 'deep-structure' of CA rule space" [18]. The most famous parameter is λ, the proportion of transitions from all of a CA's states that end up in a defined, though arbitrary, quiescent state. (The quiescent state is one of the CA's possible states. So for a binary CA that has states □ and ■, □ may be designated as the quiescent state.) Langton uses λ as a baseline against which to measure various phenomena; perhaps most significantly he has shown that the most complex behaviour – including that exhibiting universal computation – occurs in a narrow λ-region between the periodic and chaotic regimes. Typical CA dynamical patterns in this region involves a high degree of repetition, but also branching out into long (sometimes infinitely long) excursions, with strong local and global structure. Conway's Life is located in this area.

5.2 A phase transition

There appears to be a phase transition between the periodic and chaotic regimes: periodic behaviour has low values of λ (the solid regime) and chaotic behaviour high values of λ (gas phase), with complex CAs straddling the gap (the narrow liquid region). Langton suggests that these liquid boundary states can be viewed as complex phenomena, where the long transients found in this region seem to be result of the critical slowing down observed at second order phase transitions in physical systems [22, 147, 148, 149, 150, 151, 152, 153, 154]. This similarity is significant because techniques used to study phase transitions in physical systems can be applied to computation, and because information processing may be an important factor in dynamics of physical systems in the vicinity of a phase transition – computation may emerge spontaneously at the edge of chaos.

Langton [18] reports that the average transition value of λ appears "fairly insensitive" to the number of states a CA has but "highly sensitive" to the number of neighbours. The insensitivity to the number of states is perhaps not surprising as λ is a relative measure over CA states. The large state limit has been studied by Langton and Wootters (reported in [18 p. 75]) through a state stochastic CA, which probabilistically approximates an infinite number of states. They found a "very sharp, second-order phase transition" [18] at λ_c: as the number of states increases, "the distribution around the transition point shrinks" [18]. Sensitivity to the number of neighbours reflects the change in connectivity of the lattice, a result that has also been reported in percolation theory [18, 38] and random Boolean networks (§6.2, [17]).

Langton also examined other measures of CA behaviour. As well as corroborating his existing results, they provide important additional insight into the system dynamics. Three are examined next.

5.3 Correlation

If cells are to cooperate, they must be able to influence each other's behaviour in some manner. In other words, it must be possible to find correlations between events taking place in each cell. Mutual information, defined as the cells' joint conditional probabilities, is at a maximum when cells are perfectly correlated, and 0 when they are completely independent. (Mutual information is introduced in §7.7.) Langton measures mutual information through the degree to which the changes in two cells mimic each other: if they behave identically then they are perfectly correlated. A similar phase transition can be seen in these data as well: the mutual information is essentially zero below the jump, then shifts to a "moderate value" [18] before slowly decaying as λ continues to increase (Figure 5.1). Langton attributes this decaying tail to an approach to "effectively random dynamics. The lack of correlation … at high λ values means that cells are acting as if they were independent of each other, even though they are causally connected" [18].

Even at the transition, the mutual information between cells is significantly below the maximum possible ($I = 1$). To support computation, Langton suggests, cells must exhibit the right degree of

correlation. If cells are overly dependent they will simply mimic each other; if they are excessively independent they will ignore each other. Neither of these regimes is amenable to cooperative behaviour: there must be what Langton terms a meaningful signal between the cells. With too little correlation, there will be no common code; with too much correlation, there will be nothing to communicate.

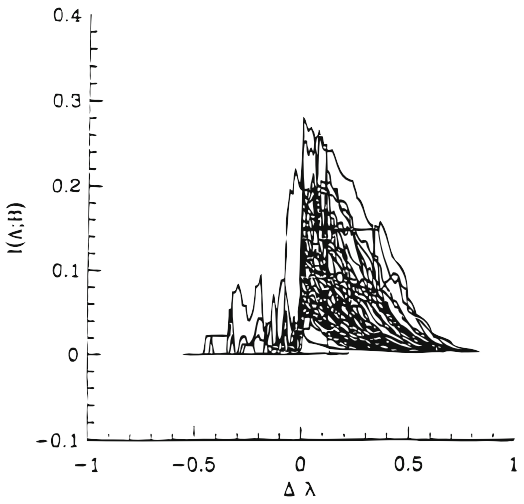


Figure 5.1 Phase transition seen plotting mutual information (correlation) *I* against λ. As the transition λ value varied slightly between experiments, Langton lined up the plots at this point ($\Delta\lambda = 0$) to emphasise the similar shape taken by all of the graphs (from [18]).

5.4 Growth dimension

One measure of a CA's capacity to propagate information is its growth dimension, the rate at which perturbations spread through a CA exhibiting 'typical' behaviour. CA structure means that perturbations can spread a maximum one unit per timestep (Langton adopts a neighbourhood of radius one; see §2 for an introduction to CAs). With this in mind, the growth dimension is defined based on the new region affected – the maximum growth dimension on a square grid CA will be 2 (a complete new square of cells is added around the perturbed area at each timestep).

Langton remarks [18] that the growth dimension is qualitatively similar to Lyapunov exponents (§4.10.4); indeed, its sign and value provide the same information as the largest Lyapunov exponent in continuous dynamical systems. In fixed point systems, perturbations will shrink and finally disappear; here the growth dimension is negative. Disturbances to periodic CAs will spread to a finite size (tending to zero growth) as with periodic dynamical systems. Chaotic CAs and dynamical systems both have positive Lyapunov exponents; the typical growth dimension of a chaotic CA is close to 2.

The most interesting CA behaviour is found in the region with a growth dimension greater than 0 but less than 2. Here there is a slow but positive spread of perturbations. Disturbances often extend through propagating, quasi-periodic structures, allowing perturbations to carry signals over long

distances without generating excess noise. Again there appears to be a phase transition between the fixed and chaotic regimes, with the CAs with the most interesting dynamics straddling the boundary.

5.5 Length of transients

Langton proposes that "another discriminator of complex behaviour is the length of time it takes a system to settle down to 'typical behaviour' following a perturbation" [18]. Fixed point and periodic systems typically have short-lived (and spatially limited) transients. In particular, transient length is independent of the system size. In fully chaotic systems, perturbations are "quickly randomised, and local regions cannot 'feel' the size of the system" [18]. They rapidly resume 'typical' random behaviour, again independent of system size. In contrast, complex systems can exhibit very long – even effectively infinite – transients (obviously dependent on system size), which is "of fundamental importance" [18] for supporting computational dynamics. See Figure 5.2.

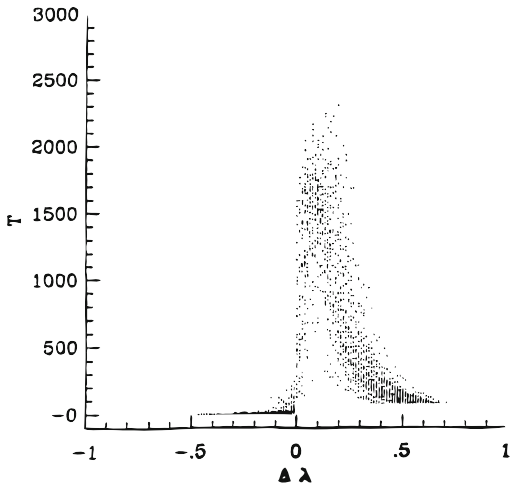


Figure 5.2 Phase transition seen plotting length of transients T against λ. Again, Langton has aligned the graphs at the transition point $\Delta\lambda = 0$ (from [18]).

Langton's assertion that the transients in chaotic systems are quickly absorbed into the general mêlée of interactions takes a similar position to that adopted in statistical mechanics (indeed he draws this analogy himself). But such a perturbation to a chaotic system can cause (is almost guaranteed to cause) substantial changes at the cell level and at a macro scale. Seen from a different perspective, it may be that these dynamics – which are not random – reveal interesting structure or other information about systems (though we do not claim to know what). Kauffman has asserted that statistical mechanics is inadequate for describing the behaviour of biological systems (§6.2, [17]), and that a different language is needed to describe their behaviour. It may be that this behaviour, viewed with the tools of a new statistical mechanics, is not so random after all.

5.6 Key points

- Computation can be viewed as a dynamical system, and significant insight can be gained by applying techniques developed for each to the other.
- The most interesting dynamic behaviour is found in a narrow liquid region at the phase transition between ordered (solid) and chaotic (gaseous) behaviour. Systems capable of universal computation are located here. For CAs, the liquid region can be found at approximately the same value of λ.
- Other measures that appear conducive to creating a complex, cooperating system capable of exhibiting order at local and global levels (such as a universal computer or life) are also maximal in the liquid region, including cell correlation, length of transients on perturbation and the growth dimension of perturbations.

6 NK LANDSCAPES AND RANDOM BOOLEAN NETWORKS

The importance of understanding landscape structure in search has already been stressed. Kauffman's NK landscapes were specifically designed as tuneable structures, able to explore how a system's capability to adapt is affected by greater epistasis. (The original motivation was to model adaptation in protein space.) It turns out, as might be expected, that increased coupling gives a more rugged fitness landscape, but many of the results Kauffman uncovers through this thorough analysis are more subtle than that. He presents interesting findings that show a limited degree of coupling (i.e. not a completely smooth landscape) gives better results than no epistasis. He also argues for tuning the search process to match the landscape, holding out the tantalising possibility of optimising for both conservation and innovation (§3.2-§3.4).

The second part of this chapter gives an overview of random Boolean networks, which Kauffman introduces as a paradigm for biological systems equivalent to statistical mechanics for ideal gases. These networks are capable of exhibiting a surprising degree of order (and robustness) despite many aspects of their construction being essentially random. Kauffman also demonstrates a transition from ordered to chaotic behaviour that mirrors the move from smooth to rugged landscapes and argues, like Langton, that systems capable of the most complex behaviour are poised on the edge of chaos. He further contends that there are significant dynamical constraints on evolvability, and that systems on the edge of chaos are best able to evolve.

6.1 The NK model

In search, hillclimbing over a fitness landscape is hardly a new concept, having been introduced by Wright in 1931 [39]. Yet it is this apparently trivial model that Kauffman chooses to investigate through his *NK* model [17 p. 40, 19]. Many aspects of the *NK* model are conventional (Kauffman relates that the *NK* model is actually a form of spin-glass [17 p. 43]). Each entity (or genome) comprises N parts (or genes), and the genome's overall fitness is simply calculated from these constituents. K reflects how coupled the system is – it measures the richness of the epistatic interactions within the genome.

- When $K = 0$, the fitness of the genotype is simply the sum of the N genes' independent fitness contributions (divided by N).
- When $K > 0$, the fitness contribution of each gene is calculated from its own fitness and the contributions of K other loci. Schemes have been proposed for modelling epistasis, one of the most common being to multiply the genes' fitness scores. Kauffman argues this is too simplistic

to account adequately for the "complex web" [17] of interactions – one cannot know, *a priori*, which allele combinations will have the highest fitness, or how any change will affect the overall fitness. He suggests we therefore admit our "total ignorance" [17] and simply assign fitness values at random. The fitness of each gene depends on itself and K other genes, each gene has 2^{K+1} possible fitness scores. These are summed and normalised as before to give the genome's overall fitness.

The ruggedness of the fitness landscape varies as N and K are altered. Many properties of the landscapes "appear to be surprisingly robust and depend almost exclusively upon N and K alone" [17]. Kauffman considers several classes of NK landscape model which are reviewed next.

6.1.1 K = 0 landscapes

There is a single global optimum, and all suboptimal genotypes can climb towards fitter neighbours (indeed all points lie on a connected path to the global optimum). The landscape is very smooth since all one mutant neighbours (by changing one gene to a different allele) have nearly same fitness. Structurally, the landscape is highly correlated – the fitness of one mutant neighbours cannot differ by more than $1/N$. The average walk length to the global optimum increases linearly with N.

6.1.2 K = N - 1 landscapes

These are effectively random terrains – the landscape is entirely uncorrelated. There are very many local optima, in fact the number of local optima increases almost as rapidly as the number of genotypes 2^N (see [17 pp. 46-54] for details). To see how easy it is to search the landscape, Kauffman assumes a rank ordering over the genotypes. As the landscape is uncorrelated, each fitter neighbour found will be, on average, half way to the top of the rank order. Thus it follows that the expected fraction of fitter one mutant neighbours dwindles by half on each improvement step. Similarly, Kauffman has established that walks to local optima are very short ([40] cited in [17]) – their length increases only logarithmically with N. For these reasons, any given genotype can only climb to a small fraction of the local optima, and only a small fraction of the genotypes can climb to any one optimum.

The incredibly complex web of constraints present in these models means that the best mutual allele choices tend to become poorer overall as the number of genes N increases. Accessible optima dwindle in height, down towards the average fitness for the genotype space. In other words, there is an inexorable move towards adaptive walks terminating on poorer solutions as N increases. Kauffman believes this to be "a genuinely fundamental constraint facing adaptive evolution" [17].

> *"As systems with many parts increase both the number of those parts and the richness of interactions among the parts, it is typical that the number of conflicting constraints among the parts increases rapidly. Those conflicting constrains imply that optimisation can attain only ever poorer compromises. No matter how strong selection may be, adaptive processes cannot climb higher*

peaks than afforded by the fitness landscape. That is, this limitation cannot be overcome by stronger selection." [17]

In the $K = N - 1$ case the fitness increases with N, but ever more slowly as N rises. If the cost is constant per locus, there comes a point beyond which growing N is no longer beneficial to the organism. This *complexity catastrophe*, Kauffman argues, is a general property of complex systems.

6.1.3 Tuneable NK landscapes

NK landscapes were actually invented to explore the region in between the two extremes already covered, and Kauffman devotes most time to this topic [17 pp. 54-60]. Mean fitness results over a range of K and N show that the complexity catastrophe kicks in as K is increased towards N (for various N), causing the final fitness attained by search to drop towards the landscape mean. Kauffman suggests that there are in fact two different regimes within the *NK* family: one in which optima remain high (when K is small relative to N); and one (for large K) in which the complexity catastrophe makes itself known. As previously reported, the fraction of fitter neighbours dwindles slowly when $K = 0$, and reduces logarithmically when $K = N - 1$. However, the move between these two extremes is quicker than might be expected: as early as $K = 2$ the fall in fitter neighbours is approximately log linear, suggesting that ruggedness is a general feature of even quite highly correlated landscapes.

Interestingly, the fitness of optima found for small values of K is actually higher than for $K = 0$ – "low levels of epistatic interaction appear to buckle the landscape … and yield fitter optima" [17] before falling again as K approaches N. Another significant result shows that, if N is increased for constant K, landscapes retain good accessible local optima. This, Kauffman suggests, "is a first hint of something like a construction requirement to make complex systems with many interacting parts which remain perfectible by mutation and selection. Each part should directly impinge on rather few other parts" [17].

Some of the most interesting results Kauffman relates pertain to non-local properties of *NK* landscapes. These are reviewed next.

6.1.3.1 A Massif Central

A surprising feature of low K landscapes is the existence of a 'Massif Central' [17 p. 60]: local optima are not distributed randomly in the genotype space but are found close to each other. In short, the fitness landscape exhibits global structure. Kauffman calculates this by measuring the Hamming distance between genotypes (how similar their alleles are). Figure 6.1 demonstrates the link between fitness and similarity of genotype. The plots show fitness against Hamming distance from the fittest local optimum found.

The distribution of basin sizes surrounding optima can be very non uniform as well: for small K, there is a tendency for the highest optima to have the largest basins. Combined with the global

structure of peak distribution, this "implies that one high local optimum has information about where other good local optima are. And further, the region between two high local optima is a good area to search for still higher local optima" [17]. As *K* increases relative to *N*, the tendency for the highest optima to have the largest basins decreases, though some do still persist.

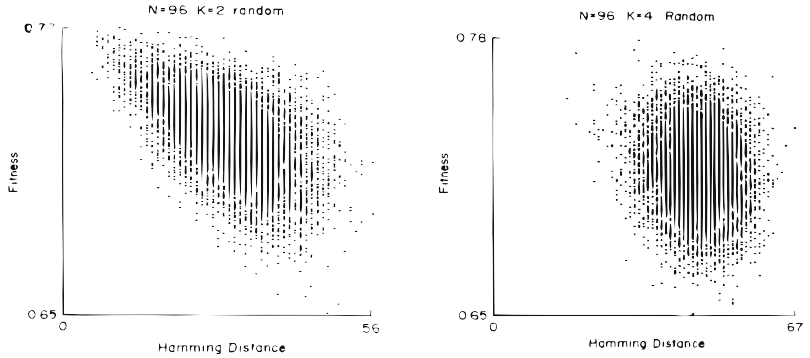


Figure 6.1 Examples of Massif Central. Diagrams show correlation between fitness and hamming distance (from [17]).

6.1.3.2 Correlation structure

As previously discussed, Kauffman defines a landscape as correlated if nearby locations have similar fitness values. By applying the autocorrelation function to measure correlation structure of *NK* landscapes, Weinberger ([41], cited in [17 p. 63]) conducted a series of experiments starting with an arbitrary genotype and then walking randomly across its surface via one mutant neighbours.

For each value of *K*, Weinberger showed an (initial) exponential drop in the autocorrelation with increasing (mutation) distance between genotypes – there is a natural correlation distance for each landscape. He also demonstrated an inverse relationship between *K* and autocorrelation, so genotypes a certain mutation distance apart will be less correlated when *K* is high.

6.1.4 Long jump adaptation

Kauffman and Levin ([40], cited in [17 pp. 69-75]) investigated whether a population could adapt successfully by making long jumps across the *NK* sequence space (landscape). (Though it is not clear what step length constitutes a long jump – analysis of *NK* landscape structure reveals hills residing on the side of hills, so a jump may be random with respect to one scale but not another.) They found a "near-universal law" [17], equivalent to Feller's theory of records [42], which states that the time taken to find each subsequent fitter solution doubles. "The critical idea is that if the searcher jumps beyond the correlation lengths of the space, then whether or not the landscape is correlated, the searcher is encountering a fully uncorrelated random landscape" [17]. On an uncorrelated landscape one expects each fitter solution found to be (on average) halfway between the rank order of the present solution and the optimum solution. With each move, the proportion of points that remain fitter halves, so (sampling randomly) it will take twice as long to find one of them.

Kauffman and Levin ([40], cited in [17]) found three adaptation timescales on rugged landscapes. Early in the acclimatisation process fitter variants near a poor solution are easy to find, but tend to be unfit themselves due to correlation within the landscape. Good distant solutions are also easy to find, and these tend to be the best choice as they are beyond the landscape's correlation distance. But after undertaking a few such long jumps, the theory of records kicks in and it becomes more efficient to search locally (within the correlation distance) for fitter solutions. The third stage takes place once the search has reached a local peak, when the search must once again try to find a better hillside some distance away (this obviously has a low success rate as the search tries to find a fitter value effectively at random). It is worth noting the similarity between this and simulated annealing, though the latter classically comprises only the first two steps.

6.1.5 Von Baer's laws, the Cambrian explosion and Permian quiescence

In well established lineages, most notably vertebrates, the early embryos of many species are more similar to each other than later stage embryos. Nascent fish, chickens and humans all look remarkably similar; embryonic humans (and other animals) develop gill slit-like features during development. These are von Baer's laws. The traditional explanation for this verity is that early stage changes are much more disruptive than those affecting later ontogeny, so mutants are much less likely to survive.

Kauffman posits that this explanation is not entirely plausible: is it really reasonable to believe that "over a time span of 600 million years, no beneficial mutants affecting early embryos should have arisen?" [40] Using Wimsatt's [43] idea of generative entrenchment (that early stage mutations cause many alterations and have more impact on development), he argues that early mutants adapt on highly uncorrelated landscapes, whereas later mutants evolve on much more correlated terrain. Thus more nascent changes are stifled by the complexity catastrophe, whereas later alterations remain relatively untroubled.

Kauffman advances related arguments to explain the vastly different paths life took during the Cambrian explosion and after the (later) Permian extinction. All of the major phyla in existence today were created, almost overnight, during the Cambrian explosion (in fact there may have been 100 phyla compared to around 30 seen today). The taxonomic tree was filled top down, with organisms first establishing phyla before filling in the lower levels. In contrast, after the Permian extinction (during which 96% of species disappeared) there is a rapid increase in species at the lower taxonomic levels, but no new phyla are created.

If one assumes that early organisms were less fit, it would have been relatively easy for life to make long jumps across a rugged fitness landscape and still find viable forms. But during the later Permian quiescence, it is likely that species suffering such random changes would be less fit and quickly become extinct.

6.1.6 The error catastrophe

Even on smooth landscapes, there is a "very general limitation" [17] to selective adaptation. As the mutation rate increases, it eventually passes a threshold – causing the *error catastrophe* – beyond which it is no longer possible to evolve towards a peak and "the population falls from rare optima toward less fit but more typical members of the ensemble" [17]. This error threshold can also be exceeded with low mutation rates simply by increasing the genome length: while selection operates at the genome level (and applies selective pressure to the genome as a whole), mutations affect each locus equally (and apply pressure to each gene individually).

Kauffman ([17], originally from [44]) outlines a simple gedanken experiment with a single diploid locus that has two alleles, $A1$ and $A2$, and genotypes $A1A1$, $A1A2$ and $A2A2$. The mutation rate from the $A2$ to $A1$ is u, and it occurs with probability v the other way. If $A1$ and $A2$ are equally fit, the equilibrium frequency of A1 is

$$f_{A1} = v / (u + v)$$

Assume $A2$ is favoured, then $A1A1$ has fitness $1 - s$, $A1A2$ $1 - s / 2$ and $A2A2$ a score of 1 ($0 \leq s \leq 1$). If s is not too close to 0, it can be shown that

$$f_{A1} = \frac{2v}{|s|}$$

The model can be extended to multiple loci. (Where the t loci each have two alleles and are considered independent and additive for calculating fitness.) To keep the maximum fitness at 1, the contribution of each locus is s/t. As t grows, the contribution of each allele decreases, but the (per site) mutation rate remains constant. Eventually mutation becomes a stronger force than selection and less favourable alleles begin to accumulate in the population. (This is despite the model assumed here, which is equivalent to an NK landscape with $K = 0$, i.e. smooth with one global optimum and no local optima.) If the forward and reverse mutation rates are equal, the expected fraction of less favourable alleles per individual due to the mutation / selection trade off is

$$f = \frac{2vt}{|s|}$$

The expected number of less favourable alleles c is

$$c = \frac{2vt^2}{|s|}$$

Or equivalently

$$v = \frac{c|s|}{2t^2}$$

This implies that, to hold the number of unfavourable alleles at a constant c per individual as the total loci t in the genome increases, the mutation rate v must decrease inversely to t^2. Stated another way, for a fixed mutation rate, the number of poor alleles accumulates with t^2 as t increases. For any model there is a critical complexity t_c above which the population can no longer obtain peak fitness and exists as a "stationary state distribution, a fixed average 'distance' from the global optimum" [17].

Below t_c, experiments have shown that the population exists as a "tight cloud centred at the global optimum" [17]. Above t_c, the population forms a thin shell a fixed distance from the optimum. Movement within the shell is selectively neutral and genotypes within it are one or two mutant neighbours of many others, so the population tends to spread evenly across the shell surface.

After presenting an analogous argument to the one above, Eigen and Schuster [44] used these results to demonstrate that, as the length of hypercyclic [44, 45] reproducing molecules increased, the system passes a threshold beyond which it is impossible to maintain the fittest variants. The population will flow away from the current best genotype and the information encoded in it will be lost. Fascinatingly, viruses appear to live close to these error thresholds [44]: predictions using error rate data for the virus Q_β suggest it could have a maximum of between 1,386 and 10,597 nucleotides; in fact, it has 4,500.

6.1.7 Conservation and innovation

It is usually assumed that must be a trade off between evolvability and sustained fitness (the balance between innovation and conservation). While it is obvious that either can be maximised at the expense of the other (§3.3, §3.4), it is not apparent whether they can jointly be optimised. Sustained fitness will be highest when the "landscape structure is tuned so that the sides of fitness peaks are steep enough to offset the mutation rate and the rate at which the landscape is deformed" [20].

Evolvability is the capacity of a population to explore a reasonable proportion of the search space. Populations with low mutation rates adapt best on smoother landscapes, while a higher rate is required for more rugged terrains. If there are too many mutations, the population ranges across the genotype space without retaining much heritable information, while if the rate is too low, the population tends to remain clustered around a local peak. Kauffman suggests evolvability [17 pp. 95-108] "may be optimised when landscape structure, mutation rate and population size are adjusted so that populations just begin to 'melt' from local regions of the space" [17] so it is able to explore the landscape while retaining much of its previously acquired fitness.

Specifically, there appears to be a phase transition between these two states. It was stated above (§6.1.6) that a population subject to mutation above t_c is able to wander neutrally throughout a given band, a "connected cluster of one-mutant near neutral genotypes" [17]. There will typically be a number of these clouds, hugging the side of peaks on the landscape. Now suppose that the mutation rate is increased so that the bands are forced down the peaks. At some point the bands will

meet and meld into one, allowing the population to wander neutrally throughout a large proportion of the landscape (and changing its behaviour radically).

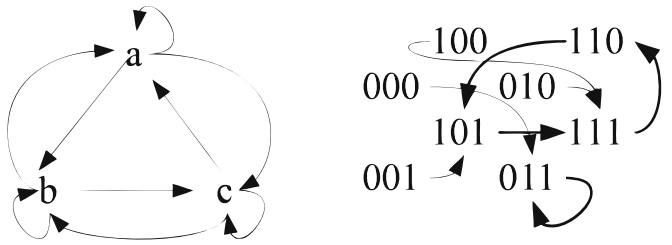
Near this phase transition a small change in the mutation rate can have dramatic effects on the population's ability to explore the landscape. Further, the population has a good chance of flowing into a basin and then climbing a high peak, so Kauffman presents a tantalising possibility that it may be possible simultaneously to optimise evolvability and sustained fitness.

6.2 Random Boolean networks

Statistical mechanics has proved to be a very powerful tool for modelling the behaviour of gases. It has allowed scientists to reason about their behaviour with considerable accuracy without needing to emulate each molecule individually. But the model is a poor one for biological systems – there are substantial differences between the two that render it unsuitable (adapted from Kauffman [17])

- The laws governing behaviour vary between elements.
- Biological systems are open, typically dissipative and often have attractors. Biological systems don't wander randomly and ergodically throughout their possibility space: consider, for example, the development of an embryo.
- Statistical mechanics can be characterised as a more or less ergodic flow within the state or phase space of a single system. Evolution is an adaptive, or drifting, process over the space of biological systems.

These differences are significant, and Kauffman argues that a new paradigm is needed adequately to describe biological systems. His suggestion is the random Boolean network (RBN). A random Boolean network is a collection of binary variables (nodes) bound together in a graph structure. Each node has a value and a logical switching rule (for example the Boolean *and* function) that controls its dynamical behaviour. The inputs of each node are outputs from other nodes in the network, and these are used to determine the nodes' next values. Each node may have a different switching function. The networks are usually assumed to be autonomous (have no external inputs) and operate synchronously. At any point, the state of the model, such as the simple three node network in Figure 6.2, is represented by a concatenation of its variables' states, for example 101. These states form the inputs to the nodes for the next transition, moving the network to 111 in this case.



abc(t)	abc(t + 1)
000	011
001	101
010	111
011	011
100	111
101	111
110	101
111	110

Figure 6.2 A small Random Boolean network and the attractor cycle followed (from [46]).

At first blush, this may seem a dangerously abstract biological model (though similar arguments could be made against the handful of equations describing an ideal gas). Kauffman [17] supplies several reasons as why the model is adequate

- RBNs have been used to model dynamical systems with thousands or millions of coupled variables, such as genetic regulatory networks (GRNs), immune systems, neural networks and autocatalytic polymer systems. The idealisation to Boolean switching elements makes the study of such enormously complex networks practicable.
- For many systems, this Boolean idealisation is either accurate or the best idealisation of non-linear behaviour. In particular, it is a good approximation for the external behaviour of sigmoidal response functions [47] such as those found in many cellular and biochemical processes (though they cannot represent the internal (intermediate) unstable steady states of their continuous progenitors).
- Due to the discrete nature of RBNs there exists a well-defined ensemble of all possible networks, so the averages of structural and behavioural properties can be assessed.
- It turns out that RBNs exhibit three major regimes of behaviours: ordered, complex and chaotic. Thus analysis of these apparently "Byzantine" [17] systems reveals unexpected simplicity, with important implications for development and evolution.

6.2.1 Dynamics of RBNs

An autonomous random Boolean network is deterministic and has a finite number of states, so (unlike continuous dynamical systems) it is guaranteed to cycle at some point. However the length of these state cycles can vary considerably, from 1 to 2^n for an n node network. These cycles partition the state space into different basins of attraction, which represent "alternate recurrent asymptotic patterns" [17] of network activity (or different system behaviours). No matter what the initial configuration, the system will be pulled down one of these basins and eventually settle down onto the attractor at its centre. As with cycle length, the size of the basin drained by each attractor can vary

substantially. Figure 6.3 shows one basin from a RBN; note the small central cycle and comparatively many states in the attractor's sphere of influence.

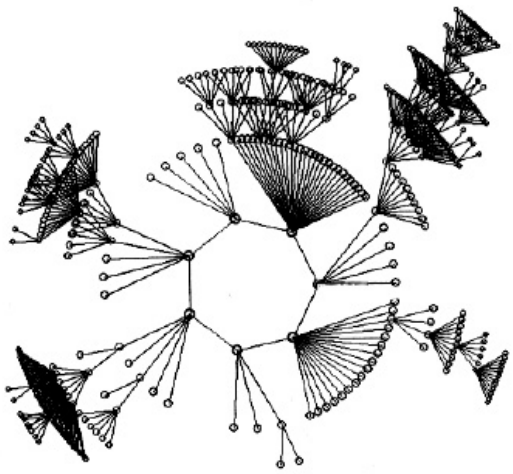


Figure 6.3 Example of RBN attractor from $K = 3$ $N = 13$ network. Attractor is loop in centre; trees lead in from 'Garden of Eden' states (from [48]).

Stability of attractors in the face of perturbations varies between networks. Kauffman defines two main categories of perturbations

- *Minimal perturbation* – transiently flipping the value of a node to a different state (for example from state 000 to 100).
- *Structural perturbation* – a permanent 'mutation' of an element's connections or the rule under which an element operates (for instance changing *and* to *or*).

6.2.1.1 Attractors and the second law

This talk of attractors (of tending to order) seems at odds with many of the ideas espoused in statistical mechanics, but Kauffman comments that, "The second law [of thermodynamics] really states that any system will tend to the maximum disorder possible, within the constraints due to the dynamics of the system" [17].

An ideal gas is ergodic, and a huge amount of work is required to box the molecules into a small corner of the state space. But – under certain circumstances – random Boolean networks can contain a few large basins draining into small attractors. The system spontaneously collapses into a small corner of the state space, and exhibits a surprising degree of order as a result.

Kauffman shows how these phenomena emerge by varying the networks' properties

- N – the number of nodes
- K – the average number of inputs to each element
- P – the bias on the elements' rules (the chance of the elements, on average, producing a 1 as output) (§6.2.1.3)

He finds that, as with *NK* landscapes, a phase transition takes place from chaotic behaviour to an ordered regime.

6.2.1.2 K = N

$K = N$ networks [17 pp. 192-194], in which every node is connected to each other, are maximally disordered and chaotic, with a median cycle length of $0.5 \times 2N / 2$. (Kauffman denotes chaotic attractors as cycles whose length increases exponentially with N, i.e. they are not actually divergent. Clearly, as the network is discrete and finite, every path will eventually repeat itself.) Minimal changes to the network, such as a transient perturbation or adopting a different starting state, result in completely different network behaviour. However, the networks do exhibit "one startling sign of order" [17]: the number of basins is small at N/e.[1] Thus a system with 10,000 elements (and cycle lengths of 2^{5000}) would only have 3700 attractors. (Though this is perhaps inevitable, as a discrete, finite network couldn't have many huge attractors – there simply aren't enough paths to go round.) Further, the basin sizes are non-uniform, with a few huge basins and many smaller ones. Since each state's successor is random (the network is fully connected and each node has an arbitrary Boolean rule), a basin's stability is proportional to its size.

6.2.1.3 K ≥ 5

Networks with between $K = 5$ and $K = N$ inputs per variable also exhibit chaotic behaviour (the same appears to be true for $K = 3$ and $K = 4$) [17 pp. 194-198]. The small number of attractors seen in $K = N$ networks persists as well.

Expected attractor cycle lengths depend on how dissipative the system is, or equivalently the extent to which trajectories converge. As a measure of convergence, Kauffman defines the internal homogeneity P of a Boolean function, the fraction of the inputs to a Boolean function that returns 0 or 1, whichever is greater than 50%. Suppose a network were constructed with a bias of $P = 0.8$, so – for any input – all functions return 1 with 80% probability. Clearly, the state 111…11 will be preferred, and many initial values will converge to it. Kauffman shows how the mean cycle length can be calculated for these *biased random mappings*

$$\text{Expected median cycle length} = 0.5\left(\frac{1}{\sqrt{P}}\right)^{N}$$

$B = \sqrt{P}$. As $B > 1$, cycle lengths increase exponentially with N

$$\text{Median cycle length} = 0.5BN \quad (5.2)$$

He says that a "critical implication of [this] is that no fixed internal homogeneity P, and hence no corresponding convergence in state space alone, will suffice to ensure that state cycles remain small as N grows large" [17].

1 e = natural log

This argument can be extended to calculate the internal homogeneity of unbiased networks for various *K*. The internal homogeneity is highest for *K* = 2 (0.6875), the mean homogeneity for all 16 binary Boolean functions) and falls towards 0.5 as *K* rises. Analysis of these results shows that attractor lengths grow approximately exponentially with network size, and increase to *N*/e as *K* approaches *N*. This is supported by numerical evidence [17 pp. 196-197].

6.2.1.4 K = 2

At K = 2 [17 pp. 198-203], a phase transition occurs and random networks suddenly display a very high degree order: in short, the attractors are few, small and have large basins surrounding them. Rather than showing cycle lengths that scale exponentially as before, the expected median cycle length drops to about $\sqrt{N}$. The number of attractors is also approximately $\sqrt{N}$. For any reasonably sized network, this represents a huge constriction: "[a] system of 10,000 elements which localises its dynamical behaviour to 100 states has restricted itself to 10^{-2998} parts of its entire state space. Here is spontaneous order indeed" [17]. A system comparable to the human genome with 25,000 elements [49] would have just 158 attractors.

This class of networks has also proved to be dynamically stable, with 80-90% of systems that are subject to a minimal perturbation flowing back onto the same attractor (and interestingly usually at the same place on the cycle it would have reached otherwise – it maintains phase). Transient changes to an element typically only affect a few other nodes so damage to the network is slight. Similarly, deleting or modifying an element usually results only in modest damage to the system. Even minimal perturbations that do change the flow from the current basin are limited to switching to neighbouring attractors. There is tremendous structure here too – a "variety of different stimuli acting on different elements … induce the same specific response" [17].

6.2.1.5 A frozen core

Kauffman [17 pp. 203-209] suggests the main reason for this spontaneous order is the development of a frozen core that divides the system into isolated islands of activity. For networks with higher *K*, he reasons that the "failure of a frozen core to percolate and leave functionally isolated unfrozen islands is a sufficient condition for chaos."

These percolating walls can be formed through *forcing structures* and *internal homogeneity clusters*. A Boolean *or* is an example of a forcing structure: if one input is true then the output will be fixed (to true), no matter what value the other input takes (this is a canalising Boolean function). If a series of forcing structures is arranged into a circle, the network develops a *forcing loop* or *descendant forcing structure*. Since these circuits are completely insensitive to (minor) perturbations they can act as frozen walls dividing the network.

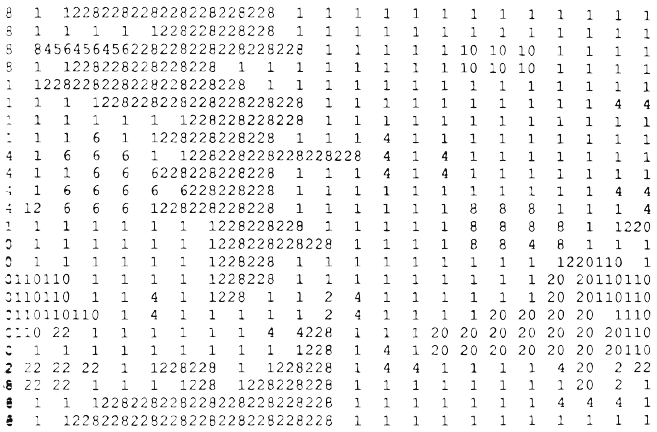


Figure 6.4 Two dimensional lattice. Sites containing 1 are frozen; note islands of activity (from [17]).

A similar chaos-to-order transition is seen when P (measure of internal homogeneity – §6.2.1.3) is altered. A number of people ([50, 51, 52], see also [17]) have studied the impact of altering P on two- and three-dimensional lattices with nearest neighbour links (a grid or cube). If P is greater than a critical value P_c (and certain states, e.g. 111…11, are sufficiently favoured) the system has a *percolating frozen core* with small embedded islands of activity. Again there is a phase transition to this state: for values less than P_c the substantially different behaviour is seen, with small islands of stillness in a writhing sea of change.

The critical value of P depends on the network structure: for a lattice (such as Figure 6.4) with $K = 4$, $P_c = 0.72$ [51, 52]; for a cube with $K = 6$, P_c is closer to 1. Kauffman distinguishes between these internal homogeneity structures and forcing clusters: homogeneity clusters, he maintains, are more general as nodes may be held in a frozen state by the joint activity of several elements – simultaneous 1s received from two neighbours may jointly guarantee that a node (or nodes) remains active. (And these nodes may similarly influence their neighbours (including the ones that 'forced' them), forming a frozen block.)

6.2.1.6 Summary of dynamics

The difference in behaviour between these ordered and chaotic regimes is analogous to the transition as K is reduced to 2. When $P < P_c$ (or equivalently $K > 2$)

- Attractors are large and grow exponentially with the network (lattice) size – they are "so large that the system can be said to behave chaotically" [17].
- Minor perturbations propagate throughout the system, affecting a large proportion of sites.
- Many perturbations drive the system onto a different attractor.
- Damage by changing the network (deleting a node or altering its function) usually affects many attractors significantly (systems "adapt on very rugged landscapes" [17]).

But if $P > P_c$ (or $K = 2$)

- There is a large frozen component that percolates throughout the lattice with a number of embedded islands of activity. These islands are isolated and cannot communicate with each other, preventing the spread of perturbations.
- Attractors are small – systems "spontaneously box themselves into very small volumes of their space and exhibit high order" [17].
- Damage does not spread through the network; system typically not altered significantly by network mutations – "such systems adapt on highly correlated landscapes" [17].

6.2.2 Annealed networks

A remarkable piece of work was carried out by Derrida [53] into the annealed approximations of Boolean networks. When a traditional (*quenched*) network is created, its elements and structure remain fixed throughout the model's life. But in Derrida's model, after each state transition the connections between each node and the nodes' Boolean functions are reallocated randomly. The only items carried over are the new state of the model and each element's identity. This is the *annealed model*.

Unlike the quenched model, which must cycle eventually, no periodic behaviour is expected. Due to the huge upheaval that takes place between each step, it would seem reasonable not to expect any ordered behaviour at all. But the annealed model has very similar properties to its quenched counterpart, converging through large basins towards small attractors for a $K = 2$ network. Derrida derives an equation that shows the distance between successive states tends to 0 for $K = 2$. For $K > 2$, there is an additional fixed point which pulls the difference between successor states towards a fixed percentage of the network size. This means that arbitrarily close starting states will diverge over time – a phase transition analogous to that seen before occurs.

The graph in Figure 6.5 shows that the phase transition takes place somewhere between $K = 2$ and $K = 3$. It may be interesting to find the precise value of K at which the transition takes place and also to examine behaviour on and around this frontier. A non-integer value of K could be effected by giving some nodes two inputs and the remainder three. Hopefully this would give the same behaviour as assigning each node a non-integer number of inputs (whatever that means); Kauffman appears to apply a similar argument when calculating P values (the high P for an unbiased $K = 2$ network is due to an equal number of functions whose outputs favour 1 and that favour 0). One possible test would be to see if (say) a mixture of 50% two input and 50% four input nodes behaves similarly to a $K = 3$ network.

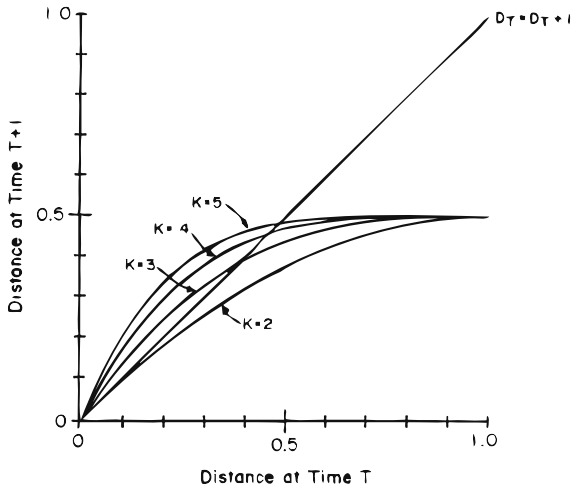


Figure 6.5 Recurrence relation showing expected distance between successive states of annealed model. A new attractor emerges when curves cross the 45° line. For $K = 2$, the recurrence curve is beneath the 45° line (from [53] via [17]).

6.2.3 Design for a brain

To motivate the evolution of Boolean networks, Kauffman introduces Ashby's Design for a Brain [54], a "delightful, elegant, and extremely clear and simple" [17] model that attempts to capture the essence of adaptation in a complex system with many interacting parts. The *System* (a coupled organism and its environment) is supposed to be deterministic, so it will flow onto an attractor and stay there. The "critical idea" [17] is that, out of the many variables that make up the organism, only a subset of essential variables is crucial to the functioning of the System. Starting the System from a given configuration, the attractor it reaches may or may not keep these essential variables in their correct ranges. If so, do nothing. If not, make a jump change to one of the organism's parameters. If this knocks the System on to an alternate attractor that does meet the specification, make no further changes; otherwise perform more random changes until it works. Ashby successfully applied his method to build a crude autopilot.

Kauffman [17] outlines the pertinence of this work to Boolean networks

- Adaptation is seen in both models as a walk in parameter space, seeking good attractors.
- Ashby's essential variables correspond to a subset of the network's variables.
- Ashby introduces the idea of *percolating walls of constancy*, the equivalent of *percolating frozen components* (§6.2.1.5).
- Jump changes in can be emulated in a RBN by mutating the logic and connections of nodes.
- Both paradigms have a *space of systems*, an ensemble of models that differ from each other by a single mutation. Adaptation is a walk through this model space.

6.2.4 Evolving random Boolean networks

Following Ashby, Kauffman defines the fitness of a network as its ability to emulate a target pattern of activity and inactivity among its N elements [17]. Using this measure, he shows that the many of the properties seen in *NK* landscapes also apply to RBNs. Data from $K = 2$ networks strongly suggests that long jump adaptation (the theory of records) holds. The complexity catastrophe also much in evidence: long jump adaptation proved much less effective as the network size was increased 20 to 100, with a statistically significant drop in final fitness.

Like *NK* landscapes, the Boolean network space is rugged with many local optima. Adaptive walks across the landscape finish on these lower peaks, well below the global optimum. This result is incredibly significant, as it implies that the network cannot adapt to any target state. "Any intuition we may have harboured that mutation and selection alone could tune attractors to arbitrary patterns of behaviour appears to be wrong" [17]. Dynamical models have been applied in many biological contexts – to neural networks, to genetic regulatory networks – in which learning or adaptation occurs by altering network weights and couplings to attain the desired attractors. But these results imply such adaptation is (in general) "extremely difficult or impossible" [17], so "either alternative means of searching rugged adaptive landscapes in networks paces must exist or adaptation and learning to not achieve arbitrary attractors" [17].

6.2.5 Adaptation towards the edge of chaos

All of Kauffman's results presented here show a phase boundary between distinct ordered and chaotic regimes. Langton's work on CAs (§5) [18] similarly demonstrated the existence of a narrow liquid region at this transition. Kauffman and Langton (and others – see [17 p. 219]) have suggested that systems capable of the most complex and interesting behaviour reside in this area. In the frozen regime, little computation can occur. In the chaotic phase, the dynamics are too disordered to be useful. But at the boundary, where the isolated islands of unfrozen elements are in "tenuous, shifting contact with each other," [17] they appear able to perform complex, controlled, parallel computation.

Straddling a phase boundary, the liquid region is a narrow one and difficult to hit by chance. Much like the spontaneous emergence theory of life's origins (see e.g. [142, 143, 144]), it would seem incredibly fortuitous that life just happened upon this fruitful degree of complexity. Kauffman [17] suggests that we did not need to be so lucky: he hypothecates that natural selection may be the force that drives complex adaptive systems inexorably towards this boundary region. Drawing on Langton's (§5, [18]) ideas, Kauffman [17] argues that systems lying in this region – systems whose structure is just melting and whose mutual information is maximised – are most capable of solving complex problems, complex problems such as adapting and evolving in a multifaceted, changing environment. There is genuine selective advantage in getting there, and it is an advantage whose efficacy increases the closer one gets to the boundary.

6.2.6 Network games

To see if they really would adapt towards the edge of chaos, Kauffman conducted an experiment in which networks were made to play a simple game [17 pp. 221-227]. The challenge selected was the mismatch game, where each player (network) is rewarded based on the difference between their binary values (so 000000 vs. 000111 pays each player ½). Competitor networks were able (through the game's construction) to sense the activity of corresponding binary values in its opponent.

The game is nontrivial with more than two players. Simply maintaining a constant output rewards the competitor 0.5 on average, so players must adopt a more subtle strategy if they want to score well. Players were allowed to evolve during the simulation; in particular they were able to change K, P and N of their network.

As expected, over time the networks alter their value of K to move towards the boundary region (for networks initially exhibiting chaotic or frozen dynamics). P is found to adjust similarly, supporting the hypothesis that liquid phase networks are best able to adapt. The most interesting result is that N also increases during play, suggesting that more complex networks are better able to solve the task than simple ones, despite the increase ruggedness of their fitness landscapes and the impending complexity catastrophe.

This increase in complexity may be because of the modular structure found in such networks. Modularity, Kauffman says, can be attained in two different ways. The first (exemplified by $K = 1$ networks) is to construct the network of structurally independent modules. Such systems comprise a collection of independent loops. The second way to achieve modularity is "radically different" [17], and relies on a percolating frozen component splitting the remaining nodes into functionally isolated islands of activity – the network is *functionally modular*.

Kauffman found that *NK* landscape models with a hierarchy of K values (so that most changes have only a small impact but a few caused large upheavals) were best able to adapt on a changing landscape. The hierarchy "yields a deep buffering" [17]: if the landscape alters slightly (a smooth landscape) the system simply tweaks its parameters to compensate; but if there is a drastic deformation the model is equally capable of rising to the occasion. Networks at the edge of chaos show just this behaviour: many mutations cause only minor revisions in behaviour, but a few cause massive changes. Thus it appears that Boolean networks evolve to evolve in two ways: they move towards the edge of chaos so they can process external changes more effectively; and they become more complex so they can process internal changes more effectively.

6.2.7 Co-evolution

The aptitude for coping with change apparently developed by Boolean networks is vital if an organism is to survive on more realistic landscapes, such as the ever changing terrain in complex co-evolutionary systems. Two types of co-evolution have been proposed: the Red Queen hypothesis ([55, 56] cited in [17 p. 242]), where an unceasing arms race takes place between the protagonists as

they struggle to maintain the same relative fitness; and evolutionary stable strategies (ESS) ([57] via [17]), where the phenotypes cease to evolve as any changes would render them less fit. Considerable effort has been devoted to determining the conditions under which each sort of co-evolution may emerge (often drawing on game theory [58 pp. 631-640]). Clearly there are often constraints on and trade offs in further evolution; Rosenzweig et al. [59] suggest constrained traits tend to adopt ESS, while those that are not relentlessly follow the Red Queen.

It is tempting to draw an analogy between the frozen and chaotic behaviour seen in Kauffman's work and ESS and the Red Queen respectively. To test this premise, Kauffman modified the *NK* genome so each species depends on *K* internal traits (as before) and an additional *C* traits from each of the *Si* species with which it interacts. He found that analogues of Nash equilibria tend to occur after some initial 'wobbles' (a sudden burst of change instigated by one partner). These wobbles become increasingly uncommon as the simulation progresses (cf. long jump adaptation).

Two principal results uncovered were

- As *K* increases relative to *C* the waiting time to hit a Nash equilibrium decreases, despite the increasing ruggedness of the landscape. There appears to be a "crude dividing line" [17] at $K = C$. When $K > C$ Nash equilibria are found rapidly, but when $K < C$ the waiting time is extended substantially.
- Increasing *N* with *K* and *C* fixed (effectively increasing the complexity of the organism) increases the time taken to hit Nash equilibria, presumably because of the density of local optima.

Kauffman also examined co-evolution amongst species with unequal *K* and *C* values (one species had higher *K* and / or *C* than the other). The most striking results were

- When *C* is high, high *K* individuals have a better mean fitness during the oscillatory period before the model reaches a Nash equilibrium than individuals with lower *K*. An individual playing against another with fixed *K* could increase its fitness by upping its own value of *K*.
- Even more remarkably, an individual with low *K* would score increasingly highly during the oscillatory period when playing another with increasingly high *K* than playing one with lower *K*. "Thus when *C* is high, increasing the *K* value of one partner helps both coevolving partners" [17].
- Individuals seem to be fitter, on average, when their *K* broadly matches *C*, so it would appear to be in a species' interest to tune its *K* to match *C* (or vice versa, though *C* "may be some more or less fixed fraction of *K*" [17]).
- Extending the model to S species shows a similar transition from ESS to the Red Queen approximately when $K = S \times C$. The waiting time to encounter a Nash equilibrium increases with the number of species.

- As S increases, the mean fitness of the coevolving partners decreases. There is also a dramatic increase in fitness fluctuations with more species in the model. The combination of lower fitness and greater fitness fluctuations makes it much more likely that some species will become extinct, lowering S and improving the situation for the remaining partners. Obviously the interaction between species in nature is more involved (in particular the graph is not fully connected) and Kauffman presents data from over 100 food webs that suggest coevolving species do in fact act to control their connectivity. The food webs appear to be largely stable and scale invariant: at lower levels each has roughly the same number of species; while higher levels aggregate similar species into single "trophic species" [17] to achieve the same outcome.

As with the independent models discussed earlier, selection in coevolving systems can act on each species independently to pull it towards and hold it at the optimal value of *K*. Though each species adapts solely for its own purposes, the move towards a mutual poised state benefits all. The "myopic adaptive agents [may modify their own behaviour], each to its own myopic advantage, [may] universally coevolve to the edge of chaos" [17].

Coevolutionary avalanches propagate through the *NK* ecosystems, and these avalanches have characteristic frequency-size distributions that change with the system parameters. In particular, the distribution is approximately power law if the system is poised at the edge of chaos (cf. self organised criticality (SOC) [60, 61]). Here, perturbations of the same initial size can unleash avalanches of a wide variety of scales. Kauffman notes that similar patterns exist in nature: avalanche size data from Raup ([62] via [17]) suggest real ecosystems show analogous behaviour, leading Kauffman to speculate tentatively that they may also have evolved to the edge of chaos. Fascinatingly, results obtained by Ray from Tierra ([63], reported in [17]) also show a "hauntingly similar" [17] distribution pattern to both Raup and Kauffman's data.

6.3 Key points

- The *NK* landscape model can be tuned to show varying degrees of ruggedness, from completely smooth to entirely uncorrelated. This ruggedness significantly affects the ability of hillclimbing to search the terrain.
- The complexity catastrophe suggests that the extra attainable fitness as the genome lengthens diminishes, and that there is a point beyond which additional complexity is no longer beneficial. This appears to be a general property of complex systems.
- Introducing a small degree of epistasis may actually be beneficial to search. Attainable optima appear slightly higher on average, and the landscape exhibits potentially exploitable global structure.
- Long jump adaptation on landscapes is equivalent of adapting on an uncorrelated landscape and presents similar problems to search.

- The error catastrophe limits the complexity that molecules can attain while still maintaining the fittest variants.
- It may be possible to overcome the trade off between evolution and conservation by optimising the search process to the landscape so that the population just begins to melt.
- Random Boolean networks may provide an effective abstract model of the behaviour of biological systems.
- Under certain circumstances, RBNs exhibit surprising order and move spontaneously into a small corner of their parameter space. This is due to a percolating frozen core surrounding isolated islands of activity.
- RBNs capable of the most complex behaviour appear to be located in a narrow band at the transition from ordered to chaotic system dynamics. Evolution may drive networks towards this region.

7 ENTROPY

Relevant entropy and thermodynamics is briefly reviewed, including its implications for life. Gibbs free energy is also introduced.

7.1 Thermodynamics

One probable[1] requirement for life is that it must have mechanisms to control energy flow [74]. Take a cell, for example. "A living cell is a dynamic structure. It grows, it moves, it synthesizes complex macromolecules, and it selectively shuttles substances in and out and between compartments. All of this activity requires energy" [75]. And all of this activity requires very precise direction and control of this energy.

When studying these energy transformations (the *thermodynamics*), there is usually one particular area, vessel or concept – a *system* – that is of interest. The system may be a cell as above, a solution in a test tube, an industrial reactor, or even a cat [74]. Anything outside the system – the entire universe apart from the cat – is called the *surroundings*. For hopefully obvious reasons, the system and surroundings are separated by a *boundary*. A boundary is defined as a division, "which may be material or not," [74] through which an "[e]xchange of work, heat or matter between a system and its surroundings occurs… A boundary may be adiabatic, isothermal, diathermanous, insulating, permeable, semipermeable." [74] Boundaries can be fixed (of constant volume) or moveable. At any point in time, the system is in a particular thermodynamic *state* (the types of molecule present, the temperature, the pressure, etc.).

A system is *open* if it can exchange heat and matter with the environment, *closed* if it can only exchange heat and *isolated* if it cannot exchange either. A sealed test tube might be a closed system, while the cat is an example of an open system. "It breathes in an exhales matter (air) continually, and it eats, drinks, defecates and urinates periodically. … Without exception, all living organisms that have ever existed are open systems" [74].

If a system exchanges energy with its surroundings, it changes its internal energy; this change is defined as ΔE. Assuming no matter is exchanged, it can be done in two ways

- Heat can be transferred to or from the system, denoted by q. A positive value indicates that heat is absorbed by the system from its surroundings. A negative value means that heat is absorbed by the surroundings from the system.
- The system or surroundings can do work, symbolised by w. This is positive if the system does work on the surroundings, and negative if the surroundings do work on the system.

1 Necessary but not sufficient. Probably.

The change in internal energy is $\Delta E = q - w$. This is the *first law of thermodynamics*, which states that energy cannot be created or destroyed (though it can be transformed) – when energy is added to the environment the same amount is taken away from the system, and vice versa. The total energy of a reaction is always constant.

To make this a little more concrete, when the cat eats, food is metabolised, releasing energy. Some of this energy is used as heat to (for instance) maintain body temperature, and the rest as work; work done includes breathing, sending nerve impulses and chasing mice.

7.2 Heat and Work

Manufacturers must determine the amount of energy contained in foods; to do this they often use a bomb calorimeter. A bomb calorimeter is a sealed vessel, immersed in a water bath, into which the food is placed and then burned. No work is done, as the reaction takes place at a constant volume (the reaction cannot do work by pushing the sides of the calorimeter out). Thus, the change in water temperature (the amount of heat transferred to the surroundings) is equal to the amount of energy released from the food, i.e. $\Delta E = q$.

However, "virtually all biochemical processes occur under conditions ... approximating constant pressure" [75]. Suppose the same reaction is carried out at constant pressure instead. The reaction is exothermic, so the temperature rises as the food burns. This causes the gases present to expand, forcing the reaction vessel to become larger to maintain the pressure.

This expansion process is quantified by the ideal gas law

$$PV = nRT$$

P = pressure, V = volume, n = number of moles[2] of gas, R = gas constant, T = temperature

Thus if the temperature increases at constant pressure (and gas molarity), the volume must increase proportionally. Work is done when the vessel expands. To push out the walls of the calorimeter against a constant external pressure, the work done is

$$w = P\Delta V$$

So less heat is transferred to the surroundings, since $q = \Delta E - w$. (Actually, work will be done by the surroundings on the system as it cools, and it might even be the case that more heat is conveyed as the overall work done by the system is negative.)

2 A mole is a specific quantity of a substance that is commonly used in chemistry. It is defined as being equivalent to "the amount of substance that contains as many elementary entities as there are atoms in exactly 0.012 kilogram of carbon 12. This quantity is known as Avogadro's number and is approximately 6.0221415×10^{23} (2002 CODATA value)" [76].

7.3 Entropy

If the cat picks up a warm sock from in front of the fire and deposits it outside, the sock will quickly cool in the cold night air; the two systems will come to thermal equilibrium. Furthermore, the equilibrium between sock and night air will persist indefinitely; it never happens that two vessels at room temperature spontaneously change so that one becomes hot and the other cold, despite the total energy of the vessels remaining constant. The first law of thermodynamics does not indicate the direction of spontaneous change, nor why it should occur in the first place.

However, something has changed to make the reaction irreversible; the *second law of thermodynamics* states what this change is and the direction it must inevitably follow. Imagine the cat has been locked in a room containing nothing but a stack of paper and a large quantity of catnip. In short order the cat flies around the room, scattering the sheets over the floor. As the cat continues to run around, the paper will become increasingly disorganised and will eventually end up distributed roughly evenly over the floor. A similar (though less dramatic) effect can be observed if dye is dropped into a glass of water: the colour spreads evenly throughout the vessel.

All systems "have a natural tendency to randomisation" [75]. The reason is that there are far more disordered states than ordered ones. There is only one way to arrange a stack of identical sheets of paper on a table, but many different ways to scatter the sheets around a room. As the number of elements in the system increases, so does the number of arranging them. The law of large numbers states that, for bigger systems, it is increasingly likely its elements will assume the average (maximally disordered) state [78]. So although it is theoretically possible for a system to resume an ordered state spontaneously, it is extremely unlikely in practice.

The degree of disorder of a system is measured by its *entropy* (S). There are a number of different ways of defining entropy, but the most useful within this context depends on the fact that a given thermodynamic state may have many substates of equal energy. The cat and paper is one example, another is gas molecules in a flask. Entropy is defined as

$$S = k \ln W$$

S = entropy, k = Boltzmann constant, W = number of substates

7.4 The Second Law and Life

Since the total entropy of an isolated system increases through spontaneous change, and the universe is an isolated system, it follows that the overall entropy in the universe must increase, that over time order must decrease, and that energy must dissipate and become unusable. Yet life appears to contradict this. "Life appears to propagate order over time. From its unquestionably simpler beginnings, the history of life as we know it has been a trajectory of increasing well-ordered complexity" [78].

Life, however, is an open system – it is not isolated and can exchange heat and matter with its surroundings. If the order in the system (the cat) increases, this must be paid for by a decrease in order in the surroundings. This typically means that the energy returned by the cat to the environment (principally through exhalation and excretion) is in a less useful form than the form in which it was taken in.

$$\Delta S_{system} + \Delta S_{surroundings} = \Delta S_{universe} > 0$$

7.5 Gibbs Free Energy

As living creatures are open systems, there is a continuous exchange of energy and matter with the environment. This makes measuring the entropy change difficult, as the system is not isolated. Additionally, both energy and entropy changes will often take place together, both of which are important in determining whether a reaction is thermodynamically favourable. The Gibbs free energy (G) is a function that is frequently used in biochemistry to determine whether a reaction is thermodynamically favourable. It is defined as

$$\Delta G = \Delta H - T\Delta S$$

G = Gibbs free energy, H = enthalpy, T = temperature, S = entropy

If the Gibbs free energy of a reaction decreases, the reaction is thermodynamically favourable. So a reaction that either results in a decrease in enthalpy[3] and / or an increase in entropy will be spontaneous. In fact, the second law of thermodynamics can be restated as, "The criterion for a favourable process in a non-isolated system, at constant temperature and pressure, is that ΔG be negative" [75]. A corollary of this is that if a process is not favourable, the reverse process will be. Processes with negative free energy changes are called *exergonic*; those for which ΔG is positive are *endergonic*.

It is possible to use the interplay between enthalpy and entropy to determine the conditions under which a process will be spontaneous. Ice changes to water above 0°C and back to ice below that temperature. While water is more disordered (has higher entropy) than ice, energy is required to break bonds within the ice crystals. At 0°C, the entropy and enthalpy terms are exactly equal. Above this temperature the entropy dominates, giving a negative ΔG and making the ice to water transition favourable. Below 0°C the converse is true, so the change from water to ice is irreversible.

From the free energy equation, it should also be apparent that, if life is to decrease its entropy through favourable reactions (and life's reactions must be favourable overall), then it must use energy to do so.

3 Enthalpy is defined as the heat absorbed by a system at constant pressure, and is the total energy of the system.

7.6 Summary of thermodynamics

- Energy transformations (in life and everywhere else) are measured with thermodynamics.
- Enthalpy is the heat absorbed by a system at constant pressure and is the important measure for living organisms.
- The second law of thermodynamics states that systems tend to disorder. Entropy is the degree of disorder in a system. Living organisms are able to maintain order over time by being open systems and increasing the disorder in their environments.
- Gibbs free energy tells us whether a reaction is thermodynamically favourable and whether it will happen spontaneously.

7.7 INFORMATION THEORY

This section gives an overview of information theory and shows how we can model life in these terms. Shannon entropy is introduced, along with joint entropies and information. We see how to use information theory to transmit data over a noisy channel without error.

7.8 Information theory

In §5, we introduced Langton's λ parameter and saw how the most complex behaviour exists in a narrow region between order and chaos, a region that supports both information retention and information transmission. Kauffman also found that the most interesting random Boolean networks reside on the edge of chaos, and suggests that this is the region most likely to support complex computation and the emergence of life.

Information retention and information transmission are both essential for life. Life needs to store information about how to survive: how to eat, move and respire. Life also needs to transmit information to and receive information from the environment: information about food, temperature, chemicals, light, movement and much more. "This connection is a universal trait of all living systems." [78]

Though hugely complex, we can investigate parts of this subject through information theory. Information theory was developed by Claude Shannon to calculate how to transmit and receive messages accurately over noisy channels, though it is far more widely applicable [79].

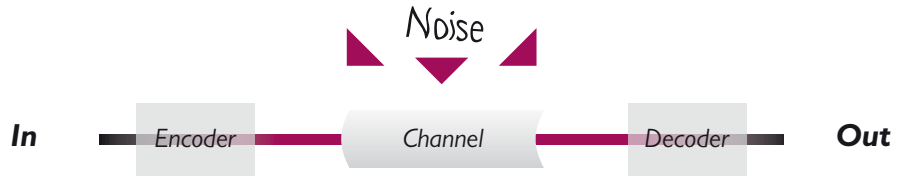


Figure 7.1 A communication channel (adapted from [78]).

Adami points out that there is a "precise analogue" [78] between the communication channel in Figure 7.1 and living organisms. Genes encode information about the environment, and this information is transmitted to the next generation through a DNA, RNA or protein channel. The channel is subject to noise in the form of replication errors and stresses from heat, light and viruses. Finally the message is decoded again through the expression of proteins useful for the host in the environment.

We would like to transmit information between generations without error to avoid potentially lethal mutations to the organism's offspring. Imagine we have a binary channel and there is a 20% chance of each bit changing during transmission. A 20% error rate is too high for most situations, and would almost certainly result in lethal mutations in the next generation if we were modelling

reproduction with this channel. Information theory shows how we can add redundancy to the channel to ensure our message is received with approaching 100% accuracy.[4] To calculate the optimal encoding for a channel, we need measures of *information* and *uncertainty*.

7.9 Shannon entropy, joint and conditional entropies

Suppose we have a *random variable X* that can be in states $x_1, \ldots, x_n$ with probabilities $p_1, \ldots, p_n$ and that $\sum_i^N p_i = 1$. The *Shannon entropy* [73] is defined to be

$$H(X) = -\sum_i^N p_i \log_b p_i$$

As we are dealing with binary models, we use logarithms of base 2. $H(X)$ is required to be monotonic (it must be higher if there are more possible states in the system) and additive (the entropy of two unrelated systems must be equal to the sum of the entropies of each).

Although the joint entropy of two unrelated systems must be equal to the sum of their entropies, most interesting composites will show some degree of correlation between its component systems. If X and Y are random variables with N and N' states respectively, we can define the probability that X will be in state x_i and Y *jointly* in state y_j as

$$P(X = x_i \text{ and } Y = y_j) = p(x_i, y_j)$$

This allows us to define the *joint entropy*

$$H(X, Y) = -\sum_i^N \sum_j^{N'} p(x_i, y_j) \log p(x_i, y_j)$$

We can show that

$$H(X, Y) \leq H(X) + H(Y)$$

The joint entropy of X and Y will only be the sum of their individual entropies if X and Y are uncorrelated; in all other cases it will be less. To see why, suppose that the cat, Y,[5] is in a room with a mouse, X. The mouse decides that the best way to avoid being eaten is to move randomly throughout the room, so there is an equal chance of it being at any location.[6] Unfortunately the cat spent too long playing with a catnip-impregnated toy before entering the room, so it too is moving randomly.

We take a picture of the room and divide the photo into N equal squares. The entropy of the mouse is $\log(N)$. The movement of the cat is uncorrelated with the mouse's motion and also has entropy $\log(N)$, giving a joint entropy of $2 \log(N)$.

4 In his Fundamental Theorem, Shannon showed that we can achieve arbitrarily accurate transmission through a noisy channel by reducing the transmission rate to the channel capacity. In [78], Adami shows how to calculate the information transmission capacity of genomes.

5 The cat's owners were notoriously unimaginative when it came to names, as their children, C1, C2 and C3, would testify.

6 We ignore edge effects, time taken to change direction, etc.

Now suppose that the cat recovers from the catnip and starts chasing the mouse. Its agility means the cat is always in a square immediately adjacent to the mouse, so when we take our next picture the entropy of the mouse is still log(N), but the cat has only eight choices, giving a joint entropy of log(N) + log(8).

The cat and mouse are now tightly correlated, and we can use this to define the conditional probability for the cat. We know that

$p(y_j \mid x_i) = ⅛$ if the cat is adjacent to the mouse

$p(y_j \mid x_i) = 0$ if the cat is not adjacent to the mouse

We can use this to define the conditional entropy of the cat. If we know that $X = x_i$

$$H(Y \mid X = x_i) = -\sum_j^{N'} p(y_j \mid x_i) \log p(y_j \mid x_i)$$

The conditional entropy for any X is

$$H(Y \mid X) = -\sum_i^{N} \sum_j^{N'} p(x_i)\, H(Y \mid x_i)$$

Which can also be rewritten using Bayes' Theorem as

$$H(Y \mid X) = -\sum_i^{N} \sum_j^{N'} p(x_i, y_j) \log p(y_j \mid x_i)$$

More generally, we can state the conditional and joint entropies as

$$H(X, Y) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y)$$

7.10 Information

Information is the *mutual entropy*, or *correlation entropy*, between two random variables (or two sets of random variables).

$$I(X : Y) = H(X) + H(Y) - H(X, Y)$$

This can be rewritten as

$$I(X : Y) + H(X, Y) = H(X) + H(Y)$$

So the correlation between the variables (the decrease in randomness between them) appears as information. This is information shared between the variables.

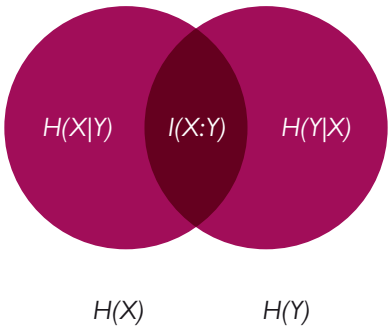


Figure 7.2 Entropy for random variables *X* and *Y*.

7.11 Noiseless encoding

We can now use these definitions of entropy and information to see if a code is sufficient to transmit a message without error. Suppose again that we have a random variable X that can be in states $x_1, \ldots, x_n$ with probabilities $p_1, \ldots, p_n$. We have to encode the messages $x_1, \ldots, x_n$ before we can transmit them. Longer messages will be more susceptible to noise, so we want to minimise average codeword length. But we must still be able to decipher messages uniquely – for instance, with the codewords 0, 01 and 010 in our language, it would be impossible to tell if the string 010 was one codeword or two. More generally, with alphabet size D and codeword lengths ni, a way of uniquely encoding messages exists if

$$\sum_i^N D^{-ni} \leq 1$$

But this only tells us if a code exists, not how well it copes with noise. Given the entropy of the source, Shannon's Noiseless Coding Theorem [73] gives a lower bound on the average length of codewords.

$$\langle n \rangle \geq \frac{H(X)}{\log D} = -\sum_i^N p_i \log_D p_i$$

7.12 Key points

- Information theory can be used to model living systems: genes encode information about the environment and this information is transmitted between generations through a DNA or protein channel.
- The Shannon entropy measures our knowledge of a system's state. A high entropy shows a high uncertainty about the system's state.
- The joint entropy of two correlated systems will be lower than the sum of their individual entropies.
- Mutual information measures the correlation between two systems.

8 EMERGENCE REVIEW

This section looks at emergence. After describing the allure of emergent properties, we discuss its many definitions, both in meaning and scope, and introduce a framework for our use here, developed further in the next chapter.

8.1 Why emergence?

Emergence is "the arising of novel and coherent structures, patterns and properties during the process of self-organisation in complex systems." [80] Locally acting rules can combine to create unexpected large-scale emergent behaviour.

We find emergence everywhere. We see it in nature in the forms of ant colonies and termite mounds, as friction and in flocking birds [82] and schooling fish [83]. We see it in economics [84], in games such as poker and in architecture [25, 26].

Emergent biological systems have many desirable properties that people try to engineer into products and services, including fault tolerance, robustness and adaptability. Unfortunately, the drive to incorporate these properties into traditionally-developed systems has led to "more or less wild claims that all we need to do to get these properties is allow them to emerge, by stripping away the centralised control" [85], which "is more a recipe for anarchy, than for emergence." [85]

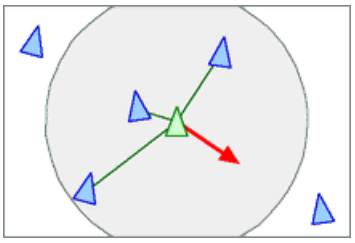
Traditional development techniques often make it difficult to exploit the potential of good emergent properties in a methodical way. Indeed many techniques, such as formal development, seek to eliminate emergent properties from the system so it meets the specification. One reason is that emergent systems are generally irreducible, making it impossible to separate them into meaningful constituent parts or to combine simpler elements to build up functionality of the system piece by piece. Though the emergent behaviour is part of the whole system, we cannot attribute aspects of it to particular pieces of the system.

Before exploring the subject in greater depth, we introduce what is perhaps the archetypical example of emergence, flocking.

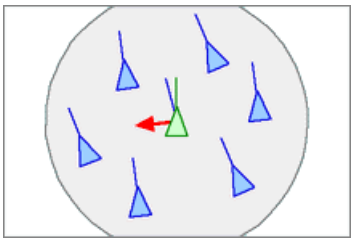
8.2 Modelling flocking behaviour

Reynolds' Boids [82] are a "particularly evocative example of emergence" [82], using just three simple rules to generate remarkably realistic flocking behaviour. A number of identical boids are placed into a 3D environment (similar experiments have also been done in 2D). The boids are left to fly around, subject to these three rules (images adapted from [82])

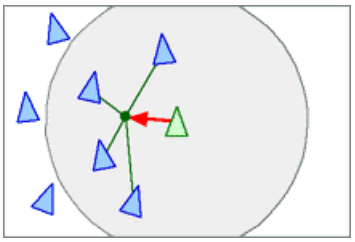
- *Separation* – steer to avoid crowding nearby boids in flock



- *Alignment* – steer towards the average heading of local boids in flock



- *Cohesion* – steer to move toward the average position of local boids



There are no special boids here with 'leader' or 'organiser' roles, and each boid only has information about its local environment. Yet remarkably, this setup leads to a "goofy kind of flocking" [86] that nonetheless seems uncannily real.

Perhaps even more surprisingly, the flock continues to behave believably when the boids are placed in novel environments. When confronted by an obstacle, the flock splits to go round it and then reforms on the other side. There is nothing explicit in the rules to prescribe this robust – and very realistic-looking – response, yet Reynolds' model copes with this and many other situations with no special conditions or extra rules: we get all of this for free (§9.19-§9.24).

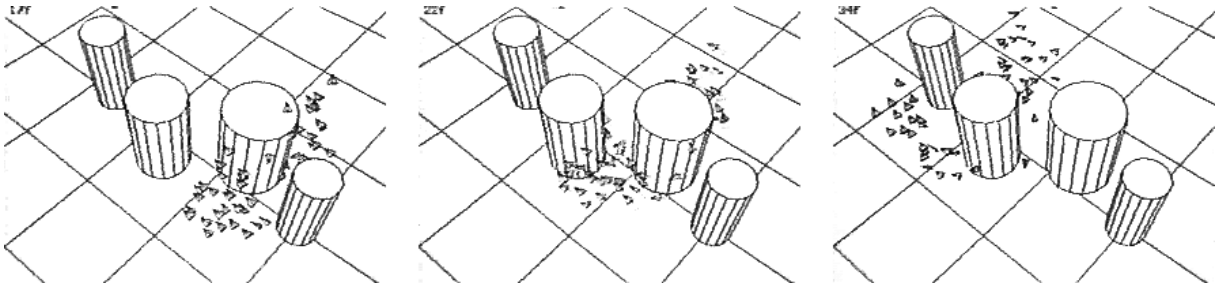


Figure 8.1 Boids moving round pillars. The flock splits and reforms after passing the obstacles. (Stills from a video at [82].)

8.3 Defining emergence

Emergence has interested philosophers (and, latterly, scientists) for hundreds of years. Aristotle [87] provides one of the earliest references, recognising "things which have several parts and in which

the totality is not, as it were, a mere heap, but the whole is something beside the parts." In other words, something that is *more than the sum of its parts*. Perhaps the first complete definition comes from George Lewes, who said

> *"Although each effect is the resultant of its components, we cannot always trace the steps of the process, so as to see in the product the mode of operation of each factor. In the latter case, we propose to call the effect an emergent. It arises out of the combined agencies, but in a form which does not display the agents in action." [88]*

There is considerable debate about how real emergence is: is it a real phenomenon where emergent models can be causal, or is it just an epiphenomenon that results from lower level behaviour?

