Exploring the genotype-phenotype map for gene regulatory networks capable of pattern formation.

James Lloyd Cotterell

Thesis Presented for the Degree of Doctor of Philosophy
The University of Edinburgh
2008
Declaration

I declare,

(a) That this thesis has been composed by myself
(b) That this work is my own except where otherwise stated
(c) That this work has not been submitted for any other degree or professional qualification.

James Cotterell

April 2008
Acknowledgements

Thanks to those people who read the thesis and gave me feedback, when you spend so long looking at the same words it's sometimes impossible to get any perspective and to see the obvious mistakes so this was a great help.

Thanks to James for being a great supervisor and more importantly teacher over the years. Many supervisors are just happy to have the extra cash that a PhD student brings, whereas James actually feels like it's his duty to teach his students. I think the fact that everyone who has been in the lab wants to come back is testament to how good a boss he is. I'd also like to say thanks for the chance to come and finish my PhD in Barcelona by the beach, though in some ways this made it difficult to work!

Thanks to people in the lab for generally keeping me entertained - they all have their own little idiosyncrasies - Sahdia 'trigger' Raja, Spjim Swoger, Jeff - Coca Cola light's best customer, Don Bernado for the random german swear words, Spen/Hank for the music, Anto, Marit, Laura the mother hen, luca 'Yeah Jackie!' and Mike.

Coming from the biological sciences I didn't know much about computers and hence I have been very reliant on those people who have had this knowledge so thanks to the geeks - without you I wouldn't know what a borg was! I'd also like to thank Bill Hill for the numerous times he helped me with computational issues and solved them within 2 minutes.

I had started another project performing confocal imaging of cell migration in the limb bud but the results were not good enough and so the project was stopped. I'd like to thank Paul Perry and Timo Zimmerman anyhow for help they gave me in this project.

Thanks to my family for the patience and the support over the years. I've forgot even more birthdays than normal recently but they haven't held it against me. Thanks to mum for all of the home cooked meals when I came home. Thanks to dad; Rach, Jo, Naomi and myself will never forget you.

Finally thanks to my nieces, Maisy, Olivia and Chloe for the inspiring questions such as 'Nanny, do you play old music because this is an old car?' I sometimes feel that in some ways we get less intelligent as we get older as we start taking the relationships we see in the world around us for granted. It's a fight to stay open minded.
For 'Dad'

23 March 1948 – 22 February 2007
Abstract

Exploring the genotype-phenotype map for gene regulatory networks capable of pattern formation.

The phenomenon of pattern formation is one of the most intriguing in developmental biology. The fundamental question is how a group of identical cells develops into a spatially organized set of different cell types. These patterning mechanisms are controlled by regulatory networks of genes in different cells interacting together to act like a cellular computer. Traditional approaches worked on a case by case basis, analyzing gene networks responsible for specific pattern formation phenomena and exploring how they function by reverse genetics. More recently systems biology approaches have been used to explore how these gene networks function, extracting features that are only apparent on a systems scale. These approaches require the computational modeling of the underlying gene regulatory networks controlling the individual patterning phenomena in a spatial context. They have been successful in identifying widespread phenomena such as the robustness of network function to mutation and noise.

Due to the advent of more powerful computers, we now believe systems biology can go much further. Instead of limiting ourselves to the gene networks we know are responsible for pattern formation in specific systems, we can model all possible gene networks up to a particular level of complexity (number of genes and interactions). By modeling all possible regulatory networks in a spatial context we are exploring how network structure (genotype) relates to the possible gene expression patterns (phenotype). Fundamental design and evolutionary principles can be extracted by mapping out the space of possibilities in this fashion, which is not possible by analyzing real gene networks on a case by case basis. As a byproduct, by using a realistic model of gene regulation, new non-intuitive patterning mechanisms can also be discovered and suggested to account for observed patterning phenomena in real biological systems.
Table of Contents

Declaration I
Acknowledgements II
Abstract III
Table of Contents IV
List of Abbreviations XII

Chapter 1: Introduction 1
Pattern formation 2
Strategies of pattern formation 4

Cell autonomous 4
Division of a heterogeneous egg 7
Asymmetric mitosis 7
Temporal dynamics coupled to mitosis 7

Morphogenetic 7
Inductive 8
Turing’s Reaction-diffusion model and lateral inhibition 9
The Gradient-threshold model 13

Inductive patterning mechanisms and gene networks 16
Network Classification 16
Modules 20

Example of modules in pattern formation: 20
The Segment polarity network 20

Modeling inductive patterning gene networks 21
Feedback and the need for modeling 21
Scales of modeling 21
Fine grained versus abstract modeling 22
When to use modeling 24
### How to model a gene network

<table>
<thead>
<tr>
<th>Types of simulation model</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The Boolean idealization</td>
<td>26</td>
</tr>
<tr>
<td>Differential equation model</td>
<td>28</td>
</tr>
<tr>
<td>Stochastic models</td>
<td>29</td>
</tr>
<tr>
<td>Stochastic differential equations</td>
<td>30</td>
</tr>
</tbody>
</table>

### Topology versus parameters

- Robustness
  - Mutational/genetic robustness
  - Noise robustness

### From a single genotype to multiple genotypes

- Genotype-Phenotype maps
- Adaptive landscapes
  - Topology of adaptive landscapes

### Neutral networks

### Questions in pattern formation

### The project

- Concepts to address these questions
  - Concept of a 'complete topology atlas'
  - Concept of modularity

### Thesis structure

#### Chapter 2: Generating the Complete Atlas of Topologies

### Introduction

- The Complete topology atlas concept
  - Topologies
  - Relationship between topologies
  - Hypercubes of gene network matrices
  - The 'Complete Topology Atlas', a stripped down hypercube
  - The relationship between the concept of a topology atlas and
Genotype space

Methods

Generating the topology atlas

Generation of a complete set of unique topologies up to four genes

Parallelization of the topology generation

Generating the topology lists with a pre-pattern

Generating the neighbor structure

Results and Discussion

Generating the topology atlas

The atlases

Concluding remarks

Summary

Chapter 3: Simulating the complete topology atlas

Introduction

Simulating the gene networks

The gene regulation model

Features included in this model

Regulatory saturation

Signaling

Noise

Initial conditions

Features not included in this model

Bimodal interactions

Delays

Choice of physical domain

Modeling different number of dimensions

Boundaries

Pre-pattern

Features requires to define the fitness function
Pattern 86
Noise robustness 86
Equilibration 86

Methods 89
The discretized form of the equations 89
Tuning the core simulation 90
Parameter range distributions 90
Initial conditions 91
The Pre-pattern 91
The pattern and noise filter 92
Equilibrium filter 97
Parallelization of the simulations 97

Results and Discussion 98
Simulating 98
Standard data sets 98
Summary 101

Chapter 4: Exploring the phenotype map 102
Introduction 103
Are there distinct groups of patterns? 103
Methods of classifying patterns (Supervised strategies) 103
Unsupervised learning approaches 105
Clustering 105
Types of clustering procedure 110
Partitional (flat) clustering 110
Hierarchical clustering 110
Estimating the number of clusters 111
Exploring the shape of phenotype space: Dealing with multi-dimensional space 115
Dimensionality reduction 115
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph theoretical methods</td>
<td>115</td>
</tr>
<tr>
<td>Spring functions</td>
<td>116</td>
</tr>
<tr>
<td>Overview of our approach</td>
<td>119</td>
</tr>
<tr>
<td>Methods and Results</td>
<td>120</td>
</tr>
<tr>
<td>Abstracting pattern (phenotype) space to a manageable number of points</td>
<td>120</td>
</tr>
<tr>
<td>Granularity</td>
<td>120</td>
</tr>
<tr>
<td>Unsupervised learning approaches</td>
<td>123</td>
</tr>
<tr>
<td>Filters</td>
<td>123</td>
</tr>
<tr>
<td>Similarity function</td>
<td>127</td>
</tr>
<tr>
<td>Hierarchical clustering</td>
<td>127</td>
</tr>
<tr>
<td>What do we learn from these dendrograms</td>
<td>130</td>
</tr>
<tr>
<td>Different types of similarity function are good for grouping</td>
<td>130</td>
</tr>
<tr>
<td>different patterns</td>
<td>134</td>
</tr>
<tr>
<td>Pattern (phenotype) space is a continuum</td>
<td>134</td>
</tr>
<tr>
<td>Spring functions</td>
<td>134</td>
</tr>
<tr>
<td>Pattern matching</td>
<td>137</td>
</tr>
<tr>
<td>Discussion</td>
<td>141</td>
</tr>
<tr>
<td>What is the structure of pattern space? Are there distinct groups of</td>
<td>141</td>
</tr>
<tr>
<td>patterns?</td>
<td></td>
</tr>
<tr>
<td>Grouping the patterns by pattern matching</td>
<td>143</td>
</tr>
<tr>
<td>Summary</td>
<td>143</td>
</tr>
<tr>
<td>Chapter 5: Exploring neutral networks and evolution</td>
<td>144</td>
</tr>
<tr>
<td>Introduction</td>
<td>145</td>
</tr>
<tr>
<td>Exploring the NN concept for gene regulatory networks</td>
<td>145</td>
</tr>
<tr>
<td>Applying the NN concept to pattern forming gene regulatory networks</td>
<td>148</td>
</tr>
<tr>
<td>Nearly Neutral Networks (NNN's) of gene networks with a continuous</td>
<td>151</td>
</tr>
<tr>
<td>model</td>
<td></td>
</tr>
<tr>
<td>Neutralism and innovation: Crossing between neutral networks</td>
<td>153</td>
</tr>
</tbody>
</table>
Our Approach 154

Methods and Results 155

Can we find nearly neutral networks? Determining the number of topologies in the largest connected component 155

What is the shape of these nearly neutral networks? 160

Sampling effects on loop holes 165

Routes for evolution: The existence of ‘loop holes’ 161

Parameter space overlap 167

Discussion: 170

The existence of Nearly Neutral Networks 170

The shape of Nearly Neutral Networks: ‘The existence of loop holes’ 170

Innovation of phenotype: moving between neutral networks 171

Parameter overlap 172

Summary 173

Chapter 6: Exploring Mechanism 174

Introduction 175

Encoding pattern forming mechanisms 175

Encoding genotype and phenotype 176

Our Goal: Encoding mechanism 176

Complexity landscapes 179

What is the topology of these landscapes? 179

Hybrid mechanisms 180

Exploring robustness 180

Our Approach 182

Methods 183

Generating complexity landscapes 183

Identifying core topologies 185

Exploring how core topologies are functioning 187

Exploring robustness of core topologies function 187
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzing hybrid topologies</td>
<td>188</td>
</tr>
<tr>
<td>Results and Discussion</td>
<td>189</td>
</tr>
<tr>
<td>The single stripe mechanisms</td>
<td>189</td>
</tr>
<tr>
<td>Exploring robustness: the parameter space that encompasses the mechanism</td>
<td>192</td>
</tr>
<tr>
<td>The distinct nature of the complexity atlas</td>
<td>198</td>
</tr>
<tr>
<td>Hybrid topologies</td>
<td>199</td>
</tr>
<tr>
<td>Summary</td>
<td>203</td>
</tr>
<tr>
<td>Chapter 7: Discussion</td>
<td>204</td>
</tr>
<tr>
<td>Key concepts</td>
<td>205</td>
</tr>
<tr>
<td>The Complete Topology Atlas concept</td>
<td>205</td>
</tr>
<tr>
<td>The neutral network concept</td>
<td>205</td>
</tr>
<tr>
<td>The landscape concept</td>
<td>206</td>
</tr>
<tr>
<td>Encoding mechanism</td>
<td>206</td>
</tr>
<tr>
<td>Questions and answers</td>
<td>207</td>
</tr>
<tr>
<td>Concluding remarks</td>
<td>210</td>
</tr>
<tr>
<td>Chapter 8: General Materials and Methods</td>
<td>211</td>
</tr>
<tr>
<td>Hardware used</td>
<td>212</td>
</tr>
<tr>
<td>Software used</td>
<td>212</td>
</tr>
<tr>
<td>Appendix A: The GUI control options</td>
<td>213</td>
</tr>
<tr>
<td>Appendix B: Patterns resulting from the granularity algorithm</td>
<td>226</td>
</tr>
<tr>
<td>Appendix C: Complexity landscapes for the single stripe pattern using alternate gene regulation models and pre-patterns.</td>
<td>232</td>
</tr>
<tr>
<td>Appendix D: Sampling parameter space</td>
<td>240</td>
</tr>
</tbody>
</table>
Appendix E: The core topologies producing the single stripe gene expression pattern.

References
List of Abbreviations

CPU  Central Processing Unit
DKK  Dickkopf (protein)
DNA  Deoxyribonucleic Acid
DPP  Decapentaplegic
E    Edge
ECM  Extracellular Matrix
EGF  Epidermal Growth Factor
FGF  Fibroblast Growth Factor
GRN  Genomic Regulatory Network
GUI  Graphical User Interface
IL-1R Interleukin-1 Receptor
LEF  Lymphoid Enhancer binding Factor
LGL  Large Graph Layout
LRP  Lipoprotein Receptor related Protein
MM  Michalis-Menten
mRNA messenger RNA
MST  Minimal Spanning Tree
NN  Neutral Network
NNN  Nearly Neutral Network
PHP  Hypertext Preprocessor
PNG  Portable Network Graphics
RNA  Ribonucleic Acid
ROI  Region of Interest
SDE  Stochastic Differential Equation
SSE  Sum of Squared Error

Tcl/Tk  Tool Command Language Took Kit

TCF  T Cell Factor

TGF  Transforming Growth Factor

TLR  Toll-like Receptors

V  Vertex

WNT  Wingless-type MMTV Integration Site Family (protein)
1) Introduction
**Pattern formation:**

The process of pattern formation poses a fundamentally simple question; how does a spatially organised group of cells of different types arise from a group of cells that were originally identical. Patterning rarely if ever occurs in such a simple scenario, some form of pre-pattern usually exists (Wolpert 2002). However a common feature which characterizes the patterning process is that the information content in terms of cell types and their spatial positions is increased.

Developmental biologists have been studying the process of pattern formation since it was recognised in the 19th century that cells of the embryo arose by cell division from the zygote. This discovery led directly to the question of how these cells became different from each other. August Weissman put forward the ‘mosaic’ model to account for this issue (figure 1.1). In the mosaic model the zygote contains molecular determinants within its nucleus. Cells must divide asymmetrically so that each daughter cell receives a different set of determinants. These determinants determine how a cell will behave and thus what cell types are available. The demonstration by Hans Driesch (Wolpert 2002) that a sea urchin develops completely normally after separation at the two cell stage of development suggested that cells did not become different from one another by a mosaic type model. This discovery also suggested that cells must communicate with one another. The later discovery of the phenomenon of induction whereby one tissue directs the development of another established this concept without doubt. In 1924 Spemann and his assistant Hilde Mangold showed that a partial second amphibian embryo could be induced by grafting one small region of a newt embryo onto a new site of another embryo (Spemann and Mangold 1924).

The establishment of the central dogmas of biology during the 20th century that genes encode mRNA which encodes proteins suggested how genes can influence development and thus pattern formation, since it could be the complement of proteins that a cell contains that largely determines its behaviour and thus its type (Wolpert 2002). Not all of these genes are important for understanding the control of pattern formation however as most of them are housekeeping genes, or genes that perform a
Figure 1.1: The Mosaic model: Adapted from Wolpert: principles of development. Factors (yellow triangle, green circle, purple square and red star) in the nucleus (grey) that are distributed asymmetrically to daughter cells during cleavage. These factors determine the future development of the cells that inherit them.
given function for a given cell type but are not involved in the decision to become that cell type (Wolpert 2002).

Gene expression is a complex process which can be regulated at several levels such as the level of RNA transcription, degradation, processing, transport (in eukaryotes), translation and post translation modification and turnover (Alberts et al., 1994; Weinzierl 1999). Any gene involved in activating or inhibiting (switching on or off) the expression of any other gene at any of these levels can be involved in the control of pattern formation. The most important genes for controlling pattern formation are those that encode transcription factors, proteins involved in activating or inhibiting the expression of other genes at the level of RNA transcription (Wolpert 2002). Transcription factor gene products act in combination to generate most of the molecular circuitry that controls developmental pattern formation (Davidson and Erwin 2006). Given that all of the functional proteins that take part in pattern formation mechanisms are themselves the products of genes then the concept of a genetic regulatory network consisting of interactions between a closed set of DNA, RNA, protein and metabolites emerges (Kumar and Bentley 2003). This is an example of what is commonly referred to as a ‘gene network’, a very simple example of which is illustrated in figure 1.2.

**Strategies of pattern formation:**

Many different mechanisms of pattern formation have been shown experimentally and can be split onto three main categories; cell autonomous mechanisms, morphogenetic mechanisms and inductive mechanisms (Salazar-Ciudad et al., 2003). An illustration of the different mechanisms is given in figure 1.3.

**Cell autonomous mechanisms:**

Cell autonomous patterning mechanisms are those that do not require any cell-cell communication either mechanically or by signalling. All cell autonomous patterning mechanisms involve one cellular behaviour; mitosis. Cell autonomous mechanisms can be implemented by the division of a heterogeneous cell, asymmetric mitosis or with temporal dynamics coupled to mitosis.
Figure 1.2: A gene network showing two genes A and B (Thick blue and orange lines respectively). Activations are represented by arrows and inhibitions by 'dead-end' lines. In this example gene A is activating gene B and gene B is inhibiting gene A. Drawn using the Biotapestry software.
http://www.biotapestry.org/
Cell autonomous mechanisms

Division of heterogeneous egg

Asymmetric mitosis

Temporal dynamics with mitosis

Inductive mechanisms

Hierarchic

Emergent

Morphogenetic mechanisms

Directed mitosis

Differential growth

Apoptosis

Migration

Differential adhesion

Contraction

Matrix modification

Figure 1.3: Mechanisms of pattern formation. Taken from Salazar-Ciudad et al., 2003
Division of a heterogeneous egg:
Egg cells usually divide in a heterogeneous manner with few exceptions (mammalian and some turbellarian clades). This occurs by different RNA's and proteins being located in different regions of the cell. Thus when the cell divides each daughter will inherit a different complement of RNA and proteins. These different compliments of RNA and proteins will affect the behaviour of the cell and thus the type that it is resulting in patterned tissue.

Asymmetric mitosis:
Asymmetric mitosis only differs from division of a heterogeneous egg in that the gene products or mRNAs are transported asymmetrically to the future daughter cells while the mother is still dividing. The result is again two daughter cells with different RNA and protein compliments, which are the determinants of cell type.

Temporal dynamics coupled to mitosis:
Another example of a cell autonomous patterning mechanism is when internal temporal dynamics are coupled to mitosis. This type of mechanism results in a spatial manifestation of a temporal pattern, which involves cycling gene expression that has become decoupled from cell division. If upon division one of the daughter cells stops or resets its temporal dynamics, then cells can acquire different states depending on the time of their mitosis. An example of this is the suggested clock and wavefront model for somite formation in vertebrates. If the clock stops cycling in one phase of the oscillation then a somite cell type will form, whilst if it stops cycling in the other phase a boundary cell type will form. As the tissue grows, a series of somite then boundary decisions will be made by the cells resulting in a repetitive pattern of cell types.

Morphogenetic mechanisms:
Morphogenetic mechanisms pattern can be defined as those that change the relative arrangements of cells over space without affecting their states and do not involve signalling between cells (although signalling may have been involved at a prior stage). Another key feature of this type of mechanism is that it involves cells of
multiple different types. The spatial reorganisation of these cell types results in a new pattern of cell types.

One mode of this type of mechanism is cell migration, which can be directional or random. The migration of pre-muscle cells into the developing vertebrate limb is a specific example of a morphogenetic mechanism involving migration (lee et al., 1999). Other modes by which this type of mechanism can occur include directed mitosis, differential growth, apoptosis, differential adhesion, contraction and matrix swelling, deposition and loss.

**Inductive mechanisms:**

This category of pattern formation includes what mathematicians usually refer to by the phrase "pattern formation". Inductive mechanisms involve one cell or tissue influencing the fate of another cell or tissue (Salazar-Ciudad et al., 2000). These mechanisms involve the phenomenon of induction described earlier. Cells can affect each other by secreting diffusible molecules, or by membrane-bound molecules or by chemical coupling through gap junctions. In these mechanisms tissue pattern changes directly as a consequence of cell state.

This type of mechanism can be implemented in a hierarchical or an emergent fashion (Salazar-Ciudad et al., 2000). The simpler version of these is the hierarchical where one cell or tissue influences the fate of another cell or tissue without being affected itself by the tissue it is influencing. In the more complicated emergent form, two or more cells or tissues reciprocally influence each others fate.

How networks of genes and their products act in combination with induction to generate pattern formation has been explored for several decades (Wolpert 2002). Several different strategies have been suggested to be responsible and some of them have become deeply entrenched in the theory of developmental biology. The three main suggested strategies; Turings reaction-diffusion model, Wolperts French flag model and lateral inhibition will now be discussed in a historical order.
Alan Turing originally proposed his reaction-diffusion mechanism as a simple way for which organisms could generate stripes of gene expression (figure 1.4 and Turing 1952). The mechanism is an emergent type that works by “diffusion driven instability” whereby small fluctuations in an otherwise well-mixed system of reacting and diffusing chemicals could become unstable, and amplification of these fluctuations could lead to a spatial pattern of these chemicals. He termed these chemicals morphogens and suggested that they could serve as a pre-pattern for development. The model was at the time counter intuitive since diffusion is usually seen as a force for evening out inequalities rather than being the driving force for the generation of an asymmetry. The mechanism involves two components; an auto-activator and a repressor. The auto-activator promotes the generation of the repressor whilst the repressor inhibits the generation of the auto-activator. Random fluctuations in an otherwise well mixed solution of these two substances will result in zones where the auto-activator can amplify its own production. This auto-activation state would spread through the whole solution if it not for the presence of the repressor. In order for stable patterns to be formed the diffusion of the repressor is much higher than the activator. The higher diffusion of the repressor relative to the auto-activator means that a higher concentration of repressor than activator can form in zones adjacent to where the amplification of auto-activator originally occurred. These zones of repressor stop the auto-activation spreading through the whole solution and result in a dynamic equilibrium of auto-activator and repressor concentration in the solution. In small or growing fields, monotonic gradients will be formed that can give positional information. In larger fields however stripes will be formed. 

The original equations that Turing suggested were the following (Turing 1952):

$$\frac{\partial x}{\partial t} = 5x - 6y + 1$$

$$\frac{\partial y}{\partial t} = 6x - 7y + 1 + D_y \nabla^2 y$$
**Figure 1.4:** Turing's reaction diffusion model. (Top) Two species; (A) an auto-activator and (B) a repressor. Activations are indicated by arrows and inhibitions by 'dead-end' lines. (Middle) In a two dimensional spatial context (as illustrated by the network in multiple squares) where both A and B can diffuse (represented by the yellow arrows). (Bottom) With appropriate parameters spots or stripes of A will form (concentration of A indicated by intensity of yellow).
Where $x$ is the auto-activator and $y$ is the repressor and $D_y$ is the diffusion coefficient of $y$. The problem with these equations is that they can result in negative concentrations and are thus not biologically reasonable. The rate of change of $x$ with respect to time is dependent on the removal term $-6y$ even if there are no $x$ molecules present. To remedy this problem, one would need a removal term that would require the number of molecules of $x$ disappearing per unit time to depend on the number of molecules of $x$ present (Kumar and Bentley 2003). The following set of equations that satisfies this requirement was proposed (Gierer and Meinhardt 1972):

$$\frac{\partial x}{\partial t} = \frac{\rho x^2}{y} - \mu_x x + D_x \nabla^2 x + \sigma_x$$

$$\frac{\partial y}{\partial t} = \rho x^2 - \mu_y y + D_y \nabla^2 y + \sigma_y$$

Where $x$ is the activator and $y$ is the repressor, $t$ is time, $D_x$ and $D_y$ are diffusion coefficients and $\mu_x$ and $\mu_y$ are the decay rates. The source density $\rho$ describes the ability of the cells to perform the autocatalysis whilst $\sigma_x$ and $\sigma_y$ are activator independent production of the two species respectively. At low activator concentrations a small $\rho_x$ can initiate the system. This work suggested that the Turing-type mechanism could be at work in real patterning systems and prompted the suggestion of the mechanism to be responsible for various patterning phenomena such as the generation of stripes of Even-skipped and Fushi Tarazu gene expression in Drosophila (Hunding et al., 1990). However as will be shown later, the mechanism responsible for generating these gene expression patterns was found to be based on a hierarchical type mechanism.

The striped pattern of the Angel fish is thought to be controlled by a Turing-type reaction diffusion mechanism (Kondo and Asai, 1995). The stripes on the angel fish do not enlarge as the fish grows, but instead their numbers increase and the space between them is always the same. This would not be expected if a hierarchical
mechanism was in control, but would be exactly what was predicted by a reaction-diffusion type mechanism.

The first direct evidence of a reaction-diffusion type mechanism has been gained recently (Sick et al., 2006). WNT and its inhibitor DKK were found to be the main determinants of murine hair follicle spacing. A mathematical model of this mechanism predicts both the normal and mutant phenotypes strongly suggesting that this is the mechanism responsible.

Skeletal patterning in the mouse limb has also been suggested to be under the control of a reaction-diffusion mechanism (Newman and Frisch, 1979). The most fundamental skeletogenic process involves the spatial separation of precartilage mesenchyme into chondrogenic and non-chondrogenic domains. Morphogens of the TGF-beta family induce the local aggregation or condensation of these cells by a process that involves the up-regulation of the cellular adhesion glycoprotein fibronectin. Cartilage differentiation then forms at those regions of condensation. In vitro work by Miura and co-workers (Miura and Shiota, 2000; Miura et al., 2000) suggested that TGF-beta2 acts as an activator molecule in a pattern-forming mechanism reminiscent of the Turing mechanism because of several observations; TGF-beta2 was found to be localized to chondrogenic sites and to promote the product of its own RNA suggesting a positive feedback. Also beads soaked in TGF-beta2 suppressed chondrogenesis around the beads, suggesting that it induces a lateral inhibition mechanism. Furthermore it was shown that TGF-beta2 promotes the expression of N-cadherin, an adhesion molecule known to be involved in this patterning system.

Other phenomena which reaction reaction-diffusion type mechanisms have been suggested to underlie include pigment patterning in the butterfly wing (Nijhout, 2001) and featherbud spacing in the avian wing (Jiang et al., 1999; Prum and Williamson, 2002).

Lateral inhibition works in a similar way to the Turing-type reaction diffusion system, but via direct cell-cell signalling rather than diffusion. The other important difference is that lateral inhibition is irreversible; once the field of cells has generated its final pattern, it does not change.
An example of lateral inhibition is in neural development in *Drosophila* (Skeath and Carroll, 1992; Skeath, 1999). In *Drosophila*, the Presumptive central nervous system is specified at an early stage of embryonic development as two longitudinal stripes of cells, just dorsal to the ventral midline (figure 1.5). This region is called the neurectoderm and consists of ectodermal cells with the potential to form neural cells and epidermis. The neurectoderm is subdivided along both its antero-posterior and dorso-ventral axis into a precise orthogonal pattern of proneural clusters. Within each of these proneural clusters only one cell via an apparently random decision will go on to become a neuroblast and the rest epidermal cells. All of these cells originally had the potential to become the neuroblast however as has been shown by cell ablation experiments (Taghert *et al.*, 1984; Doe and Goodman 1985; Simpson and Carteret 1989). The other cells are prevented from taking a neuronal fate by lateral inhibition from the cell that will become the neuroblast.

The notch and delta genes play a key role in lateral inhibition (Artavanis-Tsakonas *et al.*, 1995; Kimble and Simpson, 1997). They encode transmembrane proteins, with delta being the ligand of the receptor notch. Activation of the notch by delta leads to inhibition of the proneural genes in that cell, thus losing its ability to become a neuroblast.

*The gradient-threshold model:*

Lewis Wolpert put forward his gradient-threshold model as a general mode by which pattern formation can occur (figure 1.6 and Wolpert 1969, 2002). Wolpert used the metaphor of the French flag in which to pose the problem of pattern formation. The French flag is a simple pattern; one third blue on the left, a third white in the middle and a third red on the right. The flag represents a cell field and the different colours of the flag represent different cell types. The fact that there are many different sizes of flag but the proportions of each colour are the same is a metaphor for regulation in the developing embryo.

One way that cells could take these different identities is by them acquiring positional information. That is cells acquire some information about where they are with respect to a boundary and use this information to adopt different fates (blue, white and red in the case of the French flag).
Figure 1.5: An illustration of Lateral inhibition in proneural clusters in the *Drosophila* neurectoderm. Neurectoderm cells are represented by grey circles. Cells of a proneural cluster are shown in pink. One cell at random will become the neuroblast (darker orange and then red) and inhibit the other cells in the cluster (lighter orange) from taking the same fate using the notch-delta pathway. These other cells will go on to become epidermis like the rest of the neurectoderm.
Figure 1.6: The Gradient-threshold model. The concentration of morphogen is indicated by the height of the orange triangle. At different thresholds (dashed lines) of morphogen concentration cells (grey squares) take different fates as indicated by the blue, white and red colors.
How cells acquired their positional information could vary, though the simplest implementation is what Wolpert called a gradient-threshold type mechanism. This mechanism requires a morphogen, a chemical species whose concentration varies in space and that is involved in pattern formation. It was suggested that cells could respond to different concentrations of the morphogen to adopt different fates. In the problem of the French flag, we would assume that there is a source and a sink of the morphogen at either end of the one dimensional field of cells resulting in a gradient. Cells could then respond in 3 different ways depending on the concentration of this gradient; the blue way at high concentrations, the white way at intermediate concentrations and the red way at low concentrations. The result is the French flag pattern of cell types.

An attractive feature about this suggested mode of pattern formation was that it uncoupled the specification of positional information to how that information was interpreted. The same specification process could be used to generate the Italian flag for example if the interpretation stage was changed.

