TOWARD A BIOLOGICALLY DEFENSIBLE MODEL OF DEVELOPMENT

by

FRANK DELLAERT

Submitted in partial fulfillment of the requirements

for the degree of Master of Science

Thesis Adviser: Dr. Randall D. Beer

Department of Computer Engineering and Science

CASE WESTERN RESERVE UNIVERSITY

January, 1995

TOWARD A BIOLOGICALLY DEFENSIBLE MODEL OF DEVELOPMENT

Abstract

by

FRANK DELLAERT

This thesis discusses a biologically defensible model of development for artificial organisms. It can be used in conjunction with genetic algorithms to design autonomous agents, complete with body and nervous system. It also has potential applications in the field of theoretical biology. The approach taken is one of exploration, i.e. the model is implemented simultaneously at three different levels of complexity: regulation of gene expression, development of body morphology and finally neural development. At each level the model's behavior under evolutionary pressure is addressed. The whole integrated model will be illustrated with a hand-designed artificial organism, that is capable of executing a simple avoidance behavior in a simulated world.

Acknowledgments

I would like thank Randy Beer, for he made this thesis possible. It was to work with him that I chose to live in Cleveland, of all places, and I have not regretted it one moment.

My good friend Hurkan Balkir.

Leslie Picardo, Brian Yamauchi, Alan Calvitti and John Gallagher for their friendship and the stimulating intellectual climate they created (most of the time :-).

For providing me with a place to stay at various times: the Van Laethem Family, my parents, Diederik Dellaert, Emanuelle and Piet Vandervoort and especially Cheryle and Tom Mackie, because I could feel so at home with them.

Pattie Maes and Karl Sims, for their hospitality and for sparking my interest in Artificial Life research.

Irving Bell and Laurie for some of the little income I had in the past year.

Jan Van Gael, who understood my desire to fly.

Karin and Michael Maes, for providing me with a little space on board.

Leon Sterling, for being on my committee, and Hillel Chiel, who has helped tremendously in saving me from a deadly confrontation with the biological research community.

Paul Stephan, who has provided me with generous access to his Quadra 800.

Apple Computer, for hanging in there.

Stephen Haynesworth, who introduced me to the wonders of developmental biology, in one of the best courses I ever took.

Pauline Hogeweg, Maarten Boerlijst, Stefano Nolfi, Frédéric Gruau, Rik Belew for their helpful comments.

Joos Vandewalle and Peter Van Overschee, Katholieke Universiteit Leuven

Phil Husbands, Peter de Bourcier and the others at COGS, University of Sussex.

Most of all, I would like to thank my parents, Hilde en Koen Dellaert, and my net-surfing uncle, Frank Dellaert sr., for supporting me in various ways;

and Katrien, whose support and love have been irreplaceable.

Table of Contents

Chapter 1

Introduction

In this thesis I want to present a simplified yet biologically defensible model of development, the process by which in nature a fertilized egg transforms into an adult organism.

Modeling development has many potential applications in the field of theoretical biology: development is a process in which dynamics and self-organization play a very large role, and there are many questions about these aspects of development that remain largely unanswered. In addition, after a long period in which they were mostly studied in isolation from each other, questions about the interaction between development and evolution have received considerable attention in more recent years. A computer simulation in which both a model of development and of the evolutionary process are present might help in answering some of these questions.

Another application is in the field of Simulated Adaptive Behavior. There has been a lot of work in this area where the authors use genetic algorithms to try to evolve neural networks to control the behavior of autonomous agents. In most of these cases, however, the mapping between the network itself and its genetic description is quite trivial, leading to problems of scaling and limited complexity, apart from missing out on the advantages that a *developmental approach* can bring. Such an approach would use a *developmental specification* for agent designs and could form the basis for a new and powerful way to explore adaptive control in autonomous agents. In addition, a biologically inspired model of development opens the way towards the co-evolution of body morphologies and nervous systems in these agents.

In what follows, I will present a computer model that simulates the ‘development’ of artificial organisms. The organisms in question are quite simple, consisting of a collection of cell-like square elements that - unlike real cells - are incapable of movement or changes in shape. However, the process by which one such artificial creature develops from a single cell into a fully grown organism captures some of the essential characteristics of biological development: just like its biological counterpart, this *simulated developmental process* will be directed by information contained in the ‘genetic material’ of the organisms, it will emerge from the interaction of many simple elements, and it will eventually result in a multicellular artificial organism that is capable of behavior itself.

When this project was first conceived, there was relatively little literature on modeling development as a whole, and the approach taken here is consequently one of exploration. Instead of focusing in great detail on one particular aspect of development, a cut will be made through the whole process, from the cell behavior that underlies all of development right up to the developmental specification of a nervous system. Obviously, this necessitates abandoning detailed biological modeling, instead opting for making educated guesses and pragmatic modeling decisions where this is needed to make the model work as an integrated whole.

Throughout the following discussion, the interaction of evolution and development will be present as one of the important themes to be kept in mind at all times. Indeed, one of the primary design goals for the model is that it should be based on a genetic specification of the process, so that genetic algorithms can be used to evolve organisms beyond what we are capable of designing ourselves. This opens possibilities for the automatic design of autonomous agents, which becomes necessary as the complexity of these agents and the environment they adapt to increases. Also, many important questions in biology focus on the role of development throughout evolution, thus the emphasis on this aspect.

Given the importance attached to the integration of the model with evolutionary algorithms, another design goal is that of computational efficiency, i.e. the model should be kept as simple as possible. It is of course more agreeable to be able to experiment with the model in an interactive way, as opposed to waiting around for hours to see the outcome of the experiments you set up. However, when using genetic algorithms, efficiency becomes a must: populations of hundreds of genomes are used for many generations on end, and the developmental simulation is situated right in the inner loop. Thus, experimenting with genetic algorithms is nearly impossible unless the model is simple and fast enough.

A final goal that was envisioned from the start and conflicts to a degree with that of simplicity, is to keep the model biologically defensible. One can approach the modeling of a developmental process in many ways, depending on the goal one has in mind. For example, some authors are primarily concerned with the design of neural networks and regard correspondence to the biological phenomenon as irrelevant and even a liability. Then again, biologists often take the opposite viewpoint, and go a long way towards making the model as biologically realistic as possible. The approach taken here is to try to build a *biological defensible* model, inspired by and hopefully relevant to biology on a higher level, but not copied from it in all the low-level details.

This would not only prove impossible when making a cut through the *whole* process of development, but one could also debate over whether it is in those details that the principles of development are to be found. In addition, by taking this route, we can regard the principles of biological development as a constant source of inspiration and guidance. Thus, the viewpoint here is one that is shared with researchers in the field of Artificial Life, who regard computer simulation per se as a complementary source of knowledge about the principles that govern living systems, as opposed to considering only the biological, carbon-based systems.

The chapters of this thesis are laid out in the same way as the model itself is structured: from a chapter about the underlying mechanisms of cellular behavior and genetic regulatory modeling, to one on the emergence of multicellular development, and finally one about the development of simple nervous systems. In addition, in every chapter discussing these topics, an attempt will be made to examine the behavior of the model in the context of evolution. Using genetic algorithms to evolve cellular behavior and/or artificial multicellular organisms will yield insights into the interaction of development and evolution, as well as demonstrate the range of systems the model is able to generate.

However, because building a biologically defensible model of development necessarily involves some background in the corresponding biological process, I will start off in Chapter 2 with a short introduction to the biological mechanisms of development. In it, I will also talk briefly about the general theory of evolution, and discuss the genetic algorithms that are inspired by it.

After the background chapter, the necessary groundwork has been laid to provide the reader with a more elaborate motivation for my work. Thus, Chapter 3 discusses the application of a developmental model to the area of Simulated Adaptive Behavior, and touches upon some questions in biology that may be amenable to study by simulating an artificial developmental process in conjunction with genetic algorithms.

The first important component of the model will be addressed in Chapter 4. There, I will discuss how genetic regulatory networks can be modeled and how they relate to the behavior of single cells and multicellular development. The model presented is based upon random Boolean networks, a computational paradigm related to cellular automata. These networks will be discussed from a dynamical systems perspective, which will provide us with a powerful tool to analyze the dynamics underlying development. As this will be important for the remainder of the model, attempts will be made to explore the evolvability of these networks, attempting to attain unicellular behavior relevant to the process of development.

In Chapter 5 the regulatory network model is embedded in a model of the artificial multicellular organisms mentioned above, enabling us to explore development of morphological patterns and organization. After studying the behavior of this multicellular model under evolutionary pressure, a modification to the original gene regulation model is proposed, one that may be better suited to the evolution of stable patterns of cellular differentiation so important in development.

The development of simple nervous systems is explored in Chapter 6, where we extend the model with mechanisms of axon outgrowth and axon path finding. The simultaneous development of body and brain is demonstrated by showing a simple but complete autonomous agent that is capable of a simple chemotaxis-related task. As in the two other chapters discussing components of the model, we will also study how this extended model can be evolved, and how the nervous system can adapt to the underlying body morphology.

That chapter is then followed by a comparison of this work with related literature in Chapter 7, in which related research by two types of authors are discussed: those who take the viewpoint of theoretical biology (some of whom model selected and highly specific aspects of development as opposed to development as a whole), and those that are primarily interested in the applications of a developmental approach to engineering or AI applications (most of them focusing on the synthesis of neural networks with applications in Adaptive Behavior research).

Finally, in Chapter 8, the lessons that can be drawn from this work are discussed, as well as avenues for future work.

Chapter 2

Background

Biological Development in Overview

This chapter serves to give an overview of the developmental process in biology. Although we will not attempt to model all the macroscopic events described in what follows, it is definitely important to be able to situate our model inside a broader context. By doing this, one can get a feel later for just what aspect of development the model tries to address and what mechanisms are left out. Thus, I will try to relate the model to the biological phenomena discussed at each step.

Most of what is said here is adapted from (Walbot and Holder 1987), and I will also stick to their order of presentation, which distinguishes five stages in the developmental program. These stages are respectively: production of gametes, fertilization, cleavage, gastrulation and organogenesis. This order is helpful to provide an introduction to the mechanisms of development.

Production of Gametes

Every multicellular organism develops from a single cell, a fertilized egg. Thus, one can consider the first step in development to be the production of the egg and sperm cell that fuse to yield this cell. These two specialized cells are called the gametes.

In the model which I will propose (see Chapter 5), the genetic algorithm will be responsible for reproducing the genetic material for the next generation of organisms. Thus, there is no need in our model for simulating how organisms produce the cells that carry the genetic material for their offspring. However, it is still of value to realize that, in living organisms, there is an elaborate mechanism that takes care of this important task. In mammals, for example, a number of specialized cells and structures are present in our bodies both for the production of sperm in the male and of eggs in the female. To give you a feel for the process, Fig.2-1 and Fig.2-2 represent stages in the development and maturation of both the mammalian sperm and egg cells respectively.

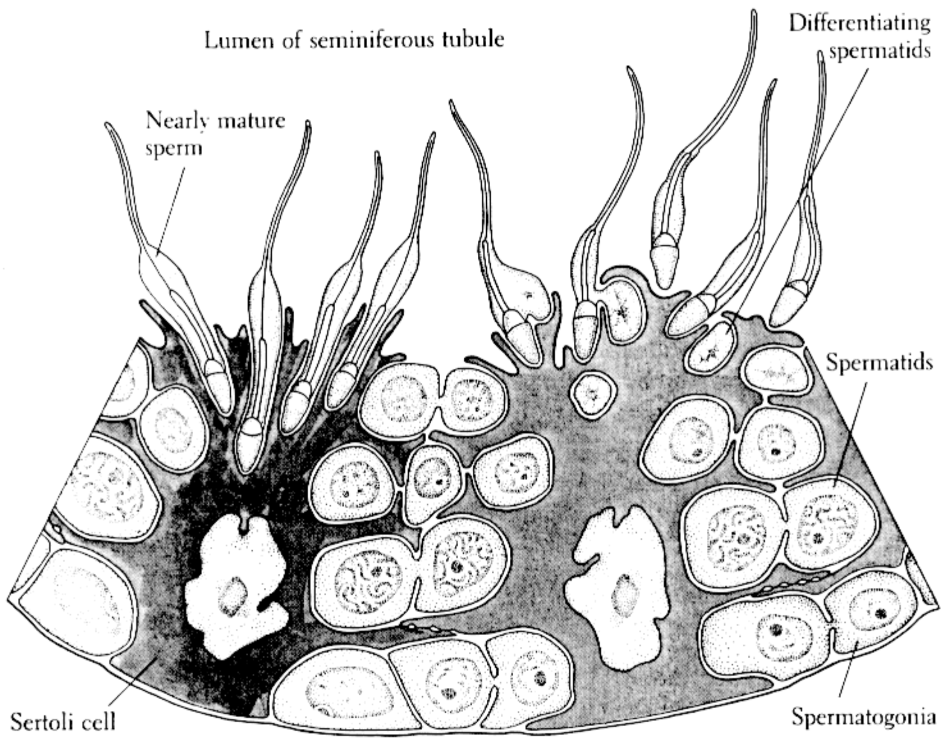


Fig.2-1. Mature sperm cells (top) differentiate from precursor cells (bottom) as they are pushed from the outside to the inside of a specialized tubular structure. From (Walbot et al. 1987).