Abbott argues that emergent behaviours are epiphenomena, which he defines as "[a] phenomenon that can be described independently of the underlying phenomena that bring it about." [89] He uses Brownian motion as an illustrative example: while dust particles appear on the surface of water to move randomly and independently, Einstein [90] explained how this movement is an epiphenomenon of the particles' collisions with atoms or molecules in the water.

Though the idea of a life force has long been discredited, many take issue with a hierarchy of sciences, where "science[n + 1] is just applied science[n]" [91]. "At each [level] entirely new laws, concepts, and generalization are necessary. … Psychology is not applied biology, nor is biology applied chemistry. … The whole becomes not only more than but very different from the sum of its parts. … [The] ability to reduce everything to simple fundamental laws … implies the ability to start from those laws and reconstruct the universe." [91], cited in [89]

According to the constructionist hypothesis, we can build up any phenomenon (such as the rules of biology or chemistry) from more fundamental rules (usually the laws of physics). Many authors are unhappy with constructionism, sometimes for lack of evidence (no-one has shown, for example, how to construct a human emotion from the laws of physics) and sometimes because it is not clear what we should use as building blocks. But the alternative is vitalism, and "Henry Bergson and Darth Vader notwithstanding, there is no life force." [92]

This is a rather stark choice – we either accept strict constructionism or subscribe to vitalism – and many authors have attempted to find ways of recasting the problem to avoid this difficulty. Campbell and Bickhard [93] start from Kim's ([94] and others) argument that reductive physicalism cannot explain everything, citing the failure of Carnap's attempts to translate all mental terms into physical states. They are not convinced that this is possible in principle either, even given a perfect understanding of the mental and physical states. They argue that, physicalists "cannot point to any plausible bridging laws that are effective in explaining such phenomena [,] their doctrine is mere dogma."[1] [93]

1 Though we note that Campbell and Bickhard provide no substantive evidence to support their alternative theory either.

Over time, scientists have discovered new (apparently) fundamental particles, moving from atoms to protons and electrons to quarks. Of course we could change our model every time new fundamental particles are discovered and insist that what we previously believed to be phenomena are now epiphenomena. While changing fundamental models happens very rarely, it does still not seem very principled to distinguish phenomena and epiphenomena by whether they are defined in terms of the most fundamental particles we happen to know about at the time.

And even with the particles we currently have (leaving aside the question of whether they are actually fundamental or not), it is difficult to explain behaviour such as the Pauli exclusion principle. Quantum theory also suggests that there are no particles, or at least not in the classic (physicalist) sense of the term. Campbell and Bickhard [93] point out that this leaves us susceptible to *causal drain*, as there is now no basis in which to define real phenomena – it's epiphenomena all the way down.

Having dismissed both reductionism and vitalism, Campbell and Bickhard [93] present their alternative, arguing for a process-based model. They suggest that particles are insufficient as fundamental components; we must consider their configuration too, and we must consider these as being just as fundamental as the particles. "[T]here is the pattern of the relationship between [a proton and an electron], and that pattern of the process, its organization, is what is crucial to the emergent properties of hydrogen."

8.4 Downward causation

Downward causation posits that emergent phenomena can be genuine phenomena, rather than just epiphenomena, and that these emergent phenomena can produce behaviour at a lower level. Campbell and Bickhard say that there is "no question downward causation exists." [93] Building on their organisational model of emergence, they posit that "if everything is configurational, there is no reason not to accept that complex configuration can generate properties and powers that are genuinely emergent." Unfortunately Campbell and Bickhard offer no reason to accept that complex configuration can generate causal emergent properties either, as their examples are easy to explain without resorting to high level phenomena or downward causation. Two examples they give are

- The location of iron atoms in a wheel is dependent on the movement of the wheel as a whole. Thus, they argue, the wheel is downwardly causal on the atoms' positions.
- A candle keeps itself alive by supplying itself with wax and oxygen and keeping itself at the right temperature. The self-maintaining candle flame is an organisation that is an *emergent causal power*.

It is true that these "cannot be explained simply as the physical resultant of the causal properties of its distinct constituents" [93], but they can be explained if we project properties of the underlying models (physical and configurational) to form the emergent model, much as we shall do later

with cellular automata (§10.1). The position of the wheel atoms is determined over time by their relationship with the other wheel atoms. If we model the bonds between and location of the iron atoms, we can calculate the position of any constituent atom. This process is hugely expensive and inefficient, but it does not require downward causation to work. A similar argument can be made for the candle.

Bedau [96] also advocates downward causation, at least partially. He defines three types of emergence: nominal emergence (a property of the macro system that the micro system cannot have – a circle is nominally emergent because its constituent points have no shape); strong emergence (systems with downward causation, in which the macro level effects influence behaviour at both the macro and micro levels); and weak emergence (systems that are causally dependent on and reducible to the low level, but that have explanatory independence and irreducibility).

Bedau dismisses strong emergence, stating that "[a]ll the evidence today suggests that strong emergence is scientifically irrelevant. … Strong emergence starts where scientific explanation ends." [96] He separates weakly and nominally emergent systems by insisting that weakly emergent systems must be underivable more efficiently than by simulation.[2] (This is in the general case, so specific setups of a system for which we can predict the outcome, or indeed setups for which we have previously noted the outcome, do not disqualify a system.)

Despite requiring the high level behaviour of his weakly emergent systems to be ontologically epiphenomenal, Bedau argues that, as weakly emergent systems are explanatorily autonomous and irreducible, they exhibit downwards causation in terms of their explanation. Or, to put it more succinctly, if we can't easily understand what's going on then we should class it as downwardly causal. We later suggest that emergence is a relative concept (there are no correct emergent properties – §9.3) and using downward causation as an intellectual sleight of hand may be useful in some circumstances, but again Bedau's examples can easily be explained without it.

Bedau uses a glider gun in Conway's Life (§2.4) to illustrate explanatory downward causation. A glider gun emits gliders at regular intervals, and Bedau argues that the production of these gliders – and the gliders' affects on the CA's cells – is caused by the glider gun as a whole. "Clearly, this repeating pattern [of a specific cell]'s behaviour is caused by the macro-level glider gun." [96] And while he acknowledges that the glider gun's behaviour is ontologically caused by the aggregation of the CA's cells causal histories, he believes that the macro explanation is autonomous from the aggregate micro explanation. "There is a macro explanation that is not reducible to the aggregation of micro histories. If those micro histories had been different, the macro explanation could still have been true." [96]

While not inherently wrong, Bedau's downward causation could be quite confusing, encouraging us to view system behaviour with reference to a causality that is not actually true at all. It's also

2 Either in principle or those that can theoretically be derived but must be simulated in practice for "a slightly weaker notion of emergence".

unnecessary: if, instead of using an ontologically false macro model of the system, we choose to project a lower dimensional model of the low level system to the emergent level (removing degrees of freedom we don't want), we will end up with a model that can naturally be described in terms of the high level behaviour and is insensitive to a change in its micro histories.[3] We describe how we can use this approach in Life to move from cells to a glider in §10.2. "No matter how real [Game of Life] patterns look, interaction among them is always epiphenomenal." [89]

8.5 Supervenience and emergence

Anderson contends supervenience is closely related to emergence. "H supervenes on (or over) L if it is never the case that two states of affairs will assign the same configuration of values to the elements of L but different configuration of values to the elements of H." Abbot [89] illustrates supervenience by supposing we have five bits in the low level L and that the high level H requires that an odd number of bits are on. Here, H supervenes over L because the truth values of the bits in L always determine truth of H. But H stops supervening L if we exclude bit 5 from L: bits 1 to 4 can be still true (so L is also still true), but if bit 5 is false then H will now be false.

While ruling out downward causation, Abbott [89] does allow emergent properties to be downward entailing. Since a Turing machine can be emulated by Conway's Life, it is one of Life's possible epiphenomena. But this also tells us something about Life: we know that the halting problem is unsolvable on a Turing machine, and as Life can implement a Turing machine, the halting problem must be unsolvable in Life too. "Thus a consequence of downward entailment is that reducibility cuts both ways. … We reach that conclusion by reasoning about the higher level as an independent abstraction and then reconnecting that abstraction to the lower level." [89]

8.6 Levels and complexity in emergence

Emergent behaviour is often described as the global behaviour of the system, appearing out of the simple, local interactions between the system's components, and many definitions talk of levels and of the increased complexity of emergence – emergent properties appear at more complex level.

In their review of emergence, Stepney et al. [85] suggest that "[t]he emergent properties form higher level structures (of patterns, of agents) in space and time. These higher level agents have their own structure and dynamics, their own (longer) length- and timescales. Longer timescales allows relative stability of higher level patterns." They quote Burns et al. [97], who introduced timebands to formalise this notion of different timescales at different levels: "The slower [higher level] band (A) can be taken to be unchanging (constant) for most issues of concern to B. … At the other extreme, behaviours in [the lower] band C are assumed to be instantaneous". Bickhard and Campbell [98]

3 Insensitive to the extent that Bedau's macro explanation is, at least.

note that, in physical systems at least, "successively higher levels often require successively lower temperatures to emerge … each level 'condenses' out of lower levels".

Anderson [91] links levels to complexity, and says that "[a]t each level of complexity entirely new properties appear" and uses this definition to reject constructionism. "The constructionist hypothesis breaks down when confronted with the twin difficulties of scale and complexity. The behaviour of large and complex aggregates of elementary particles, it turns out, is not to be understood in terms of a simple extrapolation of the properties of a few particles. Instead, at each new level of complexity entirely new properties appear, and the understanding of the new behaviors requires research which I think is as fundamental in its nature as any other." [91]

Other authors reject the link between emergence and complexity. Bar-Yam [99] provides several examples of emergent simplicity and Ryan [100] extends this to the general case, defining weak emergence (§8.8) as a loss of resolution (a definition we use in §9 and in subsequent chapters).

Shalizi defines an emergent process as one that "has a greater predictive efficiency than the process it derives from." [95] Rather than define emergence as intrinsically complex, Shalizi seeks "to filter out everything we can – get rid of all the small-but-significant inputs – so as to simplify the relationship. We are not trying to explain everything we can measure; we are trying to find what's intrinsically important in our measurements." [95] This is a very important point that we return to in §9.5, §10.12 and elsewhere. Shalizi's model is a thoughtful and thorough one in many ways, not least because he shows how one can quantify emergence (something we also discuss in §9.11 and develop further in §11).

8.7 Emergence as a change of scope or resolution

We take issue with levels as a requirement for emergence, and even that levels exist in emergent systems. Levels are useful pedagogically and provide a convenient shorthand (indeed, we shall continue to use the term extensively here (even when discussing theories of emergence without levels) and in later chapters), but they throw up more questions than they answer: What is a level? Are they ontological or epistemological? Is a certain degree of separation needed between them before they count as distinct levels? Levels smack rather of vitalism (Where do the levels come from?) and seem to draw from the "I know it when I see it" school of thought [101], rather than from rigorous (or indeed any) first principles.

Ryan also argues against levels in emergence. "The conventional explanation of emergence presented in the previous section is unsatisfactory. The use of an emergence hierarchy to account for emergent properties is alarmingly circular, given that the levels are defined by the existence of emergent properties. In hierarchy theory, levels are most often considered to be epistemic, although seemingly only to avoid the burden of proof that falls on an ontological position." [100]

He replaces levels with scope and resolution. System scope is defined by a spatial boundary, which could be conceptual, physical or formal. System resolution is the finest spatial distinction between two alternative system configurations.

Ryan stipulates that the scope of the high level[4] must be at least that of the low level, and that the resolution of the high level must be at most that of the low level. He also requires that the levels have different resolutions and scopes.

$$R_{\mathrm{M}} \leq R_{\mu}$$

$$S_{\mathrm{M}} \geq S_{\mu}$$

$$(R_{\mathrm{M}}, S_{M}) \neq (R_{\mu}, S_{\mu})$$

R = resolution, S = scope, M = macrostate, μ = microstate

8.8 Weak emergence

Weak emergent properties describe the relationship between microscopic low level behaviour and macroscopic high level collective behaviour where both levels share the same scope. Bar-Yam refers to this as "parts with positions to the whole" [99]. For the model to be emergent in Ryan's model, the resolution of the low level must be greater than that of the high level.[5]

If $S_{\mathrm{M}} = S_{\mu}$ then it follows that $R_{\mathrm{M}} < R_{\mu}$. Bar-Yam lists pressure, temperature, patterns on animal skins and traffic jams as (likely[6]) examples of weak emergence [99].

A corollary of Ryan's definition of emergence is that weak emergent properties are epistemic, as once we discover the mapping they are no longer considered emergent. Unfortunately Ryan appears to conflate mappings and paths: having a mapping between the high and low levels does not imply there is a path we can traverse between the two. Such a path allows us (theoretically) to construct a model of the system at any point between the two levels, whereas a mapping merely asserts there is a relationship between the levels. In §9.6, we assert that such a path cannot exist in (weakly) emergent systems, though mappings can and do.

Goldstein [80] also questions whether emergence is merely a provisional construct, that it is "simply an epistemic recognition of the inadequacy of any current theory for deriving macro-level properties from micro-level determinants." However, he also points out that complexity theory has built-in

4 As stated previously, we continue to use the term 'level' to identify the macro- and microstates, even though we define them in terms of scope and resolution.

5 Ryan and Bar-Yam both note that scale is not the same thing as resolution: scale is ontological and resolution is epistemological. Resolution determines the size of things we can detect (without changing their size in the model), whereas scale changes the size of the things we detect (without changing the minimum size that is detectable).

6 He adds a caveat that modelling some of these properties may require strong emergence.

limitations to predictability due to the nonlinearity of complex systems, giving examples of strange attractors (§4.10) and Conway's Life [36]. "As a result, it seems that emergence is now here to stay."

8.9 Strong emergence

Ryan says that a strong emergent property only exists at the emergent level. It cannot be identified without looking at all parts of the system together – it is only there in the collective. Ryan stipulates that the resolutions of the high and low levels should be identical for strong emergence, and so it follows that $S_M > S_\mu$.

He uses a secret scheme as an example. Imagine a secret sharing scheme that divides the secret into n pieces such that

- The secret is easily determined with knowledge of k or more pieces of the secret.
- The secret is completely undetermined with knowledge of $k - 1$ or fewer pieces.
- We can only know the secret after increasing the scope to k pieces or larger, so the secret is a strong emergent property of the system.

Bar-Yam calls this *type 3 strong emergence*. He also identifies *type 2 strong emergence*, which falls somewhere in between type 3 (Ryan's) strong emergence and weak emergence. The scopes of the high and low levels are the same, but he insists on the presence of ensembles, or global constraints that determine the properties of the whole system. There is no strong emergence when constraints act on only some components of the system. Bar-Yam gives examples of a fixed number of players on a team, quota filling (filling seats in an auditorium) and steady-state flows as examples of type 2 strong emergence.

8.10 Dynamics and emergence

Several authors have suggested a link between emergence and dynamical systems. Drawing on Broad's work [102], Newman [103] gives a "careful definition" [85] of emergence. His definition falls within Ryan's strong emergence (§8.9), and broadly states that

- An emergent system comprises structured entities in a relation.
- It is impossible to identify an emergent property of the system without looking at the system as a whole.

(The actual wording is significantly more precise.) Newman suggests that being in the basin of a chaotic attractor meets his definition, as the attractor supervenes on all variables that make up the system; and that we cannot identify the attractor from just some of its states because these cannot be finitely defined.

In a similar vein, Goldstein asserts that dynamics is one of the "common properties that identify [systems] as emergent." [80] He argues that "[e]mergence requires systems with at least the following characteristics" [80]: nonlinearity, self-organisation, non-equilibrium systems and the presence attractors, and that "[n]ew attractors show themselves when a dynamical system bifurcates, this event signifying both a quantitative and a qualitative metamorphosis." [80]

Crutchfield also links dynamics and emergence in his ε-machines model, proposing "a synthesis of tools from dynamical systems, computation, and inductive inference" [104] and, like Goldstein, believes that chaotic attractors are an example of emergence. He characterises three stages of emergence

- "The intuitive definition of emergence: "something new appears";
- Pattern formation: an observer identifies "organisation" in a dynamical system; and
- Intrinsic emergence: the system itself capitalises on patterns that appear." [104]

Flocking is an example of intrinsic emergence, as the group behaviour is identified and used by the birds in the flock.

Crutchfield introduces ε-machines to model emergent behaviour. There are four levels of ε-machines in Crutchfield's hierarchy, somewhat equivalent to the Chomsky hierarchy, though levels 0-2 (data stream, tree and finite automata) all fall within Chomsky's level 3 (finite automata), and Crutchfield's level 3 has the power of a Turing machine (Chomsky's level 0). Crutchfield's ε-machines are stochastic: he uses Bernoulli-Turing machines and statistical finite state automata to stop random behaviour from appearing maximally complex [17, 18].

Crutchfield uses ε-machine reconstruction to build a model of a system's behaviour. He starts by trying to emulate the system with a level 0 ε-machine. If the language is not expressive enough to produce a finite model, he moves up to the next level in his ε-machine hierarchy and tries again. The aim is to create a "minimal model at the least computational powerful level yielding a finite description" [104]. As with Langton's and Kauffmann's work [17, 18], Crutchfield's results show maximum complexity at a point between order and chaos.

Shalizi [95] builds on Crutchfield's work on ε-machines, developing a new creation method that he shows to be minimally stochastic and argues should include a minimal set of states. (This differs from Crutchfield's algorithm, which Shalizi says constructs the "the most complicated [model] that can be devised, given the length of histories available to the algorithm" [95]). A Shalizi ε-machine models the entire underlying process, so we can divide it up into sub-machines (strongly-connected components) and, at each timestep, output a symbol unique to that sub-machine or, if moving between sub-machines, output a symbol for that transition. Thus, "[if] the sub-machines have been chosen appropriately" [95], the sub-machines are emergent structures for the underlying model.

Finally, we have already seen Kauffmann's extensive analysis of RBNs, which he believes may be simplified models of gene regulatory networks (GRNs). "He analyses the structure and stability of their attractor spaces, and draws an analogy between these attractors and cell types: maybe somehow cells 'are' the attractors of GRNs." [85]

8.11 Abstractions in flocking

Earlier we saw that Reynolds' simple flocking rules performed surprisingly well when faced with novel environmental obstacles. Commentators disagree over whether his flocking rules are actually followed by birds and fish, or whether they are merely an uncanny facsimile. If it is the former then these three simple rules demonstrably operate successfully in a very complex world. If it is the latter then the case is less strong, but this flocking model has still been shown to be effective in a range of quite complicated and diverse simulations.

We argue that these flocking rules are so effective because there is a very high degree of abstraction within the model. There are two distinct abstractions in boids' flocking behaviour: between the boids and the flock, and between the flock (or equivalently boids) and the environment. At a boid level, the rules work by ruthlessly removing unnecessary detail: there are two classes of object – boid and non-boid – and any objects beyond a certain distance are ignored. And the flock effectively makes the same abstractions: it is not concerned with non-flock objects, and distant objects are similarly ignored.

Thus the environments considered by both boid and flock are drastically simplified, and there is no need to account for the difference between, for instance, a high, thin pillar and a low, wide wall in the rules.[7] We have much more robust behaviour than we would have otherwise through choosing the right abstraction: the model is robust *specifically because* we ignore information. This is something we expand on in §9.20.

8.12 Evolving flocking

Having described group behaviour rules for flocking, it is natural for researchers to investigate whether they can evolve similar phenomena, and indeed a number of papers have been written on the subject.

Zaera et al. [105] discuss the difficulties they encountered attempting to make fish display collective emergent behaviour. Although they succeeded in evolving simpler dispersal and aggregation behaviours (constituents of Reynolds' flocking), they were not able to persuade their fish to school. Sometimes their fish would show "degenerate" [105] schooling behaviour, where the fish would swim in tight circles to satisfy the fitness function.

7 Or, for that matter, the difference between a red pillar and a green pillar. Just because we can notice something does not mean that we should. In fact, most of the time we should ignore things unless given a good reason not to.

The authors report that they were unable to find a quantitative measure of schooling in the literature, and their own compound function (a collection of Gaussian measures favouring 'desirable' schooling qualities) proved inadequate: although it rewarded correct schooling, it also favoured many other behaviours as well. Zaera et al. suggest this is because their fitness function was not complex enough, and claim that creating an effective function (one that recognises schooling but nothing else) is likely to prove at least as difficult – and the resultant fitness function at least as complex – as hand crafting the behaviour.

In fact, it may be considerably more difficult than that. The authors argue that there is no objective definition of schooling, that the appellation of boids' collective behaviour is subjective. They contend that, even if one assumes an implicit definition – "if a group of agents do what [boids] do, then the group is schooling" [105] – it is difficult to formulate an effective quantitative fitness function that gives a useful indication of progress towards schooling. However, it could be argued that this is inevitable – by definition of the phenomenon being emergent, it is not possible to provide a nice, graded progress indicator in terms of lower level behaviour. We discuss mappings and development paths further in §9.5-§9.8.

There is another way. And there are actually incremental fitness steps towards realistic schooling behaviour, though not in the direct sense supposed in Zaera et al.'s system. But first we must expand our model – make it even more complex – so we have a reason for schooling. Biologists have suggested several motives for such collective behaviour, including reducing the risk of being eaten, making the search for food easier, increasing mating efficiency, and creating an environment for learning and reducing overall aggression [106]. Schooling may also save energy by reducing drag [107].

So if we add food or predators to our model, it is likely we will see schooling behaviour emerge, as it is selectively advantageous. The evolution of the fish in this model is likely to include changes that would be selectively neutral towards a schooling fitness function (perhaps different interactions with their neighbours that look no more like a school 'should' behave) but that are nonetheless beneficial for evading predators or finding food. And, as the fish continue to get better at eating and avoiding being eaten, they will develop recognisable schooling behaviour as a side-effect of their gradual evolution. We are unlikely to see the "degenerate" [105] 'schooling' Zaera et al. saw, as swimming in circles is not going to aid survival or locate more food.

8.13 Flocking through predation

Convincing group behaviour has been evolved by several researchers using this approach. Reynolds [108] evolved animats to evade a simple predator. He placed his creatures in a simple 2D environment and equipped them with limited visual acuity (the ability to detect and distinguish each other, obstacles and the predator, but no more). The animats were tasked with steering a safe course through the environment by avoiding collisions and being eaten.

Though none of the evolved behaviours were "anywhere near as robust and general purpose as herding behaviours seen in natural animals" [108], his model did give rise to coordinated group behaviour in the prey. Despite the simple environment, simple animats and a particularly simple predator, the model (presumably) captures an important rationale for flocking – predation – well enough for group behaviour to prove beneficial.

Ward et al. [83] evolved fish that displayed schooling behaviour. Their model was more complex than Reynolds', with multiple predators and prey being co-evolved in a minimal 2D environment. The creatures used simple neural controllers that encoded models of two major fish senses, the close range lateral line (hearing or pressure) and a longer range vision system. Each of these senses was divided into several neurons that pertained to different sectors of the fish's field of perception. The predator and prey fish models were identical structurally, but evolved significantly different neural weightings over time to match their purpose.

As with Reynolds' model, group behaviour was evolved, the authors reporting that it was useful both for finding food clumps and for giving better protection from predation. (Though less marked, schooling emerged even in models with no predators present.) Ward et al. also say that the effect of predation on the model was less pronounced than they hoped, though the authors "believe that this is purely a reflection of the measure used rather than behavior."

Ward et al. comment that "it is known from Zaera, Cliff, and Bruten's ... work that the BOIDS rules do not describe all these properties, since the rules proved insufficient as a fitness function. … A slightly more sophisticated model is needed, and for this a flexible representation of the fish's knowledge (i.e., its sensory systems) must be designed." And this is true. The complexity of Reynolds' and Ward et al.'s models is almost certainly greater than Zaera et al.'s: there are different classes of agent who interact with each other dynamically; the agents themselves are much more sophisticated and strive to meet long term goals (survival and reproduction) through their short term actions; and Ward et al.'s predator and prey fish co-evolve with each other.

But, at a more abstract level, it's actually very simple: the objective is to eat and not get eaten. And crucially this is also the level at which their fitness functions operate, and at which the models evolve. Flocking is a difficult problem to solve directly, but if we allow ourselves to talk instead about prey, predators, food and the environment it actually becomes much easier.

We are now able to use evolutionary criteria that incrementally increase the survival chances of each individual (low level fitness) and of the group (high level fitness), measures that simply weren't available in Zaera et al.'s simpler model. And, after observing the simulation for a while, we find that the evolved fitter behaviour also happens to be behaviour that gives interesting group dynamics.

In contrast, Zaera et al. just decreed, "Let there be flocking." And there was no flocking, because there is no continuous development path from the low level, individual behaviour to the flocking behaviour we want. In common with all emergent systems, this is not something that can gradually

appear (§9.6). (Contrast this with swimming in circles, which is not emergent, can gradually appear and is a way of satisfying the fitness function found by their model.)

Ray and Ward et al. were able to sidestep this problem by evolving the high and low level models in parallel towards flocking behaviour. And their approach is also much more robust: group behaviour is an effective response to a significant number of predator models, so the system is not sensitive to any one aspect being exactly 'right'. Indeed Ray's and Ward et al.'s models differ substantially from each other in their environments, prey and predator models, method of evolution, and so on – much more significantly than tweaking parameters in a schooling fitness function – but both still produce convincing group behaviour.

9 EMERGENCE

After explaining how we use emergence the scope of our investigation into the subject, we briefly touch on the importance of mappings when creating emergent models before investigating how we can quantify emergence through information theory. We use quantitative emergence to define neutral emergence, in analogy to neutral evolution, and then explain how we can develop emergent systems automatically. Many of these themes are covered in greater depth in subsequent chapters.

9.1 Defining emergence

Before delving deeper into the subject, we should clarify what we mean when we discuss emergence. We are talking exclusively about what Ryan [100] calls weak emergence: there are no changes of scope, only changes of resolution. And while we allow the (small) possibility that genuinely new emergent properties may appear in some cases (as opposed to just epiphenomena), this is unnecessary for the subset of emergence we cover in this and subsequent chapters.

We are not trying to create and justify yet another definition of emergence, weak or otherwise. What we discuss here fits within existing definitions of the topic from authors such as Ryan. Rather we are interested in exploring aspects of emergence that are often not considered. We discuss the importance of mappings between the high and low level languages; we look at developing emergent systems in analogy to conventional software development; we consider how to quantify the quality of an emergent model; and we explore the power of neutrality in emergent models for speeding up development and boosting robustness.

9.2 A starting definition of emergence

Our starting point for emergence is (an adaptation of) Ronald et al.'s [109] definition.

- "*Design* – The system has been constructed by the designer, by describing local elementary interactions between components … in a language $L1$.
- "*Observation* – The observer is fully aware of the design, but describes global behaviours and properties of the running system, over a period of time, using a language $L2$.
- "*Surprise* – The language of design $L1$ and the language of observation $L2$ are distinct, and the causal link between the elementary interactions programmed in $L1$ and the behaviours observed in $L2$ is non-obvious to the observer – who therefore experiences surprise. In other words, there is a cognitive dissonance between the observer's mental image of the system's design stated in $L1$ and his contemporaneous observation of the system's behaviour stated in $L2$." [109]

Like other authors [85], we reject 'surprise' as a criterion for emergence, but we do start from Ronald et al.'s two languages of description. We user L for the microscopic low (or local) level and S for

the macroscopic high (or global) level of the model.[1] Later we shall discuss engineering emergent systems; in this case, *S* can be thought of as the system specification and *L* its low level implementation.

9.3 Subjective emergence

Ronald et al. [109] define emergence in terms of two languages – it is a relative concept. Their definition does not say that there is a "dissonance between the observer's metal image of the system stated in *L* and his … observation of the system's behaviour stated in *the language of emergence*, *S.*" *S* could be any language (though we see later in §10.11 that some languages are better than others).

We believe there is no such thing as a correct emergent property. The emergent properties of a system we see are not right in any absolute sense – they are merely a reflection of the view (more precisely, the mapping and language – §10.5, §10.20) we have taken. If we take a different view of a system, we may well see different emergent properties.

A relativistic approach to emergence is also advocated by Gell Mann [71], who suggests that low level information may be modelled through what he calls schemata

> *"A complex adaptive system acquires information about its environment and its own interaction with that environment, identifying regularities in that information, condensing those regularities into a kind of 'schema' or model, and acting in the real world on the basis of that schema" [71].*

The system uses schemata to separate "regularities from randomness" [71], to model information acquired in *S* in terms of *L.* Gell Mann suggests that emergence occurs when the flow of information to the model deviates from that expected by the schema; this is *emergence relative to a model* [72]. The model must adapt its schemata again to account for these unexpected inputs, after which the system is no longer considered emergent. While we don't subscribe to this rather transient definition, his model provides another thoughtful example of subjective emergence.

9.4 Useful emergence

Though no view of emergence is correct, some views will be considerably more useful (or good) than others.[2] We naturally see a group of birds as a flock because this captures the most salient (and, though our genetic inheritance, the most evolutionarily significant) properties of their behaviour: their macro position, their overall speed, their general direction. We largely abstract away other details, such as the individual birds' positions relative to each other or how they interact with each other. This gives us a useful view of birds (for hunting them, avoiding them or viewing them through binoculars) without overwhelming us with information.

1 H having been claimed by entropy.

2 We define utility as relative correctness – if it's right for this situation or model, it's useful.

We use these 'natural' views even when faced with artificial (and entirely abstract) systems. When we stare at a simulation of Conway's Life, we immediately pick out patterns such as gliders as we impose our real world propensity for detecting salient objects on this mathematical space. We find it much more difficult to detect meaningful objects or patterns in the soup immediately after starting Life from a random state. (Though patterns in apparently random states can still exist [155].)

Sometimes these views fail us. If a flock of birds is confronted by a large object (such as a pillar [82]), our model of the flock as a large, single entity will not help us predict the birds' behaviour. Will they veer to the left, to the right, or split in two? If they separate, will they regroup afterwards? It is not at all obvious from our 'big blob' flock model.

Birds and fish take advantage of another failure in this natural view to avoid predators; indeed, this is thought to be one of the reasons they flock or school in the first place. It is very difficult for a predator to pick out an individual fish within a school – the school is well defined, but information at the fish level is confusing and overwhelming [83]. Thus the fish, despite being weaker and slower than their predators, stand a reasonable chance of surviving an attack.

9.5 Independence and lossy emergence

The perception of an emergent system is that its high level behaviour appears to be, to some degree at least, distinct from, and independent of, the low level behaviour. A termite mound has ostensibly little in common with the scurrying insects that constructed it; a Life glider is an object with distinct boundaries and movement, quite different from its ever-changing but stationary constituent cells; optimal commodity pricing occurs through the local (and selfish) interactions of traders.[3] We see this apparent independence in many definitions of emergence, with their references to separate languages, levels or timeframes [85, 97, 98]; some definitions insist there is no emergence once we have identified a link between the high and low levels [100].

But while these emergent properties may appear to be something new and distinct, they are not. Emergent properties are merely a subset of the underlying behaviour, albeit a carefully chosen one. Emergence doesn't add anything; it removes (or hides) aspects of the underlying system until one is left with an apparently salient, coherent core of behaviour – the high level phenomenon, often better described in a different language – that appears to be emergent. Our high level view (constrained by language, time, etc.) has blurred the underlying system so only certain aspects of its behaviour are still apparent, much like Ryan's loss of resolution [100].

Consider that archetypal example of emergence discussed earlier, flocking. In Reynolds' boids model [81], each boid has a position and a velocity. The emergent flock has just a single velocity and position. We have moved from many interacting entities occupying a $6n$-dimensional space to just

3 Though we note that efficient market theory is disputed theoretically and empirically, with researchers turning to behavioural economics and or citing imperfect information and market manipulation.

one with 6 dimensions.[4] And although we use the same terms (position, velocity) to describe the behaviour of the flock and each bird, the relationship between the levels is non-obvious.

A flock of boids is certainly simpler and distinct from the behaviour of the individuals that make it, but our imprecise, natural language definition of flocking precludes us from showing that the emergent level is a subset of the low level behaviour in this case. Later we describe a technique for moving from a natural view for describing CA cells to a natural view for describing Life gliders only through eliminating degrees of freedom in the underlying system (§10), demonstrating that the emergent gliders are a simpler subset of the underlying behaviour.

No additions or novel discovery are needed to find the Life glider, only elimination. The glider exists equally at both levels, but only by eliminating unnecessary degrees of freedom does it become apparent, does it become the natural representation. And once we have eliminated degrees of freedom (and re-expressed the model in a new language that naturally captures this subset of the underlying system's properties), it is now non-obvious how the emergent level relates to the underlying behaviour.

9.6 Discontinuities and lossy emergence

Intuitively, it should be apparent that a discontinuity must exist between the high and low levels of an emergent system. This is why the emergent behaviour can be non-obvious, given a low level description. This is what causes the (rather discredited) "surprise" [109], the "cognitive dissonance" [109] noted by observers.

This discontinuity is also responsible for much of the power we see in these systems – we can use our emergent model to predict future behaviour. We can calculate the system's salient properties at some point in the future more efficiently than by running it. This model independence (or discontinuity) is the reason we can predict the flight path of a flock of birds (and, as we will see later §11, the future state of many CAs) easily, something that would be much more challenging for a group of birds behaving independently.

The emergent model is simpler than the underlying system, so information must be lost between the two. It follows that we can only predict certain properties of the system, though the properties of the model we can predict are – by definition – those in which we are most interested. (If we weren't interested in these properties, we should have sought an alternative emergent model.)

We may also not be able to anticipate the system's behaviour correctly in all cases from our emergent model. But, if the mapping and model we choose are good, our predictions should be accurate in the vast majority of circumstances, and in particular the most common circumstances.

4 This of course assumes that the characteristics identified by Reynolds are those which are pertinent to flocking.

9.7 Projection and sampling

It is not sufficient merely to lose information when moving to a model of an emergent phenomenon; the way it is lost is important.

Sampling is one way to lose information. We could, for example, only consider five birds from a flock of fifty and cut the amount of information in our system by (approximately) 90%. But this is not a useful reduction. It gives us no extra insight into the behaviour or emergent properties of the whole flock, nor does it lend us any useful predictive power over the birds' future behaviour.

Instead, we require a projection from the whole state space of the birds ($6n$ dimensions) onto the centre of mass of the flock (6 dimensions), a *uniform dimension loss*. Because of this projection, every point in state space loses the same degrees of freedom. This is consistent with the loss of resolution described by Ryan [100].

9.8 Mappings in emergence

Emergent systems are usually described as a pair – an emergent phenomenon and a low level behaviour – distinguished by different languages, levels, timeframes, etc. Ronald et al.'s definition uses two languages [109]; Ryan has two scopes and resolutions [100]; many authors use two levels; a flock model has rules for each animal and for the group; Life has simple rules for cells and more loosely defined behaviour for gliders and other objects.

Relatively little attention has been given to the mapping between this pair – it is implicit, assumed to exist somewhere in the background. But we believe that the mapping is a crucial component in any emergent system. Without a good mapping it would be impossible to use an emergent model (even an otherwise valuable one) to predict system behaviour. We will see later (§10.20) that finding a good mapping is as crucial as finding a good high level model for describing an emergent system effectively.

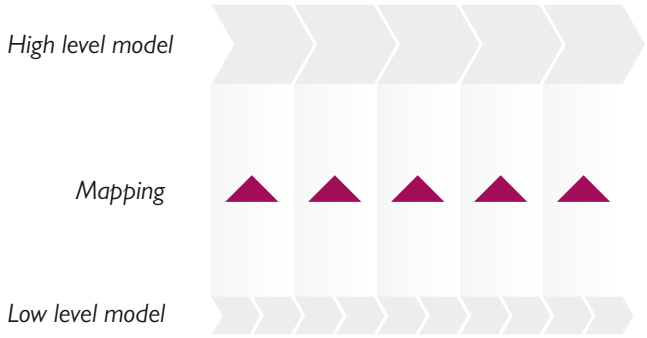


Figure 9.1 Mapping between high and low level models. The mapping translates between the two models.

And it is not always natural or obvious which mapping is the best choice, particularly when dealing with systems that, unlike flocking birds, aren't 'natural' to us. This is something we discuss later in

§10.11, when we see cases where selecting a different mapping leads to a different emergent model. Each emergent model is useful, but they reflect a different aspect of the low level behaviour.

9.9 NEUTRAL EMERGENCE

We now discuss neutral evolution and draw a parallel between it and emergence before introducing neutral emergence. We then show how we can exploit some of the strengths inherent to neutral evolution – strengths such as robustness and adaptability – when developing emergent models.

9.10 Neutral evolution

Neutral evolution is a well recognised phenomenon in evolutionary biology [15, 110]. The theory states that changes in the genotype may occur without materially affecting the fitness of the phenotype: the change is selectively neutral. This is possible because the mapping between genotype and phenotype is complex, and that there is significant redundancy in both. Different genotypes can map to the same phenotype; for example, different codons (DNA nucleotide triplets) can code for the same amino acid. Thus changes can occur in the genome that have no discernible impact on an organism's ability to survive. Similarly, the same genotype can give rise to different phenotypes due to variations in environmental conditions during development.

It is thought that this redundancy between layers allows the phenotypic population to explore more of its fitness landscape at little cost, drifting along contours of equal fitness to regions with high peaks of fitness not directly accessible from its starting point in the search.

The indirection in neutral evolution may allow organisms to evolve new behaviours by changing gene regulation (creating new paths) and not by changing the genes themselves. Thus each gene can remain a specialist, preventing compromise of its function [17]. Another suggested advantage of indirection is enhanced evolvability, which in turn promotes biodiversity [17]. It may also be that neutral evolution is almost inevitable, simply because of the complexity of the environment in which life exists.

We shall soon introduce neutral emergence, which seeks to apply the same concepts – of indirection, of evolvability – to emergence. The next few sections explore how this could work, first by looking at mutual information and evolution in detail.

9.11 Quantitative emergence

In a bid to understand the complexity of life, Adami [78, 111, 112] considers how prebiotic life could adapt to its environment. Specifically, he looks at the transfer of information from the environment to the genome, represented by a simple bit string. The strings exist within an environment with physical laws that affect how they can replicate. Adami assumes that the environment is replete with potential information and that there are many things to discover. In information theoretic evolution, the genome is seen as some kind of representation of the environment. Adami and Cerf

[112] dub the mutual information between the string and its environment the *physical complexity* (a relative measure, as opposed to the algorithmic complexity; see §9.13).

Note that, as they can reproduce, the strings must already contain a substantial amount of information. The information that allows a string to replicate is context dependent: a powerful arrangement of bits in one environment is likely to be meaningless in another if they don't allow it to harness resources in the environment in order to reproduce.

In Adami's model, the strings adapt through random mutations (caused by cosmic rays[5] or copy errors). Some of these bit flips will be beneficial, some harmful and others essentially neutral.

- Mutations of bits crucial to replication are likely to be lethal, so these areas will tend to stay fixed in the population – these are *cold bits*.
- Neutral mutations to unused or little used segments of the genome are more likely to remain, leading to diversification – these are *hot bits*.
- When a positive mutation takes place, the string has become better adapted to its environment and the allele will spread throughout the population, changing the bit from hot to cold. *Additional information about the environment has been written into the genome.*

9.12 Complexity and information

Adami models the adaptation of the strings to their environment through information theory. Information entering the genome can be seen as a measurement performed on the environment by the genome. According to information theory, this act increases the correlation between the measured system and the measurement device, and the conditional entropy of both decreases.[6]

The more cold bits there are, the higher the mutual information between the organism's genome and the environment. The mutual information, or correlation, between the string S and its environment L, $\mathrm{I}(S : L)$, is the entropy of the string, $\mathrm{H}(S)$, minus the conditional entropy of the string in the context of the environment, $\mathrm{H}(S|L)$

$$\mathrm{I}(S : L) = \mathrm{H}(S) - \mathrm{H}(S|L)$$

Mutual information = string entropy – conditional string entropy

Or equivalently

$$\mathrm{I}(S : L) = \mathrm{H}(L) - \mathrm{H}(L|S)$$

Mutual information = environment entropy (constant) – conditional environment entropy

5 Random bit mutations unrelated to replication.

6 Since this is a closed system, the unconditional (overall) entropy does not decrease during measurement. Measurements are performed spontaneously and randomly by the genomes, but the information acquired is not released just as capriciously; rather, it is used to lower the entropy of the population. Adami points out that this is the prototype behaviour of a Maxwell demon [78].

The conditional entropy $H(S|L)$ can be thought of as the amount of information in the string that cannot be explained by (correlations with) its environment. Similarly, the conditional entropy $H(L|S)$ is the amount of information in the environment that cannot be explained by correlations with the string.

9.13 Kolmogorov complexity

As entropy (and hence information) is a statistical concept, it is not possible to determine the entropy of one string. Yet some strings are clearly more regular than others. To quantify this degree of regularity, Adami uses *Kolmogorov complexity* [113], which defines how easily a string can be obtained through computation. Difficulty is measured by the length of the shortest program that can compute the string on a universal Turing machine. (The prefix code needed to simulate any other machine is vanishingly small in the limit of an infinitely long string s.) The Kolmogorov complexity (algorithmic complexity) is thus defined as

$$K(s) = \min\{|p| : s = C_T(p)\}$$

$K(s)$ = Kolmogorov complexity of string s, $|p|$ = length of program p, $C_T(p)$ = result of running program p on Turing machine T

Within this context, a string is defined to be random if the shortest program to compute it is as long as the string itself, $K(s) \approx |s|$.

However, Adami comments that the "algorithmic complexity does not seem to be a good measure of the physical complexity of a string" [14], that the regularity of a string does not necessarily reveal the complexity of the encoded information – as Adami points out, "it is possible to create an (admittedly insane) encoding scheme in which the blank tape represents all of The Brothers Karamazov" [14]. This problem can be rectified by adding a context to the Turing machine, supplying it with a 'universe' tape l. The machine then computes results from l, which allows the conditional Kolmogorov complexity to be used

$$K(s|l) = \min\{|p| : s = C_T(p, l)\}$$

$K(s|l)$ = Conditional Kolmogorov complexity of string s given string l, $|p|$ = length of program p, $C_T(p, l)$ = result of running program p on Turing machine T with input tape l

This measures the remaining randomness in s, the bits in s that are not correlated with (and that cannot be calculated from) the bits in l. The program p comprises instructions for those parts of s that can be calculated from l together with a compressed copy of the remainder of s. In the limit of infinitely long strings, p will mainly consist of the randomness of s, which allows us to state the mutual complexity

$$K(s : l) = K(s) - K(s|l)$$

Mutual complexity = environmental complexity (constant) – conditional complexity (or remaining environmental randomness)

Obtaining s from u constitutes a measurement on u, which reduces the conditional entropy of s and increases the information known about l. Adami provides an example of this process. Imagine a string s that shares some information with l, with other bits that are random. Mutations constantly change these bits and a machine checks to see if any of these bits can now be obtained from l by a computation; if they can, these constitute information, augmenting their mutual entropy. Thus the mutual information between them constantly increases (Figure 9.2) and "natural selection can be viewed as a filter … that lets information flow into the genome, but prevents it from flowing out" [111]. Returning to Adami's original model, these are simply the hot bits becoming cold.

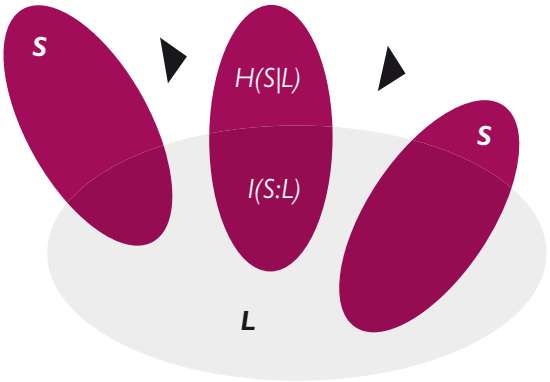


Figure 9.2 Gaining mutual information between the genome and the environment (adapted from [78]). The genome becomes better adapted over time.

9.14 Minimum model complexity

The conditional Kolmogorov complexity of an uncompressible (defined as *random*) environment is the same as its Kolmogorov complexity: its shortest description is as big as the universe itself. In terms of S and L, if L is uncompressible then S must be (at a minimum) as complex if it is to be described with no ambiguity.

This has implications for modelling problems for which the outcome cannot be predicted more quickly than by running the simulation. For example, modelling the emergent behaviour of certain classes of cellular automata will require a language that is at least as complex as the low level language.[7] It also has implications for modelling problems which (whether Turing complete or not) we cannot hope to model in their entirety. Chaotic weather systems is one example mentioned earlier, but the same will be true (often for boring practical reasons) of any system interacting with the real world.

7 For a CA, the low level language would consist of more than just the update rules, as other factors, including the initial configuration and spatial arrangement of cells, can have a significant impact on the simulation's progress.

So there will be a discontinuity between the complex environment and comparatively simple model held by systems that interact with it. (We make a similar argument in the software engineering domain §9.19.)

9.15 Information retention

Adami makes the point that, while there is selective pressure for hot bits to become cold, it is relatively rare for cold bits to revert to being hot. Information can enter the genome but not leave, and therefore "genomes are doomed to accumulate more and more information and grow longer and longer as a consequence" [78]. This has potentially serious consequences for an automatic emergent system, as the language will tend to grow and become unwieldy over time. Particularly with this sort of application, it is important that the genome (the language) is kept as clean as possible.

The error catastrophe [17, 44] should provide a partial solution here by preventing genomes over a certain length from improving their fitness. As stated previously (§6.1.6), for a fixed mutation rate, the number of poor alleles accumulates with the square of the number of loci. It is therefore possible that adjusting the mutation rate could afford crude control over the language size.

9.16 Comment on Adami's model

Adami's description of information acquisition appears somewhat simplistic, in that it assumes there are no correlations between the bits in the string S. Most tasks (undoubtedly including reproduction in protolife) are complex combinatorial problems, so when some bits go cold it will be necessary for other bits to become hot again. In other words, there is no guarantee that mutual information will increase monotonically [111]. This can affect evolutionary trajectories, as it may not be possible 'to get there from here' by the moves permitted by evolutionary operators [17].

It has been shown [17] that a population found at a certain level of fitness will tend to form a narrow cloud band clinging to side of the particular fitness peak it is climbing. There is significant neutral evolution within this band, so strings with very similar fitness may have substantially different genomes. This will lead to a disparity between the population and Kolmogorov entropies, as a number of bits will appear hot between individuals but not between each string and the environment. The most common genome in a species (the quasi-species [17, 44]) often forms a minority of the population.

The total information is shown as unchanging (the areas of ellipses S_n and L are constant) in Figure 9.2, implying that conditional information decreases as the mutual information increases. Again, this is not necessarily the case: a neutral evolutionary step could change the amount of conditional information $H(S|L)$ by increasing the size of the uncorrelated part of the genome [111]. This could happen, for example, by mutating a redundant part of the correlated genome such as could have resulted from a gene duplication event. Piszcz and Soule [125] discuss genome growth as a strategy

for increasing robustness (to perturbations from the genetic operators; see also §9.20). A similar argument means that parts of the environment L that are not correlated with S can also change neutrally (from the point of view of S): it can change without the organism 'noticing'. So S is robust to these kinds of environmental changes, too.

Gell-Mann's definition (§9.3) relies on a system "identifying regularities" [71] in its environment – there is an implication that the environment is not random. In contrast, Adami allows for a completely random environment (the entire universe can be encoded in s). In practice the two are equivalent, as a random fitness landscape offers no clues as to whether a particular mutation is beneficial: there can be no gradient leading towards better solutions that the algorithm can exploit to reach a fitness peak. So unless a string (Adami) or schema (Gell-Mann) stumbles upon a complete correlation purely by chance, the population cannot adapt.

While these limitations would need to be accounted for when applying his model, particularly in a biologically realistic setting, the simplifying assumptions made by Adami do not undermine the principle of his information-theoretic approach, particularly when it is transferred from a biological to an engineering domain.

9.17 Quantifying emergence

We can use Adami's information theoretic model almost directly to quantify the degree of emergence in a system: starting with the definition of emergence developed from §9.3, simply replace the environment by a low level language L and the genome by a high level language S and we have an *automatically developed emergent system*. Modelling (creating a high level model of the low level) and incremental system development (creating a low level model of the high level) can both be viewed as increasing the shared information between the specification and implementation. The mutual information between S and L can be expressed as

$$\mathrm{I}(S : L) = \mathrm{H}(S) - \mathrm{H}(S|L) \equiv \mathrm{H}(L) - \mathrm{H}(L|S)$$

The mutual information between S and L is the common behaviour that both languages can express, and for which S can model L. This is equivalent to how much of L is expected by, and can be predicted in, S.

The conditional entropy $\mathrm{H}(S|L)$ is the information in the system specification that has not been captured by (correlations with) its implementation. These are the 'surprising' behaviours or properties of S not explained by L. (They may also just be noise.) If we were trying to implement a specified system S in an implementation substrate L, this conditional information is the part of the specification that has yet to be captured by the proposed implementation L: more development work is required.

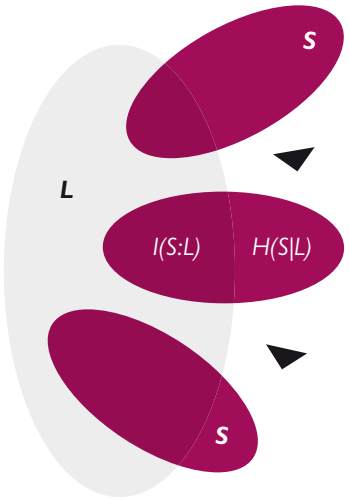


Figure 9.3 Gaining mutual information between the low level and high level (emergent) models. The emergent model captures more of the low level's behaviour over time.

Similarly, H($L|S$) are the 'surprising' properties of L, properties that may be considered L unnecessary for the realisation of S and invisible when the system is viewed through high level observation 'glasses' [114] that see only system-level properties (multiple distinct microstates nevertheless resulting in the same observed macrostate). (Note that these system-level properties are not in any sense objective or universal – they are merely those properties we have chosen to retain in our specification.)

9.18 Neutral emergence

We have already said that there is a discontinuity in the relationship between an emergent system and its environment. It follows that this relationship must be incomplete and that there cannot be a one-to-one mapping between the two, so a number of environmental behaviours will be indistinguishable to the system (and vice versa). But this is almost exactly the definition of neutral evolution given before (§9.10).

We call this *neutral emergence*. An emergent property exhibits neutral emergence when a change in the microstate L does not change the macrostate S, or vice versa. In information theoretic terms, it is a change that does not significantly alter the mutual information $I(S : L)$. It follows that neutral emergence is almost inevitable in any high level model of any sufficiently (realistically) complex implementation or other low level behaviour.

9.19 ENGINEERING EMERGENCE

The next sections discuss how neutral emergence could be used in software development instead of traditional software engineering processes. We demonstrate how to apply common and well understood evolutionary and other search techniques to neutral emergence, and discuss how to use them to find an implementation of a specification automatically.

Engineering processes traditionally focus on designing and building products in a largely top down manner. To build a bike we need wheels, a frame and a seat. We can break down these parts further into nuts, screws and tubing, until the schema reaches an atomic level (for a bike) and it is possible to construct a complete machine. But the bike also exhibits emergent properties: it can support and convey a person, but a wheel or seat alone cannot. This is an example of planned emergence (and something that is unlikely to feature in the bike's construction manual, though clearly was considered when designing each component); it may also exhibit unplanned emergent properties, such as the front wheel detaching unexpectedly [115].

Engineering processes and tools (CAD drawings, UML diagrams, etc.) [116] often have no adequate way of representing emergent properties – they focus on (and constrain themselves to) exploding one particular aspect of the system which, almost by definition, makes it virtually impossible to model these 'unexpected' aspects of the system.

Emergent software engineering attempts to manage novel behaviour by including an unexpected event observation and property modification phase into the development cycle. In the language of Gell-Mann (§9.3), the schemata are adapted to include new regularities discovered through interaction with the environment.

And by internalising this observation mechanism, we immediately get automatic, incremental emergent engineering as systems adapt their own schemata dynamically. The information theoretic definition of emergence given in §9.17 suggests how this could be done: use the mutual information as a fitness function to search for good low level implementations of a system specification. Equivalently, we can search for good models (system descriptions) of an existing implementation. (Note that, in practice, an equivalent but more efficient, problem specific fitness measure may be more appropriate.)

This is, of course, essentially what many evolutionary algorithms do. For example, genetic algorithms (GAs) change their constitution through crossover and mutation, and the algorithm's progress is observed and directed by an internal fitness function.

And this approach to emergent engineering shares another nice property with evolutionary algorithms: we only need to know if the end product is correct (that the high and low level systems behave consistently), not how to find it (exactly how the systems are related). This neatly sidesteps the discontinuity problem outlined in §9.6, as there is now no need for an obvious development

path between the specification and implementation. Instead we can search at the implementation level for good solutions.

Developing emergent systems is generally thought of as difficult, something involving a great deal of intuition and guesswork at best and black magic at worst. There is no methodical process for getting the emergent model or implementation we desire. But we have just seen how standard evolutionary and other search techniques can be used to develop emergent systems.

There is nothing magic about emergence, or developing emergent systems. The mapping to known search techniques explored here is nascent, but, as with Langton's link between physical systems and computation [18], offers the potential to exploit a large body of existing work. We may be able to apply the ideas introduced here for emergent systems to 'traditional' developments to add desirable properties such as robustness. We discuss this now.

9.20 Robustness in neutral emergence

The best solutions not only solve a problem well, they also solve it robustly. A solution is robust if minor changes do not materially affect the fitness of the solution, perhaps because the solution is insensitive to certain failure modes. It is important to search for robustness when using an artificial fitness function in an evolutionary algorithm. Branke [117] notes that "this means that not only the solution should be good, but also that the (phenotypic) neighbourhood of the solution should have a high average quality. Looking at the fitness landscape, a solution on a high plateau should be preferred over a solution on a thin peak: if the environment changes slightly ... the solution on the plateau will yield much better expected quality than the solution on the peak."

Robustness is intimately related to neutral evolution. The aspects of the search landscape exploited by neutral evolution – correlations on different scales, basins, and so on – are exactly those that can make a model robust [17, 125]. And robustness is intimately related to neutral emergence too.

An emergent system is robust if changes to either the high (emergent) or low levels do not significantly affect their mutual information. Assuming we have a good model, the conditional information $H(L|S)$ (the information in the implementation not correlated with the system specification) is behaviour in L unnecessary for the realisation of S. These additional properties will be invisible if the system is subject only to high-level observation through S – there will be basins of multiple low level behaviours 'draining' into the same high level behaviour, with multiple distinct microstates nevertheless resulting in the same observed high level macrostate.[8]

8 The same would be true if a low level representation is being sought for a high level specification, though presumably to a lesser extent, as the high level language is likely to be more abstract and less rich.

9.21 The power of neutrality

Travisano et al. [126] (also [127] and others) evolved twelve initially identical populations of *E. coli* in identical glucose-limited environments. After 2000 generations, the fitness of the bacteria in all populations had improved by about 35%, and they were all very similar in other ways, with higher maximal growth rates, larger cells and fewer cells at *stationary phase*[9] than their common ancestor. Thus "identical starting conditions and identical environmental conditions produce an almost identical evolutionary outcome" [110]. The authors then introduced the twelve populations to novel environments, substituting other sugars (such as lactose) for glucose. The fitness difference between populations in these new environments was found to be 100 times greater than the variation seen in the glucose environment. Travisano et al. conclude that, although the populations had acquired very similar traits at a phenotypic level, they differed substantially at a genotypic level: they had evolved to substantially different places within the fit glucose phenotype's basin.

A similar process could be used when engineering emergent systems. Stressing a model before it is 'released into the wild' (exposing it to the equivalent of glucose-limited and lactose-rich environments by adding a range of implementation errors or simply by adding noise to the fitness function) should encourage the solution towards the centre of the desired behaviour's basin. It should also help us land on a gently sloping plain or high plateau rather than a narrow peak or steep cliff. (See also §9.24.)

Such an implementation would be much more likely to perform to specification: slight deviations at a low level will still fall within the same basin and produce the same desired outcome. Perhaps more significantly, the implementation would be robust with respect to mutations (errors of implementation, manufacture, environment, etc.): the implementation degrades gracefully. This is a far stronger statement than can be made about formally proven systems, which do not guarantee any level of performance with even the smallest change. Large unmapped entropy $H(L|S)$ at the low level is necessary for emergent systems to be robust in this manner.

As with neutral evolution, neutral emergence should allow a system to explore its environment more widely without 'committing' to a particular model and without compromising the effectiveness of existing adaptations. The model can be adapted gracefully to introduce new behaviours.[10] *In short, it is not brittle.*

9 The rate of cell division equals the rate of cell death, so the total number of cells remains constant. Alternatively, there could be no cell birth or death. See [128] for more details.

10 It is also likely that the landscape would prove easier to explore, as the high level language would tend to be a simpler model, without so many local optima and smoother with greater correlation distances. Kauffman [17] discusses the benefits (even necessity) of starting with a simple terrain before moving to a more complex one as the system adapts (using neural networks as a vehicle), something also developed by others including [118].

9.22 Emergence is easy

Traditional software development tries very hard to eradicate harmful emergence from the implementation, often with limited success. In fact, creating an emergent system appears to be comparatively simple. And it is: emergence is easy. But developing a useful emergent system – one that models the properties we want – is considerably more difficult. It is all too easy to end up with the wrong emergent properties that reflect little of the desired behaviour, or little interesting behaviour of any kind.

9.23 Exploiting problem structure

Searching for answers in complicated landscapes is an extremely common problem in computer science, and one that has been tackled in many different ways, using everything from depth first search to evolutionary algorithms. Some techniques, for instance Dijkstra's shortest path algorithm [119], are very problem specific, whereas others, such as genetic algorithms, can usefully be applied in many different situations.

Though not universally true, it is generally the case that problem specific techniques will outperform more general ones in their areas of 'expertise' [120]. This may be inevitable (and fortunate for the individual techniques), but it is the reason why they perform better that is significant here: they are better adapted to the problem, or more specifically they are better at exploiting the structure of the problem.

It is perhaps a truism that search algorithms work better if they exploit the problem structure. Yet, with few exceptions, search and optimisation algorithms only consider complete solutions when looking for answers. There is a huge amount of information out there that they simply ignore, that they jump over in their quest for a final solution. If this information – about partial solutions, about search trajectories, about the problem landscape – were exploited, it would be possible to find better solutions, and find them more quickly.

We have seen that emergent systems can be developed in much the same way – and with much the same techniques – as 'conventional' systems. But while the methods may be conventional, the aim is to channel development in new directions and create a much more fault tolerant model, a model robust to flaws in its implementation, in its environment, and to our understanding of the problem at hand.

Finding a good emergent model of a system complex enough to be useful is difficult. But things often become a lot easier if one looks only for 'partial' descriptions. In particular, it we find it is possible to build on these partial descriptions and arrive at a fuller, more accurate model.

9.24 'Good enough' solutions

In trivial cases, finding perfect solutions is both practicable and verifiable, but in general this is not so. We suggest that we shouldn't even bother trying; instead, we should devote time to finding an answer that is robust.

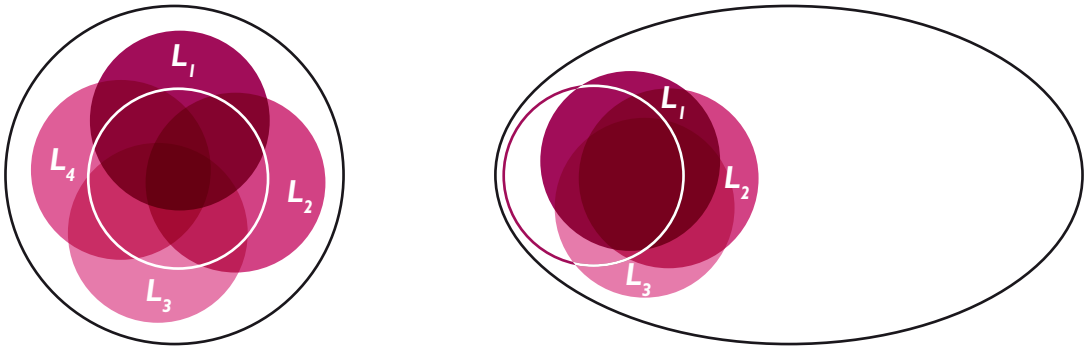


Figure 9.4 In the left diagram, the small central circle is the perfect solution and the large outer circle the space of acceptable solutions. Four 'good enough' solutions (L_1 to L_4) are also shown. They overlap significantly with the perfect solution, but will be easier to find in all but the most trivial cases. The right diagram shows the perfect solution right on the edge of the space of acceptable solutions. In addition to being easier to find, the good enough solutions (particularly L_2) are more robust too: a small error is less likely to push L_2 out of the acceptable solutions circle than if the same problem occurred in the perfect solution. We assume that the model has been stressed during development so that the most likely perturbations have been experienced.

There are two main reasons why 'good enough' solutions are actually better: they can be made more robust; and they are much easier to find. We don't want a complete overlap between the specification and implementation in Figure 9.4, we want partial answers. If we have a 'good enough' answer that is not fully defined, then the final solution can be picked from a range of essentially neutral (or if not neutral then certainly sufficiently good) alternatives. So we can perturb the specification in a number of ways (fitness weights, initial conditions, etc.) to determine the shape of the local landscape and then position the solution in the centre of this neutral area of the landscape, making it much more robust to change.

The solution will be mostly correct, most of the time, and will remain mostly correct, most of the time even if it turns out that the problem specification, implementation, deployment environment or any other factor doesn't turn out quite as was expected. (And, after all, how often does that happen in practice?)

A nice example is work carried out by Piszcz and Soule [125], which shows that evolution can favour robustness, even if the robust solutions are less fit than alternative, more brittle ones. Seeking to solve a symbolic regression problem through genetic programming (GP) [6], the authors seeded their algorithm with individuals that occupied a high but narrow peak on the fitness landscape. After running the algorithm, they found that the initial population had been replaced by solutions

that were less fit, but were more robust and resilient to the often harmful effects of GP evolution [3, 12, 13]. The final population occupied a lower, rounder fitness peak – the survival of the flattest.

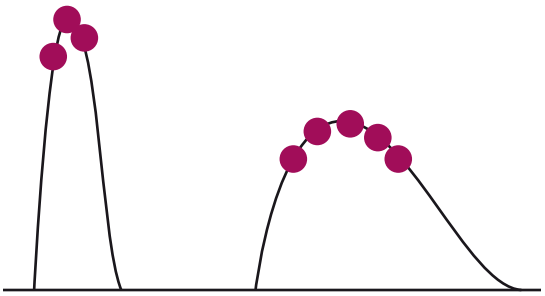


Figure 9.5 The narrow left peak contains a few highly fit individuals. The flatter, lower peak is occupied by more individuals who are more resilient to genotypic changes (redrawn from [125]).

It could be argued that these results stem from the particularly negative effects of GP's genetic operators, but we reasoned in §3.2 that there is an inherent tension in all evolutionary algorithms that attempt to balance innovation and conservation, and this outcome – individuals increasing their own chance of survival (becoming fitter) by building up resistance to the algorithm's damaging changes, even if this reduces their apparent fitness (as determined through the fitness function alone) – is not uncommon or unexpected.[11]

9.25 Key points

- There is nothing magical about emergence, or developing emergent systems. Emergence is easy.
- Emergent systems can be generated automatically. Standard evolutionary and other search techniques can be used.
- The language of dynamical systems (of basins, etc.) can usefully be applied to 'conventional' GA searches as well.
- There are positive advantages to having emergent systems. If the system is evolved such that the implementation is centred in the specification's basin of possibilities, it is likely to be more robust to perturbations than conventionally developed system.
- Exploiting the problem structure to find good enough partial solutions that are mostly right, most of the time may give better results than searching for the perfect answer: the solution can be made more robust, and the solution will be easier to find.
- If some properties are specified, but others left unspecified, we will inevitably develop an emergent system.

11 For this tendency not to exist, the overall impact of the innovation operator would have to be beneficial for each individual's fitness overall: the progeny of each individual must be, on average, fitter than their parent(s). This is manifestly not the case with most common evolutionary algorithms (see e.g. §6.1, [3]).

USING CELLULAR AUTOMATA TO INVESTIGATE NEUTRAL EMERGENCE

The previous few chapters have discussed evolutionary algorithms, computation as a dynamical system, entropy and how we model emergence, introduced neutral emergence and explored why the ideas behind neutral emergence – that of being good enough, enough of the time – may actually give more robust solutions more quickly. But so far we have provided no concrete examples of the concept.

As we are trying to understand the fundamental properties of neutral emergence, we chose to investigate it through cellular automata (CAs – §2). CAs are in many ways an ideal substrate for exploring neutral emergence: they are simple, discrete mathematical models with none of uncertainty we could see if using real world systems or even more complex (and potentially unreliable) computer simulations. At a micro level it is very easy to understand and visualise their progress, but at a macro level CAs can exhibit very involved (complex, chaotic and even Turing complete) behaviour. And, as we see in §10.5, CAs are also capable of showing emergence through *coarse graining*.

In §9.3, we discussed emergence as a relative concept: while some models may be more useful to us than others, there is no 'correct' emergent model, and the emergent properties we see depend on our point of view. We find in §10.12 that CAs show this relativism, and see how a single CA can be coarse grained to several emergent models, each of which captures a different facet of the underlying CA's behaviour. Through coarse graining we also show emergence is lossy, extracting a carefully chosen subset of a CA's low level behaviour to produce a coherent high level model in the form of another CA rule. In §11, we develop a method of quantifying the 'goodness' of a CA coarse graining (the quality of the emergent model) and show how we can use this to find emergent models automatically.

CAs help us demonstrate why thinking of emergence as neutral emergence has real advantages when we introduce *partial coarse graining* in §10.16. A partial coarse graining (as opposed to a total coarse graining) has an incomplete mapping between the low level and emergent models and can

therefore make mistakes, but also have the freedom to capture low level behaviour that is impossible to model otherwise. Sometimes these emergent models capture significantly more of the underlying CA's behaviour than any total coarse graining. Partial coarse graining can also be used to find emergent solutions in cases where an exponential explosion makes it expensive or (practically) impossible to find an answer otherwise.

While the CA models used here are nearly all too simple to show the robustness benefits discussed in §9.20 – there aren't enough possible solutions for there to be significant solution clustering – we do see through partial coarse graining that 'good enough' solutions can be found more easily in some cases. Interestingly, the extra freedom we get by not restricting ourselves to totally correct solutions also allows us to find more comprehensive or better targeted emergent models than we could otherwise (§11.7).

In many cases we are not merely looking to capture emergent behaviour; rather we wish to describe particular aspects of the underlying behaviour through our emergent model. Specifically, we want to model the behaviour of interest to us. In §12, we build towards a method of targeting specific emergent behaviour in CAs by through feature extraction, phase changes and exceptions. This allows us move beyond developing models of emergent systems automatically and to developing models of the emergent systems we want automatically.

10 COARSE GRAINING CAS

Cellular automata (CAs) are discrete, conceptually simple and well studied [18, 36] systems. They are also capable of complex behaviour and exhibiting emergent phenomena. This mixture of simplicity and complexity (in the broadest sense of the word) makes CAs a useful medium for exploring and understanding neutral emergence. This section examines emergence in CAs and how we can use these emergent phenomena to predict the behaviour of the underlying system.

10.1 Predicting cellular automaton behaviour

Predicting the behaviour of cellular automata is impossible in general [36, 69, 70]. Similarly, Polack and Stepney [121] suggest that moving from a high level specification to emergent implementation via formal refinement is, at the very least, non-trivial. But adopting the less stringent criteria of neutral emergence – getting it mostly right, most of the time – opens up interesting new possibilities. Under certain circumstances, it makes predicting CA behaviour a tractable problem. In fact predicting CA behaviour, at least in the case of Wolfram's elementary rules (including many in Classes 3 and 4 – §2.7), is not only possible but works very well.

One emergent structure found in CAs is the spaceship. Next we use the most famous CA with spaceships, Conway's Life [36, 66], to see how one particular spaceship, the glider, can be identified and how we can predict its emergence or collision. Although the method described is naïve and somewhat limited in scope, it does introduce some important concepts we shall use later.

10.2 How to find Life: identifying patterns and predicting behaviour

CA cells are small, stationary and rather dull, but the patterns they make can be large, dynamic and interesting. CA patterns are interesting because they are what we see; they are emergent behaviour of the CA: the spaceships and oscillators in Life, the fractal structure in elementary CA rule 18.[1] The path to take when translating from one to the other may seem non-obvious – surely a whole new set of incompatible concepts must be introduced? – but it is actually just a matter of throwing information away. Specifically, it is possible to convert a collection of static cells into dynamic objects through these three steps[2]

- *Object detection* – collapse the objects' cells into one state
- *Object tracking* – collapse the objects' movement over time
- *Object prediction* – collapse the objects' states into fewer states

1 References to 'rule *n*' should be taken as shorthand for 'elementary CA rule *n*'.

2 In this section (until §10.5), 'collapse' is defined to mean modelling two or more states or possibilities with one state or possibility, while 'compress' is defined to mean a more efficient model of the states or possibilities. Compression is achieved through collapsing states.

Collapsing an object's cells and movement allows it to be detected and tracked over time – these steps add meaning to the representation. They do not, however, allow the behaviour to be modelled more economically. Collapsing the object's states allows prediction: the future state of the CA can be calculated more efficiently than by simply running it, though we shall see that the approach described here is incapable of predicting the behaviour of the CA beyond a few specific cases.

Though flawed, the concepts this method embodies – that of identifying and modelling the emergent behaviour we want, and doing so automatically – are potentially powerful when generalised from this limited example, and we develop just such a general method over the next chapters.

10.2.1 Collapse the objects' cells – detecting objects

Objects must first be identified. In Life this can be done fairly trivially by looking for contiguous blocks of cells. Most objects have a defined border of dead cells separating them from other objects (though there are exceptions, such as the toad in Figure 10.1). Other CAs will have analogous distinguishing features that define object boundaries. (They must; if they did not, the CA could not contain any objects or emergent structures.)

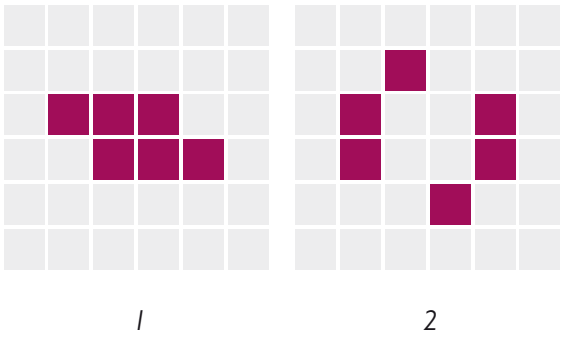


Figure 10.1 The stages of the toad, a period two oscillator in the Game of Life. (The diagram numbers show the current timestep, after which the oscillator repeats.)

Gliders can be identified by examining each 5 × 5 block (the size of a glider plus a surrounding barrier) at increments of 1 block. If it contains a glider (and no more) then it is labelled as a glider. Otherwise, it is not.[3] There is now a link between the low level cells and high level gliders, similar to the *tagged sites* used by Turner and Stepney [122].

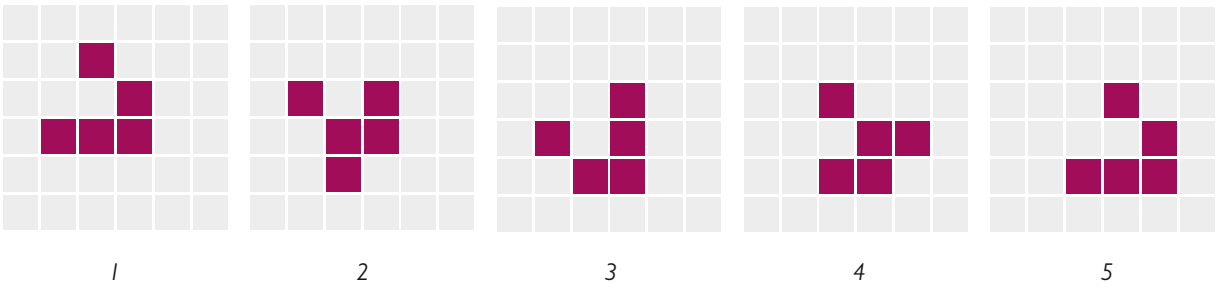


Figure 10.2 A glider gliding in the Game of Life.

3 This is patently hideously inefficient, and there are obvious immediate refinements that could be made, but for the purpose of this example it will suffice.

10.2.2 Collapse the objects' movement – tracking objects and identifying patterns

Objects can be tracked over time because of the speed of information flow in Life: at each step, an object can move at most one cell in any direction. We can search immediately around any glider we found in the previous step and identify it as the same object. The glider's position has become relative, no longer defined by its grid location.

10.2.3 Collapse the objects' states – object prediction

If an object repeats a sequence of moves, it can be classified as a period n pattern. And as the pattern must, of course, repeat again unless it hits something, we can use this to predict an object's behaviour and path (within a limited context at least).

Predicting the appearance of an object is more difficult. We can extend the object identification step to include other interesting cell configurations of arbitrary (though small) size. Some structures, such as *r*-pentomino and glider guns, are known to produce gliders. Others, including blinkers and an empty screen, are known not to. We could construct similar groupings for other objects, but there is a vast (infinite) number of patterns for which glider production is unknown.

Luckily most of these unknown patterns can be ignored. All CAs have a fixed speed of light (§2.3) that limits the spread of information through the structure, so there is a limit as to how quickly perturbations can cause (or prevent) the production of a glider at a particular location. It is perhaps reasonable to assume we could construct an enumeration of all CA structures of, perhaps, 10×10 squares and time when (or if) they produce a glider in the next five turns. Monitoring the block to see whether it remains unchanged – or equivalently changes to another with the same behaviour – will allow the nature and time of any behaviour change to be foreseen.

But there is clearly still a prediction horizon here, and extending it (by increasing the block size enumerated) will in general grow the enumeration exponentially. However, the dynamics of complex CAs such as Life suggest that most future states can be discounted. Unlike (discrete) chaotic and ordered systems, complex systems are dissipative [18]: they maintain order by constraining themselves to a small corner of their state space. This means that, as the simulation progresses, a lot of information is lost (the CA is assumed to be a closed system) [62]. This implies that a lot of information currently in the model is not going to be important in the future, that a lot of possible current states will end up in the same future state. There must be substantial equivalence classes in the current model with respect to some future model, and the size of these equivalence classes will increase (in general) the further one looks into the future. Though potentially beneficial, it is not immediately apparent how we can take advantage of these properties within the current model. We discuss this next.