**Inductive patterning mechanisms and gene networks:**
In this work we have chosen to focus on inductive patterning strategies. The aforementioned inductive patterning strategies are implemented by patterning gene networks (Salazar-Ciudad *et al.*, 2003). A key question is then: How are different types of "pattern strategy" encoded by different gene networks. What are the specific mechanisms that work by these strategies?

**Networks Classification.**
Gene networks have been classified based on the amount of feedback that they contain between genes (Salazar-Ciudad *et al.*, 2000). A network containing no feedback can be classified as a completely hierarchical network (Figure 1.7 left) whilst a network where there is feedback between all genes can be classified as a completely emergent network (Figure 1.7 middle). These are two extremes of a vast spectrum, since networks can have a mixture of both hierarchical and emergent features (Figure 1.7 right).
Figure 1.7: Three different structures of network. Letters are genes, activations are represented by arrows and inhibitions by 'dead-end' lines. (Left) Completely hierarchical, where there is no feedback. (Middle) Completely emergent where every gene has feedback. (Right) Hierarchical/Emergent network which has features of both.
In a spatial context a hierarchical network structure results in one cell or tissue (A) signalling to another cell or tissue (B). Tissue B can back-signal to tissue A but does not affect tissue A’s original signal.

The paradigm example of a network with a hierarchical structure is the network that segments the *Drosophila* along the anterior-posterior axis (Nusslein-Volhard and Wiechaus 1980; St. Johnson and Nusslein-Volhard 1992; Ingham, 1988; Jackie et al., 1993). The cascade has four tiers the highest three of which are illustrated in figure 1.8. Maternally provided gene products, like the bicoid and nanos gene products activate a class of zygotically expressed genes, the gap genes. The first signs of a repetitive pattern become apparent after the activation of the next class of genes, the pair-rule genes. These are expressed as a repetitive pattern of seven transverse stripes along the anterior-posterior axis. Each of these stripes of expression are controlled independently. The final tier in this cascade is the segment polarity genes that are expressed in 14 transverse segmental stripes. The segment polarity genes delineate the boundaries of the parasegments and their anterior-posterior polarity. The parasegments are the embryonic segmental units. This network works in a hierarchical way. Removing any of the genes in the lower tiers does not affect the behaviour of the genes in the higher tiers. Even this network has some feedback between genes however and thus can-not be described as completely hierarchical.

In contrast in a completely emergent network, removal of any of the interactions or components changes the behaviour of the network. A paradigmal example of an emergent network is the Turing type reaction diffusion mechanism described earlier. If any interaction or component is removed from this system then spots or stripes cannot be formed. The lateral inhibition mechanism is also an example of an emergent network for the same reason.

The classification of network structure as hierarchical or emergent can also be misleading since ‘emergence’ is usually a term reserved for dynamic behaviour. Hence a network with lots of feedback may be defined as an emergent network, even though it does not behave in an emergent way.
Figure 1.8: Three tiers of the *Drosophila* hierarchical gene expression cascade controlling segmentation. (Left) The genetic cascade (Right) Confocal images of embryo stained with fluorescent antibodies. (Top) Bicoid. (Middle) Hunchback (Bottom) Even-skipped. Images taken from Spirov and Holloway 2002.
Modules:
Another important aspect of gene network architecture is that of modularity. There are many different definitions for what a module is, which reflects their use in different fields (Raff and Raff, 2000; Kumar and Bentley 2000). A general description (Kumar and Bentley, 2000) states that: a module is a biological structure, process or pathway characterised by more internal than external integration. This independence requires chemical isolation, which can originate from special localisation of chemical specificity (Kumar and Bentley, 2000). There are two basic definitions of a module: structural and functional. The most intuitive modules are those of structure. Elements of phenotype for example the vertebrate limb or the set of gene sequences that constitute the paralogue groups of the hox cluster (Kumar and Bentley, 2000). Less intuitive modules are those of function, such as a signal transduction pathway (Kumar and Bentley, 2000). The independence of such modules usually comes from the components being isolated temporally.
Gene networks capable of pattern formation have been found to function in a modular fashion, examples of which are given below. Such modular nature has important consequences for how we can approach studying these networks as will be seen later.

Example of modules in pattern formation: The Segment polarity network:
All insects possess homologous segments, but segment specification can differ greatly among insect orders (Davis and Patel 2002). In some species such as Drosophila patterning occurs before cellularization in the syncitium, whilst in others it occurs after cellularization. In other groups such as beetles and parasitic wasps, pair-rule homologues are expressed in a consistent fashion with that of the Drosophila though in a cellular field. In others such as locusts, pair-rule genes may not control segmentation though some segment polarity genes and their interactions are conserved (Davis and Patel 2002). This has led to the suggestion that the segment polarity network is modular and performing the same function in these different contexts (von Dassow et al., 2000). The discovery that the segment polarity network performs its function in a robust fashion adds weight to this argument since it
suggests that the network will perform the same function with a variety of different input stimuli (von Dassow et al., 2000).

**Modelling inductive patterning gene networks:**
I will now describe why we model inductive patterning gene networks and what can be gained from such modelling.

**Feedback and the need for modelling:**
"Many individual chemical reactions in the cell involve a single enzyme catalysing a single, well defined transition in a substrate" (Mogilner et al., 2006). Such reactions produce simple and intuitive behaviours that can be plotted with a single curve and understood using a couple of Michalis Menten constants (Mogilner et al., 2006). Therefore it is simple to go from experimental measurements to theoretical understanding in these cases (Mogilner et al., 2006). The overall behaviour occurs on few timescales and can be represented by few parameters (Segel, 1988). Even when only a few players interact in non-linear ways however, complex and surprising phenomena can result. For example in the Belousov-Zhabotinskii reaction, a simple mixture of just a few chemicals spontaneously forms pulsating spiral waves in a beaker (Tyson, 1976).

Patterning gene networks often contain feedback (Freeman, 2000), which makes their dynamic behaviour hard to predict, sometimes displaying counter-intuitive features. To understand the effect of this feedback we need computers to perform numerical simulations on these networks. As Noble said "Beyond a certain level of complexity, which is often just a couple of components, qualitative thinking fails and qualitative modelling becomes essential" (Noble, 2002).

**Scales of modelling:**
The most critical issue when developing a model is choosing its scale. A good description is that "Traditional modelling is the art of abstracting a simpler system, which is a "caricature" rather than a "photograph" of the actual biological system" (Mogilner et al., 2006). For example, to capture the biochemistry underlying the dynamic instability of microtubules many possible scales exist. For example one
could choose to model the movements and interactions of each individual tubulin dimer as they exchange off and on to the ends and undergo GTPase reactions (VanBuren et al., 2005; Mogilner et al., 2006) or alternatively one could simply propose stochastic rules describing how the length of microtubules is expected to change over time (Gliksman et al., 1993; Mogilner et al., 2006). There are also many level in between these two extremes of how fine grained a model can be.

The choice of how fine-grained to make a model depends on several factors. The finer-grained model will require more detailed prior information about the components. However the coarse model will require extensive intuition about the cellular phenomenon. It is important that a model uses the right level of abstraction to answer the specific question that a researcher is asking, for example a detailed model is needed if one wants to make detailed predictions about a specific system whereas an abstract model is better for understanding how that system works and to extract principles.

**Fine grained versus abstract modelling:**

Fine detailed models are appropriate when one wants to predict with great accuracy when and where certain events will occur. However, building extremely detailed models of biological phenomena has dubious value for explaining how a particular system actually works (Mogilner et al., 2006). An example of this is the model of TLR and IL-1R signalling networks responsible for the immune response (Oda and Kitano, 2006). This model contains 652 variables describing the molecular species with 444 reactions between them and a huge number of parameters. How any useful information can be extracted from such a model for the means of understanding is difficult to envisage.

For the purpose of understanding, we should always seek the simplest model which explains the observed data. These detailed models for gene networks can be used to show that the network that researchers thought was performing a particular function is entirely consistent with the data. However they are often so fine that the details obscure a clear understanding of how the system is working. They are also too specific to be able to extract principles of the way the systems are working. Furthermore one important aspect of understanding these gene networks is through
exploration of parameter space and perturbation analysis which is often too computationally expensive for fine grained models. There are many examples of how modelling abstractions of real systems not only can be consistent with the data, but also show us how the mechanisms are working. This ‘how’ question is essential if we are to truly understand these systems as illustrated by the following examples:

The generation of stripes of gene expression on the Drosophila blastoderm is controlled by a group of genes called the segment polarity genes. They function in a network that is conserved in many insects suggesting that it is acting as a module. When modelled, this module was found to be extremely robust to parameter variation (von dassow et al., 2000). This original model however said little about how or why this network was robust. A second paper exploring the network addressed this question and showed that the robustness was due to two positive feedback loops (Ingolia 2004). Different cells were found to adopt different stable states of expression (bistability) corresponding to the different cell types in the segment polarity pattern. Positive feedback loops only lead to bistability when the parameters that describe it satisfy a particular inequality. This work showed that this bistability is necessary for the segment polarity pattern and that it is a strong predictor for when parameters will result in the correct pattern because they satisfy this inequality. Positive feedback had previously been shown to produce discrete stable expression states that are insensitive to small changes in parameter or initial conditions (Thomas and Kaufman 2001). Thus the positive feedback loops explain the robustness of the segment polarity network.

A second clear example of successful abstract modelling is the modelling of the gap gene system that works further upstream of the segment polarity network described earlier during Anterior-posterior patterning of the Drosophila blastoderm (Reinitz and Sharp, 1995; Perkins et al., 2006). Here the exact details of how genes activate and inhibit each other are abstracted to a Hill function. This group used the inverse approach to reverse engineer the network responsible for the dynamics of gene expression in this process. They used the known gene expression profiles of the gap genes as the constraining data and then looked for the network and parameters that
were the best at explaining this data using a simulated annealing approach. By not assuming anything about how the network was connected (only that there were 5 genes involved) they were able to generate the known network that generates these gene expression dynamics. The relative parameter values that they obtained also agreed with those which were known to be responsible for the real patterning process.

**When to use modelling:**
A lot of the literature in modelling biological phenomena claims that having real data is essential (Molginer et al., 2006). They suggest that modelling should only be used at a particular time during experimental practice. It should not be used too early when an absence of data leaves a model unsubstantiated and not too late when everything is clear without a model (Molginer et al., 2006). This is only true for very fine grained models designed to ask questions about specific phenomena, for example when we want to know the quantitative details of a system. For abstract models where the purpose of the model is to extract some kind of principle, data is not essential. What is essential however is a clear grasp of the way the system works. For example how the components interact together.

**How to model a gene network:**
For computer simulations to be carried out on gene networks it is useful for them to be described in some common format. The most common way to represent a network is as a directed graph (Kumar and Bentley 2003). A directed graph is made up of vertices (V) and edges (E) as seen in figure 1.9. The vertices are the nodes on the graph whilst the edges constitute the interactions between the nodes. A directed graph is one where these edges (interactions) have a direction such that one node is effecting another in a positive or negative fashion.

**Types of Simulation Model:**
Gene networks are modelled by a simulation that will result in the change of the state of the gene products with time. There are four main simulation models for examining the dynamic aspects of a gene network and thus its patterning potential; the Boolean
Figure 1.9: In a directed graph, each of the edges can be defined in a set of parenthesis. The direction of the interaction is indicated by the order of the two vertices given in the parenthesis, the first vertex being the effector and second the effected (A and B). The state of the interaction (whether it is a positive or negative interaction) is given as the last symbol in the parenthesis (the negative). (De Jong, 2003)
idealisation, the differential equation model, the stochastic model and stochastic differential equations.

1) The Boolean idealisation:
The Boolean method relies on the simplification of gene states to a discontinuous variable; on or off. For this type of method the response functions of genes to inputs must also be idealised to on and off. Such an idealisation is vastly computationally less expensive than other methods. Boolean logic has the power to capture major features of a homologous class of nonlinear dynamical systems governed by sigmoidal functions (Kaufman 1993).

It has been suggested that types of sigmoidal curve can in some cases be idealized to an on-off system (Kaufman 1993). The argument for using this idealisation goes as follows:

Take the sigmoid input-output function and the proportional response input-output function for a gene shown in figure 1.10. Even with a soft sigmoidal function, the curve is initially below the proportional response curve. In this region any input will generate an output that is less than that input. Were that reduced output fed into another input then the same would happen and over many iterations the response would dwindle to zero. At the region of the sigmoid curve above the proportional response curve, the opposite would happen and the response would be amplified to the maximum. Hence in contexts where information is fed through this function multiple times these functions tend to sharpen their responses to ‘all or nothing’ type behaviour. This happens in situations where there is either positive feedback or a chain of events such as a cascade. All of the components of the chain must influence their target in the same way; either by activation or inhibition.

Hence the Boolean idealisation can capture the logical skeleton and account for major features of continuous systems governed by Sigmoidal functions. However, Boolean logical networks cannot faithfully mimic all of the features of continuous systems. These Boolean logical systems in particular cannot represent the internal unstable steady states of the continuous systems (Kaufman 1993). Considerations
**Figure 1.10**: Boolean logic. Illustrating the input-output relationship of a proportional response curve (Red line) and a sigmoidal response curve (Blue line). Adapted from Kauffman (1993).
that lead us to believe that Boolean models are inappropriate for modelling gene networks capable of pattern formation are partly theoretical and partly biological:

**Theoretical considerations:**
It is impossible to envisage how the two main strategies that have been suggested to underlie pattern formation; that of the Turing type mechanism and the gradient-threshold model, could be constructed in a Boolean framework. A gradient for instance can obviously not be modelled using a basic Boolean model since a gradient by definition requires intermediate values of gene product concentration. Furthermore, the Turing type mechanism requires the diffusion of a proportion of repressor molecules to zones adjacent to where the activator dominates. Proportions obviously do not exist in the Boolean framework.

**Biological considerations:**
Gene expression patterns observed in development in many cases are known to take intermediate values. Many examples exist in gene expression databases such as the Mouse Atlas (http://genex.hgu.mrc.ac.uk). This database specifically maps gene expression data onto a reference embryo using high, intermediate and weak levels. Therefore the dynamics of these patterning systems cannot be accurately captured by Boolean models of gene regulation.

Extensions of the Boolean idealisation have changed the method to a multiple state system in an attempt to make the systems more biologically relevant (Van Ham 1978). Though by extending the idealization in this way some of the benefits of modelling using Boolean logic such as the speed of simulation are lost.

**2) Differential equation model:**
Ordinary differential equations are currently the most widely used model for gene networks (Kumar and Bentley 2003). The concentrations of the components of the network are time dependent having non-negative real values. Gene regulation is modelled by non-linear equations that describe the rate of production or degradation
of one component as a function of the concentration of another (deterministic). The equations have the mathematical form:

\[ \frac{dx_i}{dt} = f_i(x) \]  

1) \[ x = [-1]^T \text{and } x_i \geq 0, 1 \leq i \leq n \]  

Equation 2 is the vector for all of the concentrations of the components of the network at the beginning of the simulation. There is a differential equation (1) for each of the components in the system. \( f_i \) is usually a non-linear function and describes the rate of change of component \( x_i \) with respect to the other components in the system, possibly including component \( x_i \) itself.

3) Stochastic models:
The assumption made in the differential equation model that the concentration of the components vary continuously and deterministically may not be true in the case of gene regulation as some components will be present in small quantities. In systems with such small quantities, fluctuations in the level of the components can have a dramatic effect on the output of the system (Singer 1953). Such a problem has led to the suggestion of using stochastic models. The stochastic model "master equation" takes the form (Gillespie 1977; Kumar and Bentley 2003):

\[ \frac{dp(X,t)}{dt} = \sum_{j=1}^{m} \left( \beta_j - \alpha_j p(X,t) \right) \]

Where here \( p(X,t) \) is the probability distribution that at time \( t \), the cell contains \( X_1 \) molecules of the first species, \( X_2 \) molecules of the second species, etc. \( m \) is the number of reactions that can occur in the system. Here \( \alpha \) and \( \beta \) are coefficients for the balance of the probability that reaction \( j \) will occur in the time interval given that
that the system is in state X or that the reaction will bring the system into state X respectively.

Whereas differential models of the network determine how the state of the system changes with time, stochastic models describe how the probability of the system being in a certain state changes with time. A drawback of using a stochastic simulation however is that it is even more difficult to solve by analytical means than the deterministic rate equation. Furthermore they are more computationally costly to simulate. Examples of simulations of the gene network using the differential equation and stochastic simulation models is shown in figure 1.11. Note the noisy aspect of the time evolution of the components in the stochastic model. This effect reflects the stochastic nature of the initiation of transcription and the number of protein molecules produced per transcript may have important developmental consequences.

4) \textit{Stochastic differential equations:}

A stochastic differential equation is a differential equation like that described above, but with a noise term. In this way it has features of both the differential equations and the stochastic equations. It is a way of including the effect of noise in simulations, without having to include the more complex stochastic equations.

\textit{Why include noise in a model?}

The importance of including noise in gene network simulation relates to the fact that development takes place in an inherently noisy environment and that it is apparent that gene networks controlling developmental patterning are robust to this noise (von Dassow \textit{et al.}, 2000). By employing a model that includes noise, one avoids fragile fine-tuned results.

\textbf{Topology versus parameters:}

To model a gene network various parameters need to be defined. A parameter is a quantity that defines certain characteristics of a system. The parameters used when modelling gene network depend on the scale of the model. Typical parameters used when modelling patterning gene networks in an abstract way include the strength of
Figure 1.11: Simulations with deterministic and stochastic models. (From Kumar and Bentley 2003). (Top) the network to be simulated. This is a finer-grained model of gene-gene regulation including both mRNA and proteins. (Bottom left) Simulation with the differential model. Note the smooth changing concentrations of the mRNA's and proteins. (Bottom right) Simulation with the stochastic model. Note the 'jerkiness' of the concentration changes.
the effect of one gene on another, diffusion rates of gene products, background expression rates and responsiveness of a gene (Sharp and Reintiz 1995; Salazar-Ciudad et al., 2000). The phenotype resulting from the simulation of a gene network can change if the parameter values are changed. Therefore topology alone is not sufficient to predict the behaviour of a gene network though it can be used to suggest the potential behaviours (Ingola 2004). Just how much behaviour depends on topology versus the parameters is a question that can be answered by modelling.

When modelling a real patterning gene network then, one must either have a detailed knowledge of those parameter values or many different parameter sets must be tested to explore the potential behaviours of the network and how robust those behaviours are to changes in the parameters (See below).

**Robustness:**
Robustness describes the quality of retaining function in the face of inherent change in a systems natural operating environment (Wagner 2005). A variety of other names have been used to describe the phenomenon such as buffering, canalization, developmental stability, efficiency, homeorhesis and tolerance, but robustness has come to be the most widely used. Robustness can be split into two categories, that of mutational/genetic robustness and that of noise robustness.

**Mutational/genetic robustness:**
Mutational robustness means that a system produces little phenotypic variation when subjected to genetic variation caused by mutations (Wagner 2005; Ciliberti et al., 2007). Mutational changes can change the connectivity of a gene network or simply change the strength of those interactions. Mutational robustness involving network connectivity changes has been demonstrated in the gene network of *E. coli* by swapping around the interactions of the network (Isalan et al., 2008). Few of these changes resulted in severe phenotypic effects even though many of these connectivity changes involved changing the inputs and outputs to sigma factor genes, those genes that are present at the top of the *E. coli* transcription factor network hierarchy.
Mutational robustness involving changes to the strength of gene-gene interactions is often called parameter robustness. The most thorough exploration of gene networks responsible for pattern formation parameter robustness has been explored in silico. This has been shown for the segment polarity gene network in Drosophila (von Dassow et al., 2000) and the lateral inhibition network in Drosophila neurogenesis (Meir et al., 2002). In these examples a substantial proportion of parameter sets were found to be consistent with the function that the networks perform in vivo. A staggering 90% of randomly chosen parameter values yields a functional network in the segment polarity example. Furthermore, the segment polarity network was found to function over a wide range of parameter values and most individual parameters could be varied substantially whilst the others were held fixed without changing the function. The actual shape of the functional region of parameter space was explored for the neurogenic network. It was found that there are large regions of parameter space capable of function.

*Noise robustness:*
Noise robustness means a system produces little variation in its behaviour when subject to environmental fluctuations. These fluctuations are common in gene regulatory networks where few molecules are involved. It is for this reason that stochasticity is generally included when modelling such networks. Noise can be viewed as a temporary change in parameters and hence noise robustness and parameter robustness are related in a well defined manner. A correlation has also been shown between noise robustness and mutational robustness in gene networks (Ciliberti et al., 2007)

*From a single genotype to multiple genotypes:*
A genotype can be defined as a topology with a specific set of parameters. Modelling a genotype on its own can give insight in to how the network is functioning and how robust that function is to noise. However many questions remain such as what happens when we change the strength of gene-gene interactions or even add extra ones or remove them completely? Does the network still perform the same function or does it perform some other function?
To address these questions we need to simulate multiple genotypes and ask what phenotypes (gene expression pattern) they produce. Exploring these types of question require the employment of some of the major evolutionary concepts developed over the last century such as genotype-phenotype maps, adaptive landscapes and neutral networks that will now be described.

**Genotype-Phenotype maps:**
The distinction between genotype and phenotype was first put forward by Wilhelm Johannsen in 1909. The genetic component of an organism is its genotype, the genetic information it acquires from its parents. Its physical appearance such as its internal structure and biochemistry at any stage of its development is its phenotype.

The concept of a genotype-phenotype map describes how the genotypes relate to the phenotypes which are linked by the process of development (Fontana 2002). It is known that this mapping is often non-linear thus making predicting phenotype from genotype difficult and vice versa. For example the relationship between the genotype and phenotype has been explored for the only tractable example; RNA sequence structure relationships where genotype is represented by sequence and phenotype by structure (Schuster et al., 1994). In this case, even small changes in RNA sequence can vastly change the RNA structure.

Producing a tractable genotype-phenotype map for gene networks capable of pattern formation in multi-cellular systems is more difficult since reliable algorithms that predict phenotype have only recently been developed (Sharp and Reinitz 1995) and are computationally expensive and hence exploration of such maps has thus far not been performed. Whereas, it is relatively easy to predict how a hierarchically structure gene network will behave, when there is feedback this is not the case. This feature makes the genotype-phenotype maps of patterning gene networks complex with non-linear mappings.

However a recent dynamic analysis using a non-spatial Boolean model of gene regulation (Ciliberti et al., 2007) has shed some light on this mapping by showing that networks with very different organisations can have a very similar phenotype, whilst networks with completely unrelated phenotypes can have a similar network structure.
Adaptive landscapes:
The concept of an adaptive landscape was first proposed by geneticist Sewall Wright in 1932 (Wright 1932). The metaphor of the adaptive landscape involves visualizing the process of evolution "as a journey across adaptive hills and valleys, mountains and ravines" (figure 1.12 and McGhee 2007). His original concept is what is termed a fitness landscape today. There are two crucial features of the adaptive landscape concept (McGhee 2007):

- The first crucial feature of the adaptive landscape concept is that it includes a space of all theoretically possible genetic combinations that living organisms might produce. Furthermore, one can visualize such a complex space by considering the possible combinations of 2 genes at a time. That is if the genetic trait A had 10 different variants or alleles and genetic trait B also had 10 different variants, then there would be 100 potential genetic variants in total. These genetic combinations could be represented by a 2 dimensional 10x10 grid.

- The second important feature of the concept is that "the majority of those 100 possible combinations probably do not exist in nature" (McGhee 2007). For example 10 may exist whereas 90 could exist but do not. Wright originally proposed that those 90 combinations do not exist because they would have a fitness of zero or in other words be lethal genetic combinations. Wright then proposed that amongst those 10 variants with non-zero fitness some would have higher fitness than others and therefore a third dimension could be added to the two dimensional grid that represents the 'degree of fitness' thus resulting in a 3 dimensional landscape. The 10 existent combinations of genetic traits would be located in peaks and slopes, whilst the 90 zero-fitness trait variables would be found on flat planes.

High regions of adaptive landscapes are called adaptive peaks, whilst low regions between the peaks are called adaptive valleys. "The degree of adaptation of the possible morphological traits is determined by functional analysis of the potential forms" or in other words, how well those variants function in nature (McGhee 2007).
Figure 1.12: An adaptive landscape showing a local and a global optimum. Each position along A and Y describe the form of the allele. The position on the X and Y plane is then the genotype. W is the allelic fitness which is a function of that genotype. (From Allen Orr 2005). The green surface describes the fitness of all possible genotypes. Here a high ‘global’ optimum can be seen with a lower ‘local’ optimum. There is a ridge between these two peaks.
An organism can be viewed as a point on these landscapes and a population of organisms can be viewed as a cloud of points. A basic rule of modelling evolution on adaptive landscapes, is that natural selection will operate to move a population up the slope of an adaptive peak, from lower degrees of adaptation to higher degrees of adaptation (McGhee 2007).

**Topology of adaptive landscapes:**

The topology of these adaptive landscapes has important consequences for how a population of organisms can move around the landscape (McGhee 2007). Stuart Kauffman (1993, 1995) has explored the topology of these landscapes by using computer simulations of evolution via the process of natural selection in what he termed ‘NK fitness landscapes’. In NK fitness landscapes, N is the number of genes and K is the number of other genes which affect each of the N genes. The fitness of any one of the N genes is thus a function of its own state and the other genes that affect it (McGhee 2007). Therefore epistatic interactions are modelled.

These simulations showed that there are two extremes of the type of these NK fitness landscapes. A ‘fujiyama’ type landscape at K equals zero, and a totally “rugged” landscape at K equals N minus one (the maximum possible K value). In the Fujiyama landscape a single peak exists with very high fitness and smooth slopes. Such a fitness landscape exists when there are no epistatic interactions; that is every gene has an independent contribution to fitness. A totally rugged landscape consisting of many peaks of very low fitness is at the other extreme where each genes contribution to the fitness is effected by every other gene. A spectrum of landscapes exists between these two extremes, from the smooth Fujiyama to increasingly rugged. A further descriptive feature is whether these landscapes are isotropic (where the large peaks are distributed evenly across the landscape) or nonisotropic (where the large peaks tend to cluster together).

Kauffman has argued that the process of evolution has taken place on rugged landscapes (Kauffman 1993, 1995; McGhee 2007). If this is true then epistatic interactions must be the norm. Kauffman argued that the more interconnected the genes, the more conflicting constraints arise, and it is these conflicting constraints that lead to the rugged nature of the landscapes. Kauffman showed this concept using
a Boolean model of gene regulation. Whether this is true for a continuous model is an outstanding question.

Neutral Networks:
The importance of neutrality in evolution was first highlighted by Kimura, stating that the majority of genotypic change in evolution is selectively neutral (Kimura, 1983). This idea prompted a lot of work into the extent of neutrality in evolution and led to the further development of the neutral network concept that had already been put forward by Smith in analysis of protein space (Smith, 1970). The term ‘Neutral network’ (NN) was introduced to represent a sub-sample of genotype space that maps to a single phenotype, whereby any one genotype can be reached from any other genotype of the neutral network by a sequence of single mutational events (Smith 1970). Since any change in genotype within this NN results in the same phenotype, it therefore creates a many-to-one genotype-phenotype mapping.

Intuitively neutral networks were bound to exist since there is a large degeneracy in genotype-phenotype maps (there are many more genotypes than phenotypes), which coupled with a highly connected genotype space would lead to such extended neutral networks (Van Nimwegen et al., 1999). The existence of real neutral networks was indeed discovered for RNA secondary structure and protein structure (Fontana et al., 1993; Gruner et al., 1996; Schuster et al., 1994; Babajide et al., 1997).

Neutral networks have now been extensively studied for mapping the relationship between RNA sequence and structure. In this case the genotype is the RNA sequence and the phenotype is the RNA secondary structure. It is the only fully tractable example of genotype-phenotype relationships that is available for exploration, because it is possible to reliably predict the secondary structure in silico given the sequence. Efficient algorithms exist to predict secondary structure and most commonly make use of free-energy minimization (Zuker and Sankoff 1984; Hofacker et al., 1994; Tacker et al., 1996; Fontana. 2002). These genotypes are highly connected within this space since each purine-pyrimidine base pairs (G-C, G-U, A-U) can mutate into each other when single point mutations per replication are taken into account (Van Nimwegen et al., 1999). This means that each RNA
sequence has potentially \(2^N\) neighbours in the neutral network where \(N\) is the number of base pairs in the RNA sequence.

It has been found that around any random RNA sequence, there exists a space of sequences that map to almost all common shapes that are found (Schuster et al., 1994). Almost all structures are within reach of a few mutations from a compatible sequence and in reasonable proximity of any non-compatible random sequence.

**Questions in pattern formation:**

I will describe some of the outstanding questions about inductive pattern formation and how other groups have attempted to tackle them. Currently unanswered questions about inductive patterning mechanisms include:

1) **What phenotypes are possible?** What are the limits on possible phenotypes? What is the shape of possible phenotype space? Is it continuous or discrete? If it is discrete, how many pattern types are there?

2) **What is a patterning mechanism?** Are they discrete or do they tend to blend into one another? How many patterning mechanisms are there? How can we identify/distinguish mechanisms from one another?

3) **To what extent do topologies exhibit multiple mechanisms?** What does this say about the relative importance of topology versus parameters?

4) **Is the least complex core topology for a mechanism the most robust?** If not what design features confer robustness to the topology? What is the relationship between topological complexity and robustness?

5) **What happens when a gene network topology contains the motifs for two different patterning mechanisms for generating the same gene expression pattern?** Are they combined into some hybrid mechanism or do both mechanisms stay separate with different regions of parameter space?
performing either function? If the latter is true then how is it possible for evolution to move between these two mechanisms?

6) Do all of the topologies generating any one pattern form a neutral network (See later)? If so what is the shape of this neutral network? What does this shape say about the routes available to evolution?