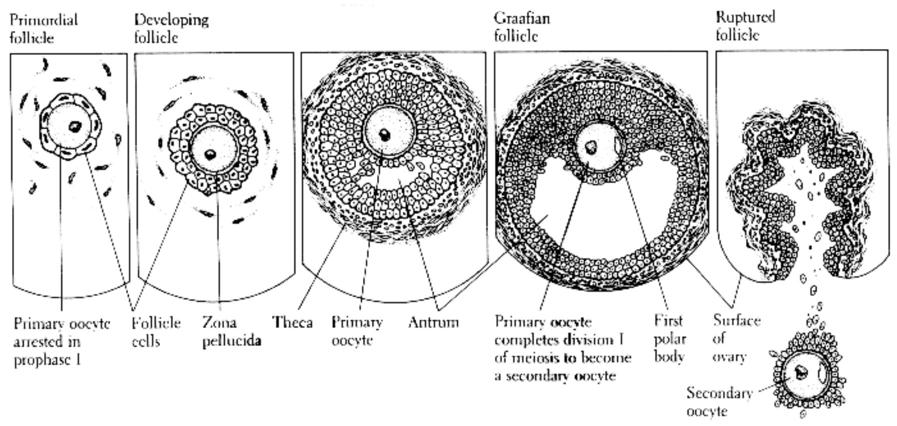


Fig.2-2. The maturation process of an egg. From (Walbot et al. 1987).

Note also that in many species the egg is produced in an asymmetric fashion. For example, in Fig.2-3 the egg of an ascidian is shown, where different cytoplasmic regions can be clearly distinguished. Later in development, the cells that originate from a certain region will give rise to the same type of tissue of the adult organism. This will be of significance for our model, as in Chapter 5 I will discuss how a model of the early stages of multicellular development will explicitly incorporate a mechanism to break the symmetry after the first 'cleavage' event.

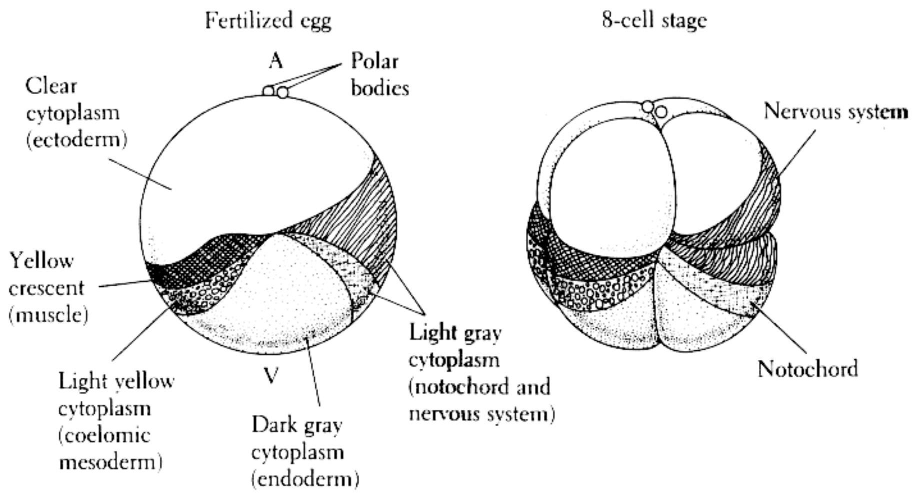

Fertilized egg
8-cell stage
A
Polar bodies
Clear cytoplasm (ectoderm)
Nervous system
Yellow crescent (muscle)
Light yellow cytoplasm (coelomic mesoderm)
V
Dark gray cytoplasm (endoderm)
Light gray cytoplasm (notochord and nervous system)
Notochord

Fig.2-3. Inside the egg of an ascidian clearly different cytoplasmic regions can be distinguished. These will later give rise to different body structures. From (Walbot et al. 1987).

Fertilization

Once the gametes have been produced, they need to get together and produce a fertilized egg, as they both carry only half of the genetic material needed to go on with the remainder of the developmental process. More precisely, they need to combine their haploid chromosomal contents to yield a full diploid genome. After that has happened, a new organism begins to develop. In the model, we will not attempt to model these details, but we will simply represent their key result -combination of genetic material (see the section about genetic algorithms, later this chapter).

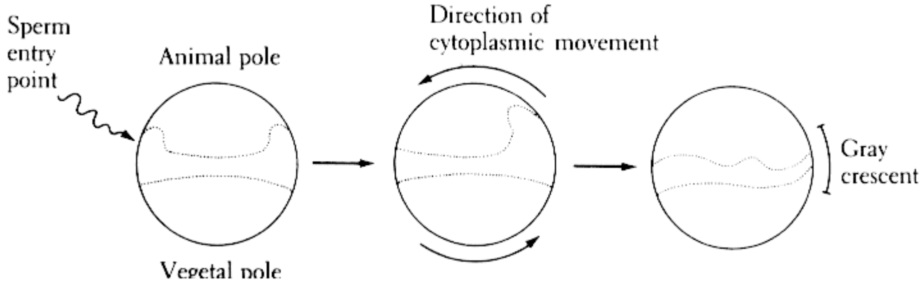


Fig.2-4. The entry point of the sperm in a frog egg starts a process by which a region (the gray crescent) develops that in turn will specify two out of the three body axes in the embryo. From (Walbot et al. 1987).

Importantly, in some (but not all) systems fertilization is also a way to introduce an asymmetry in the egg cell, that serves as the basis for the specification of one or more body axes. For instance, in Fig.2-4. is shown how the entry point of the sperm in a frog egg initiates a process by which a region develops, the gray crescent, that specifies two out of three body axes in the frog embryo. The third body axis is already predetermined by an asymmetric distribution of yolk in the egg, again illustrating the importance of this asymmetry in some species. As said before, a mechanism to break the symmetry in the model will be discussed in Chapter 5.

Cleavage

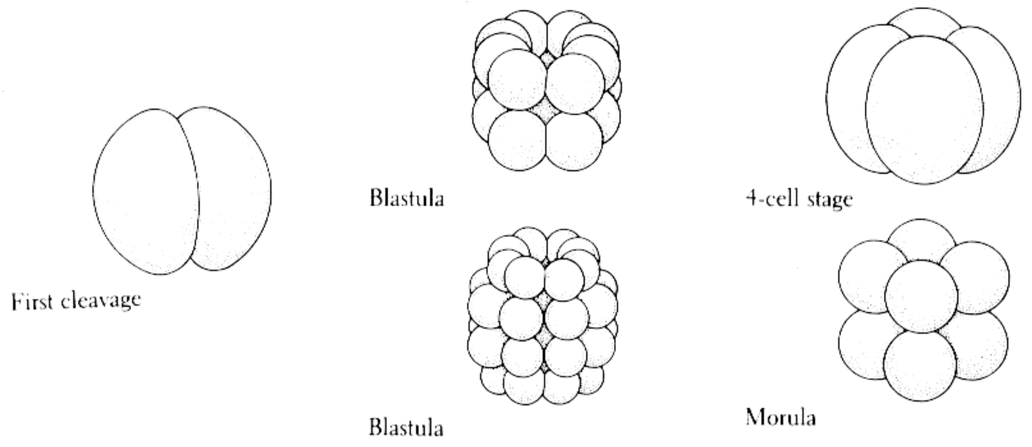


Fig.2-5. The sequence of events during cleavage. From (Walbot et al. 1987).

After fertilization the fusion product of egg and sperm (the zygote) will go through a period of rapid synchronous cell divisions called *cleavage*. A number of such divisions result in the formation of a solid ball of cells called the *morula*, and by the end of cleavage a fluid-filled cavity is formed, resulting in a hollow ball of cells called the *blastula*. This process is illustrated in Fig.2-5, and in Fig.2-6 the blastula stage of a sea urchin and a frog are shown.

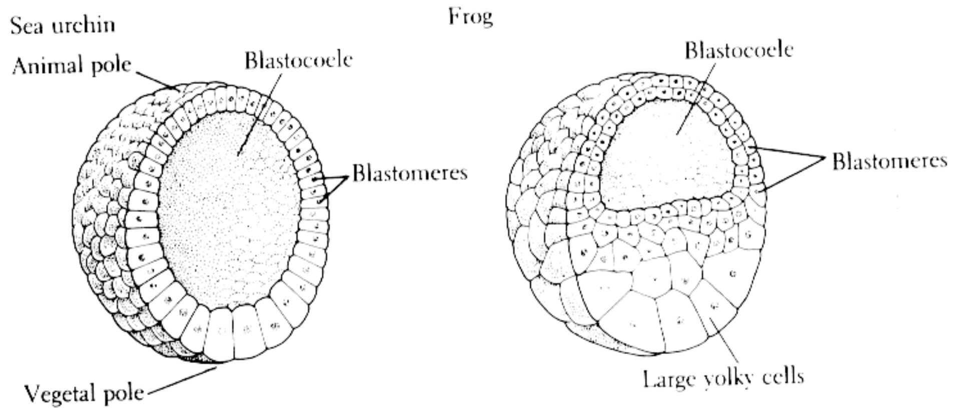


Fig.2-6. The blastula stage of a sea urchin and a frog. From (Walbot et al. 1987).

The cleavage period in many organisms is the stage wherein the initial asymmetric distribution of some component in the egg is used to set up differences in gene expression between the many cells that result from the successive divisions taking place. The developmental model will capture the aspect of symmetry breaking and the subsequent interactions that give rise to spatial patterns in an artificial 'embryo', as you will see in Chapter 5, although the mechanisms used in the model are somewhat different from those that are most commonly seen in very early development[1].

Gastrulation

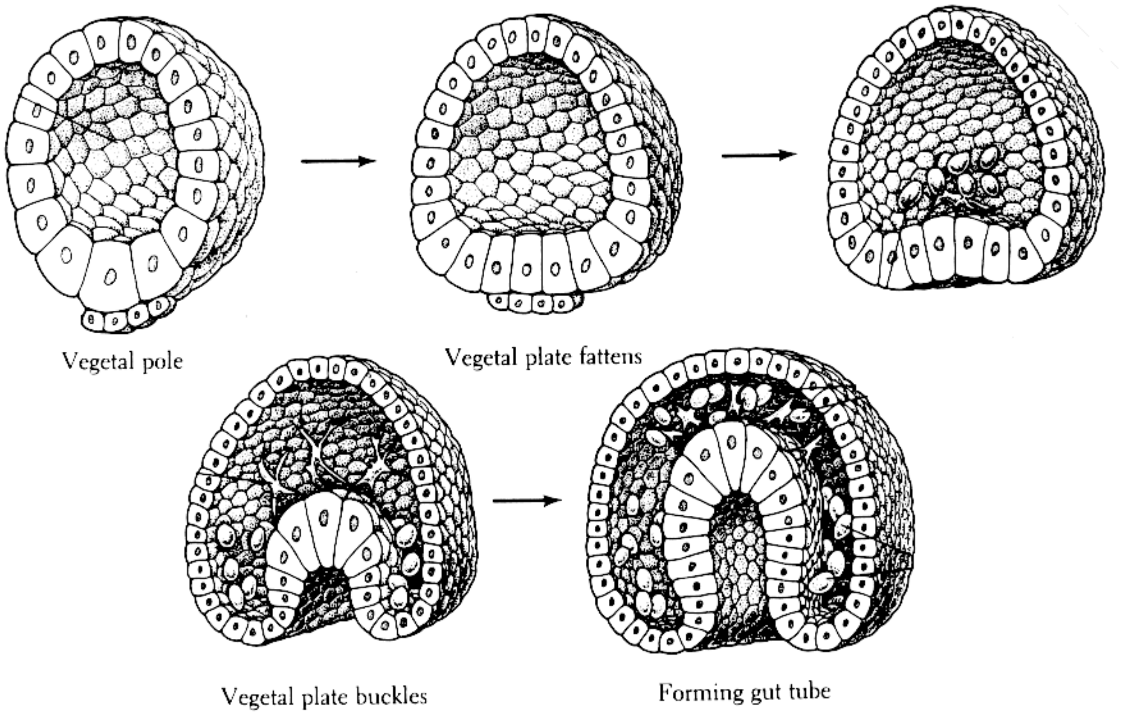


Fig.2-7. The stages in gastrulation of the sea urchin embryo. From (Walbot et al. 1987).