10.3 Predicting the future

On their own, collapsing an object's cells and movement (§10.2.1, §10.2.2) give the appearance of compressing the state, but they are actually just presentational conveniences. While less information is needed to identify an object than exhaustively enumerating its constituent cells, it is not possible to model the outcomes of all the object's potential interactions without resorting to a huge database (or, perhaps more simply, just modelling its cells). So less information may be presented to the observer, but this is just a wrapper over what is happening beneath – it provides no simpler rules for modelling the CA's future state. An analogous argument can be made for collapsing movement over time.

But if we also collapse an object's state as described in §10.2.3 then we do get true compression, and thus the ability to predict its future state more efficiently than by running it – we discussed how we can know the future state of gliders, oscillators and other known patterns. Or rather we could predict the future state if it worked beyond a few specific circumstances and a fairly limited prediction horizon. We could certainly improve the model, perhaps by refining the object edge detection routine, but these improvements would be incremental and unlikely to address its underlying fragility and narrowness of scope.

Unfortunately the prediction problem is fundamentally linked to the way in which we have abstracted objects: our objects are conceptually easy to grasp and relatively easy to find, but they are almost entirely passive. Though we have a basic method for tracking objects over time, beyond that there are no rules to define object behaviour at the high level. It is also very unclear how we could come up with a set of rules that is general enough to be useful in most circumstances and for most objects while also being small enough to be practicable.

10.4 Elucidation through elimination

Despite these flaws, the broad principles of the method outlined here are interesting and merit further investigation. This model is an example of emergence through elimination (§9.5). By limiting the choice, by selecting the correct abstraction, by deciding what is and isn't important, it has been possible to move from a cell level to an object level. A glider exists equally at both levels, but only by eliminating unnecessary DOFs (through state compression) has it become apparent. *This didn't require any search or novel discovery; it only required elimination.*

This model is rather brittle and our abstraction was chosen manually, but next we see how more robust and useful pathways that can be discovered and evaluated automatically.

10.5 COARSE GRAINING AND EMERGENCE

We now move from Life, a 2D cellular automaton, to elementary 1D cellular automata. The reasons for this are largely pragmatic: there are only 256 elementary CAs (ECAs), and just 88 that are actually different after taking out complements and reflections, compared to $2^{2^9} = 10^{154}$ possible CAs with Life's nine cell neighbourhood, and we shall shortly discuss combinations of CAs, which causes the numbers to balloon still further.

It is useful to be able to explore the search space exhaustively to reveal the whole gamut of behaviours, rather than just dipping our toe in at a promising-looking spot and hoping for the best. We also know that elementary CAs, despite their simplicity and small numbers, display many interesting and different behaviours (§2.7, [18]) including the ability to emulate a Turing machine [70].[4]

10.6 What is coarse graining?

The behaviour of cellular automata can be modelled by other CAs with coarser granularity [123]. In a coarse graining, several (adjacent) fine cells in the one model are represented by one coarse cell in another model. So for a grain of two and a 1D CA, two fine cells would map to one coarse cell, and for a 2D CA, four fine cells would map to one coarse cell. Time also slows down at the coarse level, with the fine CA making twice as many moves as the coarse CA. A simple analogy is to imagine viewing the fine-grained CA through blurry glasses, so much of the fine detail is lost but the large-scale structure still shows through.

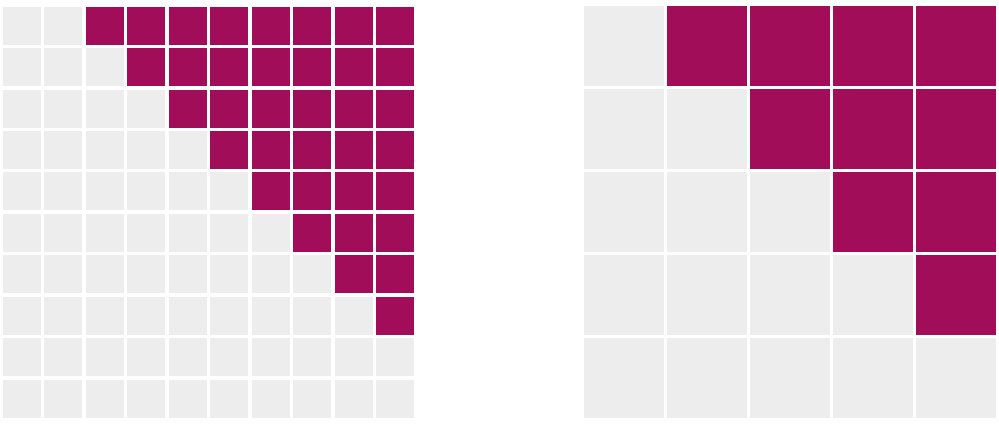


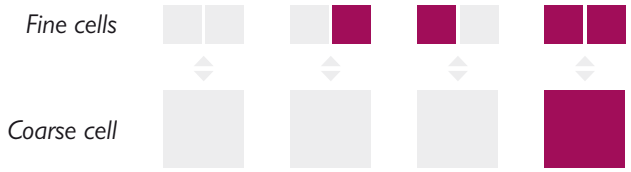
Figure 10.3 A fine CA and a matching coarse CA.

4 References to CAs in this and subsequent sections should be taken to mean ECAs unless specifically noted otherwise.

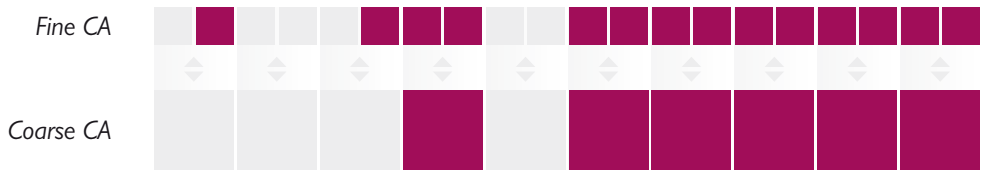
We need a way to translate from the fine CA to the coarse CA. We can do this through a mapping. Formally, the mapping is a function from g contiguous fine cells to one coarse cell (at grain g). Thus for $g = 2$

$$\text{Mapping} : \mathit{fine} \times \mathit{fine} \longrightarrow \mathit{coarse}$$

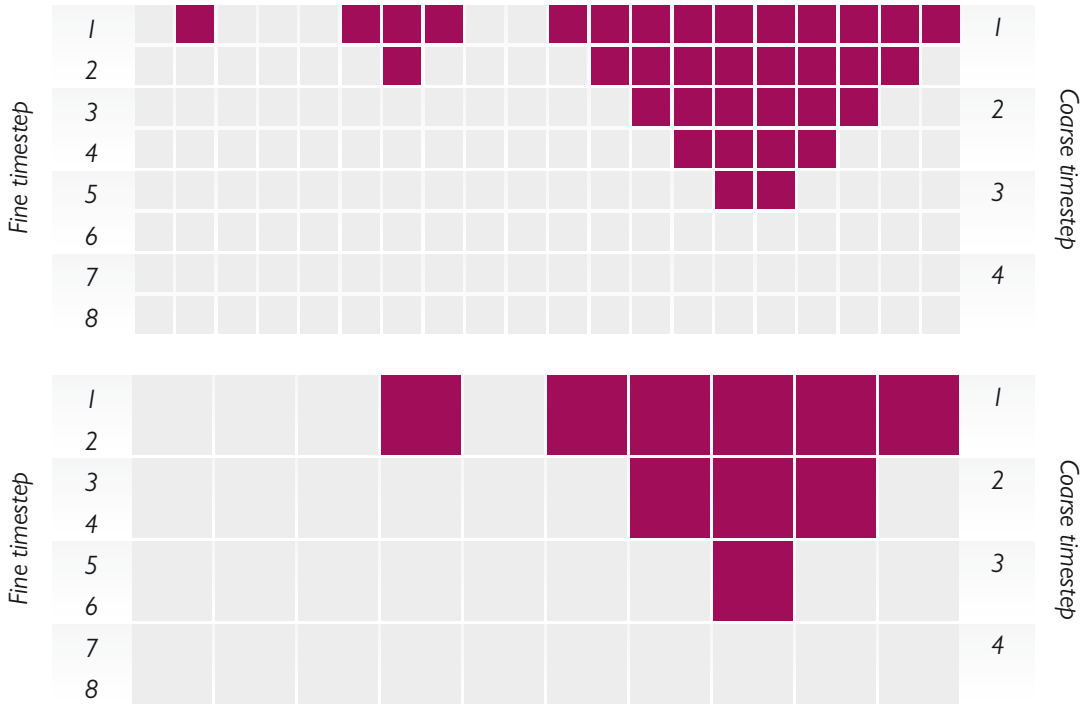
The set of mappings is similar to the transitions in a CA rule, except that they apply between fine and coarse grains rather than between timesteps. A mapping defines the relationship between every possible value combination of g cells. Figure 10.3 shows a possible mapping at $g = 2$. (Note that we ignore every other row at the fine level; this is discussed further in §11.3.)



This mapping happens to be valid (§10.8) for a coarse graining of elementary CA rule 128 to itself (so both the coarse and fine CAs use rule 128). Starting with this fine state, we use the mapping to get the equivalent coarse initial state. Both initial states are shown here.



We can now run both rules in step. Note that we only use the mapping to obtain the initial coarse rule state. After that, the CAs operate independently.



It is intuitively obvious that rule 128 will coarse grain to itself: if we view the low level CA through the blurry glasses, we end up with another CA that looks like rule 128. And while it is clear that some elementary rules like this can be coarse grained (to themselves if nothing else), there is no reason to suppose that this will hold more generally. However, it turns out that almost all of the elementary CA rules can be coarse grained: their behaviour is modelled exactly (§10.8) at a coarser grain by other elementary rules [123].

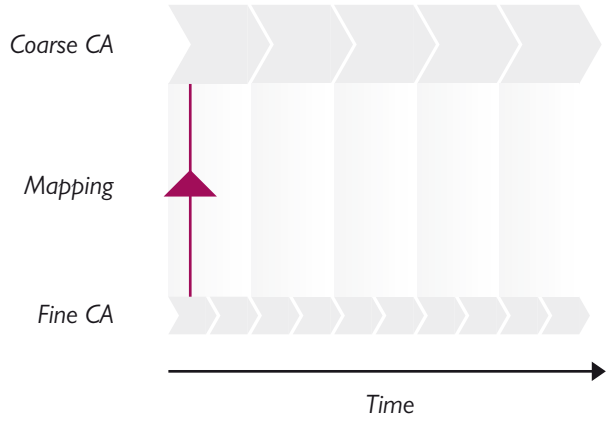


Figure 10.4 Coarse and fine CAs running over time. The coarse CA's initial state is determined from the fine CA's initial state using the mapping. The CAs then run independently. As the coarse CA has a grain of two, the fine CA takes two moves for each move taken by the coarse CA.

10.7 Why coarse grain?

Coarse graining is a simple example of emergence that fits within the definition we developed in §9.

- The emergence is subjective (§9.3). We have two languages – the high and low level CA rules.
- The coarse high level is less information rich (in general) and can be used to predict future system states (§11). The emergence is lossy (§9.5) and there is a discontinuity between the levels (§9.6).
- We keep the scope constant and lose resolution when moving to the high level, in line with Ryan's weak emergence (§8.8).
- We can develop these emergent models automatically through search (§9.17).

Just like elementary CAs, coarse graining provides a simple and transparent model of emergence that we can use to explore the phenomenon in greater detail.

10.8 Coarse graining needs consistent mappings

In order to maintain a good correlation between the high and low level models, the mapping between the layers must remain consistent. As the mapping describes the relationship between g fine cells and one coarse cell, several fine cell combinations will probably map to □ and several combi-

nations to ■. (At least one of these must be true, as there are more fine combinations than coarse combinations.)

Now consider one component of the coarse CA rule, a coarse cell triple (one component of the coarse rule, perhaps □■□) at time t and its corresponding output cell at $t + 1$. It may well be the case that two or more different underlying patterns of fine cells map to the same coarse triple, but (because the fine cells are different) the mapping to the coarse output cell is different. In fact, this sort of inconsistency would seem to be quite likely. The next sections explore how we can use these inconsistencies to find valid coarse grainings.

10.8.1 Testing a mapping

Suppose we have a fine rule and have chosen a mapping, and that we now need to see whether it is a valid mapping for our rule. Given that CAs can operate on an unbounded grid, it is clearly impossible to adopt a naïve approach to testing every starting state exhaustively. Fortunately, elementary CAs have some nice properties that make checking the validity of a mapping simpler.

- A CA cell has no memory, so each new generation (horizontal line) of a CA is equivalent to restarting the CA from that initial condition.
- All cells behave identically, so a correct mapping in one place will work in all other locations where that pattern of cells is found.
- CAs have a limited speed of information flow – information cannot be transmitted by the CA at a rate faster than one cell in any direction per step.

Taken together, these properties mean that it is only necessary, at most, to consider the input triples for each possible rule case (□□□, □□■, □■□ ... ■■■). As we shall be using this process to discover coarse grainings for a known fine rule, we actually need to generate a fine CA of these rule cases. Unfortunately it is not clear a priori which fine patterns may give inconsistent behaviour when translated to the coarse rule level, but enumerating every possible fine input condition for $3g$ cells must cover (with considerable duplication) all high level states. This means that we have to check $3g \times 2^{3g}$ fine cells.[5]

10.9 Cell mappings are sufficient

The flow of information through the CA also means that it is only necessary to consider the mappings between cells, and not between the larger number of rules that use them. We would only need to consider mappings at a rule level if it were possible to make a set of rule-level mappings that contained inconsistent cell-level mappings (for instance, if one rule case mapped □■ → □ and another rule case mapped □■ → ■). Imagine a mapping with two rules with such a cell-level mapping. For this mapping to be valid, at least one fine cell in the input state must remain isolated from the fine

5 2^{3g} possible combinations with $3g$ cells in each.

output cells. But this impossible due to the speed of information flow: even a cell at the edge of the input cells will influence the coarse rule's output.

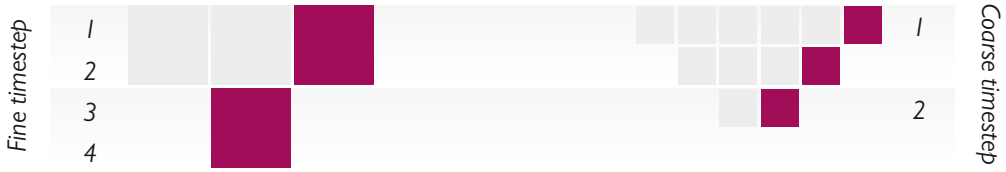


Figure 10.5 Even a cell on the edge of the input 3*g* cells affects the coarse rule's output.

So it is only necessary to consider the mapping of all states for one cell, 2^{2^g} cases for a 1D CA.[6] The entire coarse rule can then be calculated from this information. So for any elementary rule with grain $g = 2$ (which gives successful coarse grainings of most 1D CAs (§10.13, [123]), there are just sixteen possibilities

- □□ maps to □ or ■
- □■ maps to □ or ■
- ■□ maps to □ or ■
- ■■ maps to □ or ■

Fourteen have a non-trivial target state that contains both a □ and a ■.

The speed of light also means that we can simply concatenate the 3*g* cells in our test set; we do not have to consider each test case individually as they are already isolated for one coarse timestep – a cell that forms part of an adjacent test case needs at least two (at the coarse level, and $g + 1$ at the fine level) timesteps to affect the output of a given test case.

The number of possible mappings is low for small *g*, so an exhaustive search is efficient (and at higher *g*, other factors kick in to make coarse graining intractable first – see §10.15). We developed the following method to find a valid mapping and coarse grain a CA. Note that we don't know what the coarse rule is at this point – this is what we are trying to determine (along with the mapping).

10.10 How to coarse grain

1 Select an elementary rule for the fine CA.

2 Choose the granularity of the coarse graining. Here we are using $g = 2$.

3 Decide on a candidate mapping (by search, at random, or iterate exhaustively).

4 Generate a test set of all possible coarse CA input neighbourhoods at the fine CA level. We concatenate an enumeration of all possible fine states of 3*g* cells to give an initial condition of

6 Clearly this grows quickly with *g*, but in practice other factors limit the size of *g* that can be considered.

the form (for $g = 2$) □□ □□ □□, □□ □□ □■, □□ □□ ■□ … ■■ ■■ ■■. This must cover every possible low and (via the mapping) high level CA input condition. The speed of information flow means that we can simply concatenate the $3g$ cells – we are only interested in the middle output cell of each triple, and this cannot be affected by adjacent triples in one timestep.

5 Run the fine CA for the equivalent of one coarse timestep. (So for a grain of two, run the fine CA for two timesteps.) We now have the matching fine states for two successive timesteps of the coarse CA.

6 Generate a coarse CA of the fine input row (the first row) using the mapping. So if the mapping maps □□ → ■, set all □□ pairs to ■ in the coarse CA.

7 Generate a coarse CA of the output row (this is the row of the fine CA that is one coarse step on, so for $g = 2$ this will be row 3). Use the mapping again. Note that at this stage we do not know (or care) what the coarse rule is.

8 Check for consistency between the input coarse row and output coarse row: for every coarse input triple □□□ (and there will probably be several instances because multiple fine states are mapped to just two coarse ones), verify that the coarse output is the same each time (□ or ■). Repeat for all other input triples. We only look at the middle output cell as the others are not relevant to the rule and can stop immediately if we find an inconsistency.[7]

9 If it is consistent, we have a valid coarse graining and can reconstruct the coarse rule. We know that every possible coarse rule case – every possible combination of six fine cells – has been included, so unless the mapping is to rule 0 (all fine cell pairs map to □) or 255 each coarse rule instance will also be present (probably multiple times) in the mapped output and we can read off the rule.[8]

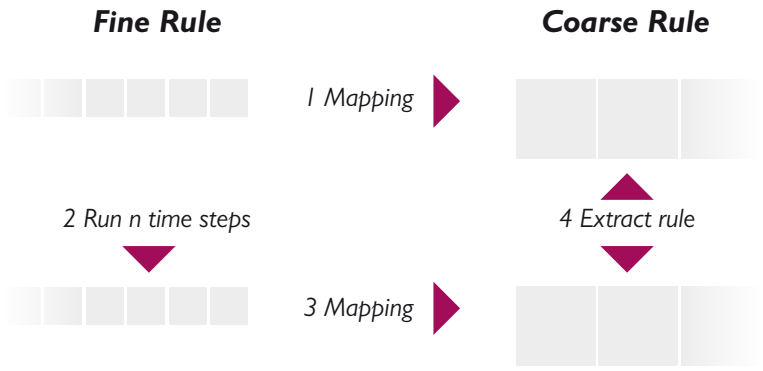


Figure 10.6 Finding a coarse rule in a coarse graining. The fine rule is run for g timesteps (where g is the grain). We use the mapping to calculate the input and output states of the coarse rule.

7 Or more specifically, each low level combination of six input and two output cells must occur elsewhere in the sequence at a point where it is checked.

8 We have decided to exclude examples that map incompletely to 0 or 255 as they show nothing interesting.

10.10.1 An example of coarse graining

1 We want to know if any coarse grainings exist for the fine rule we have chosen. In this case we are going to look at rule 188.

Figure 10.7 A run of rule 188. As before, ■ cells are represented by red squares and □ cells by white squares.

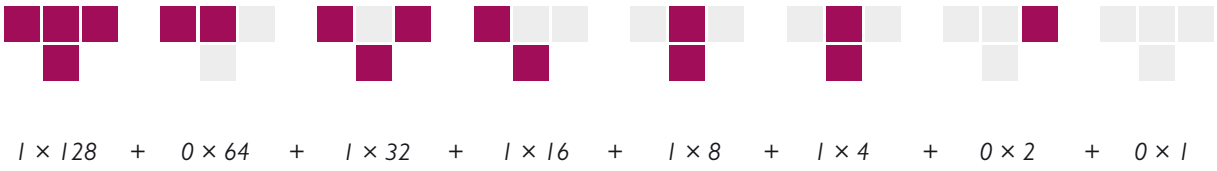
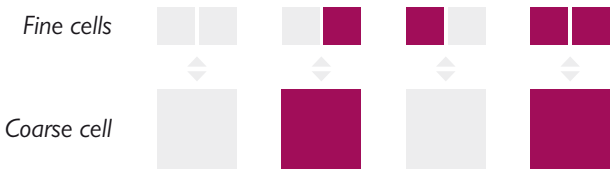


Figure 10.8 Rule 188's rule cases

2 We shall try to find matching coarse rules for 188 at a grain of two.

3 There are 14 non-trivial mappings for a $g = 2$ coarse graining. Here is one candidate, the mapping □■□■ (□□ → □, □■ → ■, ■□ → □, ■■ → ■; fine cells are always given in the same order).

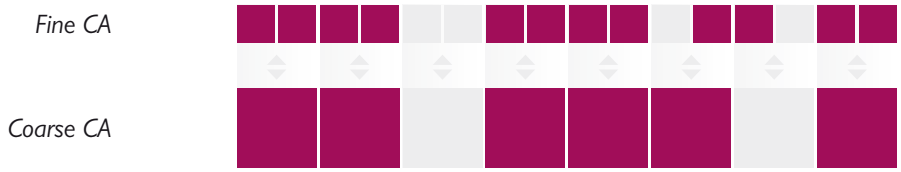


4 Create an exhaustive fine CA initial condition that includes all possible coarse rule states through the enumeration described in §10.10 step 4. This is a segment from the middle of the input state.

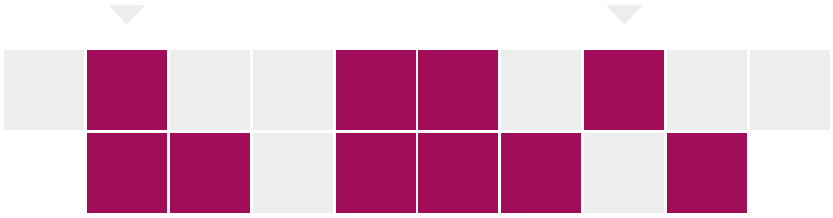
5 Run this fine CA for two steps (equivalent to one coarse step at $g = 2$). We use the rule we are trying to coarse grain (188 here).[9]

6 Use the current candidate mapping to determine equivalent coarse states for each fine cell pair for the input row.

7 Similarly, apply the mapping to the fine output row to get the coarse output row.



8 The coarse CA just created must always behave consistently if it is valid. This means that, if (for example) □■□ maps to ■ in one case, it had better map to ■ in all other cases too. Otherwise there is a problem with the mapping, and that this particular rule is not a valid coarse graining (at the current grain at least) for the fine rule in question. After placing the coarse input and output rows from steps 6 and 7 together, we can see that □■□ maps to ■ at one point and to □ at another, so it is inconsistent.



9 We repeat steps 1-8 with the mapping ■□□□. This is consistent, so we can now read off the coarse rule. Through the mapping, we see that the coarse rule is rule 192.

9 In this diagram, cells are lost from the edge of the CA at each step as their states depend on neighbouring cells not shown here.

Figure 10.9 A run of rule 192. The initial condition was calculated from Figure 10.7 via mapping ■□□□.

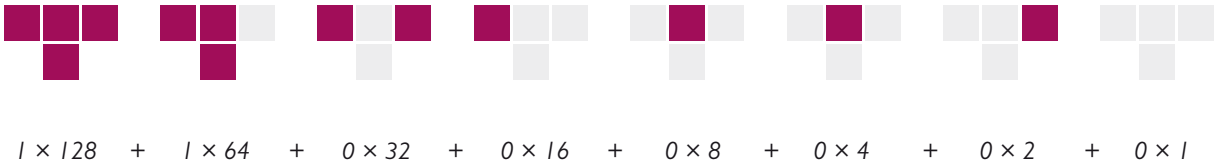


Figure 10.10 Rule 192's rule cases

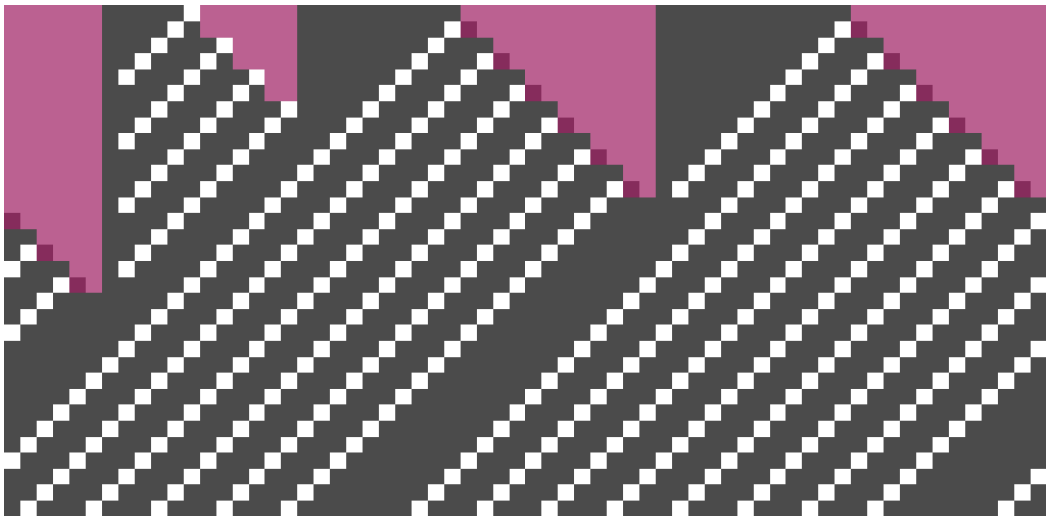


Figure 10.11 Rule 188 coarse grained to rule 192 via mapping ■□□□. The fine rule is shown in black (■ cells are black squares and □ cells white squares), overlaid with the coarse rule in red (■ cells are red squares and □ cells white squares). The coarse rule is semi-transparent, so the fine rule underneath makes a pattern of dark and light red squares.

10.11 Something of a waste

The coarse graining process described in §10.10 seems somewhat inefficient, both in terms of processor time and memory use. Rather than creating one large CA, it could easily be chunked into smaller pieces, saving a lot of memory. The representation adopted is also wasteful as only the middle cell of each rule case triple is examined.

Both of these would be significant considerations in a production system, but are not particularly relevant in this context. Reducing the memory requirements to a constant is certainly an improvement, but unfortunately the time taken to run the algorithm gets intractably long at around the same point as the naïve algorithm runs out of memory. Compressing the description also gives large savings – eliminating spatial duplication by overlapping test cases reduces the test set to just 10-20% of its original size depending on g – but again this is not significant when dealing with runtimes that increase exponentially.

10.12 Different coarse grainings

We have already discussed how emergence is subjective (§9.3). There is no right emergent property in any absolute sense, merely properties that are useful for a particular purpose. Coarse grained CAs show the same relativism.

Rule 140 can be coarse grained to 136 and 204 (with mappings □□□■ (136), □□■□ (204) and ■■□■ (204) respectively). Figure 10.12 and Figure 10.13 show the result of running these rules for a period of time. Rule 136 has captured the transient triangles in the underlying system, while rule 204 models its long term behaviour. Our choice of mapping determines which aspects of the underlying system we abstract to the high level, and thus what seems to be the model's natural emergent property. We explore this in more detail in §12.

In the following diagrams, the fine rule is shown top left (in black), the coarse rule top right (in red) and the fine and coarse rules overlaid are illustrated beneath the individual rules. The individual fine and coarse rule runs are scaled down to ¼ the size of the overlaid rules run, so a coarse cell in the coarse rule diagram is the same size as a fine cell in the overlaid diagram. The three diagrams give different views of the same CA run.

Figure 10.12 Rule 140 coarse grained to rule 136. See §10.12 for diagram interpretation.

Figure 10.13 Rule 140 coarse grained to rule 204.

10.13 Graining graphs

There are a total of 180 coarse grainings of the 256 ECAs at $g = 2$; please see §A for a complete list. Some rules, such as 0 or 128, are much more likely to appear in the set of valid coarse grainings for a given rule than others. We can plot all possible coarse grainings between the elementary CA rules on a graph. A *graining graph* is a novel way of showing the coarse graining relationships between CA rules under certain criteria, such as a particular grain and which rules were included. Rules that do not coarse grain under these conditions are not shown on the graph.

Figure 10.16 shows all coarse grainings under all mappings at $g = 2$. The graph is arranged using a force directed layout algorithm [124] (sometimes tweaked by hand). This is not significant in itself, but (in addition to providing a fairly clear layout) the algorithm tends to place related rules near to each other (as they are connected via common nodes) and move the most connected nodes towards the centre of the graph.

The graph lines show the direction of coarse graining (from the fine to the coarse rule). A few specific rules are highlighted, though all rule numbers are present inside the graph nodes. If you are reading a paper version of this report, a copy of the graphs can be found at [156].

The graph has a symmetrical structure (a result of the reflections and □ ■ symmetries in the elementary CA rule set), with certain rules acting as 'sinks' for a large number of nodes. Perhaps unsurprisingly, these are simpler rules that can coarse grain (correctly but often rather vacuously) many other rules.

The central sink in the graph is centred on 0 / 255 (all states map to □ or ■ respectively). This joins the three main arms of the graph together (as many rules coarse grain to them) and have a lot of rules that coarse grain just to them (again, unsurprisingly). We also see a large number of rules clustered around 128 / 254 (128 draws tapering isosceles triangles from cells as shown in Figure 2.7; 254 is its inverse), though here they are isolated from the main graph.

The graph's symmetry comes from mirrored pairs of rules. For instance rule 34 draws diagonal lines to the left and 48 diagonal lines to the right (Figure 10.14). There are also many inverse pairs on the graph (the same rule □ on ■ and ■ on □), though these tend to appear together as it is usually possible to coarse grain from one to the other.

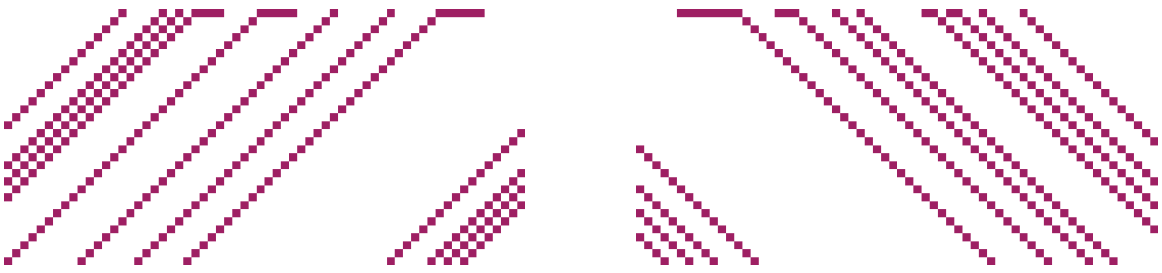


Figure 10.14 Rules 34 (left) and 48 (right) are mirror images.

Near the top of the graph, rule 136 draws right angled triangles. Rule 140, which draws right angled triangles with vertical lines extending from them, coarse grains to it (Figure 10.12). 140 also coarse grains to 204, which captures the vertical lines in its output (Figure 10.13).

Towards the bottom, rule 34 coarse grains to 170. Unlike 34, rule 170 is actually capable of drawing thick diagonal lines, but the mapping between them usually prevents rule 170 from doing so: it only allows 170 to make a line at the left edge of a contiguous fine level block, maintaining consistency between the two rules. Though if we use a pattern of alternating cells as the input condition, rule 170 does draw a thick line that corresponds to a checkerboard pattern at the lower level (Figure 10.15).

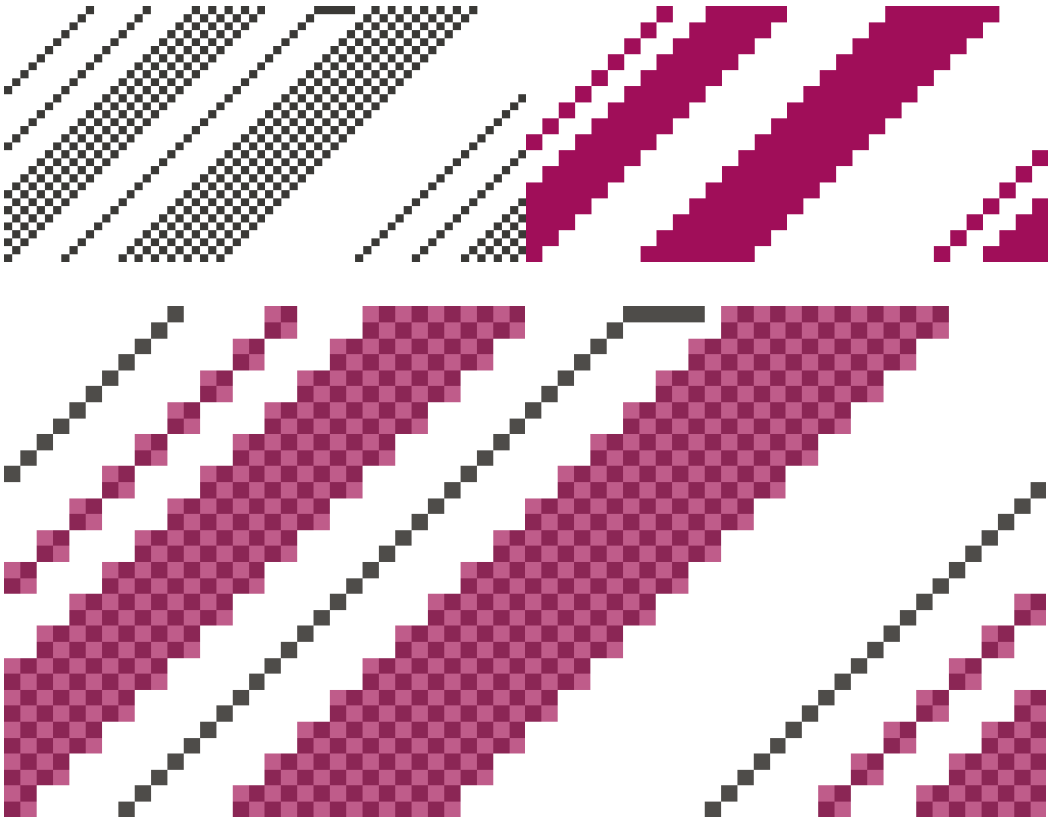


Figure 10.15 Rule 34 coarse grained to rule 170, showing the thick lines in coarse rule 170 and corresponding chequerboard patterns found in fine rule 34.

Finally, there are four pairs of isolated rules that only coarse grain to each other (and sometimes themselves).

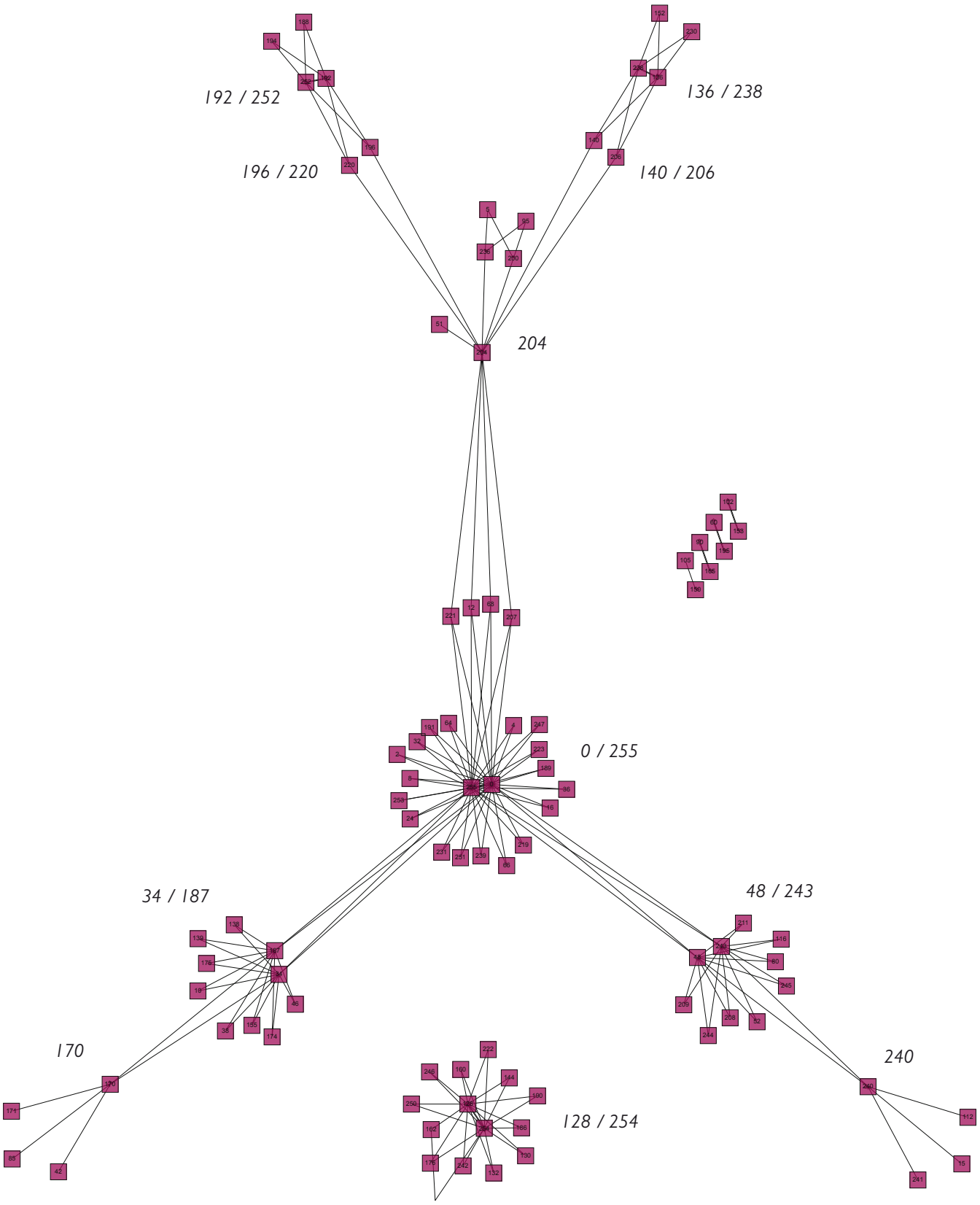


Figure 10.16 A graining graph showing all total coarse grainings at $g = 2$. The floating numbers are the rules on which that cluster is grouped, so the rules at the centre bottom are grouped around rules 128 and 254. A zoomable copy of this and the other diagrams in this section can be found at [156]. See §10.13 for further discussion.

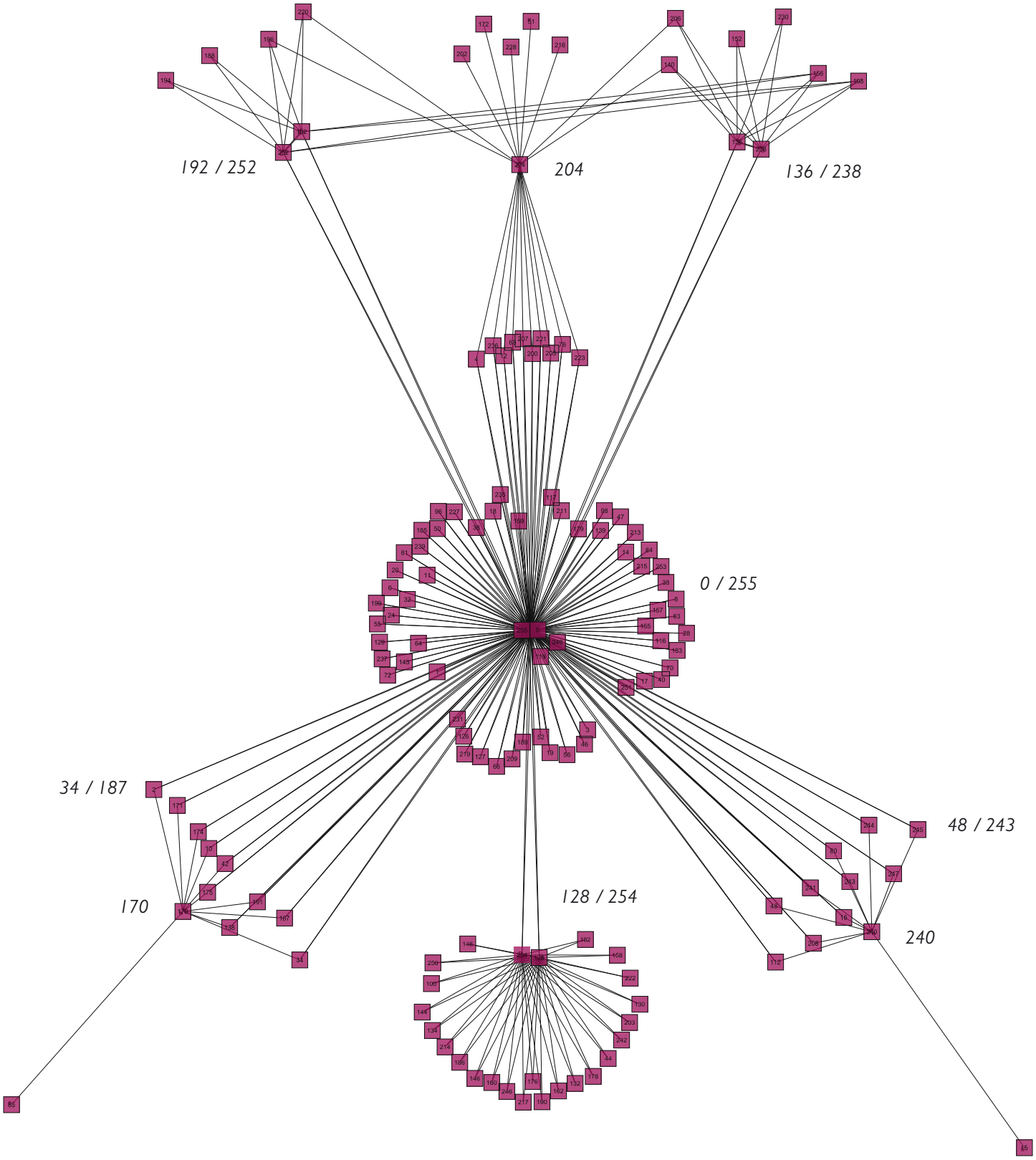


Figure 10.17 A graining graph of all total coarse grainings at $g = 3$. See §10.13 for discussion.

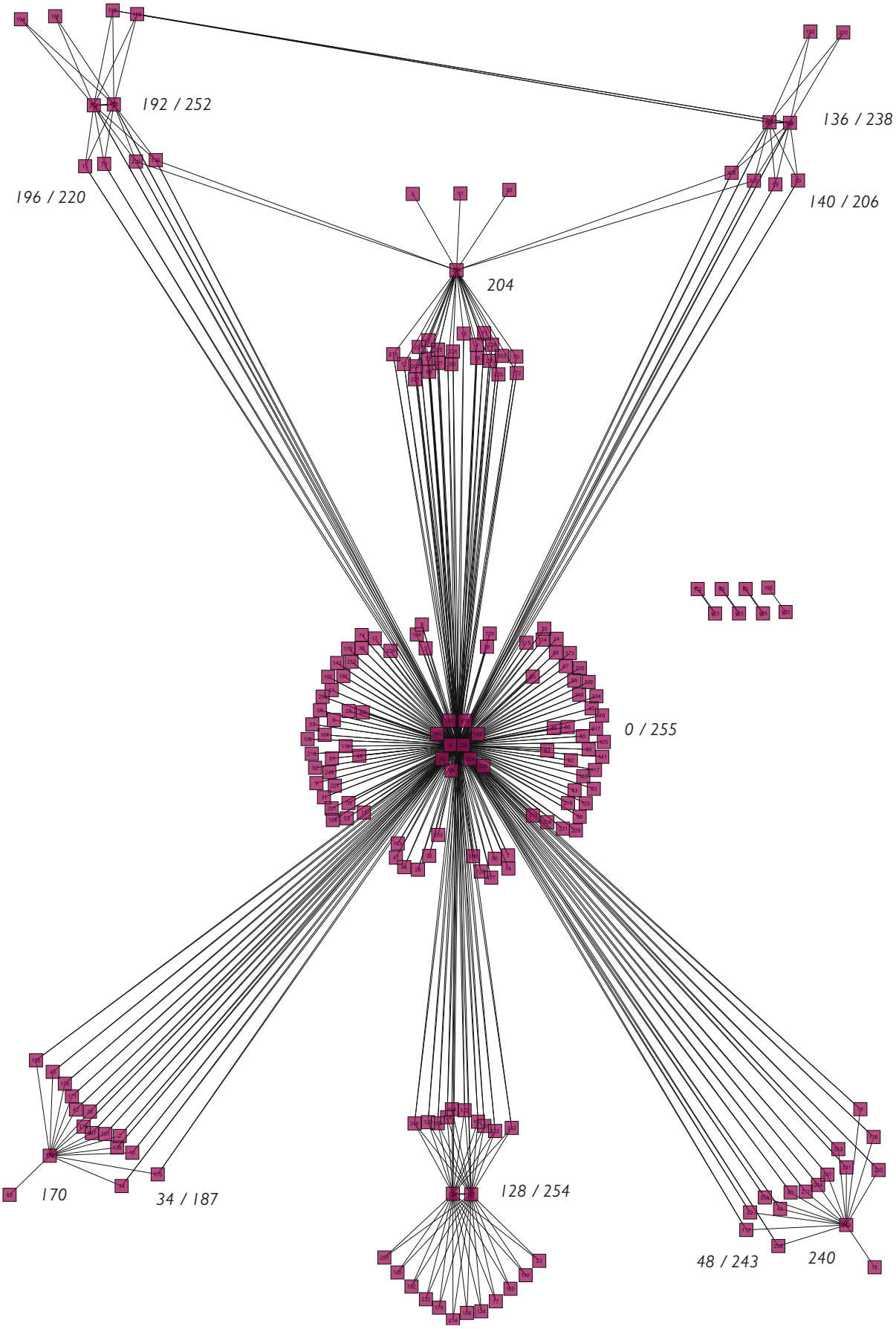


Figure 10.18 A graining graph of all total coarse grainings at $g = 4$. See §10.13 for discussion.

The graphs for $g = 3$ and $g = 4$ show a similar structure to the $g = 2$ graph.[10] There are more nodes, showing that more rules that can be coarse grained at that granularity, though most of these new coarse grainings are not that interesting as they surround large existing sinks, particularly 0 / 255. The graphs are more interconnected: the 128 / 254 cluster joins the main graph structure, and we see new links between the arms at the top of the graph.

The higher granularity graphs seem to include shortcuts, connecting some nodes directly that took two hops to reach at $g = 2$. For instance the nodes surrounding the 34 / 187 cluster (and those fanning out from 170) are directly connected to 0 / 255. Similarly, many nodes at the top of the graph are directly joined to 0 / 255, rather than having to traverse the arms and main trunk of the graph.

It should be noted that this is not simply a case of two adjacent $g = 2$ coarse grainings being linked by one $g = 4$ coarse graining. While this does happen, a coarser granularity also adds significant numbers of novel links to the graph. Rule 204, for example, is a sink and does not coarse grain to anything but itself. Rather, it seems that the elementary CA rules have a certain affinity and similarity between them and a higher granularity, with its less strict matching requirements, makes it easier to see these links. We discuss these relationships further in §10.19.

10.14 Moving to two dimensions

Many interesting CAs – most famously Conway's Life – exist in a two dimensional environment. Israeli and Goldenfeld [123] state that "[g]eneralizations to higher dimensions and different interaction radii are straightforward," but provide no examples where they did so. Nor do we. The reason for this is simple: the numbers involved are scary.

Consider coarse graining Life with $g = 2$ (the smallest size possible). Because the grid is 2D, the number of cases to consider becomes $2^{2^{g^2}}$, or 65536. Similarly, we must now consider 9 squares rather than 3. This means there are now 2.47×10^{12} ($2^{2^2 \times 9} \times 2^2 \times 3^2$) cells in the test set. A 1D CA coarse grained with $g = 2$ has just 16 cases and 384 cells in the test set.

It is possible to reduce these numbers significantly in the case of Life by exploiting some of its characteristics. Life is a totalistic CA – the outcome is the same whether 3 live cells are all on one side or all on different sides. This cuts the number of unique cases from 65536 to 6 (live cell dies of loneliness, live cell survives, live cell dies of overcrowding and analogous dead cell cases). Similarly, the size of the test set reduces dramatically.

It is not reasonable to assume that any coarse graining of Life will also be totalistic, but it must be the case that any coarse graining of Life (or at least any total coarse graining – §10.16) will have the same symmetries as the original. Life has four axes of symmetry, limiting the maximum number of

10 Note that the graphs at $g = 3$ and $g = 4$ have been rearranged to match the $g = 2$ graph – the large number of nodes around major sinks (particularly 0 / 255) skewed the layout algorithm, and we felt the node arrangement in the $g = 2$ graph was clearer for showing important structure.

tile types to three – two edge types and the centre tile – giving 5 × 5 × 2 = 50 possible cases (within each group the cells are totalistic, so the edge tiles can each be in five different states and the centre tile in two) and reducing the test set hugely. Taken together these represent a substantial saving, but the numbers are still too large to handle on a typical desktop computer.

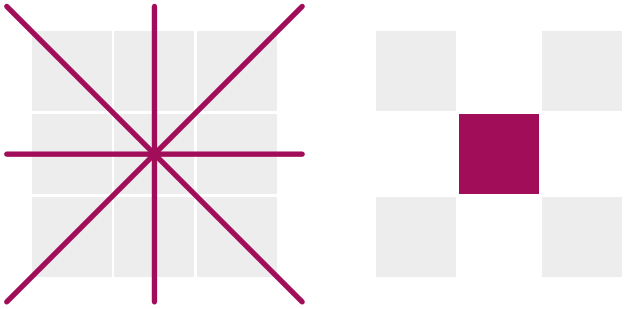


Figure 10.19 Symmetry in Life

10.15 Explosive test cases

For 1D ECAs, the number of possible mappings is very small for a grain of two, small enough for a standard PC to calculate exhaustively in less than a second. At $g = 2$, we have seen there are $2^{2^g} = 16$ possible mappings to consider. The test set of fine cells used to check the validity of coarse grainings is

$$2^{3g} \times 3 \times 2 = 384 \text{ cells}$$

Combinations of 3*g* blocks × coarse block size × grain

Due to the exponentials, these numbers increase rapidly. For $g = 3$ they are 256 and 4608 respectively; at 4 they become 65,536 and 49,152; and by $g = 5$ they have already reached an intractable 4.29×10^9 mappings over a test set of size 491,520. Multiply these factors together and we have some very big numbers.

Grain	Possible mappings	Cells in test set
2	16	384
3	256	4608
4	65,536	49,152
5	4.29×10^9	491,520
6	1.844×10^{19}	4,718,592
7	3.40×10^{38}	44,040,192
8	1.16×10^{77}	4.02×10^8
9	1.34×10^{154}	3.62×10^9
10	1.80×10^{308}	3.22×10^{10}

Figure 10.20 The number of possible mappings and test set size for various grains.

We can overcome the two problems of increasing *g* – population size and test set size – by using partial solutions.

10.16 PARTIAL COARSE GRAINING

The mappings considered so far (and those considered by Israeli and Goldenfeld [123]) are total. They give an exact match over all possible states: they lose information moving to a coarser grain, but the systems never differ in their predictions. Here we develop a new approach by searching for partial mappings, in which we allow some discrepancies between the fine and coarse CAs' results; for example, □■ may map to □ in one instance, but to ■ in another.

Formally, the mapping in a *partial coarse graining* is a relation between g contiguous fine cells and one coarse cell (at grain g).[11] For $g = 2$

Mapping: *fine* × *fine* ↔ *coarse*

This may seem like a backwards step. The previous discussion of matching mappings implies that total coarse grainings are ideal, and we should aim to get partial coarse grainings as close to that as possible. While broadly true, being total is neither necessary nor sufficient for a coarse graining to be good. (What 'good' may mean is a difficult question, and one that is addressed more fully later in §11 and §12. For now, we shall define this loosely as modelling the desired high level behaviour.)

To understand why, it is important to note what a coarse graining is: it is a high level model of certain aspects of the underlying CA's behaviour. And that is all. A total coarse graining captures these aspects in all cases; a partial coarse graining may not.

The characteristics captured may not be those in which we are interested, or (in as much as it is possible to state in absolute terms) be particularly interesting. For example we have seen that many rules coarse grain to rule 0, but arguably this is not very useful as no information about the system is retained.[12]

A total coarse graining must always model some aspect of the underlying system's behaviour without error. It cannot add any new behaviour to the system. A partial coarse graining does not have this restriction, and its erroneous behaviour adds new information to the model (§11.7). While in some sense undesirable, this can be useful if the extra information allows the model to approximate (most of the time) more of the underlying system's behaviour than would otherwise be possible with a model of that complexity.

Ideally these additions would be chosen to cover a broad range of behaviour efficiently, reflecting the behavioural facets of most interest to us and that which we wish to emulate. Usually this would be the CA acting in a 'typical manner'. (Compare this to physical emergent systems, where emergent properties only present themselves over some restricted set of all possible states, such as

11 The mapping that forms part of a partial coarse graining (along with the fine and coarse rules) is still a function (or a number of functions §10.20), but calculating the coarse CA from the fine CA with this function will sometimes be inconsistent with the actual coarse CA.

12 We only include mappings with both □ and ■ in them.

a particular temperature range.) The idea of finding a solution that works most of the time, and finding it efficiently, is a central tenet of neutral emergence (§9.9).

Partial coarse grainings also have efficiency advantages over their total counterparts. We know that calculating total mappings quickly becomes intractable even for quite small grains and elementary CAs. But we shall see that we can get useful coarse grainings from remarkably little information, averting the exponential explosion we encountered before. It also turns out that there are additional subtleties exposed by partial mappings, and it is possible to exploit this 'hidden information' to find better solutions more quickly.

10.17 A partial population

As discussed earlier in §9.10, the ideas behind neutral emergence also form a large part of the rationale for evolutionary algorithms, and we have already sketched how a genetic algorithm can be used as an alternative to a brute-force search to find emergent solutions. Here we see how we can use this general technique to coarse grain an elementary CA.

The initial population comprises random mappings from all fine states to the coarse states (for $g = 2$ □□, □■, ■□, ■■ → □ or ■ at random), a small subset of the population space. (Or, in the case of $g = 2$, a superset (more accurately superbag) of the population space as there are only 14 possibilities, less than the likely population size.)

The same evaluation process as before – looking for mismatches – is applied. Unfortunately a shortcut used last time in step 8 of §10.10 (only checking each solution until a problem is found) cannot be used here, as it is necessary to check every case for each individual – we need to know how good the result is, not just whether it is completely correct.

This GA successfully finds ECA total coarse grainings (although not on every run), supporting our assertion that mutual information is an appropriate fitness function (§9.17). Using a GA is much less efficient than a brute force search for low g. However we have seen that, as the coarse graining size increases, brute force search quickly becomes computationally intractable and GA search becomes attractive.

As this is very much an initial exploration of the area, we spend most time working with $g = 2$ coarse grainings and are able to exhaustively cover the search space in our experiments, so we do not discuss partial populations further.

10.18 Finding a partial coarse graining

Finding a partial coarse graining is similar to finding a total one, though with a few key differences

- A different (and usually smaller) input string is used instead of an enumeration of all possible states (step 4 in §10.10).

- Because novel rule cases may appear at any point along the input string, every output cell is considered during evaluation so we fully exploit the partial information (step 8). Thus ■□■■□■ yields overlapping neighbourhoods ■□■, □■■, ■■□ and ■□■. (As the total coarse graining input string is a concatenation of all possible input triples, we need only look at these triples' central output cell to obtain all CA states (e.g. does ■□■ go to □ or ■ in this rule?), but this is not always true for a partial coarse graining.)
- To find the coarse rule, we must discover all of the coarse rule's outputs. But the smaller input string may not include each of the eight coarse rule cases when mapped to the coarse rule (step 9). For instance the state ■□■ may not be present at all.

This last point may require additional work, as simply reading off the rule will not always be possible. However, unless the mapping is to rule 0 or 255,[13] then at least one output mapping to □ and to ■ must exist, and this relationship can be used to reconstruct the complete rule. We add these steps to point 9 in §10.10

9.1 Use the inverse mapping to create a fine CA equivalent to the missing coarse rule cases. (The inverse mapping is a relation, and any valid mapping will work.) Concatenate the rule cases to make the fine CA. This fine CA should use the same rule as the original fine CA.

9.2 Run the fine CA for the equivalent of one coarse timestep.

9.3 Use the mapping (forwards this time) to obtain the coarse output value for each rule case. We now have a complete coarse rule.

While any mapping will work, we see shortly (§10.20) that some mappings are better than others. We can also use results from multiple mappings to obtain a much higher quality set of coarse grainings.

10.18.1 Example of deriving the rule for a partial coarse graining

Suppose we are trying to find partial coarse grainings for rule 130 (at $g = 2$). After performing steps 1-8 in §10.10, the candidate mapping in Figure 10.21 has proved to be consistent over the partial string □■□■□□■□□■□■, but the coarse CA does not contain outputs for several rule cases. We need to find the missing output states.

13 Which are excluded.

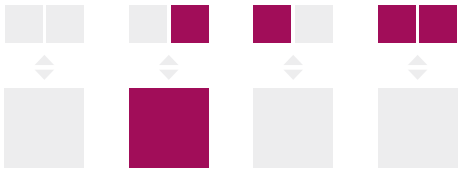
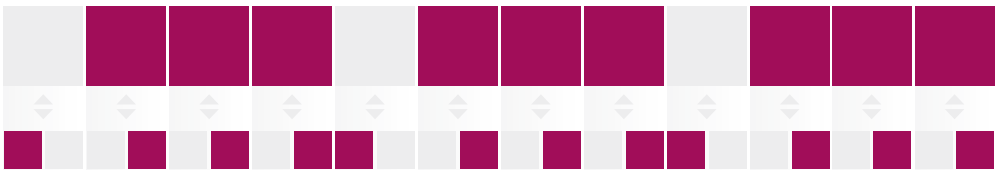


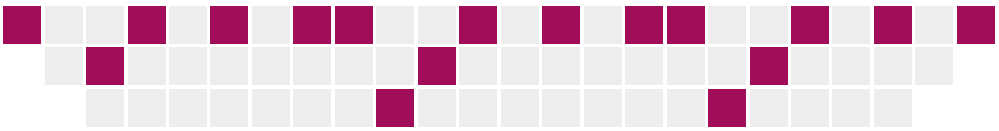
Figure 10.21 Candidate mapping for rule 130

Figure 10.22 Known coarse rule cases at this stage for the coarse graining of rule 130. We only know to what four of the eight coarse rule cases map at the moment.

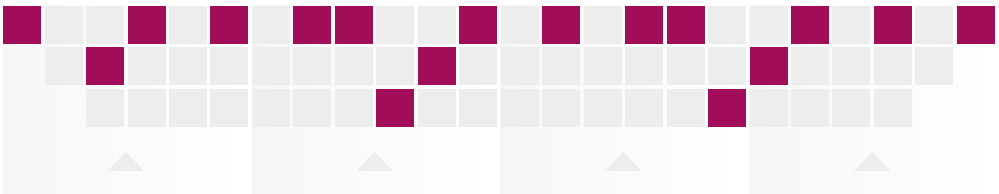
9.1 There is only one mapping to ■, so we select it. There are three mappings to □ and we arbitrarily select ■□ (see §10.20 for more on choosing mappings). We use this reverse mapping to create a fine CA of the missing coarse rule cases.



9.2 Run the fine CA for two steps.[14]

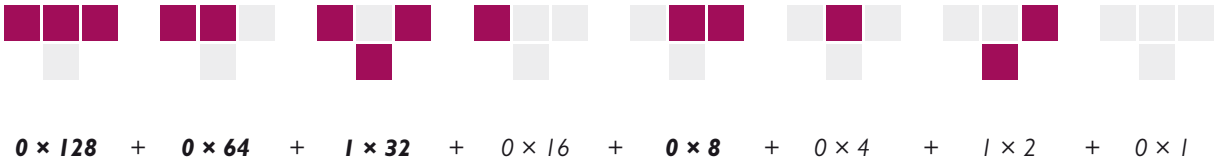


9.3 Read off the centre two cells of each group of six.



14 As before, we omit cells from the edge of the CA at each step as their states depend on neighbouring cells not shown here.

Apply the forward mapping in Figure 10.21 to these centre cells to obtain the coarse output for the missing coarse rule cases. We now have the complete coarse rule suggested by this mapping, rule 34.



0 × 128 + ***0 × 64*** + ***1 × 32*** + *0 × 16* + ***0 × 8*** + *0 × 4* + *1 × 2* + *0 × 1*

The following chart summarises these this process. Steps carried out at the coarse level are shown at the top and those at the fine level are shown at the bottom, joined together by steps using mappings.

10.19 Partial graining graphs

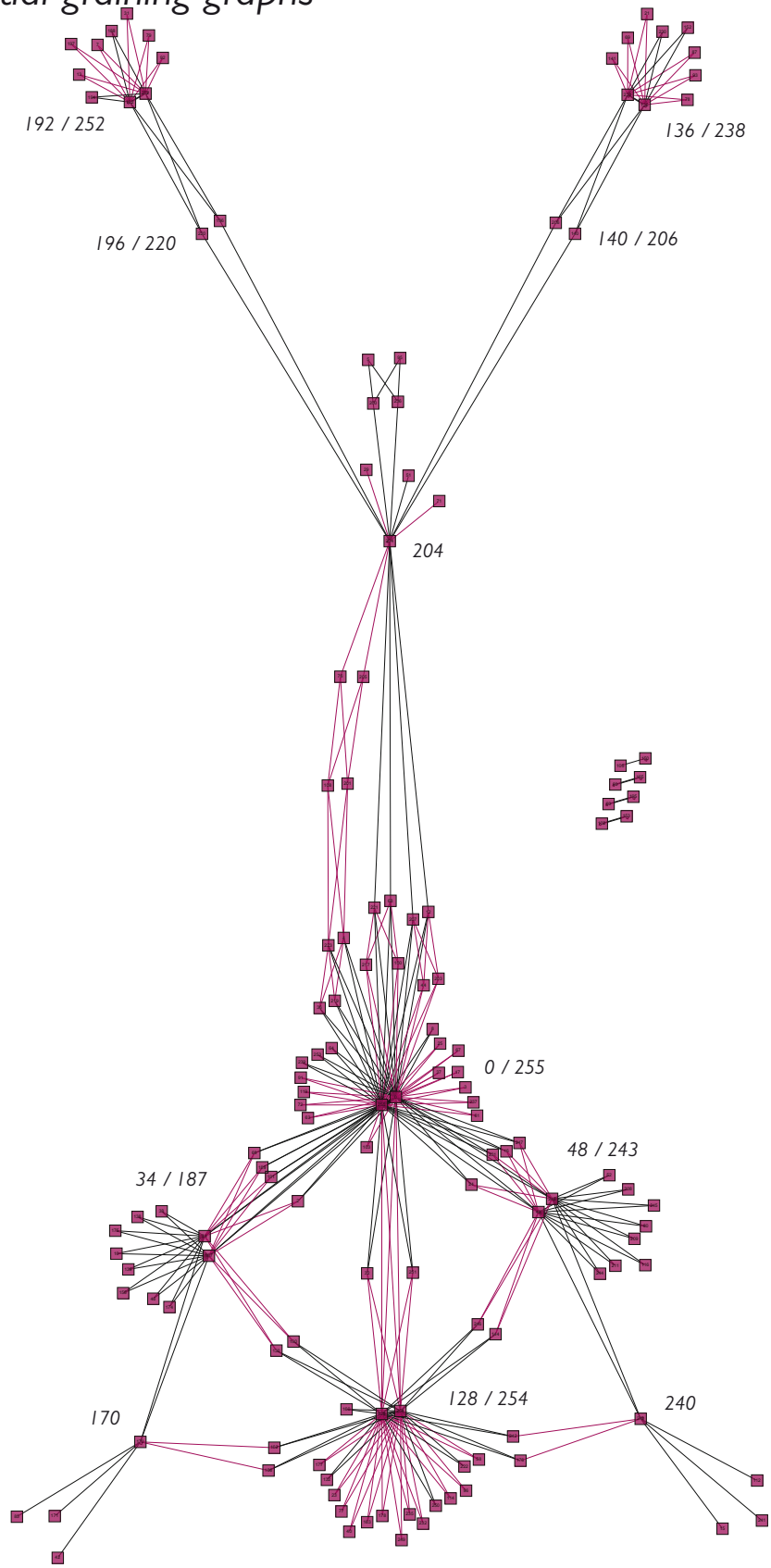


Figure 10.23 A graining graph of all partial coarse grainings from input □■□■□■□■□ at $g = 2$. Coarse grainings previously seen in Figure 10.16 are shown in black and new partial links are shown in red. A zoomable copy of this and the other graining graph diagrams can be found at [156].

The partial graining graph has a similar structure to the total graph, but adds several new joints to the central spine (the total coarse grainings are in black and new partial coarse grainings are red). The additions suggest that partial coarse graining captures the same types of relationships as total coarse graining: the graph appears to build on what was there before, offering new insights into the problem space. Rather than building a completely separate graph structure, the new links seem to expose relationships that exist but have been hidden until now. Disconnected clusters are joined together: the 128 / 254 cluster is linked to the main graph at the central 0 / 255 sink and to the lower two arms of the graph.

We see later in §11.11 that partial coarse grainings can be used to predict total coarse grainings at a higher grain. We also see that the new partial coarse graining links can capture a lot of the underlying model's behaviour – more, often, than the total coarse grainings.

10.20 Mappings

Selecting the right mapping can be as crucial as choosing the right rule when coarse graining. Once running, the coarse CA is entirely independent of the fine CA, so there is just one chance – when the coarse initial state is made from the fine initial state – to get the mapping between the levels right. We discuss this further in §12.5.

We know that there are more fine rule cases than coarse cases, so there must be several mappings to at least one of the coarse □ and ■. It is perhaps obvious that two different course grainings of a given rule will often have two different mappings. And we know from coarse graining that different mappings find different coarse rules (§10.11).

However, it seems less intuitive that a single coarse graining can also have several different mappings. In other words, one fine rule and one coarse rule can be joined through a number of distinct mappings.

This is because, although closely related, a mapping is distinct from (and, as we have seen, used to find) a coarse rule. And just as two different mappings can give different coarse rules, it is also possible for two different mappings to lead to the same coarse rule. (Remember that the only requirement here is for consistency.) It's worth noting that this happens with both complete and partial mappings (see Figure 10.24-Figure 10.27 and §11.12).

This is actually quite a common phenomenon. It can also be quite significant. In the total coarse graining procedure outlined in §10.10, any valid mapping will work at step 3. While this is still true for partial coarse grainings, some mappings will work better than others. If we were to perform the additional partial steps just outlined (rather superfluously, as the answers are already present) on a total coarse graining, it doesn't matter which mapping to □ or ■ we choose, as each case is guaranteed to give the same (perfectly correct) answers. This is not necessarily true for a partial coarse graining, however. It may well be that one mapping gives a better coarse rule than another.

We can see this phenomenon in rule 150. Rule 150 coarse grains to itself at $g = 2$ with these mappings

Mapping	Total?
□□□■	
□□■□	
□□■■	▪
□■□□	
□■□■	▪
□■■□	▪
□■■■	
■□□□	
■□□■	▪
■□■□	▪
■□■■	
■■□□	▪
■■□■	
■■■□	

Figure 10.24 Mappings for rule 150 coarse grained to itself at $g = 2$. The binary sorting applied to this table is a bit misleading as no digit is more significant than any other.

Figure 10.25-Figure 10.27 illustrate the results of running the CA with each of the mappings, starting from a single cell in an odd and even location on the grid. The coarse CA obtained from mapping □■■□ closely matches the underlying fine CA pattern in both cases. □□■□ follows the pattern in one case only, while □□□■ ignores the structure completely. Again, it's worth emphasising that the coarse rule is identical in all cases – the only change is in the coarse initial condition, determined from the mapping.

Clearly starting from just one cell is not terribly representative of the whole gamut of rule 150's behaviour, but we see similar variations in some rules with much more complex input strings as well. In some cases a marked discontinuity can be seen, switching suddenly between little common behaviour and a close correlation. We examine this further in §11.12 once we have introduced mutual information as a way of quantifying the 'goodness' of a coarse graining.

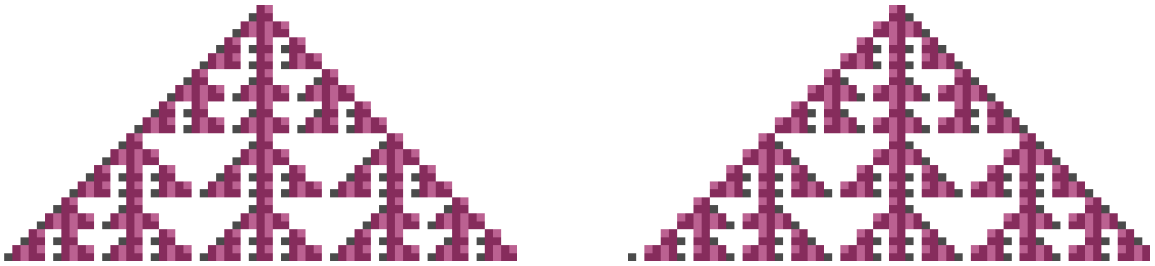


Figure 10.25 A run of rule 150 coarse grained to itself with mapping □■■□.

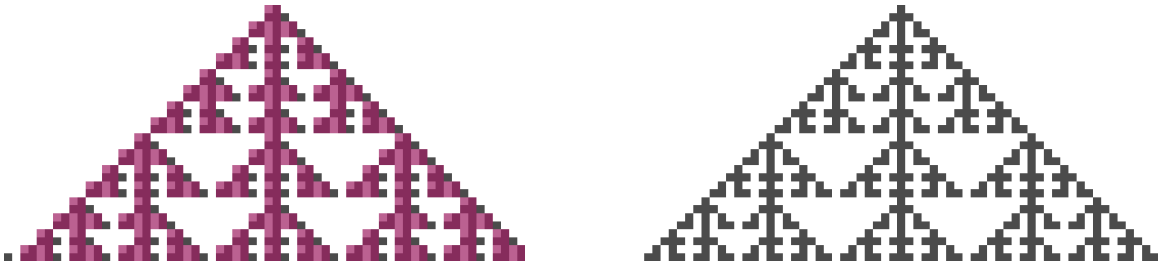


Figure 10.26 A run of rule 150 coarse grained to itself with mapping □□■□.

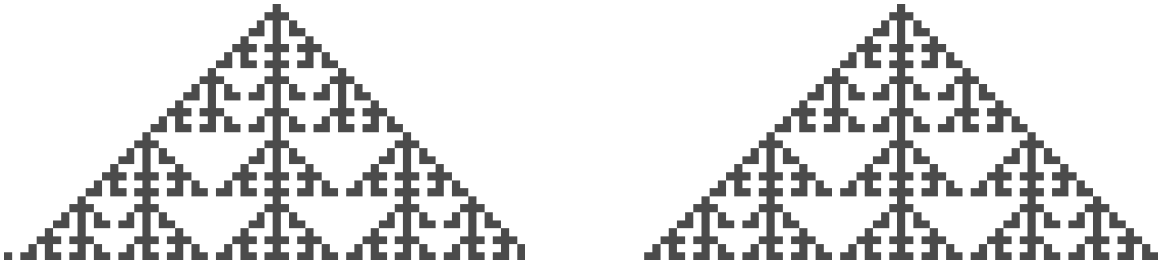


Figure 10.27 A run of rule 150 coarse grained to itself with mapping □□□■.

10.20.1 Using the first valid mapping

We have seen that some mappings are better than others when partially coarse graining, so simply selecting the first matching mapping as we have done until now would appear to be a poor choice. And it largely is, though it serves as a useful benchmark for subsequent approaches.[15]

The modified procedures described in the following sections are the same as that described in §10.10, except that we use the inverse mapping on all rule cases, not just those for which we don't know the mapping. In other words, we ignore the partial rule information we have discovered so far and use the first inverse mapping that matches instead.

This seems like a retrograde step. And (again) it is, though the answers returned are surprisingly good, considering the only information passed forward to this stage of the algorithm is that this mapping and fine rule combination has potential to work.[16]

15 Note that some of these results are influenced by our choosing the first possible mapping; had we selected (for instance) the last, we would have obtained different results in some places, though the distinctions are minor.

16 Though these arbitrary mappings do give inconsistent and misleading results in some cases, for instance indicating that a coarse graining has a lower MI when found through this partial coarse graining technique than when found through total coarse graining.

We have taken this step because it allows us to concentrate on the main issues when generating a complete rule from a partial mapping. At the end of this section, we discuss the additional problems of using existing partial matches and present a solution.

10.20.2 First valid mapping steps

The new steps we add to point 9 of the coarse graining algorithm (§10.10) are almost identical to those in §10.18.1.

9.1 Use the inverse mapping to create a fine CA equivalent to all elementary rule cases for the coarse rule. Select the first valid choice in the mapping relation in each rule case. Concatenate the rule cases to make the fine CA. This fine CA uses the same rule as the original fine CA.

9.2 Run the fine CA for the equivalent of one coarse timestep.

9.3 Convert the CA's output back to the coarse level using the (forwards) mapping (which is a function in that direction) and read off the coarse rule. We now have a complete coarse rule.

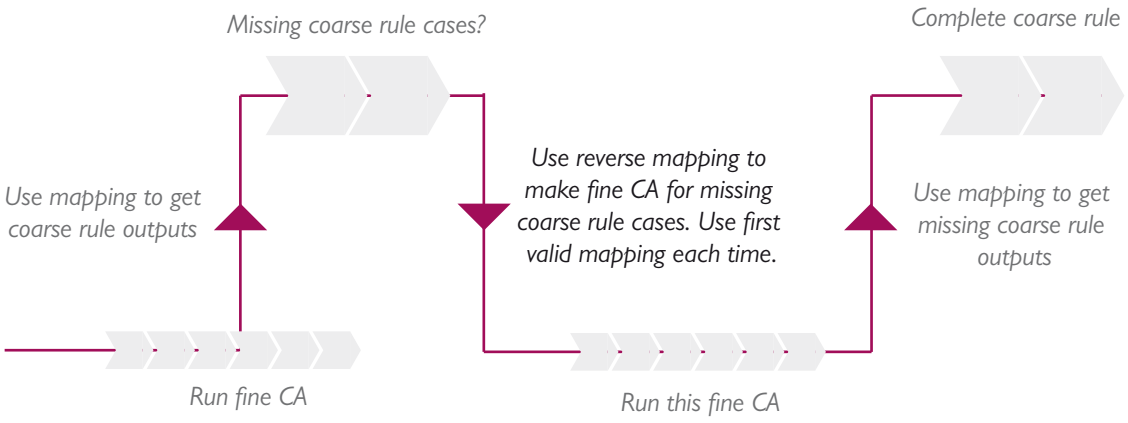


Figure 10.28 Steps for first valid mappings.