7) How can evolution change one pattern into another? Is there only one route or are there multiple? If there are multiple routes are there these routes equally likely to be traversed?

One study addressed several of these questions by randomly generating theoretical gene networks and simulated their patterning ability in a one dimensional row of cells using a coarse model of gene regulation (Salazar-Ciudad et al., 2000). This group found that networks that produce complex patterns were usually the result of combinations of networks that were able to produce simpler patterns. If a network that can achieve a certain patterning function is found by a random search of network space, then it may not be that all of the connections within the network are necessary to produce that function. In other words there may be some minimal network that is a subset of the large network that actually produces the pattern. The extra connections do not contribute to the function. This group investigated this property and found that the pattern of many networks can be produced by a minimal part of it. Many large networks that generate the same pattern may have a minimal network in common. In order to find the minimal networks this group used an algorithm that removed interactions from the network, simulated them and asked whether it could still achieve the pattern. This group was also able to use these techniques to demonstrate categories of ‘core topologies’ able to achieve different types of patterns.

Other conclusions that they could come to included the following:

- Pattern formation is a generic property of gene networks.
- Complex patterns are due to the combination of minimal networks generating simpler patterns.
Networks responsible for patterns show structural similarity and are organized in some special sets. The patterns generated by these sets also share many characteristics.

We have also seen how attempts have been made to address questions 6 and 7 by Ciliberti (et al., 2007). This work however used a non-spatial Boolean model of gene regulation. This group found that only when paths through genotype space connected many different networks with identical phenotypes are both robustness and the ability to innovate achieved. This situation is represented by the type of metagraph (network of gene networks) shown in the central panel of figure 5.1 of chapter 5. Evolving networks can reach different regions in genotype space, allowing the generation of new diverse phenotypes. They suggested that a metagraph of simulated networks had this type of structure, an intermediate to high robustness with high innovation.

The project:

Concepts to address these questions:

We wanted to address all of the questions described earlier for inductive patterning mechanisms. To do this we employed the concept of a complete atlas of topologies which can be used as a framework to help understand both patterning mechanisms and evolution of patterning mechanisms.

Concept of a 'complete topology atlas':

Questions such as how many mechanisms are there can be addressed by sampling gene network topologies. However, if we want to say something about how these mechanisms relate to one another and how it is possible to evolve between them, then it is desirable to have a map of the entire space of possible topologies. By generating and simulating all possible topologies up to a given complexity, a complete genotype-phenotype map can be generated. Something like an adaptive landscape can then be built for topologies able to generate specific phenotypes. However the result of the function of genotype in this case is not fitness but topological complexity; that is how many gene-gene
interactions there are in the topology. Such a landscape we term a 'complexity landscape'. A complexity landscape can be used to look for the least complex topologies able to generate a specific phenotype and explore the mechanism by which they are functioning (See Chapter 6).

The concept of the complete atlas of topologies and how it is built will be described in greater detail in the following chapter, however it is important to re-emphasize here that the complete topology atlas does not just contain information about the topologies themselves, but also their relatedness to each other; that is there is a comparative metric. This feature is extremely important since it gives information on the potential evolutionary routes from any one topology to any other topology.

The complete topology atlas can help to answer those outstanding questions described earlier:

1) As we have simulated all topologies up to a given size then we will have all phenotypes that can possibly be generated within the constrictions of the model. Having all phenotypes allows the shape of phenotype space to be explored and the number of unique types can be counted if indeed there are unique types.

2) By using a complexity landscape described above then we predict that mechanisms responsible for generating a specific gene expression pattern should be represented by core topologies with the least number of interactions (See chapter 6). If it turned out that there are multiple core topologies for producing any one pattern then we can ask whether there are continuities in genotype space between those core topologies. If there are then this suggests that mechanisms are not distinct whilst if there are not then mechanisms must be distinct.

3) The goal is that the number of different mechanisms that any one topology produces can be easily analysed. If this is done for all topologies then this should give a better idea as to how much mechanism depends on parameters and how much upon topology.

4) The robustness of each topology for generating any one pattern can be analysed in a number of ways. By comparing this value to topological complexity we can gain a greater insight into the relationship between the two.
5) If we can identify mechanisms as core topologies of a complexity landscape, then by tracing the routes between two core topologies we may well find "hybrid" topologies that contain both mechanisms. What happens in these topologies? Do they have two separate parameter domains for working by each mechanism? Or is there some way whereby both mechanisms can occur with the same set of parameters?

6) By collecting topologies capable of generating any one gene expression pattern we can ask whether they form a neutral network. If they do we can explore the shape of these neutral networks and the consequences for evolution.

7) By looking at the mutational routes between Neutral networks of gene networks responsible for generating different types of pattern, we can gain insight into pattern innovation. This question can then be asked in greater detail by analysing where on each of these neutral networks are the transition points? Are they spread out or localized to specific regions?

Concept of modularity:
The size of the gene network topologies that we can explore using the complete atlas of topologies approach is limited because the number of gene network topologies increases exponentially with gene number. The hope is however that complete atlas’s of topologies can be explored for ‘module’ sized networks. As we described earlier, patterning networks seem to function as modules that can be considered in isolation. Those modules seem to involve several genes rather than huge networks or interacting components (von dassow et al., 2000; Meir et al., 2002). If this type of organization is common for many patterning mechanisms then we may be able to create complete topology atlases for networks the same kind of size as those that function in real biology. Thus the power of the complete topology atlas approach could be used to study real patterning mechanisms.

Thesis structure:
The following chapters will describe our approach for using the complete topology atlas approach to addressing these types of questions.
• Chapter 2 will describe the complete topology atlas for 3 genes in more detail and how it was generated.

• Chapter 3 will describe the gene regulation model chosen and how it was implemented for simulating the topologies.

• Chapter 4 will describe approaches that we have taken to try and explore the shape of the phenotype space (set of gene expression patterns) resulting from the simulation of the gene network topologies from the complete topology atlas.

• Chapter 5 will then explore the evolution of pattern formation by generating neutral networks of gene networks capable of generating different types of gene expression pattern and exploring how evolution can move within and between them.

• Chapter 6 will address how mechanism can be explored by generating complexity landscapes.

• Chapter 7 will discuss the broad conclusions of this work including the success of the concepts that we were using to address the questions described earlier. Answers to those questions will also be given.

• Chapter 8 describes the hardware used in this project.
2) Generating the complete topology atlas.

This chapter concerns itself with the generation of the complete topology atlas, consisting of the complete set of topologies and their relationship to each other. The exact structure of these genotype spaces depends on the question that one wants to address. We are using a simple structural relationship between topologies as the goal of the project is to address mechanism. In this chapter I will first describe the concepts involved and then explain the algorithmic methods used, and finally describe the results.
Introduction

The complete topology atlas concept:

Topologies:
As described in the introduction, gene networks can be represented as graphs whereby the genes are represented by the vertices of the graph and the interactions between those genes its edges (figure 1.2 in chapter 1). These gene network topologies are known as directed graphs since there is a directionality to their edges that is one gene acts to either activate or inhibit another gene (Kumar and Bentley 2003). Any pair of genes can have two interactions between them, one from each gene to the other.

These gene networks can be represented by a two dimensional matrix of dimension sizes equal to the number of genes within the network. Figure 2.1 illustrates an example gene network and its corresponding matrix. The gene network has three genes and thus the matrix has the dimensions three by three. The horizontal dimension refers to the regulating gene and the vertical the regulated gene. In this matrix, a 1 represents activation, a 2 represents a repression and 0 represents no interaction. In the example given in figure 2.1 gene network gene A positively regulates gene B and therefore in the A column and B row there is a one. Gene C represses gene B and thus in column C, row B a 2 is found. Self regulation of a gene can be seen by a one or a two appearing in the diagonal (here: AA,BB,CC).

Relationship between topologies:
Using these matrices, the concept of a Hamming distance can be described. Take the two matrices shown in figure 2.2. A Hamming distance is a measure of the minimal distance between any two matrices. It can be measured by comparing each of the corresponding elements of the matrix. The Hamming distance is the number of elements that have a different interaction type (0, 1 or 2) between the two matrices. Hence for a 3x3 matrix as shown here, the minimum Hamming distance is 0 and the maximum is 9.
Figure 2.1: Gene networks and matrices. (Left) A gene network with 3 genes. Each letter is a gene. Arrows represent activation and 'dead-end' lines repression. (Right) The corresponding matrix that describes this gene network. The column determines the regulating gene and the row determines the regulated gene. A 1 means an activation, a 2 means an inhibition and a 0 means no interaction.
Figure 2.2: The concept of a Hamming distance. A single Hamming distance represents a single mutational change. Here the auto-activation of B is gained/lost, indicated as a red digit in the corresponding matrices.
One Hamming distance between matrices of this size represents a single step in theoretical space of possible topologies. This concept is important since it allows the evolution of topologies to be seen as a walk between topologies each one Hamming distance away from one another. By considering a single evolutionary step as a move in topology space of one Hamming distance then a type of hypercube can be generated.

*Hypercubes of gene network matrices:*
A hypercube is an n-dimensional analogue of the square. A hypercube can be used to describe a set of strings and the links between those strings that are one Hamming distance away from each other. Those strings are normally binary in nature however (0 or 1). The number of dimensions of the hypercube depends on the length of this string. The smallest hypercube of 4 dimensions for example would describe all of the combinations of binary string of length 4 (figure 2.3).

Matrices can be represented by strings and thus the matrix of a gene network can also be thought of as a string. The strings of these gene networks are not quite the same as the strings that the hypercube describes because a standard hypercube links binary strings together (containing values 0 and 1) whilst these gene network strings contain 0, 1 or 2. Therefore there are more dimensions in which one can move in the hypercube describing gene network strings as compared to a hypercube describing binary strings. In other words, a 0 can be changed to a 1 or 2 in the gene network string, but only to a 1 in the binary string.

In theory a single step could involve a change in the matrix in one of the elements from a 1 to a 2 or vice versa. This would correspond to a sign change from activation to repression in the gene network, or vice versa. We chose not to consider such changes because our goal is "mechanism" and in terms of gene network function sign changes would be expected to have a more dramatic effect.

Taking the idea that a Hamming distance of one represents a single step on a hypercube, then we can build a hypercube of gene network topologies. For a 2x2 matrix where there are 4 elements in the corresponding string, the hypercube would contain 81 \((3^4)\) corners, whilst for a 3x3 matrix where there are 9 elements in the corresponding string, the hypercube would contain 19,683 \((3^9)\) corners. This
Figure 2.3: A four dimensional hypercube describing all of the binary combinations of 4 elements and their relationship to each other. Each binary combination is shown as a 4 digit string of 0's and 1's. A line means that the two strings are different to each at a single digit. Adapted from: http://www.benpadiah.com/basic_intro.html
The hypercube can be made simpler however when an important aspect of gene networks is taken into account that will now be described.

The ‘Complete Topology Atlas’, a stripped down hypercube:
Graphs can be defined as one of two types; labelled and unlabelled (Diestel, 1997). Labelled graphs involve connected nodes (vertices) that have names. The nodes of an unlabelled graph on the other hand do not have names (figure 2.4). For each unlabelled graph, there are multiple labelled equivalents which are called isometric equivalents (figure 2.4). The set of isometric equivalents that correspond to the same unlabelled graph can be called an isometric group. In this project we are interested in mechanism which relates entirely to topology, it does not matter which gene takes which role, and therefore we are only interested in unlabelled gene network graphs. The hypercube of gene networks described above contains all of the isometric equivalents for each of the unlabelled gene network topologies. Therefore this hypercube can be made much simpler by including only one gene network topology from each of the isometric groups. By doing so we generate a ‘slimmed down’ version of the hypercube described above. We termed such a ‘slimmed down’ hypercube of gene network topologies an atlas of gene networks. Because each isometric group has several labelled topologies (the exact number depending on the symmetry of the topology), the removal of all but one of the topologies from each isometric group greatly reduces the number of the corners in the hypercube. The generation of these topology atlas’s for small gene networks was the goal of this chapter.

The relationship between the concept of a topology atlas and genotype space:
A complete genotype does not simply consist of a topology, but a topology and a specific set of parameters for that topology. What then is the relationship between genotype space and the atlas’s of topologies described here? The generation of an atlas of gene network topologies acts as a convenient way to split genotype space up such that we have a complete discretized map of the space (figure 2.5 left). The alternative is to vary interaction parameters randomly on a number of genes equal to that of the described networks (Reinitz and Sharp 1995) which can be called the
An unlabelled topology

Labelled topologies that are all consistent with the above unlabelled topology.

Figure 2.4: Unlabelled versus labeled topologies. Activation is represented by arrows and repression by 'dead-end' lines. (Above) An unlabelled topology. Genes are represented by dots. (Below) Labeled isometric equivalents of the above unlabeled graph. Genes are represented by letters and are thus labeled.
Figure 2.5: Comparison between the complete topology atlas and the complete parameter set approaches to representing genotype space. (Left) The complete topology atlas approach. The two topologies contained in the open circles are linked by a line that represents a Hamming distance of one. (Right) There are interactions from every gene to every other gene. The strengths of these interactions can vary and can be activatory or inhibitory.
complete parameter set approach (figure 2.5 right). The concept of an atlas has three important features as a representation of genotype space.

Firstly, the discretization and the linkage between those discreet elements provide a structure that we can relate to. It is then easy to envisage how one can move through such a space as multiple steps. The complete parameter set approach results in a set of topologies that will have some degree of interaction from every single gene to every other gene. Thus envisaging how one can move from any one of these genotypes to another genotype in discreet steps is more difficult since there is no concept like the ‘Hamming distance’ employed.

Secondly focusing on topology allows a clearer understanding of mechanism. Many of the genotypes generated in the complete parameter set approach would contain interactions of negligible strength. These interactions obscure a clear understanding of the mechanism of a genotype.

Thirdly the complete topology atlas approach allows an equal sampling of topologies. The complete parameter set approach would result in an oversampling of symmetrical topologies. This is because randomly generated parameter sets are more likely to result in a set of interactions that looks like a symmetrical topology than a non-symmetrical one since there are more ways to ‘build’ a symmetrical topology. Restricting ourselves to varying the interaction strengths of topologies thus results in a less biased sampling of topology space.
Methods:

The generation of the topology atlas for networks of 2, 3 and 4 genes is the goal of this chapter. The main potential problem that we expected to encounter was making sure that only one topology from each of the isometric groups was included in each atlas. This was likely to be a problem since this would require topologies to be compared to each other in every permutation which could mean the algorithms used to generate the larger topology atlas’s could have had an unfeasibly long run time.

Generating the topology atlas:

Generation of a complete set of unique network topologies of up to four genes.

Initially a system that focused on computational simplicity was devised to generate all possible topologies that are unique (version 1). A summary of this strategy is shown in figure 2.6.

A base three counting algorithm was generated that counts through all combinations of matrix containing 0’s, 1’s and 2’s (figure 2.7). Here 1 represents activation, 2, repression and 0, no interaction. The first matrix generated is placed in a group of matrices that represent unique unlabelled networks. The second matrix generated is then compared to first matrix in all permutations (Figure 2.8). If it is different in every permutation, then it is also placed in the unique unlabelled network group. The third matrix generated is then compared to both the matrices in the unique unlabelled network group in every permutation and appended to this group if it is different. This process continues for all combinations of matrix. The result is a group of labelled matrices each representing a unique unlabelled network.

Using this method, the number of comparisons needed for each newly generated labelled topology increases linearly with the number of unique unlabelled topologies already found. The affect of this is that the speed of the program decreases linearly with respect to the number of unique unlabelled topologies found (figure 2.9).

Therefore we had to find a more sophisticated approach to reduce the number of topology comparisons necessary, so we sought graph based methods of shortcutting
Figure 2.6: Summary of the basic strategy for generating a unique set of topologies. Processes are shown in blue squares. The order of those processes is indicated by the arrows.
Figure 2.7: The base three counting algorithm. Mapping the base-3 number into the gene regulation matrix. The order with which the base 3 numbers are generated are shown on the left (from top to bottom). These numbers are then fed into a regulation matrix from left to right and top to bottom as illustrated by the colored arrows.
Permutations:

- ABC = original
- ACB = example
- BAC
- BCA
- CAB
- CBA

**Figure 2.8:** The 6 permutations of one network. (Top right) The six possible permutations of three elements. (Top left) A network and an example permutation. Genes are represented by letters. Activations are represented by an arrow and inhibitions by a 'dead-end' line. (Bottom) The matrices for all 6 permutations are also shown. The black double ended arrows show the row and column switches that occurred in the example.
Figure 2.9: A scatter graph showing the progressive slowing down of the algorithm generating topologies using the simple method. The number of topologies that have been generated is plotted on the x-axis and the time that the algorithm has been running for is plotted on the y-axis. A power law trend line is shown in black.
the problem. Given the need to compare topology A with topology B, if we could find a quick feature of topology A that would immediately eliminate it as being isometric with B, without having to perform the full permutation analysis, this would be potentially much faster. It is clear that two topologies cannot be isometric if they contain different numbers of positive and negative regulatory interactions. In this way, we defined the notion of "sets" – each one comprising all of the topologies with the same number of 1's and 2's so that topologies only need be compared with others from the same set. This dramatically increases the efficiency of the program (version 2).

The success of this approach prompted us to explore whether a more fine-grained set of criteria could subdivide our collection of topologies into even smaller sets. The sets method works because single positive or negative interactions are an irreducible element of a topology, and their total number cannot change during the permutation analysis. Upon examining the problem it became clear that mutual interactions (where 2 genes directly regulate each other) constitute similar irreducible elements. In this case there exist 5 types of mutual regulation. Additionally, it is clear that direct auto-regulation can be distinguished from all other interactions, and that it also remains with the same unlabelled graph node upon permutation analysis. We therefore use this collection of 7 criteria for defining more fine-grained topology sets (version 3)(figure 2.10). The number of sets that there are with different numbers of genes is shown for the two versions of the program in table 2.11.

Parallelization of the topology generation:

A second important benefit of the sets methods of generating topologies is that it is very easy to parallelize. Each set can be implemented as a separate job as topologies only need to be compared to those within the same set. We implemented a farmer worker system on our in house parallel computer cluster.

The purpose of the farmer program is to generate jobs to be carried out by the worker program. Multiple worker programs are started, one for each processor available. Scheduling is the process of checking the state of the processors; As soon as a worker has finished a job, another worker is started on the same processor with a new job until all jobs are finished. Most farmer programs perform this 'scheduling'
Figure 2.10: Set criteria. Blue dots represent genes. Activation is represented by an arrow and repression by a 'dead-end' line. The top 5 are the mutual regulation and the bottom two, are auto-regulation. The number of each of the criteria that a topology contains defines its set.
<table>
<thead>
<tr>
<th></th>
<th>Version 2</th>
<th>Version 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 genes</td>
<td>15</td>
<td>36</td>
</tr>
<tr>
<td>3 genes</td>
<td>55</td>
<td>560</td>
</tr>
<tr>
<td>4 genes</td>
<td>153</td>
<td>6930</td>
</tr>
</tbody>
</table>

**Table 2.11:** The number of sets for version 2 and version 3 of the topology generating algorithms.
themselves. However, on our in house parallel cluster the control of this ‘scheduling’ is performed by the Sun Grid Engine. Jobs are submitted to a queue, and as soon as a processor becomes available the worker is started on that processor with the next job in the queue.

Our farmer program generates all combinations of the 7 criteria described above for version 3 of the sets system. Each combination is saved to a job file. A shell script then submits the jobs to the Sun Grid Engine queue. Once all jobs have been executed, a final script then collects the results from each of the jobs by simply appending the topology sets together.

Generating the Topology lists with a pre-pattern.

Our goal in this project is to explore the ability of these gene networks to embellish a pre-pattern. We chose to simulate the topologies with a pre-pattern input (See chapter 3). This pre-pattern was implemented by the use of a ‘virtual’ gene that activates one of the genes of each topology but is not affected by the topology itself. The strength of the regulation from the virtual gene can be non-uniform over the spatial dimensions of the lattice of cells that we choose to simulate. In this way, any form of pre-pattern can be implemented, though we chose a specific form (See chapter 3).

The pre-pattern input is represented computationally by an extension to the matrix representing the topology. For example take the topology shown in figure 2.12. The topology is described by the matrix to its right. There is a 1 in the row of the gene that the pre-pattern affects.

The questions arises which gene of the topology should this ‘virtual gene’ activate? In order to test all possibilities we decided to simulate the topology multiple times in each case the ‘virtual’ pre-pattern gene inputting into a different gene of the topology. If the number of genes in a topology is G, then the number of different topologies with all combination of pre-pattern is also G. However, there is an exception to this rule when topologies have some degree of symmetry. For example take the topology shown in figure 2.13 (top). The pre-pattern affecting both A and B creates the same topology and thus both of these possibilities do not need to be tested.
Figure 2.12: How the pre-pattern is encoded in the matrix. (Left) Genes are represented by letters. Activation is represented by an arrow and repression is represented by a 'dead-end' line. The pre-pattern inputs into one gene, in this case gene A. (Right) An extra column is added to the topology matrix and a one placed in the row corresponding to the gene which is being activated by the pre-pattern.
Figure 2.13: Symmetrical networks. Genes are represented by letters. Activation is represented by an arrow and repression is represented by a ‘dead-end’ line. (Above) A topology symmetrical about the two genes A and B. (Below) A totally symmetrical topology.
Therefore an algorithm was needed that generated all possibilities of topology with pre-pattern. This algorithm works in a very similar way to the algorithm described for generating the group of unique unlabelled networks described earlier. However, the difference is that when each permutation of the topology is made, the pre-pattern is also permutated. The six possible permutations of this network are also shown in figure 2.14. The comparison to the rest of the group of unique unlabelled networks is then exactly the same as before.

*Generating the Neighbor structure.*

The second part of the complete topology atlas describes how the different unique unlabelled networks relate to each other. This is done with the creation of a neighbor file. The first column of each line of the neighbor file states the topologies. The other columns of the neighbor file state the topologies which are one Hamming distance (a neighbor) from the topology in the first column.

This file is generated by looping through all of the topologies. For each topology (A) each position of its corresponding matrix is changed to the two other possibilities. For example if a position read 1 (an activation) then it is changed to both 0 and 2. If a position reads 0 (no interaction) then it is changed to both 1 and 2. Each altered topology is then compared to every other topology (B) within the complete topology atlas. It is compared under every permutation possible (see earlier for generating topologies). If the altered topology A is seen within the atlas, then topology B is considered a neighbor of A and is written to the next available column in the line describing the original topology A’s neighbors.
Figure 2.14: (Left) An example network. Genes are represented by letters. Activation is represented by an arrow and repression is represented by a ‘dead-end’ line. (Right) The 6 permutations of the example network with a pre-pattern. The extra column designated P described the pre-patterns influence on the network.
Results and discussion:

Generating the topology atlas (generating the genotype map):
The number of topologies for each size of atlas is shown in table 2.15. The numbers rapidly increase with gene number. The atlases of 2 and 3 gene networks were generated with each of the different topology generating algorithms. They all gave the same answers indicating that the algorithms are working correctly. The fine sets version (Version 3) of the topology generating algorithms takes only half an hour to generate the complete atlas of 3 gene topologies when ran on our in house cluster. The percentage of unique unlabelled topologies compared to the number of labeled topologies is in the order of 16% for the 3 gene complete topology atlas and 4% for the 4 gene complete topology atlas. This results in a huge reduction in the number of simulations that we need to perform in order to explore the complete set of topologies that can be generated with 3 or 4 genes. This highlights the benefit of excluding labeled topologies from the analysis.

The atlases:
By including a definition of which topology is related to which, it was possible to generate a complete atlas of topologies which is a discreet representation of complete genotype space. The atlas can be thought of as a topology of topologies, or a metagraph.
An illustration of the complete two gene atlas can be seen in figure 2.16. The complete three gene atlas is shown in figure 2.17. Each of the balls is a topology and the line a link between neighboring topologies one Hamming distance away from each other. Figure 2.17 was spaced using biolayout software (see chapter 4 for description of how this software works). This figure illustrates just how interconnected the topology space is. I.e. although it is a “stripped down” hypercube, it appears to retain a similar degree of high connectivity.

Concluding remarks:
The number of networks increases exponentially with gene number if there are no constraints. This issue limits the complexity of network with which we can simulate.
Without pre-pattern

<table>
<thead>
<tr>
<th>Number of genes</th>
<th>Number of labeled topologies</th>
<th>Number of unique topologies</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>81</td>
<td>39</td>
<td>48.1%</td>
</tr>
<tr>
<td>3</td>
<td>19,683</td>
<td>3,284</td>
<td>16.7%</td>
</tr>
<tr>
<td>4</td>
<td>43,046,721</td>
<td>1,795,514</td>
<td>4.2%</td>
</tr>
</tbody>
</table>

With pre-pattern

<table>
<thead>
<tr>
<th>Number of genes</th>
<th>Number of labeled topologies</th>
<th>Number of unique topologies</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>162</td>
<td>74</td>
<td>45.8%</td>
</tr>
<tr>
<td>3</td>
<td>59,049</td>
<td>9,710</td>
<td>16.4%</td>
</tr>
<tr>
<td>4</td>
<td>172,186,884</td>
<td>7,171,598</td>
<td>4.2%</td>
</tr>
</tbody>
</table>

Table 2.15: Number of unique topologies for atlases with different numbers of genes with and without pre-pattern. The percentage is the number of unique unlabelled topologies relative to the number of labeled topologies.
Figure 2.16: The complete 2 gene topology atlas. Topologies are shown in ellipses. Topologies one Hamming distance apart have a line between them. Genes are represented by dots. Activation is represented by an arrow and inhibition by ‘dead-end’ lines. The topologies are arranged from lowest complexity (number of gene-gene interactions) at the bottom to highest complexity at the top.
Figure 2.17: Showing the similarity of the atlas of 3 genes topologies to a hypercube. Topologies are shown as spheres and the relationships between them lines. The atlas was spaced with the Biolayout software (See chapter 4 for a description of this software).
Parameter sampling further compounds this problem, since the more genes a topology has, in general the more interactions there are and thus the more variable parameters that there are.

The modularity of the whole approach is useful in case we want to explore other types of relationship between topologies. For example we may be interested in network duplication events. Therefore we would look for topologies, which are duplicates of others based on known duplication mechanisms. In this way we don’t have to generate the whole topology list again, just another neighbour file. The other tools for simulating and exploring complete topology atlases will also work in exactly the same way just using a different neighbour file.

Generating an atlas is an alternative approach to sampling. The benefit of the atlas is that we can explore the whole of genotype space meaning that we know every possible route from any topology to any other. The biggest drawback of the atlas approach is that only small networks can be explored as the number of networks increase exponentially with gene number. However as mentioned in the introduction, it is thought that many patterning gene networks are organised into small modules of several genes. Hence by using facilities like the Mare Nostrum supercomputer and with computational power increasing we may be able to explore complete topology atlases of gene networks of the size of real patterning gene networks involving several genes in the future.

Summary:

- We have described an efficient algorithm for generating complete atlas’s of unique unlabelled gene networks up to a given level of complexity.
- By excluding labeled topologies, we were able to reduce the number of topologies by approximately an order of magnitude without losing any important information.
- Numbers of topologies increase rapidly limiting the number of genes in the complete atlas feasible to simulate.
3) Simulating the topologies.

The complete topology atlas serves as the foundation for generating the phenotype map. In order to generate the phenotype map we must simulate all of the gene network topologies of the complete topology atlas in space and time. The details of these simulations have important consequences for what can be extracted from the results. In this chapter we describe the details of the simulation model that we employed.
Introduction

Simulating the gene networks:
To generate the phenotype map, we must simulate all of the gene network topologies in space and time. We wanted to choose a model that was as abstract as possible yet still captured the logic that is used in real patterning gene networks. Choosing an abstract model allows us to sample a larger proportion of parameter space and also allows a clearer understanding of how the network is working since there are less superfluous details. A type of model including only whether genes activate or inhibit one another, not how that regulation is specifically biologically implemented has been used in the past (Mjolness et al., 1991; Reinitz and Sharp 1995; Salazar-Ciudad et al., 2000; Salazar-Ciudad and Jernvall 2002). This type of model has been shown to successfully mimic gene expression dynamics that have been seen in real contexts including the stripes of even-skipped gene expression in the Drosophila syncitium (Reinitz and Sharp 1995). Therefore we chose to use a modified version of this model. Here we discuss several aspects of our model that have important implications for the principles that can be extracted from this work.

The gene regulation model:
The following stochastic differential equation (SDE) model was implemented in a one dimensional row of non-dividing 32 cells:

$$\frac{dg_{ij}}{dt} = \chi \left\{ \Phi \left[ \sum_{l=1}^{Ng} W^{il} g_{lj} \right] \right\} + D_j \nabla^2 g_{ij} - \lambda g_{ij} + \eta(t) g_{ij}$$

The $g_{ij}$ is the concentration of the $ith$ gene in the $jth$ cell. The function defining the interaction amongst genes $\Phi$ can take the form of a Michalis-Menten or sigmoid input function. The matrix $W^{il}$ is the matrix containing the strength of regulation parameters. The squared box will sum all of the inputs into the $ith$ gene in the $jth$ cell. $\chi(x)$ is the Heaviside function where $\chi(x) = x$ for $x > 0$ and $\chi(x) = 0$ otherwise.
Its purpose is to make sure the regulation term can only take positive values. Diffusion occurs between neighbouring cells at the rate $D_j$ and is thus different for different genes. The decay parameter $\lambda$ is however the same for all genes. The noise term $\eta(t)$ generates a random number within a given range.

The only variable parameters in this model then are the strength of regulation and the rate of diffusion. Using such a simple set of variable parameters allows a denser sampling of parameter space, improving the sensitivity of the technique allowing the identification of less robust mechanisms. We simulated each topology with multiple different parameter sets.

I will now briefly discuss background about the features of this model and justify why these features took their particular forms.