Gastrulation is a process by which the cellular layer of the blastula stage undergoes a substantial reorganization, after which the germ layers are formed. The mechanism of this process is coordinated cell sheet movement. As an illustration of this process Fig.2-7 shows the stages in gastrulation of the sea urchin embryo. As you can see, after gastrulation one can distinguish three layers: an inner layer (endoderm), an intermediate loose collection of cells (mesoderm), and an outer layer (the ectoderm). These three layers are very important in the further process of biological development. In the model, since we will not attempt to model cell movement -a crucial aspect of gastrulation- this phenomenon will not be encountered. Despite the important role of gastrulation in biological development, however, the model will still address many of the

[1] In particular, the interactions in the model will be based mostly on later developmental induction events (see below), whereas the pattern formation in early embryos is not normally referred to as induction.

important phenomena that play a role in development, and of which the most important one is that of induction (see below).

Organogenesis

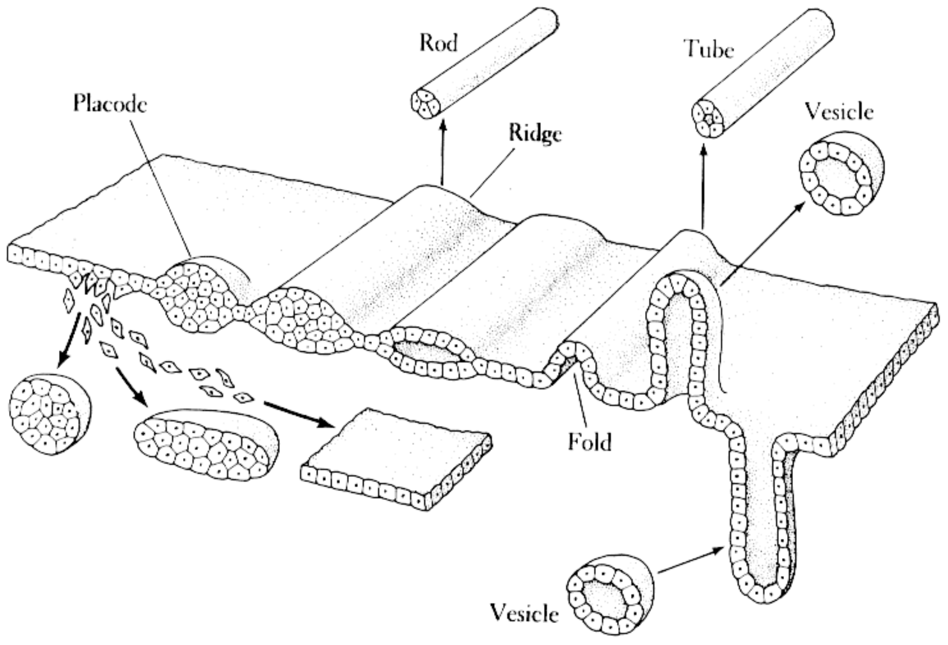


Fig.2-8. Mechanisms of tissue formation. From (Walbot et al. 1987).

Once the basic body plan has been laid out by the emergence of three germ layers in gastrulation, these layers will serve as the basis for the formation of specific organs. In this process, each germ layer gives rise to specific organ types. Thus, the ectoderm gives rise to skin and nervous system, the endoderm forms the gut and most of the organs associated with it, and the mesoderm forms bone and muscle tissue.

The mechanisms by which this happens are (1) differentiation of cells into different cell types, and (2) tissue formation mechanisms, as illustrated in Fig.2-8. As you can see, an important mechanism of morphological change is the coordinated movement of cells, by which folds, pockets, bulges etc.. can be formed. Glands, for example, will in most cases originate from a cell sheet that 'pinches off' a pouch or that invaginates in a fractal manner to form a structure with high surface/volume ratio. Of course, folding does not always have to be an active mechanism: the most familiar example of that is for example the folds in our brain cortex, the result from growth rather than active sheet movements.

Again, since cell movement is not addressed in the model which will be presented, none of these elaborate tissue formation mechanisms will take place during the development of our artificial organisms. As argued in the previous section, this will not undermine the relevance of the model, however. In addition, one could take note of the fact that in plant development, cell movement does not occur, either, and one could scarcely regard plants as uninteresting from a developmental point of view.

Induction in Development

One of the most important phenomena in development is that of induction, by which one group of cells can give a signal to cells in another group nearby. This signal 'induces' the cells in the second group to progress down a different developmental pathway than they would have done without the signal. An important example of induction is neural induction: here, ectoderm (cells that would normally give rise to skin cells later in development) is induced by underlying mesoderm cells to become neurectoderm (cells that will give rise to the nervous system). The mechanism is illustrated in Fig.2-9.

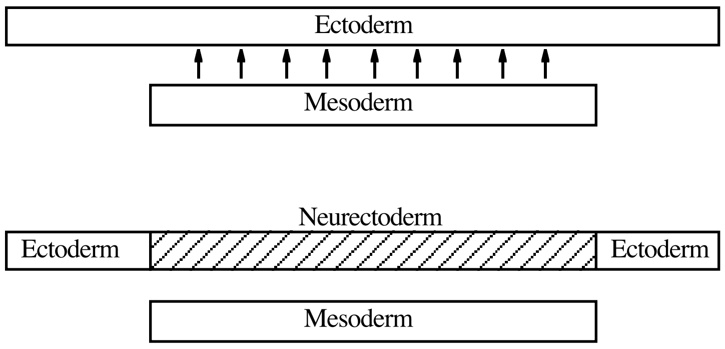


Fig.2-9. The principle of induction, illustrated by neural induction.

Induction is a central component in our model as well, and the inclusion of this mechanism *alone* could give rise to a whole set of interesting questions. Indeed, much of the research in the area of Artificial Life and Complex Systems centers around the fact that systems of many -simple- interacting elements can give rise to emergent phenomena that are more complex than one would suspect from looking at the behavior of the elements alone. Quoting Chris Langton in (Langton 1989):

> *The key feature of [complex systems] is that their primary behaviors of interest are properties of the interactions between parts, rather than being properties of the parts themselves, and these interaction-based properties necessarily disappear when the parts are studied independently.*

Thus, in the same way, development is a complex process resulting from the interactions of many cells, and of which one could say that the sum of the whole is greater than the sum of the individual parts.

Genetic Algorithms

In this section I will briefly describe genetic algorithms. For a more thorough treatment of the subject I would like to refer the reader to (Koza 1980; Goldberg 1989).

Genetic Algorithms

Genetic algorithms (GAs) are optimization algorithms that work according to a scheme analogous to that of natural evolution. Quoting from (Koza 1980, page 17):

> *In nature, the evolutionary process occurs when the following conditions are satisfied:*
>
> - *An entity has the ability to reproduce itself.*
> - *There is a population of such self-reproducing entities.*
> - *There is some variety among the self-reproducing entities.*
> - *Some difference in ability to survive . . . is associated with the variety.*

John Holland (Holland 1975) was the first one to try to apply these principles of natural evolution to artificial systems, more precisely to optimization problems, and came up with the notion of genetic algorithms. A general definition of these algorithms is given in (Koza 1980):

> *The genetic algorithm is a highly parallel mathematical algorithm that transforms a set (population) of individual mathematical objects (typically fixed length character strings patterned after chromosome strings), each with an associated fitness value, into a new population (i.e. the next generation) using operations patterned after the Darwinian principle of reproduction and survival of the fittest and after naturally occurring genetic operations (notably sexual recombination).*

Thus, when confronted with an optimization problem, the first step when using genetic algorithms is to devise an *encoding* scheme that can be used to characterize a possible solution of the problem. These encodings are the mathematical objects that will be manipulated by the GA. An encoding is often referred to as a genome, for reasons that will become clear below. A much used encoding strategy is to represent a solution as a bit string, but real-valued GA's are also commonly used. This is best illustrated with a simple example as the one shown in Fig.2-10, where the dimensions of a wine glass are shown encoded respectively on a real-valued and a binary genome.

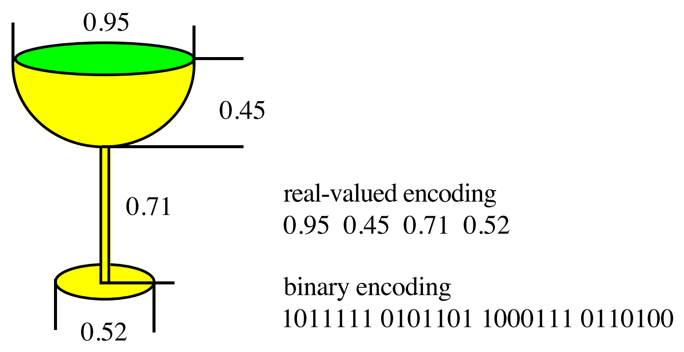


Fig.2-10. Examples of how to encode an instance of an optimization problem into a genetic encoding scheme. The glass shown can be described by 4 real numbers. In real-valued GAs the encoding is just the array of these numbers. In a binary encoding scheme, the numbers are encoded in some way as substrings on a bit string.

The idea behind the genetic algorithm is to take a whole set or population of these genomes, all representing a possible solution to the optimization problem, and to find out which individual genomes represent the best solution. For this purpose, however, we need a performance function, i.e. a way to assign a particular fitness value to each of the individuals in the population. For instance, in the wine glass example, the fitness value of a particular glass might be the volume of liquid it can hold. This can be readily calculated from the dimensions of the glass. Alternatively, one might want to accord higher fitness to a glass which is more pleasing to the eye, which is a much less trivial function.

Once the fitness values of the genomes are calculated, the individuals with highest fitness are selected, and a new complete population of genomes is generated from them. Assuming that a mutation operator is used that produces new genomes with a fitness value correlated to that of the original genome, and since the selected genomes represent the best individuals of the previous generation, it is likely that the newly generated *offspring population* now contains a higher portion of fit individuals. Thus, the central idea behind GAs is that in successive iterations, or generations, of the genetic algorithm the average and maximum fitness of the population increases, until ultimately an optimal solution is found[2].

The offspring population is generated by combining the genetic material of the individuals in this *selected population* and/or mutating some of the genomes to yield variability, using the two genetic operators, mutation and crossover. *Mutation* is simply the creation of variation by

[2] In open-ended evolution, there is no known optimal solution.

changing a small part of the genome in a random way, e.g. flipping one bit on a bitstring, or adding a small random vector to the real-valued genome. Crossover, on the other hand, combines two genomes into one by taking part of the genetic material from the first, and part from the second genome. Both operators are illustrated on the right side of Fig.2-11. It is because these operators are inspired by actually occurring genetic operators that the encoding is termed a genome and that the name genetic algorithm was adopted.

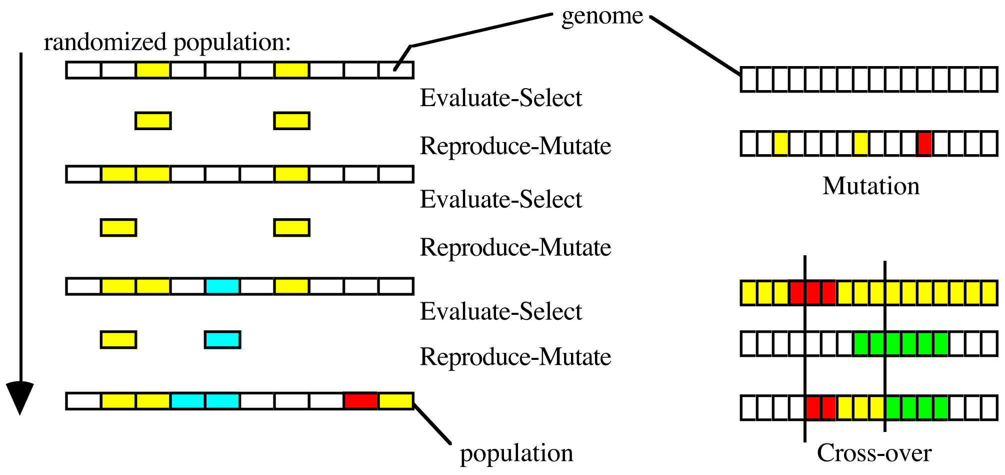


Fig.2-11. Left: the sequence of events in a generic genetic algorithm. Right: graphical illustration of the two genetic operators, mutation and crossover.