10.20.3 First valid mapping results

We recorded the number of valid coarse grainings returned with a number of different initial conditions. The numbers in Figure 10.29 don't look very good, with almost ten times as many answers as with a total coarse graining. With longer and more complex initial conditions, the number of coarse grainings decreases quite substantially, but the set is still about five times the size of the total one.

Initial condition	Coarse grainings	Percentage of total coarse grainings
□■□■□■□■□■	1432	796%
□■■□■	1237	687%
□■■□■■	1144	636%
□■■□■□■□	1045	581%
□■■□■□■□ ■■□□	794	441%
Total	180	100%

Figure 10.29 The middle column shows the number of valid partial coarse grainings returned at $g = 2$ for various initial conditions, with the number of total coarse grainings in the bottom row. These results are also given in the right column as a percentage of the number of total coarse grainings. There are significantly more partial coarse grainings than total ones.

10.20.4 Using all mappings

We can increase the quality of our results by intersecting the results from all possible mappings. This set of rules must be a superset of the total coarse grainings (though not necessarily a proper one) as all valid total coarse grainings are also valid for all mappings. This set is also likely to contain the best rules (because they work most of the time) and we see that it significantly reduces number of results returned.

This means we must try every combination of potential mappings in inverse mapping relation. Even at a grain of two, naïvely checking all possible mappings for all input cases is quite time consuming. To cover all eight rule cases, we must plough through $3^{8\times3} \times 1^{8\times3} = 2.82 \times 10^{11}$ (if we have three mappings to □ and one to ■ or vice versa) or $2^{8\times3} \times 2^{8\times3} = 2.81 \times 10^{14}$ (two mappings to each) cases. In practice we don't have to check all of these as we can stop as soon as we detect an inconsistency within the mapping, but this approach is still prohibitively time consuming.

10.20.5 All mappings steps

Use the inverse mapping to create a fine CA equivalent to all rule cases for the coarse rule. Select the first valid choice in the mapping relation in each rule case. Concatenate the rule cases to make the fine CA. This fine CA uses the same rule as the original fine CA in the coarse graining. Again, we add these steps to point 9 in §10.10.

9.1 Use the inverse mapping to create a fine CA equivalent to all elementary rule cases for the coarse rule. Select the first valid choice in the mapping relation in each rule case. Concatenate the rule cases to make the fine CA. This fine CA uses the same rule as the original fine CA.

9.2 Run the fine CA for the equivalent of one coarse timestep.

9.3 Convert the CA's output back to the coarse level using the mapping (which is a function in that direction). We now have a complete coarse rule.

9.4 Repeat steps 9.1-9.3 using the second valid choice in the mapping relation (if one exists) for the first coarse rule case. Repeat for the third choice there is one. Keep the mapping choices for the other seven rule cases unchanged.

9.5 Select the second mapping choice for the second rule case and repeat steps 9.1-9.4. Repeat this process for all mapping choices and rule cases until all combinations have been tried.

9.6 If the same coarse rule is returned by all test cases, we have a valid coarse graining. If not, we discard it.

10.20.6 All mappings results

While very expensive to calculate, the results returned by this approach are much better, increasing the total coarse grainings set size by only 54% for a very short and simple initial condition and only 17% for a more complex one. It is also worth noting that, as well as being short, the first test string in Figure 10.30 consists entirely of the pattern □■. Really quite simple and short partial mappings can get pretty close to the complete mapping.[17]

Initial condition	Coarse grainings	Percentage of total coarse grainings
□■□■□■□■□■	278	154%
□■■□■	254	141%
□■■□■■	242	134%
□■■□■□■□	222	123%
□■■□■□■□■■□□	210	117%
Total	180	100%

Figure 10.30 The number of valid coarse grainings returned at $g = 2$ for the initial conditions in Figure 10.29. The number of partial coarse grainings is significantly reduced.

17 This isn't always the case. The string ■□■■□■■□■■□■ works considerably less well at $g = 2$, though interestingly performs much better at $g = 3$, perhaps because of its periodicity.

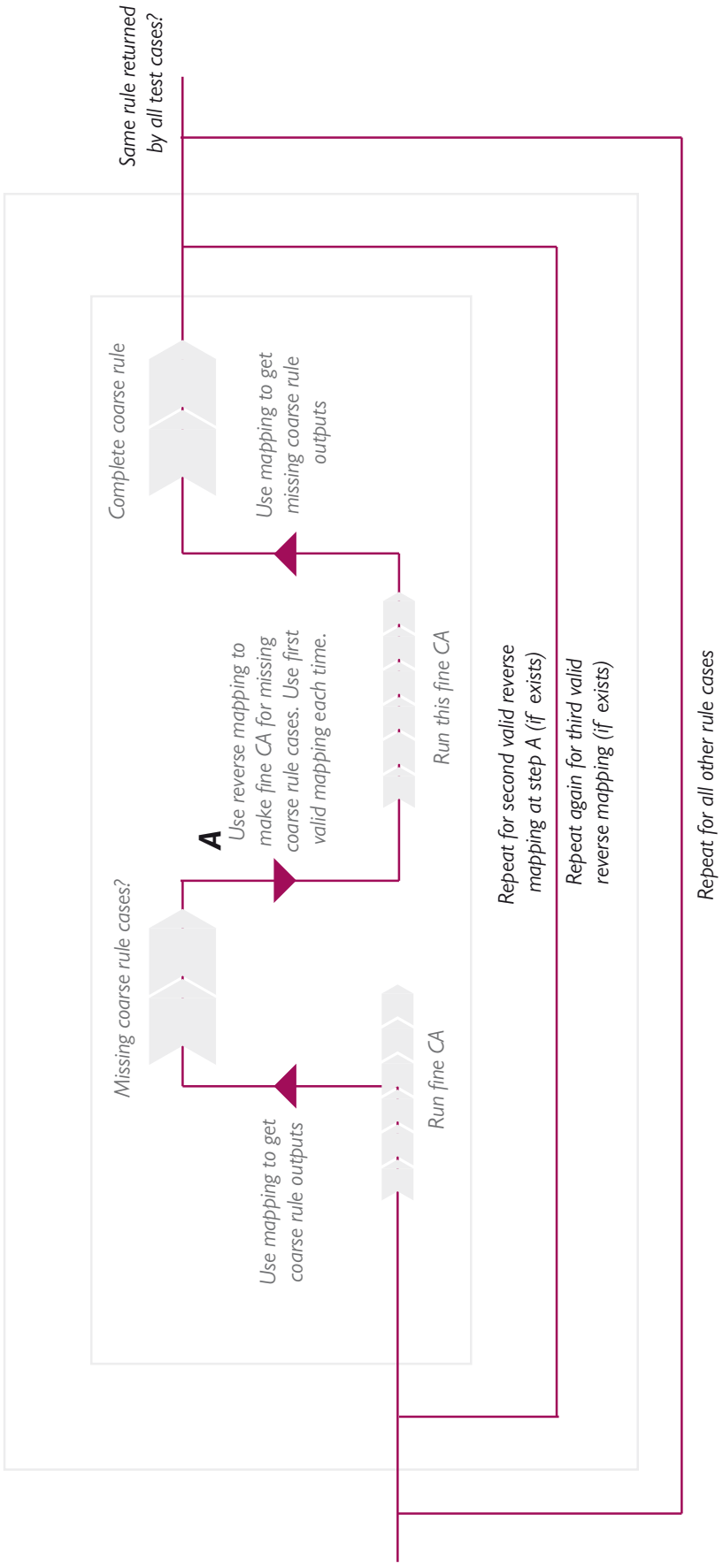


Figure 10.31 Steps for all mappings.

10.20.7 Checking all mappings more efficiently

The above process shows that we actually check the rules' consistency through their mappings, and not by verifying the underlying fine rule directly. There are, of course, far fewer mappings than fine rule cases. We also note again (§10.8.1) that the rule cases are independent of each other (indeed this is why we were able to concatenate them). These considerations allow us to move the exponential from the relatively large number of rule cases to a much smaller number of mapping cases, and thus cut the search space dramatically.

10.20.8 More efficient all mappings steps

9.1 Start with a candidate mapping as before, but this time use it to generate the fine possible equivalents (for every possible mapping) for just one particular coarse rule case (for instance □■□). Concatenate these fine rule equivalents to make the fine CA.

9.2 Run the fine CA for the equivalent of one coarse step.

9.3 Map the output back to the coarse level.

9.4 Check which outputs are present for the coarse triples (just □, just ■, or □ and ■). If both □ and ■ are present, the mapping is invalid.

9.5 If the mapping is valid, repeat steps 9.1 – 9.4 for the other coarse rule cases. If the mapping is valid for all triples, we have coarse grained the CA.

Once we have done this for all eight rule cases, we know which coarse outputs exist for this mapping and fine rule. And as we have all eight rule cases, we can immediately assemble the coarse rule, or perhaps several possible coarse rules. But we also know that if there are several possible coarse rules (i.e. any rule case has output both a □ and a ■) there must be an inconsistency, so we can immediately eliminate this rule / mapping combination. Thus we avoid the exponential entirely: we either have just one answer, or we have an invalid coarse graining.

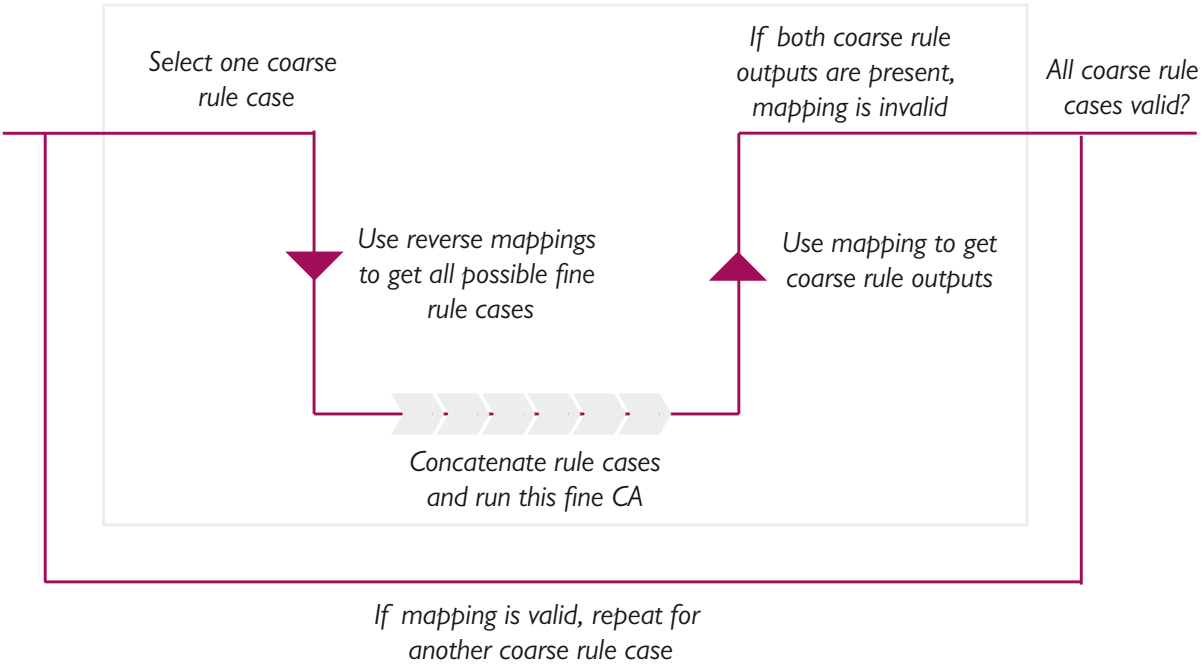


Figure 10.32 Steps for more efficient all mappings.

10.20.9 Checking all mappings even more efficiently

An equivalent but (even) more efficient technique is to use the same mapping relation for all rule cases at once and then perform an intersection on these rules. This means that we now have to check just three or four alternatives at $g = 2$. Since the rule cases are independent, and since we only need to detect an inconsistency, this is equivalent to the procedure we have just described: though done in a different order, we still check each possible option for individual rule bits and see if it is consistent.

10.20.10 Even more efficient all mappings steps

9.1 Start with a candidate mapping as before, then generate the fine possible equivalents (using one possible mapping) for all eight coarse rule cases (for instance □■□). Concatenate these fine rule equivalents to make the fine CA.

9.2 Run the fine CA for the equivalent of one coarse step.

9.3 Map the output back to the coarse level and record the outputs (□ or ■) at the coarse level for each coarse triple, giving us a candidate rule.

9.4 Repeat steps 9.1 – 9.3 for the other possible mappings. If we get the same candidate rule from each mapping, we have a valid mapping and coarse graining.

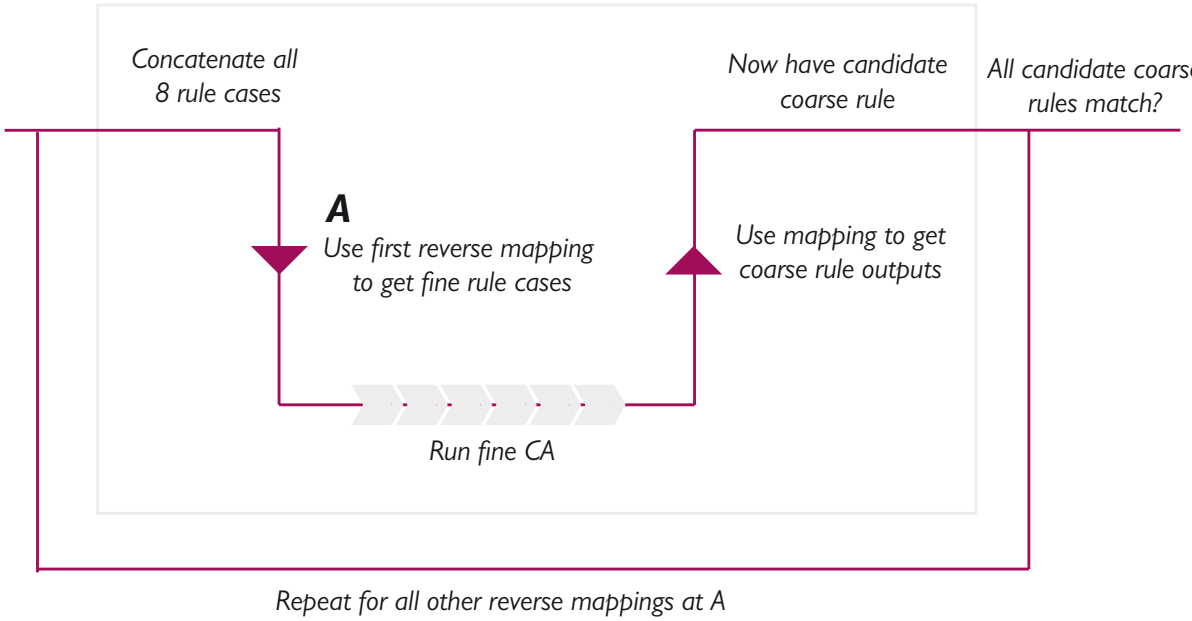


Figure 10.33 Steps for even more efficient all mappings.

10.21 Taking the union of all mappings

Though intersecting the partial coarse graining sets generally restricts the results to higher quality answers, the rejected solutions may still be useful in some circumstances. Later, in §12.5, we shall only consider one fine rule for coarse graining and we shall wish to consider the union of all possible answers instead. In this case we return all rules given by all mappings, even if they only appear in a single test case.

Returning the union of all results adversely affects algorithm performance. For the naïve exponential algorithm, we can no longer stop when find an inconsistency and must examine all possible combinations of mappings, which significantly increases the runtime.

In the more efficient algorithm, we must now calculate all combinations of mappings. This reintroduces the exponential, albeit over the much smaller set of mappings. In this case, we have a maximum of 256 alternates to calculate (2^8), and typically just 16 (2^4). In the final step, we now collate the union of all possible combinations of the coarse rule values we generated in the previous stage. Note that the more efficient technique described in §10.20.9 is not equivalent under union and returns far fewer results.

10.21.1 Comparison of union results

Initial condition	Coarse grainings	Percentage of total coarse grainings
□■□■□■□■□■	17,592	9773%
□■■□■	16,397	9109%
□■■□■■	15,955	8864%
□■■□■□■□	15,230	8461%
□■■□■□■□■■□□	9671	5373%
Total	180	100%

Figure 10.34 The union of valid coarse grainings at $g = 2$ for the initial conditions in Figure 10.29.

Initial condition	Coarse grainings	Percentage of total coarse grainings
□■□■□■□■□■	3472	1929%
□■■□■	3100	1722%
□■■□■■	2958	1643%
□■■□■□■□	2812	1562%
□■■□■□■□■■□□	1881	1045%
Total	180	100%

Figure 10.35 The union of valid coarse grainings at $g = 2$ for the initial conditions in Figure 10.29, showing non-equivalence of the more efficient technique in §10.20.9 under union.

If instead we use partial mappings from the partial coarse graining, we maintain a constant mapping for those rule cases at all times and cycle through the unknown ones as described above.

10.22 Using known rule cases

The algorithms above ignore the rule cases we know from the initial partial coarse graining steps, relying entirely on intersection to constrain the results returned. (Though we do use the partial coarse graining to perform a consistency check on those values initially before reaching the algorithm stage discussed in this section.) While perhaps an impressive feat, we would clearly like to use these existing partial results as we know them to be correct.

One approach is to fix the known rule cases when testing for consistency across multiple mappings. Unfortunately this makes passing the intersection test easier: by keeping certain parts of the string constant for all tests, we now allow through cases where these values would have varied before. This means it may become easier for a rule to be valid with a more complex test string (which gives more known values) than a simple one, and we in fact see this in the results. Clearly this situation is less than ideal.

We can avoid this by performing a double intersection. First we use one of the algorithms outlined above (varying all rule cases) to get an initial set of valid rules, then we intersect these results with our partial rule to remove any rules that disagree with the partial information we have. This approach eliminates a number of extra candidate coarse grainings. Unsurprisingly it removes more rules when using a longer test string, though the number of rules discarded reduces again as we ap-

proach the complete test string. (The complete test string will of course have eliminated all invalid coarse grainings earlier in the process.) The simple string □■□■□■□■□■ allows through all rules returned by the intersection, but □■■□■ removes an additional 36 (taking valid rules from 254 to 218) and □■■□■□■□■■□□ removes an extra 10 (from 210 to 200 valid rules).

10.22.1 Even more efficient all mappings plus known rule cases steps

9.1 Start with a candidate mapping as before, then generate the fine possible equivalents (using one possible mapping) for the eight coarse rule cases (for instance □■□). Concatenate these fine rule equivalents to make the fine CA.

9.2 Run the fine CA for the equivalent of one coarse step.

9.3 Map the output back to the coarse level and record the outputs (□ or ■) at the coarse level for each coarse triple, giving us a candidate rule.

9.4 Repeat steps 9.1 – 9.3 for the other possible mappings.

9.5 If we get the same candidate rule from each mapping, check that each rule case's output in the candidate rule matches the partial rule from step 8 (for the cases where we have an output).

9.6 If the candidate rule and partial rule from step 8 are consistent, we have a valid mapping and coarse graining.

10.22.2 Known rule cases results

Initial condition	Coarse grainings	Percentage of total coarse grainings
□■□■□■□■□■	278	154%
□■■□■	218	121%
□■■□■■	218	121%
□■■□■□■□	203	113%
□■■□■□■□■■□□	200	111%
Total	180	100%

Figure 10.36 The number of valid coarse grainings at $g = 2$ using known rule cases for the initial conditions in Figure 10.29. The results show a small but significant reduction in valid coarse grainings compared to Figure 10.30.

10.23 Key points

- Coarse graining can be used to model the emergent behaviour of CAs and predict their future states. As it models rules, rather than individual examples of CAs, it can capture CA behaviour much more generally.
- We see subjective emergence through different coarse grainings that model different aspects of a rule's underlying behaviour.

- Graining graphs can be used to show the relationships between coarse grainings under certain criteria.
- Partial coarse graining allows some discrepancies between the fine and coarse CAs, and can allow the coarse CA to model behaviour it would not otherwise be able to capture.
- Selecting a good mapping can be as crucial as choosing the right rule when coarse graining. Checking that potential partial coarse grainings are valid for all mappings greatly increases the quality of results returned.

11 EMERGENCE AND INFORMATION

Following on from quantitative emergence (§9.11) and neutral emergence (§9.18), we show how we can model similarities between fine and coarse CAs (the low and emergent level models) using information theory. After describing how to calculate the MI in a coarse graining, we see that the MI of partial coarse grainings can be higher than that of total coarse grainings. We also discuss the types of coarse grainings that have high and low MIs.

Emergent models appear independent from their underlying behaviour because of the apparent discontinuity between the high and low levels (§9.6). This discontinuity also allows us to predict the behaviour of the high level system: we can calculate the system's future state more efficiently than by running the low level system. In other words, we end up with a more compact high level representation of the low level system, or at least of some aspect of its behaviour. The emergent model stores and transfers less information (in general) than the low level system from which it emerges.

It is important to distinguish between the high level system and our view (our mental model) of that high level system. Notwithstanding emergent phenomena (§8.4), the high level system is entirely dependent on and derived from the low level system. In contrast, our view is entirely independent of the low level system and operates through its own rules.

We may think of a flock of birds as one object with one position and velocity, even though it is actually composed of many birds' positions and velocities. And we can use this view to predict the flock's future position reasonably efficiently and accurately, first synchronising our view with the real flock and then modelling with our own rules. Of course we may forecast incorrectly if, for example, the flock encounters an obstacle and splits, but that doesn't stop the model from being very useful to us most of the time, or from seeming very real.

A coarse grained CA is another such model: the coarse CA captures the salient emergent properties of the low level fine CA. A mapping is used to translate the fine rule's input condition to initialise the coarse rule, but after that they operate independently, using their own rulesets to calculate their next states. We have a compact representation of some aspect of the fine rule's behaviour, requiring only 25% of the calculations at $g = 2$ to reach any future state.[1]

11.1 Types of information loss

Sometimes the fine detail of the original rule disappears when coarse graining. This can be seen as a loss of irrelevant degrees of freedom (DOF) – though we lose information within each coarse cell, we don't lose anything that, at the fine level, extends beyond this coarse cell. In other cases, the high level rule may be a CA rule of lower complexity than the rule at level L: "the system (the update rule, not the cell lattice) does not contain enough information to be complex at large scales" [7].

1 Assuming we calculate the next states naïvely. Many rules, of course, can be modelled more efficiently.

This is a loss of relevant DOF and information propagated over time by the fine CA cannot always be modelled at the coarse level, as all fine level states that represent this information map onto a single coarse state.

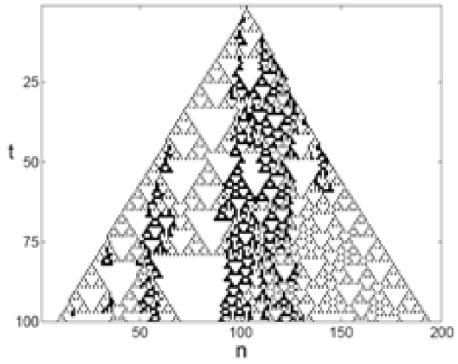


Figure 11.1 The difference (modulo 2) in the trajectories resulting from replacing a □■□ segment in the initial condition with □■■. (From [123])

Israeli and Goldenfeld [123] illustrate such a loss with the coarse graining of rule 146 to rule 128 (Figure 11.1). Because rule 128 is a simpler rule than 146, relevant DOF must be lost. Modifying the initial condition by replacing □■□ by □■■ gives different behaviour unbounded in space and time. Since both □■□ and □■■ map to □ at the coarse level, these discrepancies cannot be modelled. In other cases, coarse grainings seem to highlight some of the underlying structure (for example, various propagating 'signals') by smoothing out other 'irrelevant' structure.

Israeli & Goldenfield elaborate on the general case

> *"Let us illustrate the difference between coarse-graining of relevant and irrelevant DOF. Consider a dynamical system whose initial condition is in the vicinity of two limit cycles. Depending on the initial condition, the system will flow to one of the two cycles. Coarse-graining of irrelevant DOF can project all the initial conditions on to two possible long time behaviors. Now consider a system which is chaotic with two strange attractors. Coarse-graining irrelevant DOF is inappropriate because the dynamics is sensitive to small changes in the initial conditions. Coarse-graining of relevant DOF is appropriate, however. The resulting coarse-grained system will distinguish between trajectories that circle the first or second attractor, but will be insensitive to the details of those trajectories." [123]*

11.2 Coarse graining and mutual information

We saw in §10.11 that several different coarse grainings exist for many CAs, and argued in §10.16 that total coarse grainings, though accurate, are not necessarily good if little of the fine rule's behaviour is retained.[2] (We lost too many degrees of freedom.) We have also seen that partial coarse

2 We assume here that we wish to retain at least some substantial aspects of the fine rule's behaviour.

graining appears to give better results – high level models that capture more of the underlying behaviour – in a substantial number of cases. But it is difficult to say whether one coarse graining is actually better than another, or to compare them more objectively than seeing if they look roughly right.

Mutual information (MI – §7.7) is one way we can qualify this goodness: we suggest that it is a good indicator of the quality of a coarse graining, and it can be used to direct the search for better solutions. Intuitively, mutual information seems like a useful metric as it measures similarities (similar distributions of values) between two models. It is also a well understood and broadly accepted measure. There are also specific reasons why mutual information appears well-suited to this problem.

- Coarse graining is a simple example of the emergent model described in §9.17, with the coarse CA representing the emergent behaviour (or specification) and the fine CA the low level behaviour (or implementation). We saw there how MI was used to determine the quality of information theoretic protolife and emergent models.
- Mutual information is a quantitative measure, which makes it a candidate for guiding the automatic development of emergent systems.
- Mutual information is fairly simple to calculate, making it suitable for practical applications.

11.3 How to calculate the MI

Before we can calculate the mutual information between two systems, information theory requires us to have a language through which to interpret the systems, and we also need to decide how to divide up the systems' state spaces to determine their entropies (§7.9). As both are elementary CAs, they already share a common language (binary cells), but the different spatial and temporal scales of the fine and coarse levels requires a little consideration.

We must be able to say whether two states of a system are the same or not, and we must know the possible states in which a system can be. It seems appropriate to divide the state space into all the possible coarse rule states, splitting the coarse CA into blocks of three coarse cells and yielding eight different states. Six fine cells map onto these blocks at a grain of two, so each block of fine cells can be in one of 64 states (and potentially have a higher entropy). (We explore different block sizes in §12.17, though we find a coarse block size of three to be one of the best for our purposes in this and the following chapters.)

It is less clear how to divide up the state space over time. There are three possibilities

- Expand the state space to include a temporal dimension. At $g = 2$, each chunk in the fine CA state space would now comprise twelve cells (six cells spatially by two temporally).
- Include each coarse CA row several times, once for each fine row to which it corresponds.
- Only look at the fine CA when it coincides with the top of a coarse row. For $g = 2$, ignore every other row.

Recounting coarse rows or expanding the state space seem like the better solutions – each coarse cell replaces several fine cells in the mapping, so surely we should take account of them all? But, because of the way a mapping works, it turns out that we shouldn't.

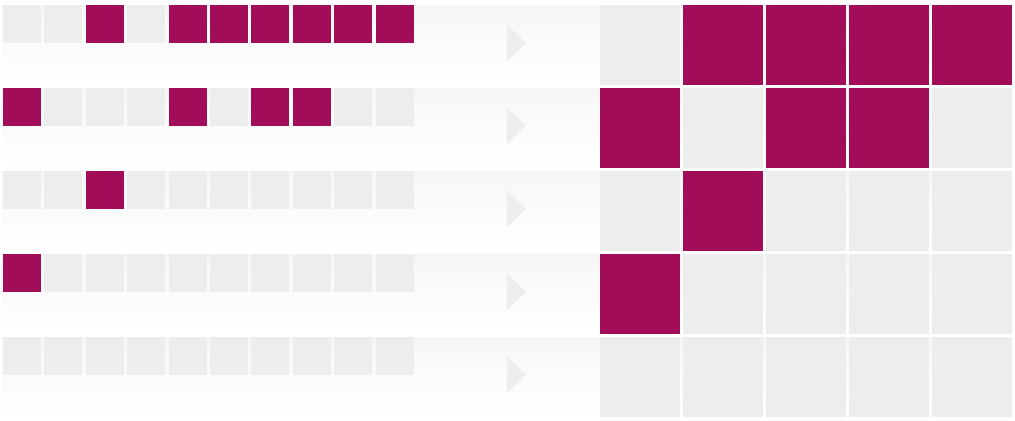


Figure 11.2 A coarse graining of rule 130 to rule 162, showing the fine cells used and not used in the mapping. The mapping used is □□■■.

At grain two, a mapping links six fine input cells to three coarse input cells and, later on, links two output cells to one coarse cell. It does not say anything about the intervening row of fine cells and, in fact, they do not generally correspond. So it does not make sense to measure the MI including these intermediate rows.

In some cases we see a significant drop in the MI if we include all fine rows in our calculations: the CAs match well at every other row as expected, but the others – where the fine CA is at an intermediate stage – are much poorer. In other cases the MI remains roughly the same when we include all fine rows, though this is merely a serendipitous artefact of the CAs' structure. Rule 204, for example, draws vertical lines and has the same cell pattern in all rows and is unaffected by how we measure its MI.

11.4 Calculating the MI

We now describe how to calculate the mutual information between fine and coarse CAs.

1 Decide on the coarse graining rule pairs to test and select a test string for determining the MI.

2 Create coarse and fine CAs from the test input string. The coarse CA should be calculated from the fine CA via the mapping as usual.

3 Run the CAs until the time at which we want to measure the MI. Following §11.3, this must be a time at which the fine CA is aligned with the top of a coarse row. We may choose to measure the MI at several times throughout the run.

4 Calculate the MI between the rules.

4.1 Split the fine CA's states at the current time into blocks of size $3g$.

4.2 Count the number of blocks in each state. For example, if the first three blocks are □■■■□■, □■■□□□ and □■■■□■, we have two blocks in state □■■■□■ and one block in state □■■□□□.

4.3 Split the coarse CA into blocks of size three.

4.4 Count the number of blocks in each state.

4.5 Use the fine and coarse block counts to obtain the entropy for each CA and calculate the MI between the rules.

Fine CA entropy $H_f = \sum p(x_f) \log_2 p(x_f)$

Coarse CA entropy $H_c = \sum p(x_c) \log_2 p(x_c)$

Joint entropy between fine and coarse CAs $H_{cf} = \sum p(x_{cf}) \log_2 p(x_{cf})$

$MI = H_f + H_c - H_{cf}$

5 Run the CAs until the next time at which we want to calculate the MI. Repeat for additional MI values at later times.

11.5 Calculating the MI example

Here we calculate the MI between the fine rule 34 and coarse rule 170 at a grain of two. One valid mapping (§10.20) between these two rules is □□■□. We found this mapping through coarse graining rule 34 as described in §10.10.

We shall use the test string □□□■□■■□■■■□□■■□■■■□□□□□ to calculate the MI between rules 34 and 170.

We construct a fine CA using rule 34 and a coarse CA with rule 170. The initial conditions for the CAs are □□□■□■■□■■■□□■■□■■■□□□□□ for the fine CA (the test string) and □□□■□■□■□■□□ for the coarse CA, calculated via the mapping □□■□.

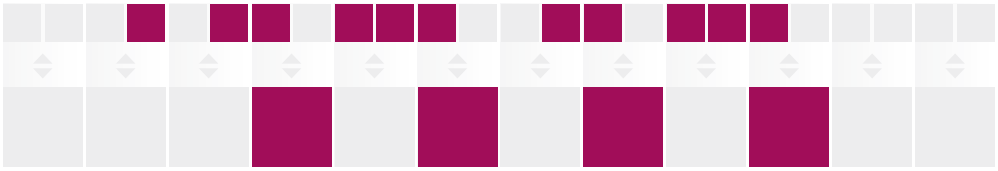


Figure 11.3 The fine and coarse CA input test strings (coarse string calculated via the mapping).

We decide to calculate the MI at fine rule timestep 5 (equivalent to 3 coarse timesteps).[3]

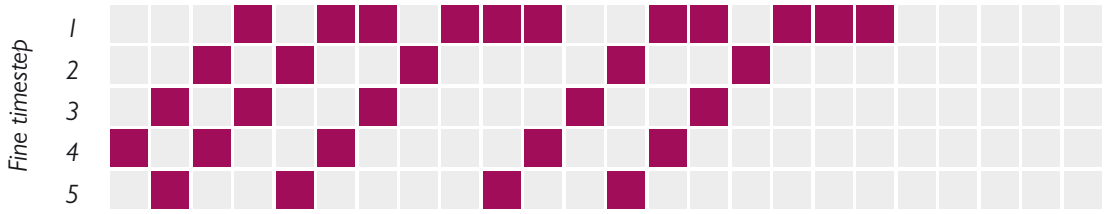


Figure 11.4 The fine CA, after running for 5 (fine) timesteps.

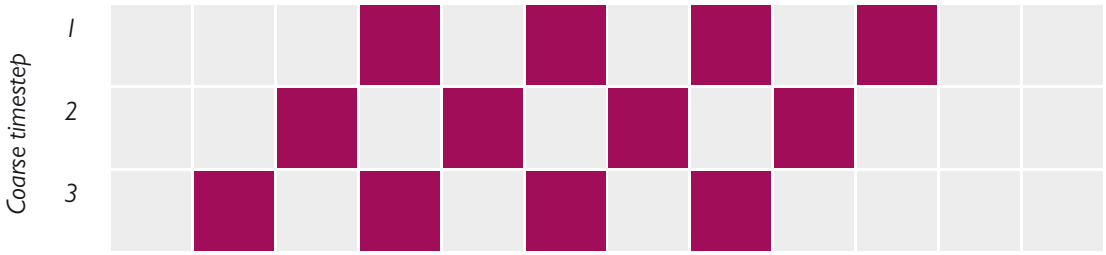


Figure 11.5 The coarse CA, after running for 3 (coarse) timesteps.

Split the coarse CA's state at timestep 3 into blocks of size 3. Do the same for the fine CA's state at the timestep 5 and with blocks of size $3g = 6$.

Figure 11.6 The fine and coarse CAs' outputs, split into blocks.

3 We have assumed the cells adjacent to the initial condition given are □ so we can calculate the MI on the full width of the initial condition at step 5. In practice we would want to use a much larger initial condition (§11.6) and would probably arrange the CAs on infinite periodic grids (§2.1).

Count the number of blocks in each state for both CAs.

□□□□□□	1
■□□□□□	1
□□□■□□	1
□■□□■□	1

Figure 11.7 Fine state counts. The binary states seen in the fine CA's blocks at time 5. (States with no instances in the CA have been omitted.)

□□□	1
□■□	2
■□■	1

Figure 11.8 Coarse state counts. The binary states seen in the coarse CA's blocks at time 3 (fine time 5). (States with no instances in the CA have been omitted.)

	□□□	□■□	■□■
□□□□□□	1		
■□□□□□		1	
□□□■□□			1
□■□□■□		1	

Figure 11.9 Joint state counts. The binary states seen in the fine and coarse CAs' blocks at time 5. State (□■□□■□, □■□) indicates that the fine CA was in state □■□□■□ and the coarse CA in state □■□ for that particular block (the first block in Figure 11.6 in this case). States with no instances in the CA have been omitted.

Calculate the entropy for each rule.

$$\text{Fine CA entropy } H_f = \sum p(x_f) \log_2 p(x_f)$$

$$\text{Coarse CA entropy } H_c = \sum p(x_c) \log_2 p(x_c)$$

$$\text{Joint entropy between fine and coarse CAs } H_{cf} = \sum p(x_{cf}) \log_2 p(x_{cf})$$

In this limited example, we have distinct values for all four cases in the fine CA and one duplicate value in the coarse CA, yielding a ¼ probability of being in any one fine or joint state and a ¼, ¼ or ½ probability of being in a coarse state (depending on the state). Inserting values into the formulae gives

$$H_f = -¼ \times \log_2(¼) - ¼ \times \log_2(¼) - ¼ \times \log_2(¼) - ¼ \times \log_2(¼) = 2.0$$

$$H_c = -¼ \times \log_2(¼) - ¼ \times \log_2(¼) - ½ \times \log_2(½) = 1.5$$

$$H_{cf} = -¼ \times \log_2(¼) - ¼ \times \log_2(¼) - ¼ \times \log_2(¼) - ¼ \times \log_2(¼) = 2.0$$

Calculate the mutual information between the rules.

$$MI = H_f + H_c - H_{cf} = 1.5$$

11.6 The significance of 1.5

Mutual information of 1.5 is the number of bits of information shared between the fine and coarse rules. The information content (entropy) in a string of a single state is 0, and indeed we see this when calculating the entropy for rules 0 or 255. Similarly, the mutual information between rules 0 or 255 and any other rule is also 0. The maximum entropy at the coarse level is 3 bits (this is a random string) and 6 bits at the fine level (for $g = 2$). The maximum MI between a fine and coarse rule is limited by the maximum coarse entropy, and so is also 3 bits.

The MI and entropy figures we calculated in our example suggest that there is a reasonable amount of information (relative to the maximum possible, 1.5 bits out of a possible 3) at the coarse level and between the CAs, and that the fine CA is fairly regular (less random and less information-rich, 2 bits out of a possible 6), though it is more varied than the coarse CA. But entropy and information are statistical measures, and the sample we used is too small to give us meaningful results. If we use the larger, 384 cell string described in §11.8, we end up with

$$H_f = 4.19,\ H_c = 2.43,\ H_{cf} = 4.19,\ MI = 2.43$$

This shows that the entropies and MI were underestimated in our first calculation – all are significantly more information-rich with this larger input string – and that the fine CA does actually contain more information than the coarse CA. (See §11.8 for more on choosing test strings.) Also note that, as this is a total coarse graining, the coarse entropy equals the MI. We discuss this next.

11.7 Partial coarse graining and mutual information

Partial coarse grainings often have high (sometimes very high) MIs. We see later, in §11.9, that the MIs of a significant number of partial grainings are higher than that of many total coarse grainings. Despite the mistakes their coarse rules make through novel behaviour, these partial coarse grainings have more in common (at least to the extent that MI is a determinant of similarity) with the underlying behaviour than many total coarse grainings, which are all too often vacuously correct.

Mutual information between two rules will be high if the CAs' behaviours are non-trivial and tightly coupled (they mirror each other closely). Canonical examples of high MI coarse grainings are those that comprise a pair of complex or chaotic rules, though simpler rules from class 1 or 2 may also have a high MI in certain circumstances – for instance some class 2 rules duplicate the state of the previous step, so if we start these with a high MI initial condition we shall keep a high MI throughout the run.

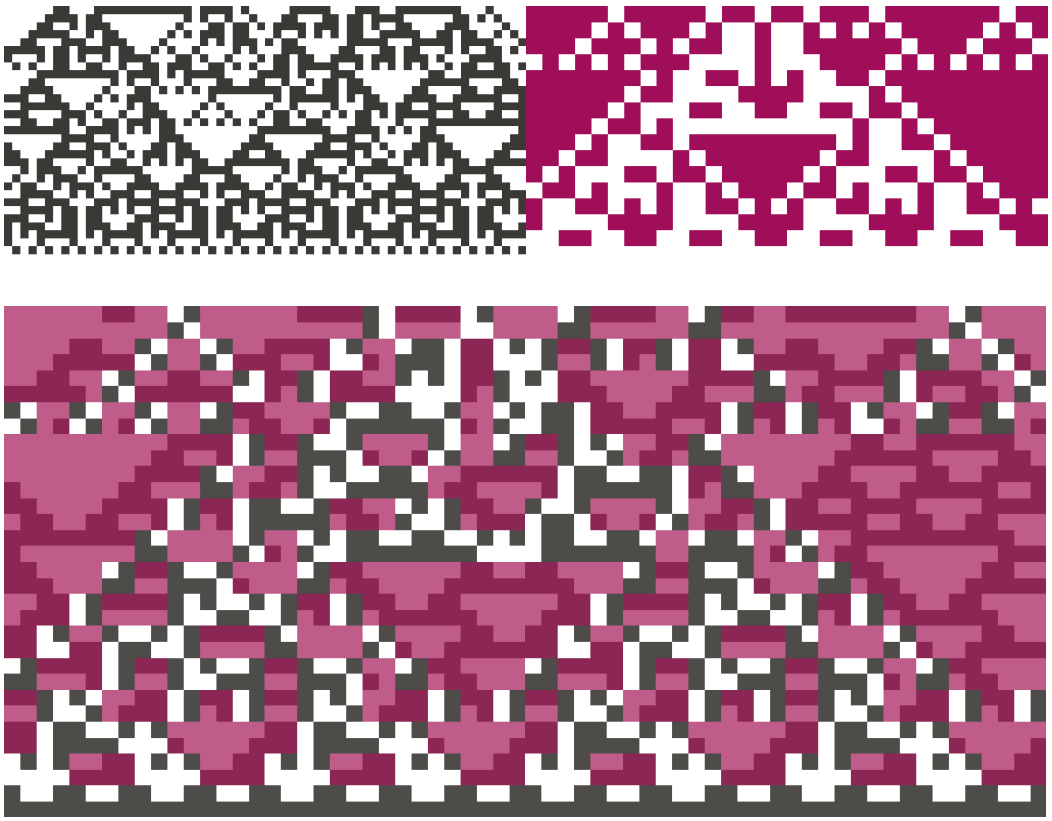


Figure 11.10 Rule 90, a Class 3 (chaotic) rule, coarse grained to rule 165 (also Class 3). This run has a high MI between the rules. See Figure 10.11 for interpretation.

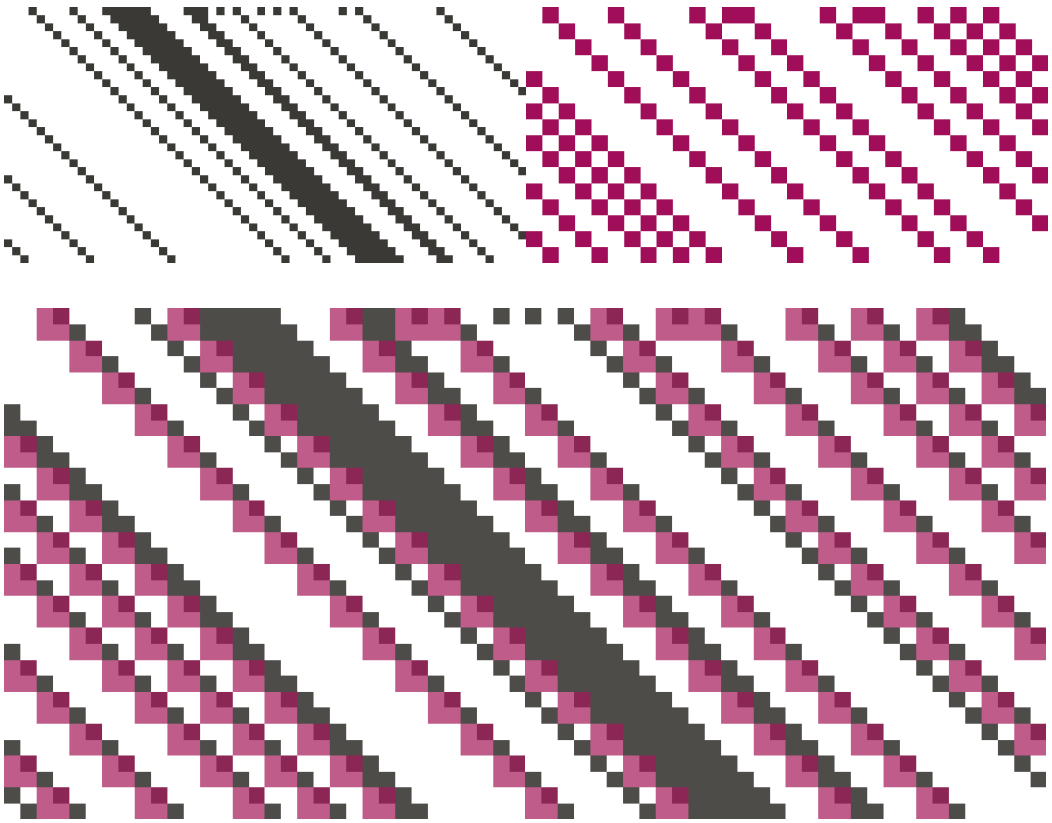


Figure 11.11 This coarse graining of rule 208 to rule 243 also has high MI because it duplicates the high MI initial condition.

The CAs in a total coarse graining must always mirror each other and the MI will always be maximal; this is not true for partial coarse grainings. Note that maximal does not necessarily mean high in some absolute sense, it merely means that the MI must equal the coarse entropy. If a graining has a simple coarse rule, the MI will be much lower than the maximum of 3 bits.

Having a low MI does not necessarily indicate that the coarse graining is a poor one. If the fine rule is simple, it is possible for a coarse graining to be accurate (and presumably good) but still have a low MI. However, if a non-trivial fine rule is totally coarse grained to a simple coarse rule, it is likely that another (perhaps partial) coarse graining of the rule will have a higher MI. Such a simple pairing would be accurate, but probably not good.[4] It is therefore important to judge the MI relative to the fine rule's entropy. (Though we usually want to find the best coarse graining for a single fine rule, so this point is moot.)

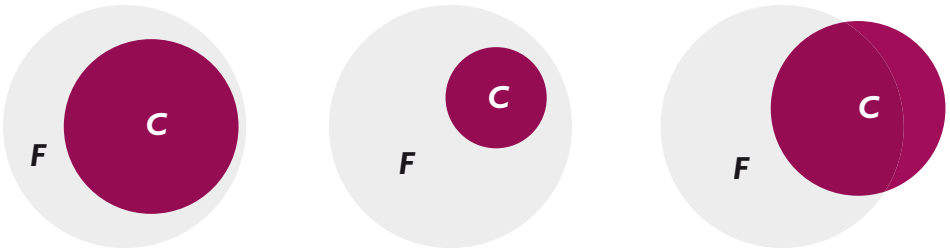


Figure 11.12 Representation of three different coarse grainings. The circles indicate the amount of entropy: the large ovals represent the entropy in the fine CA F; the smaller ovals the entropy in the corresponding coarse CA C; the overlap represents the mutual information. The left figure shows a total coarse graining with high MI; the middle a total coarse graining with low MI; the right a partial coarse graining with high MI.

11.8 Choosing MI test strings

Mutual information is a statistical measure, so we need a reasonably substantial data set before we can be confident in our inferences. We initially generated 100 random strings, each 1000 characters long with P(0.5) of each digit being a ■ to use as CA starting states, giving us a confidence interval of ±1.05% at 99% certainty for six cell (for $g = 2$) blocks. (We performed the same experiments for 100 strings with P(0.2) and P(0.8) of each digit being a ■. These results are in §B.)

Testing 100 strings of 1000 characters takes a significant amount of time. However we found we could approximate this entire set of strings with just one string: the 384 digit concatenation of all input fine states used to check the validity of total coarse grainings at $g = 2$. While there is nothing magical about this sequence of digits, the string does have some nice properties: it has an equal number of ■s and □s; and it covers every possible input condition.

In experimental runs, the 384 character string gives very similar results to the 100 random strings, albeit with more variation in the MI. Figure 11.16 and Figure 11.17, discussed next, show that the

4 We explore what it means to be good further in §12.21.

single string results are less smooth, with a few large steps, but that the overall shape of both graphs is very similar. This will be useful when we use MI to find good coarse grainings, as being able to use just one small string to approximate many large ones makes the process much more efficient.

The initial condition chosen will inevitably have an impact on the MI results from later times in a CA's run, but this influence can vary considerably from rule to rule. For instance rule 204 just duplicates the result of the previous state. This isn't particularly complex behaviour, but, with an input condition such as a random string of digits, it has a high entropy.[5] For other rules, such as 128, which draws triangles that dissipate over time, the effect of the initial condition on the entropy is much smaller.

Figure 11.13 Rule 204's output (left) duplicates the input, so its MI is very dependent on the initial condition. In contrast, rule 128 (right) dissipates quite quickly and is much less sensitive to the run's starting state.

Figure 11.14 Rule 204

Figure 11.15 Rule 128

5 That the rule is simple and yet has a high entropy is not a problem for using MI as a measure of goodness. While we have equated (for now at least) goodness with a high MI, a coarse graining to 204 must also contain behaviour that matches rule 204 at the low level – if it does not, the MI will be low, no matter what rule 204's entropy is.

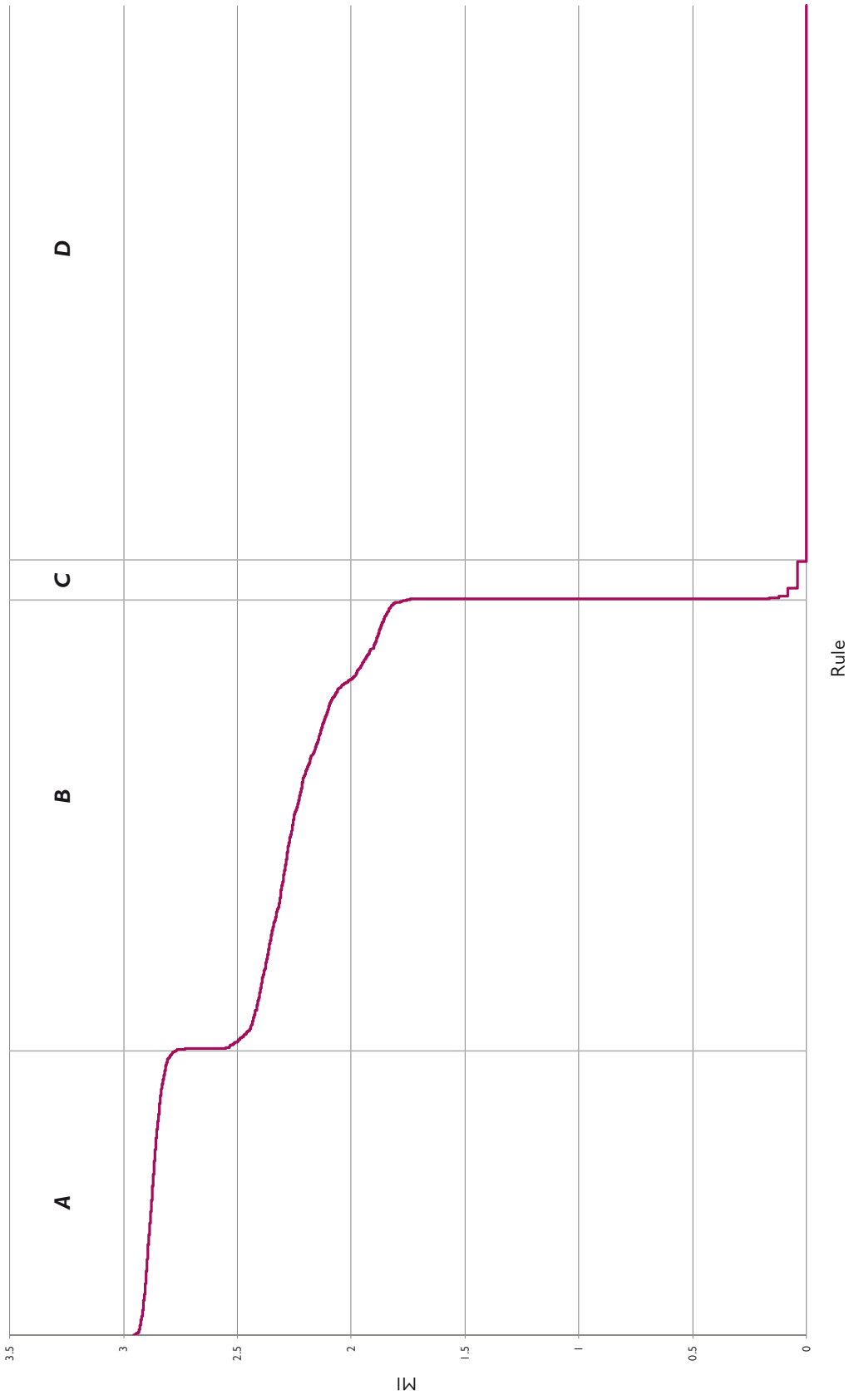


Figure 11.16 32,000 MI results obtained at timestep 9 from totally coarse graining 100 random P(0.5) binary strings 1000 digits long. The graph is sorted by MI. See §11.8 for details.

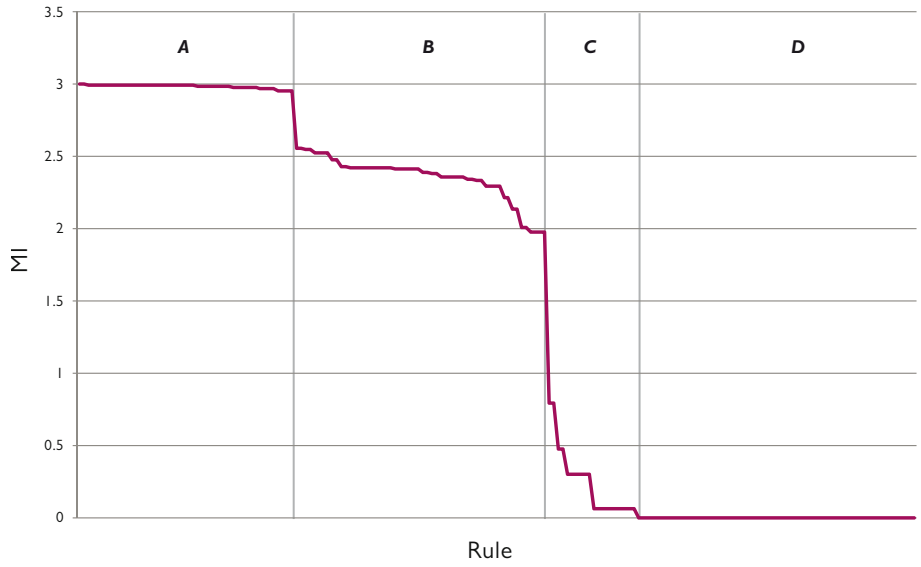


Figure 11.17 MI results from the 390 total coarse grainings at timestep 9 obtained using the 384 digit string described in this section. The graph is sorted by MI. See §11.8 for details.

Figure 11.16 shows the MI of 32,000 total coarse grainings over all rules at $g = 2$, sorted by their MI at time 9. The MIs were obtained from 100 random binary strings P(0.5) 1000 digits long. (The graph is restricted to 32,000 of 100,000 results due to limitations in the graphing software; the data set was sampled representatively.) Grainings with a high MI at that time (close to the maximum of 3) are found on the left of the graph, while those with 0 MI are on the right. Figure 11.17 shows all 390 total coarse grainings and mappings at $g = 2$ sorted by MI at time 9. The MIs were obtained using just the 384 digit concatenation of fine input states from §11.8.

Note that coarse grainings appear multiple times in Figure 11.16, once for each test string. The different initial conditions mean that there will be a spread of MIs for each coarse graining, and the same pair of rules will appear in several locations on the graph. This also means that the results from one particular coarse graining will be interspersed with other coarse grainings with similar MIs (a common occurrence) – see §11.9 and Figure 11.21.

Figure 11.16 (and Figure 11.17) appears to have several steps. At first the MI remains almost maximal at 3 bits in section A, before dropping sharply. Then the graph descends gradually during section B until it reaches another steep drop, before finally progressing relatively quickly towards 0 MI (section C leading to section D).

As expected from the graining graphs in §10.19, which show many rules clustered around the rule 0/255 nodes, a significant number of coarse grainings have low mutual information: approximately a quarter of the grainings have trivial or no MI. (These are predominantly rules that coarse grained to rules 0 or 255.) Rules that have very high MIs are mainly in Wolfram's Class 3 (such as rules 102 and 153 – §2.7), though some simpler Class 2 rules are also found in this section (for instance rule 204; Figure 11.13) because they propagate high MI initial conditions.

The long slope in the middle of the graph (section B) mainly comprises Class 2 rules. The change in MI along the slope is largely due to the initial condition and not the rules: some graining pairs are more common at the top of section B than the bottom by a factor of approximately two, but all grainings present in section B are found at all locations throughout the section in significant numbers. Rule 204 also reappears in section B, presumably the result of lower MI initial conditions.

Most of the rules after the drop to section C are Class 2 rules that discard much of the initial state before settling on a simple repeating pattern and a few from Class 1 that have not died out by that point. (Rule 128, for example, has no MI in this experiment as we chose to measure the MI at time 9.[6])

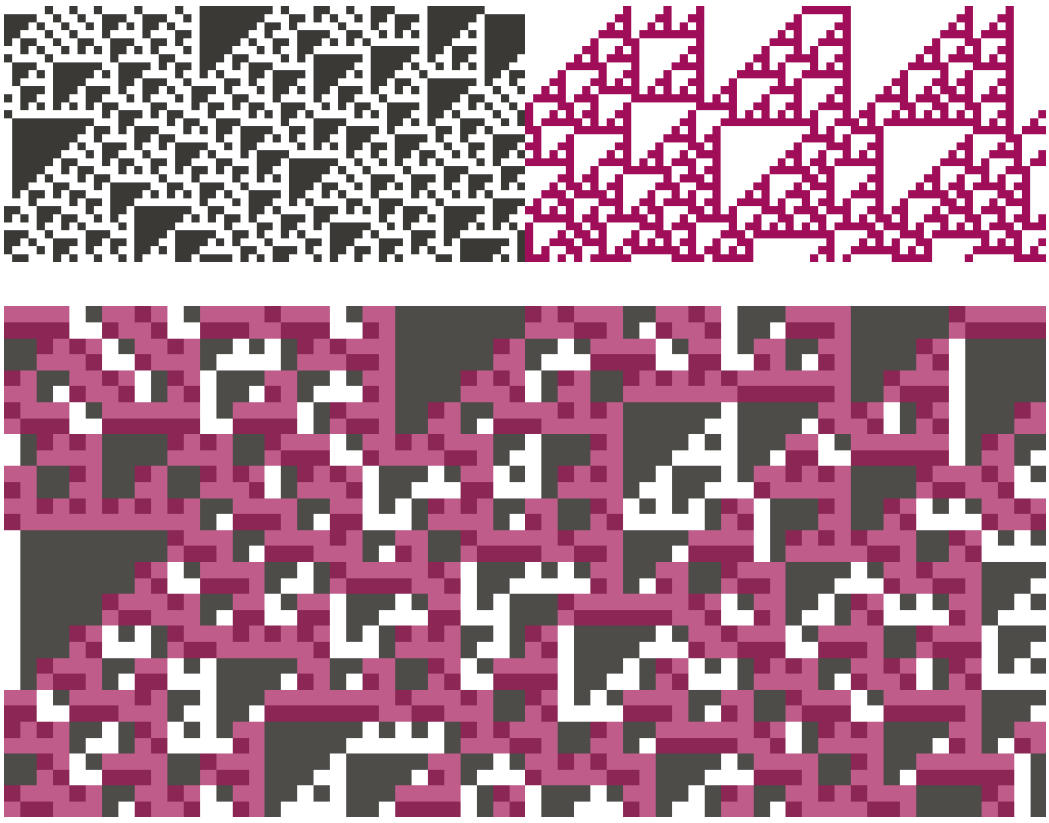


Figure 11.18 An example of a coarse graining with Class 3 rules found in section A. The picture shows 153 coarse grained to 102.

6 For rule 128 to give a non-zero MI here, its initial condition would have to contain a sequence of at least 19 ■s – preferably significantly longer than that and preferably several of these sequences – something that is pretty unlikely in a 1000 character random string.

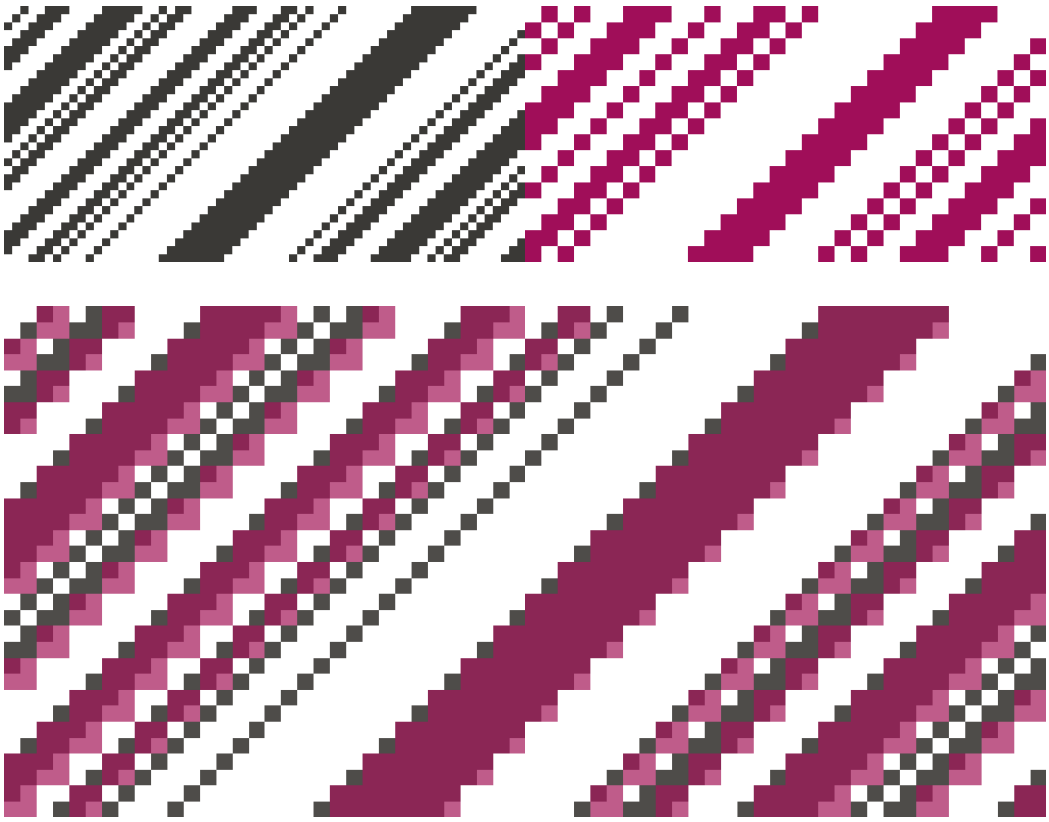


Figure 11.19 A coarse graining of Class 2 rules, typical of those in section B. Here rule 170 is coarse grained to itself.

Figure 11.20 An example of a rule found in section C section of Figure 11.16. Here we see the Class 2 rule 140 coarse grained to Class 1 136.

11.9 Mutual information of different coarse grainings

We have already seen that we get slightly different MI values from the different initial conditions used in each of the 100 runs. However, the distribution of each coarse graining in the MI graph Figure 11.16 is not uniform: for some grainings, the initial condition has relatively little impact on its MI at time 9 and gives tightly clustered results, whereas for others it is quite significant and the results are spread over a wider MI range.

Figure 11.21 shows the prevalence of each coarse graining as the MI decreases in Figure 11.16, with corresponding sections A-D from the graph marked. The width of each series shows the relative incidence of a coarse graining out of all coarse grainings present at that point. Note that all mappings for each coarse graining are included in the graph, so a coarse graining with three valid mappings will have three times as many results overall as a coarse graining with just one mapping.

Including all coarse grainings results in a very busy graph, so we remove coarse grainings that return identical or similar results to others to show the prevalence of coarse grainings in Figure 11.16 more clearly.[7] For a graph with all coarse grainings, see §C.

- A few high MI rules (e.g. 60 → 60 (Figure 11.22), 238 → 136) dominate initially in section A before a large spike in popularity of 51 → 204 as we reach the end of the section.
- Then there is another period of relative quiescence, dominated by rules such as 51 → 204 (Figure 11.23) and 170 → 170 as we follow the MI descent in section B.
- In section C, we switch to a completely different set of low MI rules for a short period. These include 160 → 128 (Figure 11.24) and 128 → 128.
- Finally, the 0 MI rules (represented here by 0 → 0) take over completely in section D.

7 This also impacts the relative sizes of sections A-D, though they remain similar.

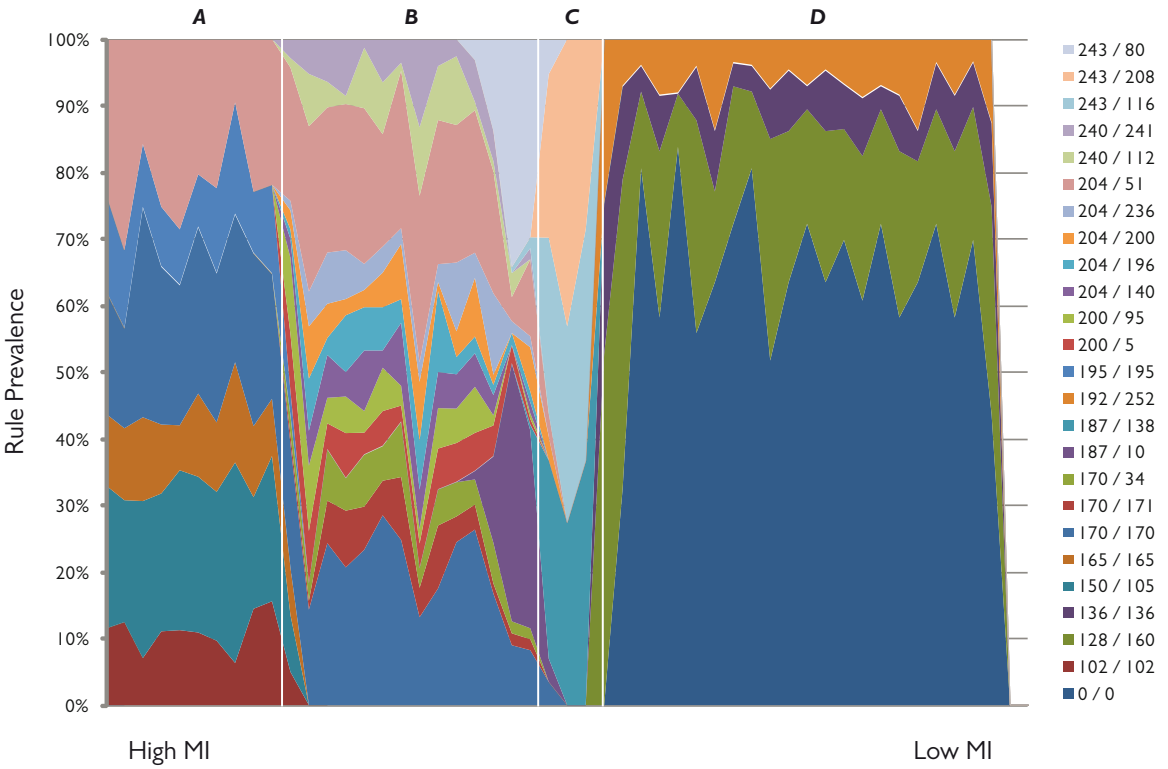


Figure 11.21 The distribution of each coarse graining, sorted by MI, using the same data as Figure 11.16. A coarse graining on the left of the graph has high MI and one on the left has low (or zero) MI. The spread of each coarse graining horizontally indicates the variation in its MI results, while the spread vertically shows its (relative) prevalence at that MI value. Sections A-D match those in Figure 11.16. See §11.9 for discussion.

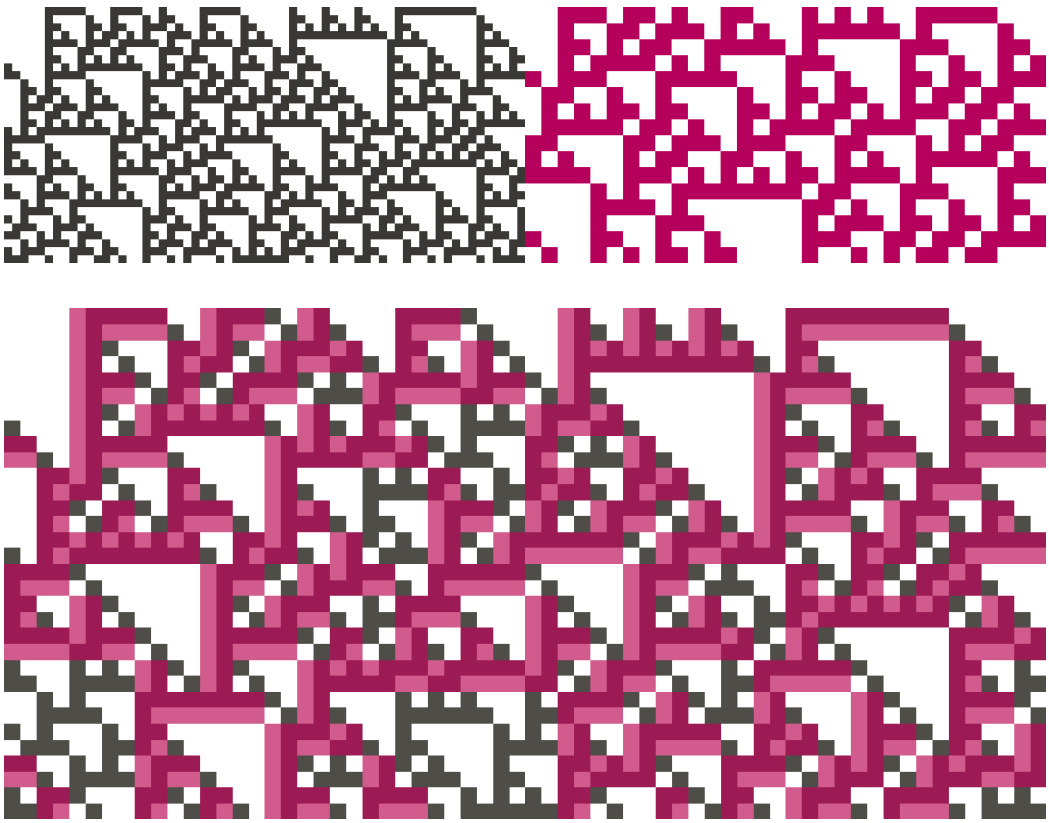


Figure 11.22 Rule 60 coarse grained to rule 60.

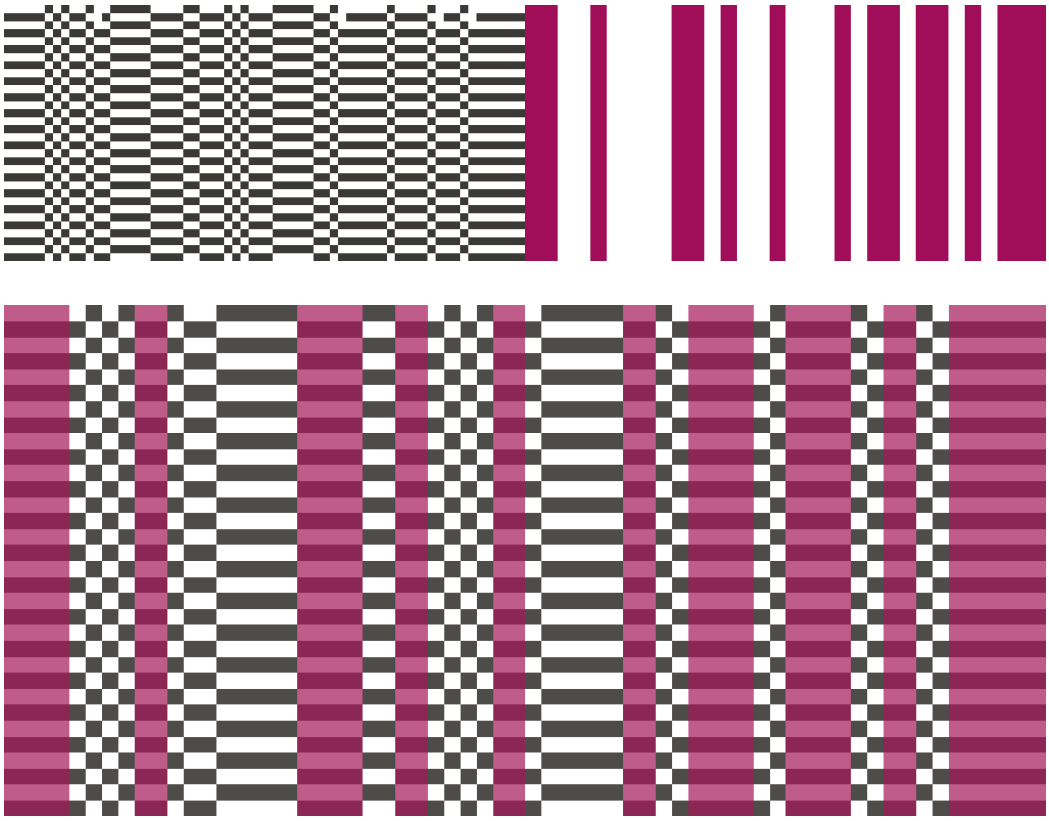


Figure 11.23 Rule 51 coarse grained to rule 204.

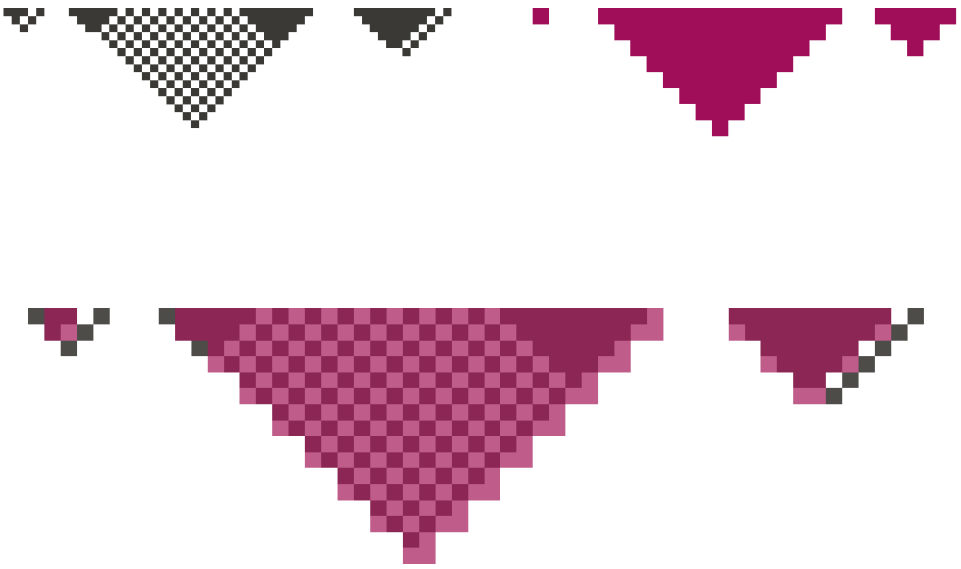


Figure 11.24 Rule 160 coarse grained to rule 128.