**Features included in this model:**

_Regulatory saturation:_

The simulation includes an input function for each gene such that the production rates of gene products saturates at high activation. The basis for such saturation in real biological systems may vary but in all contexts it exists. For example the availability of tRNA molecules may become the constraining factor for the rate of protein translation. Rather than limiting our results to one particular saturation function, we chose to use two different functions; the Michalis-Menten and the Sigmoid function (figure 3.1).

_Signalling:_

Several different intercellular signalling mechanisms have been discovered. One such mechanism is where a secreted molecule acts as a morphogen, giving nearby cells information based on its local concentration (Wolpert 1969). In this way a cell can directly signal to another cell several cell diameters away. Several morphogens have now been discovered. Examples include Bicoid and Decapentaplegic in _Drosophila_, FGF’s, EGF, Wnt, Hedgehog and TGF-beta families and the vitamin A metabolite retinoic acid (Lander and references there in, 2007). In the _Drosophila_ syncitium patterning occurs in both the anterior-posterior and dorsal-ventral axis.
Figure 3.1: Functions showing how gene regulation input is related to output gene expression. (Top) The Michalis-Menten input function. (Bottom) The Sigmoid input function.
before cell membranes have formed (Wolpert 2002). Diffusion in this system is thus much faster; a typical transcription factor can diffuse approximately 1-100μm/s (Isalan et al., 2005; Istrail et al., 2007).

Diffusion has been disputed as the main method by which cells signal to each other by some authors except in cases where there is no cell membrane (Kerszberg and Wolpert, 1998, 2007; Pfeiffer and Vincent, 1999). This criticism arises because the extracellular matrix (ECM) likely presents a variety of binding sites for morphogens and hence can "trap" these molecules (Zhu and Scott, 2004). In the Drosophila embryo, cell-surface glypicans (proteoglycans), such as dally and dally-like have been proposed to shape the gradients of Decapentaplegic and Wingless (Baeg et al., 2004; Kirkpatrick et al., 2004). The ECM also has a complex shape and so effective diffusion may increase as much as 5 fold resulting in local variations in morphogen gradient (Lander et al., 2002). A second way for cells to signal to one another is similar to diffusion, but involves the morphogen being relayed from cell to cell (Han, 2005).

Another intercellular signalling method is directly through plasma membrane proteins such as Notch and delta that are involved in lateral inhibition (See introduction chapter). The main difference between this type of signalling and signalling via morphogens is that direct cell signalling acts locally between cells that are adjacent whilst morphogen signalling can occur between cells many cell diameters away.

Cell are also known to be able to directly mix their cytosolic complement of small molecules via GAP junctions (Bennett et al., 1991), allowing molecules of up to 1000 daltons to pass (Alberts et al., 1994). These gap junctions are increasingly becoming recognised as key regulators of development, particularly in establishing right-left asymmetry in the morphogenesis of the heart, brain and viscera of many animal body plans (Oviedo and Levin 2007).

Cytonemes are actin-based filopodial extensions from cells. In the Drosophila wing imaginal disc they have been proposed to play a role in the distribution of Decapentaplegic and Wingless morphogens in the patterning of the anterior-posterior and dorsal-ventral axis (Ramirez-Weber and Kornberg, 1999; Hsiung et al., 2005).
The question arises as to how do these different signalling regimes affect how we should model patterning systems? Importantly, many of the details of how cells signal to each other are not important for the way we model signalling, the many of the results would be the same no matter which model was used. For example Salazar-Ciudad (et al., 2000) has performed simulations of sampled gene networks in space and time and shown that some results were almost the same irrespective of whether a model was used that is based on diffusion or direct signalling between cells. For example, despite the fact that the patterns exhibited by the diffusion and the direct cell signalling models were different, they showed the same four basic types of network module were responsible.

Another complication of the direct cell signalling models as opposed to diffusion based models is that the wiring of networks can be much more complex (figure 3.2). In these direct contact models, a gene in the signalling cell can in theory affect any of the genes in the adjacent receiving cells independently. In this way there is an increase in parameters with respect to the diffusion model. A gene only has one parameter for diffusion but has as many parameters as there are genes in a direct contact model, each one describing the effect of this gene on each of the genes in the adjacent cells.

In the current work we have used diffusion as a proxy for signalling. This was chosen mainly for pragmatic purposes as it allows signalling to be modelled without having to include a great number of new parameters as would occur for direct cell signalling.

**Noise:**

How cells function and process information when the underlying molecular events are stochastic is a central question in biology. Despite the stochastic nature of regulatory circuits within cells, most cellular events are ordered, precisely regulated and repeatable. Development in *Caenorhabditis Elegans* is so regular that the lineage of every cell can be traced (Sulston and Horvitz 1977; Sulston et al., 1983). The similarity of identical twins is also testament to the reproducibility of a given genetic program in animals with variant lineages.
Figure 3.2: Direct contact model versus the diffusion model. Cells are represented by light blue rounded squares. Genes are represented by blue dots with letters inside. Activation is represented by arrows and inhibition by 'dead-end' lines. The variable parameters are shown in the right hand grids. (Top) Direct contact model. (Bottom) Diffusion model.
Stochastic fluctuations in molecular populations have consequences for gene regulation (McAdams and Arkin 1999). Noise sources have been identified and noise propagation has been studied. Noise can be split into two different classes, intrinsic and extrinsic. Intrinsic (internal) noise describes the fluctuations in populations due to the stochastic nature of gene expression and thus is inherently related to the wiring of the gene regulatory network (Thattai and van Oudenaarden 2001). Intrinsic noise is increased in biological systems involving small numbers of molecules since variation increases when samples are taken from small populations as compared to larger populations (Thattai and van Oudenaarden 2001). Extrinsic (external) noise is generally defined as fluctuations and variability that arises in a system due to disturbances originating from its environment (Hasty et al., 2000). Examples include fluctuations in transcriptional regulatory signals, varying efficiencies of the transcriptional and translational machinery as well as factors such as variation in gene copy number and population dynamics. The distinction between intrinsic and extrinsic noise is not black and white and thus depends largely on how the boundaries of the system are defined.

Evolution has taken place on a background of noise. Therefore it would be expected that biological mechanisms should exist to deal with this noise or that perform useful functions using this noise. Negative feedback has been shown to suppress noise (Becskei and Serrano, 2000) in engineered genetic circuits in *Escherichia coli*, and demonstrated explicitly in a theoretical model of gene regulation (Thattai and van Oudenaarden 2001). Some organisms have even been shown to exploit noise to introduce diversity into a population, such lamba phage at the lysis-lysogeny bifurcation (Arkin et al., 1998) or bacteria during DNA inversion (van de Putte and Goosen 1992).

Although noise is ubiquitous, most models do not have noise in their system, though some use random initial conditions as a proxy for noise. However, noise is apparent through the entire time that a network is functioning and may affect later network behaviour that is not picked up by these methods. We thus implemented noise in our system by choosing to use a stochastic differential equation. This is essentially an ordinary differential equation plus a noise term (See Introduction chapter). We implemented a simple noise term that increases or decreases the concentration of
each gene product in every cell randomly up or down a small amount at each iteration of the simulation. This form of noise was chosen since it was simple to implement and captures the ever present effect of noise throughout real network function. We did not expect that the exact form of the noise would change the results of the simulations. By simulating each network with each set of parameters multiple times then we could measure which dynamics are robust and which are bizarre, accidental fragile ones.

Initial conditions:
Initial conditions can have a substantial effect on the subsequent evolution of the patterning system. Two types of initial conditions have been used for modelling gene networks capable of pattern formation:

1) Typical initial conditions: This is where one or more of the model players are given initial values and the simulation started. The most typical scenario is that randomly selected players are given some small concentration. Examples of models using these types of initial condition include the work by Salazar-Ciudad on theoretical gene networks (et al., 2000; et al., 2001) and on gene networks involved in the patterning of mammalian teeth (Salazar-Ciudad and Jernvall, 2002).

2) External input: This is where a separate molecular component is given an initial value. This component feeds into the network but is not affected by the network. Classic examples of models using these types of initial condition include:(a) the network controlling segment polarity gene expression in Drosophila where there is a direct pre-pattern input from the upstream pair-rule genes, and (b) the neurogenic network also in Drosophila where the network was modelled with 3 different pre-pattern input contexts (von dassow et al., 2000; Meir et al., 2002; Ingolia 2004). Furthermore the work on modelling the segmentation cascade in Drosophila used the gene expression patterns of the Maternal coordinating genes to initiate their simulations (Reinitz and Sharp 1995; Reviewed in Jaeger and Reinitiz 2006)
We decided to start the simulations with an external pre-pattern since we expect that most real gene networks function in the presence of some form of pre-pattern. The exact form of this pre-pattern will be discussed in the section below on the choice of physical domain.

**Features not included in this model:**

**Bimodal interactions:**
A component that can act as both an activator and an inhibitor on the same downstream node is not modeled explicitly. However this is modeled in an indirect way. One gene can activate another directly and then inhibit it indirectly or vice-versa. This is actually what happens in real biological systems since some type of cofactor is needed to turn the activator into an inhibitor. A good example is the mechanism utilized during canonical Wnt signaling (Reviewed in Bejsovec 2005). In the absence of Wnt signaling, cytosolic β-catenin is targeted for ubiquitination and degradation and target genes are repressed. Binding of Wnt ligand to the Frizzled/LRP-5/6 receptor complex leads to stabilization of hypophosphorylated β-catenin, which interacts with TCF/LEF proteins in the nucleus to activate transcription.

**Delays:**
Delays are known to have potentially dramatic effects on mathematical models (Kerszberg 2004). In fact, delays may sometimes be introduced instead of molecules; for example a system with \( N \) molecular species plus a delay behaving in a way similar to one with \( N + 1 \) species (Kerszberg 2004). Delays are known to be useful for performing certain functions in engineering such as the coherent feed-forward loop (Mangan and Alon, 2003). Oscillations in circuits comprising very few species have been shown using models including delays (Lewis 2003, Monk 2003). Delays are also apparent in our model, though they are not directly implemented but result from the way the simulations occur. Any gene that affects another indirectly will be delayed in relation to another gene being affected directly. This is because all gene expression values are updated sequentially and thus it takes an extra iteration of the algorithm for the effect to be passed on to the next gene. Furthermore if the
regulation parameters are low, the effect of an interaction can be slow to become apparent.

**Choice of physical domain:**

*Modelling different number of dimensions.*

We chose to simulate a one-dimensional row of 32 cells. Modeling only one dimension rather than multiple dimensions has the following computational benefits:

- A much smaller computational time is needed for the simulation and that the analysis is much simpler.
- A choice of lattice topology or neighborhood is not needed but is for higher dimensions.

Real multi-cellular organisms do not contain pattern in a single dimension as can be seen from looking at any animal for example. However pattern formation often occurs primarily in one axis. Examples include the segmentation of arthropods and mammals along the anterior-posterior body axis (Reviewed in Peel *et al.*, 2005; Reviewed in Dequéant and Pourquié, 2008) and the patterning of the vertebrate neural tube along the rostrocaudal axis (Ericson *et al.*, 1995)

Furthermore, this work is a theoretical exploration of gene networks capable of pattern formation. In such an exploration the ability to accurately capture the logic of gene regulation is more important than the actual context in which the simulations take place. A gene network capable of pattern formation in one dimension will obviously be capable of pattern formation in more dimensions. Hence the general results of a study involving one dimension should be applicable to multiple dimensions.

**Boundaries:**

The existence of boundaries in biological patterning systems has an important effect on how we chose to model these systems *in silico*. Obviously we cannot simulate an infinite lattice of cells and this would not be very biologically relevant either. Hence we needed to define some condition for cells at the end of the lattice that is biologically reasonable. Different types of boundary condition include periodic, zero-
flux and fixed-value boundary conditions. The different types of boundary conditions are illustrated in figure 3.3 for a one dimensional lattice of cells. Periodic boundary conditions are the closest to simulating an infinite lattice. All that is needed to implement these conditions is to define the topology of the lattice. The topology for a one dimensional simulation is a ring. The two dimensional version of this lattice has the topology of a torus. Biological patterning domains are not infinite domains and thus this type of boundary condition is not appropriate.

Fixed value boundary conditions are obtained by simply assigning a fixed value to each of the cells in the boundary. A fixed value boundary condition would be like the boundary acting like a continuous source of reactants into the system. Although this may be the case for some boundary produced morphogens (such as Decapentaplegic or Wingless) for other morphogens this may not be true (Dahmann and Basler 1999). Reflective or zero-flux boundary conditions are best suited to simulating systems that also have a boundary. Zero-flux boundary conditions do not allow any diffusion in or out of the system, therefore modelling as an isolated system. Zero-flux boundary conditions are appropriate for modelling biological patterning field for this reason. As has been just described patterning fields are often boundary restricted zones that are patterned relatively independently. Zero-flux boundary conditions were used in our model of pattern forming gene networks.

*Pre-pattern:*

The pre-pattern took the form of a gradient over the one dimensional field of cells, a common feature in patterning networks (Driever and Nusslein-Volhard 1988; Jaeger and Reinitz 2006). Many field of cells are believed to be patterned by positional information provided by gradients of morphogens. This pre-pattern does not change throughout the simulation and activates only one of the genes of the network to the same extent throughout the simulation.

**Features required to define the fitness function: (figure 3.4)**

Not all outputs of gene networks are potentially useful to an organism undergoing development. Here we discuss two features of the simulation output that define
Periodic Zero-flux Fixed boundary conditions

No implementation needed except topology of cell communication given explicitly

At the end of each iteration of the simulation, the two boundary cells take the same state at the adjacent cells resulting in no diffusion between the two (or zero-flux)

The two boundary cells maintain the same concentration of gene products at all times and thus act as a continuous source of gene product into the system.

Figure 3.3: Different types of boundary condition. Cells are illustrated by squares. (Left) Periodic. (Middle) Zero-flux. (Right) Fixed boundary conditions
usefulness and hence fitness; pattern and noise robustness. We also describe one other important feature of the network simulation; equilibration.

**Pattern:**
The pattern of gene activity within a cell determines its type (Kauffman 1993). Thus in order to make cells of a field different from one another, a patterning mechanism must generate a gene expression pattern such that there are different gene activities in different cells of the field. Thus expression patterns that are essentially a flat line over the field of cells are not of interest to us. A flat line would correspond to a gene being expressed at the same level by every cell in the field. Patterned outputs can be selected by how different to a flat line they are.

**Noise robustness:**
As described earlier we have included a noise term in our simulations. If the output of a gene network is not robust with respect to that noise then it is not a very useful network. Networks producing noise robust dynamics can be identified by simulating the network with a specific set of parameters a number of times. If the network with this set of parameters is robust to noise, then the resulting dynamic should be more or less the same.

**Equilibration:**
Some patterns do exist in real organisms that are not stable with respect to time for example the shell ornamentation of molluscs (Meinhardt and Klinger, 1987). Moreover there is no reason why any network dynamic should have to reach equilibrium in order to be useful to an organism so long as it performs its function in a repeatable fashion. This leads directly to the question; when do we stop a simulation? This is a difficult and important question to address since when a simulation is stopped effects the gene expression patterns that result from it. Equilibrium at least gives us a clearly defined point in which to stop the simulations. Hence for this reason an equilibrium filter was employed. This filter stopped any simulation that had reached equilibrium.
Is there pattern?

Yes

No

Is the dynamic robust to noise?

Robust to noise

Non-Robust to noise/Fragile

Figure 3.4: Pattern and noise robustness. (Left) Is there pattern. A particular gene expression pattern resulting from a simulation is illustrated by the continuous orange line. The average of this gene expression pattern is shown by the dashed purple line. The amount of pattern can be worked out by how different it is to this average. (Right) Is the dynamic robust to noise? Different gene expression patterns resulting from 4 different noise runs are illustrated by the different coloured continuous lines. Noise robustness can be calculated based on how different these gene expression patterns are.
Furthermore, the one thing we know for certain is that there is no need to simulate any further a network dynamic that has reached equilibrium. Hence for pragmatic purposes the filter is useful to reduce the amount of computational time that the simulations would take.
Methods:

The discretized form of the equations:
How the concentration $x$ of a gene $i$ will change in any given cell $j$ at time $t$ is described by the following equation:

$$x_{i+1}^{l,j} = \psi \left[ x_t^{l,j} + \chi \left[ \Phi \left[ \sum_{t=1}^{N_g} W^{i,j} x_t^{l,j} \right] + D_t \left[ (x_t^{l,j-1} + x_t^{l,j}) + (x_t^{l,j+1} + x_t^{l,j}) \right] - \lambda x_t^{l,j} + \eta(t)x_t^{l,j} \right] \right]$$

Here $W$ is the interaction matrix described earlier and $\Phi$ is the input function. $D$ is the diffusion coefficient for each gene $i$ and $\lambda$ is the decay parameter which is the same for every gene. The noise term $\eta(t)$ selects a random number within a given range. $\chi(x)$ is the Heaviside function where $\chi(x) = x$ for $\forall x > 0$ and $\chi(x) = 0$ otherwise. Its purpose is to make sure the regulation term can only take positive values. The function $\psi$ is a second Heaviside function where $\psi(x) = x$ for $\forall x > 0$ and $\psi(x) = 0$ otherwise. The function is introduced in order to avoid negative concentrations. The details regarding the different terms of the equation were as follows:

Regulation term:
The input function describes the relationship between the activation and inhibition of a gene and its actual expression. The main input function used in this project was that of a Michalis-Menten function which is defined by the following equation:

$$Output = \frac{\text{input}}{(1 + \text{input})}$$

A second input function was used to test the dependence of the results on the exact input function. This function was a sigmoid function and took the form:
\[ Output = \frac{1}{1 + (e^{5-\text{input}})} \]

The graph for these functions are shown in figure 3.1.

**Diffusion term:**
The amount of diffusion into or out of each cell to an adjacent cell is calculated as the difference in concentration between a gene product in the two cells multiplied by the diffusion parameter.

**Decay term:**
The decay variable (always 0.05) of the gene is multiplied by the concentration of the gene product in that cell and subtracted from the concentration.

**Noise term:**
The noise term randomly generates a number between -0.01 and 0.01 in a uniform distribution and therefore increases or decreases the concentration by up to 1%.

**Boundary conditions:**
After the expression state of all genes in all cells have been processed for a given iteration, the gene products of the boundary cells are set to the same concentration of the adjacent cells (zero-flux boundary conditions).

**‘Tuning’ the core simulation:**
**Parameter range distributions:**
The simulations take place on a theoretical one dimensional row of 32 cells. For each topology a number of different parameter sets are tested. There are up to 12 variable parameters for a 3 gene network; diffusion for each individual gene and then the strengths of the interaction values between the genes. The parameters are chosen randomly though biased towards lower numbers through a logarithmic decay function. The reason for the logarithmic decay function is that the proportional
difference between parameters equal to 0.1 and 0.2 is much higher than that between 1.1 and 1.2 for example. Hence there is more scope for different behaviours at lower parameter values. The logarithmic decay function is described by:

\[ \text{Parameter value} = 0.9995^i P \]

Where \( i \) is a random number between 0 and 10,000 and \( P \) is the parameter range. Parameter ranges are as follows; regulation 0 - 10, diffusion 0 - 0.05. The regulation range was chosen to be in tune with the input function. The input function saturates at inputs of about 10 (See figure 3.1) and thus these values should result in topology simulations that generally fall within the dynamic range of the input function. Regulation ranges between 0 and 10 allow a single gene to fully activate another. The more positive inputs into a gene there are, the more likely it is to be fully activated.

**Initial conditions:**
The simulation starts with every gene in every cell set to have a concentration of 0.1. This is important because the noise used is a percentage noise term and thus if the concentration were always 0 then any percentage of 0 is also 0 meaning that the products of any genes with positive feedbacks without any other input would remain at 0. The simulation is also initiated by the positive input from the pre-pattern gradient that does not change throughout the simulation.

**The pre-pattern:**
The pre-pattern was chosen to give an approximate input range to the gene that it affects of either 10-50% or 10-90%. The pre-pattern is defined by the following equation:

\[ \text{Prepattern input} = Id^c \]

Where \( I \) is the pre-pattern concentration in the left-most cell of the field, \( d \) is the reduction of morphogen concentration in each subsequent cell of the pre-pattern
gradient and c is the cell position, 0 being left and 31 being right. For the 10-50% input range, \( l=1 \) and \( d=0.93 \) was used for the Michalis-Menten input function and \( l=5 \) and \( d=0.98 \) was used for the Sigmoid input function. For the 10-90% input range, \( l=10 \) and \( d=0.865 \) was used for the Michalis-Menten input function and \( l=7.2 \) and \( d=0.97 \) was used for the Sigmoid input function. The two pre-patterns used for the Michalis-Menten function are shown in figure 3.5.

The pattern and noise filter:

Once a simulation of a network with a particular set of parameters has finished, the data is processed by 2 filters: one which tests the pattern score and the other which tests the fragility score.

A patterning score describes how much ‘pattern’ or heterogeneity there is. A patterning score is calculated for each gene of the network. It is measured as the Euclidean distance from the mean concentration of each gene over the field of cells to the actual concentration of the gene product in each cell. It can be defined by the following equation:

\[
\text{Pattern Score} = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(P_i - \bar{P})^2}
\]

Where \( P \) is the vector of the gene concentration of the field of cells, \( \bar{P} \) is the average value of the vector \( P \) and \( N \) is the number of cells. The score is thus normalized to the number of cells.

The fragility score is an inverse measure of the noise robustness of the dynamic. The fragility score is also calculated for each gene of the network. Each network is simulated multiple times (always 4 in this project) with the same set of parameters. Each individual simulation, we term a noise run. An average noise run is generated from these four noise runs. The fragility score is measured as the total Euclidean distance between each of the noise runs and the average noise run. It can be defined by the following equation:
Figure 3.5: The two pre-patterns used in this thesis for the Michalis-Menten function. The Shallow pre-pattern (Blue) is the 10-50% activation range whilst the steep pre-pattern (Red) is the 10-90% input range.
\[
\text{Fragility Score} = \frac{1}{(NR)} \sum_{j=1}^{R} \sum_{i=1}^{N} \sqrt{(P_{i,j} - \bar{P}_i)^2}
\]

Where \(P\) is the array of gene concentrations of each cell \(i\) in each noise run \(j\), \(\bar{P}_i\) is the average value for cell \(i\) between the noise runs. \(N\) is the number of cells and \(R\) the number of noise runs. The score is normalized to the number of cells and the number of noise runs.

A consistent strategy to define what is an acceptable degree of patterning and of fragility had to be found. In order to explore appropriate thresholds of acceptability we plotted patterning score versus fragility score for the resulting dynamics of random topologies from the 3 gene atlas with random sets of parameters (figure 3.6a). Each point on this scatter plot represents the resulting patterning and fragility score for one dynamic. A degree of organization to the distribution of points can be seen. There are particular regions that are rich in points so it seems that only particular dynamics are possible. We wanted to explore what the dynamics of these different regions actually were. To this end we plotted randomly selected dynamics that fell into these different regions. The results are shown in figure 3.6b. Many of the patterns are "left-handed thresholds" reflecting the abundance of this pattern in topology simulations.

Using this illustration of dynamics, we deemed the following thresholds as suitable for selecting a dynamic patterned and stable to noise:

- The pattern score must be above 10. Below this threshold there are only weak patterns.
- The pattern score divided by the fragility score must be above 10. It can be seen that there is a relationship between the pattern and fragility score from the scatter plot. This is unsurprising given the noisier a network simulation is, the more patterned it will be. We looked at random simulations with different pattern score divided by fragility score values. This suggested that the mass of points where the pattern score was approximately equal to the fragility score were not robust enough to be considered for further analysis. For this reason we chose a conservative ratio of 10 as the threshold defining a patterned and robust dynamic.
Figure 3.6 (a): Scatter plot of Pattern versus Fragility score for randomly chosen topologies of the 3 gene atlas simulation with randomly chosen parameter sets. Showing 2,851 points.
Figure 3.6 (b): Example patterns for multiple noise runs. The pattern resulting from the different noise runs are shown in red blue and green. Scatter plot of Pattern versus Fragility score for randomly chosen topologies of the 3 gene atlas simulation with randomly chosen parameter sets. Showing 2,851 points. The thick red line delineates the region where patterns are considered patterned and robust to noise.
**Equilibrium filter:**

To ask whether the dynamic had reached equilibrium, another score is measured. This score measures the Euclidean distance between the gene expression profile (the gene expression patterns of all of the genes) at two time points with 50 iterations of the simulation between them. This measure is made every 50 time steps of the simulation. If this difference was within 1% then the pattern was considered in equilibrium. A maximum of 1,000 time steps are performed.

**Parellization of the simulations:**

We ran these simulations on our in-house computer cluster. The simulation of each topology is not dependent on any other topology. Therefore this is an embarrassingly parallel problem and easy to split the task into smaller tasks for parallel computing. We split the full atlas of topologies into jobs; subsets of topologies of a suitable size for simulation on the cluster. A general shell script is then used to submit each of these jobs to the Sun Grid Engine running on the cluster that was described in chapter 2.
Results and discussion:

Simulating:
What size of atlas can be simulated:
Initial tests showed that a 3 gene network took about one minute to simulate and a 4 gene network took approximately 1 and a half minutes to simulate with 10,000 parameter sets on a typical processor. The full 3 gene atlas has 9,760 topologies and the full 4 gene atlas has approximately 7.2 million topologies. This equates to approximately 150 CPU hours to simulate the full 3 gene atlas and 300,000 CPU hours to simulate the full 4 gene atlas with 1,000 parameter sets. As our in house cluster has 75 CPU’s, it is possible to simulate the complete topology atlas of 3 gene networks with 10,000 parameter sets overnight. This was chosen as the typical run for purely pragmatic reasons as it allowed feedback for any changes. For simulating the 4 gene atlas, we employed the Mare Nostrum supercomputer. These simulations are ongoing at the time of writing.

Standard data sets:
We generated standard data sets using different features for modeling the 3 gene atlas. Four standard data sets were used, shown below in table 3.7. They consisted of the results from the run of the complete topology atlas of networks with 3 genes with 10,000 parameter sets per topology. Approximately 3.7 million different time-stable and noise robust gene expression patterns are produced when the complete topology atlas of 3 gene networks are simulated with 10,000 parameter sets and the Michalis-Menten function. A histogram of the number of topologies producing different numbers of patterns is shown in figure 3.8. This figure only includes the topologies capable of producing a pattern, since the size of the category for 0 would skew the histogram. 54% of all topologies can generate at least some sort of pattern. This suggests that the ability to embellish a pre-pattern is an inherent feature of these topologies (so long as they’re not simple read outs of the pre-pattern). This is substantially higher than the 0.6% of 4 gene topologies that Salazar-Ciudad (et al., 2000) found could generate some form of pattern. The set up of the two models is similar, the main difference being the
<table>
<thead>
<tr>
<th>Data set name</th>
<th>Input function</th>
<th>Input Range (%)</th>
<th>Number of cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Michalis-Menten</td>
<td>10-50</td>
<td>32</td>
</tr>
<tr>
<td>B</td>
<td>Michalis-Menten</td>
<td>10-90</td>
<td>32</td>
</tr>
<tr>
<td>C</td>
<td>Sigmoid</td>
<td>10-50</td>
<td>32</td>
</tr>
<tr>
<td>D</td>
<td>Sigmoid</td>
<td>10-90</td>
<td>32</td>
</tr>
</tbody>
</table>

**Table 3.7:** The 4 different standard data sets used in this project.
Figure 3.8: A histogram of patterning ability for the complete atlas of 3 gene topologies.
continual presence of a pre-pattern in our model. The difference likely comes from the fact that such a pre-pattern can act as a continual source of positional information over the entire field of cells. In many ways 54% is surprisingly low since a pattern could simply be a read out of the pre-pattern.

This result could be due to a three possibilities:

- Many topologies may be overloaded with positive or negative feedbacks resulting in complete saturation or complete repression of all of the genes. This would mean that the simulation does not pass the pattern and fragility filter due to lack of pattern.
- Alternatively the dynamics could be noisy and thus again not passing the pattern and fragility filter.
- The final possibility is that many topologies may be undergoing oscillations and thus not passing the criteria of the equilibrium filter.

It is likely that the reasons that simulations do not pass the filter criteria are different for different topologies with different parameter sets. It will be interesting in future to perform the same simulations again, but turning off the pre-pattern gradient after different numbers of iterations of the simulations.

Summary:

- An abstract yet realistic model of gene regulation has been implemented.
- The ability to ‘embellish’ a pre-pattern is common.
4) Exploring phenotype space

This chapter of the thesis concerns itself with the exploration of the Phenotype space. The result of simulating the topologies is a large set of time-stable and noise robust gene expression patterns. Not all dynamics are possible within the context of the model described in chapter 3 and thus not all gene expression patterns are possible. This section describes how we determine what patterns are possible and how they relate to each other.

As mentioned in the introduction chapter, we wanted to address how many distinct patterns are possible, if indeed there are distinct patterns, and if there are not distinct patterns, what is the shape of phenotype space? The tools developed in this chapter are aimed at addressing these questions.
Introduction

Are there distinct groups of patterns?
The result of simulating all of the gene network topologies of the complete topology atlas is a list of time-stable and noise robust patterns. If we consider the number of variables that describe our patterns, each cell can have a different gene expression level. Thus the pattern can be considered a vector with each element of the vector corresponding to one of the cells of the expression pattern. Vectors can be thought of as points in multidimensional space. Thus as we have simulated a row of 32 cells, each gene expression pattern can be represented by a point in 32 dimensional phenotype space.

The possible gene expression patterns that result from the simulation of the complete topology atlas can thus be represented as a cloud of points in this multi-dimensional space. The questions are then; how diffuse is this cloud? Are there regions of space where no points exist? Are there regions that are much denser than others? Are there multiple separate clouds that would correspond to distinct groups?

We primarily wanted to address whether there were distinct groups of patterns. If there are distinct groups of patterns, then it should be possible to assign these points to categories based on their positions. Assigning points in a multi-dimensional space to a given category is the role of pattern classification.