In summary, the GA works with a population of genomes, and alternates evaluation, selection and reproduction over the population to search the space of possible solutions for better ones, yielding the following generic algorithm, illustrated on the left side of Fig.2-11:

1. Generate a random population of genomes
2. *Evaluate* the performance of each genome in the population by applying the performance function
3. *Select* the genomes with the best fitness values
4. *Reproduce* using the selected genomes to replenish the population, at the same time producing variation by mutation and/or crossover
5. Go to step 2 or exit when done

There are many variations of genetic algorithms, differing from each other in the way the solutions are encoded, the way selection is handled and how the reproduction step is implemented. In what follows I will discuss the two specific algorithms that will be used throughout this thesis, the local GA and the steady state GA.

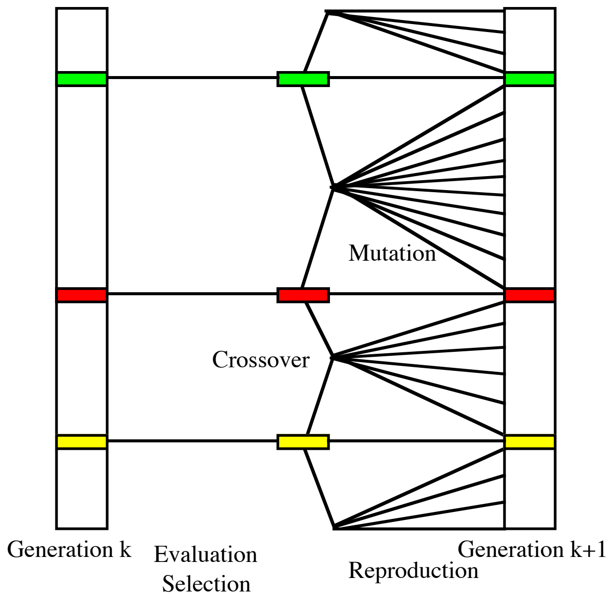


Fig.2-12. Graphical illustration of how the local GA works.

The *local GA* I use is different from the 'classical' GA approach in the sense that reproduction is localized: all the genomes are arranged along one dimension, as shown in Fig.2-12, and after selection the genomes that will become parents retain their position. Selection takes place by dividing the population into a number of adjacent intervals, equal to the number of individuals that need to be selected (as specified by a parameter, see below), and selecting the best individual in each interval. Mutated offspring is generated only in the interval dominated by the ‘parent’ (arbitrarily chosen to be either to the left or the right of the parent), and likewise crossover happens only between two adjacent parents.

I have found this arrangement to be less prone to premature convergence (one genome taking over the entire population) than other schemes. Indeed, when a successful individual is generated, there is the danger that it will quickly spread its genetic information throughout the population, wiping out all possible solutions that have different strategies to arrive at high fitness values. In this way, the variability in the population is lost. This is disadvantageous, because it might be the case that this one solution is stuck in a local maximum, and some of the other solutions that have been lost might have carried the seed for the true optimal solution. By restricting the sphere of influence of a particular successful individual to a certain interval in the population, it will take some time before premature convergence will happen, giving a chance to individuals with a different strategy to establish themselves as true competitors.

An actual implementation of this algorithm depends of course on the optimization problem at hand, more specifically on the encoding that is used to represent a possible solution. In addition, for this encoding, the operators of mutation and crossover need to be defined and implemented. Also, once an implementation is available, a number of parameters need to be chosen before the algorithm can be executed. In the case of the local GA, these parameters are the population size, the fraction of the population that is selected at each step, mutation rate and probability of crossover. The termination criterion used will in most cases be the maximum number of generations for which the algorithm is allowed to run.

The Steady State genetic Algorithm

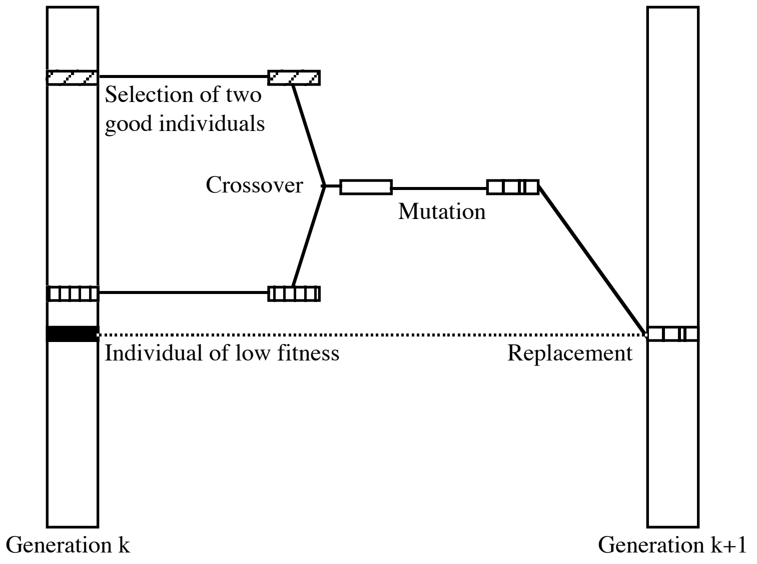


Fig.2-13. The steady state GA, with mutation and crossover are schematically indicated (see text for an explanation).

In contrast with the local GA, where whole populations are processed at once, the *steady state GA* proceeds on an individual per individual basis. As is illustrated in Fig.2-13, at each generation two of the fittest individuals in the population selected using *tournament selection*. In this scheme, a number of individuals are drawn from the population at random, and the two fittest individuals from this sample are selected. They are then used to generate one offspring individual, applying crossover and mutation, in that order. The single offspring organism then replaces one of the less fit individuals in the population, with a probability dependent on the fitness of the individual considered for replacement. In addition, at each step there is a certain probability that the newly generated individual is not used to replace this less fit individual, but that instead a completely new and random genome is used instead. This is beneficial because it opens the possibility to inject new genetic material into the population when a plateau is reached (when there is no improvement in maximum fitness over time).

Again note that an actual implementation is dependent on the encoding used, and that for each run the following parameters must be chosen: population size, tournament size, maximum number of generations, random fraction, mutation rate and probability of crossover.

Steady state genetic algorithms do an even better job than the local GA for maintaining adequate variability in small populations and thus avoid premature convergence, as they replace the genetic material in the population only in a very gradual fashion. Specifically, even with no introduction of randomized genomes and with mutation rate set to zero, I have noticed that using only crossover can maintain high variability in the population for a relatively long time. In addition, the tournament selection scheme gives a non-zero probability of less fit individuals to generate offspring, too, thus excluding the fittest individuals from taking over the population.

Chapter 3

Motivation

This chapter is divided into two sections: the first one discusses the advantages of interposing a model of development between genotype and phenotype in the synthesis of autonomous agents. Since this is the area of research that I am most familiar with, this section is quite large. In contrast, a much smaller section touches upon some questions in theoretical biology that may be addressed using a computer model of development.

Autonomous Agent Synthesis

Introduction: Intelligence as Adaptive Behavior

The view that has long dominated the field of Artificial Intelligence is what we shall call the *classical AI methodology*, whose working assumptions can be summarized as follows (paraphrased from Beer 1990, p.5):

- deliberative reasoning is seen as the hallmark of intelligence
- the manipulation of symbolic representations of the world is the mechanism that underlies intelligent behavior
- the holy grail of generally intelligent behavior in unconstrained interaction with the real world can be attained by synthesizing insights gained from modeling particular aspects of intelligence in restricted domains.

However, this view has met with growing dissatisfaction within the AI community, as expressed most succinctly in (Beer 1990):

> *The "intelligence" exhibited so far by current AI systems is extremely narrow and brittle, depending for its success upon a careful circumscription of the problem domain. Early expectations have not been met, and it is not clear that we are any closer to a deep understanding of intelligent behavior than we were thirty years ago.*

In contrast to classical AI, researchers in the field of *Simulated Adaptive Behavior* regard *adaptive behavior*, i.e. the ability to cope in a flexible way with a complex and ever-changing environment, as the defining characteristic of intelligence. In this view, behavior results from the interaction of the internal dynamics of an intelligent agent with the dynamics of its environment, and this behavior is adaptive if the agent manages to maintain its essential variables within a certain viability zone (Meyer and Guillot 1994). The methodology and credo of Simulated Adaptive Behavior research were elegantly stated by Dave Cliff in (Cliff et al. 1994, preface):

> *Adaptive Behavior research is distinguished by its concentration on the creation and analysis of whole, coping animal-like systems, which -however simple at the moment- may be one of the best routes to understanding intelligence in natural and artificial systems.*

Crucial to this approach is the assumption that deliberative reasoning processes are thought to have evolved in a continuous fashion from the mechanisms that allowed our evolutionary ancestors to react adaptively to their complex environments. Thus, the holy grail of general intelligence is approached from the bottom-up, in an incremental fashion: before anything else, we should try to synthesize simple, autonomous agents and gain insight into how they cope and adapt to either a simulated world or the real world (when the focus is on robotic agents).

The Synthesis of Autonomous Agents and Genetic Algorithms

In many cases, the design of these autonomous agents is done by hand. However, especially in the case of robotic agents, this is a complex task that typically takes up a lot of time and effort: both the physical implementation of robotic agents as well as their control architectures present increasingly difficult design challenges as the complexity of the system at hand increases.

Thus, a number of people (see for example (Beer and Gallagher 1992; Harvey, Husbands and Cliff 1993)) have argued it might be advantageous to try a different approach by using evolutionary methods. In this approach, the control system and/or implementation details of the agent are encoded in a genetic representation, defining a genotypic search space. This search space can then be searched by using genetic algorithms as explained in Chapter 2.

However, most of the work involving genetic algorithms to automatically design autonomous agents employs some form of 'direct mapping' between the design and its description on the artificial genome used by the GA's: typically there is a one-to-one correspondence between some substring on the genome and an associated parameter or feature in the final design. Consider, as an example, an approach where the topological structure of a network is specified a priori and the GA is then used to find the correct weights between the neurons (Beer et al. 1992). The mapping from genome to network in this case is quite straightforward: an a priori defined sub-section of the genome (in this example a bitstring) corresponds to a certain weight in the network. The topological structure is not encoded in the genome.

This approach brings with it a number of problems:

(1) The approach does not scale well: for every parameter in the design one needs to allocate some space on the genome, because of the typical one-to-one correspondence used. It is obvious that as the system to be evolved increases in size and complexity, the search space of possible parameters assignments (genotypic space) grows combinatorially with this increase. Indeed, for every additional parameter you add the size of the search space is multiplied by the number of possible values that this parameter can take on. For example, when using a binary genome, every additional bit on the genome will double the search space.

(2) The structure of the solution is often determined beforehand. In the example above, the architecture of the network is already given and only the parameters need to be optimized. Although it is clear that there will always be constraints on the possible designs to be found by the GA, what is meant here is the degree to which this architecture is amenable to variation: in the example, the possibility of changing the number of weights would add some flexibility, changing the number of layers even more, and changing the overall structure of the net, e.g. from feed-forward to recurrent would bring with it even higher flexibility. In addition, one could argue that by providing an a priori architecture, most of the problem has already been solved and that, moreover, the chosen architecture will probably apply to a small class of problems only, and not generalize very well.

(3) In many cases the number of parameters to be optimized is fixed a priori. In the example, there is no variability in the number of weights that have to optimized, as the number of neurons and connections are specified beforehand. This fixed dimensionality limits the GA to finding optimal solutions in a finite search-space of fixed size. The role of the genetic algorithm is thus reduced to searching a large number of possible designs and finding an optimal one, like a hill-climbing algorithm but then geared toward complex fitness landscapes. In contrast, were the number of parameters variable, the genetic algorithm could conceivably be used to incrementally improve working designs by adding new variables to the system and optimize the extended design thus obtained. Of course, the architecture of the design should lend itself to such an open-ended evolution scheme. An approach that does exhibit this ‘variable complexity’ is the SAGA architecture, discussed in (Harvey 1994).

A Developmental Alternative

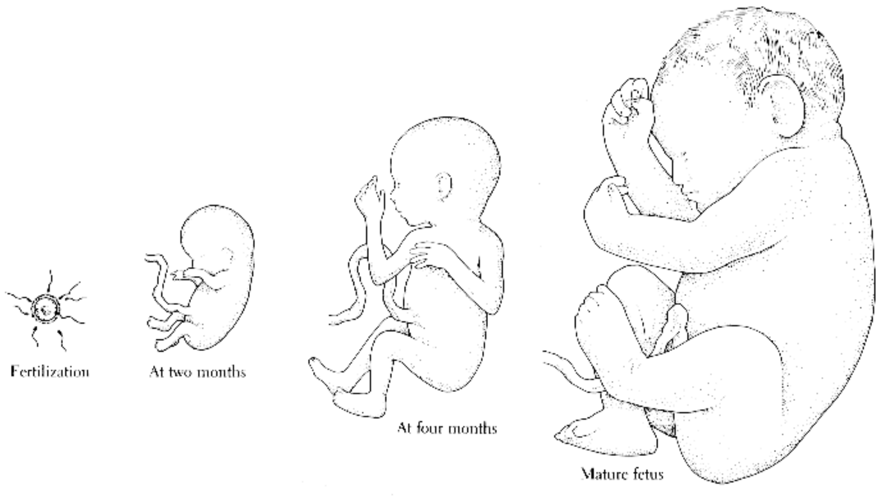


Fig.3-1. In biology, development starts with the fertilized egg, and after a complex growth process a complete organism emerges. From (Walbot et al. 1987).