The regions in Figure 11.16 are also linked to mapping symmetry: in the first step (section A), the mappings all have an even number of ■s and □s, while on the main section of the slope (in section B) the mappings are odd. We see a similar pattern in the small bumps at the end of this descent, switching first to even mappings, then back to odd again before the steep drop at the start of section C. It is unclear why rules with even and odd mappings are grouped together like this.

The MI graph for partial coarse grainings (Figure 11.26) has a gentler slope down than the total graph (Figure 11.16). (Of course this graph includes all of the total results as well.) It turns out that the partial and total coarse grainings remain largely separate from each other when sorted by MI.

- The flat section A is completely composed of total rules (this is also section A in the previous graph).
- The first half of section B is 95% total rules, before switching to become 93% partial.
- The small section C before reaching zero MI is approximately 50% partial and total.
- The zero MI section D is 66% total rules.

With these divisions in mind, we can see sections in the partial graph that exactly mirror parts of the total graph.

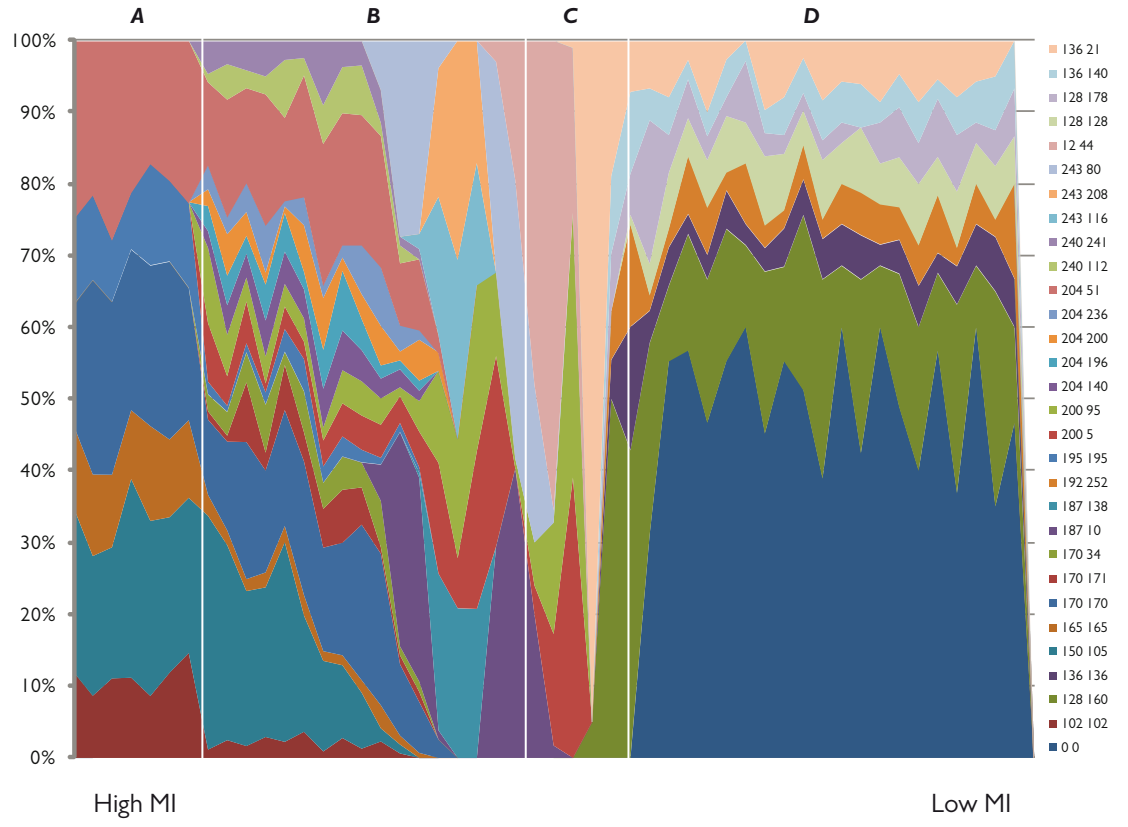


Figure 11.25 The distribution of each coarse graining, sorted by MI, using the data from Figure 11.26. As with Figure 11.21, a coarse graining on the left of the graph has high MI and one on the left has low (or zero) MI. Sections A-D match those in Figure 11.26.

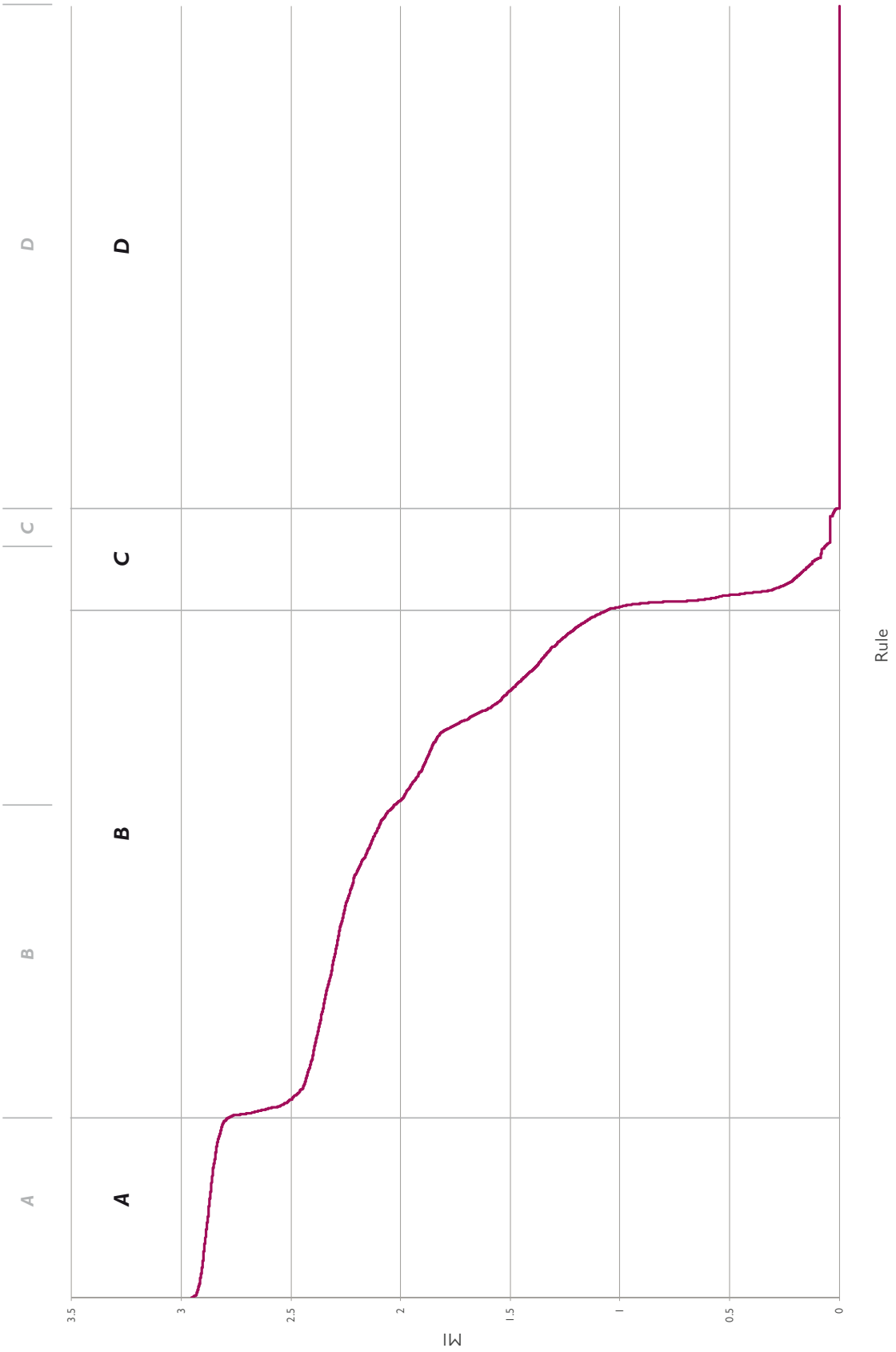


Figure 11.26 32,000 MI results obtained at timestep 9 from partially coarse graining the same random binary strings as Figure 11.16. The graph is sorted by MI. The approximate positions of sections A-D in Figure 11.16 are also shown in grey at the top of the graph (see §11.9).

11.10 Graining graphs and MI

We now revisit the graining graphs we saw in §10.19, exploring how the MI varies throughout the graphs.

Almost all of the nodes on the central spine of the graphs have a high MI between them and their fine or coarse grainings (with obvious exceptions such as rules 0 and 255, which have 0 entropy and thus 0 MI). That the most connected nodes – those that are most useful for coarse graining – are usually those with the highest MI adds weight to the link between mutual information and the goodness of a coarse graining we proposed in §11.2.

The partial coarse grainings have high (sometimes very high) MIs that are higher than the MIs of a lot of grainings that exist – even to the same rule – in the total graining graph. These partial coarse grainings often get closer (at least to the extent that MI is a determinant of closeness[8]) to capturing the underlying fine behaviour than many of their total counterparts, which we have seen are often vacuously correct.

While a number of low MI coarse grainings (e.g. to rule 0) are also added to the graph, partially coarse graining tends to include new links that form part of the central graph structure and have high MIs. Only 41% (40 of 98) of the new partial coarse grainings at $g = 2$ are added to the 0/255 or 128/254 clusters. This contrasts sharply with totally coarse graining to higher grains: moving to $g = 3$ adds 193 new coarse grainings, 162 (83%) of which are joined with low MI only to 0/255 or 128/254; $g = 4$ is similar, with 354 extra coarse grainings, 300 (85%) of which join these two clusters.

Though from the graphs in §11.8 and §11.9, it would be reasonable to conclude that total coarse grainings are better – the first third of Figure 11.26, with the highest MIs, is almost exclusively total. But the middle third of the graph is nearly all partial grainings, coarse grainings that all have higher MIs than nearly half of the total grainings.

In aggregate, partial coarse grainings have slightly lower MIs than total coarse grainings, but there are good and bad examples of both. And the real point is that partial coarse grainings provide new options – including some very valuable ones – that were not available otherwise. We have also seen how partial coarse grainings let us exploit aspects of the problem that are inaccessible when coarse graining totally because of its need for consistency all of the time.

11.11 Predicting good coarse grainings

As coarse graining to higher granularities takes significant time, it would be useful to predict higher g total coarse grainings through partial coarse graining, but unfortunately the partial graph is relatively poor at such predictions. Just 14 of the 98 extra partial coarse grainings found with the string

8 The caveat mentioned in §11.8 about propagating complex patterns applies here too of course.

□■□■□■□■□■□■□■□■□■ at $g = 2$ become total coarse grainings at $g = 3$. 38 of them become total coarse grainings at $g = 4$ (this includes all of the ones at $g = 3$). In contrast, 190 of the 193 extra coarse grainings at $g = 3$ are also valid at $g = 4$.

But partial coarse grainings are better predictors of interesting behaviour. Perhaps because of the less stringent requirements of higher granularities and partial coarse graining, the partial coarse grainings that form part of the new spine can predict good complete coarse grainings at higher granularities. The partial graining graph at $g = 2$ predicts the joining of the 128/254 blob to the main graph. While there are a few other direct forecasts like this, there are substantially more indirect predictions. Eight of the ten indirect links between 128/254 and 0/255 become direct links at $g = 4$. Similarly, a lot (though not all) nodes on the path to 204 from 0/255 are direct links at $g = 4$.

It may be possible to use this approach to find specific coarse grainings at higher grains or in more complex domains where examining all possible combinations quickly becomes intractable.

- Perform a partial coarse graining at a low grain.
- Look for solutions one or more steps away from the fine rule on the graph to form a potential rule pair.
- See if this rule pair is a valid coarse graining at the target grain.

Partial test string 1 □■■□■□■□□■■□■□■□□■■□■□■□

Partial test string 2 □■□■□■□■□■□■□■□■□■

	Coarse Grainings
Total $g = 2$	180
Partial 1 $g = 2$	234
Partial 2 $g = 2$	278
Total $g = 3$	323
Partial 1 $g = 3$	394
Partial 2 $g = 3$	400
Total $g = 4$	498
Partial 1 $g = 4$	556
Partial 2 $g = 4$	556

Figure 11.27 Number of coarse grainings returned at different granularities when totally coarse graining and when partially coarse graining using test strings 1 and 2.

	Partial 1 $g = 2$	**Partial 2 $g = 2$**	**Total $g = 3$**	**Partial 1 $g = 3$**	**Partial 2 $g = 3$**	**Total $g = 4$**	**Partial 1 $g = 4$**	**Partial 2 $g = 4$**
Total $g = 2$	54	98	130	130	130	144	144	144
Partial 1 $g = 2$		44	140	142	142	170	182	182
Partial 2 $g = 2$			144	148	150	182	198	198
Total $g = 3$				71	77	312	312	312
Partial 1 $g = 3$					6	360	371	371
Partial 2 $g = 3$						360	373	373
Total $g = 4$							58	58
Partial 1 $g = 4$								0

Figure 11.28 The number of new coarse grainings found at higher granularities or with less restrictive test strings. Partially coarse graining at $g = 2$ with string 1 finds 54 coarse grainings that are not present when totally coarse graining at $g = 2$.

	Total $g = 3$	**Partial 1 $g = 3$**	**Partial 2 $g = 3$**	**Total $g = 4$**	**Partial 1 $g = 4$**	**Partial 2 $g = 4$**
Partial 1 $g = 2$ (54)	10	12	12	26	38	38
Partial 2 $g = 2$ (98)	14	18	20	38	54	54
Partial 1 $g = 3$ (71)				48	59	59
Partial 2 $g = 3$ (77)				48	61	61

Figure 11.29 The number of coarse grainings predicted at higher granularities. 10 of the 54 extra coarse grainings when partially coarse graining at $g = 2$ with string 1 are also found when totally coarse graining at $g = 3$.

	Total	**0/255**	**128/254**
Total $g = 2$	82 (45%)	54 (30%)	28 (15%)
Partial 1 $g = 2$	110 (47%)	78 (33%)	32 (14%)
Partial 2 $g = 2$	122 (44%)	86 (31%)	36 (13%)
Total $g = 3$	244 (76%)	194 (60%)	50 (16%)
Partial 1 $g = 3$	304 (77%)	254 (64%)	50 (13%)
Partial 2 $g = 3$	312 (78%)	258 (65%)	54 (13%)
Total $g = 4$	382 (77%)	334 (67%)	48 (10%)
Partial 1 $g = 4$	426 (77%)	378 (68%)	48 (9%)
Partial 2 $g = 4$	426 (77%)	378 (68%)	48 (9%)

Figure 11.30 The number of coarse grainings that coarse grain to 0/255 and 128/254. An increasing number of the coarse grainings go to these rules as g rises.

	Mean MI at 4 or 5	Mean MI at 9 or 10
Total *g* = 2	0.99	0.95
Partial 1 *g* = 2	0.82	0.77
Partial 2 *g* = 2	0.82	0.78
Total *g* = 3	0.32	0.29
Partial 1 *g* = 3	0.28	0.26
Partial 2 *g* = 3	0.28	0.26
Total *g* = 4	0.24	0.23
Partial 1 *g* = 4	0.19	0.18
Partial 2 *g* = 4	0.19	0.18

Figure 11.31 The mean MI over all coarse grainings at timestep 4 or 5 (depending on which coincides with a coarse step at that grain) and step 9 or 10. There is a marked drop in mean MI as *g* rises, principally because of the coarse grainings that include 0/255 or 128/254.

	Partial 1 *g* = 2	**Partial 2 *g* = 2**	**Total *g* = 3**	**Partial 1 *g* = 3**	**Partial 2 *g* = 3**	**Total *g* = 4**	**Partial 1 *g* = 4**	**Partial 2 *g* = 4**
Total *g* = 2	0.31/0.24	0.67/0.61	0.23/0.23	0.22/0.22	0.22/0.22	0.13/0.13	0.19/0.18	0.19/0.18
Partial 1 *g* = 2		1.06/1.03	0.23/0.23	0.23/0.23	0.23/0.23	0.13/0.12	0.14/0.14	0.14/0.14
Partial 2 *g* = 2			0.24/0.23	0.23/0.23	0.23/0.23	0.13/0.13	0.15/0.15	0.15/0.15
Total *g* = 3				0.18/0.17	0.20/0.20	0.26/0.26	0.25/0.26	0.25/0.26
Partial 1 *g* = 3					0.49/0.49	0.46/0.45	0.29/0.29	0.29/0.29
Partial 2 *g* = 3						0.46/0.45	0.28/0.28	0.28/0.28
Total *g* = 4							0.23/0.23	0.23/0.23
Partial 1 *g* = 4								-/-

Figure 11.32 The mean MI over the extra coarse grainings in Figure 11.28. The MIs were recorded at timesteps 4 or 5 and 9 or 10. (Shown as time 4 or 5/time 9 or 10.)

11.12 Mappings and MI

We saw in §10.20 that our choice of mapping can significantly affect the usefulness of a partial coarse graining, even for the same pair of fine and coarse rules. Mutual information offers us a way to quantify this difference, and a way to select the best mapping.

We ran a series of experiments to explore the influence of different mappings on MI, starting with the 88 unique elementary CA rules and partially coarse grained them at $g = 2$. There were typically two mappings for each coarse graining, though the number varied from one to four.

We see little difference between most mappings. Clearly the MIs of total coarse grainings must be identical for all mappings (as all mappings are perfect §10.20), but for some partial coarse grainings, such as rule 160 to rule 128, we see substantial differences between the MIs of the mappings. The mapping ■■□■ has an MI of 2.41 at time 5 with the complete test string (§11.8), whereas □■□■ only gives 1.06. The reason for this disparity should be immediately obvious from the pictures.[9]

9 It may seem that the lower MI mapping looks like a better match, and a fairly strong case could be made for that. But they are both valid partial coarse grainings and we have argued that there is no correct emergence – we may not want the 'better' match if it

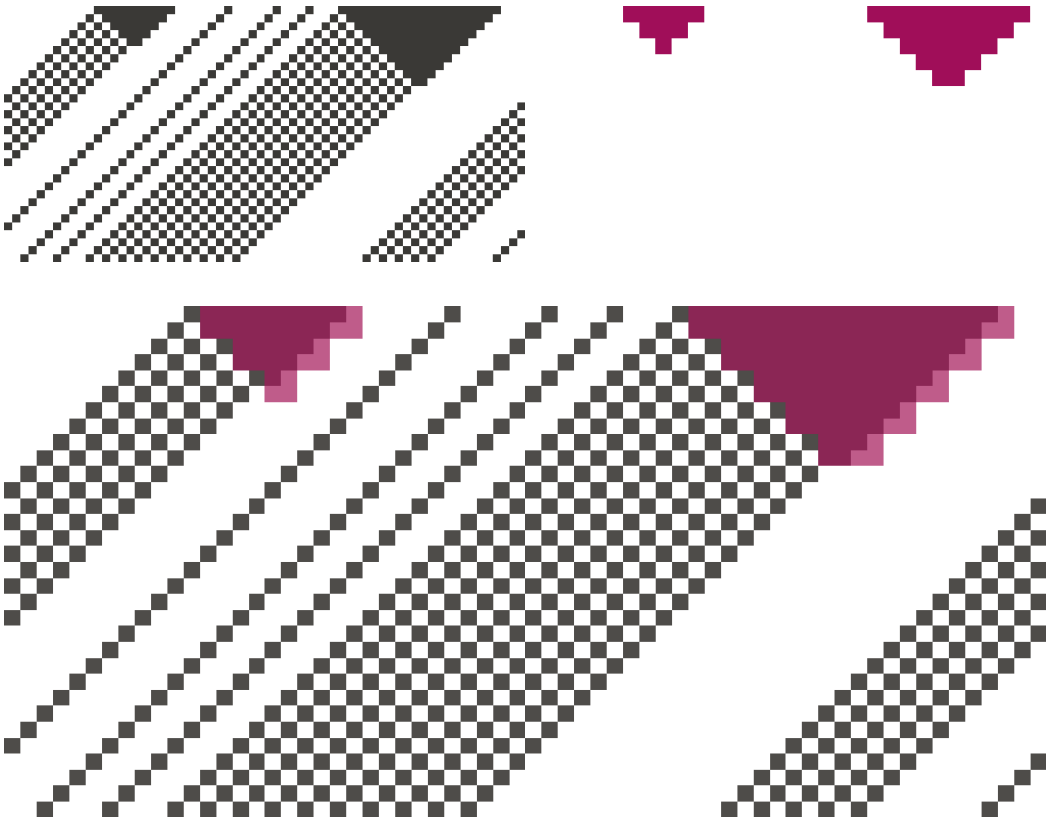


Figure 11.33 Rule 162 partially coarse grained to rule 160 with mapping □■□■.

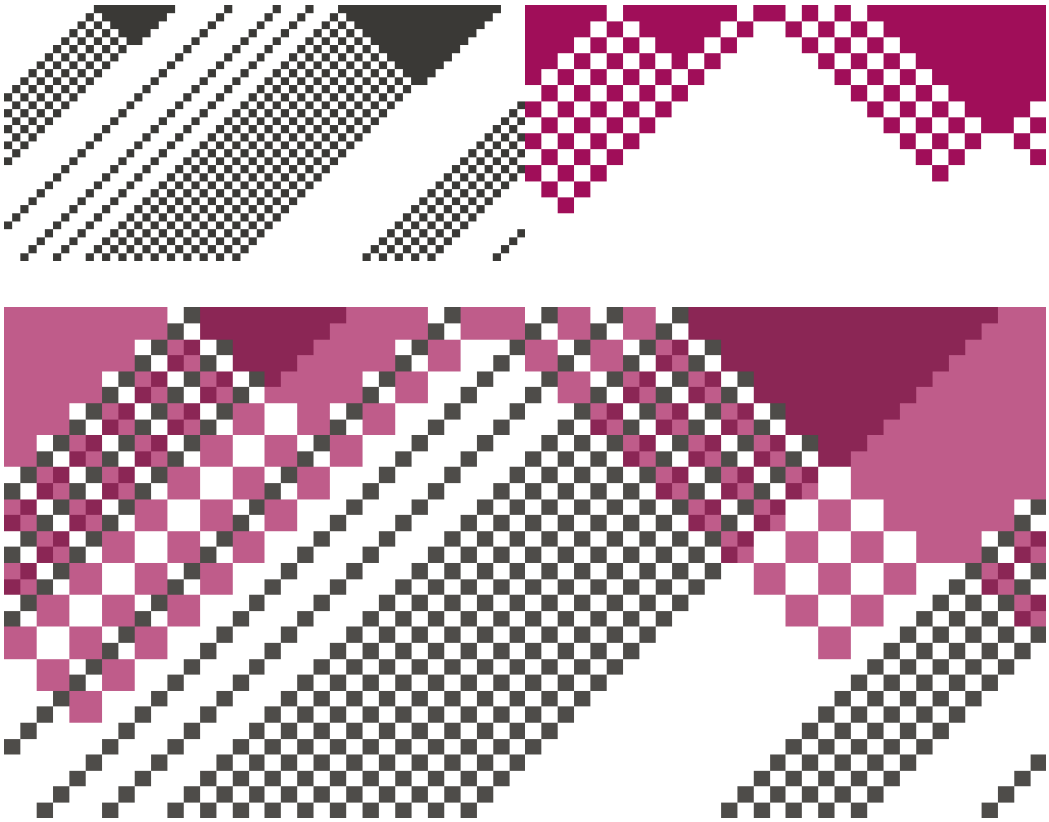


Figure 11.34 Rule 162 partially coarse grained to rule 160 with mapping ■■□■.

performs worse in our model. And more pertinently, we aren't aiming to find the best model here (by any criteria), merely to show that different mappings can have a significant impact on the results.

We also sometimes see this when additional partial mappings are returned for a total rule. 162 totally coarse grains to 128 with mapping □□□■, but also partially with mapping □■□■ (and others). The total coarse graining has a significantly lower MI (0.12) than the partial one (0.78). The distinction in this case is subtle: the partial mapping maps ■□ to ■ and so includes the right edges of more triangles than the total mapping.

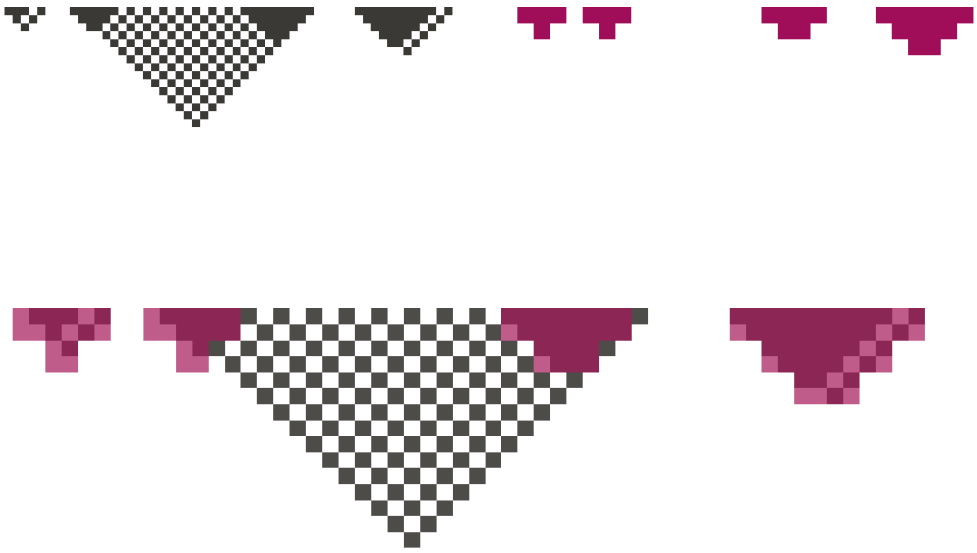


Figure 11.35 Rule 162 totally coarse grained to rule 128 with mapping □□□■.

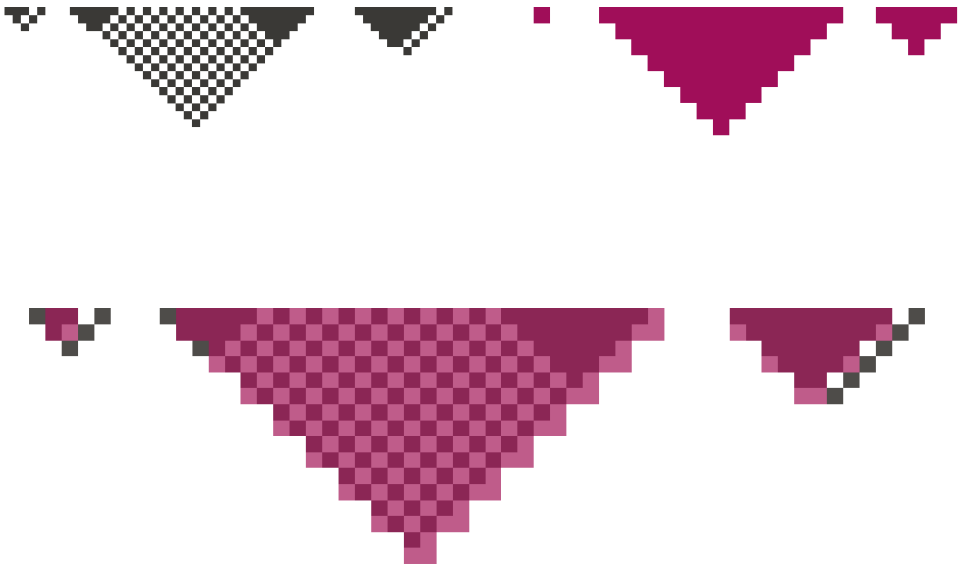


Figure 11.36 Rule 162 partially coarse grained to rule 128 with mapping □■□■.

It is important to ensure we use the best mapping when we are choosing between partial coarse grainings. As mentioned in §11.8, we can do this efficiently by checking the MI of each mapping against just one string.

11.13 Key points

- Mutual information is a quantitative measure of the goodness of a coarse graining.
- Partial coarse grainings can have MIs that are higher than total coarse grainings: making some mistakes gives them the freedom to capture low level behaviour that total coarse grainings can't.
- The choice of mapping can have a significant impact on the MI and goodness of a coarse graining.
- For some rules, the test string used can have a significant impact on the MI of a coarse graining, though for other rules it does not. It is therefore important to use a test string typical of the problem in question to find good coarse grainings.
- We found that one test string that concatenated all input states gave a good approximation for a much larger test set of strings, significantly reducing the time needed to test coarse grainings

12 SUBJECTIVE EMERGENCE

This chapter provides the final elements needed for our goal of finding emergent models of the behaviour we want in the underlying system, a requirement for engineering emergence (§9.19). First we show how to find good coarse grainings of the different behaviours of a CA over space and time. We use these ideas to demonstrate phase changes in CAs and show how to catch the phase transition at the high level.

The behaviour we want to capture at the high level may not be the most complex behaviour the CA can produce; in particular, we see that some partial coarse grainings have high MIs, despite looking like poor matches to us. We introduce extra entropy, the amount of uncorrelated behaviour at the high level, as a way to find such rules.

We also introduce directed coarse graining, a way to extract the features we want as coarse CAs by adding exceptions to the behaviour of fine CAs. We find an interesting example of adding an exception to rule 102 (a Class 3 rule) that gives us a larger scale copy of rule 110 (a Class 4 rule capable of universal computation [70]).

We already know that emergence is subjective and that there is no such thing as a correct emergent property. It may be that a number of different views, each with a different emergent model, exist for a particular system; indeed we often see this with coarse grainings.

When we stare at Conway's Game of Life, it is natural to notice emergent, moving patterns such as gliders. We, as humans with our accompanying genetic baggage, are imposing our real world propensity for object detection (a historically useful emergent abstraction) on this chequerboard representation of a mathematical space. We find it much more difficult to detect meaningful objects or patterns in the transient 'soup' seen initially on starting Life from a random state. (Though we should note that patterns can still be in there [155].)

In §10.2, we introduced a technique to identify objects in Life, but this wasn't a very general, expandable or flexible technique. It didn't help predict the future state of the system (except in very specific circumstances) and it wasn't robust even to quite small and obvious changes in the system state. We could certainly improve the technique, perhaps by refining the shape edge detection routine, but these improvements would be incremental and unlikely to address its underlying fragility and narrowness of scope.

This chapter develops a more robust technique for finding objects and patterns through coarse graining.

12.1 Feature extraction in emergent systems

Coarse graining captures similarities between pairs of CAs by creating a high level (coarse) model of certain aspects of the low level. This is analogous to identifying specific emergent behaviours seen in Life, such as gliders. With coarse graining we can extract general features of a CA based on a set of broad behaviours, not just one particular instance based on a specific example. For instance, if we establish that we want elementary CA rule 128 (the transient triangle rule) as our coarse feature detection rule, we could then use it to detect CAs that produce triangles of any size and in any location and other triangle-like structures.

Figure 12.1 Rule 128

But these coarse grainings have been quite general, seeking to model (at least some aspect of) the whole underlying CA. We may want to capture particular, predetermined features of the CA in our model, similarly to identifying just the gliders in Life. (This is relevant to engineering emergence in §9.19, for which we naturally seek a model that captures the behaviour we want, not just any behaviour.) One important prerequisite for this is locality, which we introduce now through feature extraction. Feature extraction allows us to find the best (highest MI[1]) coarse graining for particular areas of interest on the CA grid.

At its simplest, we can split the underlying CA into large *chunks* and select the best coarse graining for each chunk. We now explain how feature extraction works, assuming we have already coarse grained a CA and obtained several possible coarse CAs through it.

12.2 How to extract features from an elementary CA

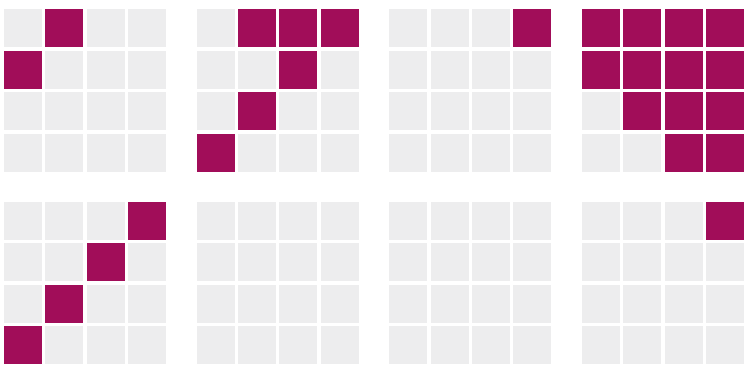


Figure 12.2 A CA divided into chunks for feature extraction.

1 While we revise the definition of 'best' in this section, for now we continue to equate high MI with goodness.

1 Coarse grain the CA as described in §10.10. (We may choose to loosen the criteria for valid coarse grainings; see §12.22.) We assume that we are considering just one fine rule. Though there may be some cases for which comparing several fine and coarse rules makes sense, generally we shall wish to find the best high level model for a single fine rule's behaviour in each chunk.

2 Select which of the coarse grainings returned should go forward to the next stage. We may choose all of coarse grainings, or we may select a subset based on their MI or other criteria. See also §12.30.

3 Create one fine CA and a coarse CA for each coarse graining that we are considering.

4 Initialise and run the CAs.

5 Divide the CAs up into chunks.

  5.1 If we decide to use four chunks spatially, we would split each CA (the fine and all coarse CAs) into four equal pieces horizontally. (So if the CA were 80 cells wide, it would have an *x-chunk* of 10, equal to 10 cells at the coarse level and 20 cells at the fine level at $g = 2$.)

  5.2 If we decide to use a *y-chunk* of 2 and have run the CA for 40 timesteps, we would divide the CAs into 10 segments of 2 coarse cells (4 fine cells) temporally.

6 Iterating through each coarse rule in turn, calculate the MI for a chunk. We do this exactly as before, but this time only using the cells in that chunk.

7 Select the coarse CA with the highest MI for each chunk. We now have the best coarse grainings for different areas of the CA, distinguishing different features in the low level CA.

12.3 Example of feature extraction

To illustrate feature extraction, we coarse grain rule 140. Valid coarse grainings include rules 136 and 204 (§10.11, §A), and we limit ourselves to choosing between these two for this example.

- We use the initial condition

  □■□■■□□■□□□■□□□□□■■■■■■■□□□■□□□□□■■■■■■■■■■
  ■■■■■■■■□□□□□□□□□□□□□□□□■□□□□□■□■■□□■□□□□□□□□
  □□□□□■□□□□□□□□■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

- We know that rule 140 draws right angled triangles with vertical lines at their tips (Figure 12.3). We also know that rule 136 captures these triangles at the coarse level and that rule 204 matches fairly well to the vertical lines. The initial condition is designed to show chunking, so we have two long series of ■s that will make large triangles at the low level, separated by a relatively sparse area with some ■s for rule 204 to model. At 128 cells wide, the initial condition used here is larger than in previous examples so we get non-trivial MI results after chunking.

- We divide the space into chunks with an *x*-chunk of 16 (16 coarse cells across spatially) and a *y*-chunk of 2 (2 coarse cells down temporally). We run the CA until fine timestep 24, so the CAs are divided into four chunks horizontally (16 × 4 = 64 coarse cells = 128 fine cells) and six chunks vertically (2 × 6 = 12 coarse cells = 24 fine cells).

Figure 12.3 CAs showing rules 140 (fine rule in black), 136 (coarse rule in purple) and 204 (coarse rule in red) started from the example in §12.3's initial condition. The bottom picture shows the *x*- and *y*-chunks over rule 140.

- We calculate the MI for each chunk. Looking at the runs of rules 140, 136 and 204 in Figure 12.3, we see that rule 136 should be a good match in chunks two (initially, at least) and four. Rule 204 appears to correlate better with the other chunks.

		***x*-chunk (cells)**			
		1-32	**33-64**	**65-96**	**97-128**
y-chunk (time-steps)	**1-4**	0.72	1.92	0.65	0.97
	5-8	0.72	1.92	0.00	1.37
	9-12	0.00	1.37	0.00	1.92
	13-16	0.00	1.37	0.00	1.92
	17-20	0.00	0.00	0.00	1.92
	21-24	0.00	0.00	0.00	1.37

Figure 12.4 Mutual information between rules 140 and 136, divided into four chunks horizontally (*x*-chunks) and six chunks vertically (*y*-chunks). The cell indices corresponding to each chunk are also shown.

		***x*-chunk (cells)**			
		1-32	**33-64**	**65-96**	**97-128**
y-chunk (time-steps)	**1-4**	1.92	1.52	1.46	0.72
	5-8	1.92	1.52	1.46	0.72
	9-12	1.92	0.97	1.46	0.72
	13-16	1.92	0.97	1.46	0.72
	17-20	1.92	0.72	1.46	0.72
	21-24	1.92	0.72	1.46	0.72

Figure 12.5 Mutual information between rules 140 and 204.

- We select the coarse graining with the higher MI as the better match for that chunk. There is a broad agreement between the rule that appears the better match in Figure 12.3 and the rule with the higher MI at each point: Figure 12.6 looks like a coarser version of the CAs' outputs.

		***x*-chunk (cells)**			
		1-32	**33-64**	**65-96**	**97-128**
y-chunk (time-steps)	**1-4**	204	*136*	204	*136*
	5-8	204	*136*	204	*136*
	9-12	204	*136*	204	*136*
	13-16	204	*136*	204	*136*
	17-20	204	204	204	*136*
	21-24	204	204	204	*136*

Figure 12.6 Table illustrating which rule matches better (has a higher MI) out of 136 and 204 for each chunk.

This example was chosen to give particularly convincing results, but the correlation between the rule with the highest MI and the rule that 'looks right' is almost always very good. (We discuss cases where it is not, and a solution to the problem, in §12.21.) Further examples can be found in §E.

There are cases where feature extraction works less well, for instance when a chunk falls entirely within a block of cells that are all of one state. All rules have zero MI in such a chunk, so it is impossible to choose the correct coarse graining, even though it looks as though it should be the same as

the surrounding shape. A more significant problem occurs when a high entropy coarse rule matches better (has a higher MI) than a low entropy coarse rule that seems to model the behaviour we desire. Again, we address this issue in §12.21.

12.4 Forcing contiguous blocks

Rather than simply selecting the highest MI graining for each chunk, we may decide to favour contiguous blocks of a single coarse graining. We can do this by requiring any coarse graining using a new rule to have an MI that is at least a certain percentage above the MI of the coarse graining we are currently using, or alternatively by requiring that no rule can replace the current one unless its MI drops below a stated proportion (perhaps 50%) of its peak value (or a moving average). These techniques were devised specifically for investigating CA phase transitions (see §12.6). While they work well in some situations, they did not prove useful in this context and have thus not been developed further.

12.5 TOWARDS DIRECTED COARSE GRAINING

Coarse graining with feature extraction is a potentially powerful technique: we have characterised the broad behaviour of a region and we can use this to predict, with reasonable accuracy, the future state of this region. We can now identify and label salient, high-level features of the underlying model in a fairly robust and general way.

But we are only half way towards our goal of finding specific structures in ECAs. We have not directed the search so far: we have looked for interesting things, rather than specifically searching for things in which we are interested. The next sections investigate how we can perform a directed coarse graining through phase changes in cellular automata.

12.6 Phase changes in cellular automata

Water has three different states – solid, liquid and gas – that have very different properties. We see water undergoing a phase transition when it changes between these states, but at a molecular level it remains essentially the same throughout (a characteristic common to many emergent systems).

Despite their mathematical transparency – we can model them unambiguously and completely with a few equations – CAs can show phase transitions too. One of the clearest examples of this is rule 130. Rule 130 contains behaviour that can most naturally be coarse grained to a rule in Wolfram's Class 1 (the triangles of rule 128) and, after the initial transient has died out, to a rule in Class 2 (the lines seen in rule 34 and others). There is a phase transition between these two emergent models that is not present at the low level.

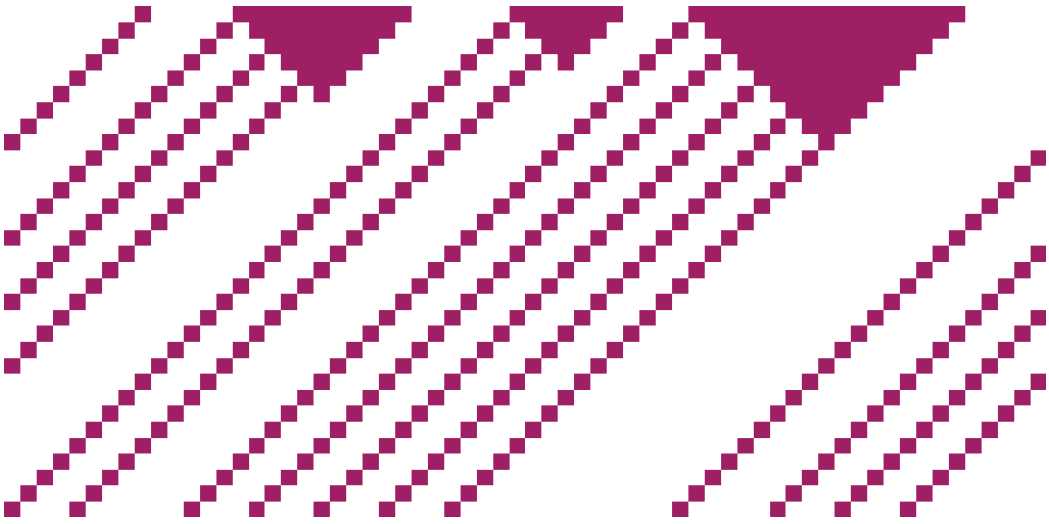


Figure 12.7 Rule 130, the fine rule in this example.

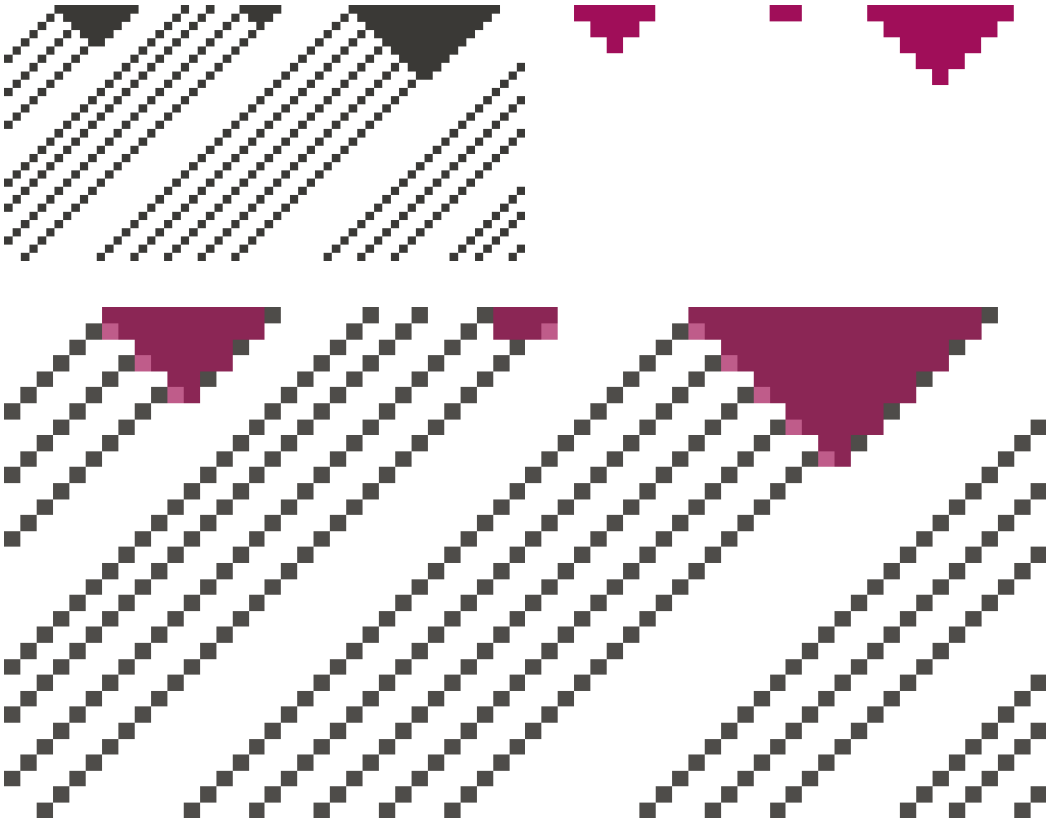


Figure 12.8 Rule 128, the coarse rule that matches rule 130 before the phase transition.

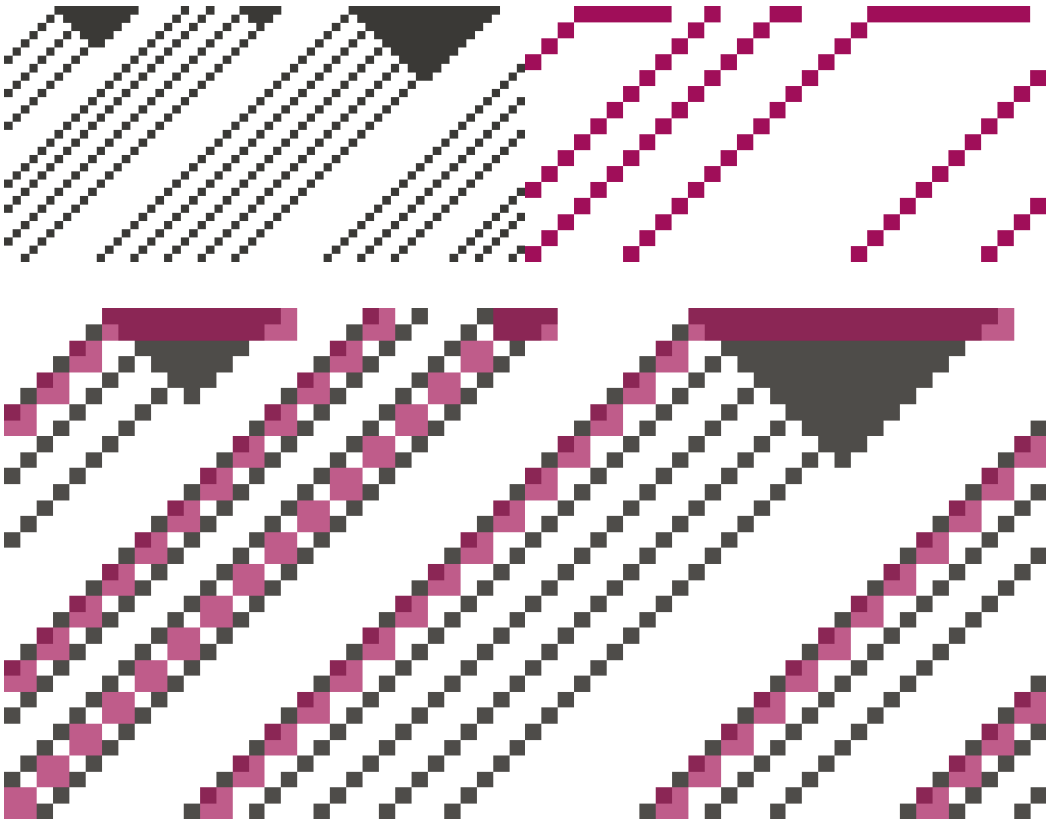


Figure 12.9 Rule 34, the coarse rule that matches rule 130 after the phase transition.

Most of the examples in §12.5-§12.22 use three rules: the fine rule 130, and coarse rules 128 and 34. To show the phase transition between coarse rules 128 and 34 to best effect, we created a special CA initial condition, the development of which is described in §D.

12.7 Catching the transition

Clearly it would be useful if we were able to detect these phase transitions automatically, allowing us to trigger a rule change at the high level. We have already seen how we can extract features to assign the most appropriate rule to each chunk of the CA run, and it is tempting to do so again.

Suppose we try to use feature extraction and start a simulation of rules 130 (fine), 128 (coarse) and 34 (coarse) with an initial condition similar to that shown in Figure 12.7. Based on our experience in §12.3, we expect that rule 128 will be a good match during the initial transient, and rule 34 thereafter. Rule 128 does have a high MI at first that reduces over time (as expected), but unfortunately rule 34 is not always there to take over as a good match with high MI. Both coarse rules started from the same initial condition, but this is a condition suited only to the transient 128. Rule 34 has a fairly tenuous model of the underlying system – in Figure 12.8, there were few places for it to map to – and will continue to propagate this poor facsimile even when the low level becomes much more favourable for mapping to its behaviour.

If instead we were to initialise rule 34 later, perhaps halfway down the large triangles in Figure 12.6, we would get a much better correlation with the underlying model than we were able to get at the start. And, as the underlying model presumably favours rule 34 after the phase transition, we would get an even better correlation if we restarted it at that point (see §12.9 for an example). This problem of a poor post-transition model must exist generally: for a phase transition to occur, the initial lower level state must suit one high level model well and the other (or others) poorly.

Figure 12.10 shows the MIs between rules 130 and 34. Two series are shown: one for which we restarted rule 34 at each coarse step, and another with no restarts. It also shows the MI between 130 and 128 from the same initial condition. (Restarting 128 has no effect on the MI as the coarse graining is total.)

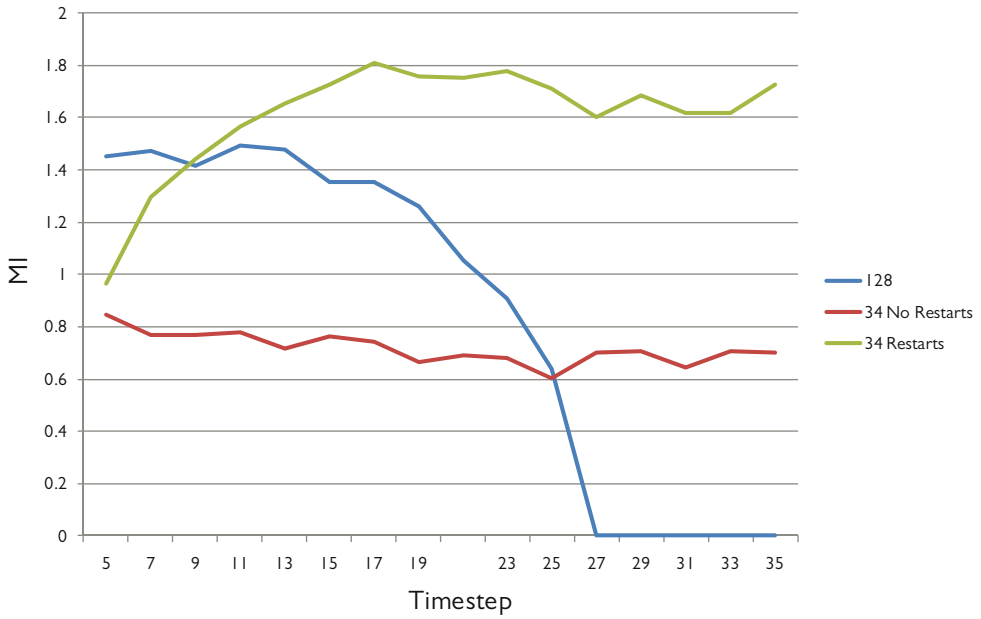


Figure 12.10 The MIs between rule 130 and rule 34 with no restarts and rule 34 restarted at each coarse timestep. The graph also shows rule 128 (a total rule, so it returns identical results with and without restarts). The initial condition in §12.8 was used at $g = 2$.

There are a number of possible ways to exploit the phase transition to get a better match between the high and low levels. We now consider the two most obvious ones.

12.7.1 Restart at the phase transition

One solution is to restart rule 34 (reinitialise the mapping between the rules) at the phase transition. To capture the phase transition most accurately, we claim that we should follow the red line on Figure 12.11: follow rule 128's curve down until its MI drops below that of the (poorly matched) rule 34, then restart rule 34 and switch to that. This changeover point is approximately the time at which the triangles in rule 128 disappear, and is a strong candidate: although the diagonal lines that match rule 34 have been increasing in number before then, the large triangles still dominate the CA visually until approximately that point. There is also a distinct change in the CA, as a major structure that has been present vanishes completely.

Figure 12.11 The graph in Figure 12.10 is reproduced, with the addition of a suggested phase transition point between rule 128, which matches fine rule 130 initially, and rule 34, which matches rule 130 subsequently.

12.7.2 Restart at each step

Another solution is to restart rule 34 at each timestep. This avoids the need to calculate (perhaps at some cost) the right point at which to restart the coarse rule, and would guarantee a good mapping at every point (as the rules have no time to drift away from each other).

Unfortunately restarting the coarse CA at each point negates a large part of the reasoning behind coarse graining by reducing it to just a mapping. We effectively ignore the rule of the CA – we no longer obtain the current coarse state from the previous coarse state, but from the current fine state instead – and in the process lose the predictive power of coarse graining. We also find that rules such as 213 (which, like 34 and 130, draws diagonal lines, but draws them in the wrong direction) become extremely good matches based on their MI.

We could try to ameliorate these difficulties by restarting the coarse CA some steps earlier: instead of calculating the coarse rule state directly from the fine state at each step via the mapping, we reinitialise the coarse rule earlier and allow it to run for a few steps before using it in the MI calculations. By running several coarse CAs with staggered restart times in parallel, we can read off results at each step from a CA initialised a fixed time in the past.

Allowing the CA to run for several steps does sometimes help, but its efficacy is very dependent on the current cell pattern and CA rules. If rule 213 is restarted from a state similar to that in Figure 12.12, it continues to overlap with the underlying CA – and have high MI – for quite some time, even as the fine and coarse rules diverge. Also pushing the initialisation point further up delays the

time at which we can act on the real phase transition, so the disadvantages of this technique outweigh the small (and occasional) benefits.

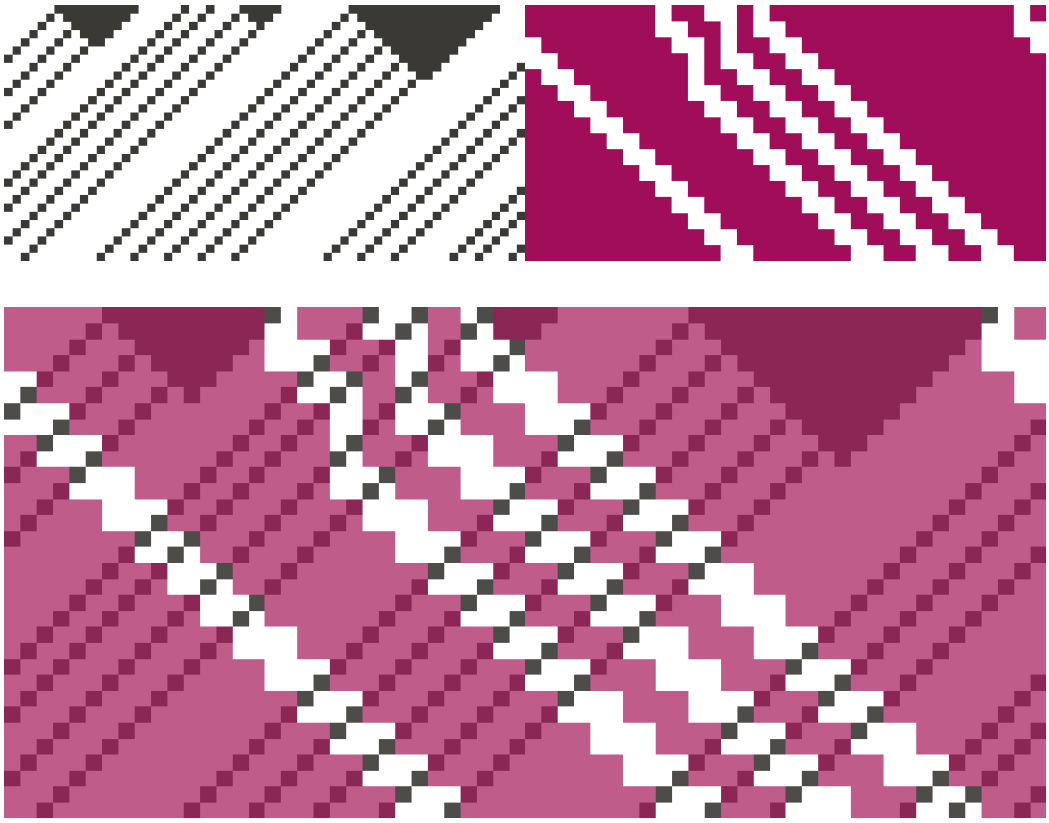


Figure 12.12 Rule 130 coarse grained to rule 213. Though a poor match for rule 130 in terms of its behaviour, rule 213 has a high MI with rule 130 for some time after being restarted.

12.8 MI and phase transitions

We have just seen that not restarting the coarse rule at all often gives poor results, and restarting it every step reduces the coarse graining from a useful model to just a mapping. Restarting the coarse rule just once, at the transition point, still seems like the best solution. We also suggested that the transition point for rule 130 is at the point when the triangles matched by coarse rule 128 disappear.

But we still have little to justify this claim other than 'it looks right'. In fact, Figure 12.10 suggests that we should switch to rule 34 sooner, given that its MI is higher, after a restart, than 128 from around timestep 9.

To explore this hypothesis, the coarse rule was restarted (over many runs) at fine times 1-35, comfortably covering the interval during which triangles are present for the initial condition in §12.8 (steps 1-26). For each run, the MI between rules 130 and 34 and rules 130 and 128 was recorded at times 1-51.

12.9 Exploring restart position

Figure 12.13 shows many runs of rule 34, restarting each at a different time. The coarse graining becomes significantly better over time as the fine rule creates more lines to which 34 can map. Successive images show the fine CA shifted up by two (fine) timesteps, with the restart taking place at the top of each picture.

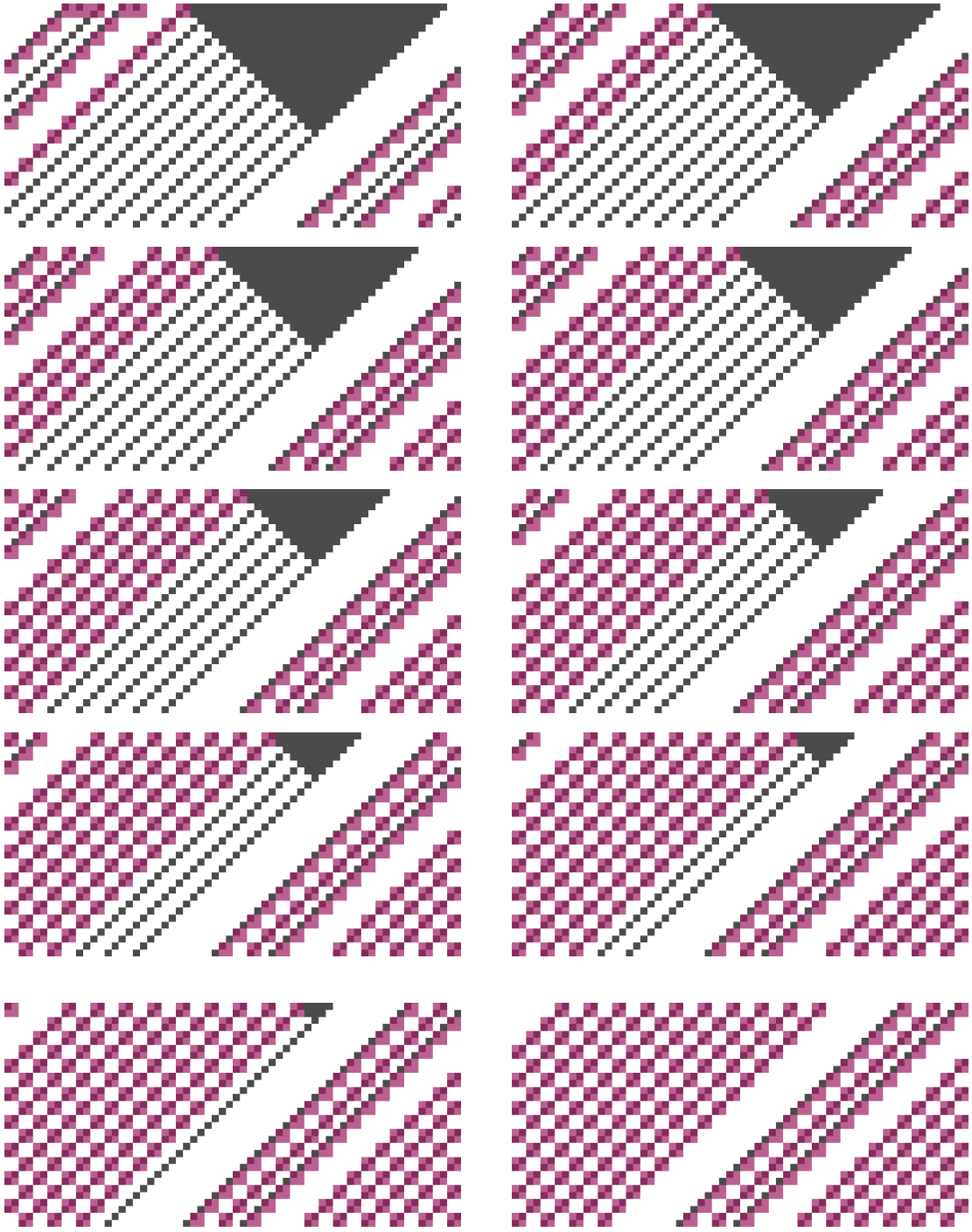


Figure 12.13 In these pictures, rule 34 is able to capture substantially more of the underlying behaviour after being restarted. Further, the amount of behaviour captured by rule 34 increases for later restarts, up to the point where the triangles have disappeared.

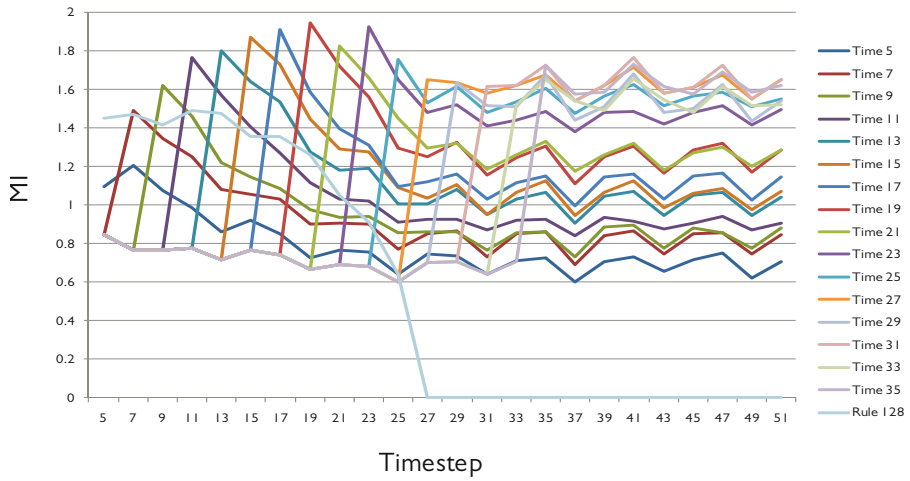


Figure 12.14 MI over time between fine rule 130 and coarse rule 34, restarting rule 34 at (fine) times 1-35 (in steps of 2). The MI between rule 130 and coarse rule 128 is also shown.

Figure 12.14 shows the results of restarting coarse rule 34 at (fine) times 1-35. We would expect the MI between the rules to jump immediately after the coarse rule has been restarted (because it has had no time to diverge from the fine rule's behaviour), and indeed this is what we see.

We also see that that the MIs of the runs that were restarted early (at times 5-11) quickly decline, dropping quite close to the MI we saw when rule 34 was not restarted at all. The first runs whose MI remains high are those restarted at times 25 and 27, approximately when the triangles disappear (time 26). This lends credence to our assertion that this is the transition point – we get the best MI results over time by restarting at this time.[2]

12.10 A clearer transition graph

With its plethora of crossing lines, Figure 12.14 is perhaps not the easiest graph to interpret. In particular, it is not obvious when the phase transition takes place, so we now explore how we can present the MI data to show phase transitions more clearly.

First we note that we are not necessarily trying to find the rule with the highest MI; we are trying to find the rule that matches some (significant) structure of the underlying fine CA, and also trying to find the best point to switch between the coarse rules. We are also considering just two rules, and know that one of them reflects the underlying model during its transient phase and the other the model thereafter. As a corollary, we also know that one of these rules has a sharply declining MI over time and the other is relatively constant.

So the question is whether the second rule is better relative to the first, transient rule. With this in mind, we can subtract rule 128's MI results from each run of rule 34's MI results, giving us Figure

2 Note that the MI bump we see in the restarts over times 15-25 is due to rule 34 matching the sides of rule 130's triangles, which of course disappear over time.

12.15. This *normalised graph* has a distinct S-curve, swinging from negative to positive at the transition point as rule 128's MI drops to zero. The runs restarted at times 25 and 27 have the highest value at this point, and remain amongst the highest for the remainder of the graph.

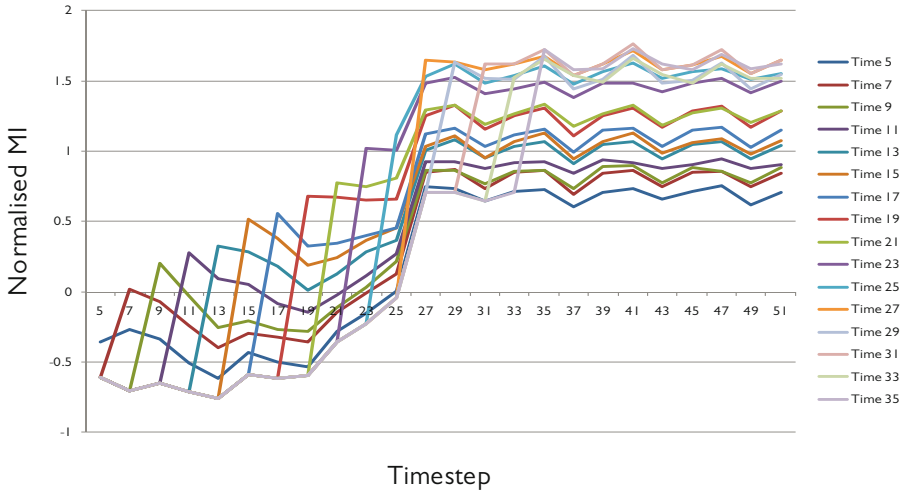


Figure 12.15 A normalised graph showing the MI over time between fine rule 130 and coarse rule 34, relative to rule 128. As with graph, rule 34 was restarted over multiple runs at (fine) times 1-35 (in steps of 2).

12.11 Coarse graining rule 130 to rules 128 and 213

The same graph normalisation technique of subtracting MIs can be used even if one coarse rule is a poor match for the underlying behaviour. As before, we use fine rule 130 and coarse rule 128 to capture the transient behaviour, but we use coarse rule 213 instead of rule 34. Rule 213 draws thick diagonal lines in the opposite direction from 130's lines. It has a (very) high MI with 130 immediately after being restarted (§12.7.2), but falls off as the high and low level models diverge. (Though the MI can increase again if 213's lines cross those produced by another part of the model, yielding higher than expected MIs. This behaviour can be seen in Figure 12.17, where a distinct hump appears at time 35-41.) Despite looking like a poor match, the MI between 130 and 213 is approximately as high as between130 and 34.

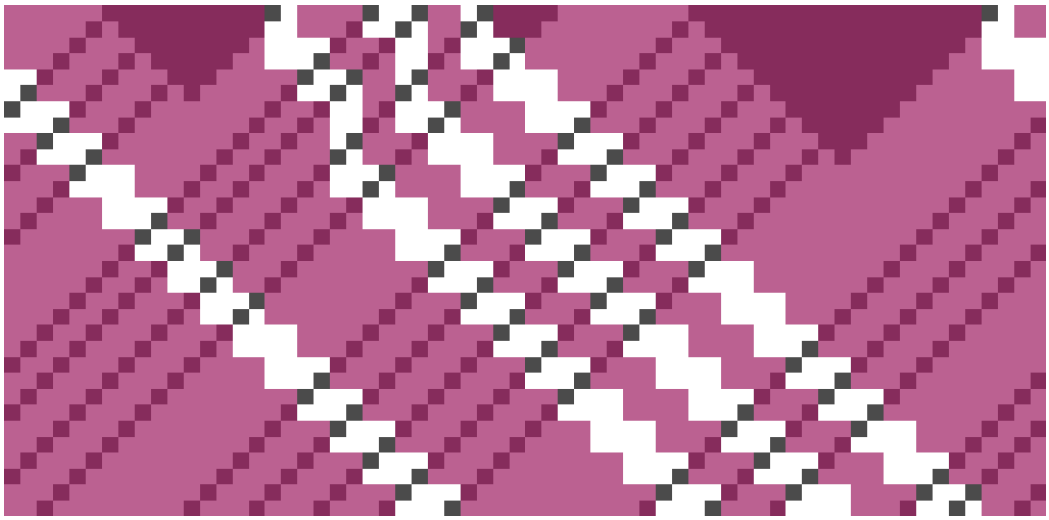


Figure 12.16 A run of rule 130 coarse grained to rule 213. Though apparently a poor match, 213 has a high MI because its diagonal lines frequently cross those of the fine rule.

Even with this relatively poor rule, we see the same 'phase transition' shape as with the graphs in §12.10. In fact the same MI patterns can be seen for a number rules, both those that are good coarse grainings and bad coarse grainings of the underlying model.

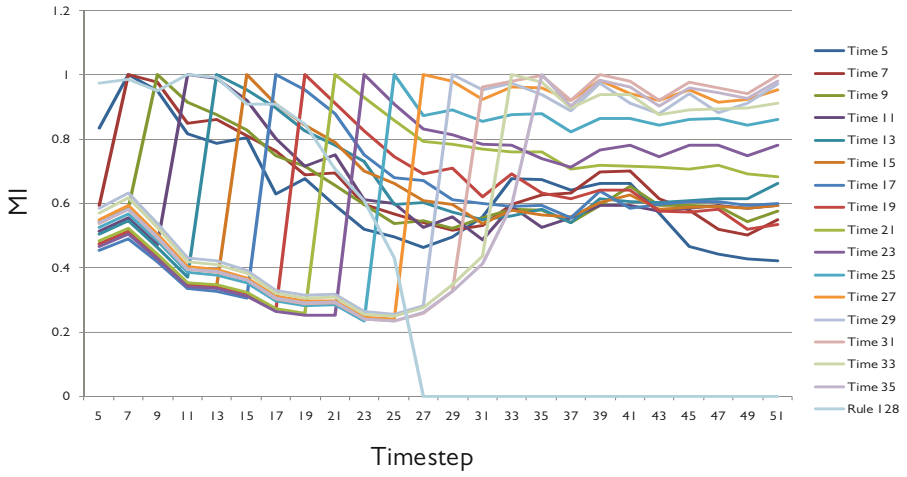


Figure 12.17 Graph showing MI over time of restart times 1-35 for rule 130 coarse grained to rule 213. We also see the MI between rules 130 and 128 over the same period. Although 213 looks like a poor match, it has a high entropy and often intersects with 130's diagonal lines, which gives it a high MI too.

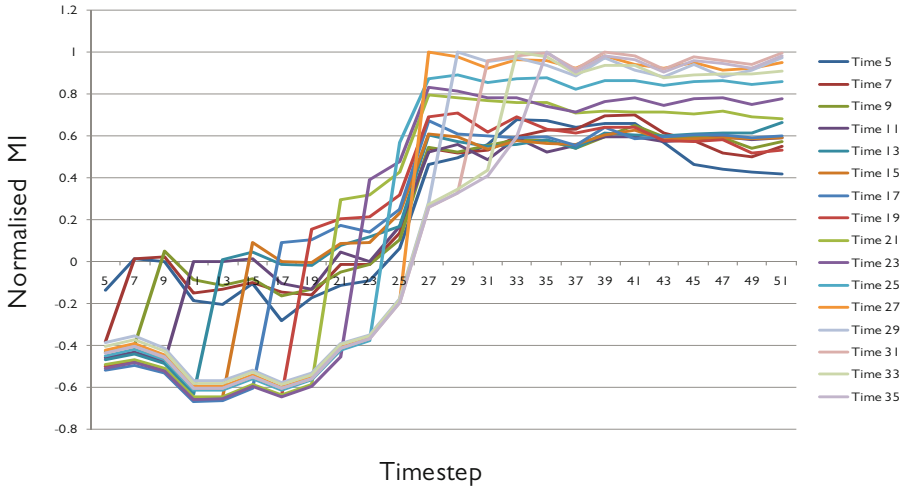


Figure 12.18 Normalised graph of Figure 12.17. We see the same basic S-curve as Figure 12.15, though the finer detail appears messier: the lines are significantly less coherent and there is quite a bit of overlap as the runs progress.

12.12 Coarse graining rule 130 to rules 128 and 84

Rule 84 is another poor coarse graining of rule 130. Unlike rule 213, rule 84 captures very little of the underlying behaviour (even immediately after a restart), so the MI between it and rule 130 is low. This is reflected in Figure 12.20 and Figure 12.21, where the MI between rules 130 and 84 is low and quite noisy. The reasons for this are similar to rule 213: restarting the coarse CA briefly gives a higher MI while the CAs still overlap, before they diverge in opposite directions. The MI is lower than Figure 12.17 as rule 84 usually produces fewer lines than 213. The normalised graph has a similar shape to the graphs in §12.10 and §12.11.