In the following sections, I will first give an overview of the two main strategies employed to tackle the problem of pattern classification, namely supervised and unsupervised strategies, and why we chose the latter to perform our pattern classification. I will discuss some of the common methods of the unsupervised strategy relating their suitability to our problem and then I will discuss the specific implementation that we have employed.

Methods of classifying patterns (Supervised strategies):
Pattern recognition aims to classify patterns based on either a priori knowledge or on statistical information extracted from the patterns, which are multidimensional (Duda et al., 2001). Pattern recognition systems require some form of feature abstraction that computes numeric or symbolic information from the observations and a
classification scheme that uses these abstracted features to classify each pattern. The classification scheme usually requires a set of patterns that have already been classified (labelled) that are known as the training set. Procedures that use labelled samples in their training sets are said to be supervised. This training set is used to abstract the features that the classifier will then use on the test set which are unlabelled. The conceptual boundary between the feature extraction and the classifier are somewhat arbitrary however as a perfect feature abstractor would make the classifiers job trivial.

*Feature abstraction:*
Features are individual measurable heuristic properties of the phenomena being observed (Duda et al., 2001). For example in text character recognition features may include the number of holes and stroke detection of the individual letters. Choosing discriminating and independent features is key to a pattern recognition algorithm’s success. Extracting features is currently more of an art form than a science and with the exception of some neural networking and genetic techniques that automatically intuit features, hand selection of good features forms the basis of almost all classification schemes.

*Pattern classifying (matching):*
Pattern classifying is the process whereby a pattern is checked for the presence of features of a rigidly defined pattern. Pattern classifiers do not necessarily have to assign every pattern to one of the defined groups as some patterns may not contain all of the features that define each group.

As described, supervised strategies of classifying patterns require some training data that has already been assigned a category. However in our context we have no training data as this project is the first step in the exploration the space of possible gene expression patterns. For this reason supervised classification strategies are not the most suitable for this project. To tackle problems where nothing is known about the set of data to be classified one must turn to unsupervised strategies.
Unsupervised learning approaches:
Unsupervised learning procedures use unlabelled samples, such that the system is not given an *a priori* labelling of patterns, instead establishing the patterns itself based on the statistical regularities within the patterns (Duda et al., 2001). It has been commented “One might wonder why anyone is interested in such an unpromising problem and whether or not it is possible even in principle to learn anything of value from unlabelled samples.” (Duda et al., 2001). However, when nothing is known about the structure of the space being explored, there is no other way of approaching the problem. These unsupervised types of approaches can be used to gain some insight about the structure of the space being analyzed for example, its continuity or discontinuity as described.

Clustering:
A well known type of unsupervised learning approach is that of clustering of which there are several types. Clustering is the classification of objects into different groups so that the objects of a given group share some common trait(s) (Duda et al., 2001). Clustering algorithms decide which objects belong to which groups based on some kind of similarity function. Data clustering algorithms can be hierarchical or partitional. Hierarchical algorithms finds successive clusters using previously established clusters, whereas partitional cluster algorithms find all clusters in one go. All clustering methods require a similarity measure and some form of global criterion function that will now be discussed.

Similarity measures and the similarity matrix:
A similarity measure is a measurement of the similarity between a pair of objects. Many types of similarity measure exist between objects defined by many continuous variables. The most commonly used is the Euclidean distance, as illustrated in figure 4.1. It is simply the distance in multidimensional Euclidean space which can be worked out by performing the following function:

\[
\text{Euclidean distance} = \frac{1}{N} \sum_{n=1}^{N} \sqrt{(a_n - b_n)^2}
\]
Figure 4.1: Euclidean distance measurement for two points (blue dots) in two dimensional space. The distance between the two points is measured along each of the dimensions and squared to make them all positive. These component distances are summed and the square root taken.

Euclidean distance = \sqrt{(dy^2 + dx^2)}
Where the Euclidean distance is the difference between two vectors $a$ and $b$ with a given number of dimensions (N).

Regardless of which similarity measurement was used, if a set of objects are compared using the similarity measure in all pair-wise combinations, the result is a triangular matrix of similarity (figure 4.2). How much of the information in this matrix is used to group the objects then depends on the method of grouping.

**Criterion function:**

The purpose of the criterion function is to define the quality of the clustering. The criterion function is a measure of how good is the partition of the total set of objects into a particular set of clusters. The clustering problem then is one of finding a partition that extremizes the criterion function. Let $D = \{x_i, i = 1,2,...n\}$ denote the data set to be clustered such that each point $x_i$ is a vector of $p$ features. Suppose a clustering algorithm divides the data set $D$ into $k$ different clusters $C_r$, $r = 1,2,...,k$. $C_r$ is now the number of objects within that cluster.

**Sum of squared error:**

The simplest and most widely used criterion function for clustering is the sum-of-squared-error (SSE) criterion (Duda et al., 2001). The sum-of-squared error for a given cluster (SSE) is defined by:

$$SSE = \sum_{i=1}^{C_r} (x_i - \bar{m})^2$$

For a given cluster, the mean vector $\bar{m}$ is the best representative of the samples in the cluster in the sense that it minimizes the sum of the squared lengths of the “error” vectors $(x - \bar{m})$ in the cluster. The total squared error then is the sum of this value for all of the clusters. If $k$ is the number of clusters and $C_r$ is the number of objects in that cluster then the total SSE for a given number of clusters can be defined by:

107
Figure 4.2: (Left) A triangular matrix describing all pair-wise distances between 4 objects (A, B, C and D). (Right) The four objects in two dimensional space. Objects are shown as blue dots. An example distance is given between objects D and C (blue double ended dashed arrow).
The sum-of-squared-error criterion is well suited to situations where the clusters form compact clouds that are rather well separated from one another. There can be problems with this criterion however when there are big differences in the number of samples in the different clusters (Duda et al., 2001). In this situation, this criterion can split a large cluster that would otherwise form a natural grouping. This situation frequently arises because of the presence of outliers.

Cluster dispersion:
An alternative criterion function is the cluster dispersion otherwise known as the within-cluster sum of squares, which is a measure of compactness of the cluster. The within cluster dispersion for a given number of clusters ($W_k$) can be defined as:

$$W_k = \sum_{r=1}^{k} \frac{1}{2C_r} \sum_{ij \in C_r} (x_{r,i} - x_{r,j})^2$$

Where $(x_{r,i} - x_{r,j})^2$ is the distance between the two vectors $x_{r,i}$ and $x_{r,j}$. This value is summed for all pair-wise combinations of the vectors in this cluster and averaged.

Once a criterion function has been chosen, clustering becomes a well-defined problem in discrete optimization (Duda et al., 2001). In other words a partition of a set of samples needs to be found that extremizes the criterion function. Although there are only a finite number of samples, testing all possible cluster combinations is usually unfeasible. For example there are $c^n/c!$ ways of portioning a set of $n$ elements into $c$ subsets. Even with only 100 samples, the exponential growth with $n$ means that there are $10^{67}$ possible partitions for an exhaustive search for the best set of 5 clusters.
Some form of approach then needs to be used to find the optimal portioning of the data in a much more efficient way. Various clustering procedures have been developed with exactly this goal.

Types of clustering procedure:

Partitional (flat) clustering:

K-means clustering:

k-means clustering is a partitional clustering approach (Duda et al., 2001). Each point is assigned to the group whose centre (centroid) is the nearest. The centre is the Euclidean average in multi-dimensional space of all of the points within the group. The k-means algorithm proceeds as follows:

- Choose a number of groups k.
- Randomly generate k groups and determine the centers, or directly generate k random points as group centers.
- Assign each point to the nearest group.
- Re-compute the new group centres.
- Repeat until a conditional is met. Usually that some convergence criterion is met such as the assignment hasn’t changed.

Fuzzy k-means clustering:

In k-means clustering each data point is assumed to be a member of exactly one cluster. If this condition is relaxed, and we assume that each data point has some graded or “fuzzy” membership in a cluster then we have what is known as fuzzy k-means clustering (Duda et al., 2001).

Hierarchical clustering:

The clustering algorithms just described, form disjoint or flat clusters. However, sometimes clusters have subclusters and subsubclusters and so on (Duda et al., 2001). A good example of this is in biological taxonomy where kingdoms are split into phyla which are split into subphyla which are split into orders, suborders, families, subfamilies, genus and species. If we seek to reveal such structure in data
then hierarchical algorithms are needed. The resulting hierarchical structure is revealed in a dendrogram.

Because we expected to find patterns and ‘elaborations’ of those patterns, that is the same pattern but with additional less important features, we expected a hierarchical structure to best describe the gene expression patterns resulting from the simulations. Thus a hierarchical clustering approach seemed very suitable for this project given that it is unsupervised and can highlight structure at multiple levels.

**Estimating the number of clusters (The problem of cluster validity)**

Estimating how many unique clusters there are in a set of data is often difficult and there is no strict definition of what represents a different cluster. Two people may look at the same set of data and say that a different number of clusters exist. What we want is some index that gives us a measure of separability. This index must be independent of cluster volume, size and shape and also be sensitive to outliers (Duda et al., 2001). Computer based approaches to this problem at least give a more explicit definition of what is considered a distinct cluster and can be categorized as local and global approaches (Gordon, 1999). Local approaches determine the number of clusters by first grouping the objects into small clusters and then testing whether those small clusters should be merged into one based on some criteria. Global approaches in comparison determine the number of clusters by optimizing some global criterion function that measures the overall quality of the clustering.

Local approaches:

Local approaches state a threshold of similarity for the creation of a new cluster (Duda et al., 2001). These type of approaches are particularly useful for ‘online clustering’ cases where an algorithm must work as the input data is being presented to it. They are also useful when the number of input points is large since they do not require the comparison of every point to every other point. The granularity algorithm described later (See methods) uses this type of approach for deciding the number of different patterns in the raw simulation data. A drawback of local approaches is that they depend more strongly on the order of data presentation.
Global approaches:

The elbow criterion (or L-curve method):

The elbow criterion is a common rule of thumb to determine what number of clusters best represents the data (Duda et al., 2001). The elbow criterion says that you should choose a number of clusters so that adding another does not add significantly more information. This information can be calculated using the criterion function such as the cluster dispersion.

A graph of cluster dispersion can be drawn plotted against the number of clusters. The first clusters will explain a lot of variance, but at some point the marginal gain will drop giving an angle on the graph (the elbow). The elbow cannot always be unambiguously identified.

Global approaches for cluster validation are more suitable to our problem, since we do not have to cluster while the gene network simulations proceed (online clustering), but simply cluster after the gene network simulations have finished. These global approaches are more accurate than the local approaches since they use more of the available pair-wise similarity data and do not depend so much on the order of data representation to the algorithm.

The number of clusters within a hierarchical cluster?

A cluster in the case of a dendrogram refers to starting from the root node and moving towards the leaves based on the next clustering event that had the largest difference. At each step a new cluster is added. For example in figure 4.3, after 2 moves from the root node we have 3 clusters and after 6 moves we have 7 clusters.

Within a hierarchical dendrogram, the total number of clusters is not a very useful parameter, since the point of a hierarchical cluster is to reveal structure at multiple levels (i.e. clusters within clusters). The elbow criterion can be used with data from hierarchical clusters however, since multiple elbows can appear within the data, illustrating where those levels occur (figure 4.4).
When $k = 7$

When $k = 3$

Figure 4.3: The elbow criterion. (Left) Example clusters of the same set of data with 7 or 3 leaf nodes. (Right) A plot of the cluster dispersion against the number of clusters showing an elbow at $k=3$. 
Figure 4.4: Multiple elbows in the cluster as indicated by the red arrows.
Exploring the shape of phenotype space: Dealing with multidimensional space:
As mentioned earlier, we wanted to explore the shape of viable phenotype space. Phenotype space is 32 dimensional when using the gene expression states of a gene in each cell as the features. How then can we get an idea of the shape of this space? I will now discuss approaches for representing higher dimensional space in fewer dimensions.

**Dimensionality reduction:**
"Part of the problem of deciding whether or not a given clustering means anything stems from our inability to visualise the structure of multidimensional data" (Duda et al., 2001). One way to address this problem is to try and represent the data points as points in some lower dimensional space. The aim is to map points close in source space so that they are close in lower dimensional target space. "In preserving neighbourhoods in this way, such maps are said to be ‘topologically correct’” (Duda et al., 2001).

**Graph-theoretic methods:**
One approach to represent the original data in some lower dimension, is by representing the data as a graph (a network), that can be viewed in 2 or 3 dimensions. One way to generate this graph is by using a threshold distance whereby two points (representing objects) are connected if the distance between them (from the triangular similarity matrix) is below this value. This results in a similarity graph, whereby points are joined by edges that represent the differences that are below this threshold.

By using a so called ‘spring algorithm’ this graph can then be spaced in 2 or 3 dimensions. The result of such a spring algorithms is to space the data points and does not result in objects being assigned to specific clusters. Such tools are useful to explore the shape of the space instead. How spring algorithms work and two specific implementations of them will now be discussed.
Spring algorithms (force directed algorithms):

Force directed algorithms can be used for drawing graphs in an aesthetically pleasing way. They act to position the nodes of the graphs in 2 or 3 dimensions so that all edges are of more or less equal length and that as few edges are crossing as possible. Force directed algorithms assign forces amongst the set of edges and the set of nodes in the graph. A straightforward approach is to treat the edges like springs by implementing Hooke's law and the nodes as electrically charged particles by implementing Coulomb's law. The springs act to pull the nodes together and the charges on the nodes act to push them apart. The graph can be simulated as if it were a physical system and an equilibrium reached after enough iterations.

An alternative approach defines an ideal distance between any two nodes based on some similarity function. The more similar the objects the smaller the ideal distance. This type of force directed algorithm does not require a separate repulsive force. Other forces that can be involved in these algorithms include magnetic and gravitational fields. In the case of the spring-and-charged particle graphs, the edges tend to have uniform length because of the spring forces whilst any nodes that are not connected tend to be pushed apart due to the electrical repulsion.

Two examples of freely available spring algorithm software are Large Graph Layout and Biolayout. They will now be discussed.

- **Large Graph Layout (LGL)**

  *LGL is based on a mass spring algorithm where edges play the role as springs*

Because biological networks are often very large, visualisation tools can often be inadequate to display them. For this reason, Adai (*et al.*, 2004) developed an algorithm than employs a forced-based algorithm to space and display this data (*Adai et al.*, 2004). They used this algorithm to visualise an extensive protein map consisting of 145,579 proteins from 50 genomes. Edges within this graph depend on the level of homology between the proteins that correspond to the nodes at the ends of that edge. The resulting graph served as a map of gene function and showed clearly distinct modules (Figure 4.5).

An important feature of this layout algorithm is that it employs a guide tree, a strategy that has been shown to effectively separate dense networks (*Cheswick et al.*, 2004).
Figure 4.5: An example of the results of the large graph layout algorithm. A protein sequence homology map from multiple genomes showing how distinct map regions correspond to distinct cellular systems. The points are proteins and the lines connecting the proteins are similar sequences. From Adai et al., 2004
A guide tree means that the force based algorithm is not performed on all of the nodes simultaneously. Instead nodes are threaded on to the graph and after each node is added, some iterations of the force based algorithm are performed until equilibrium is reached. The order by which nodes are threaded on to the graph depends on their position within a Minimal spanning tree. The minimal spanning tree is defined as the minimum set of edges necessary to keep the network connected, where each edge is weighted by the distance function between the nodes, in this case the associated BLAST E-value. The nodes are added to the force algorithm starting from the root of the minimal spanning tree until the leaves are reached.

- **Biolayout**

Biolayout is an alternative general purpose automated graph layout algorithm for similarity visualization (Enright and Ouzounis 2001; Goldovsky et al., 2005). Biolayout utilizes a modified version of the Fuchterman and Rheingold graph layout algorithm specifically for the use in similarity analysis in biology. This algorithm rapidly generates 2D and 3D graphs representing similarity relationships such as protein sequence similarity. This algorithm uses the types of force directed algorithm described where vertices in the graph repel each other whilst edges serve to attract. This algorithm by contrast uses random placement of vertices rather than using anything like a guide tree. However a temperature function is used to refine the process. The temperature function slowly decreases the movement of the vertices reminiscent of simulated annealing algorithms.
Overview of our approach:
We ideally wanted to use an unsupervised learning approach, since this approach is more likely to yield the true structure of the space rather than imposing structure upon it. Furthermore, we were seeking to find structure at multiple levels. Thus we tried hierarchical clustering with a global criterion function to define the number of clusters. We found that although the general structure of pattern (phenotype) space could be explored, clearly distinct pattern groups did not result.
We also tried using the Biolayout and LGL force directed algorithms to represent the 32 dimensional data in 2 or 3 dimensions. The aim was to gain some insight into the general shape of the phenotype space and to validate the structure observed through clustering. These algorithms can use much more information from the triangular similarity matrix than the clustering algorithms (depending on the threshold of similarity used and thus number of edges in the graph) and therefore may lead to a better representation of the structure than the clustering. However these algorithms did not yield any more information than the clustering algorithms.
We then used a pattern matching approach to classify the patterns into different groups for the purposes of generating ‘Nearly Neutral Networks’ (See Chapter 5). We defined the spatial gene expression patterns by hand and encoded them digitally and assigned pattern to these groups if they conformed to the digital definitions. Using such an approach we could assign 96% of patterns to 10 pattern groups.
Methods and Results:

The methods and results section will be organised in the order with which the different approaches were attempted. First I will discuss how we abstracted phenotype space to a manageable number of points. Then I will discuss the clustering and spring algorithms approaches and how they were implemented and finally describe how we performed pattern matching. A summary of the data flow through the algorithms is given in figure 4.6.

Abstracting pattern (phenotype) space to a manageable number of points:
Granularity:
The simulation of the complete topology atlas of 3 networks of 3 genes with 10,000 parameter sets resulted in 3.7 million time-stable and noise robust gene expression patterns (See chapter 3). This number is currently too large for any sophisticated pattern grouping algorithm to feasibly use: it would take too much computational time. Just to compare each pattern to every other pattern would require $13.7 \times 10^{12}$ comparisons.

To remedy this problem we employed an algorithm called granularity. The purpose of the granularity algorithm was to abstract the phenotype space so that the original set of points is represented by less points (granules), though with a similar distribution in phenotype space (figure 4.7).

Each granule is represented by a single pattern that falls within that region of phenotype space. The granularity algorithm assigns a pattern to a granule if the Euclidean distance between the pattern and the pattern that represents the granule is less than a certain threshold ($D$). If a pattern is not within the distance $D$ to any of the granules, then a new granule is created with this pattern as the representative of that granule. The granularity approach is simple and thus is computationally inexpensive. Our granularity algorithm only depends on a single parameter (provided as an argument to the function). That parameter is the maximum number of granules allowed. The granularity algorithm uses a small distance threshold ($D$) as the maximum Euclidean distance described above. If the number of granules generated
Gene expression patterns resulting from simulations

Granularity

Filters

Pattern Matching

Filters

Hierarchical Clustering

Spring Functions

Figure 4.6: A summary of the algorithms used to explore phenotype space
Figure 4.7: The effect of applying the granularity algorithm on phenotype space. (Left) Each phenotype is a point in multidimensional space (here represented by 2 dimensions). Applying the granularity algorithm discretizes this space. (Right) Each resulting granule contains information about the number of original phenotypes within the space indicated here by the depth of shading.
exceeds that given as the argument, then granularity will increment the threshold (D) by 30% and start the whole process again. In this way, an approximate number of granules is generated that is less than the maximum number used as an argument rather than a specific number.

Granularity does not have to be sophisticated as it is simply a way of reducing the number of data points inputted into the more sophisticated clustering algorithms. Thus granularity should only be performed using a small threshold (D). It is important that granularity does not go too far and start grouping patterns together in the same granule that are too different from each other. If it does so, then the resultant clusters will not be a good representation of pattern (phenotype) space. If clustering is used with a particular level of granularity to do further analysis, then the cluster should have the same general structure if a much lower level of granularity was used. Indeed this is the best criteria for measuring the level of granularity required.

The granularity algorithm was run using a maximum number of phenotypes of 1250 and 2500 for standard data set A (which resulted in 1051 and 2416 patterns respectively). These two values were chosen to show that the overall cluster structure does not depend on the degree of granularity used.

Unsupervised learning approaches:
Filters were applied to the pattern data in order to aid the pattern grouping function. I will now describe the filters we generated to use on the gene expression data resulting from the granularity algorithm and the similarity function used to generate the triangular similarity matrix. The exact filters used depends on the type of analysis being performed and will be described later.

Filters:

Smoothing:
Smoothing was used to reduce the affects of noise. The input argument for the smoothing filter was the window size $W$ of the smoothing function. The smoothing
function was carried out for all cells. The new expression level of a gene $G$ in a cell $i$ is worked out by the following equation:

$$G_i = \frac{1}{(2W + 1)} \sum_{j=(i-W)}^{i+W} G_j$$

For cells within $W$ of the boundaries on the left and right hand side of the cell field, this function is not possible. For these cells, the expression values are summed until the boundaries and the total sum only divided by the number of cells contributing to that sum.

**Discretization Abstracting:**
The purpose of the abstracting filter was to make the data discreet. The inputs to the abstracting function included the discretization value of the abstraction. This was an integer that describes how many distinct expression levels there will be. The maximum value that any expression level could be was 20. This is because the maximum output from the input function described in the gene regulation model in chapter 3 is 1 and the decay parameter is always set to 0.05. When production of gene product balances decay of gene product at the highest expression rate then the concentration is 20. The function that performs this discretization Where the new expression level of a gene $G$ in a cell $i$ and the discretization value is $d$ is the following:

$$G_i = G_i - (G_i \% (20/d))$$

Therefore if a discretization value was given as 10, the algorithm would set a value of 18.73 to be 18, or 0.52 to be 0 for example. Abstracting can be thought of as another way of performing smoothing, however it does not suffer from the effects that smoothing has on small scale features within the patterns. For example take the
pattern in figure 4.8. After smoothing, this pattern the small dip (which is not due to
noise) before the threshold is damped and thus although it is an important feature of
the pattern will contribute less to the assignment of this pattern. However with
discretization abstraction, the dip remains (so long as the discretization value is large
enough). This figure illustrates a negative point about this filter however in that it
breaks gradients into a set of steps.

Rank Abstracting:
Rank abstraction has been used in other works to classify pattern types (Salazar-
Ciudad et al., 2000). The algorithm loops through every cell and finds the cell with
the highest expression level. It gives this cell a rank of 32 (the number of cells). Then
it loops through the cells until it finds the next highest value and gives this cell a rank
of 31. This is continued until every cell has been assigned a rank. Problems with the
rank abstraction include its sensitivity to noise and problems with flat and gradient
regions. Because ranks are enforced on the pattern, ranks are enforced on a flat
region based on noise and thus two similar patterns with flat regions maybe
negatively scored due to this feature. It is an advantage to perform a discretization
abstraction before a rank abstraction so as to minimise these effects.

Amplifying to the maximum expression level:
This procedure involved expanding the expression pattern for each gene so that the
cell with the maximum gene expression of the pattern $M$ is set to the maximum gene
expression that any gene can possibly take (which is 20; see discretization
abstraction above). The amplification factor $Af$ can be defined as:

$$Af = \frac{20}{M}$$

The new expression level of a gene $G$ in a cell $i$ is then as follows:

$$G_i = Af G_i$$
Figure 4.8: Discretization abstracting versus smoothing. (Left) The original gradient and peak pattern. (Right) the resulting patterns after the different filters have been applied.
By performing this filter, like the rank abstraction, similar patterns but have a different scale in the gene product concentration axis will get a better score when compared. However unlike the rank abstraction, this filter does not have problems with flat areas and does not convert a gradient into a set of steps.

**Similarity function:**
The similarity functions between patterns described below were first calculated for each pair-wise combination of patterns. The data was stored in the triangular similarity matrix that was used by the appropriate algorithm to space the phenotypes.

**Absolute similarity:**
The absolute similarity is the most simple type of similarity function. It measured the distance between any two patterns (say A and B) in 32 dimensional Euclidean space.

\[
\text{Euclidean distance} = \frac{1}{\text{cells}} \sum_{c=1}^{\text{cells}} \sqrt{(a_c - b_c)^2}
\]

Where \(a\) is the vector of gene expression values for pattern A and \(b\) is the same vector for pattern B. \(c\) is the current cell being processed and \(\text{cells}\) is the total number of cells.

**Attempting to group the patterns:**
*Clustering: Exploring the different functions and filters on a real data set:*
We explored which were the best similarity functions and filter sets for grouping the different phenotypes. We explored several filter types (described in materials and methods).

**Hierarchical Clustering:**
At the beginning of clustering each pattern is set to be in its own group. For example pattern 1 in group 1, pattern 2 in group 2 etc. The number of clustering steps required to generate a full cluster is always the same no matter what the structure of the
cluster. That number is \((N-1)\) with \(N\) being the number of objects (or leaf nodes) that are to be clustered. The following pseudocode performs the clustering:

Set each pattern to be in its own group
Number of nodes = number of objects to be clustered

**Do**

Find smallest distance in triangular similarity matrix

**If** the two patterns are not in the same group.
    Amalgamate the two groups of these patterns
    Number of nodes--
**Else** find next smallest distance in triangular similarity matrix

**While** number of nodes > 1

*File format:*
The clusters were outputted in Newick format, which is one of the standard formats for describing cluster trees. The Newick format uses a string of parenthesis and commas to describe the structure of a tree. Anything contained within two matching parentheses represents a single branch in the cluster and a comma represents a clustering event. Examples of the format and the corresponding trees are shown in figure 4.9. Note that the Newick format allows for a node to have more than two daughters though our clustering algorithm does not. The string must terminate with a semicolon. Labels and size of the cluster step can also be added using this format

The full Newick string was generated by saving a string in Newick format for each of the groups (branches) of the cluster as the clustering progressed. Every time two groups were amalgamated, their Newick strings representing those groups were also amalgamated (and the correct syntax added). Hence at the end of clustering, there is one big Newick string describing the structure of the whole cluster.

*Dendrogram layout:*
A Graphical User interface (GUI) was written in TclTK in order to allow the exploration of the clusters in a visual and interactive way. Appendix A describes the available control options for this GUI. Once the structure of the cluster has been
Figure 4.9: The Newick file format. (Above) Example dendrograms (or trees). Both have 4 leaf nodes. (Below) The corresponding Newick strings. Branch points are indicated by a comma.
defined using the output array, the coordinates need to be worked out for this GUI. The nodes of the cluster that need to be assigned coordinates are both the leaf nodes (the original objects to be clustered) and internal nodes (cluster branch points). This part of the algorithm works by starting from the root node of the cluster. The root node is assigned coordinates (0,0) It uses the Newick data structure to find the daughters of this node and spaces them in the y axis by adding the distance in the triangular similarity matrix to the y distance of the parent (in this case 0). It spaces the daughters in the x axis by simply taking the x axis value of the parent and adding 1 to one of the daughters (daughter A) and -1 to the other (daughter B).

The algorithm then loops through all of the nodes that have been assigned an x axis previously. If any of these nodes has an x value equal or greater to daughter A then 1 is added to the x value of this node. If any node has an x value equal or less than daughter B then 1 is subtracted from the x value of this node. This scheme acts to space apart the dendrogram so that all leaf nodes will be lined up on the x-axis, spaced apart by one unit.

Example dendrograms for Standard data set A are shown in figure 4.10 with 23 and 82 leaf nodes. Trees with different numbers of leaf nodes such as this can be built by changing the maximum number of patterns allowed to result from the granularity algorithm (See earlier).

What do we learn from these dendrograms?

Different types of similarity function are good for grouping different patterns:
The following results were obtained using the results from standard data set A. The results are shown for the data set with 2,416 leaf patterns. The clusters using the basic Euclidean distance similarity function show that the patterns are grouped with regard to general features. For example, large groups exist for left-hand and right-hand patterns (figure 4.11). Amplifying the patterns so that the cell with the highest expression of the gene has the maximum possible expression value results in extra intricacies to the grouping (figure 4.12 top). For example the dip patterns are starting to come together as large
Figure 4.10: (Top) A example cluster with 23 leaf nodes. (Left) Showing the branching structure and nodes only. (Right) Showing the leaf patterns on top of the branching structure. (Bottom) A example cluster with 82 leaf nodes. (Left) Showing the branching structure and nodes only. (Right) Showing the leaf patterns on top of the branching structure.
Figure 4.11: Hierarchical clustering 2416 patterns using a simple Euclidean distance similarity measure. (Left) Showing the Dendrogram structure. (Right) Showing the patterns overlaid. Two general groupings can be seen as indicated by the white arrows.
Figure 4.12: Hierarchical clustering of 2416 patterns with a filter amplifying gene expression values to that of the maximum cell. (Top left) Showing the Dendrogram structure. (Top right) Showing the patterns overlaid. Three general groupings can be seen as indicated by the white arrows. A region of interest showing the successful clustering of the dip patterns is shown as a blue dashed square. (Bottom) The region of interest.
groups (figure 4.12 bottom). The other filters made little difference to the quality of the clustering however.

**Pattern (phenotype) space is a continuum:**
We calculated the cluster dispersion of the clustering for the data set with 2,416 leaf patterns with the basic Euclidean distance similarity function for different numbers of clusters to investigate whether there are any elbows (figure 4.13). The graph reveals that there is one large elbow. However this elbow occurs at approximately 800 clusters. At this stage, the majority of the patterns belong to two large groups. This suggests that pattern (phenotype) space at least when based on raw gene expression levels is a continuum within these two large clusters.

**The spring functions:**
In order to explore in greater detail the structure of phenotype space and to address whether there really were two large continuous groups of patterns we tried both the Biolayout and the LGL algorithms to space our data. Our clustering algorithm produced the input file formats needed for these algorithms. They consisted of a simple two column list where each row corresponds to each link, the columns denoting the start and end node (representing patterns in the graphical layout). An optional third column can denote the weight of the link. After the triangular similarity matrix is generated, the clustering algorithm asks whether the distance between each pair-wise combination of the nodes is under a threshold of 2,500 (This value was used as it allows as many node-node links to be included as possible without causing memory problems for the LGL or Biolayout algorithms). If so then it writes a link for this pair of patterns to the input file and also sets the weight of that link to be the distance between the two nodes in the triangular similarity matrix. These input files were then spaced and imaged with the appropriate LGL or Biolayout programs.