The previous section concerned itself with the problems of the direct mapping approach that is used so often. Here I argue, as others have done before me, that adding a 'developmental' component to the evolutionary process addresses some of these problems and possibly provides a better way to approach the automatic design of complete autonomous agents.

At this point it is useful to introduce the important distinction to be made in living organisms between *genotype* and *phenotype*. Quoting from (Langton 1989):

> *The genotype is the complete set of genetic instructions encoded in the linear sequence of nucleotide bases that makes up an organism's DNA. The phenotype is the physical organism itself -the structures that emerge in space and time as the result of the interpretation of the genotype in the context of a particular environment. The process by which the phenotype develops through time under the direction of the genotype is called [development].*

In biology, there is no direct mapping between genotype and phenotype. Rather, plant and animal morphologies are the result of a growth process that is directed by the genotype. For example, in Fig.3-1, the familiar developmental sequence of a human is shown: the process starts with a fertilized egg, that starts dividing and dividing, and a complex growth process ensues resulting in the adult (since development is a process that continues after birth). The genotype does not specify the dimensions of each body part in the adult, but rather contains information on how the growth process takes place.

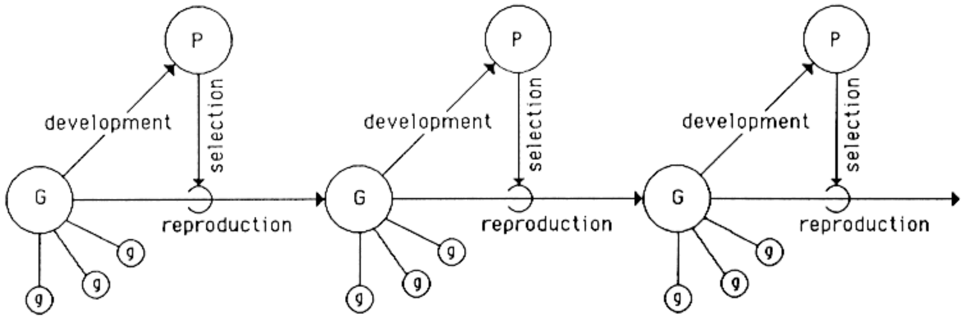


Fig.3-2. The relation between evolution and development. From (Langton 1989).

Thus, natural evolution plays out along two orthogonal axes: along the *evolutionary time axis*, the genotype undergoes change: the genotype is that which is passed on from parents to offspring in successive generations, and it is on this level that recombination and mutation takes place to yield variation in the genetic material. For each individual of the successive generations in the evolution of a species, however, the genotype specifies a developmental process by which the organism emerges, and which plays out along the *developmental axis*. The success of that organism in its environment determines whether that genotype will carry over into the next generation: thus, it is at the phenotype level that selection takes place. This interplay between evolution and development is indicated in Fig.3-2.

When using genetic algorithms to design some system, the same two axes may be considered. The evolutionary axis is present in all versions of the genetic algorithm: it is simply the succession of the evaluation, selection and reproduction phases that are at the core of all evolutionary algorithms. However, in most cases the developmental axis is virtually non-existent: the function that maps the genotype to the phenotype is frequently a trivial one-to-one mapping from substrings of the genotype to parameters in a fixed design, i.e. what has been termed the direct-mapping approach above.

Many people have argued that there are important advantages to be gained if one would replace this trivial direct mapping function by a more complex, developmental function, a growth process that results in the phenotype. The genotype should then be viewed as directing this developmental process, just like its biological equivalent. This thesis is partly about how we can make these advantages come true for the automatic design of autonomous agents and their neural network control systems.

Advantages of a Developmental Approach

Where the previous section set the stage for the introduction of a developmental mapping function in the classical genetic algorithm picture, this section specifically addresses what advantages could be gained from such an approach.

A developmental specification encodes rules in the genome, not just parameters. It can be viewed in a way as a (structured) computer program, to invoke a much used metaphor. As an example, suppose you want to print out the numbers from 1 to 100: to do this, you can either write a program using a for-next loop or one that contains 100 individual print commands. The former provides rules to execute the task and the execution of the program can be compared with development, which in a sense is also the interpretation of rules. The latter program just provides an exhaustive list of what to do, and can be compared with a direct mapping approach.

The computer program metaphor is useful to explain some of the advantages of a developmental approach, which will be elaborated on below. However, before going any further, it is important to realize that the computer program metaphor should not be taken too far: there is for instance no single master program that directs the developmental process. Instead, the process of development emerges as many simple agents interact with each other, yielding a more complex process than one might suspect from their *relatively* simple behavior. The emerging collective behavior of the cells then constitutes the developmental process.

A compact genotypic encoding of complex phenotypes

When interpreted not as a direct encoding but as a set of developmental rules, a genetic description can lead to much more complex morphologies than achievable with direct mapping. As illustrated by the for-next loop example used above, it is obvious that it constitutes a much more compact specification of the task at hand than the exhaustive list of simple commands doing the same thing. This argument can be strengthened with an example from biology: the number of neurons in our brain, estimated at 10^{11} (Kandel, Schwartz and Jessell 1991, page 18) greatly exceeds the number of individual genes that are carried in the human genome, estimated at about 150000 (Walbot et al. 1987, page 144). It is thus impossible that each neuron, let alone the connections between them, is coded for by individual genes. Rather, there will be rules in the genome that specify how - and when - neurons grow and how they should be connected together.

Substantial phenotypic mutations by simple genotypic mutations

The computer metaphor is also helpful in illustrating how mutations in a rule-based specification of some phenotype act very differently from a direct-mapping encoding: for example, the loop example can be easily modified to print out 200 numbers instead of 100. A simple mutation thus leads to a drastic change in the phenotype (if we regard the output of the program as the phenotype). In biological development, such large mutations of phenotype can result from (even small) changes in the timing of different processes and/or the rate at which certain molecules are produced, for instance.

Naturally provides a way to try out a spectrum of small and large scale mutations

Development plays out over a period of time, with some genes governing early events and some genes only -predominantly- active during later development. Thus, one can expect that mutations in genes acting early in development, affecting the building blocks for later development, will have larger consequences for the phenotype than genes acting at a later time that will have no cascade effects on later events (since they come last). In addition, the later genes are more likely to act in restricted parts of the body, being expressed for instance in certain tissues or organs only. Quoting from (Buss 1987):

> *It is axiomatic that a random alteration introduced early in ontogeny [i.e. development] will likely be manifested in a cascade of subsequent morphogenetic events, whereas a modification introduced later in ontogeny can have relatively minor effects.*

Thus, development naturally provides a way to try out a spectrum of mutations, ranging from early gene mutations with large phenotypic effects to late gene mutations with smaller effects. In (Kauffman and Levin 1987) such a range from local hill-climbing operations to long jumps beyond the correlation length of the genetic search space[3] is argued to be beneficial for aiding optimization by evolutionary processes. This is easy to see: the long jumps introduce a certain randomness in the search, while the hill-climbing takes you to a nearby optimum.

---

[3] When they talk about correlation length, they mean the average magnitude of the mutations that take you with high probability to a nearby point in genotypic space with comparable fitness. To be exact, developmental long jumps take you beyond the correlation length in *phenotypic* space.

Incremental building

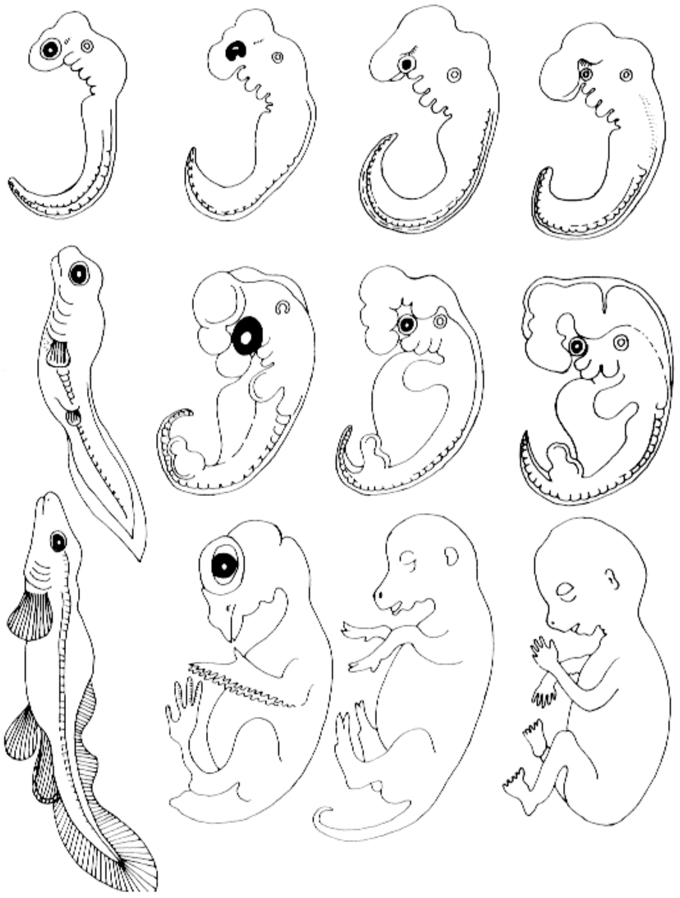


Fig.3-3. Likeness in embryos of different species. From (Raff and Kaufman 1983).

Development is ideally suited to build upon previous discoveries in an incremental fashion: to realize how this has been used during our own evolutionary history, consider Fig.3-3, where embryos of several different species have been compared at different stages of their development. It is most striking that they are very similar in their initial stages, and that differences arise only later in development. This serves as an excellent illustration that evolution takes whatever is there and modifies it to serve its purposes.

Easy specification of modular designs

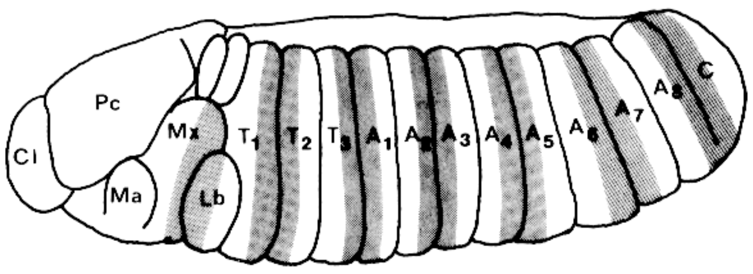


Fig.3-4. Segmentation pattern in early *Drosophila* development.

Easy specification of a modular design is another advantage of a developmental specification (Gruau and Whitley 1993; Sims 1994).It is analogous to the use of subroutines in computer programs: a subroutine-like construct in development could be used to repeat growth of several instances of the same module for a number of times. The classic example in biology is of course that of segmentation: many complex life forms (e.g. insects, worms, vertebrates) have several repeated body segments (see Fig.3-4), whose origin is determined by homologous genes in all these different species. In some species this subroutine is executed more times than in others, resulting in different numbers of segments, e.g. a centipede has many more segments than a fly.

Using a Developmental Specification Based on Morphology

If, specifically for autonomous agent synthesis, we use developmental specification of *body morphology* as the basis for subsequent evolution of neural control systems, a number of additional advantages are gained.

Symmetry

An important issue in the context of autonomous agent design is that of symmetry: in many applications in adaptive behavior research, symmetrical layout of body patterns and neural control systems is highly desirable. Whereas in most current work this desired symmetry has to be artificially hardwired in the specification of their artificial genome (see for example (Beer et al. 1992)), it comes virtually for free in a morphology-based developmental model. This is explained in more detail when the specifics of the model are presented.

Constraint by body morphology

Many authors (Miller, Todd and Hegde 1989; Kitano 1990; Gruau et al. 1993) address only the development of neural networks. In contrast, the developmental model proposed here is conceived from the start as based on a physical body morphology, to be extended later towards neural development. In this way, the breadth of phenomena that can be explored and/or mimicked using the model is not constrained to neural networks. In addition, the development of neural architectures can benefit when a morphological substrate is available as a scaffolding but also as a system introducing physical constraints: for example, it might be very useful that not every neuron can be connected to every other neuron. Indeed, these and other reasons prompted Kitano to extend his L-system-based model of development towards morphogenesis as opposed to pure neural networks (Kitano 1994).