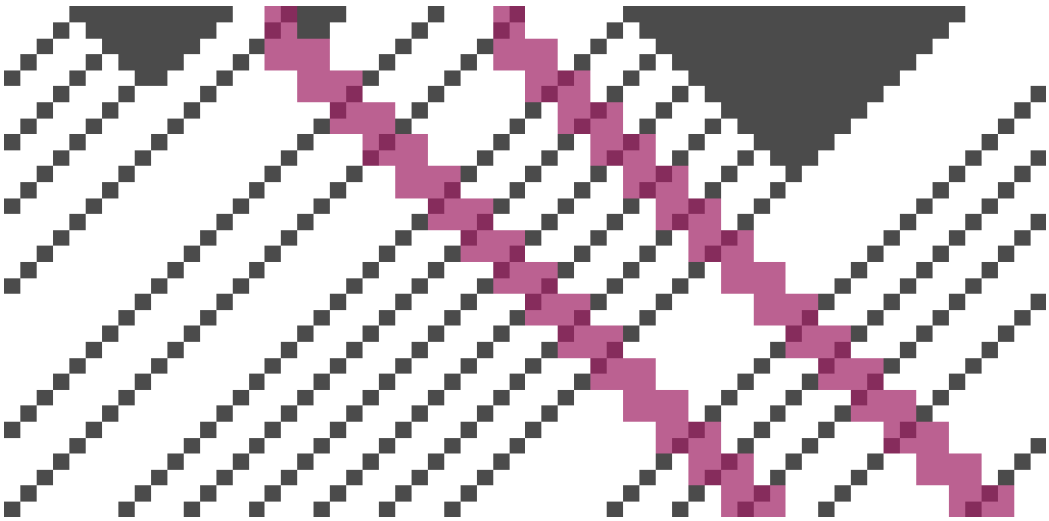


Figure 12.19 A sample run of rule 130 coarse grained to rule 84. 84 is a poor match for 130.

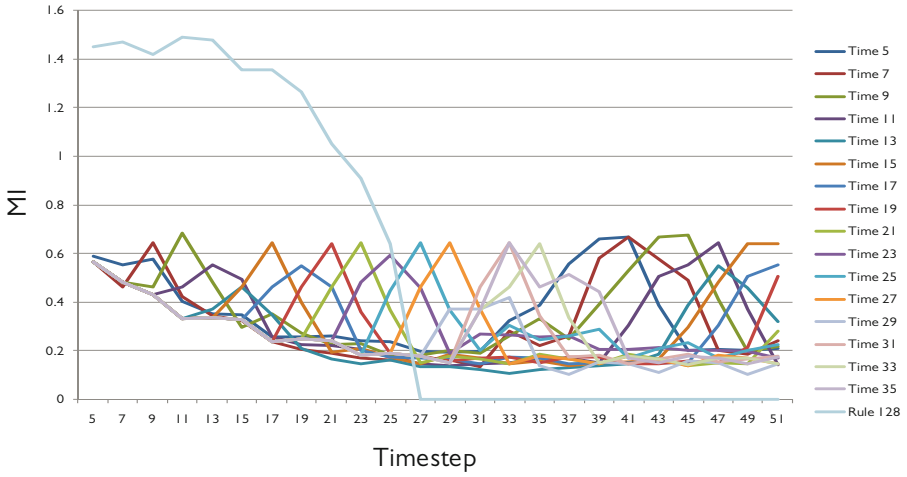


Figure 12.20 Graph showing rules 130 coarse grained to rules 84 and 128. Rule 84 is a poor match for rule 130: its MI is significantly lower than that of rules 34 or 213, with a baseline of just 0.2 bits. It shows a brief rise in MI when the rule is restarted, before dropping quickly back down.

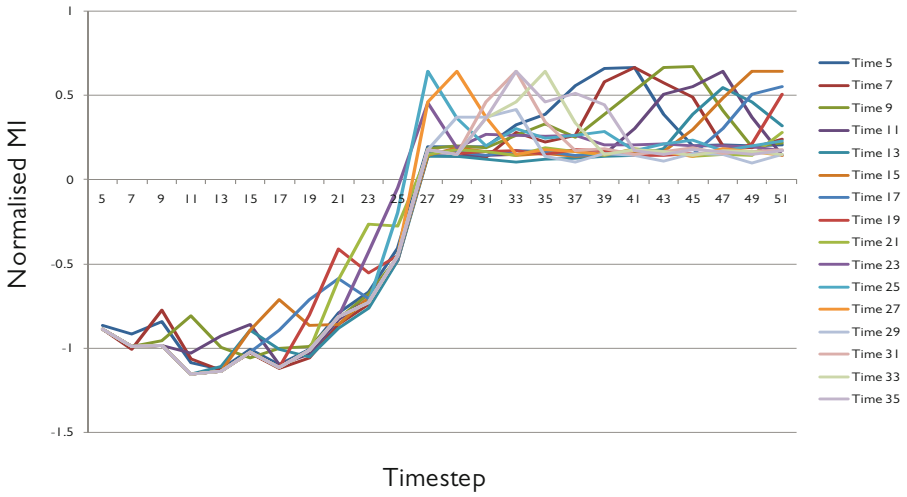


Figure 12.21 Normalised graph of rule 130 coarse grained to rules 84 and 128. We see the same pronounced S-curve as in Figure 12.15 and Figure 12.18.

12.13 Relative graphs

As this section considers the relative merits of two rules over time – and specifically relative to themselves as much as to other rules – rather than the absolute mutual information, it may make sense to normalise the rules' MIs. Experiments have shown that this is unnecessary: although rules 128 and 136 have a relatively low absolute MI, the MI drop at the transition point is still considerably greater than the differences seen in the second rule throughout the run. There is also a risk of normalising away differences between series of the same rule: for example, the bump we see between times 5-25 in Figure 12.14 is lost, though it is worth noting that this has little impact on the choice

of transition point. There is nothing intrinsically wrong with this approach, but it is unnecessary and slightly reduces the clarity and quantity of information conveyed by the graphs.

12.14 The inevitable S-curve

S-curves are often associated with phase transitions, but the curves found in the graphs above seem a touch convenient. In all of the cases examined, there is a transition at the high level from a coarse rule that draws triangles (128 or 136). When that rule runs out of triangles, its MI rapidly falls to zero and the normalised graph's data series switch from negative to positive, yielding an S-curve. Surely this behaviour is almost inevitable?

This is true, but it largely misses the point. The aim here is to find the best point at which to switch between the transient and non-transient rules. We know a priori that one coarse graining has a higher MI early on and that the other coarse graining has a higher MI later. We also believe that switching rules (and restarting) too early gives a poorer coarse grained mapping than switching at the right point. The objective here is to get a better insight into the right transition point so the sometimes higher MI of the second rule doesn't force a switch too early.

We know that the MI of the transient rule will be high and rapidly fall to zero. We know that the MI of the other rule will remain roughly constant throughout the run, or at least will remain so during the non-transient phase of the model.[3] We exploit these facts to transform the MIs and find the transition point we want. This wouldn't work if the rules had arbitrary MI curves over time, but then nor would there be a phase transition.

12.15 Analysis of finding transition points

The previous sections spend considerable time discussing how to find the correct transition point and when to restart the non-transient rule. The transition point seems to be at approximately the time where the transient triangles disappear: as well as looking right, we get the best post-transition correlation (highest MI) rules by restarting the second coarse rule this point. We have also developed a graphing technique that shows this transition point clearly.

Some fairly stringent assumptions were required for the process to work: the transient rule's MI must decrease rapidly, while the other rule's MI must not; and the absolute MIs of the rules are largely immaterial. These provisos aren't terribly restrictive when considering CA phase transitions, as they are all inherent to the problem. But they are restrictive if we want to use this approach in other domains. And we are still no closer to deciding which of 34 and 213 is a better match for the

3 This is certainly true here, as we are modelling Wolfram Class 1-type behaviour at the fine level with a Class 1 rule, but any post-transition high level model must maintain a comparatively good and consistent representation of the system's ongoing behaviour for it to be considered useful.

post-transition 130, given they have very similar MIs (the same problem we faced when discussing feature extraction in §12.1).

In fact, we appear to be running up against a limitation of using mutual information in this context. Though any useful emergent model must capture at least some of the underlying model, even the most comprehensive one will throw away much of this behaviour – it is this predictability (over the underlying system) that makes these emergent models useful (and interesting) in the first place (§9.5). And while some of these emergent models retain a useful subset of the underlying behaviour, others capture an equally large, but less desirable, fraction. Continuing to use MI as a proxy for model goodness may not be possible, since we are no longer interested in capturing as much of the underlying behaviour as we can, but in capturing specific facets of it; facets that, like rule 128, may not even be that information rich.

12.16 Overcoming the mutual information problem

We could restrict the rules we consider to a small subset of possibilities, perhaps excluding rule 213 in this case because its behaviour is obviously wrong. Limiting the scope of the problem is useful – essential, even, for almost any real world situation – but it will only get us so far. After all, if we have such a good idea of what the solution looks like already, why bother searching?

Another approach would be to search for coarse grainings with an MI closest to a target value, rather than the maximum possible. But this runs into similar difficulties: finding this target value is likely to be non-trivial and would probably require us to have a pretty good idea of what the answer is before we start.

We now look at a couple of more promising approaches that change the way we calculate mutual information by increasing block size and moving to a temporal MI measure.

12.17 A temporal dimension in mutual information

Usually we want to match large-scale structures that span a considerable distance in CA time and space – the structures that are meaningful to human observers. We know that rule 213 produces the wrong type of patterns, but still has a high MI with rule 130.

It seems plausible that rule 213 appears well-matched because the information used to evaluate its MI is too local: looking down on the CA, we can see that the high and low level lines cross at right angles, but this is far less apparent from the tiny blocks we use to calculate the MI. There is no temporal component to our measure of mutual information as we only use the current time, and the blocks we use have a pretty limited spatial component too.

So we could extend the mutual information measure to include more space and time. Until now the state space has been divided into blocks three coarse cells long (so six at the fine level for $g = 2$)

and one cell (one timestep) high.[4] Going forward, this dissection is described as having an *x-block* of 3 and a *y-block* of 1.

Calculating the MI for a different block size is a generalisation of the process described in §11.4. Consider a model with *x*-block = 4, *y*-block = 2, an *x*-chunk of the whole row and *y*-chunk = 2.

- Initialise and run the fine and coarse CAs as described before.
- Divide the state space into rows 2 blocks high – the chunks for this calculation.
- Divide each row into blocks of 4 × 2 cells (8 × 4 cells at the fine level, with two rows ignored as they don't map to the coarse level).
- Count the number of blocks in each state at the fine and coarse levels.

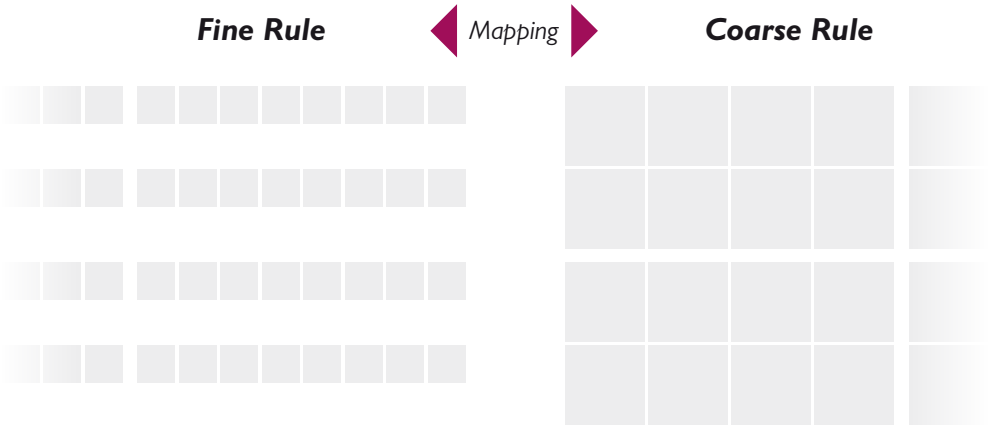


Figure 12.22 An example block with *x*-block = 4 and *y*-block = 2. Two of the rows at the fine level are ignored.

Using all of the block states within each chunk, calculate the fine and coarse level entropies and mutual information for the chunk.

By expanding the block size, we hope to see an improvement in our ability to distinguish between good coarse grainings and poor ones that have a high MI at small block sizes.

Using rule 130 and the initial condition described in §12.8, we calculated the MI for coarse rules 34, 128 and 213 with block sizes 1 × 1 to 15 × 4. Moving to a larger block size evidently introduces more possible states and therefore permits higher MIs, so the results of a 4 × 2 block are not directly comparable with a 3 × 1 block. Rather, we are interested in whether rule 213's MI (as the poor rule) increases more slowly than that of 34 (good mapping) or 128 (perfect mapping) as the block size grows in space and time.

Unfortunately this is not what we see – the MI of all three rules increases at roughly the same rate. There is actually very little difference between the MIs' relative values for block sizes larger than 2 × 1, and even 2 × 1 is sometimes sufficient. (At 1 × 1 the MI results are often unreliable.) We see the same proportionate increase in MIs for other rules too.

4 As discussed in §11.3, the intermediate rows at the fine level do not map to anything at the coarse level.

We gain similarly little by keeping the *x*-block constant and varying the *y*-block, with the MIs again increasing approximately in step. It is worth noting that the level of detail we can discern decreases quite markedly as the *y*-block rises; in particular, the hump when the diagonal lines of rules 130 and 213 cross in Figure 12.2 is significantly less evident when *y*-block = 4 than when *y*-block = 1.

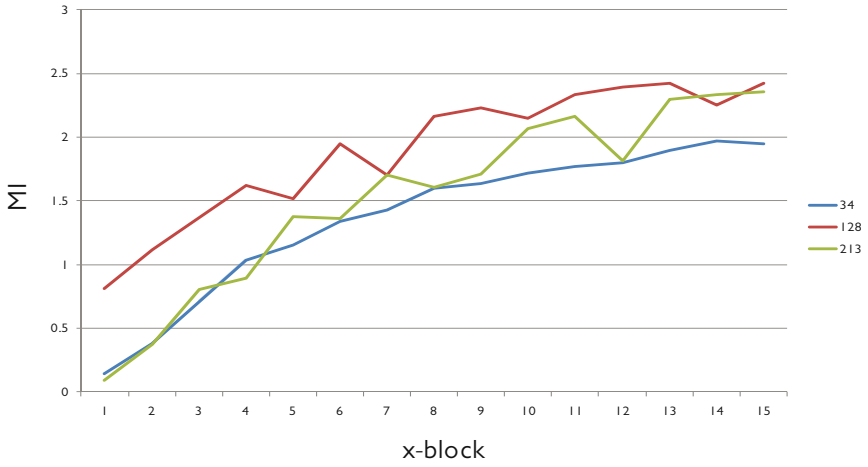


Figure 12.23 The MI between rule 130 and rules 34, 128 and 213 for various *x*-blocks. The *y*-block is always 1 and the same initial condition (detailed in §12.8) was used throughout. The graph shows the mean MI, calculated over timesteps 1-49. We see the MI of each rule increasing roughly in step.

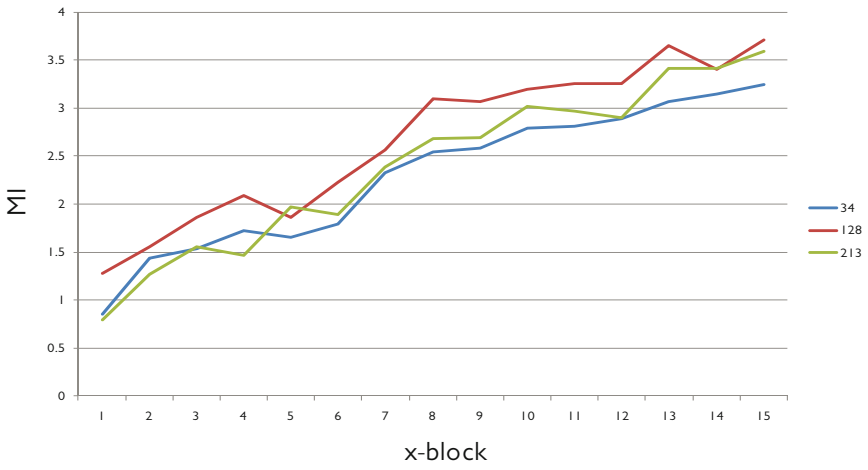


Figure 12.24 MI between rule 130 and rules 34, 128 and 213 for various *x*-blocks. The *y*-block is 4 for this graph. The initial condition is identical to Figure 12.23, as was the method for calculating the MI. Again, we see the MI of the rules increasing together.

The next graphs show the MI between rules 130 and 34 and rules 130 and 213 for *y*-blocks 1, 2 and 4. The *x*-block remains fixed at 3 throughout. The level of detail in the plots decreases quite markedly as the *y*-block rises; in particular, the hump when the diagonal lines of rules 130 and 213 cross again is significantly less evident with *y*-block = 4 than *y*-block = 1.

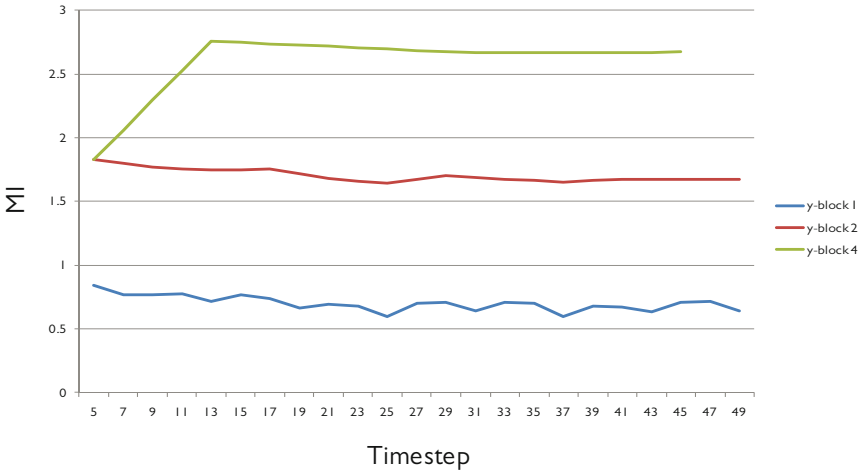


Figure 12.25 MI between rules 130 and 34 for various *y*-blocks. The *x*-block is 3 for this graph.

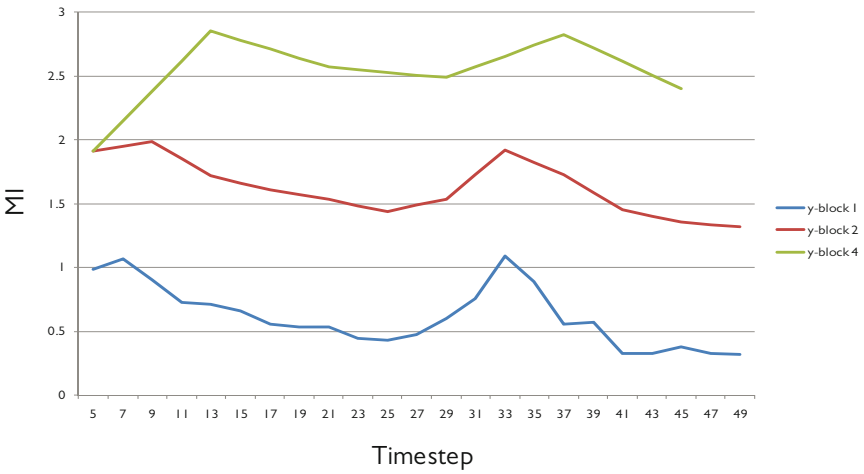


Figure 12.26 MI between rules 130 and 213 for various *y*-blocks. Again, the *x*-block is fixed at 3.

12.18 True temporal mutual information

Though we have now incorporated time into the MI calculation, our measure is still dominantly spatial: we still consider the CA space row by row. We can also flip our measurement grid through 90° and calculate the MI for each column in space, just as we have done until now for each row in time.

Figure 12.27 Rotating the CAs through 90° to calculate temporal mutual information.

Unlike an individual CA, a coarse graining pair is not rotationally symmetric: while all of the fine columns correspond to cells at the high level through the mapping, only half of the fine rows do (§11.3). But this doesn't affect the ratio of fine to coarse cells in MI calculations, so we can perform an analogous calculation by adjusting the pattern of cells included in each fine block.

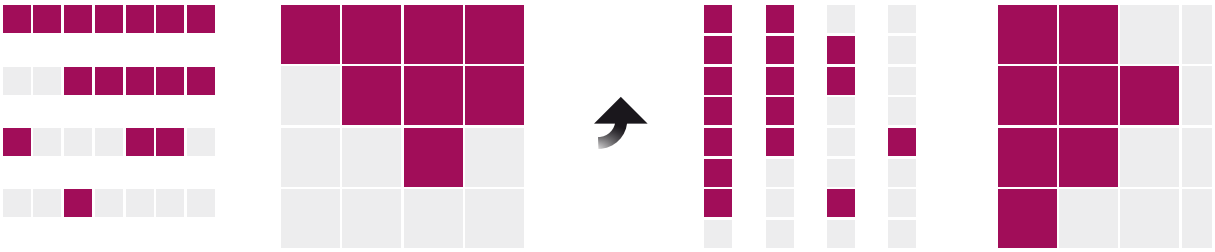


Figure 12.28 As the CAs are rotated by 90°, we need to exclude different cells when calculating temporal MI.

If we use feature extraction on rules 130, 34, 128 and 213, we see similar results to spatial coarse graining. Here we have used a *y*-chunk of 16 (equivalent to the *x*-chunk in spatial MI) for a total of four vertical chunks and an *x*-chunk of 4. We started the CAs from the initial condition described in §12.8. The large, regular triangles made by the initial condition are clearly visible in Figure 12.30: their edges have high MI, with low or zero values inside and outside those ridges. Figure 12.29, showing the MI between rules 130 and 34, picks up on the diagonal lines of 34. Unfortunately, the results from rules 130 and 213 in Figure 12.31 show that rule 213's MI remains high when calculated through this method.

0.00	0.00	0.00	0.00	0.16	1.72	1.83	1.61	0.25	0.00	0.00	0.00	0.00
0.00	0.00	0.00	1.31	1.95	1.68	0.57	0.00	0.00	0.00	0.00	0.00	0.87
0.00	0.25	1.55	1.92	1.01	0.05	0.00	0.00	0.00	0.00	0.48	1.82	1.92
1.24	1.88	1.44	0.32	0.00	0.00	0.00	0.00	0.15	1.54	1.90	1.54	0.30

Figure 12.29 Results of coarse graining rule 130 to rule 34 using temporal MI. The initial condition discussed in §12.8 was used and rule 34 wasn't restarted. We can see the diagonal lines emanating from the single cells in between the long runs of ■s that 34 is able to model from the beginning.

1.69	0.16	0.00	0.00	0.16	1.69	0.48	0.00	0.92	1.40	0.00	0.00	1.41
0.00	1.31	1.31	1.31	1.31	0.00	0.00	0.00	0.00	0.15	1.62	1.00	0.12
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 12.30 Results of coarse graining rule 130 to rule 128 using temporal MI. Here we see the edges of the triangles formed by rules 130 and 128. Note that the lines of rule 34 come from between the triangles.

0.84	0.00	0.00	0.00	0.00	0.00	1.57	1.72	1.41	0.92	0.00	0.00	0.00
0.99	1.49	0.44	0.00	0.00	0.00	0.00	0.36	0.91	1.03	1.68	0.44	0.00
0.64	1.03	1.67	1.40	0.00	0.00	0.00	0.00	0.05	0.64	1.24	1.92	2.02
0.00	0.78	1.66	1.03	0.82	0.21	0.00	0.00	0.00	0.00	0.78	1.75	1.07

Figure 12.31 Temporal MI results of coarse graining rule 130 to rule 213. Like rule 34, rule 213 latches on to the single cells in between the triangles, though its diagonal lines move to the right over time. The MI values are very close to those of rule 34.

12.19 Analysis of temporal mutual information

This approach works, but the results are very similar to those we got when calculating MI spatially. We have learned nothing new about the rules and have no new way to distinguish between the rules we want and those we don't. Rule 213's MI is still greater than 34's, and by approximately the same amount as we saw before.

It's worth noting that the initial conditions in this chapter – including the one used here – were specifically designed to be uninteresting horizontally: we wanted predictable, repeating behaviour in space to give a clean phase transition. Many rules have more interesting structures over space, but in general this approach seems more limiting than spatial coarse graining with an appropriate chunking resolution. The reason for this is simple: stuff happens over time. There may be changes over space too, but this is because of changes over time. Changes over time can happen without changes over space, but changes over space cannot happen without changes over time.

At a more practical level, we can obtain results and make decisions while the simulation is running if we measure over time. We also note that the example above is spectacularly bad at catching the

phase transition as the state space is divided into four large chunks temporally. Now clearly we could increase the vertical resolution, but – once we have changed the *y*-chunk down to one or two, and increased the *x*-chunk in turn to allow useful statistical calculations (§12.8) – there comes a point at which spatial MI looks like the better choice.

12.20 Analysis of different block shapes and sizes

Increasing the block size in spatial mutual information seems to offer little benefit (at least once we go larger than 2 × 1), and temporal MI offers similarly little additional insight while imposing some unfortunate limitations.

On reflection, these results are not surprising. Rules 34 and 213 draw diagonal lines. After flipping the state space through 90°, they still draw diagonal lines at the same relative positions to each other. Time and space are essentially isomorphic for these rules, so temporal MI won't offer new ways to distinguish between them.

Figure 12.32 Rules 34 (left) and 140 (right). Rule 140 draws vertical lines, so there is no post-transition temporal information in the run.

Temporal MI would be even less effective with rules coarse grained from rule 140, as there is almost no non-transient temporal information in the rule (it draws vertical lines). This is an example of the issue raised in §12.19: rules that undergo phase changes must retain some information that can be seen when calculating MI spatially (or there would be no repeating element), but they do not have to retain any that can be seen when doing so temporally.

Changing the block size also didn't help us distinguish between good and bad rules because the block size wasn't increased enough: even a 15 × 4 block (the largest considered here) is comfortably subsumed within the crossing lines rule 213 uses to get its high MI. We would need to move to block sizes closer to 50 × 25 to stop this happening for the initial condition used here, and potentially much larger blocks in other cases. So despite using a significantly larger block, we still aren't able to distinguish CA objects in any useful manner.

The graphs also show another problem with larger block sizes: loss of resolution. The *y*-block = 4 series in Figure 12.26 shows significantly less detail than the *y*-block = 1 series, missing out almost entirely on the bump when rule 213's lines cross with 130's lines. Similarly, the transition point with *y*-block can only be given as between times 25 and 33, whereas we have been able to state it

far more precisely as 25-27 before. We see the same loss of resolution spatially as we increase the *x*-block, though this is not an issue here as the CA uses a repeating pattern throughout (and we have divided the width into four large chunks).

The question of what *x*-block and *y*-block values to use is somewhat problem specific. Here we care very little about horizontal resolution and care very much about the vertical resolution as we attempt to pinpoint the transition point. But if we had a CA whose spatial behaviour we wanted to track precisely, we now know that using a block of 1 × 3 would be equivalent to using one of 3 × 1 (in terms of the rules' relative MIs) and would afford us a significantly higher horizontal resolution.

12.21 Extra entropy: a better distinction

We know that rule 213 is a poor match for rule 130, but its MI still manages to equal that of rules such as 34 that appear to us, as observers, to capture aspects of the underlying rule much better. And we have just seen that calculating MI using larger blocks does very little to reduce rule 213's apparent goodness.

The problem with rule 213 is that there is too much going on. There is too much other stuff happening that doesn't match very well and that we don't like, alongside aspects that correlate well with the underlying rule. (If we draw only the parts of 213 that match the fine rule, it actually appears a pretty decent choice.)

Or, to put it another way, there is too much *extra entropy* in rule 213. In a total coarse graining, all of the coarse level behaviour forms part of the mapping between the rules and there is no extra entropy. This is not the case for all partial coarse grainings, but the vast majority of the high level behaviour must map onto the lower level for us to consider it a good coarse graining. (Otherwise the high level's unmapped behaviour would tend to dominate, as it does with 213.) So a good partial coarse graining, in addition to having a high MI, should have a low extra entropy.

$$\text{Extra Entropy} = H(C) - \text{MI}$$

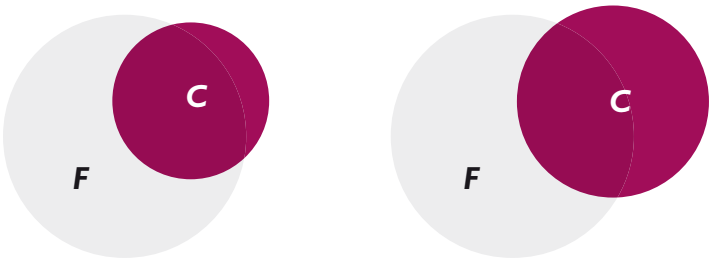


Figure 12.33 Both these diagrams have the same mutual information, but the coarse rule in the right diagram has considerably more extra entropy than the coarse rule in the left diagram.

With an *x*-block of 3 (and a *y*-block of 1), rule 34's extra entropy doesn't venture much above 40% of the MI value, with a mean of 25% over the run. For *x*-blocks of 4, 5 and 6, the mean extra en-

tropy improves to 4-7%. In the case of 213, the extra entropy gets as high as 509% of the MI at *x*-block = 3, with a mean of 156%. (Again this improves as the *x*-block increases, but only to 189% and 49% respectively by *x*-block = 6.)

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	258.3%	120.8%	41.6%	13.8%	10.8%	12.1%
Mean	161.2%	75.9%	25.4%	6.9%	4.0%	3.3%

Figure 12.34 Extra entropy when coarse graining rule 130 to rule 34 for various *x*-blocks. The *y*-block is always 1.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	52030.4%	603.5%	509.0%	220.1%	140.4%	189.0%
Mean	11451.7%	305.7%	156.2%	77.2%	45.2%	48.9%

Figure 12.35 Extra entropy when coarse graining rule 130 to rule 213 for various *x*-blocks. The *y*-block is always 1.

These large differences in extra entropies aren't limited to this pair of rules either: we see in §12.26 and §12.27 that extra entropy tallies very closely with how good a rule appears subjectively to us, as high level observers.

These are significant differences, and being able to use extra entropy as an additional discriminator (alongside MI) is significant too. As well as being easy to obtain – we already have these figures from the MI calculation – this method has the distinct advantage of being local. Mutual information is so useful because it allows us to take a multidimensional, global question – does this large-scale pattern match this other large-scale pattern? – and reduce it to a single value, calculated locally.

We have not had to consider a plethora of special cases or expensive exceptions, nor faced prediction horizons from exponentially complex calculations, because the local MI calculations scale linearly with space and time. Like Reynold's Boids (§8.2), we do not know or care what is happening globally, and we don't have to allow for any number of obscure vagaries in the environment. It just works.

12.22 Liberal coarse graining

We loosened our criteria for a valid (partial) coarse graining for the experiments described §12.26 and §12.27 by using the union of the results of all mappings, rather than intersecting them as before. Earlier we justified intersection as an effective quality filter, but we can more easily 'get away' with this here as we are only considering one fine rule at once.

We do get significantly more results for each fine rule, including a substantial number of subjectively poor ones. But we also get a number of interesting and perhaps unexpected results. We have seen that rule 130 coarse grains to rules 34 and 128, which effectively model the underlying rule's behaviour during its transient and post-transition phases. But the union of all coarse grainings of rule 130 also includes rule 162, which models the entire range of the rule's behaviour fairly accurately (and with high MI).

This illustrates the "mostly correct, most of the time" ethos of neutral emergence again. Arguably we have missed out on the best coarse graining for rule 130 by intersecting our result set. But we also excluded 68 poor rules and did manage to include 34 and 128, both of which are good matches. Usually intersection is the better choice as it acts as an effective quality filter, but there are times when taking the union of the returned results is more appropriate.

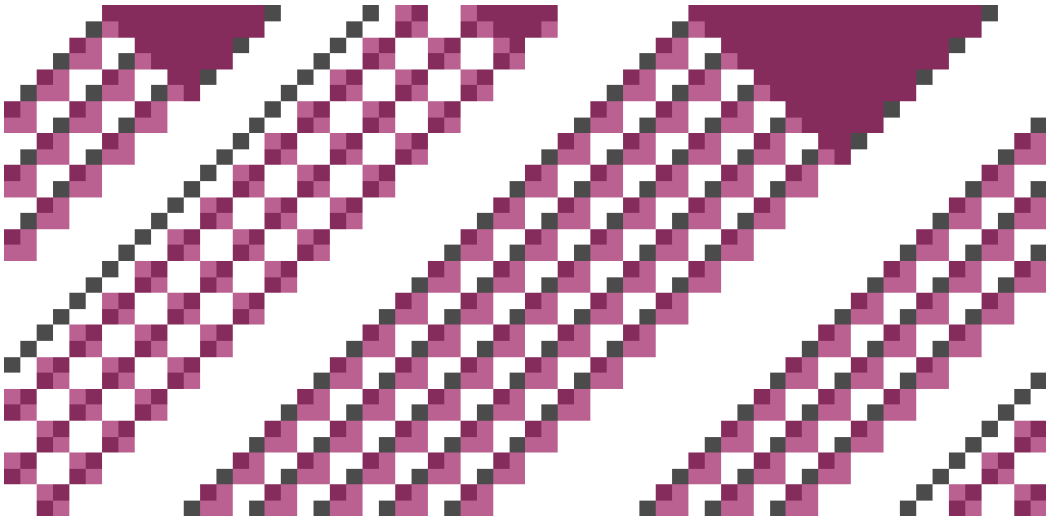


Figure 12.36 Rule 162 is probably the best coarse graining for rule 130. It captures more of the underlying pattern than other coarse rules and has little uncorrelated behaviour, yet is excluded by taking the intersection of the results when coarse graining as we have done until now. We conjecture that this is an unfortunate consequence of the "mostly correct, most of the time" ethos of neutral emergence (see §12.22).

12.23 Applying extra entropy to other coarse grainings

We have used extra entropy to judge the quality of the coarse grainings of rules 130, 140, 43 and 192 and saw the same pattern within the results in each case. Results for 130 and 140 are included here; the other rules' results can be found in §E.

The tables show the relative difference between the coarse entropy and the mutual information for each rule over the run. In each case, the mean and maximum values are given. For the absolute figures from which they were calculated, see §E.

$$\text{Extra Entropy}(t) = (\text{H}(C, t) - \text{MI}(t)) \,/\, \text{MI}(t)$$

$$\text{Mean Extra Entropy} = \sum_{t=1}^{n}(\text{Extra Entropy}(t)) \,/\, n \times 100\%$$

$$\text{Max Extra Entropy} = \text{Max}_{t=1}^{n}(\text{Extra Entropy}(t)) \times 100\%$$

12.24 Graphs of rule 130's extra entropy

Rule 130 draws triangles with diagonal lines coming out of their left sides. We have already seen that rules 34 and 162 match rule 130 closely, whereas rule 213 does not.

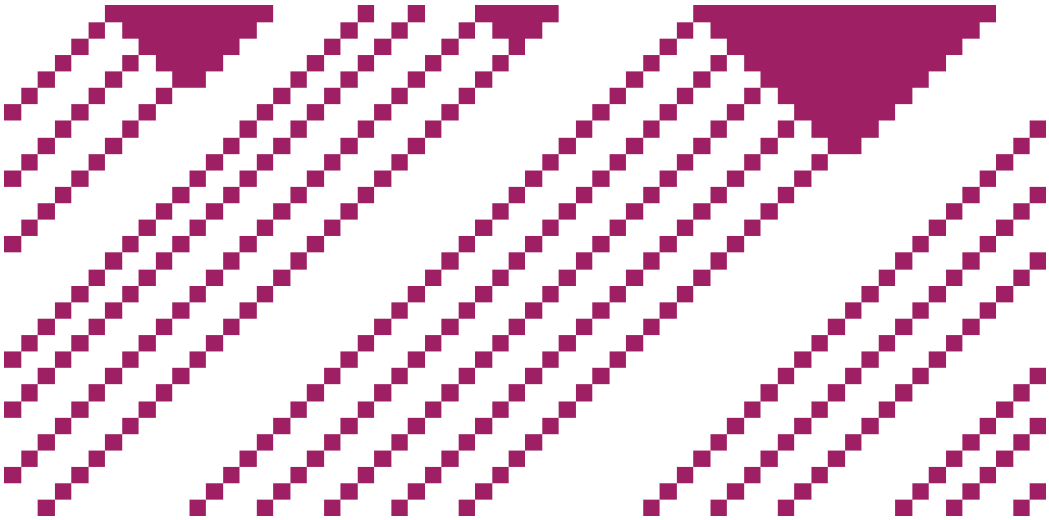


Figure 12.37 Rule 130

Figure 12.38 shows the (absolute) extra entropy during a run of rule 130 coarse grained to rule 34 for various *x*-blocks. We used the initial condition in §12.8, which shows a phase transition at time 25-27. The size of the extra entropy increases initially as the coarse rule 34 diverges from 130's transient triangles before settling at an approximately constant extra entropy once the phase transition has taken place.

The difference with an *x*-block of one or two is significantly larger than larger *x*-block sizes. In fact the relationship appears to be logarithmic: Figure 12.39 shows the line spacing to be roughly equal on that scale. (Results for *x*-blocks larger than 4 × 1 aren't shown on this graph as noise dominates the results.)

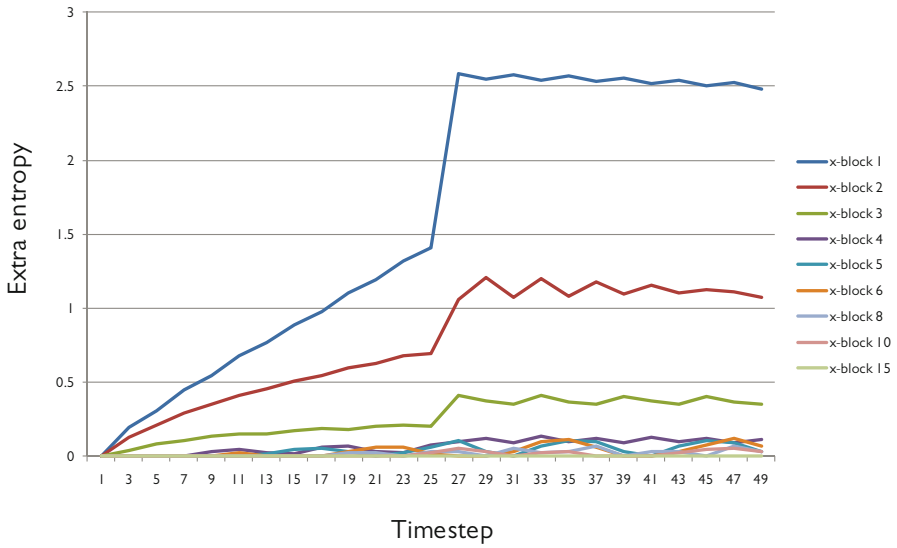


Figure 12.38 Extra entropy for rule 130 coarse grained to rule 34 for various *x*-blocks.

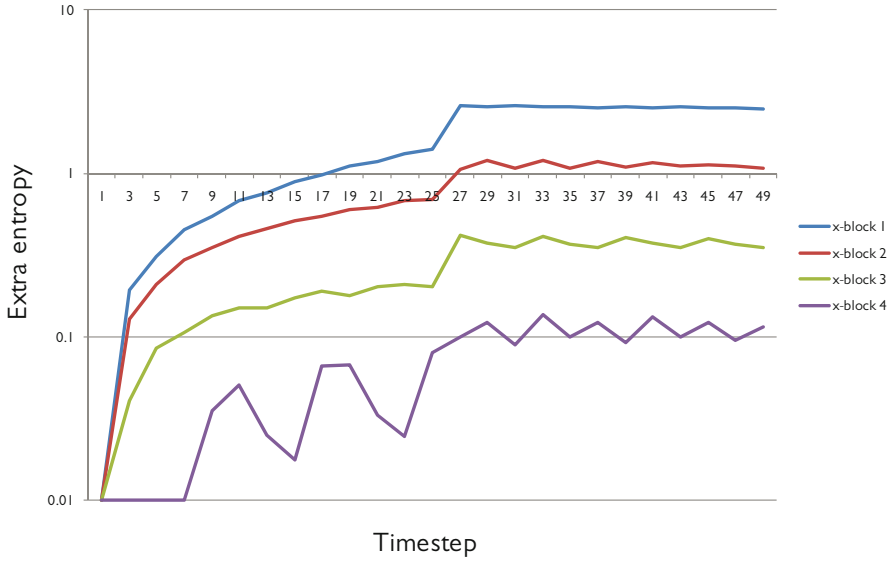


Figure 12.39 Extra entropy for rule 130 coarse grained to rule 34 on a logarithmic scale. Noise dominates for *x*-blocks larger than 4 × 1.

We see the same pattern for another good coarse graining of 130 to 162. Again, there is an increase in extra entropy during the transient phase before moving to a constant value (Figure 12.40). The lines of the logarithmic graph in Figure 12.41 aren't quite so regularly spaced as Figure 12.39.

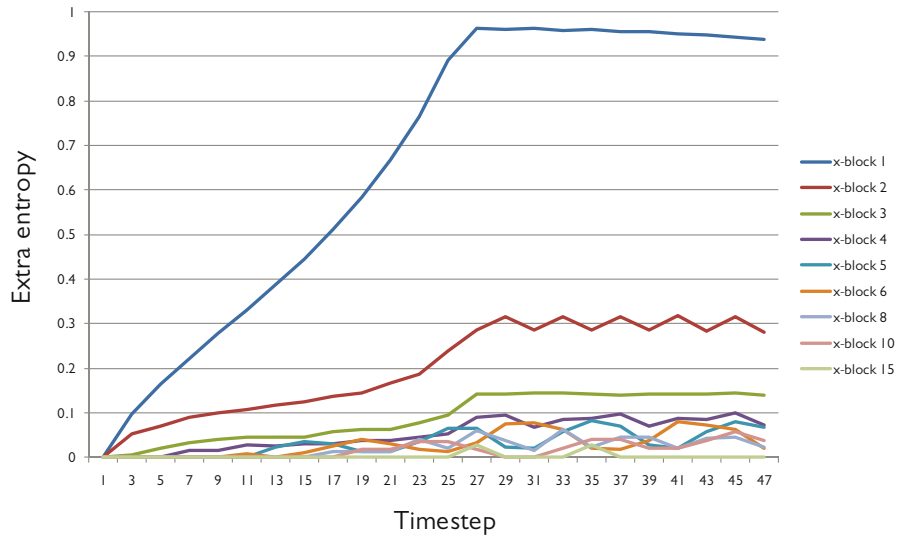


Figure 12.40 Extra entropy for rule 130 coarse grained to rule 162.

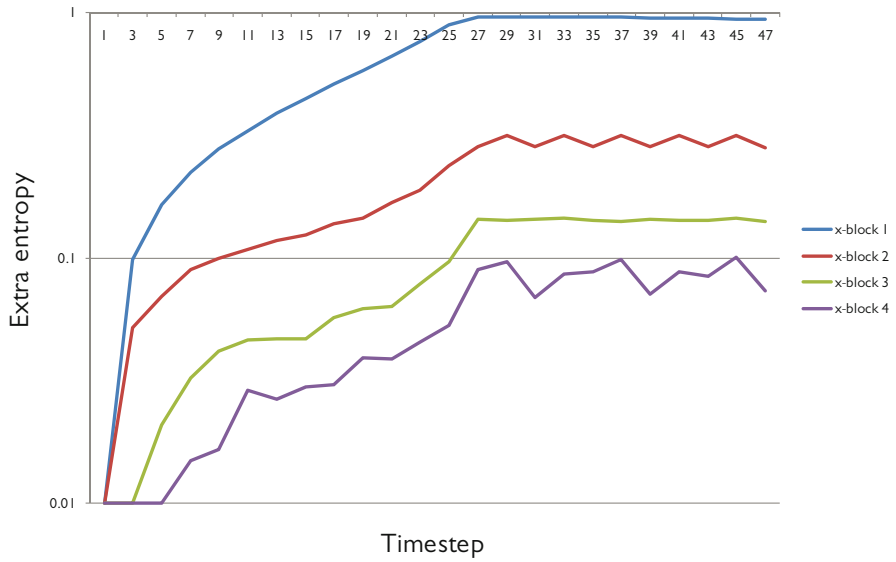


Figure 12.41 Extra entropy for rule 130 coarse grained to rule 162 on a logarithmic scale. Again noise dominates for *x*-blocks larger than 4 × 1.

Rule 213's graphs are much less regular. The extra entropy for block 1 × 1 is vast (Figure 12.42) and appears to increase throughout the run (though the peaks and troughs on the curve could be due to vagaries in the calculation). If we remove the *x*-block = 1 results (Figure 12.43), we see a rather irregular curve, again rising during 130's transient phase before peaking and dropping into a valley when 213's lines cross 130's. After that point, the extra entropy rises quickly again. The logarithmic plot (Figure 12.44) is also far less regular, with the lines seeming much less in consort than we saw for rules 34 and 162. As the extra entropy for 213 is much higher than the well matched rules, we are able to include logarithmic plots of larger *x*-blocks without noise dominating.

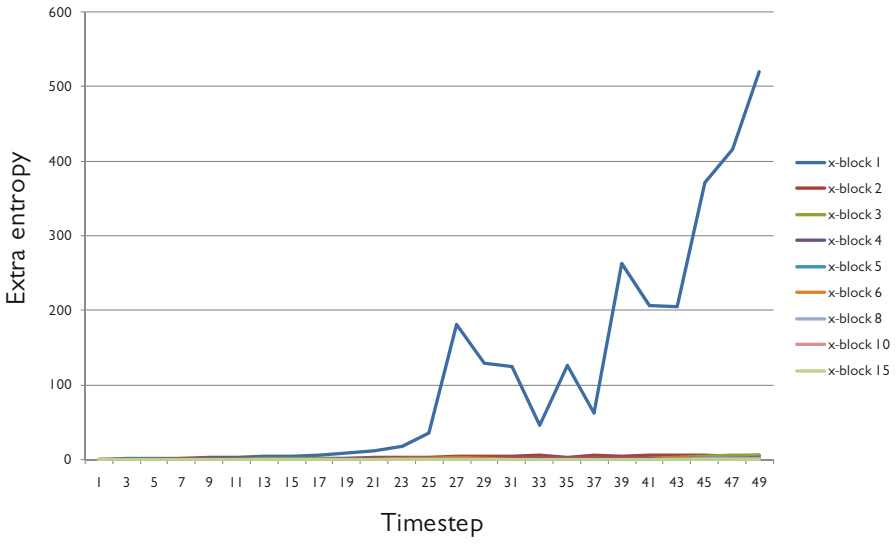


Figure 12.42 Extra entropy for rule 130 coarse grained to rule 213. The graph is dominated by the *x*-block = 1 results.

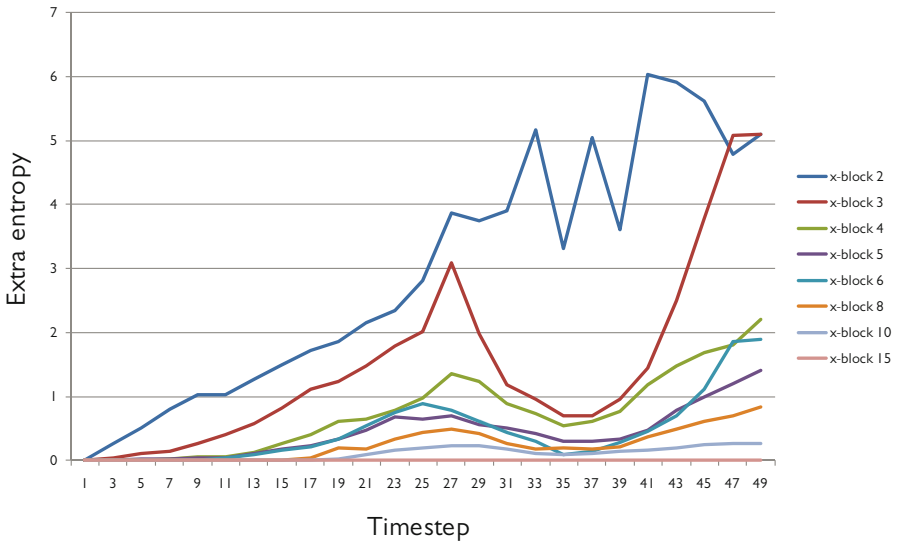


Figure 12.43 Extra entropy for rule 130 coarse grained to rule 213 without *x*-block = 1 results.

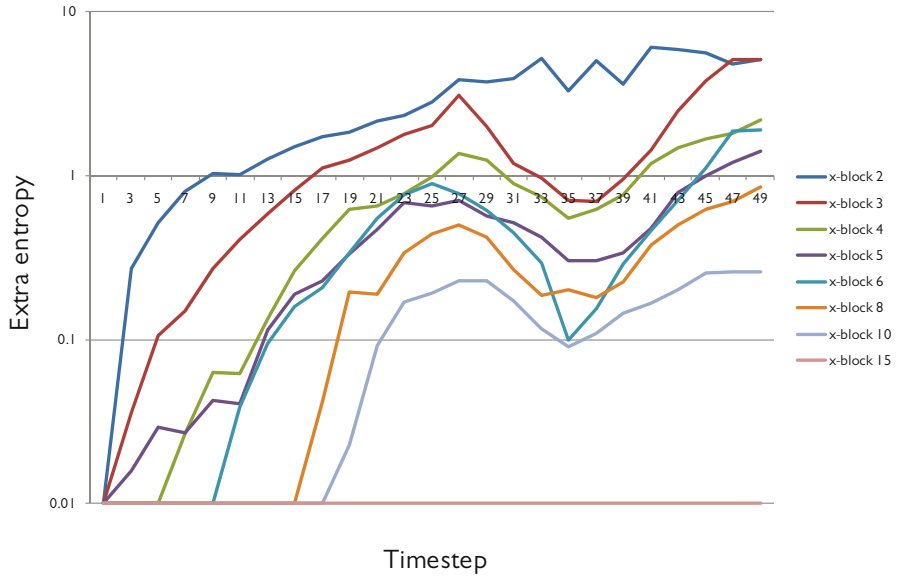


Figure 12.44 Extra entropy for rule 130 coarse grained to rule 213 on a logarithmic scale. Noise does not dominate at larger block sizes because of the larger extra entropy, so we can include more results.

12.25 The best block size

We see in §12.26 that the difference between the entropy and MI is significantly greater than the MI for *x*-block = 1 for all coarse grainings of rule 130. Results from *x*-block = 2 are also high in most cases. There is substantial variation in the results reported for these *x*-blocks as the absolute MIs are often quite low, so a small increase in the absolute difference can give rise to a big percentage change. In particular, noise can have a significant impact on the reported results.

This, and the graphs in §12.24, suggest that a block size of 3 × 1 or 4 × 1 may well be the best choice at $g = 2$: by that stage, we have moved beyond the small blocks that give high extra entropy (even with good rules), but stand to gain little by changing to even larger blocks due to exponentially decreasing extra entropy drops.

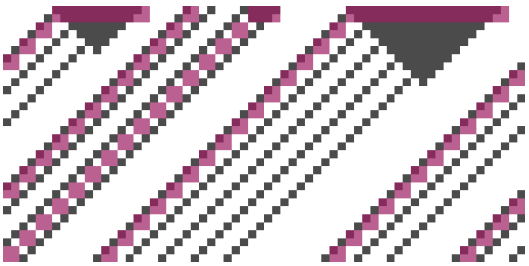
Plus, if we are trying to use extra entropy as a discriminator, it makes sense to use it at a level where the entropy itself, and not noise, dominates. When the *x*-block size gets larger than 6, and certainly by 10, we start to see a drop off in the utility of this measure.

12.26 Extra entropy of rule 130's coarse grainings

These tables show the maximum and mean extra entropy between rule 130 and its coarse grainings for different *x*-blocks over 51-step runs with the initial condition in §12.8. The *y*-block is one for all runs.

12.26.1 Rule 34

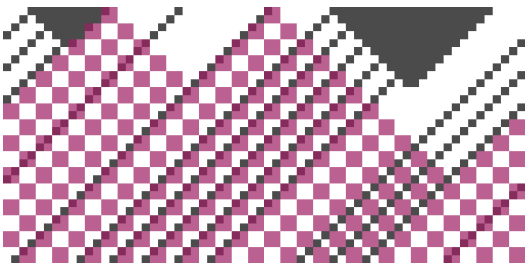
Diagonal lines that tally with those made by rule 130. A good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	258.3%	120.8%	41.6%	13.8%	10.8%	12.1%	6.9%	5.6%	0.0%
Mean	161.2%	75.9%	25.4%	6.9%	4.0%	3.3%	1.9%	1.3%	0.0%

12.26.2 Rule 50

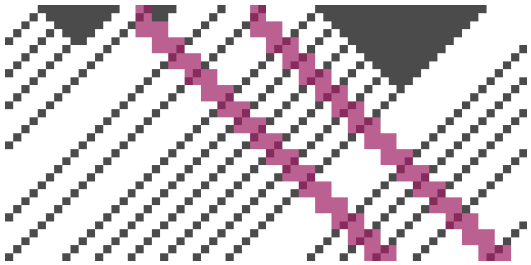
Chequered pattern starting from single nodes and expanding outwards. Not a good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	1275.6%	285.9%	181.6%	96.7%	79.8%	45.8%	40.9%	16.1%	5.8%
Mean	523.0%	147.6%	89.0%	42.9%	39.5%	26.2%	17.5%	8.5%	1.1%

12.26.3 Rule 84

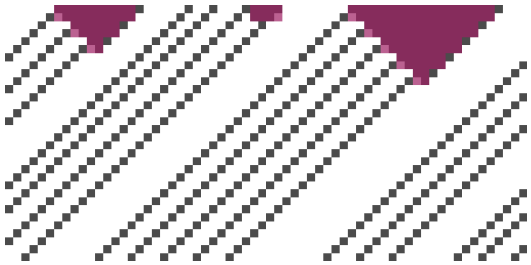
Thick diagonal lines in the opposite direction from rule 130's lines. Not a good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	26603.9%	748.1%	308.2%	203.7%	152.3%	138.9%	88.0%	26.8%	3.6%
Mean	6314.4%	315.2%	119.5%	72.6%	49.3%	39.2%	22.9%	9.6%	0.9%

12.26.4 Rule 128

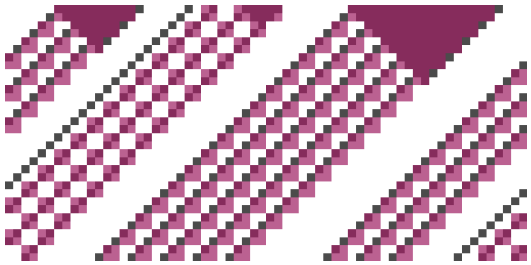
Triangles that correspond with those seen in rule 130. A good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Mean	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

12.26.5 Rule 162

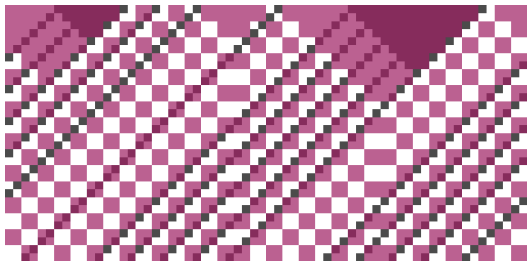
Triangles and lines that correspond to rule 130. A good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	96.2%	31.8%	14.6%	10.1%	8.4%	7.9%	6.3%	5.7%	2.9%
Mean	66.0%	20.1%	9.1%	5.3%	3.3%	3.0%	2.2%	1.7%	0.2%

12.26.6 Rule 179

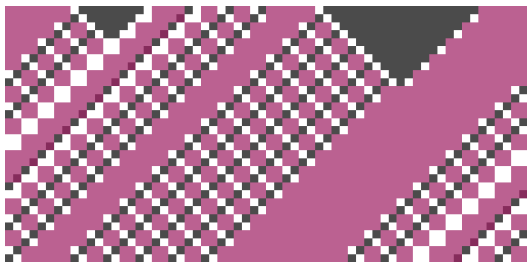
Triangles overlaid on areas of contiguous ■s and □s in the initial condition (subsuming rule 130's triangles). Chequered pattern elsewhere. Not a good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	3205.9%	475.7%	214.8%	117.3%	83.2%	73.6%	50.7%	24.4%	6.3%
Mean	978.8%	237.3%	115.1%	64.5%	42.3%	37.7%	24.0%	11.3%	1.2%

12.26.7 Rule 186

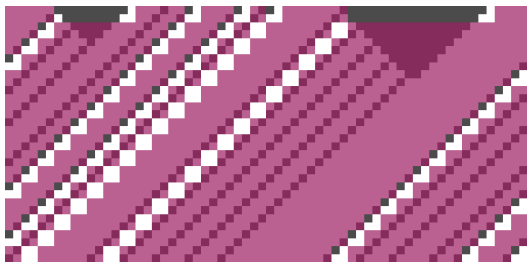
Inverse of rule 162. A good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	64.4%	24.5%	6.5%	3.4%	0.0%	0.0%	0.0%	0.0%	0.0%
Mean	45.4%	14.2%	3.0%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%

12.26.8 Rule 187

Inverse of rule 34. A good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	290.8%	148.7%	46.5%	23.0%	14.0%	13.1%	7.7%	8.4%	2.3%
Mean	212.7%	86.6%	28.1%	12.1%	7.9%	6.4%	4.7%	2.9%	0.8%

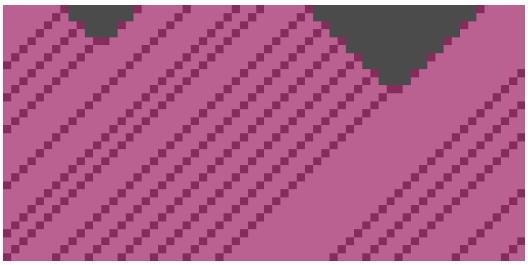
12.26.9 Rule 213

Inverse of rule 84. Not a good match.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	52030.4%	603.5%	509.0%	220.1%	140.4%	189.0%	84.6%	26.1%	6.0%
Mean	11451.7%	305.7%	136.2%	77.2%	45.2%	48.9%	25.9%	11.3%	1.0%

12.26.10 Rule 254

Inverse of rule 128. A good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Mean	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

12.26.11 Analysis of rule 130's coarse grainings

We see a clear distinction between well- and poorly-matched rules: the well-correlated rules have mean differences between 0 and 25% with poorly-matched grainings coming in at several times that figure. Rule 128 and its inverse 254 are total coarse grainings of rule 130 and therefore their MIs and coarse entropies are identical (yielding 0 extra entropy).

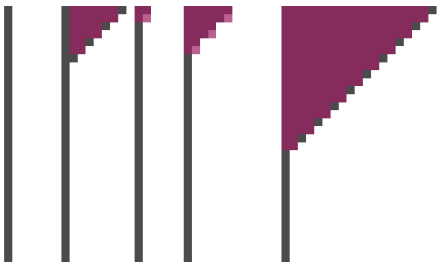
12.27 Extra entropy of rule 140's coarse grainings

Rule 140 draws right-angled triangles with vertical lines coming out of their base and from single blocks in the starting state.

As with the results above, these tables show the maximum and mean extra entropy for different *x*-blocks over 51-step runs. The *y*-block is one for all runs.

12.27.1 Rule 136

Corresponds to fine triangles in rule 140. A good match and total coarse graining.



x-block	1	2	3	4	5	6
Max	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Mean	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

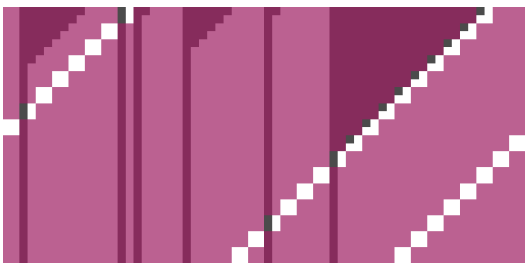
12.27.2 Rule 140

Covers fine rule 140 well, catching most vertical lines. A good match.

x-block	1	2	3	4	5	6
Max	153.6%	66.4%	33.9%	17.1%	6.9%	6.8%
Mean	83.7%	38.9%	19.8%	10.5%	4.1%	3.8%

12.27.3 Rule 184

Dominant feature is diagonal lines in same direction as rule 140's triangle edges. Not a good match.



x-block	1	2	3	4	5	6
Max	23648.8%	788.1%	344.9%	203.8%	121.4%	78.3%
Mean	3900.8%	283.6%	110.6%	62.4%	38.2%	25.0%

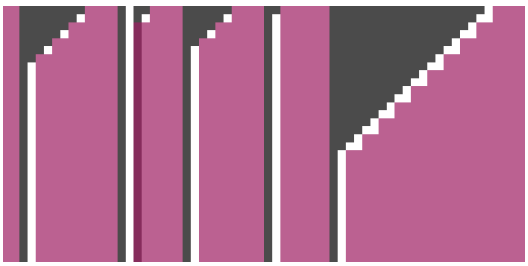
12.27.4 Rule 204

Straight lines approximately the inverse of rule 140's lines. A good match.

x-block	1	2	3	4	5	6
Max	1480.8%	87.1%	29.4%	6.6%	6.9%	3.3%
Mean	898.6%	66.7%	22.4%	5.5%	5.1%	2.3%

12.27.5 Rule 206

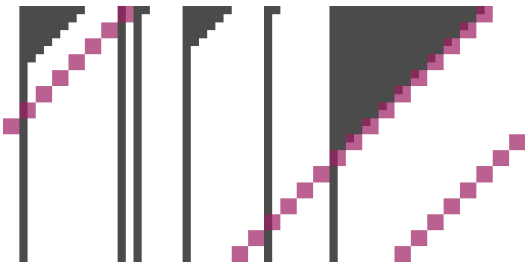
Inverse of rule 140. A good match.



x-block	1	2	3	4	5	6
Max	96.8%	55.9%	32.5%	15.6%	8.4%	7.5%
Mean	58.2%	31.6%	17.9%	6.7%	1.9%	2.0%

12.27.6 Rule 226

Diagonal lines following edge of triangle and starting at other points. Not a good match.



x-block	1	2	3	4	5	6
Max	2888.9%	832.3%	523.4%	412.3%	216.4%	128.3%
Mean	1089.8%	294.1%	128.7%	73.4%	40.1%	25.6%

12.27.7 Rule 238

Inverse of rule 136 and a total coarse graining. A good match.

x-block	1	2	3	4	5	6
Max	0%	0%	0%	0%	0%	0%
Mean	0%	0%	0%	0%	0%	0%

12.27.8 Rule 252

Inverse triangles in wrong direction. Not a good match.

x-block	1	2	3	4	5	6
Max	1886.6%	340.1%	152.8%	70.3%	49.9%	31.7%
Mean	562.4%	154.1%	73.5%	32.9%	21.4%	14.1%

12.27.9 Analysis of rule 140's coarse grainings

As with rule 130, the well-matched rules show low mean difference percentages at around 20% of the MI for *x*-block = 3, while the poorer rules have much larger differences, approaching an order of magnitude more. The split is quite stark.

12.28 Analysis of extra entropy

Despite being a local measure, extra entropy allows us to observe large scale, high level features we naturally identify when looking at a CA and find high level rules that capture these features. We have seen a close correlation between the level of extra entropy and our subjective judgement of the quality of a coarse graining over a variety of fine and coarse rules.

In short extra entropy works very well, but it is important to highlight a couple of limitations of this technique

- It must be used alongside MI, or we would fall into the trap of selecting a (total) coarse graining that models a minimum of the underlying rule's behaviour perfectly (in particular, rule 0 would always be a great match).
- It should only be used when we seek to model a particular subset of the underlying rule's behaviour accurately. This is a good technique for directed coarse graining (§12.5, §12.29), but there may be other rules that capture more behaviour and that are rejected because they also model too much other behaviour that is novel.

12.29 DIRECTED COARSE GRAINING

Despite their conceptual simplicity, elementary CAs display a remarkable range of varied and sometimes complex behaviour. At least one ECA has equivalent power to a Turing machine [70], and we have seen that they show interesting emergent behaviour, including phase transitions.

There are obviously easier ways of finding patterns such as triangles in ECAs or gliders in Life than the approaches we have developed over the last few chapters, but we believe we can use these concepts (total and partial coarse graining, maximising mutual information, minimising extra entropy) with larger, more complex and less well-defined systems – the sorts of systems encountered when developing real systems that interact with the real world – just as examples of emergence, complexity, order and chaos are found in both ECAs and other systems.

We now know how to use mutual information and extra entropy together to find emergent models that capture a lot of the underlying behaviour. This chapter provides the last step towards finding the emergent behaviour we want, and developing emergent systems automatically.

12.30 Finding behaviour of interest

As observers, we don't discuss the patterns produced by a CA in terms of rules; instead, we talk of lines and triangles and other shapes. Directed coarse graining should be a process that starts with a command like, "Find me something that models triangles like those at a high level," and ends with a coarse rule that captures that behaviour.

Suppose we are using rule 130 and have shown the run in Figure 12.45 to our observer.

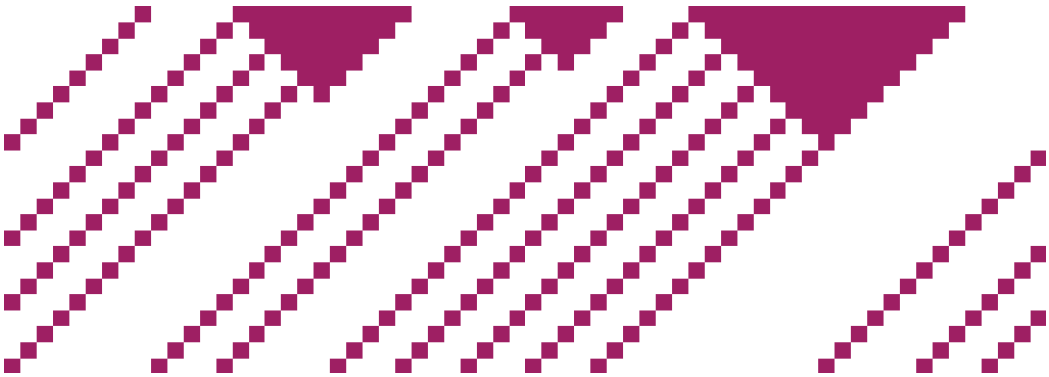


Figure 12.45 Rule 130

The observer uses a drawing package to highlight the areas of the run that are of interest. In this case, the triangles are highlighted.[5]

5 The details of the highlighting process are unimportant here, but once complete we know which squares the observer is interested in for this particular run.

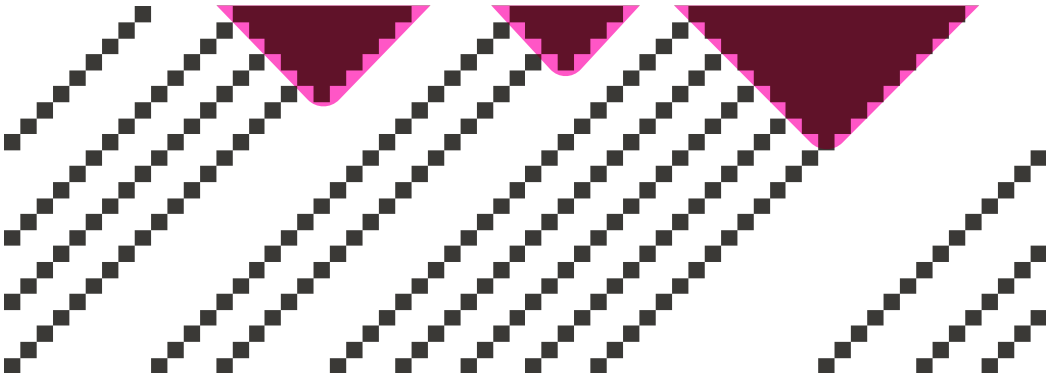


Figure 12.46 Highlighted triangles on rule 130.

Rule 130 has two states, ■■■ and □□■, that generate a ■ at the next timestep; the rest go to □. State ■■■ behaves exactly like rule 128: when three ■ cells occur next to each other, the output cell is also ■, but if the cells are on a boundary between ■ and □ (■■□ or □■■) then the output is □. This behaviour draws the distinctive triangles seen here and in rule 128. □□■ outputs a ■ if the input cell one step to the right is ■ and both other input cells are □. This part of the rule draws diagonal lines from an initial □□■. Requiring the other two cells to be □ sets the minimum spacing possible between lines, seen in Figure 12.46. It should be apparent that the behaviour the observer wishes to model is that of rule 128. We need to keep the ■■■ → ■ rule case and change the □□■ → ■ case, and we now detail the steps to do so.

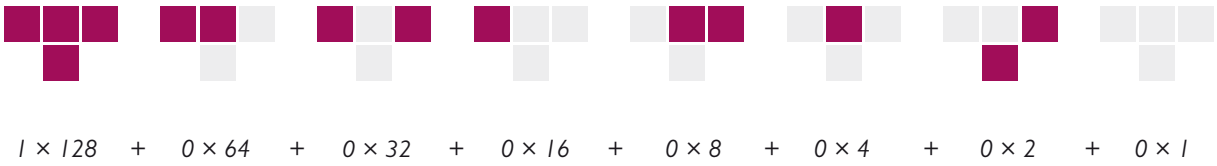


Figure 12.47 Rule 130

12.31 Creating an exception to a rule

- Examine the whole CA run, looking for places where the rule's behaviour differs from the highlighted area. As just explained, we see that every case where □□■ → ■ diverges from the result we want, and changing it to □ corrects the problem in all instances. (We are only looking at the fine level at this stage.)
- Create an exception in the rule, overriding □□■ so that it now maps to □. We could just remap □□■ to □ in this case, but we shall see presently that exceptions are more general than this simple example shows.
- Coarse grain the rule plus exception. We use the same coarse graining procedure as before, except that, in cases where the fine rule contains □□■, we follow the exception and return □.

Despite its simplicity, this example introduces a couple of important – and generalisable – concepts. We have outlined a way to move from the large scale objects of interest to observers to the rules that can be used by CAs. Recasting the problem in terms of rules allows us to abstract away from this specific instance and to a general form that is applicable to other similar cases. We also introduced the idea of exceptions.

12.32 Exceptions

The exception in the example above changed the output of the triple □□■ from ■ to □, effectively changing the rule from 130 to 128. But exceptions can be much more expressive: their purpose is to restrict the behaviour just to those aspects of the rule in which we are interested, something that will not always be possible by switching to another elementary rule. In such cases, we can expand the neighbourhood of the exception to include more cells.

Figure 12.48 Rule 138

Rule 138 draws diagonal lines that are (usually) the thickness of the initial condition, so starting with a block of five cells will (again usually) produce a diagonal line five cells wide.[6]

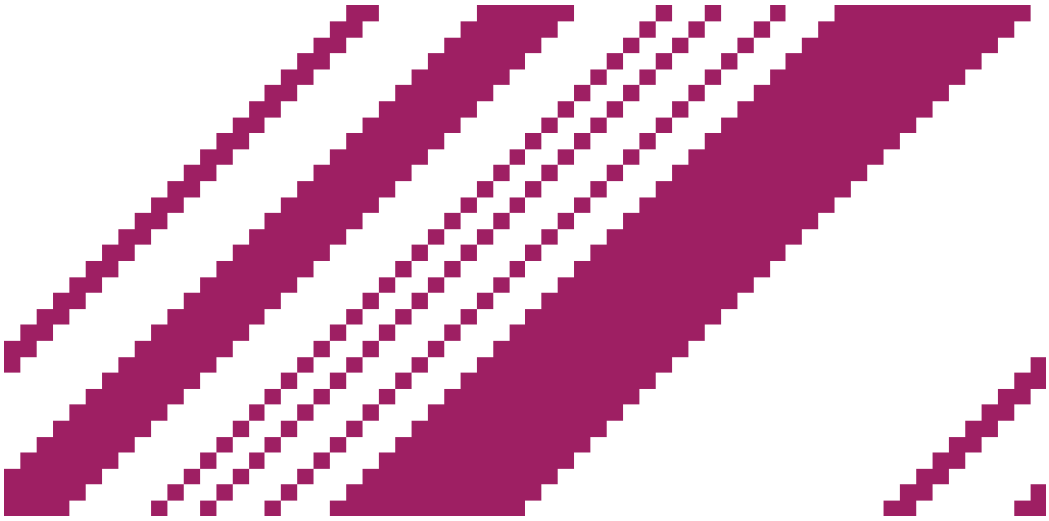


Figure 12.49 Rule 138

Rule 138 has just one different state from rule 130, mapping □■■ to ■ instead of □, which is sufficient to give the rule a significantly different output. As □■■ yields ■, we can duplicate the left

6 If there is a gap of just one cell between adjacent lines, the right line will be one cell narrower than the initial condition.

side of any thick lines, while ■■■ continues to copy the middle of the lines. ■ cells are still lost from the right side of the line because ■■□ maps to □, but they are added on the left side by □□■ → ■, giving constant width, diagonal lines.

Suppose we only wanted to include lines that are two cells thick or wider. There is no elementary rule that behaves like this (there can't be as the neighbourhood is too small), but we can model this behaviour if we increase the scope of the exception to a neighbourhood of five cells.

A single width line will match the input state □□■, and we wish to change (or filter) the CA's output if the cell to the right of the ■ is a □ from a ■ to a □. (Of course a single line input may also match □■□, ■□□ and ■□■, but these states already output □.)

Specifically, we are trying to distinguish between input states with the pattern x□□■□ and the pattern x□□■■. We only need to override the rule in the first case. Of course there are two possible values for x, so we introduce the exceptions

□□□■□ → □

■□□■□ → □

Figure 12.50 Rule 138 with the exceptions □■□□□ → □ and □■□□■ → □. Note that the single width lines now disappear immediately.

12.33 Sparse exceptions

The 'rule plus exception' is equivalent to a 1D CA with neighbourhood five but with almost all of its states set to their defaults for a neighbourhood three CA. We continue to refer to these as exceptions because of the sparseness of the rule set – we define only a handful of extra rules of out a possible 1024 – and because we seek only to modify the elementary CA's behaviour slightly and don't come close to exploiting the additional possibilities of a larger neighbourhood (for instance by doubling the speed of light – §2.3).

This is an important practical point for engineering emergent systems. There are only 256 ECA rules, but, by expanding the neighbourhood to five, we move to a possible 5.29×10^9 rules in the model space. Examining that many possibilities is feasible (indeed we have done so when looking at $g = 4$ coarse grainings in §11.11), but it takes significant time. Limiting ourselves to standard ECAs plus a few exceptions makes the process nearly as efficient as before, while giving us substantial additional options for controlling and directing the search.