The input data for the spring functions is the same set of input patterns as for the clustering algorithm. The results of the LGL and Biolayout functions are shown in figure 4.14. The structure for the LGL is shown in 2 dimensions whilst for Biolayout...
Figure 4.13: The cluster dispersion for different numbers of clusters. The main elbow is shown by the red arrow.
Two groups of patterns that are more intraconnected than interconnected.

Figure 4.14: (Top) A map of phenotype space created using the Biolayout algorithm (edges are those similarity measures of 2,500 maximum. Each of the balls represents one of the 2,416 input patterns. The lines represent those pairs of patterns that have a similarity of below 2,500. (Bottom) The LGL spacing of the patterns (edges are those with maximum of 2,500 similarity score). Each of the points represents one of the 2,416 input patterns. The lines represent those pairs of patterns that have a similarity of below 2,500.
in 3 dimensions. As can be seen, no obvious clusters can be observed, though a general grouping of points into two large groups is apparent in the LGL image, but not that of Biolayout. The general conclusion therefore still remains that phenotype space when assessed in this unbiased way, shows a continuum.

**Pattern matching:**
Although it appears that the viable region of phenotype space is fairly continuous, with patterns all blending into each other, and no obvious clusters, nevertheless for subsequent analysis (in chapters 5 and 6) we wanted to explore the space in a different way. We wanted to see if the gene network topologies underlying regions with specific patterns form ‘Nearly Neutral Networks’. In order to generate groups of specific patterns, we employed a pattern matching approach which grouped patterns together based on the presence of subjectively defined pattern features.

**Extracting features:**
The granularity algorithm was run on the resulting time-stable and noise-robust patterns from the simulation of the full atlas of 3 gene topologies run with 10,000 parameter sets. The maximum number of patterns allowed was set to 1,000. The resulting patterns were clustered using raw data and Euclidean distance (without any filters). This gave a rough structure to the patterns, which helped to identify the features needed to classify them by eye.
Patterns were then defined by scanning the pattern from left to right and using the criteria such as is it flat, going up or going down. If the region was flat then another criteria included whether it was zero or non-zero. Repetitive arrangements such as stripes were also included.
Hence a stripy pattern can be digitally defined as follows after discretization filtering:

*It goes up*
*It goes down*
This sequence must be repeated at least two of times. Also the following definition belongs to this group.
*It goes down*
It goes up

This again must be repeated at least two times. If the pattern is at any time flat then this pattern fails the criteria.

Before the pattern matching algorithm can be run, the patterns resulting from the atlas simulation must be filtered in some way to remove the noise that may result in an incorrect classification. Various different abstraction filters were used depending on the pattern group (see results for details). This is because filters such as the discretization filtering have problems when it came to gradients as gradients become step like when put through this filter (figure 4.8).

Pattern matching was then performed on the 3.7 million patterns resulting from the simulation of the complete atlas of 3 gene topologies. There are no problems with computational time when performing pattern matching because the number of comparisons does not increase with a power of 2 like in clustering. Each pattern only needs to be compared with each digital definition.

A list of the 10 defined patterns and their digital definitions for the run of PreOneN1-3Kmax3 with 10,000 parameter sets is given in table 4.15. These patterns definitions were run on the raw data set resulting from the simulations and 96% of the patterns could be assigned to one of these groups.
<table>
<thead>
<tr>
<th>Pattern type</th>
<th>Features</th>
<th>Filters applied first</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stripes</td>
<td>Goes up, Goes down (Repeated at least two times) or Goes down, goes up (Repeated at least 2 times)</td>
<td>Discretization abstraction</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td>Single stripe</td>
<td>0 or 1 for at least 3 cells, goes up, goes down, is 0 or 1 for at least 3 cells</td>
<td>Discretization abstraction</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td>Left-handed gradient</td>
<td>Each cell has less expression than the cell to the left.</td>
<td>Smoothing or rank abstraction</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td>Right-handed arc</td>
<td>Each cell has more expression that the cell to its right.</td>
<td>Smoothing or rank abstraction</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td>Dip</td>
<td>Is more than 0. Stays the same. Goes down to non-zero, goes back up to the same level and stays the same.</td>
<td>Discretization abstraction</td>
<td><img src="image" alt="Example" /></td>
</tr>
</tbody>
</table>

**Table 4.15:** Pattern matching. Showing the 10 pattern types, the features used to classify them, the filters that were applied first, and an example pattern.
<table>
<thead>
<tr>
<th>Pattern type</th>
<th>Features</th>
<th>Filters applied first</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left-handed threshold</td>
<td>Is more than 0. Stays the same. Goes down to 0 within the space of 3 cells</td>
<td>Discretization abstraction</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td>Right-handed threshold</td>
<td>Is 0. Goes to more than 0 within the space of 3 cells, stays the same.</td>
<td>Discretization abstraction</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td>Left-handed gradient and peak</td>
<td>It goes down, it goes up, it goes to 0.</td>
<td>Rank abstraction for gradient, discretion abstraction for the rest</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td>Left-handed gradient and threshold</td>
<td>If goes down, it goes up within the space of 3 cells, it stays the same</td>
<td>Rank abstraction for gradient, discretion abstraction for the rest</td>
<td><img src="image" alt="Example" /></td>
</tr>
<tr>
<td>Double peak</td>
<td>Is 0, It goes up, It goes down, It goes up, It goes down to 0.</td>
<td>Discretization abstraction</td>
<td><img src="image" alt="Example" /></td>
</tr>
</tbody>
</table>

**Table 4.15**: Pattern matching. Showing the 10 pattern types there features used to classify them, the filters that were applied first and an example pattern.
Discussion.

What is the structure of pattern space? Are there distinct groups of patterns?

Pattern clustering and Force directed algorithms were used to gain some insight into the general structure of pattern space. Hierarchical pattern clustering results in two large groups of patterns. The patterns in those groups are typically right or left-handed gradients and thresholds (See results). The result of the LGL algorithm also showed two large groups of patterns that appear more internally than externally connected. Interestingly, the result from the Biolayout algorithm does not show these two large groups. This difference is likely to come from the fact that the LGL algorithm uses a guide tree whilst the Biolayout algorithm does not. Thus the Biolayout result is possibly stuck in local optima.

Looking at the patterns by eye there appeared to be some genuinely different pattern types (See appendix B). The fact that we are seeing many different pattern types but only two large groups clustering together suggests that patterns are distributed in phenotype space in two large continuums (one for the left and one for the right-handed patterns). There are no large jumps between adjacent patterns in this space, patterns can gradually change into one another. For example a left-handed gradient and threshold pattern can become a left-handed gradient pattern by gradually reducing the size of the threshold (Figure 4.16).

An illustration of this type of distribution is shown in figure 4.16. In this example some patterns exist in ‘lobes’ of the shape where one can only gradually change into other patterns via specific directions in phenotype space. These patterns are likely to be the more complicated patterns such as stripes which are generated by complex mechanisms. Gradual changes to these patterns then can only occur in specific directions in phenotype space since the mechanisms for generating them can only be changed in specific ways.

Other patterns that occur in more central regions of the continuum are likely to be the more common simple patterns. This is because they are likely to be generated by simple mechanisms that can be changed in more ways than complex mechanisms.
Figure 4.16: An illustration of how we imagine the distribution of patterns in pattern space. The dimensions are the concentration of gene expression in each cell. Here we assume only 2 dimensions since we can not visualise 32 dimensional space. Each point represents a gene expression pattern. (Top) Examples of patterns that correspond to the points are also shown. The dotted arrow represents how one can change from one pattern type gradually into another. A lobe is illustrated by the continuous blue line. (Bottom) showing examples of patterns falling into each group (black dashed line) using the pattern matching algorithm.
Grouping the patterns by pattern matching:

For the purposes of generating ‘Nearly Neutral Networks”, we generated arbitrary pattern groups based on more abstract features of the gene expression patterns. These pattern groups were chosen to be representative of the pattern space as a whole. Hence these groups include simple patterns such as gradients and more complex patterns such as stripes.

Types of patterns were identified subjectively by eye and then digitally encoded. For example a single stripe pattern may be described as to have 0 or low expression on the left hand-side, the expression goes up for some time, and then it goes down, and then it is 0 or low again. Using this type of approach patterns could be assigned to the group if they conformed to the digital pattern definition. This approach does not result in all of the patterns being assigned a group. Those patterns that are not assigned to a group are left out of the analysis.

The 10 distinct patterns that we defined may not represent all of the distinct patterns that are present in the data. Indeed there several are other patterns that occur that also could be defined digitally.

Summary:

- Pattern (phenotype) space based on simple gene expression values consists of two large continuum groups as determined using hierarchical clustering and force-directed algorithms.
- Pattern matching schemes using human feature extraction to define digital definitions of patterns allowed us to define 10 groups of gene expression patterns that cover 96% of patterns generated. These pattern groups are used to generate ‘Nearly Neutral Networks’ of gene network topologies.
5) Exploring the routes of evolution and pattern innovation.

In this chapter we explore the potential routes of evolution of patterning mechanisms through the concept of a neutral network (see below). Gene network topologies capable of producing particular phenotypes were found to form a neutral network of topologies (see results). The connectivity and shape of these neutral networks generating a particular pattern has important consequences for evolvability. These neutral networks are multi-dimensional shapes however and thus cannot be directly visualised. Therefore to get an idea of their shape, we need to abstract features. The abstraction of features of shape will be one of the main goals of this chapter. Another major goal of this chapter was to explore pattern innovation; how can any given pattern change into another. This question can be explored by analysing the routes between neutral networks.
**Introduction:**

*Exploring the NN concept for gene regulatory networks:*

The concept of a neutral network in the context of RNA folding has been useful to show that it is possible to traverse much of genotype space, without changing phenotype but so far there has been hardly any application of this concept to more complex biological systems such as multi-cellular pattern-formation. Recently Ciliberti (Ciliberti et al., 2007) extended the concept for the first time into gene networks.

In their system, the genotype is the gene regulatory network and the phenotype is the gene expression patterns “profiles” by simulation of these networks in single cells. By profiles, we mean the complement of gene activity states in that single cell. The model they used is a type of Boolean model and is described by the following equation:

\[
S_i(t + \tau) = \sigma \left[ \sum_{j=1}^{N} w_{ij} S_j(t) \right]
\]

Where \( S_i(t) \) is the state of the component \( i \) at time \( t \), \( \tau \) is a constant and \( \sigma \) is a Heaviside step function where \( S_i \) assumes a value of \( \pm 1 \). The interaction matrix \( w_{ij} \) describes the effect of the gene \( j \) on the gene \( i \) which can be 0, 1 or -1. Viable gene regulatory networks in this model are those that start from a pre-specified expression state \( S(0) \) and arrive at a pre-specified equilibrium state \( S_e \).

In these neutral networks, a single mutational event is the change from one gene network topology to another with one gene-gene interaction that is different (Or one hamming distance as described in chapter 2). These neutral networks were generated by sampling gene networks of a given size and in this sense are ‘incomplete’. They were able to show that the genotypes residing on these neutral networks were almost as far apart genotypically than if they had been picked at random from genotype space. This result suggested that networks of neutral paths percolated the entire sequence space. Furthermore, like the RNA example, they were able to show that
two networks with completely unrelated phenotypes could be found close together in genotype space.

Taken together, these two observations, (that it is possible to reach many different phenotypes with few mutations, and that it is possible to traverse most of genotype space on a neutral network) have led to a formulation of how evolution can occur in the presence of mutational robustness, two seemingly paradoxical features (Ciliberti et al., 2007; Wagner., 2008). This formulation involves a genetic architecture like that shown in figure 5.1. Only an architecture like that shown in the centre is compatible with these observations. Such architecture allows evolution “to explore the great diversity of new phenotypes needed to sift potential innovations through natural selection” (Ciliberti et al., 2007). The shape of a neutral network thus has important consequences for pattern innovation.

This work however has several limitations:

- As mentioned, the simulations take place in single cells. The effects of signalling between cells in a spatially extended system are not taken into account. Networks that are viable under the definition used in their work may not necessarily generate a gene expression pattern in space. Therefore this definition of viability is questionable in a patterning context.

- The regulation model used is Boolean. As described in the introduction chapter Boolean models can only accurately capture the features of a subclass of continuous systems. In particular, they are not suitable for the exploration of pattern forming gene regulatory networks. Furthermore this model is a specific kind of Boolean model where genes can take 1 or -1 states. With such model conditions a repressor can actually activate its target gene whilst an activator can repress its target gene (assuming this is the only input to that target gene). This is because according to the model the effect on a gene is the multiplication of the regulation with the state of the affecting gene. Thus a repressor (regulation of -1) with state -1 would result in the target gene being switched on whilst an activator (regulation of 1) with a state -1 would result in the target gene being switched off. Such features arouse questions as to whether or not the model is biologically reasonable.
Low robustness
Low innovation

High robustness
High innovation

High robustness
Low innovation

Figure 5.1: Genetic architectures and their relation to robustness and innovation (from Ciliberti et al., 2007). Genotypes are shown as balls. Those that are one mutant neighbours are linked by lines. Different phenotypes are illustrated by the different colours.
• The functions that a gene network must perform that they have chosen are arbitrary, that is they have analysed neutral networks for randomly picked gene expression profiles. Furthermore, by randomly picking gene expression profiles, one is likely to pick the more simple common profiles rather than those that perform a more complicated and potentially useful function. The results then that they obtain suggesting that these neutral networks are huge and span much of genotype space may represent the fact that the phenotype is trivial. Only a small part of the genotype may be functionally constrained, the rest being free to mutate in any way it wishes resulting in this apparently large span.

**Applying the NN concept to pattern forming gene regulatory networks:**

At the time of writing no attempts to explore NN’s for pattern forming gene regulatory networks have been published. So the question arises what can be said about NN’s of pattern forming gene regulatory networks based on the more abstract contexts of RNA sequence-structure relationship and single cellular Boolean gene regulatory networks? As has been mentioned, these simpler contexts have an important purpose “to sharpen the questions – perhaps even to understand what the questions are” (Fontana 2002). However we never know whether an observed result from studying a more abstract model is due to some “quirky” feature of that model and so would not hold for the more realistic model. Therefore wherever possible we should employ more realistic models and therefore in this project we use a model known to reflect real patterning in *Drosophila* (See chapter 3).

The Questions to be addressed regarding neutral networks of gene networks using a continuous spatial extended model include:

1) Do genotypes capable of producing a given phenotype form neutral networks?

2) If so, how big are the neutral networks and how much of genotype space do they span?

3) What are the shapes of these neutral networks and what consequences do these shapes have for evolvability?
The complete atlas of topologies can help to address these questions in the following ways:

1) Because we have simulated all topologies up to a given complexity and have explicit one hamming distance neighbour definitions of these topologies we can build neutral networks of topologies able to produce a given gene expression pattern (Assuming they exist). Due to the completeness of the atlas, there will be no gaps from a topological perspective assuming that the parameter sampling was high enough (See Appendix E).

2) As we have all of the topologies able to produce a given pattern (function) then we know exactly how big the neutral network is for a given number of genes. Furthermore, we can measure the average topological distances between topologies in the neutral network to determine how much of genotype space the neutral network spans.

3) Because we know all routes from any one topology (A) to any other (B), we can explore which are the optimal routes to get from A to B. Does the shortest route tend to be the most viable? A simple but apparently reasonable consideration based on mechanism suggests this to be true. Any given network embodies a certain mechanism (or collection of mechanisms) to achieve its function. The more topological changes are made to the network, the further its mechanism can diverge from the original one. If A and B are thought to use similar mechanisms, then the shortest route through the atlas will limit the topological divergence to a minimum. The resulting path will define the set of networks which are topologically closest to A and B (figure 5.2). Compared to this, a longer, indirect route through the atlas will include more divergent topologies which are presumably less likely to embody the same mechanisms. The shape of the NN could also have important consequences for these routes. For example the most direct route between some topologies might lie outside the NN.
Figure 5.2: Routes in topology space. Circles are topologies and lines are links between topologies one hamming distance apart (A and B). The most direct route (orange) which has a hamming distance of 3 is shown between two topologies (red circles). An indirect route with a hamming distance of 6 is also shown (red).
Nearly Neutral Networks (NNN’s) of gene networks with a continuous model:

We can address the question as to whether neutral networks exist for continuous models of gene regulation in a spatial context. Given the degeneracy of the genotype-phenotype mapping we would expect that they do. However continuous models of gene regulation do not generate discrete on-off type phenotypes and thus neutrality in this context is more difficult to define. In this analysis we will include those phenotypes that all fall within the same group as defined by the pattern matching algorithm described in chapter 4. Because the phenotypes produced by the different genotypes of the NN are not identical in this case but instead share the presence of some abstracted features, we refer to these NN’s as ‘Nearly Neutral Networks’ (NNN’s). In other words their fitness levels are similar, but not identical (Gavrilets 2004)

A second issue with using continuous model of gene regulation involves the nature of the mapping between genotype and phenotype. With Boolean models there is a one-to-one mapping since there are no adjustable parameters each topology can only have a single behaviour and thus phenotype. However with continuous models there is a one-to-many genotype phenotype mapping because the strength of regulation of genes on one another are adjustable parameters and their values determine the behaviour of the topology and thus the phenotype produced. Therefore with continuous models for any given topology the phenotype produced depends on the parameter values.

The NN concept suggests one can move from any topology to any other topology without changing the phenotype. However moving between two gene networks that are a single hamming distance apart in the same NNN may not necessarily represent a viable evolutionary change. The reason for this is the two topologies could be producing the phenotype using two very different regions of parameter space. In such a scenario, adding or removing a single gene-gene interaction will result in a genotype that cannot produce the phenotype of interest (figure 5.3 top). To change from the viable region of one parameter space to the viable region of the other, would require changing more than one parameter simultaneously.

A second possibility is that any one topology (B) can produce the same phenotype by two different mechanisms involving different regions of parameter space (figure 5.3
Figure 5.3: Not all topology changes represent viable steps in an evolutionary walk on a NNN. Two examples are given showing the parameter spaces of topologies. (Top) In the first, the two topologies (A and B) each work by a single but different mechanism (Viable parameter domains are shown in solid green). Changing A into B or vice versa then leads to a non-viable region of parameter space (Green dashed oval). (Bottom) Topology (B) can generate the gene expression pattern with two mechanisms (Solid green). Two other single hamming distance neighbour topologies (A and C) can also produce the pattern but using either one of these two mechanisms (Solid green). Changing from either A or C to B is viable. However changing from A to B to C or vice versa is only possible if the change between the viable regions of topology B involves a change to a single parameter.
Two one hamming distance neighbour topologies (A and C) each function to produce the phenotype with only one of these two mechanisms, which is different for each. Under this scenario it may be a viable evolutionary step to change from A to B or B to C but to change from one viable parameter region of B to the other may require more than one simultaneous parameter change and thus not be viable.

If NNN’s are built only including those links between topologies one hamming distance apart and producing the phenotype using at least a partially overlapping parameter space then many links between topologies one hamming distance apart may be lost. The loss of these links may result in the break-up of the single connected NNN into multiple disconnected NNN’s. For identified neutral networks, we explored whether this phenomenon was common and whether it did indeed lead to the breakup of neutral networks identified using topology alone.

Neutralism and innovation: Crossing between neutral networks

A debate that has been ongoing since the time of Darwin is whether evolution has proceeded gradually or has been punctuated by sudden transitions at the level of phenotype (Gould and Eldredge 1993). Both of these modes of transition however are gradual at the level of genotype (Fontana and Schuster 1998; Wagner, 2005b). Therefore the question of how evolution can change one phenotype into another, amounts to the question of how one can cross from one neutral network to another? This throws up further questions such as; Can one cross from any neutral network to another directly or indirectly? If so how likely is it? What are the routes?

An interesting real world example of crossing between NN’s has been shown for rybozyme evolution. One group has shown that two ribozymes with totally different tertiary structure and different catalytic activity can inhabit adjacent NN’s (Schultes and Bartel 2000). These two ribozymes could be converted into one another by a series of point mutations. Most of these mutations are neutral, that is they do not reduce catalytic activity. Intermediates at the intersection between the NN’s carry both functions but with reduced activities. These intermediates however are more than 40 mutations away from either of the original ribozymes. The region of reduced activity spans approximately 5 mutations over the intersection.
Surprisingly, although this phenomenon has been explored for this ribozyme example, computer exploration of transitions between neutral networks using gene regulation models has not been done. Because we can build neutral networks for multiple different defined phenotypes, we can explore the routes between them; that is the overlap of topologies between these neutral networks and any single mutation events that cross between them. Furthermore, as we have all topologies of gene network up to a given complexity we can map all these transitions and where they occur on a neutral network.

Our approach:

We set out to address the questions described earlier as to whether NNN’s exist for patterning gene networks with a continuous gene regulation model. And if so what is their size and shape and what consequences does this have for the evolution of patterning mechanisms. We attempted to avoid NNN triviality by specifically including phenotypes in our analysis that were more complex than simple gradients for example.

After finding that NNN’s of gene networks capable of pattern formation do exist, we started exploring the shapes of these NNN’s by abstracting features and exploring the potential mutational routes of innovation from one NNN to another.
Methods and Results:

We defined 10 distinct patterns that resulted from the simulation of the complete atlas of 3 gene topologies. Those patterns and a list of their features are described in table 4.17 in chapter 4.

Can we find nearly neutral networks? Determining the number of topologies in the largest connected component:

To determine the number of topologies in the largest connected component of genotype space able produce a particular pattern we used what we called a 'catch' algorithm which is illustrated in figure 5.4. The catch algorithm works according to the following pseudocode:

Set maxconnectedcomponent = 0
Foreach topology capable of producing the pattern
    Put topology in the catch
    Set catchsize = 1
    Do
        Set numinnewcatch = 0
        Foreach topology in the catch
            Ask if any one hamming distance neighbours are capable of producing the pattern and have never been in the catch before.
            If yes, put the neighbour in the newcatch, numinnewcatch++
            Clear catch
            Copy newcatch to catch
            Catchsize = catchsize + numinnewcatch
        Until numinnewcatch equals 0
        Clear catch
        If catchsize > maxconnectedcomponent
            Set maxconnectedcomponent = catchsize
Figure 5.4: The catch function. Circles are the topologies of the neutral network and lines are links between topologies that are one hamming distance apart. The figure illustrates starting the catch from the red topology. The topologies in the new catch array at the different loops of the simulations are illustrated by the different colours. This function is performed starting from each different topology i.e. each topology is set to be red.
For the rest of this chapter the largest connected component will be referred to as a nearly neutral network (NNN). Table 5.5 shows the percentage of topologies in the NNN for each of the pattern types explored in this analysis. For almost all pattern types, over 90% of the topologies form one large NNN. A clear exception to this is the NN for the stripes pattern where only half of the topologies are in the NNN. In order to check the statistical significance of these results, we performed permutation tests.

These permutation tests were performed using randomly picked topologies from the raw complete topology atlas. For each number of randomly sampled topologies (100, 200, 400, 800, 1600 and 3200), we asked what was the largest NNN formed. We did this 10 times for each number of topologies and averaged the result. The results of the permutation tests are shown in figure 5.6. This shows that any sample of topologies above approximately 1000-1,500 topologies will form one large NNN by chance. Our larger NNN’s that we have found then would be expected by chance. However the smaller NNN’s such as the stripes or right handed-gradient show that a single large NNN would not be expected by chance. Even with the smallest percentage of topologies in the NNN (52% for stripes) with 62 topologies only a few percent would be expected to form the largest NNN.

We selected 5 of these patterns to perform further analysis on; the left-handed gradient, the double-peak, the single stripe, the dip and the stripes patterns. We chose these 5 patterns because we deemed them representative of the whole group. Thus we chose some with large NNN’s, some with small. We also chose some that we obviously formed with more complex dynamics (such as the stripes pattern) and some that clearly were not (such as the left-handed gradient).
<table>
<thead>
<tr>
<th>Pattern Group</th>
<th>Topologies (in NNN/total)</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single stripe</td>
<td>347/359</td>
<td>97%</td>
</tr>
<tr>
<td>Stripes</td>
<td>32/62</td>
<td>52%</td>
</tr>
<tr>
<td>Double peak</td>
<td>43/45</td>
<td>96%</td>
</tr>
<tr>
<td>Left-handed gradient</td>
<td>1888/1892</td>
<td>99.8%</td>
</tr>
<tr>
<td>Right-handed arc</td>
<td>41/45</td>
<td>91%</td>
</tr>
<tr>
<td>Dip</td>
<td>1081/1083</td>
<td>99.8%</td>
</tr>
<tr>
<td>Gradient and threshold</td>
<td>1842/1852</td>
<td>99.5%</td>
</tr>
<tr>
<td>Left-handed threshold</td>
<td>3016/3019</td>
<td>99.9%</td>
</tr>
<tr>
<td>Right-handed threshold</td>
<td>3096/3096</td>
<td>100%</td>
</tr>
<tr>
<td>Left-handed gradient and peak</td>
<td>3333/3333</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 5.5: The number of topologies in the NNN of the 10 different pattern types defined for pattern matching. The ‘total’ is the total number of topologies found to produce the given pattern. The percentage is that of the number of topologies in the largest NNN compared to the total number of topologies able to produce that pattern.
Figure 5.6: Permutation tests. the percentage of topologies in the largest connected component for randomly selected sets of topologies. The line is the average of 10 random samples.
What is the shape of these nearly neutral networks?

It is difficult to measure the shape of a highly multidimensional complex graph. However we wanted to gain some insight into this shape since it has important consequences for pattern innovation as described in the introduction. The shape of the NNN’s was analysed using the topology internalness and the path length distribution. The internalness measure is a measure of how internal a topology is in the NNN. It is calculated for each topology by the following equation:

\[
\text{Internalness} = \frac{\text{Number of one hamming distance neighbours in NNN}}{\text{Total number of one hamming distance neighbours in raw atlas}}
\]

This internalness value can range between 0 and 1. A topology with a internalness value of 0 would be completely isolated whilst a topology with a internalness value of 1 would have no direct links to the rest of the atlas that does not form part of the NNN. This internalness value calculated for all topologies and a histogram built of these values. The value for each topology is essentially its mutational robustness since this value states the likely-hood of it losing the phenotype.

The path length between topology A and B is the number of single hamming distance steps needed to go from topology A to topology B in the NNN only via other topologies in the NNN. This value is worked out for all pair-wise combinations of topologies in the NNN and the distribution plotted on a histogram.

The algorithm uses a catch method, which is employed for each of the topologies, like that described to determine the topologies in the NNN (figure 5.4). The only difference here is that a variable called pathlength is set to equal 1 when the catch is started for a new topology. Each time the catch is updated, the pathlength variable is incremented by 1. Every time a new topology is found during the catch, the distance between the topology that seeded the catch and the new topology can be recorded as the current pathlength. We measured the path length distribution for all pairs of topologies in the NNN’s for the 5 patterns.
Histograms for the internalness index and path length distributions are shown in figure 5.7 and 5.8 (left) respectively. For the larger NNN’s (Left-handed gradient, single stripe and dip) there is a broad distribution of internalness. The left-handed gradient NNN contains a few very internal topologies. Both the stripes and the double peak patterns show a maximal internalness score of round about 0.35 and are similar sized NNN’s. However the distribution is skewed towards smaller values in the case of the stripes and to higher values in the case of the double peak. In the pathlength histograms, the stripes and the single stripe NNN’s have a broad pathlength distribution whilst the others have a more narrow distribution. The maximum pathlength that can be seen for the single stripe NNN is 15 (red arrow). The consequences of these distributions will be addressed in the discussion.

The maximal number of topological changes needed to turn any topology into any other without any constraints is 9 since the interaction matrix has 9 positions and this would involve changing all of those positions. This lead directly to the questions of (a) how could the actual minimal path length between two topologies in the NNN be longer than that seen in the raw atlas and (b) to what extent is this phenomenon apparent?

*Routes for evolution: The existence of ‘loop holes’:*

The first of these questions can be addressed theoretically. Between any two topologies of a NNN there exists a direct route through genotype space. For some pairs of topologies however it seems that a block exists preventing this short direct mutational path; a non-viable block (figure 5.9). Instead however it is possible to travel around this block via a longer but viable route. Therefore we called this structure a “loop hole”.

In order to explore the prevalence of these loop holes we compared the difference between topological distances in the NNN’s to that in the raw atlases for every pairwise combination of topologies in the NNN’s. The path length difference between two topologies A and B can simply be calculated as the following:

\[
\text{path length difference} = \text{dist}_{AB} \text{ in } \text{NNN} - \text{dist}_{AB} \text{ in } \text{raw atlas}
\]
Figure 5.7: Exploring the shape of the NNN’s using the internalness measure. Histograms are shown for the number of topologies of the NNN with the given internalness score.
Figure 5.8: Path length distributions of neutral networks and compared to the raw atlas
Figure 5.9: Loop holes. Forbidden direct routes when neutral networks have an irregular shape. Circles are topologies and lines are links between topologies one hamming distance apart. Connected circles are part of the neutral network whilst those that are non-connected are those in the complete topology atlas that are not in the neutral network. The direct route (dashed orange) between two topologies (red circles) which has a hamming distance of 4 is not functional. The loop hole (red line) with a hamming distance of 6 is however viable.
Any path length difference greater than 0 is a loop hole. Histograms of the path length differences are shown for the NNN’s in figure 5.8 (right). Even simple patterns such as the left-handed gradient show some loop holes, though they are much less pronounced than in the NNN for the stripes and the single stripe. Those NNN’s that have the wide distribution of path lengths are also those with more loop holes (stripes and single stripe).