Interaction between mechanical and genetic factors

As others have argued before me, the interaction between genetic and mechanical factors can be a critical element in determining developmental patterns (Fleischer and Barr 1994). Thus the importance of having a mechanical model of body morphology, however simple it may be (see Chapter 5).

Allows for co-evolution of body morphology and nervous systems

The fact that the development of body morphology underlies neural development in the model opens the intriguing possibility to co-evolve bodies and nervous systems in autonomous agents. Nervous systems should be able by their developmental origins to adapt to an evolving body, in times when that is under selective pressure to change. Conversely, body morphology might adapt to new capabilities of nervous control. A successful example of this co-evolutionary approach can be found in (Sims 1994).

Using a developmental model greatly favors co-evolution: one of the great strengths of a developmental specification is that rules of growth are coded in the genome rather than exact blueprints: if the body morphology undergoes a sudden change, the nervous system can still be wired up correctly anyway. In biology there are examples where an extra leg is grafted onto a chick embryo in an abnormal place, and still the extra limb is innervated correctly by the developing nervous system, as discussed in (Purves and Lichtman 1985, page 119). Of course, there are also numerous examples where the opposite is true, and where a small change in body development due to a genetic mutation can lead to disastrous consequences for the developing nervous system.

Thus, new innovations in body design are given a chance to be at least partly functional in a newly evolved organism, as the developmental specification for the nervous system will try to adjust to what it has to work with. Then, after this first drastic innovation, the nervous system can undergo a number of incremental changes to adapt optimally to the new morphological feature. Where before it was on an (at least locally) optimal point in genetic search space, the new body feature has given it new and ample room to improve. As you can see, the notion of coupled fitness landscapes holds quite well here (Kauffman and Johnson 1992).

Conversely, bodies can adapt to better nervous systems. For some reason, the nervous system might get better at something, e.g. as a side-effect of an adaptation of the nervous system to a new environment. The same developmental innovations that help vision improve, for example, might trigger an improvement in the processing capability of olfaction. Thus, more processing power available for smell interpretation will make it advantageous to evolve a better nose, increasing the adaptiveness of the species as a whole.

Some Questions in Theoretical Biology

There are many open questions in theoretical biology related to development that are subject to intense debate. Here I will suggest some areas and questions that might benefit from the use of a simulated developmental process, even if it is not a biologically realistic one.

As a general point, many of the potential advantages of a developmental specification that have been discussed in the previous section can be restated in the form of a question: is it really the case that evolution makes use of these advantages and has it really happened throughout evolutionary history? Simulation might be a way to help answer this question, but it can also provide us with more precise and testable questions and hypotheses.

Embryos, Genes and Evolution

One of the properties of development (already mentioned above) is succinctly pinned down in (Gould 1980):

A small genetic variation might account for a large change in phenotype.

Does evolution benefit from this, or is it merely distracting and/or destructive ? For example, since because of this property of development, mutations can take place at many different scales, ranging from far-reaching to minor details, one would suspect (as indeed already mentioned above) that this is advantageous to the evolutionary process. Simulation experiments might confirm this, at least for the simple artificial organisms we are considering.

Heterochrony

One of the mechanisms that explain why the previous point holds true, is called heterochrony, i.e. how small changes in the timing of developmental processes can produce large variations in the resulting adult organism. To illustrate this, consider the following passage from (Gould 1980) on Goldschmidt's work with the gypsy moth:

> *[Goldschmidt] found that large differences in the color patterns of caterpillars resulted from small changes in the timing of development: the effects of a slight delay or enhancement of pigmentation early in growth increased through ontogeny and led to profound differences among fully grown caterpillars.*

Since in a model we have all parameters under control and we can measure every aspect of the simulated developmental process, simulation might provide an excellent means to study this phenomenon. For example, one might want to examine under what evolutionary conditions events are postponed to a later developmental time or when they are advanced, or whether there is indeed a preference for one or the other.

Gradualistic vs. Punctuated Evolution

Related to this point is the role that development plays in species formation, a hot topic in biology. In recent years, the view that evolution takes place in an extremely gradual way has made way for a new view, as summarized in the preface to (Raff et al. 1983):

> *The essential position is that there is a genetic program that governs ontogeny, and that the momentous decisions in development are made by a relatively small number of genes that function as switches between alternate states or pathways. The significance of this view, if correct, is that evolutionary changes in morphology occur mechanistically, as a result of modifications of these genetic switch systems. If our prediction that there are a relatively small number of such gene switches is correct , then the potential exists for geologically rapid and dramatic evolutionary changes. Such macroevolutionary events are apparently associated with the origins of new groups of organisms.*

Ever since the controversial geneticist Richard Goldschmidt laid the foundation for this idea in the 1940's (the terms 'hopeful monster' and 'micro and macro-evolution' have been coined by him, see (Goldschmidt 1940)), the debate has been raging between proponents of a view related to this one, and those that favor a much more gradualistic view of evolution. Simulating the evolution of species in artificial organisms, combined with some model for how these organisms develop, might provide additional data to fuel the debate. Of course, it cannot resolve the question one way or the other, merely demonstrate its plausibility in an artificial system.

Recapitulation

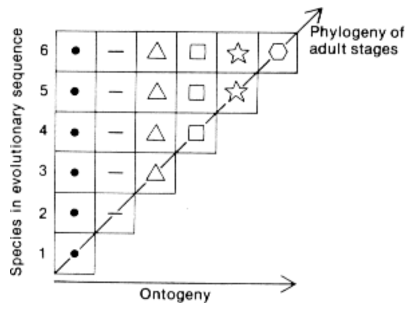


Fig.3-5. The (discounted) theory of pure recapitulation: as time goes by, evolution proceeds by adding new stages to the developmental process. From (Gould 1977).

An even more controversial debate centers on the theory of recapitulation, illustrated in Fig.3-5, which was proposed by Haeckel as far back as 1879 (see (Gould 1977; Raff et al. 1983) for an extended discussion). Long in disfavor since the analytic approach to developmental biology became rule, this theory stated that ‘ontogeny recapitulates phylogeny’, i.e. that the developmental process in a certain organism is actually a replay of the evolutionary history of the organism’s species. Although the extreme version as expressed by Haeckel’s ‘biogenetic law’ is accepted by nobody anymore, the concept seems to be more valued recently than it used to be, especially since the renewed interest in the role of development in evolution. Again, a model might help here by providing a new, experimental system to address questions regarding this phenomenon. Indeed, at least one developmental model of artificial organisms has been used to do just this (Nolfi and Parisi 1993), as we will see in the related work section.

Chapter 4

Modeling Genetic Regulatory Networks

Cell Behavior and Development

In this section I will argue that biological development is an emergent property of cellular behavior. We will see that cellular behavior is a consequence of which structural genes are expressed in the cell, and that this in turn is dependent on the developmental pathway that the cell has traversed. Finally, we will arrive at the concept of genetic regulatory networks that govern these differentiation pathways.

We are interested in building a model of an abstract developmental process in artificial organisms. However, in building it, we want to be able to take inspiration from biology and we want the model to bear some relevance to questions from theoretical biology. So, we will now look towards biological development and try to learn something about how we should attack the problem.

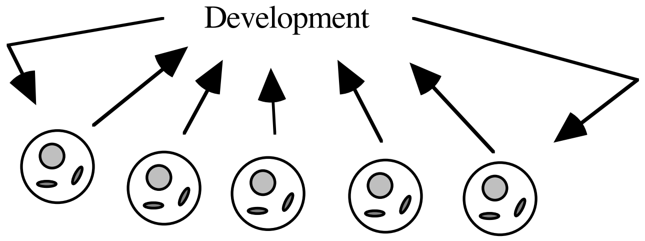


Fig.4-1. Development can be seen as an emergent property of the interaction of cells.

The basis for development is the behavior of cells. All events in the developmental process can be looked upon as the decision of a single cell or a group of cells to behave in one way or another. Examples are numerous: cells differentiate, start to express different genes, move to certain places in the embryo, stick or do not stick together, communicate; cells even die for the benefit of development. Different cells will make different decisions as to how to behave, depending on their internal state and on the state of their environment. This pattern of decision-making repeats itself throughout development. Induction, for example, is simply the fact that a group of cells makes a decision to go down a different developmental pathway in response to a signal from other cells. In conclusion, the process of development is the emergent property of the interactions between many different cells (Fig.4-1).

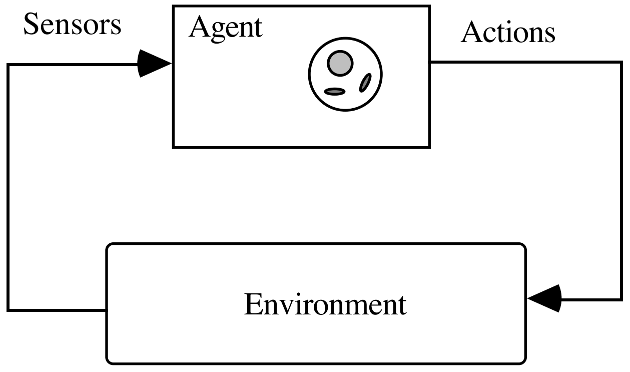


Fig.4-2. One could borrow terminology from SAB and view the cell as an animat.

It is interesting to note that this view of the cell as a behavioral agent is in complete accordance with the definition of an animat in the field of Simulated Adaptive Behavior. This view is depicted schematically in Fig.4-2: the animat or agent senses its environment by means of sensors, it makes a behavioral choice taking into account this sensory information but also its internal state, and consequently performs an action (or not) which will in turn affect the environment. Much of the research in that field is centered on the mechanism by which a certain behavior is selected and how it is generated, and we will ask ourselves the same question in the case of the cell.

Since the behavior of cells plays such a crucial factor in development, it makes sense to start off the modeling process with a model of an artificial cell. These artificial cells will then make up the artificial organisms. Thus, we are interested in the underlying mechanisms of cellular behavior, and we will address that shortly. It has to be realized, however, that in the end we will only be interested in the phenomena that are important for development. In fact, we will soon limit our attention to the mechanism of differentiation only.

Having an abstract cell model is also in line with an as yet unstated goal, i.e. the fact that the model should be conceived as based on morphology right from its conception. Since our artificial cells will have a spatial extent, we will automatically have a body for our artificial organisms. This will become important when neural development is modeled (see below).

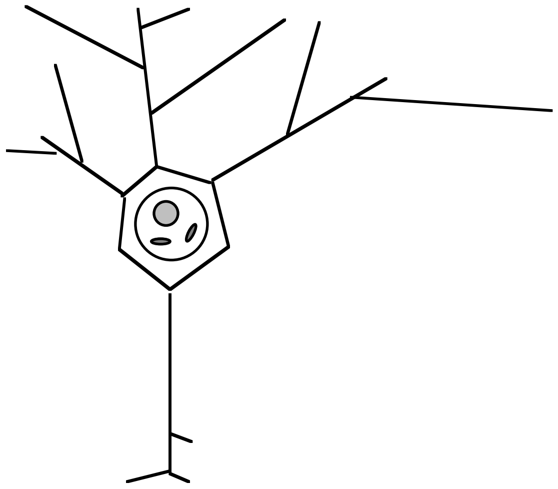


Fig.4-4. Structural genes are needed to endow the cell with the necessary equipment to exhibit the correct behavior. In this schematic neuron structural genes are needed for the cell to have an axon, dendrites, synapses, neurotransmitters, ion-channels and so on.

The way cells behave is largely an immediate consequence of which *structural genes* are expressed within the cell. Structural genes code for the proteins and enzymes that make cell behavior possible. Proteins provide many of the building blocks that cells are made of, and consist of strings of amino acids that are arranged in a particular order, as encoded on the genome. Enzymes are also proteins, with the added property that they are catalysts to the chemical reactions that are crucial to the cell. Some structural genes code for building materials, for instance the proteins that make up the cytoskeleton. Other structural genes code for proteins and enzymes that are responsible for a whole range of housekeeping functions, such as the cell's metabolism. Many other structural gene products play very active roles in diverse cellular behaviors ranging from cell movement in scavenging cells to action potentials in neurons. Proteins coded for by structural genes can function as receptors, ion-channels, messenger proteins, enzymes and so on.

Cell behavior resulting from the actions of the structural proteins is quite complex. Concentration levels of gene products play an important role and exhibit complex nonlinear dynamics. Genes and gene products are being up and downregulated all the time, especially in the adult body when cells take up their specialized functions within the organism. In complex behaviors like cell movement or the regulation of the cell cycle, concentration levels and levels of expression will vary in a complex manner over time, to say nothing about what goes on structurally.