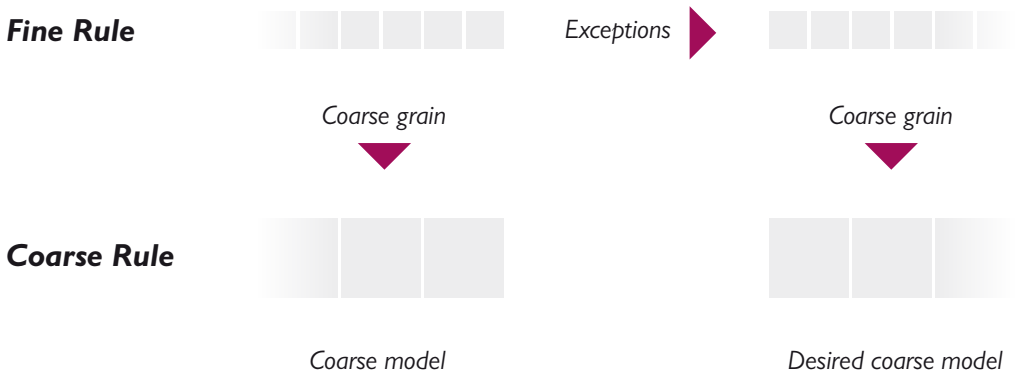


Figure 12.51 Exceptions in coarse graining. Adding exceptions to the fine rule lets us find the coarse model of the system we want.

12.34 Exceptions at the coarse level

We use exceptions exclusively with fine rules in this section, and it seems logical to ask why we have not applied them to the coarse rule as well. This is a perfectly reasonable thing to do, but it does not further our aims in this section.

We are trying to direct the coarse graining towards aspects of the underlying model we find most interesting. There are two parts to this: changing the fine rule through exceptions and investigating the coarse grainings we get from this modified model. Adding exceptions to the coarse level instead would limit us to investigating just the former.

Coarse exceptions would probably allow us to find better high level models (particularly if we are trying to match rules with exceptions), but it is difficult to decide how expressive we should allow the coarse rules to become. Should we limit ourselves to a fixed number of exceptions? Presumably exceptions that exceed the speed of light (for the original CA) should not be allowed? But what if using such an exception allows us to capture interesting behaviour in two close locations at the fine level? There a danger that we would end up with a 'colouring the squares' approach to coarse graining, where we define an exception to counter every bit of unwanted behaviour and learn little about the underlying CA or coarse graining.

12.35 Coarse graining with exceptions

We get different results if we coarse grain rule 138 with and without the exception. If we select only the intersection of all mappings (§12.22), we get two valid coarse grainings (to 34 and 187) without the exception and no valid coarse grainings with it. By using the union of all mappings, we get 43 coarse grainings without the exception and 35 with it, of which 24 are common to both.

A single width line is clearly a fine grain feature, so it's not surprising that none of the unique coarse grainings highlight the exception clearly. The rule with an exception coarse grains to rule 10, whereas the rule without does not. The opposite is true for rule 98. By squinting hard it's perhaps possible to see how these match slightly better to their fine rules, but the results are not terribly convincing.

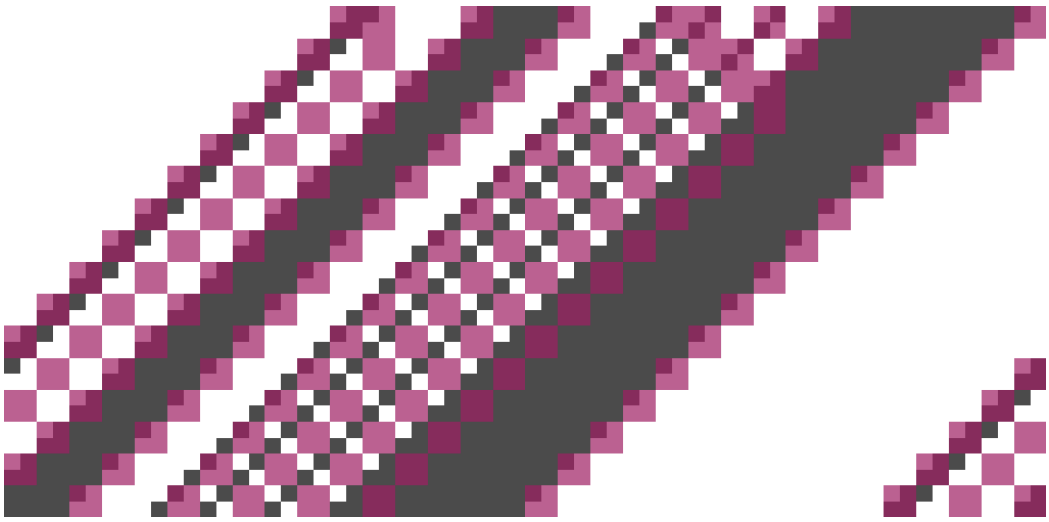


Figure 12.52 Rule 138 coarse grained to rule 98. Note that the coarse rule mirrors the single width, fine level lines.

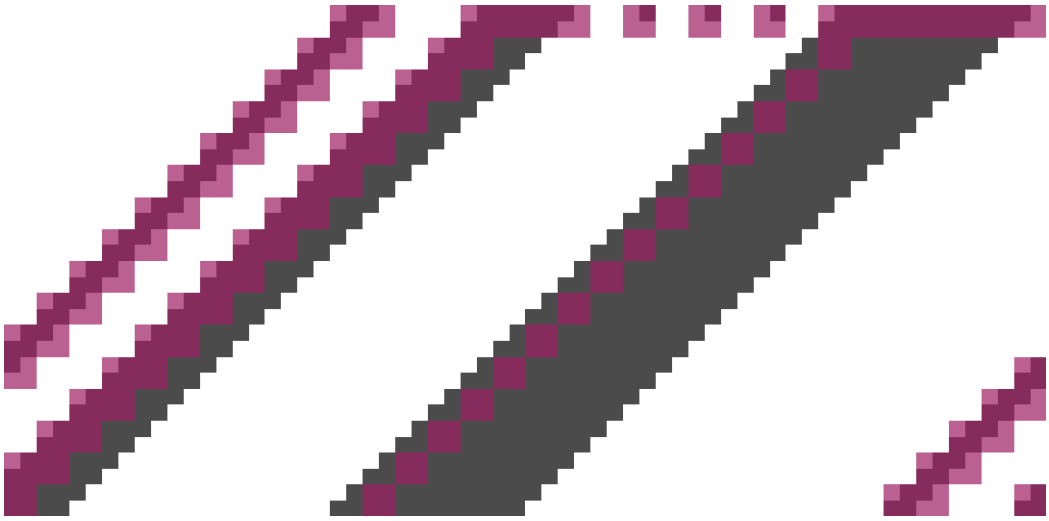


Figure 12.53 Rule 138 with exceptions coarse grained to rule 10. Note that the coarse rule stops immediately when matching single width, fine level lines (though we also note that it does the same with double width line in most cases, as shown here).

12.36 Another exception using rule 138

Suppose we want to model just the single width lines at the coarse level in rule 138 and that we want all other lines to taper down until they become one cell wide, as shown in Figure 12.54.

Figure 12.54 The behaviour we want to create in rule 138.

The three ■-producing rule cases in 138 are □□■, □■■ and ■■■. We know ■■■ on its own is rule 128 and that it will give us the triangles we want. So we just need to override the other two rule cases, but only when the line they are making a thick line (which must be on the right side of the input triple because both have a □ as their leftmost cell). With an exception of neighbourhood five, we need to override x□□■■ and x□■■■, which we do by adding the following exceptions to rule 138.

□□□■■ → □

□□■■■ → □

■□□■■ → □

■□■■■ → □

These exceptions give behaviours that look remarkably like rule 128 (triangles) and rule 34 (lines). Rule 34 is a valid coarse graining of the unmodified rule 138, but the only other rule to which is coarse grains (totally or partially with intersection) is 187, the inverse of 34. After coarse graining the modified rule, we find 34 is still a valid coarse graining, and we also find that it now coarse grains to 128 (and its inverse 254), even when we consider the intersection of the results.

Rule 128 is also a good coarse graining of the rule with exceptions – its MI is amongst the highest of all the rule-plus-exception's coarse grainings if we start it from an initial condition containing several contiguous strings of ■s. The point here isn't to prove that rule 128 is the best coarse graining in all circumstances for this modified rule – clearly it is not – but to show that coarse graining

a rule with exceptions can have as high an MI as coarse graining an unmodified rule that naturally correlates well with 128, such as 130 (e.g. Figure 12.45).

Figure 12.55 Rule 138 with this exception coarse grained to rule 128.

12.37 Adding an exception to a chaotic rule

Rule 102 draws fractal right-angled triangles. Starting from a single ■ cell, the pattern expands left at 45°, incorporating triangles that double in size as the run progresses.

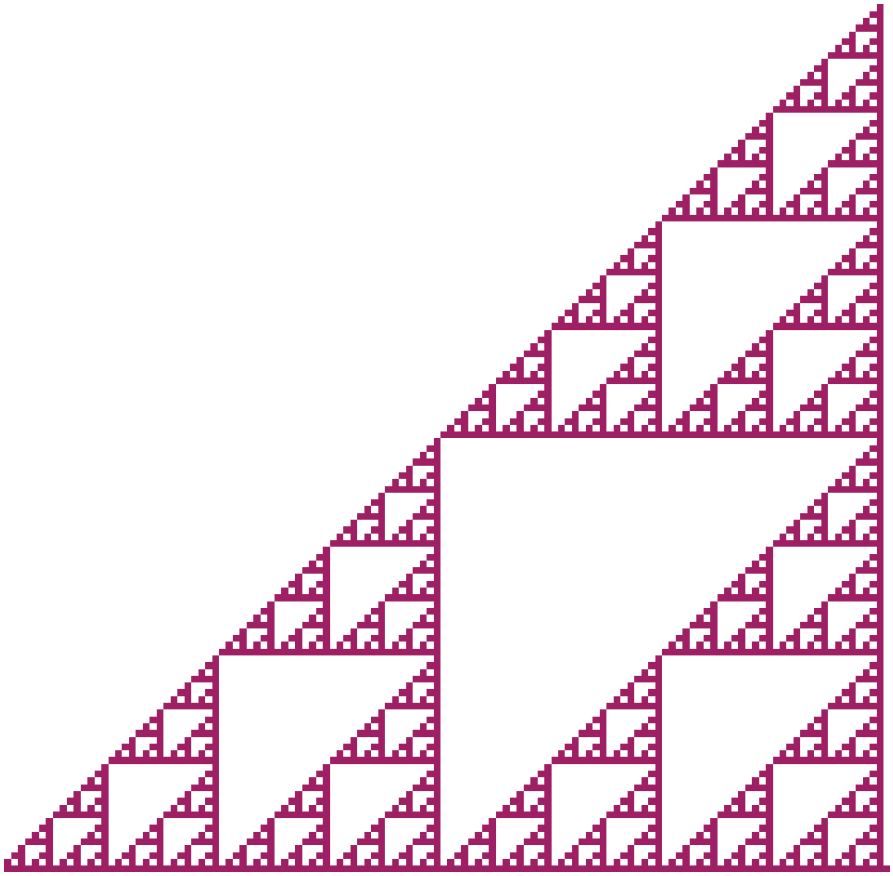


Figure 12.56 Rule 102, a chaotic rule that draws a series of ever-larger triangles.

We introduce the exception

□■■■■ → ■

This only has an effect on rows of cells that adjoin the right-angled corner of triangles and stops triangles that are five cells or wider one row early. This upsets the rule's output, as large triangles are no longer twice the height and width of smaller ones. The pattern it produces looks more interesting than the vanilla rule 102, and we shall see presently that it is actually complex.

The exception □■■■■ matches the side length of one of rule 102's triangles, but of course it also contains triangles that are twice, four, eight, etc. times that length, and indeed half that length. By halving the exception length to three[7] we get □■■, and effectively modify the rule from 102 to 110.

If we compare the output of rule 102 with the length five exception and rule 102 with the length three exception (which is rule 110), we notice that the rule plus larger exception seems to give exactly the same output pattern as the smaller one, only scaled up by a factor of two. In other words, each triangle in rule 110's output is now twice as wide and high as before.

7 The triangle sides length are one cell shorter than the exception as we also need to include the leftmost □ cell in the exception.

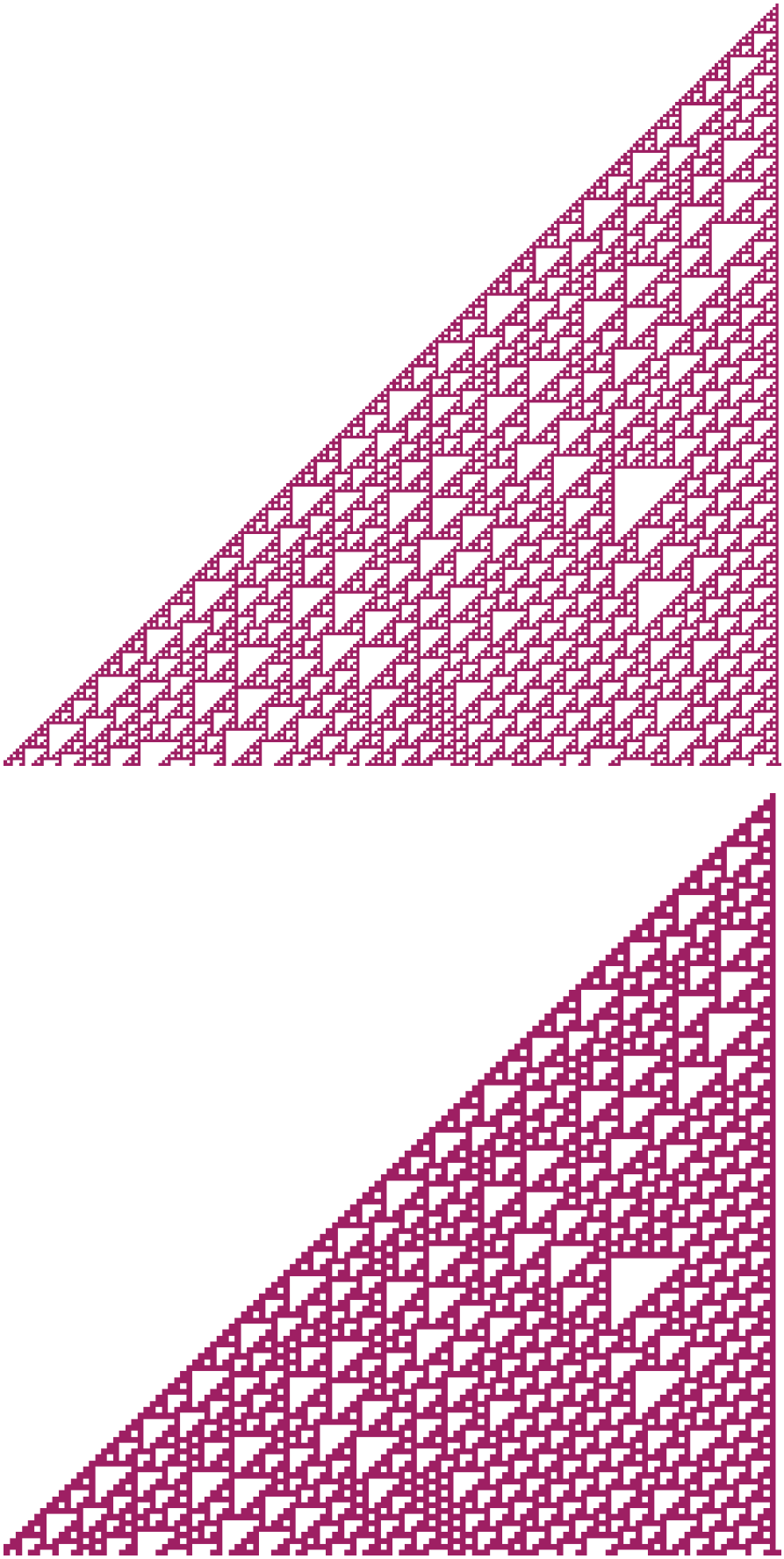


Figure 12.57 Comparing rule 102 plus exception (top) to rule 110 (bottom). The rule with exception follows the same pattern as rule 110, but scaled up by a factor of two.

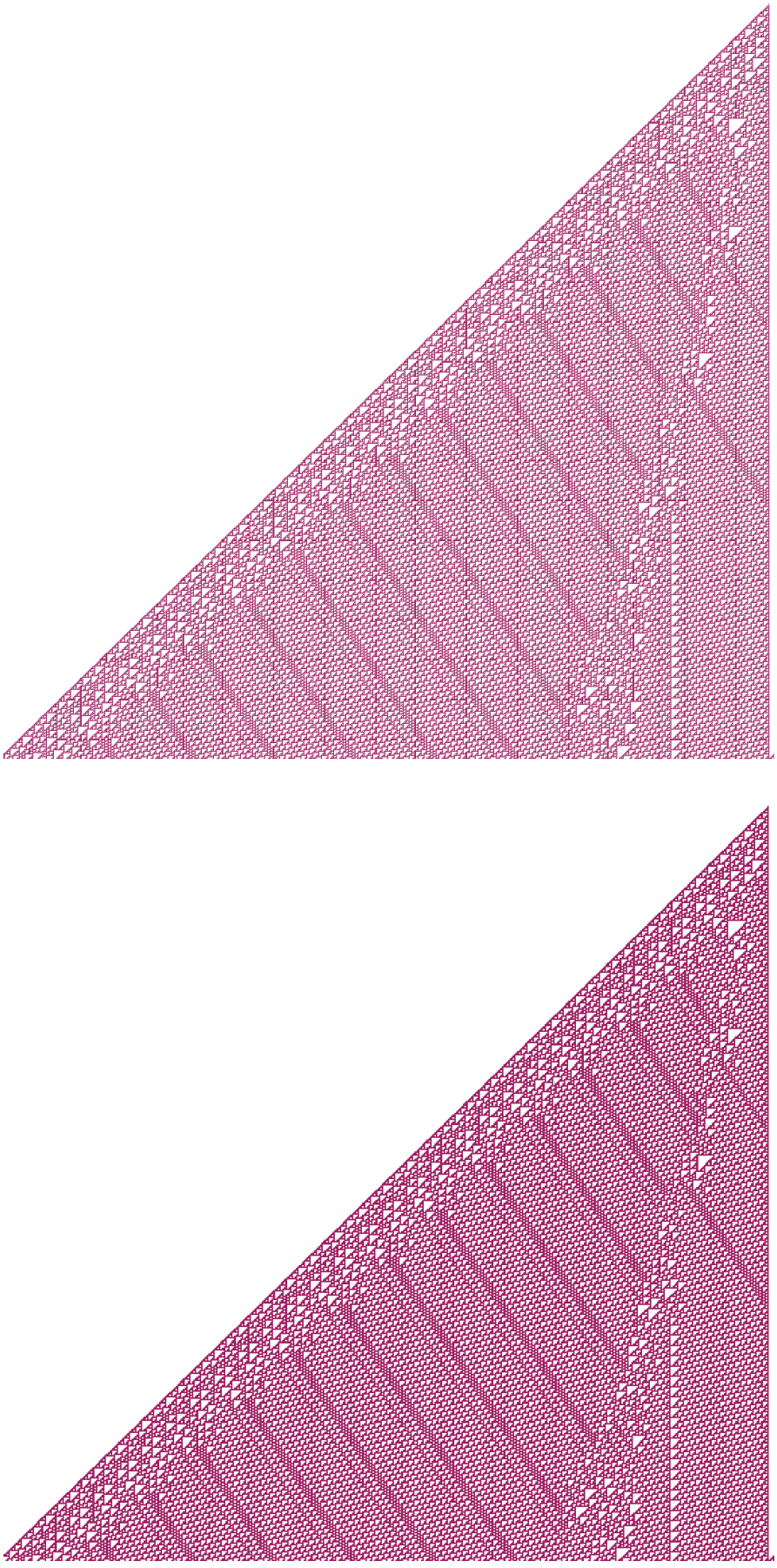


Figure 12.58 Larger CAs of rules 102 plus exception (top) and 110 (bottom).

This should mean that we can coarse grain this new rule-plus-exception to rule 110 (as they follow the same pattern), and indeed we can: superimposing the fine rule 102 plus exception and the coarse rule 110 shows a perfect match.

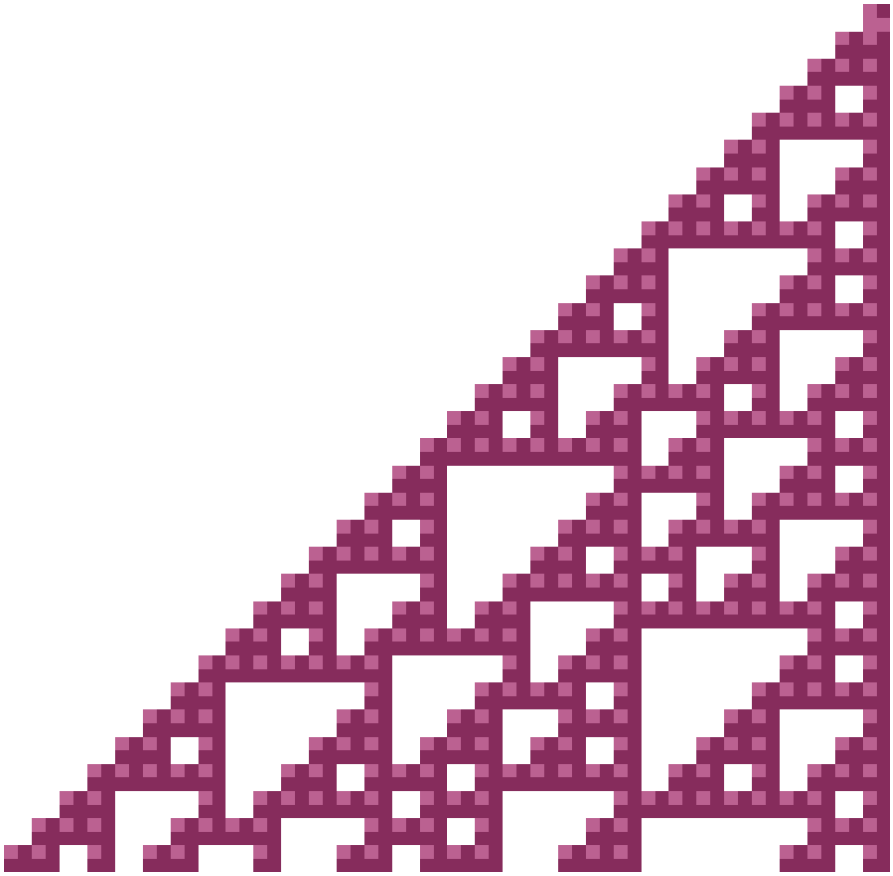


Figure 12.59 Rule 102 plus exception coarse grained (at $g = 2$) to rule 110. The coarse graining is total and shows a perfect match.

These pictures suggest that the complexity in rule 110 comes from disturbances (caused by the exception) to the regular pattern created by rule 102. With the three cell exception – or rule 110 – the grid has a granularity of one cell, but we see the underlying regular structure of rule 102 as we move to the five cell or larger exceptions, interrupted at intervals by rule 110's patterns.

As rule 102 consists of fractal triangles, it seems reasonable to ask if we can increase the size of the exception again from five to nine and double the scale of the 110 rule pattern again. This does indeed happen, and we continue to get the same doubling when we scale the exception further. (We have only tried a limited number of cases and have no proof that this pattern continues indefinitely, but it does seem reasonable to suggest it would, given the structure of rule 102.)

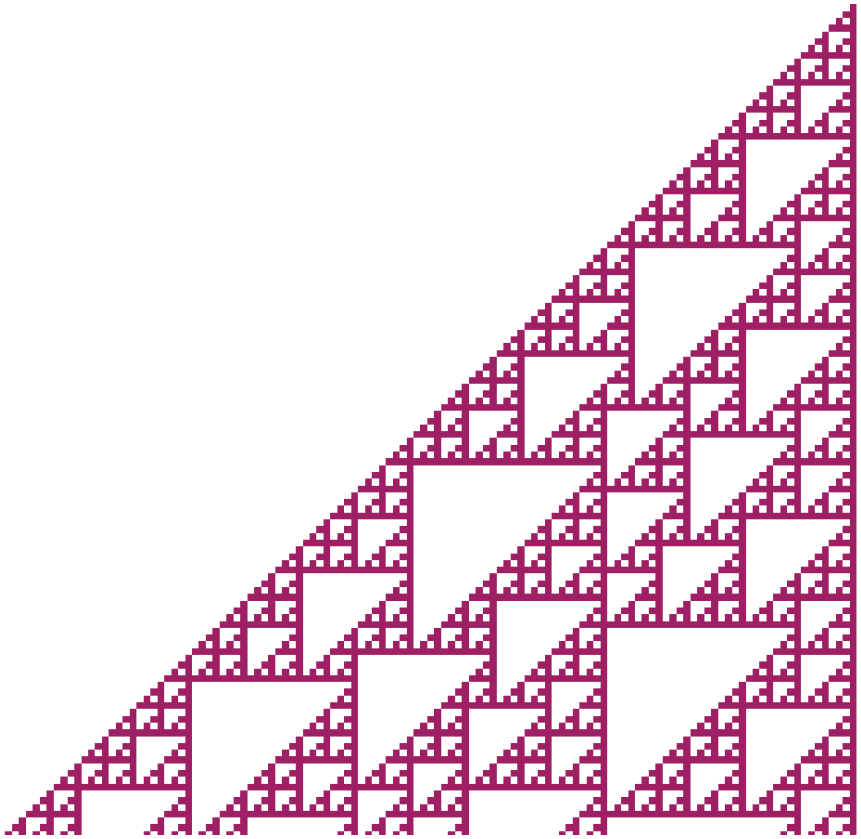


Figure 12.60 Rule 102 with a nine cell exception, showing the same pattern as Figure 12.59 with the scale doubled.

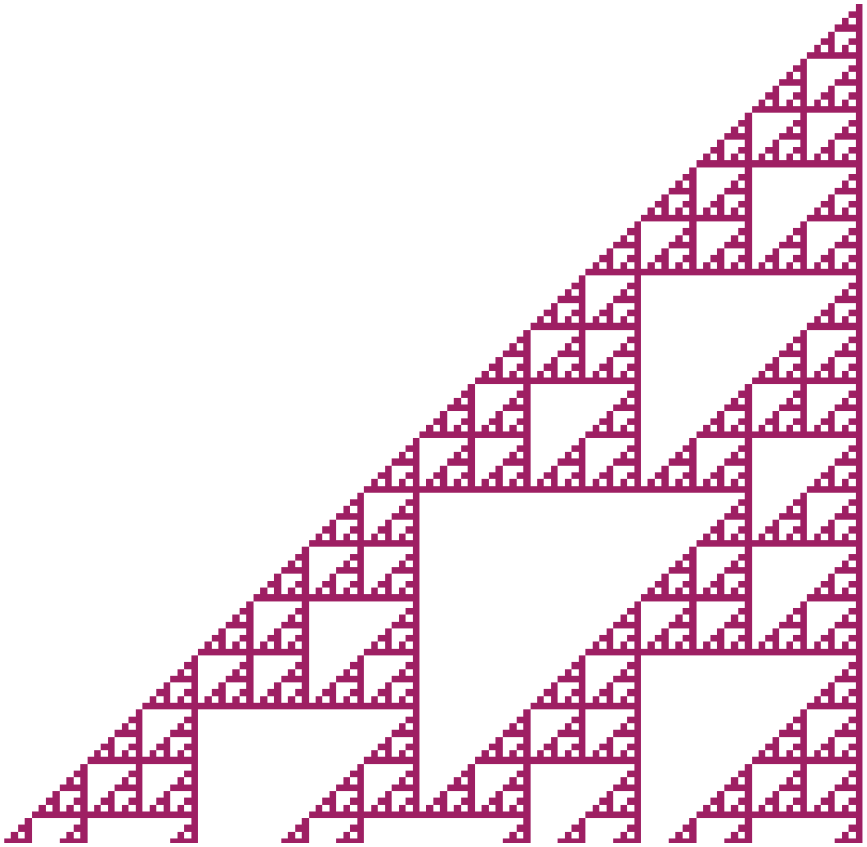


Figure 12.61 A section of rule 102 with a 17 cell exception, showing the same pattern as Figure 12.60 but with the scale doubled again.

12.38 Directed coarse graining through exceptions

Now we have added exceptions, we are able to do what we set out to do at the start of §10. We can do the equivalent of finding gliders in Life, but in a much more general and robust way. We can now select particular aspects of an underlying system that interest us and find a high level model of this behaviour.

This is directed coarse graining. We have shown, within the context of elementary CAs at least, that we can develop models of emergent systems automatically – that we can develop the models of emergent systems *we want* automatically. This is a key requirement for engineering emergence (§9.19), and a central tenet of neutral emergence (§9.9).

Mutual information allows us to see which models capture more of the underlying behaviour. Also considering extra entropy allows us to find models that not only capture the underlying behaviour well, but model it closely. Adding exceptions allows us to specify which particular aspects of the underlying behaviour we wish to capture well and model closely. And, just like many evolutionary algorithms, we can do all of this without knowing how to find better solutions; we merely need to recognise better solutions when we come across them.

The examples in this section were carefully selected. They had to be, because the high level language we are using (ECAs) is less expressive than the low level language (ECAs with exceptions), and there's only so much of this low level behaviour we can model. Exceptions at high level would give far more power, but this doesn't add much to the argument presented here: essentially, we would show that having a more expressive language means we can do more with it.

If we consider modelling a Life glider again, one of requirements for success is surely that the high level language is actually capable of modelling a glider. And while it is perhaps not always obvious what can be modelled by a coarse ECA a priori (or at least without experimenting for a while), it must be reasonable to assume that, in the development of any emergent system, we would choose high and low level languages that are expressive enough to model our desired behaviour.

12.39 Key points

- Feature extraction can be used to find the best coarse graining for a CA's behaviour at different points in space and time over the CA run.
- Some CAs show phase changes at the emergent level. We can detect the phase transition at the emergent level by monitoring the MI of the transient rule, and we developed normalised transition graphs to show this.
- The mutual information of some partial coarse grainings is high, despite the rules looking like poor matches visually. Extra entropy is an effective way to exclude such rules. Extra entropy is

a local, single value measure that, together with MI, effectively distinguishes between good and poor coarse grainings globally.

- MI resonance can be a problem with some initial conditions and coarse graining configurations. It is important to use a sufficiently varied test string to avoid the issue.
- Directed coarse graining adds exceptions to fine rules, allowing us to capture the underlying behaviour we want at an emergent level.
- The exceptions we added to rule 102 gave us a larger scale model of rule 110. It suggests that rule 110's complex behaviour may be the result of exceptions to a chaotic pattern.

13 CONTRIBUTIONS

This thesis investigates emergence. Emergence is not a very well defined (or understood) phenomenon; as we saw in §8 – and as with intelligence – authors disagree on even its most basic requirements and disqualifiers.

Our notions of emergence are intimately wrapped up in our experiences of the world around us. Most people would conclude that flocking is an emergent phenomenon (§8.2), but fewer would agree that temperature or a secret sharing scheme (§8.9) are really emergent. Not because they are innately bad examples, but because they fall less within our purview and because, like intelligence, we want to attach something 'natural' or 'human' to a phenomenon before admitting it is emergent. The same is true for emergence in CAs, which not only operate in an abstract mathematical space, but where we can also see exactly what is happening at any time.

Though we have purposefully avoided giving our own definition of emergence here, we implicitly adopted a fairly liberal and relativistic model. We have approximately used what Ryan calls weak emergence, which he defines as a model with a change of resolution, but not scope, between the high and low levels (§8.7, §8.8, [100]).

13.1 Emergence

We have seen the utility of relative emergence, basing it on the two language description in Ronald et al's definition (§9.3, [109]). We suggested that there is no correct model of emergence in a system, and have seen instances where we get several different – and useful – emergent models from a single underlying system.

And we have seen that emergence is not that special: it is actually very common (difficult to avoid, even), though most emergence captures aspects of the underlying behaviour that are not particularly enlightening or useful (§10.13, §10.19).

At least within the scope of weak emergence (as used here), we can create emergent models just by eliminating degrees of freedom from the underlying system, though this must be through uniform dimension loss (or projection). This lossy emergence is what makes the emergent model independent from low level behaviour (§11).

Drawing on Adami's information theoretic model of protolife, we introduced quantitative emergence through mutual information and Kolmogorov complexity (§9.17). Using this, we were able to quantify the information lost in an emergent projection (§11).

We also introduced neutral emergence, in analogy to neutral evolution (§9.9). We claimed that all emergence is neutral emergence[1] – because it must be lossy – but thinking about emergence as neutral emergence confers distinct advantages for developing robust solutions (§9.19).

13.2 Coarse graining and emergence in cellular automata

Despite their simplicity and mathematical transparency, elementary cellular automata can exhibit emergent behaviour (§10.5). We have shown that coarse graining is a good way to model these emergent properties. We were also able to use coarse graining to demonstrate emergence through elimination, creating emergent models as we mapped from four fine states (two fine cells at $g = 2$) to two coarse states (one coarse cell). Nothing was added to create the high level model, yet we often saw cohesive behaviour at the emergent level – behaviour that can be modelled with another elementary CA rule.

A total coarse graining must model aspects of the low level system's behaviour without error. A partial coarse graining does not have this restriction, and its errors add new behaviour to the high level (introduced in §10.16). This freedom to make mistakes can allow a partial coarse graining to capture more of low level system than a total coarse graining, which are all too often vacuously correct (§11.10). This shows us that correctness is not always a synonym for goodness, though they are usually related.

We found that using remarkably short and simple input test strings with partial coarse graining gave results that were almost as good as those we got from using test strings that covered all CA input states with total coarse graining (§10.20).

We introduced graining graphs as way of visualising all coarse grainings at a certain grain (or a specified subset of that grain). Graining graphs allowed us to explore the coarse graining space and showed us a progressive change in ECA behaviour over the rule space (§10.13). The graining graph of partial coarse grainings added significant numbers of useful new rules and showed us how partial coarse graining is able to exploit similarities in the rules that are not accessible when totally coarse graining (§10.19).

We have shown that mappings matter when modelling emergent systems. In §11.12, we saw that a poor mapping can reduce an otherwise good coarse graining with a high MI to one that captures little of the underlying behaviour. We also discussed different criteria for selecting valid coarse grainings through mappings: insisting on a rule being present for all possible mappings (the intersection of all mappings) gave consistently high quality results (§10.20); using the union of all mappings gave far more coarse grainings, many of which were of poor quality, though the extra choice was sometimes useful (§10.21).

1 Again, we limit ourselves to speculating about weak emergence here.

We briefly showed that we could find emergent models with evolutionary algorithms, using a genetic algorithm to find coarse grainings through their mutual information (§10.17). Due to the small state space we predominantly used, the 256 ECAs at $g = 2$, we did not investigate this further.

13.3 Emergence and information

We explained how to calculate the mutual information between the two CA rules in a coarse graining and showed a strong correlation between a high MI and the quality of a coarse graining. We found that the partial coarse grainings of a rule may have a higher MI than the total coarse grainings of that rule, despite their erroneous behaviour (§11.9).

We introduced feature extraction in order to model the principal emergent properties of a CA at different stages in a CA run (§12.1). Building on feature extraction, we showed that ECAs can undergo phase transitions at the emergent level that are not evident at the low level (§12.6). We found that we needed to restart the post transition coarse rule in order to ensure a good correlation between the high and low levels, and that the best point at which to perform this restart was at the phase transition. We also introduced transition graphs, with their distinctive S-curves, as a way of modelling these phase changes in CAs (§12.10). After trying different block sizes to calculate the MI in a coarse graining, we found that expanding the block size (or calculating the MI over time rather than space) did not help us distinguish between good and poor rules, though it did reduce the resolution of our model.

Adding new behaviour through a partial coarse graining can let us capture more of the fine CA than is possible when totally coarse graining. However, we had difficulty using mutual information to discriminate between good coarse grainings and coarse grainings where only a relatively small amount of their very high entropy was correlated to the low level. We introduced extra entropy, the conditional coarse entropy, as a way to distinguish between these two situations (§12.21). Despite being local measures, using both extra entropy and mutual information allowed us to get very close to our visual assessment of the goodness of a coarse graining.

We introduced rule exceptions (§12.32), alterations to CA rules that override the rule's behaviour for certain input conditions. Exceptions can have a larger neighbourhood than the rule to which they are applied. Rule exceptions, in combination with mutual information and extra entropy, allow us to perform directed coarse graining – to find a model of the emergent behaviour we want automatically. We showed several examples where directed coarse graining gave us different models depending on the exceptions we introduced at the low level.

Finally, we used a rule exception to find a way of changing the scale of rule 110 and coarse grained this rule plus exception to rule 110 (§12.37). We speculated on the relationship between

rules 102 and 110 and how rule 110's complex behaviour may be layered on top of the chaotic base structure provided by rule 102.

14 LOOKING FORWARD

The aim of this thesis was to understand a bit more about emergence and how to develop emergent systems automatically. We have seen that emergence is not a magical property – it is often remarkably mundane – and we have described ways to find emergent systems automatically, evaluate the quality of these systems and direct development towards the solutions *we want*.

This is (we hope) useful in itself – developing emergent systems is hard – but researchers are interested in emergent systems because of their innate robustness and adaptability, properties they want to introduce into systems that solve real industry or scientific problems.

Almost all of the models used here were limited in scope. Exhaustive search was an efficient technique for most of our examples, and we had little need to consider the problem landscape or how best to search it. Clearly this would not be true for most real world problems, and expanding the work here to such domains is the next step.

In §10.17, we used a genetic algorithm to find emergent models automatically, and we suggest that we can take advantage of the large body of existing work about evolutionary algorithms to find solutions to more difficult problems. But we also know that many evolutionary algorithms hit a complexity ceiling: they have difficulty scaling to problems beyond a certain difficulty [3], and it is likely that developing interesting emergent models would fall on the wrong side of this barrier. However, if we consider how we can exploit the problem structure – its landscape, its dynamics – we believe we can find the answers we want.

14.1 Emergence and robustness

If we accept the ideas of emergence developed here, all emergence is neutral emergence. Or all neutral emergence is emergence. But we didn't introduce neutral emergence as a new form of emergence; we introduced it as a new way of thinking about emergence.

Robustness is intimately related to neutral evolution and emergence. The aspects of the search landscape exploited by neutral evolution – correlations on different scales, basins, and so on – are exactly those that can make a model robust [17, 125, 126, 127]. And robustness is intimately related to neutral emergence too – as with Langton's link between physical systems and computation [18], this analogy should allow ideas and techniques from one field to be applied to the other.

We have seen some hints of this even in the simple ECA models considered here. The rule-based approach to emergent modelling and development used in coarse graining means that solutions are much more generally applicable than a case-by-case model and normally work well in other, similar circumstances. Crucially, these circumstances *only have to be similar with respect to the rules*. Just like Boids' flocking rules (§8.2), we can ignore behaviour in which we are not interested –

what the rest of the system is doing does not concern us if it does not impinge on our emergent model.

14.2 Exploiting problem structure

The literature review highlights a number of cases (§4, §5, §6) where the dynamical structure of a system has limited, targeted, focused or even spurred development (in the most general sense of the word). While interesting in their own right, the important point here is that doing this – the equivalent of constraining a gas to one small corner of a room – requires no effort on the part of system: it just happens because of the way the system is set up.

A lot of the literature review focuses on understanding how these physical constraints can be channelled, so they can be made to work in tandem with other system processes (evolution, etc.). Neutral emergence suggests how to exploit landscape structure to ascertain which solutions are robust. It also suggests that manipulating the abstraction of a problem, rather than the problem itself, may actually help find better solutions (we saw this again in §8.12). This idea can be taken further. If the system model is gradually built up then it can be made to solve a succession of incrementally more difficult problems, but always from the vantage point of the previous answer. Such a developmental approach should allow this progression to occur incrementally, gradually and autonomously.

14.3 Developmental systems

Though still relatively rare in computer science, there are some encouraging examples of using developmental paradigms. Kauffman [17] presents a suggestive discourse on how neural network synapse weights should be built up over time, changing the search landscape from smooth to rugged; the neural network model NEAT [128] takes development more literally, successively adding nodes to the network as it is trained (*complexification*); AlChemy [129] builds up chemical networks from simple beginnings.

More than just adopting a developmental paradigm, it seems important to create a space in which the system can develop itself. Like AlChemy and Tierra [130], we believe it is crucial that the model is allowed to organise itself and find its own path to the solution. Such a framework is likely to be subtle, but, as with many non-standard computation problems, nature is likely to be replete with examples if one just knows where (and how) to look.

In particular, gaining some understanding of the interaction between evolution and development is almost certainly vital to the answers being sought here. As with Ashby's Brain (§6.2.3), there is a need to grasp which factors – out of the plethora of possibilities – constitute the essential variables.

The development of a human foetus relies on an intricate interplay of hormones, pressure, gravity, nutrients and many other factors.

- The vast majority of DNA appears to have no function [3, 15].
- Just 1.5% of DNA codes for proteins, but 3% is occupied by regulatory functions [131].
- Virtually the same 'master toolkit' of homeobox genes controls basic development in every living creature from yeast to yak [132].
- The vast majority of human genes are shared with the mouse and chimp [131] and we are not all that genotypically dissimilar from the fruit fly or cabbage (or *Arabidopsis Thaliana* at least [133]), yet morphologically we are substantially different.
- Some interesting pointers from the field of (non-standard) computation are the emergent runtime topology of genetic algorithms [134], runtime structure of genetic programming trees [135, 136] and the (order from chaos) dynamics of random Boolean networks [17, 19].

All of these examples are suggestive. Although there are undoubtedly many pieces missing from the puzzle, we believe the next step here is finding a model that provides a developmental pathway to emergent solutions.

A COARSE GRAININGS

Total coarse grainings at g = 2 [table: coarse rule / fine rule]

0	0	255	66	128	162	243	208	0	251
255	0	0	68	254	162	48	209	255	251
0	2	204	68	90	165	243	209	192	252
255	2	255	68	165	165	48	211	252	252
0	4	48	80	170	170	243	211	0	253
255	4	243	80	170	171	0	219	255	253
200	5	170	85	34	174	255	219	128	254
236	5	90	90	187	174	192	220	254	254
0	8	165	90	34	175	204	220		
255	8	200	95	187	175	252	220		
34	10	236	95	128	176	0	221		
187	10	102	102	254	176	204	221		
0	12	153	102	128	186	255	221		
204	12	150	105	254	186	128	222		
255	12	240	112	0	187	254	222		
240	15	48	116	170	187	0	223		
0	16	243	116	255	187	255	223		
255	16	128	128	192	188	136	230		
0	24	254	128	252	188	238	230		
255	24	128	130	0	189	0	231		
0	32	254	130	255	189	255	231		
255	32	128	132	128	190	204	236		
0	34	254	132	254	190	136	238		
170	34	136	136	0	191	238	238		
255	34	238	136	255	191	0	239		
0	36	34	138	192	192	255	239		
255	36	187	138	252	192	240	240		
34	38	34	139	192	194	240	241		
187	38	187	139	252	194	128	242		
170	42	136	140	60	195	254	242		
34	46	204	140	195	195	0	243		
187	46	238	140	192	196	240	243		
0	48	128	144	204	196	255	243		
240	48	254	144	252	196	48	244		
255	48	150	150	204	200	243	244		
204	51	136	152	204	204	48	245		
48	52	238	152	136	206	243	245		
243	52	102	153	204	206	128	246		
60	60	153	153	238	206	254	246		
195	60	34	155	0	207	0	247		
0	64	187	155	204	207	255	247		
255	64	128	160	255	207	128	250		
0	66	254	160	48	208	254	250		

Total coarse grainings at g = 3 [table: coarse rule / fine rule]

0	0	0	34	204	76	255	136	255	175	204	202
255	0	170	34	255	76	0	138	128	176	128	203
0	1	255	34	0	80	170	138	254	176	254	203
255	1	0	36	240	80	255	138	128	178	204	204
0	2	255	36	255	80	0	139	254	178	0	205
170	2	0	38	0	81	255	139	0	179	204	205
255	2	255	38	255	81	136	140	255	179	255	205
0	3	0	40	0	84	204	140	128	182	136	206
255	3	255	40	255	84	238	140	254	182	204	206
0	4	0	42	85	85	0	143	0	183	238	206
204	4	170	42	170	85	255	143	255	183	0	207
255	4	255	42	0	96	128	144	0	185	204	207
0	6	128	44	255	96	254	144	255	185	255	207
255	6	254	44	0	98	128	146	128	186	0	208
0	8	0	46	255	98	254	146	254	186	240	208
255	8	255	46	128	100	128	148	0	187	255	208
0	10	0	47	254	100	254	148	170	187	0	209
170	10	255	47	0	112	136	152	255	187	255	209
255	10	0	48	240	112	238	152	192	188	0	211
0	11	240	48	255	112	0	155	252	188	255	211
255	11	255	48	0	116	255	155	0	189	0	213
0	12	0	50	255	116	136	156	255	189	255	213
204	12	255	50	0	117	192	156	128	190	128	214
255	12	51	51	255	117	238	156	254	190	254	214
0	14	204	51	0	119	252	156	0	191	0	215
255	14	0	52	255	119	0	157	170	191	255	215
15	15	255	52	0	126	255	157	255	191	204	216
240	15	0	55	255	126	128	158	0	192	128	217
0	16	255	55	0	127	254	158	192	192	254	217
240	16	0	56	255	127	0	159	252	192	0	219
255	16	255	56	0	128	255	159	255	192	255	219
0	17	0	63	128	128	128	160	192	194	192	220
255	17	255	63	254	128	254	160	252	194	204	220
0	18	0	64	255	128	128	162	192	196	252	220
255	18	255	64	0	129	254	162	204	196	0	221
0	19	0	66	255	129	170	170	252	196	204	221
255	19	255	66	128	130	0	171	136	198	255	221
0	20	0	68	254	130	170	171	192	198	128	222
255	20	204	68	128	132	255	171	238	198	254	222
0	24	255	68	254	132	204	172	252	198	0	223
255	24	0	70	128	134	0	174	0	199	204	223
0	28	255	70	254	134	170	174	255	199	255	223
255	28	0	72	0	136	255	174	0	200	0	227
0	32	255	72	136	136	0	175	204	200	255	227
255	32	0	76	238	136	170	175	255	200	204	228

Total coarse grainings at g = 3 (2) [table: coarse rule / fine rule]

136	230
238	230
0	231
255	231
0	235
255	235
0	236
204	236
255	236
0	237
255	237
0	238
136	238
238	238
255	238
0	239
255	239
240	240
0	241
240	241
255	241
128	242
254	242
0	243
240	243
255	243
0	244
240	244
255	244
0	245
240	245
255	245
128	246
254	246
0	247
240	247
255	247
0	249
255	249
128	250
254	250
0	251
255	251
0	252
192	252
252	252
255	252
0	253
255	253
0	254
128	254
254	254
255	254

Total coarse grainings at g = 4 [table: coarse rule / fine rule]

0	0	204	19	170	46	0	71	255	96	0	132
255	0	255	19	255	46	204	71	0	98	128	132
0	1	0	20	0	47	255	71	255	98	254	132
204	1	255	20	255	47	0	72	0	100	255	132
255	1	0	21	0	48	204	72	255	100	128	134
0	2	255	21	240	48	255	72	102	102	254	134
170	2	128	23	255	48	0	74	153	102	0	136
255	2	254	23	0	49	255	74	150	105	136	136
0	3	0	24	255	49	0	76	0	108	238	136
255	3	255	24	0	50	204	76	255	108	255	136
0	4	0	26	255	50	255	76	0	111	0	138
204	4	255	26	204	51	128	77	255	111	170	138
255	4	0	27	0	52	254	77	0	112	255	138
204	5	255	27	240	52	0	78	240	112	0	139
0	6	0	28	255	52	255	78	255	112	170	139
255	6	255	28	0	53	0	79	0	114	255	139
0	7	0	29	255	53	192	79	255	114	0	140
255	7	204	29	0	55	252	79	0	115	136	140
0	8	255	29	204	55	255	79	255	115	204	140
255	8	0	31	255	55	0	80	0	116	238	140
0	9	255	31	0	56	240	80	240	116	255	140
255	9	0	32	255	56	255	80	255	116	0	141
0	10	255	32	0	58	0	81	0	117	255	141
170	10	0	33	255	58	255	81	255	117	0	143
255	10	255	33	0	59	0	82	0	119	255	143
0	11	0	34	255	59	255	82	255	119	0	144
255	11	170	34	60	60	0	83	0	123	128	144
0	12	255	34	195	60	255	83	255	123	254	144
204	12	0	35	0	63	0	84	0	125	255	144
255	12	255	35	255	63	255	84	255	125	128	146
0	13	0	36	0	64	170	85	0	126	254	146
192	13	255	36	255	64	0	87	255	126	128	148
252	13	0	38	0	65	255	87	0	127	254	148
255	13	170	38	255	65	0	88	204	127	150	150
0	14	255	38	0	66	255	88	255	127	136	152
255	14	0	39	255	66	90	90	0	128	238	152
240	15	255	39	0	68	165	90	128	128	102	153
0	16	0	40	204	68	0	92	254	128	153	153
240	16	255	40	255	68	255	92	255	128	0	155
255	16	0	42	0	69	0	93	0	129	170	155
0	17	170	42	136	69	136	93	255	129	255	155
255	17	255	42	238	69	238	93	0	130	136	156
0	18	0	44	255	69	255	93	128	130	192	156
255	18	255	44	0	70	204	95	254	130	238	156
0	19	0	46	255	70	0	96	255	130	252	156

Total coarse grainings at g = 4 (2) [table: coarse rule / fine rule]

0	157	0	181	238	198	255	217	0	238	128	254
255	157	255	181	252	198	0	219	136	238	254	254
128	158	128	182	0	199	255	219	238	238	255	254
254	158	254	182	255	199	0	220	255	238		
0	159	0	183	0	200	192	220	0	239		
255	159	255	183	204	200	204	220	255	239		
128	160	0	184	255	200	252	220	240	240		
254	160	255	184	0	201	255	220	0	241		
0	162	0	185	255	201	0	221	240	241		
128	162	255	185	0	202	204	221	255	241		
254	162	0	186	204	202	255	221	0	242		
255	162	128	186	255	202	0	222	128	242		
0	163	254	186	0	203	128	222	254	242		
255	163	255	186	255	203	254	222	255	242		
90	165	0	187	204	204	255	222	0	243		
165	165	170	187	0	205	0	223	240	243		
0	167	255	187	204	205	204	223	255	243		
255	167	192	188	255	205	255	223	0	244		
0	168	252	188	0	206	0	224	240	244		
255	168	0	189	136	206	255	224	255	244		
170	170	255	189	204	206	0	226	0	245		
0	171	0	190	238	206	255	226	240	245		
170	171	128	190	255	206	0	227	255	245		
255	171	254	190	0	207	255	227	0	246		
0	172	255	190	204	207	0	228	128	246		
204	172	0	191	255	207	204	228	254	246		
255	172	170	191	0	208	255	228	255	246		
0	173	255	191	240	208	0	229	0	247		
255	173	0	192	255	208	255	229	240	247		
0	174	192	192	0	209	136	230	255	247		
170	174	252	192	240	209	238	230	0	248		
255	174	255	192	255	209	0	231	255	248		
0	175	192	194	0	211	255	231	0	249		
170	175	252	194	240	211	128	232	255	249		
255	175	60	195	255	211	254	232	128	250		
0	176	195	195	0	213	0	234	254	250		
128	176	0	196	255	213	255	234	0	251		
254	176	192	196	128	214	0	235	255	251		
255	176	204	196	254	214	255	235	0	252		
0	177	252	196	0	215	0	236	192	252		
255	177	255	196	255	215	204	236	252	252		
128	178	0	197	0	216	255	236	255	252		
254	178	255	197	204	216	0	237	0	253		
0	179	136	198	255	216	204	237	255	253		
255	179	192	198	0	217	255	237	0	254		

Partial coarse grainings at g = 2 □■■□■□■□□■■□■□■□□■■□■□■□ *[table: coarse rule / fine rule]*

0	0	255	37	200	95	165	165	48	208	254	250
255	0	34	38	236	95	170	170	243	208	0	251
0	2	187	38	0	100	170	171	48	209	255	251
255	2	170	42	255	100	34	174	243	209	192	252
0	3	34	46	102	102	187	174	48	211	252	252
255	3	187	46	153	102	34	175	243	211	0	253
0	4	0	48	0	103	187	175	0	219	255	253
255	4	240	48	255	103	128	176	4	219	128	254
200	5	255	48	150	105	254	176	223	219	254	254
236	5	204	51	4	108	128	178	255	219		
192	7	48	52	223	108	254	178	192	220		
252	7	243	52	240	112	128	186	204	220		
0	8	60	60	48	116	254	186	252	220		
255	8	195	60	243	116	0	187	0	221		
34	10	0	61	0	119	170	187	204	221		
187	10	255	61	255	119	255	187	255	221		
0	12	0	63	128	128	192	188	128	222		
204	12	255	63	254	128	252	188	254	222		
255	12	0	64	128	130	0	189	0	223		
192	13	255	64	254	130	255	189	255	223		
252	13	0	66	128	132	128	190	136	230		
240	15	255	66	254	132	254	190	238	230		
0	16	0	67	136	136	0	191	0	231		
255	16	255	67	238	136	255	191	255	231		
0	17	0	68	34	138	192	192	204	236		
255	17	204	68	187	138	252	192	136	238		
136	21	255	68	34	139	192	194	238	238		
238	21	136	69	187	139	252	194	0	239		
0	24	238	69	136	140	60	195	255	239		
255	24	204	71	204	140	195	195	240	240		
0	25	128	77	238	140	192	196	240	241		
255	25	254	77	128	144	204	196	128	242		
204	29	192	79	254	144	252	196	254	242		
192	31	252	79	150	150	204	200	0	243		
252	31	48	80	136	152	4	201	240	243		
0	32	243	80	238	152	223	201	255	243		
255	32	170	85	102	153	0	203	48	244		
0	34	136	87	153	153	255	203	243	244		
170	34	238	87	34	155	204	204	48	245		
255	34	90	90	187	155	136	206	243	245		
0	36	165	90	128	160	204	206	128	246		
4	36	0	91	254	160	238	206	254	246		
223	36	255	91	128	162	0	207	0	247		
255	36	136	93	254	162	204	207	255	247		
0	37	238	93	90	165	255	207	128	250		

Partial coarse grainings at g = 2 □■□■□■□■□■□■□■□■ *[table: coarse rule / fine rule]*

0	0	0	34	255	72	187	138	128	190	192	220
255	0	170	34	128	77	34	139	254	190	204	220
0	2	255	34	254	77	187	139	0	191	252	220
34	2	0	36	192	79	136	140	34	191	0	221
187	2	4	36	252	79	204	140	187	191	204	221
255	2	223	36	48	80	238	140	255	191	255	221
0	3	255	36	243	80	128	144	192	192	128	222
255	3	0	37	170	85	254	144	252	192	254	222
0	4	255	37	136	87	150	150	192	194	0	223
255	4	34	38	238	87	136	152	252	194	255	223
200	5	187	38	90	90	238	152	60	195	136	230
236	5	170	42	165	90	102	153	195	195	238	230
192	7	0	44	0	91	153	153	192	196	0	231
252	7	12	44	255	91	34	155	204	196	48	231
0	8	207	44	136	93	187	155	252	196	243	231
255	8	255	44	238	93	128	160	204	200	255	231
34	10	34	46	200	95	254	160	4	201	128	232
187	10	187	46	236	95	128	162	76	201	254	232
0	12	0	48	0	100	170	162	205	201	204	236
204	12	240	48	68	100	254	162	223	201	0	237
255	12	255	48	221	100	90	165	0	203	255	237
192	13	204	51	255	100	165	165	12	203	136	238
252	13	48	52	102	102	170	170	207	203	238	238
240	15	243	52	153	102	170	171	255	203	0	239
0	16	60	60	0	103	34	174	204	204	255	239
48	16	195	60	255	103	187	174	136	206	240	240
243	16	0	61	150	105	34	175	204	206	240	241
255	16	255	61	4	108	187	175	238	206	128	242
0	17	0	63	76	108	128	176	0	207	240	242
255	17	255	63	205	108	240	176	204	207	254	242
136	21	0	64	223	108	254	176	255	207	0	243
238	21	255	64	240	112	128	178	48	208	240	243
128	23	0	66	48	116	254	178	243	208	255	243
254	23	34	66	243	116	128	186	48	209	48	244
0	24	187	66	0	119	170	186	243	209	243	244
48	24	255	66	255	119	254	186	48	211	48	245
243	24	0	67	128	128	0	187	243	211	243	245
255	24	255	67	254	128	170	187	0	217	128	246
0	25	0	68	128	130	255	187	68	217	254	246
255	25	204	68	254	130	192	188	221	217	0	247
204	29	255	68	128	132	252	188	255	217	48	247
192	31	136	69	254	132	0	189	0	219	243	247
252	31	238	69	136	136	34	189	4	219	255	247
0	32	204	71	238	136	187	189	223	219	128	250
255	32	0	72	34	138	255	189	255	219	254	250

Partial coarse grainings at g = 2 (2) □■□■□■□■□■□■□■□■□■□■ *[table: coarse rule / fine rule]*

0	251
255	251
192	252
252	252
0	253
255	253
128	254
254	254

Partial coarse grainings at g = 3 □■■□■□■□□■■□■□■□□■■□■□■□ *[table: coarse rule / fine rule]*

0	0	255	21	255	54	204	95	255	138	255	171
255	0	0	23	0	55	0	96	0	139	204	172
0	1	255	23	255	55	255	96	255	139	0	174
255	1	0	24	0	56	0	98	136	140	170	174
0	2	255	24	255	56	255	98	204	140	255	174
170	2	0	26	0	59	128	100	238	140	0	175
255	2	255	26	255	59	254	100	0	143	170	175
0	3	0	28	0	63	0	108	255	143	255	175
255	3	255	28	255	63	255	108	0	144	128	176
0	4	0	31	0	64	0	112	128	144	254	176
204	4	255	31	255	64	240	112	240	144	128	178
255	4	0	32	0	66	255	112	254	144	254	178
204	5	255	32	255	66	0	115	255	144	0	179
0	6	0	34	0	68	255	115	128	146	255	179
255	6	170	34	204	68	0	116	254	146	0	181
0	7	255	34	255	68	255	116	0	147	255	181
255	7	0	35	0	69	0	117	255	147	128	182
0	8	255	35	255	69	255	117	128	148	254	182
255	8	0	36	0	70	0	119	254	148	0	183
0	10	255	36	255	70	255	119	0	152	255	183
170	10	0	38	0	72	0	126	136	152	0	185
255	10	255	38	255	72	255	126	238	152	255	185
0	11	0	40	0	76	0	127	255	152	128	186
255	11	255	40	204	76	255	127	0	155	254	186
0	12	0	42	255	76	0	128	255	155	0	187
204	12	170	42	128	77	128	128	136	156	170	187
255	12	255	42	204	77	254	128	192	156	255	187
0	13	128	44	254	77	255	128	238	156	0	188
255	13	254	44	0	79	0	129	252	156	192	188
0	14	0	46	255	79	255	129	0	157	252	188
255	14	255	46	0	80	0	130	255	157	255	188
15	15	0	47	240	80	128	130	128	158	0	189
240	15	255	47	255	80	170	130	254	158	255	189
0	16	0	48	0	81	254	130	0	159	0	190
240	16	240	48	255	81	255	130	255	159	128	190
255	16	255	48	0	82	128	132	128	160	170	190
0	17	0	49	255	82	254	132	254	160	254	190
255	17	255	49	0	84	128	134	128	162	255	190
0	18	0	50	255	84	254	134	170	162	0	191
255	18	255	50	85	85	0	136	254	162	170	191
0	19	51	51	170	85	136	136	0	167	255	191
255	19	204	51	0	87	238	136	255	167	0	192
0	20	0	52	255	87	255	136	170	170	192	192
255	20	255	52	0	93	0	138	0	171	252	192
0	21	0	54	255	93	170	138	170	171	255	192

Partial coarse grainings at g = 3 (2) □■■□■□■□□■■□■□■□□■■□■□■□ *[table: coarse rule / fine rule]*

0	194	128	217	254	242
192	194	254	217	0	243
252	194	0	219	240	243
255	194	255	219	255	243
192	196	192	220	0	244
204	196	204	220	240	244
252	196	252	220	255	244
136	198	0	221	0	245
192	198	204	221	240	245
238	198	255	221	255	245
252	198	128	222	0	246
0	199	254	222	128	246
255	199	0	223	240	246
0	200	204	223	254	246
204	200	255	223	255	246
255	200	0	227	0	247
0	201	255	227	240	247
255	201	204	228	255	247
204	202	0	230	0	249
128	203	136	230	255	249
254	203	238	230	128	250
204	204	255	230	254	250
0	205	0	231	0	251
204	205	255	231	255	251
255	205	0	232	0	252
136	206	255	232	192	252
204	206	0	235	252	252
238	206	255	235	255	252
0	207	0	236	0	253
204	207	204	236	255	253
255	207	255	236	0	254
0	208	0	237	128	254
240	208	255	237	254	254
255	208	0	238	255	254
0	209	136	238		
255	209	238	238		
0	211	255	238		
255	211	0	239		
0	213	255	239		
255	213	240	240		
128	214	0	241		
254	214	240	241		
0	215	255	241		
255	215	128	242		
204	216	240	242		

Partial coarse grainings at g = 3 □■□■□■□■□■□■□■□■□■ *[table: coarse rule / fine rule]*

0	0	255	21	255	54	204	95	0	138	0	171
255	0	0	23	0	55	0	96	170	138	170	171
0	1	255	23	255	55	255	96	255	138	255	171
255	1	0	24	0	56	0	98	0	139	204	172
0	2	255	24	255	56	255	98	255	139	0	174
170	2	0	26	0	59	128	100	136	140	170	174
255	2	255	26	255	59	254	100	204	140	255	174
0	3	0	28	0	63	0	104	238	140	0	175
255	3	255	28	255	63	255	104	0	143	170	175
0	4	0	31	0	64	0	108	255	143	255	175
204	4	255	31	255	64	255	108	0	144	128	176
255	4	0	32	0	66	0	112	128	144	240	176
204	5	255	32	255	66	240	112	240	144	254	176
0	6	0	34	0	68	255	112	254	144	128	178
255	6	170	34	204	68	0	115	255	144	254	178
0	7	255	34	255	68	255	115	128	146	0	179
255	7	0	35	0	69	0	116	254	146	255	179
0	8	255	35	255	69	255	116	0	147	0	181
255	8	0	36	0	70	0	117	255	147	255	181
0	10	255	36	255	70	255	117	128	148	128	182
170	10	0	38	0	72	0	119	254	148	254	182
255	10	255	38	255	72	255	119	0	152	0	183
0	11	0	40	0	76	0	126	136	152	255	183
255	11	255	40	204	76	255	126	238	152	0	185
0	12	0	42	255	76	0	127	255	152	255	185
204	12	170	42	128	77	255	127	0	155	128	186
255	12	255	42	204	77	0	128	255	155	170	186
0	13	128	44	254	77	128	128	136	156	254	186
255	13	254	44	0	79	254	128	192	156	0	187
0	14	0	46	255	79	255	128	238	156	170	187
255	14	255	46	0	80	0	129	252	156	255	187
15	15	0	47	240	80	255	129	0	157	0	188
240	15	255	47	255	80	0	130	255	157	192	188
0	16	0	48	0	81	128	130	128	158	252	188
240	16	240	48	255	81	170	130	254	158	255	188
255	16	255	48	0	82	254	130	0	159	0	189
0	17	0	49	255	82	255	130	255	159	255	189
255	17	255	49	0	84	128	132	128	160	0	190
0	18	0	50	255	84	254	132	254	160	128	190
255	18	255	50	85	85	128	134	128	162	170	190
0	19	51	51	170	85	254	134	170	162	254	190
255	19	204	51	0	87	0	136	254	162	255	190
0	20	0	52	255	87	136	136	0	167	0	191
255	20	255	52	0	93	238	136	255	167	170	191
0	21	0	54	255	93	255	136	170	170	255	191

Partial coarse grainings at g = 3 (2) □■□■□■□■□■□■□■□■□■□■ *[table: coarse rule / fine rule]*

0	192	254	214	240	240
192	192	0	215	0	241
252	192	255	215	240	241
255	192	204	216	255	241
0	194	128	217	128	242
192	194	254	217	240	242
252	194	0	219	254	242
255	194	255	219	0	243
192	196	192	220	240	243
204	196	204	220	255	243
252	196	252	220	0	244
136	198	0	221	240	244
192	198	204	221	255	244
238	198	255	221	0	245
252	198	128	222	240	245
0	199	254	222	255	245
255	199	0	223	0	246
0	200	204	223	128	246
204	200	255	223	240	246
255	200	0	227	254	246
0	201	255	227	255	246
255	201	204	228	0	247
204	202	0	230	240	247
128	203	136	230	255	247
254	203	238	230	0	249
204	204	255	230	255	249
0	205	0	231	128	250
204	205	255	231	254	250
255	205	0	232	0	251
136	206	255	232	255	251
204	206	0	233	0	252
238	206	255	233	192	252
0	207	0	235	252	252
204	207	255	235	255	252
255	207	0	236	0	253
0	208	204	236	255	253
240	208	255	236	0	254
255	208	0	237	128	254
0	209	255	237	254	254
255	209	0	238	255	254
0	211	136	238		
255	211	238	238		
0	213	255	238		
255	213	0	239		
128	214	255	239		

Partial coarse grainings at g = 4 □■■□■□■□□■■□■□■□□■■□■□■□ *[table: coarse rule / fine rule]*

0	0	204	19	170	42	255	65	255	84	0	116
255	0	255	19	255	42	0	66	170	85	240	116
0	1	0	20	0	44	255	66	0	87	255	116
204	1	255	20	255	44	0	67	255	87	0	117
255	1	0	21	0	46	255	67	0	88	255	117
0	2	255	21	170	46	0	68	255	88	0	118
170	2	128	23	255	46	204	68	90	90	255	118
255	2	204	23	0	47	255	68	165	90	0	119
0	3	254	23	255	47	0	69	0	91	255	119
255	3	0	24	0	48	136	69	255	91	0	123
0	4	255	24	240	48	238	69	0	92	255	123
204	4	0	25	255	48	255	69	255	92	0	124
255	4	255	25	0	49	0	70	0	93	255	124
204	5	0	26	255	49	255	70	136	93	0	125
0	6	255	26	0	50	0	71	238	93	255	125
255	6	0	27	204	50	204	71	255	93	0	126
0	7	255	27	255	50	255	71	204	95	255	126
255	7	0	28	204	51	0	72	0	96	0	127
0	8	255	28	0	52	204	72	255	96	204	127
255	8	0	29	240	52	255	72	0	98	255	127
0	9	204	29	255	52	204	73	255	98	0	128
255	9	255	29	0	53	0	74	0	99	128	128
0	10	0	31	255	53	255	74	255	99	254	128
170	10	255	31	0	55	0	76	0	100	255	128
255	10	0	32	204	55	204	76	255	100	0	129
0	11	255	32	255	55	255	76	102	102	255	129
255	11	0	33	0	56	128	77	153	102	0	130
0	12	255	33	255	56	204	77	0	103	128	130
204	12	0	34	0	57	254	77	255	103	254	130
255	12	170	34	255	57	0	78	150	105	255	130
0	13	255	34	0	58	255	78	0	108	0	131
192	13	0	35	255	58	0	79	204	108	255	131
252	13	255	35	0	59	192	79	255	108	0	132
255	13	0	36	255	59	252	79	204	109	128	132
0	14	255	36	60	60	255	79	0	110	254	132
255	14	0	37	195	60	0	80	255	110	255	132
240	15	255	37	0	61	240	80	0	111	128	134
0	16	0	38	255	61	255	80	255	111	254	134
240	16	170	38	0	62	0	81	0	112	0	136
255	16	255	38	255	62	255	81	240	112	136	136
0	17	0	39	0	63	0	82	255	112	238	136
255	17	255	39	255	63	255	82	0	114	255	136
0	18	0	40	0	64	0	83	255	114	0	137
255	18	255	40	255	64	255	83	0	115	255	137
0	19	0	42	0	65	0	84	255	115	0	138

Partial coarse grainings at g = 4 (2) □■■□■□■□□■■□■□■□□■■□■□■□ *[table: coarse rule / fine rule]*

170	138	128	160	255	181	252	196	254	214	0	234
255	138	254	160	128	182	255	196	0	215	255	234
0	139	255	160	254	182	0	197	255	215	0	235
170	139	0	162	0	183	255	197	0	216	255	235
255	139	128	162	255	183	136	198	204	216	0	236
0	140	170	162	0	184	192	198	255	216	204	236
136	140	254	162	255	184	238	198	0	217	255	236
204	140	255	162	0	185	252	198	255	217	0	237
238	140	0	163	255	185	0	199	0	219	204	237
255	140	255	163	0	186	255	199	255	219	255	237
0	141	90	165	128	186	0	200	0	220	0	238
255	141	165	165	170	186	204	200	192	220	136	238
0	143	0	167	254	186	255	200	204	220	238	238
255	143	255	167	255	186	0	201	252	220	255	238
0	144	0	168	0	187	204	201	255	220	0	239
128	144	255	168	170	187	255	201	0	221	255	239
254	144	170	170	255	187	0	202	204	221	240	240
255	144	0	171	0	188	204	202	255	221	0	241
0	145	170	171	192	188	255	202	0	222	240	241
255	145	255	171	252	188	0	203	128	222	255	241
128	146	0	172	255	188	255	203	254	222	0	242
254	146	204	172	0	189	204	204	255	222	128	242
128	148	255	172	255	189	0	205	0	223	240	242
254	148	0	173	0	190	204	205	204	223	254	242
150	150	255	173	128	190	255	205	255	223	255	242
0	152	0	174	254	190	0	206	0	224	0	243
136	152	170	174	255	190	136	206	255	224	240	243
238	152	255	174	0	191	204	206	0	226	255	243
255	152	0	175	170	191	238	206	255	226	0	244
102	153	170	175	255	191	255	206	0	227	240	244
153	153	255	175	0	192	0	207	255	227	255	244
0	155	0	176	192	192	204	207	0	228	0	245
170	155	128	176	252	192	255	207	204	228	240	245
255	155	240	176	255	192	0	208	255	228	255	245
136	156	254	176	0	193	240	208	0	229	0	246
192	156	255	176	255	193	255	208	255	229	128	246
238	156	0	177	0	194	0	209	0	230	254	246
252	156	255	177	192	194	240	209	136	230	255	246
0	157	128	178	252	194	255	209	238	230	0	247
255	157	204	178	255	194	0	211	255	230	240	247
128	158	254	178	60	195	240	211	0	231	255	247
254	158	0	179	195	195	255	211	255	231	0	248
0	159	204	179	0	196	0	213	128	232	255	248
255	159	255	179	192	196	255	213	204	232	0	249
0	160	0	181	204	196	128	214	254	232	255	249

Partial coarse grainings at g = 4 (3) □■■□■□■□□■■□■□■□□■■□■□■□ *[table: coarse rule / fine rule]*

0	250
128	250
254	250
255	250
0	251
255	251
0	252
192	252
252	252
255	252
0	253
255	253
0	254
128	254
254	254
255	254

Partial coarse grainings at g = 4 □■□■□■□■□■□■□■□■□■□■ *[table: coarse rule / fine rule]*

0	0	204	19	170	42	255	65	255	84	0	116
255	0	255	19	255	42	0	66	170	85	240	116
0	1	0	20	0	44	255	66	0	87	255	116
204	1	255	20	255	44	0	67	255	87	0	117
255	1	0	21	0	46	255	67	0	88	255	117
0	2	255	21	170	46	0	68	255	88	0	118
170	2	128	23	255	46	204	68	90	90	255	118
255	2	204	23	0	47	255	68	165	90	0	119
0	3	254	23	255	47	0	69	0	91	255	119
255	3	0	24	0	48	136	69	255	91	0	123
0	4	255	24	240	48	238	69	0	92	255	123
204	4	0	25	255	48	255	69	255	92	0	124
255	4	255	25	0	49	0	70	0	93	255	124
204	5	0	26	255	49	255	70	136	93	0	125
0	6	255	26	0	50	0	71	238	93	255	125
255	6	0	27	204	50	204	71	255	93	0	126
0	7	255	27	255	50	255	71	204	95	255	126
255	7	0	28	204	51	0	72	0	96	0	127
0	8	255	28	0	52	204	72	255	96	204	127
255	8	0	29	240	52	255	72	0	98	255	127
0	9	204	29	255	52	204	73	255	98	0	128
255	9	255	29	0	53	0	74	0	99	128	128
0	10	0	31	255	53	255	74	255	99	254	128
170	10	255	31	0	55	0	76	0	100	255	128
255	10	0	32	204	55	204	76	255	100	0	129
0	11	255	32	255	55	255	76	102	102	255	129
255	11	0	33	0	56	128	77	153	102	0	130
0	12	255	33	255	56	204	77	0	103	128	130
204	12	0	34	0	57	254	77	255	103	254	130
255	12	170	34	255	57	0	78	150	105	255	130
0	13	255	34	0	58	255	78	0	108	0	131
192	13	0	35	255	58	0	79	204	108	255	131
252	13	255	35	0	59	192	79	255	108	0	132
255	13	0	36	255	59	252	79	204	109	128	132
0	14	255	36	60	60	255	79	0	110	254	132
255	14	0	37	195	60	0	80	255	110	255	132
240	15	255	37	0	61	240	80	0	111	128	134
0	16	0	38	255	61	255	80	255	111	254	134
240	16	170	38	0	62	0	81	0	112	0	136
255	16	255	38	255	62	255	81	240	112	136	136
0	17	0	39	0	63	0	82	255	112	238	136
255	17	255	39	255	63	255	82	0	114	255	136
0	18	0	40	0	64	0	83	255	114	0	137
255	18	255	40	255	64	255	83	0	115	255	137
0	19	0	42	0	65	0	84	255	115	0	138

Partial coarse grainings at g = 4 (2) □■□■□■□■□■□■□■□■□■ *[table: coarse rule / fine rule]*

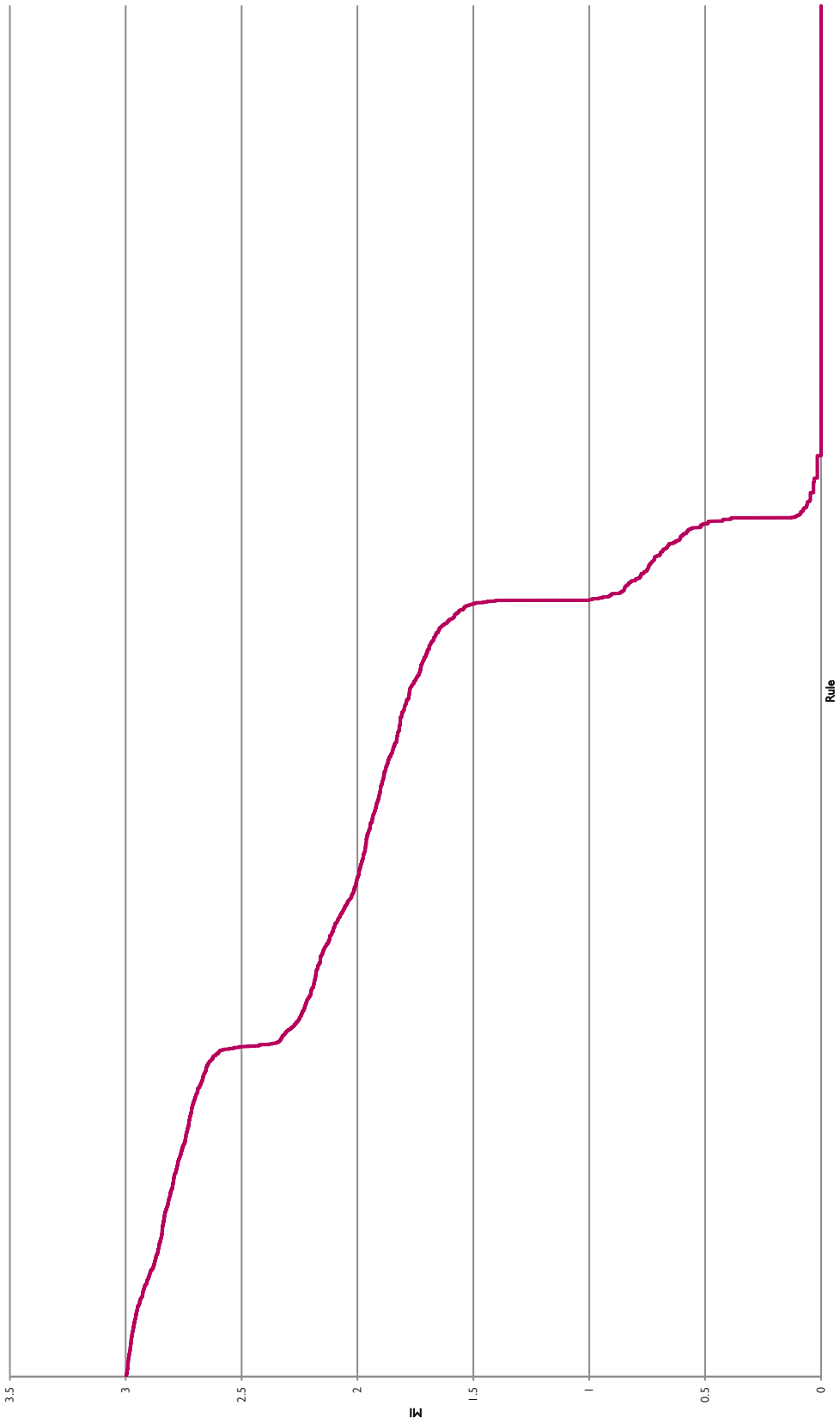
170	138	128	160	255	181	252	196	254	214	0	234
255	138	254	160	128	182	255	196	0	215	255	234
0	139	255	160	254	182	0	197	255	215	0	235
170	139	0	162	0	183	255	197	0	216	255	235
255	139	128	162	255	183	136	198	204	216	0	236
0	140	170	162	0	184	192	198	255	216	204	236
136	140	254	162	255	184	238	198	0	217	255	236
204	140	255	162	0	185	252	198	255	217	0	237
238	140	0	163	255	185	0	199	0	219	204	237
255	140	255	163	0	186	255	199	255	219	255	237
0	141	90	165	128	186	0	200	0	220	0	238
255	141	165	165	170	186	204	200	192	220	136	238
0	143	0	167	254	186	255	200	204	220	238	238
255	143	255	167	255	186	0	201	252	220	255	238
0	144	0	168	0	187	204	201	255	220	0	239
128	144	255	168	170	187	255	201	0	221	255	239
254	144	170	170	255	187	0	202	204	221	240	240
255	144	0	171	0	188	204	202	255	221	0	241
0	145	170	171	192	188	255	202	0	222	240	241
255	145	255	171	252	188	0	203	128	222	255	241
128	146	0	172	255	188	255	203	254	222	0	242
254	146	204	172	0	189	204	204	255	222	128	242
128	148	255	172	255	189	0	205	0	223	240	242
254	148	0	173	0	190	204	205	204	223	254	242
150	150	255	173	128	190	255	205	255	223	255	242
0	152	0	174	254	190	0	206	0	224	0	243
136	152	170	174	255	190	136	206	255	224	240	243
238	152	255	174	0	191	204	206	0	226	255	243
255	152	0	175	170	191	238	206	255	226	0	244
102	153	170	175	255	191	255	206	0	227	240	244
153	153	255	175	0	192	0	207	255	227	255	244
0	155	0	176	192	192	204	207	0	228	0	245
170	155	128	176	252	192	255	207	204	228	240	245
255	155	240	176	255	192	0	208	255	228	255	245
136	156	254	176	0	193	240	208	0	229	0	246
192	156	255	176	255	193	255	208	255	229	128	246
238	156	0	177	0	194	0	209	0	230	254	246
252	156	255	177	192	194	240	209	136	230	255	246
0	157	128	178	252	194	255	209	238	230	0	247
255	157	204	178	255	194	0	211	255	230	240	247
128	158	254	178	60	195	240	211	0	231	255	247
254	158	0	179	195	195	255	211	255	231	0	248
0	159	204	179	0	196	0	213	128	232	255	248
255	159	255	179	192	196	255	213	204	232	0	249
0	160	0	181	204	196	128	214	254	232	255	249

Partial coarse grainings at g = 4 (3) □■□■□■□■□■□■□■□■□■□■ *[table: coarse rule / fine rule]*

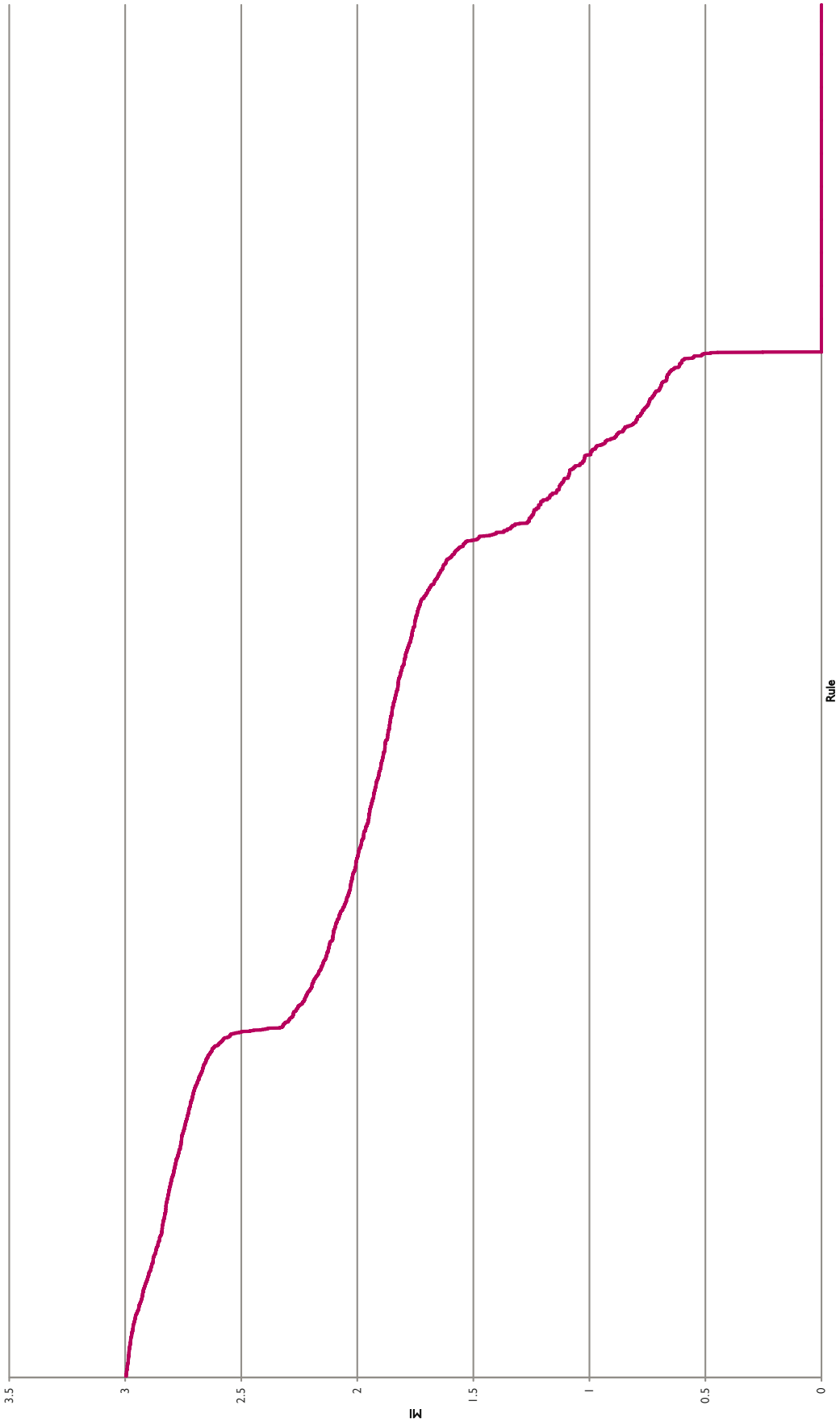
0	250
128	250
254	250
255	250
0	251
255	251
0	252
192	252
252	252
255	252
0	253
255	253
0	254
128	254
254	254
255	254

B MI OF RANDOM STRINGS

The MI of 32,000 total coarse grainings over all rules at g = 2, sorted by their MI at time 9. MIs were obtained from 100 random binary strings P(0.2) 1000 digits long. See §11.8 for context.