**Sampling effects on loop holes:**
These loop holes could potentially arise due to sampling effects. In other words the loop hole is seen simply because a topology in the direct path between two pattern generating topologies was not sampled enough. The question as to what extent parameter sampling has on these loop holes was addressed by sub-sampling the NNN. This analysis was performed on the single stripe NNN as this is the largest NNN where the loop-hole effects are prevalent. The NNN was regenerated for different thresholds of robustness as defined by the fraction of parameter space. Any topology generating the pattern with robustness below that threshold was excluded from the NNN. The path length analysis was then performed like before on these sub-sampled NNN’s. A scatter graph showing the average loop hole length increase compared to the robustness increase is shown in figure 5.10. There is a weak inverse correlation between the robustness increase and the loop hole length increase (Pearson’s coefficient of -0.06). Therefore loop holes that make a large difference to robustness are short, whilst those which make a small difference are generally longer. Those longer loop holes are likely to be artefacts of sampling whilst the shorter ones and likely to be true features of the NNN.

**Innovation – jumping from one Neutral network to another:**
As a first step to exploring the routes of evolution between NNN’s we measured the topology overlap between all pair-wise combinations of NNN’s using the Jaccard index. The topology overlap is the number of topologies that are shared between NNN’s generating different gene expression patterns. The overlap in the topologies of each neutral network was worked out using an algorithm that compared all of the
Figure 5.10: The relationship between the length of loop holes and the robustness threshold for the nearly neutral network generating the stripe of expression.
topologies in one NNN to all of the topologies in another NNN. The overlap score was defined as:

\[
\text{Overlap (Jaccard index)} = \frac{\text{Matches}}{(\text{Total topols in NNN 1} + \text{Total topols in NNN 2}) - \text{Matches}}
\]

It is also possible to cross from one NNN to another by a single regulatory change. We defined such a transition as a 'bridge'. For each topology in one NNN, the algorithm asks if the one hamming distance neighbours are in the other NNN. A bridge constitutes each case where this is true. Bridges were not included where either end of that bridge was an overlap topology.

The results of this analysis are summarised in the tables shown in figure 5.11. There is at least some topology overlap between all pairs of NNN’s, though this quantity can vary greatly. Interestingly one pair of NNN’s has no bridges but all of the topologies from one of the two NNN’s are also in the other NNN.

**Parameter space overlap:**

The overlap in parameter space between two topologies one hamming distance away from each other in the same NNN could be determined using many different methods. The approach that we took we called 'parameter swapping'. Parameter swapping is a measure of functional overlap.

This analysis was performed on the NNN of topologies generating the single stripe pattern. For each pair-wise combination of one hamming distance neighbour topologies in the NNN, parameter swapping analysis was performed. Take say neighbouring topologies A and B. The parameter sets that could produce the gene expression pattern of interest are collected for both A and B. For each pair-wise combination of viable parameter sets of A and B the swap analysis was performed. One of the topologies will have one more gene-gene interactions than the other. This larger topology will be simulated without that extra interaction which we termed a 'swap'. If the resulting dynamic conforms to the digital definition as used for this gene expression pattern in the pattern matching, then this swap was successful and
<table>
<thead>
<tr>
<th></th>
<th>Double peak</th>
<th>Left-handed gradient</th>
<th>Dip</th>
<th>Single Stripe</th>
<th>Stripes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double peak</td>
<td>X</td>
<td>62% (28/45)</td>
<td>100% (45/45)</td>
<td>80% (36/45)</td>
<td>18% (8/45)</td>
</tr>
<tr>
<td>Left-handed gradient</td>
<td>1.5% (28/1892)</td>
<td>X</td>
<td>29% (544/1892)</td>
<td>7.0% (133/1892)</td>
<td>2.4% (45/1892)</td>
</tr>
<tr>
<td>Dip</td>
<td>4.2% (45/1083)</td>
<td>50% (544/1083)</td>
<td>X</td>
<td>20% (222/1083)</td>
<td>5.3% (57/1083)</td>
</tr>
<tr>
<td>Single Stripe</td>
<td>10% (36/359)</td>
<td>37% (133/359)</td>
<td>62% (222/359)</td>
<td>X</td>
<td>3.3% (12/359)</td>
</tr>
<tr>
<td>Stripes</td>
<td>13% (8/62)</td>
<td>73% (45/62)</td>
<td>92% (57/62)</td>
<td>19% (12/62)</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Double peak</th>
<th>Left-handed gradient</th>
<th>Dip</th>
<th>Single Stripe</th>
<th>Stripes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double peak</td>
<td>X</td>
<td>44</td>
<td>0</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Left-handed gradient</td>
<td>44</td>
<td>X</td>
<td>465</td>
<td>314</td>
<td>74</td>
</tr>
<tr>
<td>Dip</td>
<td>0</td>
<td>465</td>
<td>X</td>
<td>243</td>
<td>25</td>
</tr>
<tr>
<td>Single Stripe</td>
<td>13</td>
<td>314</td>
<td>243</td>
<td>X</td>
<td>43</td>
</tr>
<tr>
<td>Stripes</td>
<td>14</td>
<td>74</td>
<td>25</td>
<td>43</td>
<td>X</td>
</tr>
</tbody>
</table>

**Figure 5.11:** (Top) Topology overlap between different NNN's. Showing percentage of topologies of the NNN defined by the row (denominator in brackets) that overlap with the NNN defined by the column. The exact number of overlaps is given by the numerator in brackets. (Bottom) Bridges between the NNN's. Showing the ratio of bridges:topologies of the NNN defined by the row (denominator in brackets) that link to NNN defined by the
these parameters are considered to be within an overlapping volume. The final overlap was defined as the successful swaps divided by the total swaps made. If there is no overlap, then the link between A and B is removed from the NNN. When this analysis is carried out between all neighbouring topologies in the NNN we find that 82% of topology – topology links remain. We tested how big the largest connected group of topologies was using the catch function described earlier for testing whether NNN’s exist with the neighbour definitions that only included these 82% viable links. The largest resulting NNN contains the same number of topologies as it did before indicating that it has not been broken up into multiple components.
Discussion:

The existence of Nearly Neutral Networks:
The vast majority of topologies capable of generating a particular pattern form one NNN. We have shown that this would not be expected by chance by permutation analysis. Even for the stripes pattern with the smallest ratio of topologies in the main NNN, the fraction is substantially more than would be expected by chance. The existence of NNN’s of gene networks capable of pattern formation suggests that it is possible to access many new pattern innovations without non-functional intermediates.

The shape of Nearly Neutral Networks: ‘The existence of loop holes’.
The shapes of these NNN’s have important consequences for the robustness of patterning mechanisms and the innovation of pattern (Ciliberti et al., 2007). We have attempted to gain some insight into the shapes of these NNN’s by measuring the distribution of topology internalness and path length. The NNN’s for all of the different patterns show a heterogeneous distribution of the internalness of their topologies. A similar distribution was found for single cell networks that could viably produce a stable gene expression pattern (Ciliberti et al., 2007b). Though in their study the distribution differed slightly in that it was skewed towards the lower internalness values.
An interesting comparison is between the NNN’s producing the stripes and the double peak. These NNN’s are of similar size but showed differences in their internalness and path length distributions. The stripes NNN had an internalness distribution skewed to the left and a broad path length distribution whilst the double peak NNN had an internalness distribution skewed to the right (up to the maximal value of 0.35) and a more narrow path length distribution. These results suggest that the double peak NNN has a more compact shape whilst the stripes NNN is more sparse with a higher surface area to volume ratio. The number of loop holes also reflect this difference. The stripes NNN has many more loop holes than the double peak NNN. Loop holes correspond to the local irregularities in a sparse NNN.
The probable mechanistic reason behind the difference in the shapes of the NNN’s is that the mechanism(s) producing the stripes is more complex than the mechanism(s) producing the double peak. Hence there are more constricting constraints for the mechanism(s) producing the stripes. These conflicting constraints will manifests themselves as irregularities in the shape of the NNN.

Loop holes are a manifestation of how development constrains evolution since a more optimal path for evolution cannot be taken as it results in a non-viable gene network or too fragile function. To our knowledge, this is the first time that such a phenomenon has been described for NN’s.

A thorough investigation into the mechanistic features responsible for these loop holes will be performed in the future. An interesting feature that we have observed by looking at the topology changes in the paths that are longer in the NNN as compared to the raw atlas is that many seem to involve the gain and loss of an auto-inhibition. This suggests that the topology has become ‘out of tune’ with the mechanism. That is for example a direct path may result in the generation of a network with too much positive activation, such that the viable parameter space is reduced.

These loops holes appeared to be much more prevalent in the NNN’s for the single stripe and stripes pattern. Presumably this reflects the fact that these patterns are generated by complex or multiple mechanisms. More complex mechanisms would be expected to have more conflicting constraints meaning that the underlying gene network can only be changed in certain ways resulting in loops holes.

In summary, the existence of loop holes is good evidence that these NNN’s have an irregular shape. Overall mutational robustness is less in more irregular shapes as compared to more regular shapes. A consequence of this finding is that gene expression patterns produced by more complex or multiple mechanisms with conflicting constraints are likely to be less mutationally robust.

**Innovation of phenotype: moving between neutral networks:**

The NNN’s that we have generated here shows in greater depth how the phenotype innovation can occur. There are two potential ways of crossing between NNN’s without loss of function; via topology overlaps or via bridges.
Topology overlap involves a change in a single parameter that corresponds to a single mutation event but with no change in topology. This parameter change alters the dynamic of the topology so that it produces a different time stable and noise robust pattern (Ingolia, 2004). This type of transition can only occur for topologies that are in both NNN’s. For this reason we measured the number of topologies that were common between each pair-wise combination of NNN’s.

The bridge route involves a mutation that adds or removes a gene-gene interaction resulting in a change of topology. This is the type of evolutionary change that would require a new protein-protein interaction, or some other new interaction in the regulatory circuit. We called such a transition point a ‘bridge’ between NNN’s. These bridges are prevalent; there are at least some bridges between all pairs of neutral networks except the dip and double peak pattern. The lack of any bridges between the dip and the double peak NNN can be explained by the fact that the mechanism that generates the double peak, requires the generation of a dip (not vice-versa). Hence all topologies that can generate a double peak, can also generate a dip. Therefore there are no bridges between the NNN’s as there is total topology overlap between the two NNN’s.

The fact that it is possible to move from one NNN to any other NNN suggests that there are many routes for innovation between NNN’s. This finding opens up further questions such as; where exactly on the NNN are these overlaps or bridges located? Are they concentrated in particular areas? Or are they spread throughout the NNN? Such features would be expected to affect the dynamics of NN exploration and innovation (Fontana and Schuster 2000; van Nimwegen et al., 1999). For example it was found that populations of mutating RNA do not move over a NN at random, but concentrate at highly connected regions (more internal) of the NN. Therefore innovation would be facilitated if bridges and topology overlaps were found at more connected regions of the NNN. In the future we will explore exactly these questions.

**Parameter overlap:**

Although topologies may be one hamming distance neighbours in topology space, changing from one to the other may not represent a viable evolutionary step. It may be that one topology will perform the patterning mechanism(s) for generating a given
gene expression pattern using a completely different set of parameters to its neighbour. Therefore to check whether topologies are true functional neighbours we needed to check whether there was an overlap in their functional parameter sets. The parameter spaces can only overlap if we do not consider the different interaction between the two neighbouring topologies. This different interaction is the mutation that would need to occur in order to change one into the other. The functional overlap in parameter space was measured by swapping parameters of the neighbouring topologies and asking whether the network could achieve the same function with the parameters of its neighbours.

When this analysis was performed the majority of topology – topology links in the NNN responsible for the single stripe pattern were found to have at least some degree of parameter overlap (82%). The NNN remains the same size as before and does not break into multiple components.

A further complication is that it is possible that any one topology from an NNN may include multiple non-connected parameter domains that are responsible for generating a pattern as discussed in the introduction. Each of these parameter domains may correspond to a different patterning mechanism. For evolution to move from one parameter domain to the other, may involve a change in more than one parameter and thus may not be a viable evolutionary step. Taking this effect into account could result in the breaking up of the neutral network into multiple components. We are currently analysing this effect in our NNN’s capable of pattern formation.

Summary:

• Patterning gene networks tend to form a NNN’s.
• Development constrains evolution leading to loop holes in functionally viable genotype space.
• NNN’s of gene networks capable of pattern formation may break up into multiple components when the existence of multiple functional parameter domains is taken into account.
6) Exploring mechanism.

In this chapter we investigate patterning mechanisms. The general questions that can be asked about such mechanisms include: Are they intuitive? How many are there? How distinct are they? Are there hybrid mechanisms? We introduce the concept of a complexity landscape in order to aid the exploration of these questions. We built a complexity landscape of the gene networks responsible for the single stripe gene expression pattern as a test case.
**Introduction:**

**Encoding pattern forming mechanisms:**

This question put simply asks “how do things work?”. In our specific context mechanism describes how a particular gene expression pattern is generated (the goal). The mechanism explains the order with which genes change their expression states in different cells in order to achieve this goal. That order is determined by the causal relationships between the genes.

In the introduction chapter we described some key general strategies by which pattern formation has been suggested to occur such as the gradient threshold and Turing strategy and described how those strategies are carried out by gene networks. The problem is that we do not know how these key general strategies are related to each other. Furthermore how can we evolve between these mechanisms? Are these all the mechanisms of pattern formation that exist?

Our goal is to try and bring these ideas under the same framework. To do this we need some kind of metric for measuring how different two mechanisms are. This in turn requires a method for parameterizing or encoding mechanism.

We can consider encoding to be the process of identifying the independent parameters of an entity such that any entity is unambiguously described by the set of those parameters in a non-redundant fashion.

By encoding, a space of possible entities is generated. Each individual entity has a location in this space of possibilities. The number of dimensions in this space is the number of independent parameters used to encode. For any one entity, the value of each of it’s parameters determines where it lies along each dimension of this space.

One criterion of a successful set of parameters would be if by counting through all combinations of parameter, we successfully enumerate all possible entities.

One of the key questions of encoding is choosing the number of parameters to use. There are two conflicting constraints in this argument. On the one hand, if we take two few parameters two entities that we consider different could be put in the same category. On the other hand, if we take too many parameters we introduce
redundancy, meaning that the same set of entities could be unambiguously defined with a smaller set of parameters.

Other issues with encoding include how discrete are the parameters and how to identify those that are independent. Having discrete parameters makes enumeration much simpler, since with discrete entities, it is simple to count through the possibilities.

**Encoding genotype and phenotype:**
Genotype is easy to encode since the parameters are independent and discrete. In this case the parameters are the gene-gene interactions. The number of parameters defining the space is \( N^2 \) where \( N \) is the number of genes. The location of a topology in the space of possibilities along each dimension is defined by whether there is no interaction, an activation or a repression in that specific gene interaction position. Hence each dimension has 3 discrete values.

The picture is slightly complicated by the issues of labelled versus unlabelled topologies described in chapter 2. This complication is why the space of possibilities corresponds to a stripped down hypercube rather than a full one. The result of encoding genotype is the complete atlas of topologies described in chapter 2. This is an optimal parameterization of the topology space since the atlas only includes one version of each topology (unique unlabelled topologies).

Phenotype is less easy to encode. Independent parameters in this case are not so obvious. In chapter 4 we explored phenotype space by comparing phenotypes using the gene expression states in each cell as the parameters. These gene expression states are not independent from one another since cells can signal to one another. The parameters of a well-encoded system should be independent from one another in order to reduce redundancy and improve clarity. Hence (if our goal was to encode phenotype, then) more abstract features might result in a more efficient encoding of the space of possibilities.

**Our Goal: Encoding mechanism**
Mechanism as already mentioned is not as tangible an entity as genotype or phenotype and thus choosing the parameters to abstract and parameterize the space
of possibilities is even more difficult. One of the key goals of this chapter was therefore to identify a way of encoding mechanism.

Mechanism can be considered the set of all causal relations in a system responsible for a particular goal. The causal relations in a gene network topology are the regulatory interactions between genes. Therefore the topology of a gene network may contain a lot of the necessary information for encoding a mechanism. However the strengths of these regulatory interactions can change, potentially giving any one topology multiple behaviours. Therefore topology alone is not enough to unambiguously define a mechanism since a single topology could map to multiple mechanisms.

If anyone topology is achieving a particular goal by some particular mechanism, then it may not be that all of the interactions of that topology are necessary for that mechanism. It should be possible then to remove these superfluous gene-gene interactions without affecting this particular mechanism. Once all of the superfluous interactions have been removed, we are left with a "core" gene network topology that contains only the necessary interactions (causal relationships) for that particular mechanism. We hypothesized that these "core" topologies are where mechanisms map to and that each mechanism can be represented by a core topology. By representing a mechanism by the core topology capable of producing it, we might arrive at a convenient way of encoding mechanism, because topology itself is easily encoded.

A similar approach had been taken in the past to identify the core topologies responsible for generating a particular gene expression pattern (Salazar-Ciudad et al., 2000). This approach involved sampling larger gene networks and simulating them. When networks were found that could perform some kind of patterning function, interactions were removed and the network re-simulated to ask whether that interaction was necessary for the patterning function. In this way this group identified core gene network modules responsible for the patterning function (figure 6.1).

We however have the complete atlas of all possible topologies. Having every topology allows us to conveniently find the simplest core gene network topologies
Figure 6.1: An illustration of the approach taken by Salazar-Ciudad et al. Genes are represented by \( g \) and hormones by \( h \). A hormone is a gene whose product can diffuse. Activation is represent by arrows and inhibition by 'dead-end' lines. (Top) A large network (left) is found to generate a particular gene expression (right). (Bottom) After rounds of removing individual interactions and re-simulating a minimal topology (left) is found that can still generate the gene expression pattern (right). Right-hand images from Salazar-Ciudad et al., 2000.
through the idea of a complexity landscape. These complexity landscapes can then be used to test the hypothesis that core topology can be used to encode mechanism.

**Complexity landscapes:**
The usefulness of the metaphor of a landscape for understanding the process of evolution has been described in the introduction chapter. That metaphor however is useful in any situation where one wants to know how to get from point to point in the space of possibilities when there are constraints. The fitness landscapes described in the introduction uses the fitness of a genotypic combination as the constraining factor. These landscapes are generated by plotting a function in space. The encoded parameters of the genotype are the inputs to this function whilst the fitness is the resulting output.

We developed the idea of a complexity landscape to aid the exploration of the space of possible mechanisms. The idea of a complexity landscape involves plotting a function that calculates the complexity of a topology (we have chosen the number of gene-gene interactions as our measure of complexity). If we collect from the complete topology atlas the topologies that are capable of producing a particular gene expression pattern, then we can build a complexity landscape. Single evolutionary steps in this landscape are represented by links between topologies. This is because these links represents the addition or removal of a single gene-gene interaction such that one topology is changed into another. The landscape has to be spaced in such a way as to reduce overlapping links as much as possible. Once appropriately spaced, the core topologies that are responsible for a particular mechanism for producing this gene expression pattern should be seen as the topologies that are the local minima in these landscapes.

**What is the topology of these landscapes?**
The topology of these complexity landscapes has important consequences for how we think about patterning mechanisms. Do these landscapes have multiple local minima or a single local minima? If there are multiple minima, then this suggests that there are multiple mechanisms for producing this particular pattern. More
importantly this topological feature would suggest that these mechanisms are also relatively distinct from one another.

**Hybrid mechanisms**

If there are multiple mechanisms then we can start to explore what happens in the transition between these two mechanisms? Is there a smooth gradual transition or is there a sudden change in behaviour of the gene network? Furthermore what happens if a gene network topology exists that has the topological features of two different core topologies responsible for different mechanisms? Does the network simply perform the two mechanisms using different regions of parameter space, or can some kind of hybrid exist between the two mechanisms?

**Exploring robustness:**

As we mentioned in the introduction chapter, an important aspect of a network function is robustness. For any core mechanism(s) found, we can explore how robust they are to parameter changes.

A common measure of parameter robustness described in the literature is the fraction of parameter space able to achieve the desired function (Von dassow *et al.*, 2000; Meir *et al.*, 2002). This is a measure of the volume of parameter space able to achieve that particular patterning function. However this is not necessarily a true reflection of actual robustness since mechanisms with the same fractions can have domains of parameters that are responsible for the mechanism with very different shapes and connectivity (figure 6.2).

A better measure of robustness is the surface area to volume ratio of these parameter domains where a low surface area to volume ratio indicates high robustness. This is because it takes into account these features of parameter space and is a measure of the chance of a parameter change crossing out of the successful parameter domain and into a region which cannot achieve the function. However, surface area to volume ratio is difficult to measure in multidimensional space mainly because we sample only a subset of the entire space and thus some way is needed to actually define the domain in the first place; where is “in” and where is “out”. This requires algorithms such as convex hull algorithms that are computationally expensive,
Figure 6.2: The shape of robustness. Three different functional parameter domains with the same fraction of parameter space. (Left) A robust functional parameter domain with a low surface area to volume ratio. (Middle) A less robust functional parameter domain with a higher surface area to volume ratio. (Right) Multiple disconnected and less robust functional parameter domains.
especially in multidimensional space and may not be accurate without enough sampling.

An alternative approach is to measure the extent to which single parameter variation is tolerable. This involves changing each of the parameters independently whilst holding the others fixed. How much each of the individual parameters can be changed without changing the behaviour of the network is a measure of robustness to parameter change. This approach must be performed for multiple parameter sets as the results for any one parameter set may be simply due the specific location of that parameter set in the functional parameter domain. This approach can be thought of as an indirect measure of the surface area to volume ratio since it describes the likelihood of crossing from the patterning domain into the non-patterning domain.

**Our Approach:**

We built a complexity landscape for topologies capable of generating a single stripe of gene expression. We chose this phenotype because it is not so simple that we would be swamped with topologies capable of generate it and not so complex that very few topologies would be able to generate it.

We identified core topologies in this landscape and analysed how they were functioning. We analysed the robustness for each of these core topologies. We looked for hybrid topologies in this landscape and analysed how they were functioning to explore what happens at the transition between mechanisms.
Methods:

**Generating a complexity landscape:**

Once topologies have been collected that could generate the single stripe gene expression pattern, a complexity landscape can be built. Complexity landscapes are built by an algorithm that assigns y coordinates based on complexity. Like in the NNN’s from chapter 5, topologies one hamming distance apart are connected by a “link”. These links should cross as little as possible in order to show large scale features of the shape of the landscape. Crossing links can be avoided by grouping the topologies that are one hamming distance neighbours as we rise from the lowest until the highest complexity. The position in the x-axis then depends on that grouping. The approach that we took to group the topologies and assign x-coordinates is illustrated in figure 6.3. The following pseudocode describes how the algorithm generates the group structure:

Place least complex topologies in their own groups
If these topologies are neighbours of each other
    Merge their groups
Record the group that these topologies are in (thus first assigned to)
Do
    Find the topologies with the next lowest complexity
    If these topologies are neighbours of topologies already placed in groups
        Place in these groups
    If not place in their own groups
    If these topologies are neighbours of each other
        Merge their groups, record the merges
        Record the group that these topologies are in (thus first assigned to)
Until complexity = the highest complexity of the topologies

Once this part of the algorithm has finished, there should be just one group if all of the topologies are connected in one large NNN, or there could be separate groups for multiple smaller NNN’s.
Figure 6.3: Building the complexity landscape. (1) The input data is the set of topologies in the landscape. In this example there are 6 (A-f). The complexity and neighbours of these topologies is calculated. (2) The group structure is calculated from lowest to highest complexity. (3) Coordinates are assigned to groups and then to topologies based on the first group they were assigned to.
The next part of the algorithm assigns x coordinates to the groups. It does this by working from the highest to the lowest complexity. At the highest level of complexity, the algorithm will give all the groups that are seen an arbitrary value incremented for each group (0, 1, 2..etc). At the next level of complexity, the algorithm asks if there are any new groups that have not yet been assigned an x coordinate. If there is a new group, then this group must have been lost at a merge event at this level of complexity when the group structure was built. The algorithm finds the group that the new group was merged into and gives the new group an x coordinate such that it is placed adjacent to the group it was merged into. The other groups of the cluster then have to be shifted to the left or right to make space for this new group.

Once all groups have been assigned an x coordinate, the algorithm then finally assigns an x coordinate to each topology by reading which group the topology was first assigned to when the group structure was made and giving the topology the x coordinate of this group.

**Identifying core topologies:**

In order to facilitate the exploration of the landscape, an extra tool was generated for the complexity landscape view of the TclTK graphic user interface. This is a manual tool requiring the user to interact with the GUI described in appendix A. This tool changes the way individual topologies are moved around, so that if any topology is moved around, so are other topologies that conform to the following conditions: They are more complex than the moved topology and they can be reached from the moved topology via a path that never reduces in complexity. The selected topology moves like normal, but the other topologies move by a percentage of the amount the selected topology was moved. In this way, moving a topology that occurs at a local minima in the atlas, acts to move the whole inverted peak (figure 6.4). This tool can be useful for organising the topologies and in particular for identifying peaks behind other peaks, since this is a 2 dimensional space being used to represent a much higher dimensional space these kinds of overlaps are likely to happen.
Figure 6.4: Moving local minima. (Left) Two minimas are overlapping. (Middle) The minimal topology (red) of one of the minima is moved to the right. (Right) The other topologies that can be reached from a path that never decreases in complexity from the minimal topology are moved by a percentage of the distance the minimal topology was moved by (orange).
Exploring how core topologies are functioning:
Once the core topology(s) have been found in the complexity atlas for a given pattern, the exact mechanism of how they are working can be ascertained. Several extra tools were added to the GUI to help address this problem (See appendix A). To investigate how a mechanism is working GUI tools are used to find a set of parameters for a selected topology to achieve a given pattern function. These parameters are stored in a “genenet” file. The parameters stored in the genenet file can be manipulated in any way we like in order to explore the dynamics using NetworkAnalyser (See appendix A). Note that the analysis is usually done for a few gene net files to check parameter dependence. For example by setting the diffusion parameter to zero for every gene we can determine whether the mechanism is working in a cell autonomous way or not.

Exploring robustness of core topologies function:
We explored the robustness of topology function by re-simulating the topologies that could generate the single stripe pattern with 200,000 parameter sets. The basic definition of robustness of topologies based on the fraction of parameter sets able to achieve a given function could be worked out by taking the ratio of number of parameter sets able to achieve a particular function relative to the number of parameter sets that were tested. The average percentage of the range of each parameter that was compatible with a given function could be worked out using this fraction. The calculation requires the following equation:

\[ F = p^n \]

Where \( F \) is the fraction of viable parameter space, \( p \) is the average viable range of any one parameter and \( n \) is the number of variable parameters. This equation can be rearranged to:

\[ p = e^{\frac{\ln F}{n}} \]
We looked to see whether parameters were restricted to confined values by producing radial plots of the parameter sets, such that there is one spoke for every parameter.

The more refined definition of robustness measured by single parameter variation analysis was measured as follows: The set of parameter sets able to achieve a given pattern for each core topology were collected. For three of these sets that were randomly chosen, each parameter was individually changed incrementally (0.001 for diffusion and 0.1 for regulation) in both positive and negative direction and at each increment simulated. The gene expression pattern resulting from the simulations were compared (using the Euclidean distance) to the gene expression pattern resulting from the simulation with no changes. We refer to the Euclidean distance measured as the goodness of fit score. This analysis continued for each parameter for the entire range of that parameter type (0 to 0.05 for diffusion and 0 to 10 or 0 to -10 for regulation). A plot can then be made of the parameter value against the goodness of fit score (Euclidean distance from the original template).

*Analyzing hybrid topologies:*

We identified the 'core hybrid’ topologies between each pair-wise combination of the core topologies responsible for producing each mechanism. By ‘core hybrid’ we mean that it contains the topological features of each of the two core topologies and nothing more. For some pair-wise combinations of core topologies, these core hybrids did not exist because of topological contradictions between the two core topologies. Where contradictions did not exist we explored the dynamics of the hybrids to see whether they simply function using two different regions of parameter space or whether some form of functional overloading was occurring.
Results and Discussion:

The tools developed here make it very simple to explore the mechanisms responsible for the generation of a particular gene expression pattern. We concentrated on a single gene expression pattern; that of a single stripe. We chose this pattern as it was a non trivial pattern, but also not so complex that very few topologies would be able to produce it.

The complexity landscape for the single stripe pattern is shown in figure 6.5. The shape/topology of the complexity landscape is striking. It clearly shows that there are multiple distinct inverted peaks. Each of the larger inverted peaks ends in a single topology except one which splits into two smaller peaks again ending in a single topology. These topologies at the ends of the inverted peaks thus are the ‘core’ topologies of the complexity landscape.

We explored whether each of those core topologies corresponded to a mechanism and found indeed that they do (See below). This finding confirms one of our hypothesis that we mentioned in the introduction, that mechanism could be conveniently encoded by core topologies.

The single stripe mechanisms:

There are five ways in which a single stripe can be generated which are all cell autonomous mechanisms. They are described here. Their actual dynamics can be seen in the movies of the accompanying thesis CD.

- The first is the classical hierarchical mechanism whereby a left-handed gradient acts to inhibit a second gene that either activates itself or is generally activated over the field. This acts to generate an opposing gradient. A third gene, which is also generally activated or has self-activation is repressed by both of these gradients. If the parameters are right then this gene can be expressed in the central domain where the repression from the other two genes is the least.
- A second hierarchical way of generating the single stripe exists which we called overlapping gradients. In this regime, the first gene forms a left-handed gradient and acts to activate a second gene. Either activation or diffusion
Figure 6.5: Complexity atlas for the single stripe pattern. Dots are topologies and lines are links between one hamming distance neighbors. The shade of the dot is indicative of the fraction of the parameter space able to achieve the pattern (white is high and black is low). Also showing 5 core topologies for generating the single stripe of gene expression.
• effects act to give this gene a different expression range to the gradient of the first gene. The gene with the longer expression range acts to activate a third gene, whilst the other one represses it. Only in the overlap region can the third gene (which is generally activation or has self-activation) be expressed.