The picture sketched above is not entirely true. It suggests that the set of genes is expressed in a certain cell is constant and does not change over time. Obviously this is not true during development, which we will discuss in a little while, but even in cells of the adult organism changes in gene expression are an integral part of the mechanism underlying behavior.

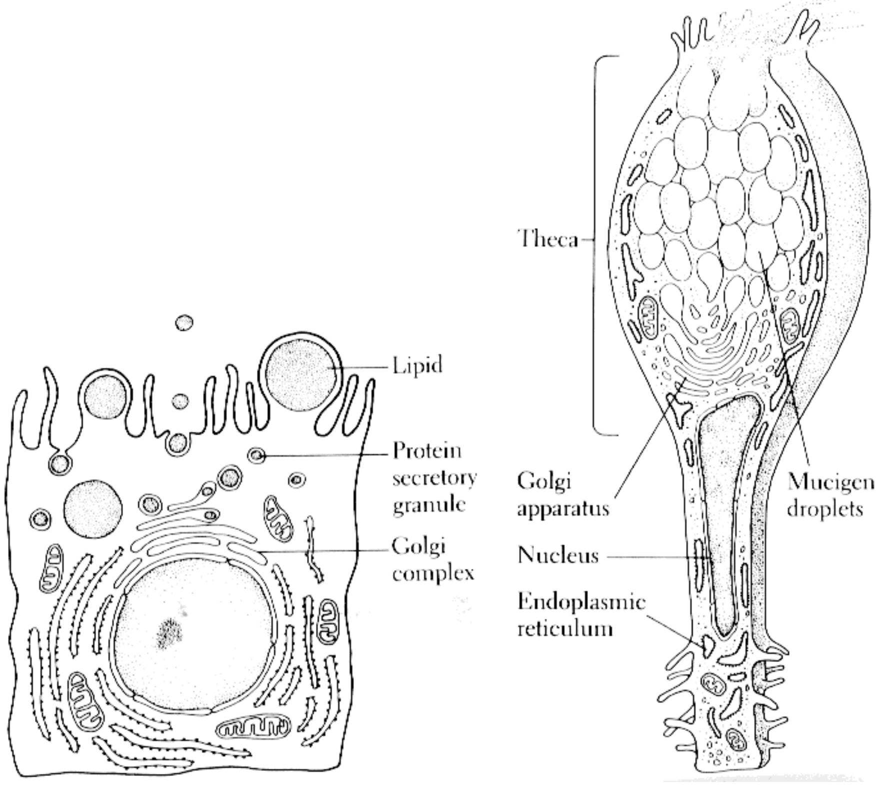


Fig.4-6. Two different cell types that are found in the adult human body, a mammary gland cell (left) and a goblet cell (right). From (Walbot et al. 1987).

Different cell types will have different sets of structural genes expressed. That is in fact the meaning of differentiation: cells go down a developmental pathway, and at the end of the pathway they will express the set of genes that is suitable for the function the cells will take up in the organism. A neuron expresses structural genes for neurotransmitters or ion channels. A gland cell expresses genes that code for the hormones it produces. Likewise, skin cells express the genes for a certain protein (keratin) that will make it suitable for the harsh conditions it will encounter during its lifetime. For illustration purposes, two different cell types are shown in Fig.4-6, expressing a different set of structural genes.

Differentiation of cells into different types is what interests us most at this point: since in development differentiation is a crucial process, we need to know how different sets of structural genes are turned on and off. As it happens, the expression of structural genes is regulated by a special set of genes called the *regulatory genes* (see Fig.4-7). These code for proteins (or RNA molecules) that influence the expression of the structural genes, and of other regulatory genes.

They can be switched on in response to environmental clues and/or the expression of other regulatory genes, and they are responsible for the correct and timely expression of all other genes.

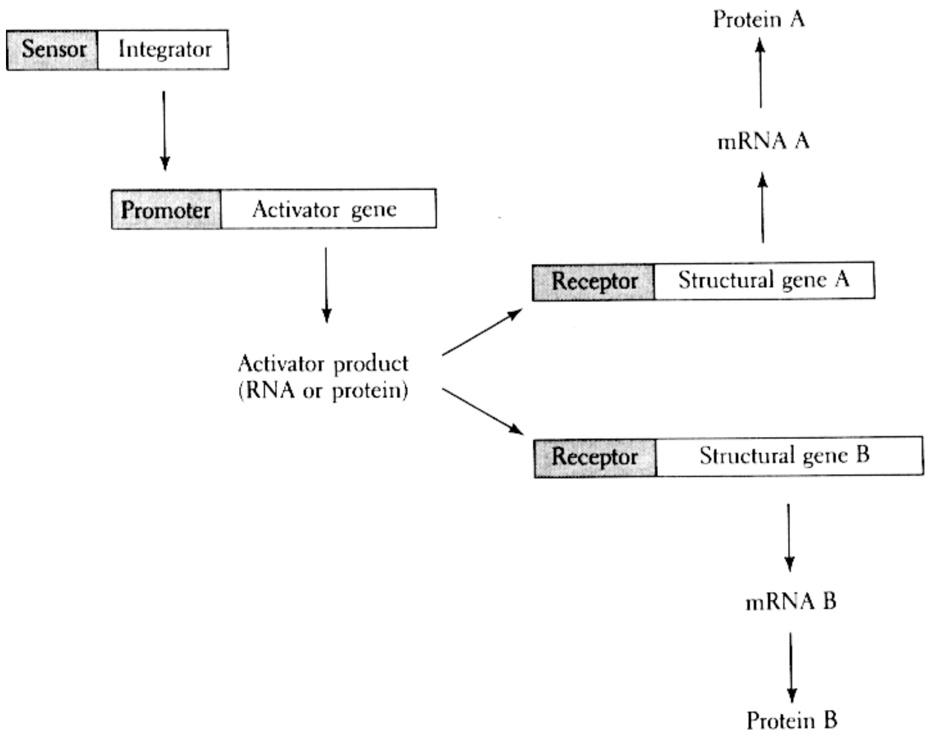


Fig.4-7. The Britten-Davidson model of gene regulation during development. The structural genes A and B are regulated by regulatory genes (see also text). From (Walbot et al. 1987).

However, some classes of structural genes play a large role in gene regulation as well. For example, cell surface receptors sense signals from other cells, or the environment of the cell, and pass that signal on to the regulatory genes, via other structural gene products such as second messenger proteins. This signal transduction pathway involves membrane-associated proteins, enzymes, nuclear proteins and others, all coded for by structural genes.

Thus, we can state that regulatory and some structural genes and gene products form a complex network of interacting elements, and the dynamics of this network underlie the differentiation of cells into cell types. We will call these networks *genetic regulatory networks*, as this term has already been coined in a similar context by Stuart Kauffman (Kauffman 1969). Note that we make no claim about biology here: it is simply a way of looking at what happens in the biological cell to regulate gene expression. Nothing has been said yet about how the elements in the network influence each other, or about the actual nature of the elements. Also, keep in mind that these elements are not merely genes, but also gene products, i.e. enzymes, proteins, receptors and all other molecules that play a role in the regulation of genes. See Fig.4-9 for an illustration.

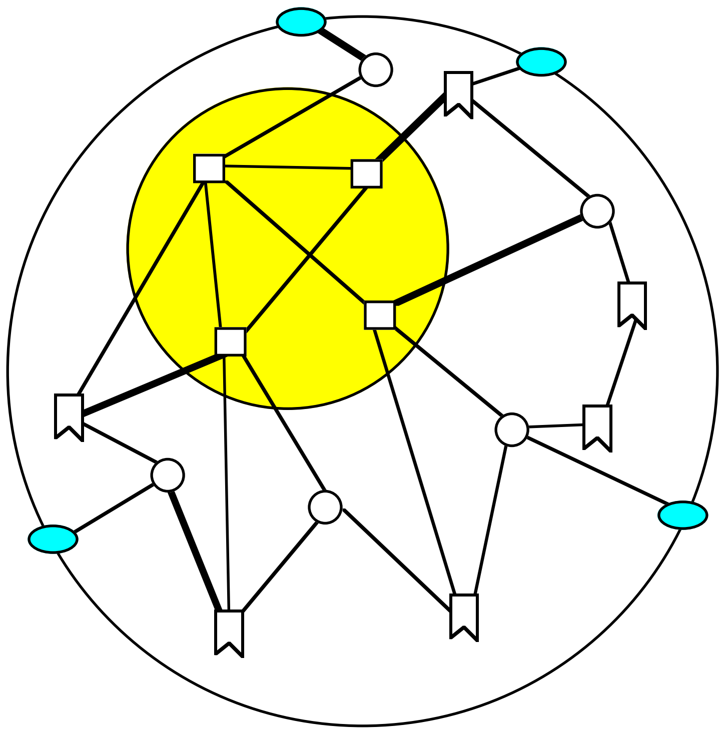


Fig.4-9. A genetic regulatory network. The symbols represent the many different elements that play a role in gene regulation. The lines represent the different ways in which these elements influence each other.

The genetic regulatory network is a concept that we define to apply to the regulation of gene expression, not to cell behavior in general. The structural genes and/or gene products that have no effect on gene expression in a direct or indirect way are not included as elements in the network. They can be thought as being regulated by this network, but are in a manner of speaking 'downstream' elements, i.e. they do not feedback into the network (if they did, they would be, by definition, included in the network). It is this genetic regulatory network that we will model in the next section by random Boolean networks, and consequently we will not have built a model of general cell behavior, only a simplified model of gene regulation.

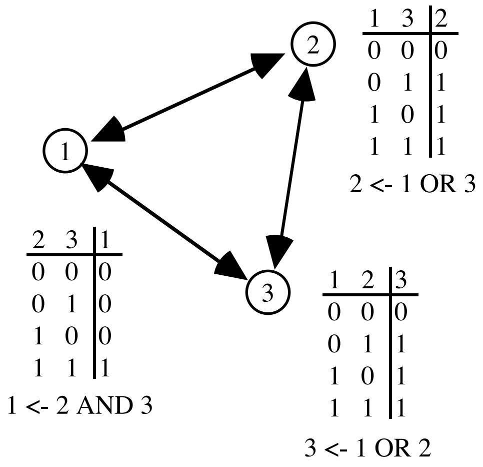
The Random Boolean Network Model

Since we are interested in modeling a developmental sequence in our artificial organisms, we want to arrive in this section at an abstract cell model, and give it a mechanism by which it can go through a developmental pathway, i.e. a differentiation process. In the last section, it became clear that gene regulation in cells is realized by means of what we have termed a genetic regulatory network. Thus, we'll have to think of a way to model those networks in a biologically defensible way. Note that we have already taken the network of interacting elements from the biological example, and thus we have already gone some way towards a biologically inspired model. In contrast, many developmental models use grammars that in fact are syntactic descriptions of developmental phenomena at a higher level. We have already made the decision here to go deeper into the underlying dynamics.

One overriding concern must be stated beforehand: as this model of genetic regulatory networks is going to be the core of the developmental model, it will be the determining factor as far as computational complexity is concerned. In the following chapter, the model we propose here will have to function in every (model) cell of an artificial multicellular organism. One such organism may comprise up to several hundreds of cells, or even more. In addition, many of these organisms will constitute a population that is being evolved by genetic algorithms, for many generations. Thus, the model to be developed here needs to be as simple and computationally efficient as possible.

The Model: A Random Boolean Network

It was decided to model the genetic regulatory network using a random Boolean network (RBN), as first pioneered in this context by (Kauffman 1969) and extended by (Jackson, Johnson and Nash 1986) to systems of multiple, communicating networks. This model is both readily understood, efficiently implemented and easily analyzed (Wuensche 1994), in contrast with the more complex continuous time dynamical neural networks as used by (Mjolsness, Sharp and Reinitz 1991). Of course, this choice will have to be defended on better grounds than convenience alone, showing that we do capture the crucial properties of what we are trying to model. However, let us first explain random Boolean networks, and then describe how they map to the genetic regulatory networks.



1	3	2
0	0	0
0	1	1
1	0	1
1	1	1

2 <- 1 OR 3

2	3	1
0	0	0
0	1	0
1	0	0
1	1	1

1 <- 2 AND 3

1	2	3
0	0	0
0	1	1
1	0	1
1	1	1

3 <- 1 OR 2

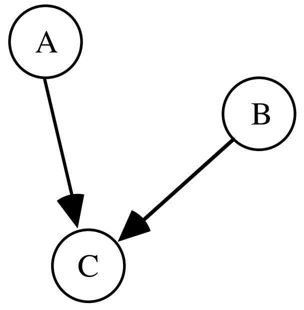
Fig.4-10. A simple random Boolean network. Taken from (Kauffman 1993).