The MI of 32,000 total coarse grainings over all rules at g = 2, sorted by their MI at time 9. MIs were again obtained from 100 random binary strings P(0.8) 1000 digits long.



C COARSE GRAINING DISTRIBUTION

Distribution of total coarse grainings, sorted by MI. See §11.9 for context.

Distribution of partial coarse grainings, sorted by MI. (Graph shows only 255 of 280 series due to limitations of the graphing software.)

D PHASE CHANGES INITIAL CONDITION

This section discusses the initial condition developed to explore phase transitions in CAs. See §12.5-§12.22 for context.

Most of §12.5-§12.22 is devoted to studying the behaviour of three rules: the fine rule 130, and two coarse rules, 128 and 34, to which it partially coarse grains at $g = 2$. (We also look at rule 213 as an example of a poor coarse graining for rule 130.) For the purpose of this investigation, we created an initial CA condition designed to show the phase transition between coarse rules 128 and 34 to best effect. While we could use almost any state (by chunking the CA space to allow feature extraction), we want to increase the accuracy of our results by using initial conditions for which the whole CA would undergo a phase transition at approximately the same point. We also want our runs to show the CA in both its pre- and post-transition states for a significant period of time. Finally, while we want the first part of the run to favour rule 128, it is important that there should be something that rule 34 can match from the outset.

We know the phase transition takes place approximately when the triangles captured by rule 128 disappear. We also know that these triangles become two squares narrower at each timestep, so we must start with a fairly long string of contiguous ■s if we are to delay the phase transition appreciably. Rule 34 will match the edges of fine CA patterns, including single cells. We therefore append a few individual ■s, surrounded by □s, to the long string of ■s. Finally, we repeat the whole pattern several times to increase accuracy.

The initial condition used was a string of 53 ■s followed by a smaller number of □s interspersed with ■s (pattern □□■□□■□■□□□□□■□□□), and then a nearly identical string of 52 ■s joined with the same short string of □s and ■s (139 characters long in total). The whole string was repeated six times, giving an initial condition of 834 cells. The blocks of 52 and 53 ■s should show that the phase transition takes place at timestep 26, when the triangles disappear.

Figure D.1 Rule 130 showing a segment of the initial condition described in this section. The phase transition takes place at step 26, the time at which the triangles disappear.

D.1 Investigating string length

There is a substantial increase in variation within the MI results if we reduce the length of the initial condition. The next three graphs show MI results of 130 coarse grained to 128, starting from the initial condition we have just outlined. In all cases, the CA has been divided into four chunks horizontally and has a *y*-chunk of one.

We calculate the MI for some graphs differently in this section. Previously, the state space has been divided into blocks three coarse cells long (so six at the fine level for $g = 2$) and one cell (one timestep) high. Such a model is described as having an *x-block* of 3 and a *y-block* of 1, in analogy to *x*-chunk and *y*-chunk (§12.2). This notation is fully introduced in §12.17.

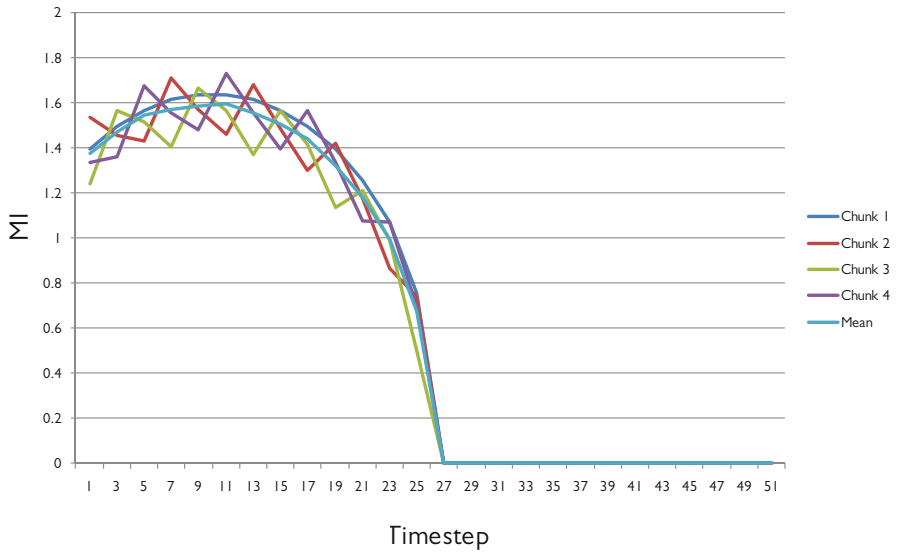


Figure D.2 MI between rules 130 and 128 for the mixed 52 and 53 ■s initial condition with *x*-block = 3, *y*-block = 1. There is relatively little variation between the four *x*-blocks, and the mean curve is smooth.

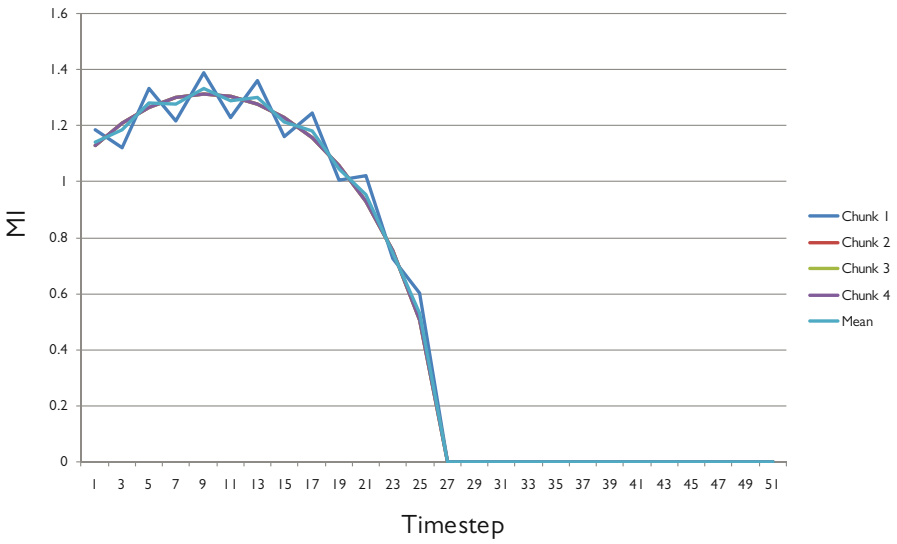


Figure D.3 MI between rules 130 and 128 for the mixed 52 and 53 ■s initial condition with *x*-block = 2, *y*-block = 1. As with the *x*-block = 3 graph, there is little variance between the chunks and the mean graph is smooth.

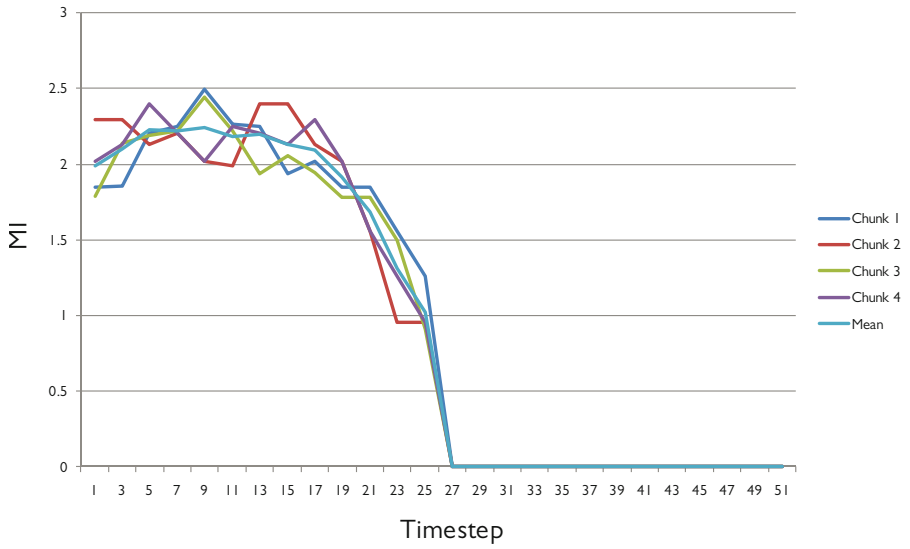


Figure D.4 MI between rules 130 and 128 for the mixed 52 and 53 ■s initial condition with *x*-block = 6, *y*-block = 1. There is a little more variation in the lines and the mean curve is less smooth, but the series is still close to graphs above.

The next three graphs show the MI between rules 130 and 128 for a shorter initial condition 40% as long as before (340 cells) and only containing strings of 52 ■s interspersed with the pattern □□■□□■□■□□□□□■□□□ (i.e. we no longer alternate between blocks of 52 and 53 ■s). We explore this next.

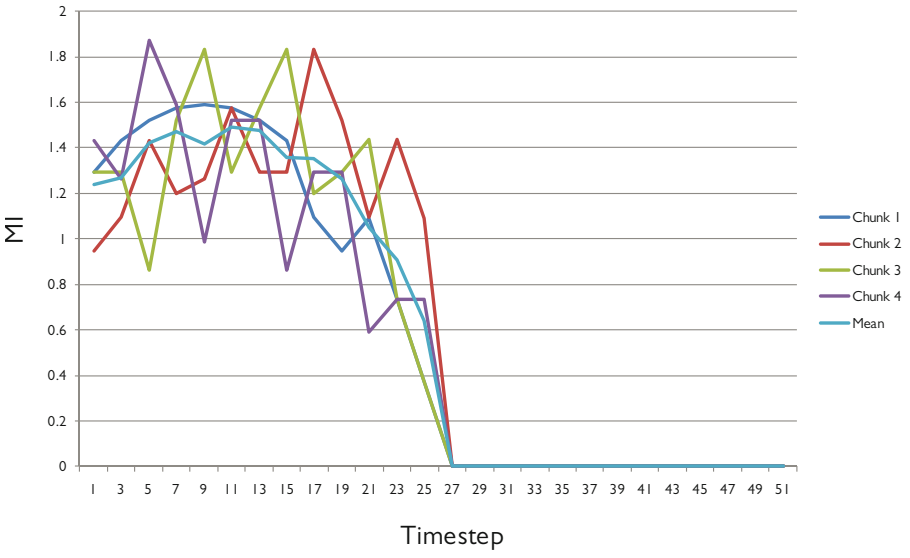


Figure D.5 MI between rules 130 and 128 with *x*-block = 3, *y*-block = 1 for the shorter 340 cell initial condition. Though the mean curve is still smooth, there is significantly more variation in each chunk's MI.

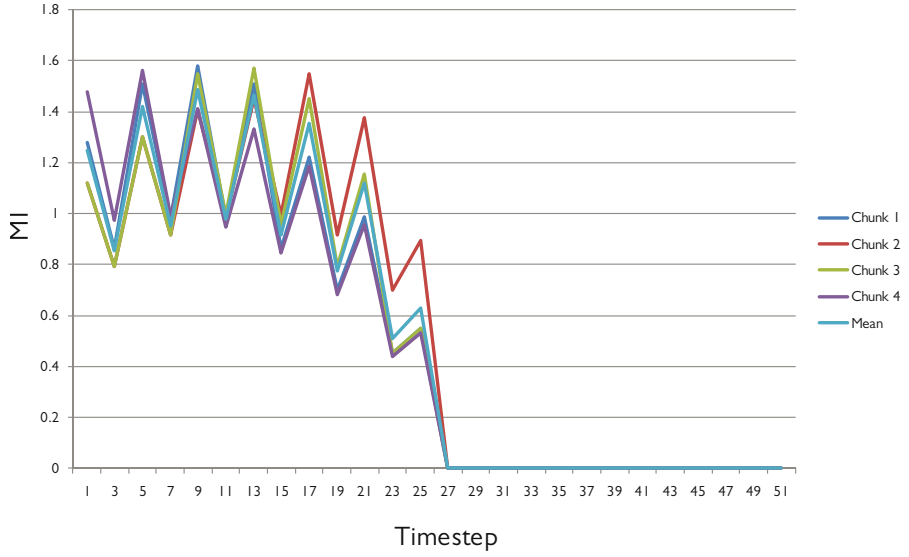


Figure D.6 MI between rules 130 and 128 with *x*-block = 2, *y*-block = 1 for the shorter 340 cell initial condition. We see significant resonance in the MIs as the simulation progresses, overpowering the mean graph's curve (see §D.2).

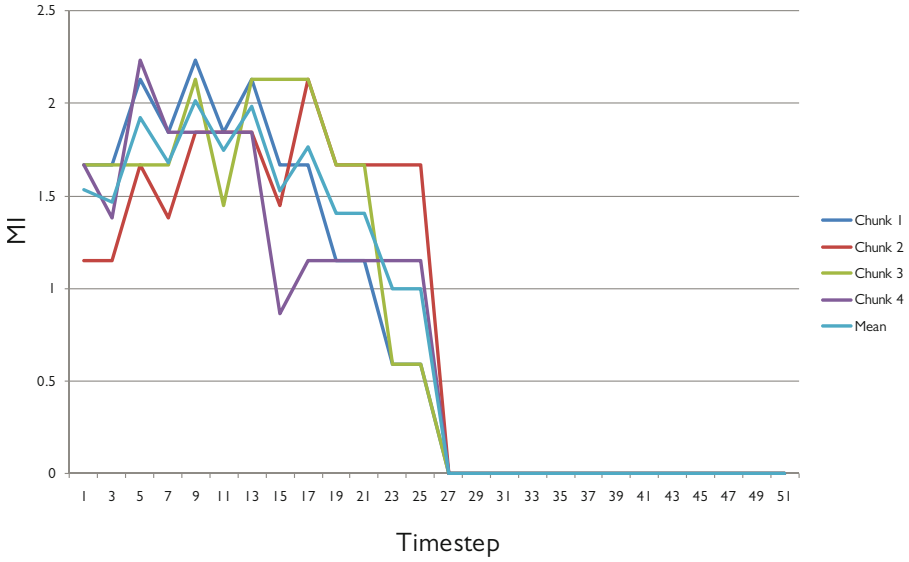


Figure D.7 MI between rules 130 and 128 with *x*-block = 6, *y*-block = 1 for the shorter 340 cell initial condition. The differences between each chunk are not in sync with each other as they were with *x*-block = 2, so the mean curve is better, though there is still some resonance. The individual chunks' curves are quite disparate, leading to an uneven mean curve.

D.2 Avoiding MI resonance

When the experiments were carried out with the shorter 340 cell initial condition, we found significant periodic variations in the MI at certain *x*-block sizes. Typically all of the chunks' MIs go up and down in consort; we call these unwanted fluctuations *MI resonance*.

Figure D.6 shows serious MI resonance and has *x*-block = 2, but Figure D.5, with *x*-block = 3, looks relatively free from resonance. Figure D.5 gives much better results because 52 does not divide into three (or 104 into six at the fine level), whereas it does (of course) divide into two. This means that the edges of all six triangles fall at the same position within a block throughout the run. Further, it must be the case (at the coarse level) that both edges of each triangle fall on the boundary of a chunk at every other timestep.

The triangles – specifically the triangle edges – contribute significantly towards the entropy of rule 130 (and totally towards the MI between it and 128) before the phase transition. But these edges will be invisible to the MI calculations if they fall on a block boundary. With triangles of base size 52 and *x*-block = 2, these edge states are omitted every other step, and a significant MI resonance is seen.

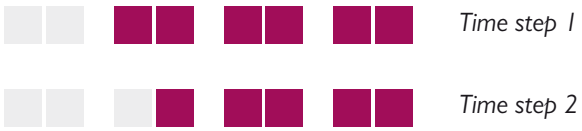


Figure D.8 These close-ups show the locations at which blocking can slice up a segment of rule 130's triangles. The initial condition tries to ensure that we always have all of these blocks when calculating the MI. Using the shorter initial condition (with triangles that are 52 cells wide only) means that we miss out on the combined red and grey state every other timestep with some *x*-blocks, resulting in significant MI resonance.

Using a triangle of base size 53 helps significantly as it is odd and thus prevents both edges from vanishing every other step with *x*-block = 2. It is also prime, which prevents the problem reoccurring at larger *x*-blocks. Despite its primality, these triangles still show some unwanted resonance for certain odd block sizes, presumably because the majority of edges vanish together at those sizes. We found that including triangles of size 52 and 53 reduced resonance to an acceptable level.[1]

1 It is important that the triangles are almost the same size so we see a sharp phase transition – using primes 47 or 59 would have left too large a gap.

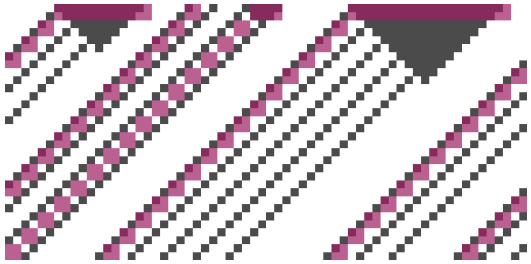
E EXTRA ENTROPY OF SELECTED RULES

E.1 Extra entropy of rule 130's coarse grainings

These tables show the maximum and mean extra entropy between rule 130 and its coarse grainings for different *x*-blocks over 51-step runs with the initial condition in §12.8. The *y*-block is one for all runs.

E.1.1 Rule 34

Diagonal lines that tally with those made by rule 130. A good match.

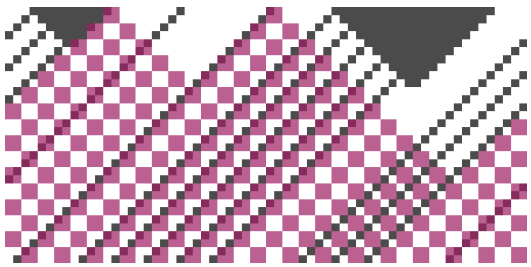


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	258.3%	120.8%	41.6%	13.8%	10.8%	12.1%	6.9%	5.6%	0.0%
Mean	161.2%	75.9%	25.4%	6.9%	4.0%	3.3%	1.9%	1.3%	0.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.229	0.343	0.255	0.136	0.125	0.141	0.128	0.106	0.103
Mean	0.174	0.250	0.168	0.0694	0.0467	0.0407	0.0297	0.0295	0.0289

E.1.3 Rule 50

Chequered pattern starting from single nodes and expanding outwards. Not a good match.

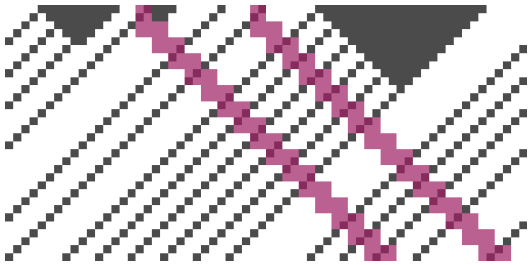


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	1275.6%	285.9%	181.6%	96.7%	79.8%	45.8%	40.9%	16.1%	5.8%
Mean	523.0%	147.6%	89.0%	42.9%	39.5%	26.2%	17.5%	8.5%	1.1%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.925	1.13	1.19	0.954	0.946	0.704	0.690	0.679	0.501
Mean	0.676	0.737	0.698	0.509	0.523	0.393	0.348	0.313	0.225

E.1.4 Rule 84

Thick diagonal lines in the opposite direction from rule 130's lines. Not a good match.

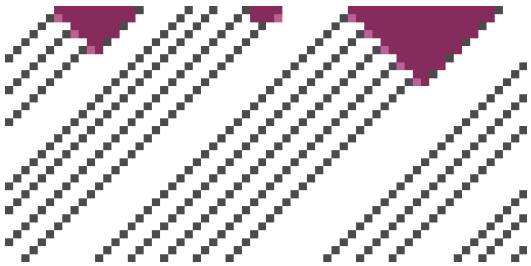


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	26603.9%	748.1%	308.2%	203.7%	152.3%	138.9%	88.0%	26.8%	3.6%
Mean	6314.4%	315.2%	119.5%	72.6%	49.3%	39.2%	22.9%	9.6%	0.9%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.513	0.816	0.894	0.951	0.847	0.919	0.673	0.770	0.556
Mean	0.422	0.588	0.538	0.452	0.391	0.366	0.270	0.277	0.164

E.1.5 Rule 128

Triangles that correspond with those seen in rule 130. A good match.

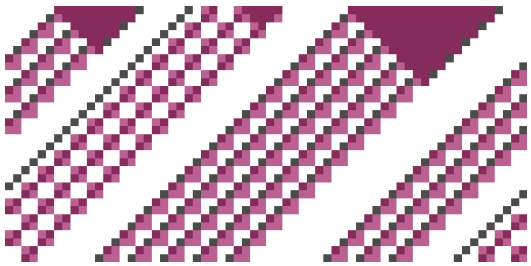


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Mean	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Mean	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

E.1.6 Rule 162

Triangles and lines that correspond to rule 130. A good match.

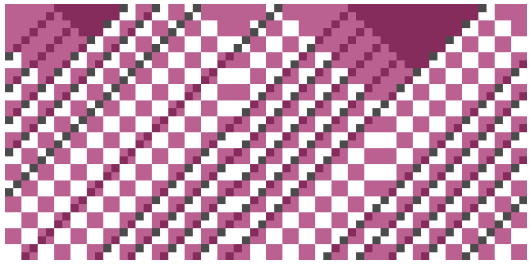


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	96.2%	31.8%	14.6%	10.1%	8.4%	7.9%	6.3%	5.7%	2.9%
Mean	66.0%	20.1%	9.1%	5.3%	3.3%	3.0%	2.2%	1.7%	0.2%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.483	0.350	0.233	0.193	0.183	0.181	0.161	0.159	0.148
Mean	0.347	0.239	0.153	0.107	0.0715	0.0674	0.0475	0.0552	0.0495

E.1.7 Rule 179

Triangles overlaid on areas of contiguous ■s and □s in the initial condition (subsuming rule 130's triangles). Chequered pattern elsewhere. Not a good match.

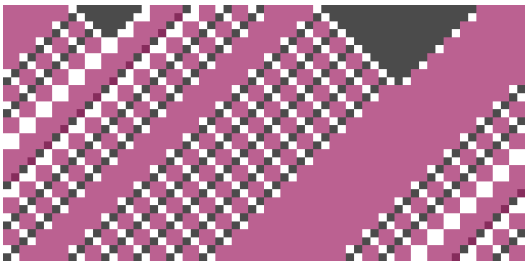


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	3205.9%	475.7%	214.8%	117.3%	83.2%	73.6%	50.7%	24.4%	6.3%
Mean	978.8%	237.3%	115.1%	64.5%	42.3%	37.7%	24.0%	11.3%	1.2%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.970	1.11	1.07	1.00	0.863	0.879	0.733	0.847	0.584
Mean	0.711	0.787	0.716	0.625	0.503	0.500	0.330	0.412	0.256

E.1.9 Rule 186

Inverse of rule 162. A good match.

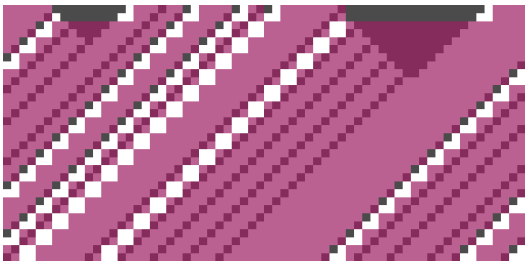


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	64.4%	24.5%	6.5%	3.4%	0.0%	0.0%	0.0%	0.0%	0.0%
Mean	45.4%	14.2%	3.0%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.255	0.251	0.0931	0.0577	0.000	0.000	0.000	0.000	0.000
Mean	0.185	0.159	0.0450	0.0178	0.000	0.000	0.000	0.000	0.000

E.1.11 Rule 187

Inverse of rule 34. A good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	290.8%	148.7%	46.5%	23.0%	14.0%	13.1%	7.7%	8.4%	2.3%
Mean	212.7%	86.6%	28.1%	12.1%	7.9%	6.4%	4.7%	2.9%	0.8%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.316	0.594	0.471	0.383	0.352	0.331	0.356	0.318	0.256
Mean	0.255	0.450	0.346	0.255	0.168	0.176	0.101	0.133	0.116

E.1.13 Rule 213

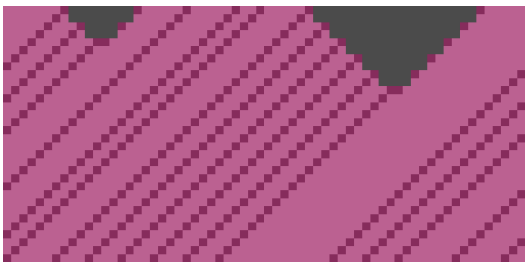
Inverse of rule 84. Not a good match.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	52030.4%	603.5%	509.0%	220.1%	140.4%	189.0%	84.6%	26.1%	6.0%
Mean	11451.7%	305.7%	136.2%	77.2%	45.2%	48.9%	25.9%	11.3%	1.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.663	0.976	1.36	0.940	1.13	1.08	0.773	0.885	0.556
Mean	0.555	0.719	0.782	0.456	0.497	0.460	0.322	0.334	0.202

E.1.15 Rule 254

Inverse of rule 128. A good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Mean	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**15**
Max	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Mean	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

E.1.16 Analysis of rule 130's coarse grainings

We see a clear distinction between well- and poorly-matched rules: the well-correlated rules have mean differences between 0 and 25% with poorly-matched grainings coming in at several times that figure. Rule 128 and its inverse 254 are total coarse grainings of rule 130 and therefore their MIs and coarse entropies are identical (yielding 0 extra entropy).

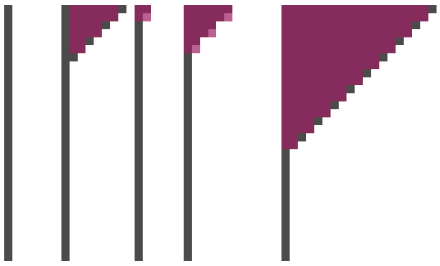
E.2 Extra entropy of rule 140's coarse grainings

Rule 140 draws right-angled triangles with vertical lines coming out of their base and from single blocks in the starting state.

As with the results above, these tables show the maximum and mean extra entropy for different *x*-blocks over 51-step runs. The *y*-block is one for all runs.

E.2.1 Rule 136

Corresponds to fine triangles in rule 140. A good match and total coarse graining.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Mean	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.000	0.000	0.000	0.000	0.000	0.000
Mean	0.000	0.000	0.000	0.000	0.000	0.000

E.2.3 Rule 140

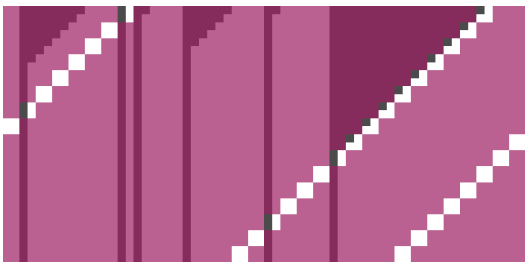
Covers fine rule 140 well, catching most vertical lines. A good match.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	153.6%	66.4%	33.9%	17.1%	6.9%	6.8%
Mean	83.7%	38.9%	19.8%	10.5%	4.1%	3.8%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.253	0.327	0.298	0.221	0.114	0.126
Mean	0.182	0.236	0.199	0.152	0.0706	0.0704

E.2.5 Rule 184

Dominant feature is diagonal lines in same direction as rule 140's triangle edges. Not a good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	23648.8%	788.1%	344.9%	203.8%	121.4%	78.3%
Mean	3900.8%	283.6%	110.6%	62.4%	38.2%	25.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.653	1.12	1.34	1.41	1.31	1.15
Mean	0.511	0.751	0.733	0.671	0.547	0.457

E.2.6 Rule 204

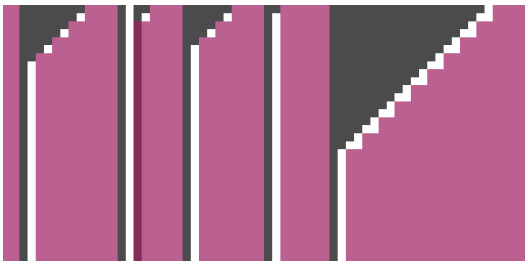
Straight lines approximately the inverse of rule 140's lines. A good match.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	1480.8%	87.1%	29.4%	6.6%	6.9%	3.3%
Mean	898.6%	66.7%	22.4%	5.5%	5.1%	2.3%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.476	0.465	0.309	0.0973	0.112	0.0629
Mean	0.383	0.369	0.234	0.0781	0.0800	0.0415

E.2.8 Rule 206

Inverse of rule 140. A good match.

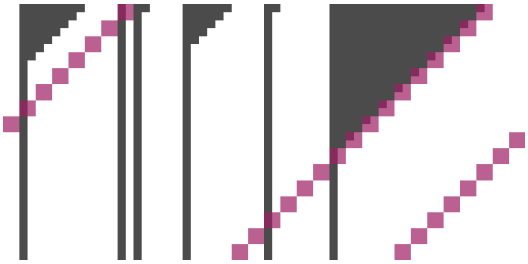


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	96.8%	55.9%	32.5%	15.6%	8.4%	7.5%
Mean	58.2%	31.6%	17.9%	6.7%	1.9%	2.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.366	0.425	0.393	0.308	0.197	0.169
Mean	0.229	0.270	0.227	0.119	0.0439	0.0444

E.2.9 Rule 226

Diagonal lines following edge of triangle and starting at other points. Not a good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	2888.9%	832.3%	523.4%	412.3%	216.4%	128.3%
Mean	1089.8%	294.1%	128.7%	73.4%	40.1%	25.6%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.366	0.425	0.393	0.308	0.197	0.169
Mean	0.229	0.270	0.227	0.119	0.0439	0.0444

E.2.10 Rule 238

Inverse of rule 136 and a total coarse graining. A good match.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	102.72%	32.04%	9.82%	8.54%	5.21%	3.72%
Mean	30.41%	12.13%	3.89%	4.66%	1.58%	1.08%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.178	0.149	0.0818	0.148	0.105	0.0639
Mean	0.0816	0.0650	0.0300	0.0471	0.0184	0.0113

E.2.11 Rule 252

Inverse triangles in wrong direction. Not a good match.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	1886.6%	340.1%	152.8%	70.3%	49.9%	31.7%
Mean	562.4%	154.1%	73.5%	32.9%	21.4%	14.1%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.880	0.951	0.959	0.788	0.680	0.490
Mean	0.309	0.360	0.333	0.245	0.198	0.144

E.2.12 Analysis of rule 140's coarse grainings

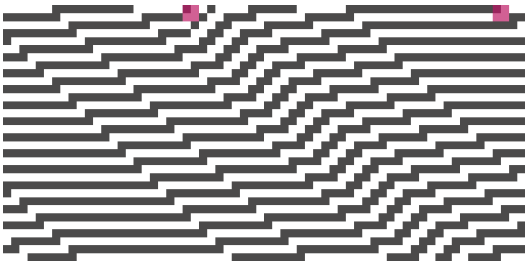
As with rule 130, the well-matched rules show low mean difference percentages at around 20% of the MI for *x*-block = 3, while the poorer rules have much larger differences, approaching an order of magnitude more. The split is quite stark.

E.3 Extra entropy for rule 43's coarse grainings

Cells in rule 43 alternate between on and off at each timestep. At places where the initial condition changes from ■ to □, the CA traces a diagonal pattern that reflects this switch.

E.3.1 Rule 136

Maps a ■□ fine pattern to ■. If several ■□ appear together, draws right angled triangles. A poor match.

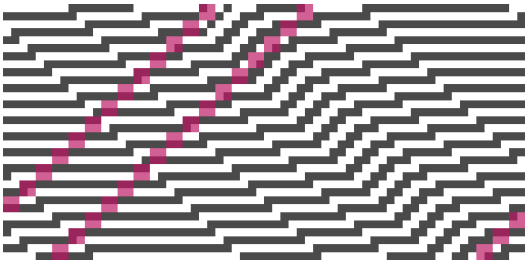


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	182.4%	64.0%	0.0%	0.0%	0.0%	0.0%
Mean	91.2%	32.0%	0.0%	0.0%	0.0%	0.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.0869	0.105	0.000	0.000	0.000	0.000
Mean	0.00334	0.00403	0.000	0.000	0.000	0.000

E.3.2 Rule 170

Draws diagonal lines starting from a ■□ fine pattern. Not a good match.

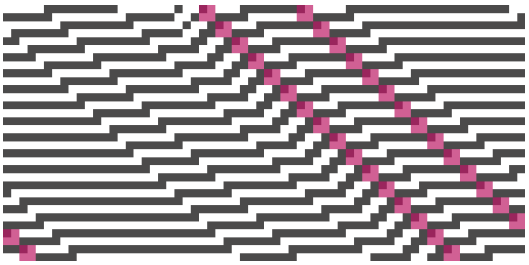


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	3436.8%	539.4%	321.1%	105.9%	74.4%	55.1%
Mean	1802.5%	289.2%	120.7%	54.8%	38.1%	25.3%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.592	0.998	1.26	1.12	1.12	0.949
Mean	0.524	0.793	0.792	0.679	0.641	0.490

E.3.3 Rule 184

Draws diagonal lines starting from a ■□ fine pattern. Not a particularly good match, though better than most.

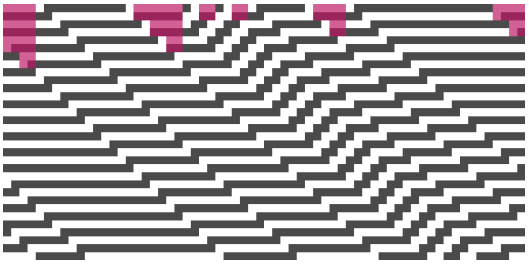


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	1130.8%	205.3%	118.7%	84.4%	48.9%	48.9%
Mean	523.6%	146.4%	78.2%	44.4%	29.7%	21.0%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.561	0.780	0.911	0.997	0.846	0.867
Mean	0.472	0.633	0.651	0.587	0.531	0.425

E.3.4 Rule 192

Draws right angled triangles. Not a good match.

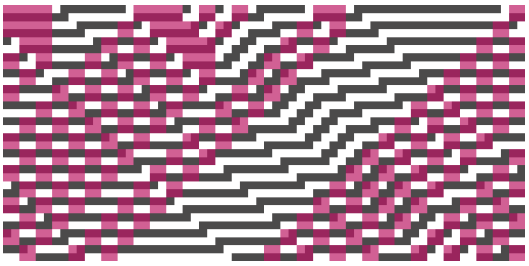


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	309.6%	172.4%	98.2%	82.0%	29.6%	57.4%
Mean	161.3%	70.3%	36.0%	31.5%	11.4%	11.8%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.208	0.197	0.0938	0.109	0.0703	0.0977
Mean	0.0516	0.0376	0.0282	0.0291	0.0148	0.0116

E.3.5 Rule 226

Draws diagonal lines starting from a ■□ fine pattern. Not a good match.

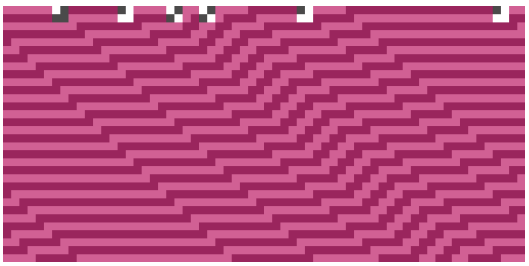


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	1705.3%	281.8%	167.9%	96.8%	62.9%	50.9%
Mean	692.6%	173.0%	101.6%	52.3%	37.8%	22.4%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.741	1.05	1.24	1.16	1.04	0.960
Mean	0.621	0.846	0.887	0.774	0.683	0.478

E.3.6 Rule 238

Similar behaviour to rule 136, though inverted. Not a good match.

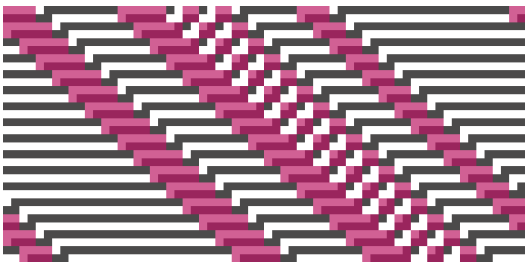


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	351.9%	200.4%	129.0%	10.6%	25.6%	56.0%
Mean	240.9%	58.3%	28.5%	2.4%	5.3%	9.3%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.443	0.313	0.105	0.0825	0.0969	0.0833
Mean	0.0414	0.0281	0.0126	0.00608	0.00643	0.00321

E.3.7 Rule 240

Draws diagonal lines that match fine rule's diagonal pattern. A good match.

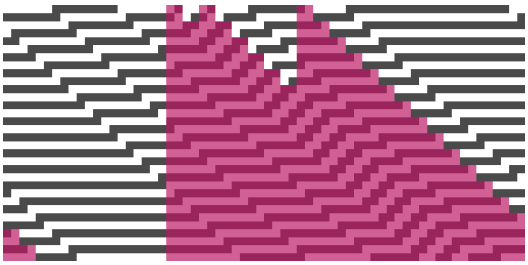


***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	772.9%	156.0%	85.4%	62.0%	52.5%	48.5%
Mean	450.1%	130.2%	74.4%	52.7%	43.9%	38.2%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	5.00	1.31	0.822	0.638	0.455	0.362
Mean	2.51	0.953	0.545	0.409	0.273	0.179

E.3.8 Rule 252

Draws right angled triangles following some of fine rule's structure. Not a good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	343.1%	172.4%	101.9%	60.6%	37.7%	41.6%
Mean	216.7%	87.4%	41.2%	28.6%	11.5%	12.3%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.612	0.486	0.444	0.410	0.316	0.281
Mean	0.322	0.281	0.229	0.208	0.111	0.102

E.3.9 Analysis of rule 43's coarse grainings

Rule 43 has fewer good coarse grainings than rules 130 and 140, but we see a similar correlation between coarse grainings that appear to match well and those with a low extra entropy. These results do show the importance of considering the absolute extra entropy in addition to percentage figures: rule 192, which is a poor match, has extra entropy percentages significantly lower than the well matched rules 240 and 184. And it does match a portion of the underlying CA quite well, but this portion is tiny. This is reflected in the absolute values, which are tiny.

E.4 Extra entropy of rule 192's coarse grainings

Rule 192 draws right angled, solid triangles.

E.4.1 Rule 12

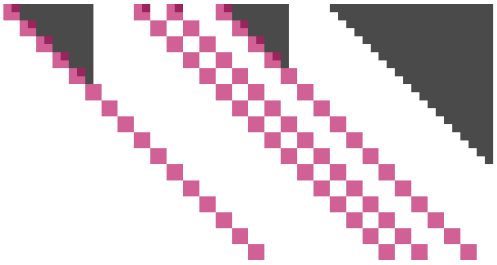
Vertical lines descend from edges of rule 192's triangles. Not a good match.

x-block	1	2	3	4	5	6
Max	6304.1%	3395.5%	1119.0%	611.5%	857.7%	332.0%
Mean	2669.7%	1262.3%	515.5%	309.8%	359.5%	165.4%

x-block	1	2	3	4	5	6
Max	34.4	35.3	15.6	10.8	19.4	7.50
Mean	14.0	12.5	6.64	5.00	7.61	3.41

E.4.2 Rule 48

Diagonal lines from edges of rule 192's triangles. Lines extend beyond edge of triangles. Not a good match.



x-block	1	2	3	4	5	6
Max	3020.1%	1454.3%	591.2%	556.6%	351.4%	303.9%
Mean	1500.7%	684.1%	309.3%	245.4%	182.9%	130.9%

x-block	1	2	3	4	5	6
Max	18.0	16.5	8.88	11.7	8.22	7.97
Mean	8.43	7.27	4.27	4.61	3.93	2.90

E.4.3 Rule 192

Rule 192 maps perfectly onto itself. A good match.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	845.8%	462.1%	177.1%	273.9%	209.4%	125.1%
Mean	225.0%	132.4%	39.8%	66.0%	49.4%	68.8%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	0.000	0.000	0.000	0.000	0.000	0.000
Mean	0.000	0.000	0.000	0.000	0.000	0.000

E.4.4 Rule 207

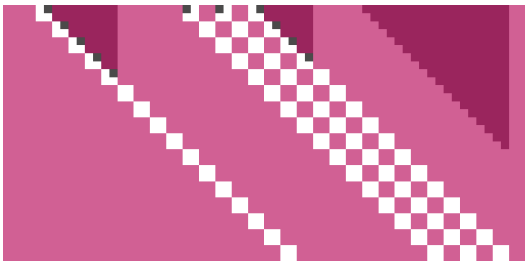
Inverse of rule 12. Not a good match.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	6917.2%	3096.5%	1103.5%	668.7%	504.1%	324.6%
Mean	1253.1%	501.3%	226.0%	136.3%	127.3%	83.6%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	51.7	43.3	22.0	16.4	13.3	9.28
Mean	16.0	11.8	7.43	5.37	5.48	3.73

E.4.5 Rule 243

Inverse of rule 48. Not a good match.



***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	3622.8%	2225.3%	899.0%	670.4%	508.5%	445.0%
Mean	1515.8%	716.3%	370.4%	256.5%	195.1%	137.9%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	26.6	30.5	17.3	16.5	13.8	13.1
Mean	10.6	9.15	6.65	5.46	4.90	3.56

E.4.6 Rule 252

Inverse of rule 192. A good match.

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	1419.4%	836.0%	531.1%	347.3%	293.1%	280.3%
Mean	418.3%	256.9%	187.9%	122.6%	129.9%	99.4%

***x*-block**	**1**	**2**	**3**	**4**	**5**	**6**
Max	2.93	3.02	1.93	1.18	1.42	1.39
Mean	1.31	1.32	1.08	0.600	0.771	0.711

E.4.7 Analysis of rule 192's coarse grainings

We see the same pattern emerging again with rule 192. The two good matches, rules 192 and 252, have significantly lower excess entropies than the other rules, whether we look at the relative or absolute figures.

The relatively high percentages for rule 192 (which has an absolute extra entropy of 0 in all cases) are due to rounding errors: one typical calculation divides 3.2×10^{-8} by 3.7×10^{-9}.

REFERENCES

1 D. Dumitrescu, *Evolutionary Computation*. CRC Press, 2000.

2 J. R. Koza, in W. Banzhaf, P. Nordin, R. E. Keller, F. D. Francone, *Genetic Programming: An Introduction*. Morgan Kaufmann Publishers, 1998.

3 W. Banzhaf, P. Nordin, R. E. Keller, F. D. Francone, *Genetic Programming: An Introduction*. Morgan Kaufmann Publishers, 1998.

4 R. Dawkins, *The Blind Watchmaker*. Longman Scientific & Technical, 1986.

5 J. H. Holland, *Adaptation in Natural and Artificial Systems.* MIT Press, 1992.

6 J. R. Koza, *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

7 I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.

8 A. Weeks, "Hannah ~ Chemistry Inspired Search," Department of Computer Science, MEng Final Project, University of York, 2005.

9 Inman Harvey, "Open the Box," Workshop on Evolutionary Computation with Variable Size Representation, ICGA97, Intl. Conf. on Genetic Algorithms, Michigan State University, 1997.

10 W. Comisky, J. Yu, J. R. Koza, "Automatic Synthesis of a Wire Antenna Using Genetic Programming," 2000 Genetic and Evolutionary Computation Conference, pp. 179-186, 2000.

11 J. R. Koza, F. H. Bennett, D. Andre, *Genetic Programming III: Automatic Programming and Automatic Circuit Synthesis.* Morgan Kaufmann, 1999.

12 P. Nordin, W. Banzhaf, "Complexity Compression and Evolution," Genetic Algorithms: Proceedings of the Sixth International Conference, pp. 310-317, 1995.

13 A. Teller, "Evolving Programmers: The Co-Evolution of Intelligent Recombination Operators," *Advances in Genetic Programming 2*, P. J. Angeline, K. E. Kinnear eds. MIT Press, pp. 45-68, 1996.

14 C. Adami, *Introduction to Artificial Life*. Springer, 1998.

15 M. Ridley, *Evolution*. Blackwell Publishing, 2004.

16 J. Watson, N. H. Hopkins, J. W. Roberts, J. Argetsinger-Streitz, A. M. Weiner, *Molecular Biology of the Gene*. Benjamin / Cummings, 1987.

17 S. A. Kauffman, *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

18 C. G. Langton, "Computation at the Edge of Chaos: Phase-Transitions and Emergent Computation," PhD thesis, University of Michigan, 1991.

19 S. A. Kauffman, *At Home in the Universe*. Oxford University Press, 1995.

20 J. Gleick, *Chaos*. Vintage, 1998.

21 Wikipedia (undated). *Kepler's laws of planetary motion*. http://en.wikipedia.org/wiki/Kepler%27s_laws_of_planetary_motion

22 S. H. Strogatz, *Nonlinear Dynamics and Chaos*. Perseus Books Publishing, 1994.

23 D. Poole, *Linear Algebra: a modern introduction*. Brooks Cole, 2006.

24 B. P. Tu, S. L. McKnight, "Metabolic cycles as an underlying basis of biological oscillations." *Nature Reviews Molecular Cell Biology* September 2006 7(9), pp. 696-701, 2006.

25 Wikipedia (undated). *Tacoma Narrows Bridge*. http://en.wikipedia.org/wiki/Tacoma_Narrows_Bridge

26 S. Doole, A. R. Champneys (undated). *Tacoma Narrows Bridge Disaster*. http://www.enm.bris.ac.uk/research/nonlinear/tacoma/tacoma.html

27 E. N. Lorenz, "Deterministic nonperiodic flow." *Journal of Atmospheric Science* 20 2, pp. 130-148, 1963.

28 W. Malkus, L. Howard. Malkus and Howard invented and built the waterwheel mechanical model of the Lorenz equations at MIT in the 1970s (cited in [22]).

29 Wikipedia (2005, Sep. 29). *Image:Roessler attractor.png*. http://commons.wikimedia.org/wiki/Image:Roessler_attractor.png

30 P. L. Read (undated). *Three Dimensional Systems: Lecture 6: The Lorenz Equations*. http://www.atm.ox.ac.uk/user/read/chaos/lect6.pdf

31 Wikipedia (2005, May 25). *Image:Lorenz system r28 s10 b2-6666.png*. http://commons.wikimedia.org/wiki/Image:Lorenz_system_r28_s10_b2-6666.png

32 J. Lighthill, "The recently recognized failure of predictability in Newtonian dynamics." *Procedures of the Royal Society of London* A 407, pp. 35-50, 1986.

33 Wikipedia (2005, Sep. 14). *Image:LogisticMap BifurcationDiagram.png*. http://commons.wikimedia.org/wiki/Image:LogisticMap_BifurcationDiagram.png

34 M. Feigenbaum, "Universal behavior in nonlinear systems." *Physica 7D: Nonlinear Phenomena*, pp. 16-39, 1980.

35 P. Cvitanovic, *Universality in chaos*. Taylor & Francis, 1989.

36 E. R. Berlekamp, J. H. Conway, R. K. Guy, *Winning Ways for your Mathematical Plays*. Academic Press, 1982.

37 S. Wolfram, "Universality and complexity in cellular automata." *Physica D* 10, pp. 1-35, 1984.

38 E. W. Weisstein (undated). *Percolation Threshold, MathWorld*. http://mathworld.wolfram.com/Percolation-Threshold.html

39 S. Wright, "Evolution in Mendelian populations." *Genetics* 16, pp. 97-159, 1931.

40 S. A. Kauffman, S. Levin, "Towards a general theory of adaptative walks on rugged landscapes." *J. Theoret. Biol* 128, pp. 11-45, 1987.

41 E. D. Weinberger, "Correlated and uncorrelated fitness landscapes and how to tell the difference." *Journal of Biological Cybernetics* 63 5, pp. 325-336, 1990.

42 W. Feller, *An Introduction to Probability Theory and Its Applications volume 2 (Probability & Mathematical Statistics)*. John Wiley & Sons, 1971.

43 W. C. Wimsatt, "Developmental constraints, generative entrenchment, and the innate-acquired distinction." *Integrating Scientific Disciplines*, Bechtel, ed., Dordrecht, 1986.

44 M. Eigen, P. Schuster, "The Hypercycle: A Principle of Natural Self-Organization." *Die Naturwissenschaften* 64, pp. 541-565, 1977.

45 M. Eigen, "Selforganization of Matter and the Evolution of Biological Macromolecules." *Die Naturwissenschaften* 58, pp. 465-523, 1971.

46 C. Gershenson, "Introduction to Random Boolean Networks," Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX), pp. 160-173, 2004.

47 S. Haykin, *Neural Networks: a comprehensive foundation*. Prentice-Hall, 1999.

48 A. Wuensche, "Genomic Regulation Modelled as a Network with Basins of Attraction," *Proc. Pac. Symp. Biocomput.*, pp. 89-102, 1998.

49 Wikipedia (undated). *Human Genome*. http://en.wikipedia.org/wiki/Human_genome

50 B. Derrida, D. Stauffer, "Phase transitions in two-dimensional Kauffman cellular automata." *Europhys. Lett* 2, pp. 739-745, 1986.

51 D. Stauffer, "Random Boolean networks: Analogy with percolation." *Phil. Mag.* B 56, pp. 901-916, 1987.

52 D. Stauffer, "On forcing functions in Kauffman's random Boolean networks." *J. Statis. Phys* 40, pp. 789-794, 1987.

53 B. Derrida, Y. Pomeau, "Random networks of automata: A simple annealed approximation." *Europys. Lett.* 1, pp. 45-49, 1986.

54 W. R. Ashby, *Design for a Brain.* Wiley, 1960.

55 M. L. Rosenzweig, "Evolution of the predator isocline." *Evolution* 27, pp. 84-94, 1973.

56 L. Van Valen, "A new evolutionary theory." *Evolutionary Theory* 1, pp. 1-30, 1973.

57 J. Maynard Smith, G. R. Price, "The logic of animal conflict." *Nature* 246, pp. 15-18, 1973.

58 S. Russell, P. Norvig, *Artificial Intelligence: a modern approach.* Prentice Hall, 2003.

59 M. L. Rosenzweig, J. S. Brown, T. L. Vincent, "Red queens and ESS: The coevolution of evolutionary rates." *Evolutionary Ecology* 1, pp. 59-84, 1987.

60 P. Bak, C. Tang, K. Wiesenfeld, "Self-organized criticality." *Phys. Rev.* A 38, pp. 364-374, 1988.

61 P. Bak, *How Nature Works: the science of self-organized criticality*. Oxford University Press, 1997.

62 D. M. Raup, "Biological extinction in earth history." *Science* 231, pp. 1528-1533, 1986.

63 T. S. Ray, *Evolution, Ecology and Optimization of Digital Organisms*. Santa Fe Institute working paper 92-08-042, 1992.

64 S. Wolfram, *A New Kind of Science*. Wolfram Media, 2002.

65 N. Owens, S. Stepney, "Investigations of the Game of Life cellular automata rules on Penrose Tilings: life-time, ash and oscillator statistics." *Journal of Cellular Automata*, 2010.

66 M. Gardner, "The fantastic combinations of John Conway's new solitaire game "life."" *Scientific American*, 1970.

67 J. von Neumann, A. W. Burks, *Theory of self-reproducing automata*. University of Illinois Press, 1966.

68 A. Flammenkamp, *Top 100 of Game-of-Life Ash Objects* (undated). http://wwwhomes.uni-bielefeld.de/achim/freq_top_life.html

69 P. Chapman (2002, Nov. 11). *Life Universal Computer*. http://www.igblan.free-online.co.uk/igblan/ca/

70 M. Cook, "Universality in Elementary Cellular Automata." *Complex Systems* 15, pp. 1-40, 2004.

71 M. Gell-Mann, *The Quark and the Jaguar: Adventures in the Simple and the Complex*. Abacus, 1995.

72 P. Cariani, "Emergence and Artificial Life." *Artificial Life II,* C. G. Langton, C. Taylor, J. D. Farmer, S. Rasmussen eds. Addison-Wesley, pp. 775-798, 1991.

73 C.E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, vol. 27, pp. 379–423, 623-656, July, October, 1948.

74 D. T. Haynie, *Biological Thermodynamics*. Cambridge University Press, 2001.

75 C. K. Mathews, K. E. van Holde, K. G. Ahern, *Biochemistry*. Addison-Wesley, 2000.

76 Wikipedia, *Mole (Unit)* (undated). http://en.wikipedia.org/wiki/Mole_%28unit%29

77 S. Levy, *Artificial Life*. Vintage Books, Random House, 1992.

78 C. Adami, *Introduction to Artificial Life*. Springer, 1998.

79 C. E. Shannon, "A Mathematical Theory of Communication." *Bell System Technical Journal*, pp. 379-423, 1948.

80 J. Goldstein, "Emergence as a construct: History and issues." *Emergence*, pp. 49-72, 1999.

81 C. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model." *Computer Graphics* 21, pp. 25-34, 1987.

82 C. Reynolds. *Boids: Background and Update* (undated). http://www.red3d.com/cwr/boids/

83 C. R. Ward, F. Gobet, G. Kendall, "Evolving collective behavior in an artificial ecology." *Artificial Life* 7, pp. 191-209, 2001.

84 E. F. Fama, "Efficient capital markets II." *J. Finance*, pp 1575-1617, 1991.

85 S. Stepney, F. Polack, H. Turner, "Engineering Emergence," *ICECCS'06*, pp. 89-97, 2006.

86 C. Reynolds, quoted in S. Levy, *Artificial Life*. Vintage Books, Random House, 1992.

87 Aristotle, *Metaphysics*. Volume book H (VIII), 350 BC. Translation from W. D. Ross, *Aristotle's metaphysics*, Oxford University Press, 1924.

88 G. H. Lewes, *Problems of Life and Mind Vol 2*. Kegan Paul, Trench, Turbner, & Co., 1875.

89 R. Abbott, "Emergence explained: getting epiphenomena to do real work." *Complexity* 12, pp. 13-26, 2006.

90 A. Einstein, "Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen." *Annalen der Physik* 322, pp. 549-560, 1905.

91 P. W. Anderson, "More is different." *Science* 177, pp. 393-396, 1972.

92 S. Weinberg, "Reductionism Redux." *The New York Review of Books* 5 October 1995, pp. 39-42, 1995.

93 R. J. Campbell, M. H. Bickhard, *Physicalism, emergence and downward causation*, (2001). http://www.lehigh.edu/˜mhb0/physicalemergence.pdf.

94 J. Kim, *Mind in a Physical World: An Essay on the Mind-Body Problem and Mental Causation*. MIT Press, 1998.

95 C. R. Shalizi, "Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata," PhD thesis, University of Wisconsin at Madison, 2001.

96 M. A. Bedau, "Downward Causation and the Autonomy of Weak Emergence." *Principia* 6, pp. 5-50, 2002.

97 A. Burns, I. J. Hayes, G. Baxter, C. J. Fidge, "Modelling temporal behaviour in complex socio-technical systems." *Technical Report YCS-2005-390*, Department of Computer Science, University of York, 2005.

98 M. H. Bickhard, D. T. Campbell, "Emergence," *Downward Causation*, P. B. Andersen, C. Emmeche, N. O. Finnemann, P. V. Christiansen eds. Aarhus University Press, 2000.

99 Y. Bar-Yam, *Dynamics of Complex Systems*. Westview Press, 1997.

100 A. J. Ryan, "Emergence is coupled to scope, not level." *Complexity* 13, pp. 67-77, 2007.

101 J. A. Silver, *Movie Day at the Supreme Court or "I Know It When I See It": A History of the Definition of Obscenity*, (2003 May 15). http://library.findlaw.com/2003/May/15/132747.html

102 C. D. Broad, *The Mind and its Place in Nature*. Kegan Paul, 1925.

103 D. V. Newman, "Chaos, Emergence, and the Mind-Body Problem." *Australasian Journal of Philosophy* 79, pp. 180-196, 2001.

104 J. P. Crutchfield, "The Calculi of Emergence: Computation, Dynamics, and Induction." *Physica D* 75, pp. 11-54, 1994.

105 N. Zaera, D. Cliff, J. Bruten, "(Not) Evolving Collective Behaviours in Synthetic Fish." *Animals to Animats 4: Proceedings of the Fourth Internatinal Conference on Simulation of Adaptive Behavior*, MIT Press, pp. 635-644, 1996.

106 G. M. Werner, M. G. Dyer, "Evolution of herding behavior in artificial animals." *From animals to animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, MIT Press, pp. 393-399, 1992.

107 B. L. Partridge, T. J. Pitcher, "Evidence against a hydrodynamic function of fish schools." *Nature* 279, pp. 418-419, 1979.

108 C. Reynolds, "An Evolved, Vision-Based Behavioral Model of Coordinated Group Motion." *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, MIT Press, pp. 384-392, 1992.

109 E. M. A. Ronald, M. Sipper, M. S. Capcarrère, "Design, observation, surprise! A test of emergence." *Artificial Life*, pp. 225-239, 1999.

110 S. C. Stearns, R. F. Hoekstra, *Evolution: an introduction*. Oxford University Press, 2000.

111 C. Adami, "What is complexity?" *BioEssays* 24, pp. 1085-1094, 2002.

112 C. Adami, N. J. Cerf, "Physical complexity of symbolic sequences." *Physica D* 137, pp. 62-69, 2000.

113 A. N. Kolmogorov, "Three approaches to the definition of the concept "quantity of information,"" *Probl. Peredachi Inf.*, pp. 3-11, 1965.

114 J. A. Clark, S. Stepney, H. Chivers, "Breaking the model: finalisation and a taxonomy of security attacks." *REFINE 2005*, pp. 225-242, 2005.

115 Bicycle Retailer and Industry News. *Suit Filed Against Dynacraft, Wal-Mart For Defective Bicycles*, http://www.bicycleretailer.com/bicycleretailer/headlines/article_display.jsp?vnu_content_id=1000806103

116 I. Sommerville, *Software Engineering*. Addison-Wesley, 2001.

117 J. Branke, "Creating robust solutions by means of evolutionary algorithms," *International Conference on Parallel Problem Solving from Nature*, A. E. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel eds. Springer-Verlag, pp. 119-128, 1998.

118 S. T. Abedon, *Microbial Growth*, (1998 Apr 3). http://www.mansfield.ohio-state.edu/~sabedon/biol2025.htm#stationary_phase

119 E. W. Dijkstra, "A note on two problems in connexion with graphs." *Numerische Mathematik 1*, pp. 269-271, 1959.

120 D. H. Wolpert, W. G. Macready, "No Free Lunch Theorems for Optimization." *IEEE Transactions on Evolutionary Computation* 5, pp. 295-296, 1997.

121 F. Polack, S. Stepney, "Emergent Properties Do Not Refine." *Electr. Notes Theor. Comput. Sci.*, pp. 163-181, 2005.

122 H. Turner, S. Stepney, "Rule Migration: Exploring a design framework for modelling emergence in CA-like systems." *Int. J. Unconventional Computing* 3, pp49-66, 2007.

123 N. Israeli, N. Goldenfeld, "Coarse-graining of cellular automata, emergence, and the predictability of complex systems." *Phys. Rev. E*, 2006.

124 yWorks Developers' Guide, *Organic Layout Style*, (undated). http://www.yworks.com/products/yfiles/doc/developers-guide/smart_organic_layouter.html

125 A. Piszcz, T. Soule, "Dynamics of evolutionary robustness." *GECCO 2006*, pp. 871-878, 2006.

126 M. Travisano, F. Vasi, R. E. Lenski, "Long-term experimental evolution in Escherichia coli. III. Variation among replicate populations in correlated responses to novel environments." *Evolution* 49, pp. 189-200, 1995.

127 F. Vasi, M. Travisano, R. E. Lenski, "Long-term experimental evolution in Escherichia coli. II. Changes in life-history traits during adaptation to a seasonal environment." *The American Naturalist* 144, pp. 432-456, 1994.

128 K. O. Stanley, "Efficient Evolution of Neural Networks Through Complexification," PhD thesis, University of Texas at Austin, 2004.

129 W. Fontana, L. W. Buss, "The barrier of objects: From dynamical systems to bounded organization Boundaries and Barriers," *Boundaries and Barriers*, J. Casti and A. Karlqvist eds. Addison-Wesley, pp. 56-116, 1996.

130 T. S. Ray, *Evolution, Ecology and Optimization of Digital Organisms*, Santa Fe Institute working paper 92-08-042, 1992.

131 S. B. Carroll, *Endless forms most beautiful: the new science of evo devo and the making of the animal kingdom*. W. W. Norton & Company, 2005.

132 W. J. Gehring, *Master Control Genes in Development and Evolution: The Homeobox Story*. Yale University Press, 1998.

133 New Science Foundation, *First-Ever Complete Plant Genome Sequence Is Announced* (2000 Dec. 13), http://www.nsf.gov/od/lpa/news/press/00/pr0094.htm

134 J. L. Payne, M. J. Eppstein, "Emergent Mating Topologies in Spatially Structured Genetic Algorithms." *GECCO 2006*, pp. 207-214, 2006.

135 J. M. Daida, "Characterizing the Dynamics of Symmetry Breaking in Genetic Programming." *GECCO 2006*, pp. 799-806, 2006.

136 J. M. Daida, H. Li, R. Tang, A. M. Hilss, "What makes a problem GP-hard? Validating a hypothesis of structural causes." *GECCO 2003*, pp. 1665-1677, 2003.

137 A. Weeks, S. Stepney, F. A. C. Polack, "Neutral Emergence: a proposal." *Symposium on Complex Systems Engineering*, 2007.

138 A. Weeks, S. Stepney, F. A. C. Polack, "Neutral Emergence and Coarse Graining." *ECAL 2007*, pp. 1131-1140, 2007.

139 A. Weeks, S. Stepney, F. A. C. Polack, "Investigating emergence by coarse graining Elementary Cellular Automata." *ALife XI*, pp. 686-693, 2008.

140 A. Weeks, "Towards Automated Proof Using Genetic Programming." Third Year Project, Department of Computer Science, University of York, 2004.

141 A. Weeks, "Chemistry Inspired Search." Fourth Year Project, Department of Computer Science, University of York, 2005.

142 G. Wald, "The Original Life," *Scientific American* August 1954, pp. 44-53, 1954.

143 R. Shapiro, *Origins: A Skeptic's Guide to the Creation of Life on Earth*. Heinemann, 1986.

144 A. W. Schwartz, G. J. F. Chittenden, "Synthesis of uracil and thymine under simulated prebiotic conditions," *Biosystems* September 1977, pp. 87-92, 1977.

145 V.G. Red'ko. *Principia Cybernetica, Hypercycles*, http://pespmc1.vub.ac.be/HYPERC.htm, 1998.

146 W. Fontana, L. W. Buss, "The barrier of objects: From dynamical systems to bounded organization," *Boundaries and Barriers*, J. Casti, A. Karlqvist eds. Addison-Wesley, pp. 56-116, 1996.

147 S. K. Scott, *Chemical Chaos*. Clarendon Press, 1993.

148 B. P. Belousov, "A periodic reaction and its mechanism," personal archives, 1951. Translated and published in *Oscillations and travelling waves in chemical systems*, R. J. Field, M. Burger eds. Wiley, 1985.

149 A. M. Zhabotinskii, "Periodic processes of the oxidation of malonic acid in solution (study of the kinetics of Belousov's reaction)." *Biofizika*, p. 306, 1964.

150 A. M. Zhabotinskii, A. B. Rovinsky, "Mechanism and nonlinear dynamics of an oscillating chemical reaction." *J. Stat. Phys.* 48, pp. 959-976, 1987.

151 J. L. Hudson, O. E. Rössler, "Chaos in simple three- and four-variable chemical systems," *Modelling of patterns in space and time*, W. Jäger, J. D. Murray eds. Springer, pp. 135-145, 1984.

152 H. C. Killory, O. E. Rössler, J. L. Hudson, "Higher chaos in a four-variable chemical reaction model." *Phys. Lett. A* 122, pp. 341-345, 1987.

153 O. E. Rössler, "An equation for continuous chaos." *Phys. Lett.* 57A, pp. 397-398, 1976.

154 O. E. Rössler, "Chaotic behavior in simple reaction systems." *Z. Naturforsch.* 31a, pp. 259-264, 1976.

155 J. P. Crutchfield, J. E. Hanson, "Attractor vicinity decay for a cellular automaton." *Chaos* 3, pp. 215-224, 1993.

156 A. Weeks, *Coarse Graining Graphs for Elementary Cellular Automata*. http://www.cs.york.ac.uk/nature/group/theses/AndrewWeeks/coarse-graining-graphs.zip, 2010.

157 J. C. Spall, *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Whiley and Sons, 2003.