• A third way of generating the stripe involves delayed negative feedback which we called bi-stable switch. Two genes of the network form a bi-stable switch which can either be on or off depending on how much repression is coming in from the rest of the network. An inhibition from the rest of the network on to the self activating gene, states that the switch can come on in the centre of the domain but nowhere else.

• A fourth way of generating the stripe involves mutual inhibition. This mechanism involves the gene receiving the pre-pattern input activating both the other genes. These other genes then inhibit one another. One of the two genes wins the competition between the two at either end of the field of cells, whilst the other wins in the centre forming the single stripe of expression. The question arises as to how can this happen?

• A fifth way which we called overlapping domains works in a similar fashion to the overlapping gradients. This mechanism does not however require the negative feedback from the patterned gene onto one of the overlapping gradients genes. In fact it does not require the overlapping gradients to generate the stripe. There is self-positive feedback on the gene forming the single stripe.

Only one of these stripes mechanisms, the classical mechanism seems to be involved in the generation of stripes in the best experimentally understood example of stripe formation, that is, the patterning of anterior-posterior axis of the drosophila blastoderm. In this system seven stripes of both the even-skipped and the fushitarizu gene are generated independently from each other. These genes are generally activated over the field (here mimicked by the patterned genes auto-activation) but then repressed by different inhibitors on their anterior and posterior domains (Kumar and Bentley 2003). This is exactly what happens in the classical hierarchical mechanism found here.
The mechanisms described here for generating a single stripe of gene expression, may not represent all of the mechanisms that are possible with this number of genes. Even though we have included every topology, our model is still constrained by its specific details such as the variable parameters or the pre-pattern. Other mechanisms may exist when these details are changed. Furthermore, any mechanism that requires very specific parameter sets may have not been sampled. However, this is to our knowledge the most comprehensive exploration of mechanisms for a particular type of gene expression pattern. Furthermore our approach has resulted in the identification of patterning mechanisms that are non-intuitive yet realistic.

**Exploring robustness: the parameter space that encompasses the mechanism:**

The results of the number of parameter sets found for each of the core topologies and the corresponding parameter variation tolerance is shown in table 6.6. A substantial fraction of the parameter space is compatible with each mechanism, in the order of 35-40%, though not as much as that reported for the segment polarity network which showed that a randomly picked parameter had about a 90% chance of being compatible with the desired behaviour (von Dassow et al., 2000).

We plotted the parameter sets on a radial graph with each spoke corresponding to a different parameter. The radial graphs show that whilst some parameters are confined to narrow sub-ranges, most are not (figure 6.7).

We performed single parameter sensitivity variation on our core networks. The results are summarised in figure 6.8. Some parameters can tolerate more variation than others. Some parameters are clearly not important for the function (usually diffusion) as can be determined by non-changing goodness of fit score with changes in the parameter that span the entire range. In general apart from specific parameters, they do not tolerate a large degree of variation, as can be inferred from the steep changes in goodness-of-fit score that occur in most of the mechanisms.

"In parameter space, abrupt transitions delineate zones within which the gene network behaves as desired from zones of qualitatively different behaviour" (Von dassow et al., 2000). In contrast to the generally broad regions of working territory that were found for the segment polarity network (Von dassow et al., 2000), we find that most of the canyons of working region are narrow. This suggests that although a
<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Num. Of parameter sets generating pattern</th>
<th>Num. Of variable parameters</th>
<th>Parameter variance tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual inhibition</td>
<td>293</td>
<td>7</td>
<td>40%</td>
</tr>
<tr>
<td>Classical</td>
<td>45</td>
<td>8</td>
<td>35%</td>
</tr>
<tr>
<td>Overlapping domains</td>
<td>255</td>
<td>7</td>
<td>39%</td>
</tr>
<tr>
<td>Overlapping gradients</td>
<td>125</td>
<td>7</td>
<td>35%</td>
</tr>
<tr>
<td>Bi-stable</td>
<td>195</td>
<td>7</td>
<td>37%</td>
</tr>
</tbody>
</table>

Table 6.6: Parameter variation tolerance for each of the identified core mechanisms capable of generating a stripe of gene expression after 200,000 parameter sets.
Figure 6.7: Radial graphs of parameter ranges. Topology structure can be found in appendix E. The individual parameters are spokes on the radial graph. Individual parameter sets are represented by coloured loops around these radial graphs. (Left) Gene-gene regulation parameters. (Right) Diffusion parameters.
Figure 6.8: **Single** parameter variation analysis. Topology structure can be found in appendix E. (Left) Overlapping gradients mechanism. (Right) Mutual inhibition. Vertical axis Max=4,500. Horizontal axis Ranges 0-0.05 for diffusion and 0 to 10 or 0 to -10 for regulation.
<table>
<thead>
<tr>
<th>Parameter set</th>
<th>Diffusion A</th>
<th>Diffusion B</th>
<th>Diffusion C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$B > A$

$A > B$

$C > B$

$A > C$

$B > A$

$C > B$

$B > C$

$C > C$

**Figure 6.8:** Single parameter variation analysis. Topology structure can be found in appendix E. (Left) Overlapping domains mechanism. (Right) Bi-stable mechanism. Vertical axis Max=3,000. Horizontal axis Ranges 0-0.05 for diffusion and 0 to 10 or 0 to -10 for regulation.
Figure 6.8: Single parameter variation analysis for the classical mechanism. Topology structure can be found in appendix E. Vertical axis Max=4,500. Horizontal axis Ranges 0-0.05 for diffusion and 0 to 10 or 0 to -10 for regulation.
substantial proportion of parameter sets are compatible with each of our mechanisms, the parameter values must have specific values relative to each other. Changing any one parameter by itself can 'break' the mechanism. The reason for the difference between our single stripe generating mechanisms and the segment polarity mechanisms may lie in the fact that these networks are the core networks without the addition details that may increase the robustness of the mechanism. The segment polarity network had 48 variable parameters whilst the maximum for the mechanisms described here is 8. The fact that a substantial proportion of parameter space is capable of achieving a function and few are confined to narrow sub-ranges, yet the individual parameters are sensitive to being changed suggests that the viable parameter region(s) must have a complex shape or be discontinuous. A similar observation has been seen in the neurogenic network of drosophila (Meir et al., 2002) by using other methods to measure robustness. Their analysis showed that there were a few large regions of functional space along with many small regions of parameter space that were potentially isolated.

The distinct nature of the complexity atlas:
The topology of this complexity landscape has important consequences for mechanism since it suggests that mechanisms that produce non-trivial gene expression patterns are distinct. That is topologies conforming to each of the mechanisms are more intraconnected than interconnected as suggested by the distinct inverted peaks that are apparent in the complexity landscape. Another way of exploring what happens when we change between mechanisms is to look for hybrid topologies; topologies containing the necessary topological features for two mechanisms. These hybrid topologies are transition points between the two mechanisms. Here then we can address the question as to whether there is a gradual transition between the two mechanisms or whether there is a sudden change in behaviour.
Hybrid topologies:

We explored whether there were hybrid topologies. We specifically looked for the ‘core’ hybrid topology; that is the topology containing the topological features of the ‘core’ topologies for each of the mechanisms and nothing more. Not all combinations of the 5 mechanisms are completely topologically complementary within the 3 gene framework because there are topological clashes; where one core topology needs activation the other needs an inhibition. Only where combinations of core topologies are complementary can we explore what happens in hybrid topologies. Only 3 out of the 10 pair-wise combinations of mechanism core topology are topologically complementary (table 6.9). We therefore analysed the dynamics of several parameter sets for the hybrids of these 3 hybrid topologies (figure 6.10).

Those dynamics showed that the hybrid topology either behaved like one or other of the core mechanisms. At no point did we observe something that was like a hybrid type mechanism. It was this analysis that prompted a deeper theoretical consideration of how hybrid mechanisms could occur.

At a very simple level it is trivial to have a hybrid topology if there are enough genes for the hybrid to contain both core topologies independently (Figure 6.11 top). This would mean that the genes of the two core topologies could behave completely differently. For a true hybrid mechanism however, at least one gene must be shared between the two core topologies (minimal hybrid in figure 6.11). This shared gene or genes must be behaving in exactly the same way in both core mechanisms so as not to contradict either core mechanism.

There are differing degrees of hybridisation depending on the number of genes that the core topologies share (Figure 6.11). At some point increasing the hybridisation of the core topologies could result in a contradiction; that is one core topology requires an activation where the other core topology requires an inhibition. At this point the core topologies cannot be hybridised any further. Some core topologies may not contradict at all like the two illustrated in Figure 6.11. These two topologies can be completely overlapped. However at the point of complete overlap, both mechanisms cannot be occurring simultaneously since their dynamics would contradict if this were true. It is this reason why hybrids between the single stripe mechanisms can never be seen. All of the single stripe forming mechanisms require 3 genes and thus
<table>
<thead>
<tr>
<th>Mechanism pair</th>
<th>Topologically complementary</th>
<th>Functional Overloading?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MI versus OG</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MI versus OD</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MI versus Bi-stable</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MI versus Classical</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OG versus OD</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OG versus Bi-stable</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OG versus Classical</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OD versus Bi-stable</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>OD versus Classical</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Bi-stable versus Classical</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**Figure 6.9:** Analysis of hybrid topologies. Showing the three possibilities for functional overloading. MI=Mutual Inhibition, OG=Overlapping gradients, OD=Overlapping domains.
Figure 6.10: Hybrid topologies. Showing the three possibilities of functional overloading. MI=Mutual Inhibition, OG=Overlapping gradients, OD=Overlapping domains.
Figure 6.11: Extremes of hybridisation. The core topologies for the two individual mechanisms are the mutual inhibition (red dashed line) and the overlapping gradients (green dashed line). A trivial hybrid is shown at the top, there is no overlap between the topologies in this case. The complete overlap is shown at the bottom. Incomplete overlaps can be found in between where one or more of the genes is acting the same way in the two mechanisms. The minimal hybrid is the hybrid where there is only one overlapping gene.
in a framework of 3 genes there will always be contradictions. It will be interesting in future to explore hybrids further with the atlas of 4 gene topologies since the extra gene here allows for incomplete hybrids of 3 gene networks to be generated.

Summary:

- We have shown how by exploring the atlas of gene network topologies, finding the core topologies responsible for any given gene expression pattern is relatively trivial.
- We have shown in the chapter how we can dissect the way these topologies are working to generate the gene expression pattern.
- We have shown how our approach can be used to discover new unpredicted patterning mechanisms. For example we have shown 4 un-described ways of generating a stripe of gene expression.
- We have shown that although a large fraction of parameter space is compatible with each mechanism, single parameter variation analysis reveals that the actual functional domain of parameter space must have a complex shape.
- Functional overloading is not possible for mechanisms involving 3 genes in a 3 gene framework.
7) Discussion.
Key concepts:

The Complete Topology Atlas concept:
The main goal of this thesis was to build and use the concept of a complete atlas of gene network topologies to ask questions about pattern formation and its evolution. Having a complete atlas of topologies proved particularly powerful, since it allowed us to employ two other concepts that have been found to be very successful in the past for asking these kinds of question; the neutral network (NN) concept and the fitness landscape concept. How those two concepts helped us to address our specific questions will now be described.

The neutral network concept:
The concept of a NN allowed us to ask questions about the evolvability and robustness of pattern formation. Because we are using a continuous model of gene regulation, we refer to these NN’s where the phenotypes have some specific features in common as nearly neutral networks (NNN’s). By exploring the shape of these NNN’s we discovered the phenomenon of loop holes. To our knowledge this is the first time that this phenomenon has been described in a biological context. This phenomenon shows how developmental mechanisms can constrain evolution since evolution must take a longer route than is optimal to get from a given gene network to another one. Some early observations as to why these loop holes exist suggest that a topology becomes ‘out of tune’ with a given mechanism. That is an intermediate topology on a direct route between two other topologies may contain features that reduce its robustness.

By looking for the potential routes from one NNN to another NNN, we could analyse how pattern innovation is possible. Exploring where on NNN’s the routes are to other pattern mechanisms has important consequences for how innovation can occur. We showed that there are some routes between every pair-wise combination of NNN. Where these evolutionary ‘doors’ are on the NNN has important consequences for the dynamics of evolution (Fontana and Schuster 2000; van Nimwegen et al., 1999).
The landscape concept:
The concept of a landscape was used to generate complexity landscape of gene networks capable of generating a particular gene expression pattern. The tools that we developed in our GUI allowed us to extract the maximal benefit from such a concept. The GUI made it simple to explore the shape of the landscape and to find the local minima that corresponded to ‘core’ topologies responsible for generating any given pattern. The GUI also made it simple to explore how the ‘core’ gene networks at these minima were functioning. Furthermore it made finding the hybrid topologies containing the core networks easy.

Encoding mechanism:
We have seen in chapter 6 that core topologies seem to represent a convenient way to encode mechanism. By measuring the hamming distance between two core topologies, we have a metric to measure how different to patterning mechanisms are. This gives us a theoretical framework to start bringing the ideas of pattern formation together. We have seen in chapter 6 how we could explore the topology of the complexity landscape. This analysis showed that there were multiple mechanisms for producing the single stripe gene expression pattern and that they were generally discrete from each other. Hybrid topologies containing the necessary features for two mechanisms worked either by one mechanism of the other. ‘functional overloading’ was not possible with these mechanisms since they each required 3 genes. When using an atlas of 3 gene topologies then, only a complete hybrid is possible (figure 6.10 and 6.11). In this situation it is impossible to avoid contradictions in gene function between the two mechanisms.

We are currently in the process of taking this analysis one step further towards real networks that function in living systems by simulating the complete list of 4 gene networks. This analysis has required the use of the Mare Nostrum supercomputer. Furthermore this dataset should allow the further exploration of the phenomena of functional overloading since hybrids other than the complete type are possible (figure 6.10 and 6.11).
Questions and answers:
I will now discuss the success of this approach for addressing the specific questions that we raised in the introduction chapter.

1) What phenotypes are possible? What is the shape of possible phenotype space? Is it continuous or discreet? If it is discreet, how many pattern types are there?

Answer:
Phenotype space appeared to form a large continuum that contains two fuzzy clusters corresponding to the right and left handed gradient patterns. Observing the patterns by eye suggests that patterns can only blend into one another via specific directions in phenotype space. This corresponds to an irregular distribution of the patterns in phenotype space where more complex patterns are found in ‘lobes’ of the continuum.

2) What is a patterning mechanism? Are they discreet or do they tend to blend into one another? How many patterning mechanisms are there?

Answer:
Complex patterning mechanisms like those generating the single stripe of gene expression described in this work have clear delimiting boundaries. The topology of the complexity landscape suggests this to be the case (Chapter 6). Analysis of hybrid topologies, shows that these mechanisms are working using different regions of parameter space suggesting them to be distinct. Indeed if the number of individual parameter domains for any one topology was taken into account then this connected NNN may break up entirely into multiple smaller NNN’s, one for each of the patterning mechanisms. Theoretical considerations suggests that functional overloading could occur if the number of genes involved in the mechanism is less than the number of genes in the network. Exploration into this phenomenon in 4 gene networks will be performed in future.
3) To what extent do single topologies exhibit multiple mechanisms? What does this say about the relative importance of topology versus parameters?

*Answer:*
The fact that some topologies are found to overlap between neutral networks generating very different patterns is clear evidence that topology alone is not enough to predict network behaviour (Chapter 5). Hybrid topologies where a topology can generate the same pattern using two different mechanisms are another example of this (Chapter 6). However our demonstration that ‘core’ topologies can be used to encode mechanism suggests there may be a one to one relationship between patterning mechanism and core topology in general. This illustrates how considering topology alone can be useful.

4) Is the least complex core topology for a mechanism the most robust? If not what design features confer robustness to the topology? What is the relationship between topological complexity and robustness?

*Answer:*
The complexity atlas shown in figure 6.5 suggests that the ‘core’ topologies are never the most robust for generating a particular pattern of gene expression. There is usually some optimal complexity for generating a gene expression pattern. Overloading a topology with too many interactions reduces robustness. This is exactly as would be expected when thinking about how patterning mechanisms could work since it can be envisaged how extra interactions could make a topology more robust for example through redundancy. In the future we will perform a more thorough analysis of the relationship between complexity and redundancy.

5) What happens when a gene network topology contains the motifs for two different patterning mechanisms for generating the same gene expression
pattern? Are they combined into some hybrid mechanism or do both mechanisms stay separate with different regions of parameter space performing either function? If the latter is true then how is it possible for evolution to move between these two mechanisms?

*Answer:*

We have shown that it is theoretically impossible for functional overloading for mechanisms involving 3 genes in 3 gene topologies since conflicting constraints are unavoidable (Chapter 6). We are therefore currently analysing hybrid topologies in 4 gene networks to see if this is possible.

6) Do all of the topologies generating any one pattern form a neutral network in topology space? If so what is the shape of this neutral network? What does this shape say about the routes available to evolution?

*Answer:*

The vast majority of topologies generating all of the defined patterns form a connected component (a neutral network) in topology space that would not be expected by chance (Chapter 5). The shapes of these neutral networks seems to be irregular resulting in loop holes in genotype space. This irregular shape must be a consequence of epistasis, conflicting constraints between genes, showing clearly how development can constrain evolution.

7) How can evolution change one pattern into another? Is there only one route or are there multiple? If there are multiple routes are there these routes equally likely to be traversed?

*Answer:*

Almost all pair-wise combinations of NNN’s have some degree of topology overlap and bridges between there topologies (Chapter 5). This suggests that
it is possible to change from any one gene expression pattern to any other. It will be interesting in future to map where these overlaps and bridges are on the neutral networks. Are they spread evenly throughout? Or are they concentrated in particular regions of the neutral networks. These features have important consequences for the dynamics of evolution (Fontana and Schuster 2000; van Nimwegen et al., 1999).

Concluding remarks:
A genetic system is usually considered well understood if we can predict its behaviour and the behaviour of its mutants and explain how that behaviour comes about. In the language of the topology space, this amounts to an understanding of a given gene network and its local one hamming distance neighbours. The complete topology atlas takes understanding one step further, in that it maps out the possible behaviours for all topologies. This approach gives us a whole theoretical framework for exploring patterning mechanisms. Because a mechanism has a “location” in design space we can relate other mechanisms to it. We believe such a theoretical framework can be used to start extracting fundamental principles of patterning mechanisms. As mentioned, modelling approaches don’t simply serve as tools for exploring specific instances of design but also as a tool for exploring “design space that is shaped by fundamental principles, structural, environmental and evolutionary constraints”(Kitano 2007). Our attempt to explore whether mechanism space is discrete or continuous is an example towards this end.
On a final note I would like to emphasise the role of the metaphor of an “atlas” in aiding our understanding of patterning mechanisms with the following quote:

"Perhaps our ultimate understanding of scientific topics is measured in terms of our ability to generate metaphorical pictures of what is going on. Maybe understanding is coming with metaphorical pictures.” (Bak 1996).
8) General Materials and Methods
**Hardware used:**
Small topology simulation jobs were carried out on the lab linux PC which has 2 XEON processors, each with a speed of 2.6Ghz, a physical memory of 1Gb and a virtual memory of 4Gb. Large jobs were simulated using the program computer cluster that consisted of 20 calculation nodes and a master node. The nodes are HP proliant BL35p, each with 2 dual core AMD opteron processors 280 2410Mhz and 2 4Gb DDR RAM PC1200. This results in a total of 80 processors for calculations. Data was stored on the local Procsys files server secured via the RAID strategy.

**Software used:**
Graphic Interchange Format (GIF) images representing gene expression patterns were generated using a php script. All Graphical User Interface (GUI) software was generated using TclTk. All other programs were written using the C programming language.
Appendix A: The GUI control options:
Generating small gif images of the gene expression patterns resulting from the granularity algorithm:

Gif files of 100x100 pixels are made for each pattern using a php script that reads the expression values for each gene and draws a one dimensional graph with each gene represented as a different colour. These images can be conveniently browsed using linux’s browse view as icons options. These gifs are imported into the cluster and network analysis graphical user interfaces to represent the leaves of the dendrogram.

The cluster and complexity landscape graphical user interface:

Once booted the Cluster Toolbar appears that has several options (Figure A1 top). Before anything can take place, either a cluster or a complexity atlas directly needs to be loaded. This can be done by selecting “OpenCluster” or “LaunchAtlasWithoutCluster” and inputting the correct pathname to the data directory.

If “OpenCluster” was selected:

The program will then boot a visual dendrogram of the cluster with patterns shown at the leaves of the tree (Figure A2 left). Toggle options “TogglePatterns” and “ToggleInternals” in the cluster toolbar allow the basic branch structure of the dendrogram to be seen (Figure A2 right). Control options include moving the cluster by holding down the middle mouse button and dragging the cluster, and zooming in and out by holding down the right mouse button and moving the mouse down and up respectively. Any node, leaf or internal (those that are at the branch points of the cluster) can be added to the ‘region of interest’ list by double left clicking on it. The node will then have a blue outline (Figure A3). In this way a region that represents one pattern type can be selected. Whole branches should represent a single pattern of interest. Therefore to make things easier, one can also double left click on internal nodes turning them blue and adding all of its leaf patterns to the region of interest.
Nodes can be deselected from the region of interest list by again double left clicking on them or by clicking on the “clearROI” button in the cluster toolbar.

Once the patterns of interest has been selected, a landscape can be build of the topologies that could generate those leaf patterns in the ROI by clicking the “BuildAtlasUsingROI” option. This will call a c script that performs the complexity landscape building algorithm described in chapter 6 and will bring up the Atlas Toolbar (Figure A1 bottom) once it has finished.

By clicking on the button “BringUpAtlas” on the atlas toolbar, the complexity landscape can be brought up in a new window (Figure A4 left). The nodes of this window represent the topologies that can generate the patterns of interest. There y position is determined by their complexity and x position by the algorithm described in chapter 6. A scale bar at the side of the landscapes indicates the complexity of the topologies. The colour of the node represents the fraction of parameter space that achieved the patterns of interest. The brighter the shade, the higher the fraction of parameter sets. By double left clicking on a topology on the landscape view, its topological neighbours are highlighted in green and brought to the top of other topologies if there is more than one topology in the same place. By double left clicking again those topologies return to their original colour. Nodes representing topologies can be individually moved using the left hand mouse button and the whole atlas can be moved or zoomed in and out using the same controls as for the dendrogram. If the nodes are moved, then the interactions between those nodes automatically follow. The topologies that the nodes of the landscape correspond can be seen by double left clicking on the node. A new window will appear (if it is not already there) with space for 16 topologies and the topology just clicked on will appear in one of these spaces (Figure A4 right). This functionality allows commonalities in the topologies that can produce the pattern to be explored in a visual way. Genes of the topologies can be seen as coloured dots. The pre-pattern ‘virtual’ gene is always coloured green. A green line coming from a gene indicates an activation of this gene to which ever other gene it is pointing. Red indicates inhibition and a gap means no interaction. If the gene has a coloured outline, it indicates auto-regulation.
A useful feature for the incidence landscape view is the reorganise option which is the functionality that allows overlapping inverted peaks to be teased apart as described in chapter 6. The reorganise option is called by holding down the “r” button on the keyboard. This button will cause a tcl script to loop through all of the topologies and move them left or right depending on the complexity of their neighbouring topologies.

If “LaunchAtlasWithoutCluster” was selected:
This option allows one to open complexity landscapes without using the dendrogram view to generate them. This is what happens for instance when complexity landscapes are generated via pattern matching. If this button is selected another window appears where the data directory can be inputted resulting in the complexity landscape being brought up with all of the same options as described above.

Topology options:
By middle clicking on the topology of interest in the topology window the tcl script will output the topology structure to a ‘topology.txt’ file and bring up the ‘find mechanism’ window of the GUI (Figure A5). One must then choose a digital definition of the pattern to find in the ‘find mechanism’ (currently done manually) window and click ‘go’ which will call a c script called TopologyAssessor_FindMechanism.
The role of TopologyAssessor_FindMechanism is to find a set of parameters for the given topology that confer the patterning ability. It is written in c as c is much faster than tcl and these calculations are computationally demanding. TopologyAssessor_FindMechanism will read the topology and generate random parameter sets for it. It will simulate the topology with a given set of parameters and ask whether the resulting pattern matches the digital definition. This continues until TopologyAssessor_FindMechanism finds a set of parameters that can achieve the appropriate digital definition. It will then create a “Tfindmech.txt” file (Figure A6) which is written in our in house ‘genenet’ file format, describing the topology and parameters in a format that NetworkAnalyser can read, and then call NetworkAnalyser.c using this
"Tfindmech.txt" file as an argument. This NetworkAnalyser.c will resimulate the topology with the particular set of parameters and produce a basic temp.txt file describing the concentrations of the gene products in every cell at every time point for every noise run. It will then call the NetworkAnalyser GUI written in TclTk to display the results (Figure A7).

*Network Analyser to interactively view gene expression dynamics:*  
The NetworkAnalyser GUI window will appear showing the dynamics for which that topology can produce that pattern (Figure A7). This window has a frame on the left showing the structure of the topology. The genes appear as dots and the interactions as lines between those dots. A red line means inhibition and green means activation. The half line towards one of the dots shows its strength of regulation on the other gene. The thickness of the line is a direct readout of the strength of this regulation. If a gene regulates itself then it is surrounded by an outline, again green if it positively activates itself or red if it represses. The thickness of this outline is again a direct read out of the strength of this self regulation. In the middle frame of this GUI is shown the exact gene expression pattern for all of the genes over the field of cells (a profile) and there are two sliders beneath. The first changed the time of the simulation so one can see exactly how the mechanism unfolded. The slider underneath shows the dynamics for each of the noise runs so one can check just how stable this output was to the noise.

*Exploring how topologies function:*  
How topologies are functioning can be explored by manually editing the "Tfindmech.txt" file. For example the diffusion parameters can be set to 0 for each of the genes to explore the effect of diffusion.

*Generating movies of gene expression dynamics:*  
Movies can be generated of these dynamics by the use of a generate_geneexpression_animation.php program. This program reads the same temp.txt file as the NetworkAnalyser GUI, but uses the data to generate png's for each of the time steps of a given noise run in the same
representation as that seen in the GUI. An avi movie can be generated using the ImageJ software by reading in the image series.
**Figure A1:** The Graphical User Interface toolbars.
Figure A2: The cluster dendrogram view.
Figure A3: Zooming in on a Region of Interest (ROI) on the cluster.
Figure A4: The complexity landscape and topology viewing window.
**Figure A5:** The topology viewing window and controls to simulate a topology.
Figure A6: The ‘Genenet’ file format
Figure A7: The NetworkAnalyser program for viewing topology dynamics.
Appendix B: Patterns resulting from the granularity algorithm.
Each square is a gene expression pattern resulting from the granularity algorithm (See chapter 4). The x-axis is the cell position (1-32) and the y axis is the gene product concentration (0-20).
Each square is a gene expression pattern resulting from the granularity algorithm (See chapter 4). The x-axis is the cell position (1-32) and the y axis is the gene product concentration (0-20).
Each square is a gene expression pattern resulting from the granularity algorithm (See chapter 4). The x-axis is the cell position (1-32) and the y axis is the gene product concentration (0-20).
Each square is a gene expression pattern resulting from the granularity algorithm (See chapter 4). The x-axis is the cell position (1-32) and the y axis is the gene product concentration (0-20).
Each square is a gene expression pattern resulting from the granularity algorithm (See chapter 4). The x-axis is the cell position (1-32) and the y axis is the gene product concentration (0-20).
Appendix C: Complexity landscapes for the single stripe pattern using alternate gene regulation models and pre-patterns.
<table>
<thead>
<tr>
<th>Data set</th>
<th>Description</th>
<th>Number of topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Michalis-Menten Input range 10-90%</td>
<td>710</td>
</tr>
<tr>
<td>C</td>
<td>Sigmoid Input range 10-50%</td>
<td>619</td>
</tr>
<tr>
<td>D</td>
<td>Sigmoid Input range 10-90%</td>
<td>1,273</td>
</tr>
</tbody>
</table>

**Figure C1:** The number of topologies in the complexity landscape using the other standard data sets
Figure C2: The complexity landscape for standard data set B
Figure C3: The topologies of complexity landscape C2 of standard data set B. This shows all 5 core topologies as for standard data set A.
Figure C4: The complexity landscape for standard data set C
Figure C5: The topologies of complexity landscape C4 of standard data set C. This shows mutual inhibition and overlapping gradient core topologies and a simpler version of the overlapping domains core topology.
Figure C6: The complexity landscape for standard data set D
Figure C7: The topologies of complexity landscape C6 of standard data set D. This shows overlapping domains, mutual inhibition and overlapping gradient core topologies.
Appendix D: Sampling parameter space.
The NNN's and complexity landscapes described in this work may not include all of the topologies able to generate a particular pattern. The reason for this is that topologies that could produce this pattern but with a very small parameter volume could be missed by parameter sampling.

We tested how many random parameter sets would be needed to sample mechanisms with different parameter volumes. These values were generated theoretically using the following formula:

\[ N = \text{number of parameters sets} \]
\[ P = \text{fraction of parameter space encompassing the mechanism} \]

\[ \text{Chance of sampling mechanism} = 1 - ((1 - P)^N) \]

The term \((1-P)\) describes the chance of a randomly chosen parameter set not residing in the mechanism region. This chance of missing a topology reduces as parameter number increase. The data for this formula is shown on the graph in figure D1.

In this work we have used 10,000 parameter sets, hence we have a 95% certainty of finding topologies with a functional parameter fractions as low as 0.03%.
Figure D1: The graph describes the chance of finding any one particular topology can produce a given pattern when it has different fractions of parameter space that are functional (different coloured lines)
Appendix E: The core topologies producing the single stripe gene expression pattern.
Figure E1: The 5 core topologies are shown. A, B and C are the labelled genes. Arrows represent activation and 'dead-end' lines represent repression. The gene that the pre-pattern feeds into is indicated.
References:


Diestel R. (2007). Graph theory. 2nd ed, New York: Springer.