A random Boolean network is a collection of nodes. Each node is characterized by a binary state variable, the node’s value, say 0 or 1. There are N nodes in a network. Each node is connected by incoming edges to K other nodes whose values constitute the node’s inputs. The edges can be recurrent, i.e. nodes can connect to themselves. Each node synchronously updates its value in discrete time steps. The value of a node at time t+1 depends on its particular Boolean function applied to the value of its inputs at time t.

Thus, these networks behave much like cellular automata, but where in the latter the neighborhood of a node is fixed - typically the direct neighbors of the node - and the updating rule is the same in all nodes - a uniform rule -, there is no such restriction in a random Boolean network. Each node has its own associated pseudo neighborhood of size K and its own updating rule. Classical cellular automata are thus a special case of random Boolean networks. Note that the qualification ‘random’ does not imply that there is any randomness in determining the next state of the network: it refers simply to the fact that the updating rules and pseudo neighborhoods may be different for each node, i.e. random instead of fixed. A more formal definition of this type of networks can be found in (Wuensche 1993), where they are discussed at some length. An example network is shown in Fig.4-10.

The mapping between genetic regulatory networks and these random Boolean networks is simply:

- Each element is modeled by one node in the random Boolean Network. We will talk about a *genetic element* in what follows, i.e. a node representing an element in the genetic regulatory network
- The regulation of an element in the genetic regulatory network is modeled as a Boolean function



A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Fig.4-11. The activity of genetic element C is determined by the activity of A and B: only if both A AND B are active C will become active, too.

As a simple example of this mapping, consider the three genetic elements A, B and C of Fig.4-11. C will only be active if and only if both A *and* B are active, thus realizing the Boolean AND function. Other examples of Boolean functions are OR, XOR, etc.... Note that the Boolean function is specified by a lookup table with 2^K entries, i.e. in the example this is 4, resulting in 16 possible Boolean functions for K=2.

P	R	S
0	0	0
0	1	0
1	0	1
1	1	0

Fig.4-12. The structural genes in the lactose operon are only expressed if the promoter site is active AND the repressor site is inactive, yielding the Boolean rule shown.

In (Kauffman 1993) the use of random Boolean networks is illustrated with a biological example, i.e. the lactose operon. The structural genes that make up the enzyme are only transcribed if the promoter site P next to the coding region is active (forms an active transcriptional complex) and if the repressor site R is inactive (a more detailed explanation is given in (Kauffman 1993, page 444-446), where it is shown that the activity of the promoter is in turn a function of the presence of four other elements). This relation can be expressed using a Boolean lookup table as shown in Fig.4-12, one of the sixteen possible Boolean updating rules for two inputs.

Implications of the Random Boolean Network Model

Modeling the genetic regulatory networks in this way has certain implications:

- As all nodes are alike, differences between regulatory elements are ignored: there is only one type of genetic element
- The genetic elements influence each other all in a similar way; thus the differences between different regulation strategies in the biological cell are ignored as well
- The genetic elements are modeled as either active or inactive, i.e. all notion of concentrations and graded response in the biological regulatory networks is lost

However, these are justifiable simplifications for the purpose we have in mind. In the following we discuss these implications in more detail and argue why they are adequate for our purposes.

This model ignores different regulation strategies

In the biological cell there are a number of strategies for the regulation of gene expression: essentially each step of the pathway between the coding sequence on the DNA and its final gene product presents an opportunity to regulate the expression of that particular gene (Alberts et al. 1991, page 551-556). Thus, a regulatory gene could code for a transcription factor that binds to the DNA in the neighborhood of another gene, and in that way influence its expression. It could also code for enzymes that break down the messenger RNA molecules of another gene, thus negatively influencing the expression of that other gene (RNA degradation control). It could be that a certain gene product is necessary to correctly process RNA of another gene (e.g. splice out an intron, a piece of non-coding RNA), so that the expression of that gene is directly dependent on the first one.

An important decision had to be made concerning the level of detail at which we wish to simulate these regulatory elements and their interactions. Do we make a distinction between transcriptional control and RNA degradation control and incorporate them as different building blocks in our model ? Do we really need to model every player in this complex process: the enzymes, RNA molecules, regulatory proteins and DNA sequences ?

I decided to ignore all differences between the players in the genetic regulatory network and model one type of genetic element and one type of interaction between elements, because of the following reasons: (1) Because of computational limitations, it is unrealistic to try to model all these different mechanisms of gene regulation; (2) It made it possible to accommodate the random Boolean Network model, which is available in the literature. In addition, it is fast and there are powerful tools to analyze it; (3) Even with access to unlimited computing power it is unlikely that we could attempt a model at that level of detail. The biology of these processes is not fully elucidated, and we would have to make too many assumptions about the - unknown - details of the molecular mechanisms. Thus, our final model would certainly be too far removed from the actual biological facts to be useful; (4) Finally, such detail is simply not needed for our current purposes. We are trying to capture the phenomenon of cell differentiation into an abstract cell model for a developmental process in artificial organisms. I see no immediate advantage in modeling all these different regulation strategies that are, after all, a consequence of the implementation level details of the biological cells: DNA, RNA, proteins and so on.

Since there is only one type of genetic element, this almost automatically implies that we will model only one kind of interaction between these elements, because most of the time the differences between alternative regulation strategies in the cell come down to differences in the type of players involved. In Fig.4-14 a random Boolean network is shown schematically, to be compared with the genetic regulatory network of Fig.4-9: as you can see, the differences between the elements have disappeared and all elements influence each other the same way.

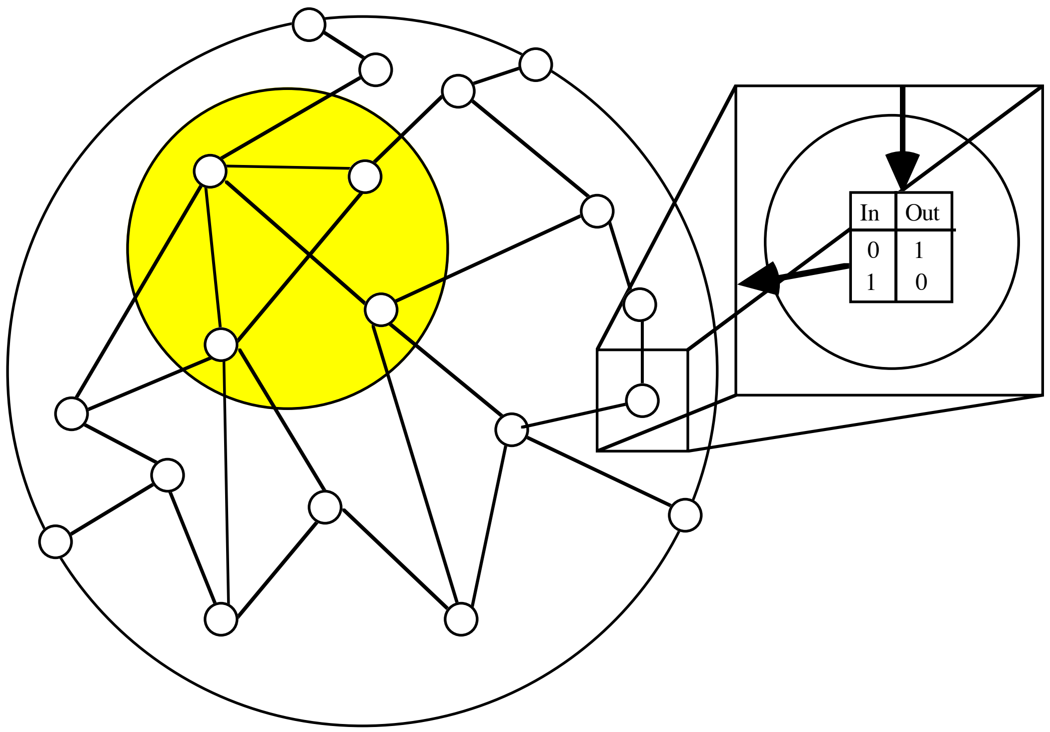


Fig.4-14. The random Boolean network corresponding to the regulatory network in Fig.4-9.

The genetic elements are modeled as on/off elements

The genetic elements themselves can be modeled by anything ranging from simple binary to complex quantitative models. A genetic element must include the concept of activity, as the regulation of these activities is what the model is about. This could be implemented in various ways. For example, we could use binary elements, in which one of the states is the active state and the other the inactive state. In previous attempts at modeling the gene expression in cells, this Boolean idealization approach has been taken by (Kauffman 1969; Jackson et al. 1986). An extension of this might be to use multi-level logic, as was done for example in (Thieffry and Thomas 1993). Yet another strategy is not to use discrete states at all, but assume that the activity of a gene is a continuous variable; this approach has been taken by (Mjolsness et al. 1991), who implemented the genetic regulatory networks as dynamical neural networks (see Fig.7-2). The latter work is more focused on parameter identification of actual biological processes, however, in which case this more detailed approach makes sense.

I have chosen to model the genetic elements as binary elements, for the following reasons: (1) A binary model is especially attractive from a computational viewpoint: their simplicity will allow us to simulate a large number of them in a reasonable time. As explained before, this is a necessity when we will use this model in a multicellular context in conjunction with genetic algorithms; (2)

Many phenomena that occur during embryology have an on/off quality. To a first approximation, one can view differentiation pathways as the turning on or off of certain genes, leading to the presence or absence of certain gene products. Certainly in eukaryotic gene regulation there are whole operons that are tightly packed in heterochromatin, i.e. these genes are really OFF. And others are really ON. So there is, at least at that level, a Boolean aspect; (3) Simplifying genetic elements to binary variables can be defended on a deeper ground: a lot of 'continuous' regulation events have some mechanism of self-amplification. For example, biochemical pathways linking cell-surface receptors to the DNA have lots of amplification steps built in (Walbot et al. 1987, page 321), ensuring that their response is on/off like. In (Kauffman 1993) the author presents additional arguments why the major features of many continuous dynamical systems can be captured by a Boolean idealization.

Some Additional Issues

Random Boolean Networks update synchronously, i.e. every time step the whole state vector is computed using the values of the state vector at the previous time step. Discrete time, synchronously updating networks are certainly not biologically defensible: in development the interactions between regulatory elements do not occur in a lock-step fashion. The alternative is to update all nodes asynchronously, each node having a given probability at any time to recompute its value from its inputs at that time. This introduces an element of non-determinism, however, that might render any genetic search in a space of such networks very difficult. In addition, they are less readily analyzed than their synchronous counterparts, for which there are excellent analysis tools available (Wuensche 1994). On the other hand, asynchronous random Boolean networks be useful to examine phenomena like spontaneous symmetry-breaking interactions between cells, that can not occur with lock-step updating. Also, traveling wave patterns seem to benefit from stochastic updating techniques (Maarten Boerlijst, personal communication).

I ended up using a comparatively small number of elements in the Boolean networks. When one looks at the function of genes in eukaryotic genomes, one finds that the vast majority of gene products will be responsible for housekeeping functions that are common between cell types, and most of the others are cell-type specific genes (Alberts et al. 1983, page 553; Walbot et al. 1987, page 174-175). In addition, genes have been found that switch on whole gene-batteries at a time (McGinnis and Kuziora 1994), thus acting as a representative for a whole class of genes. This could suggest that in actuality a small number of genes might be responsible for the regulatory mechanisms within the cell.

A Dynamical Systems Perspective

The state of the random Boolean Network, i.e. the activities of all the genetic elements at a given time, will tell us something about what the cell is capable of at a given moment. Up to this point we have not yet discussed communication with other cells, nor discussed the functions that a cell might execute within an artificial organism. In the next chapters the activity of the genetic elements will be very relevant, however: they will determine whether a cell will respond to a signal from other cells, the cell type and, for example, whether a 'neuron' cell will sprout an axon or not.

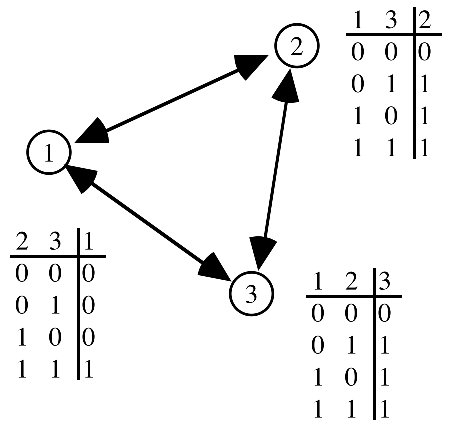
We will be very interested in how the state of the random Boolean Network within the cell evolves over time during the simulated developmental process. In fact, in the adult artificial organism the state of the network will correspond to the cell type of the cell, as we have deduced before: this is inspired by the biological equivalent, where the set of structural genes expressed determines the cell type. During development, the cell state (we will use the terms *cell state* and *random Boolean Network state* interchangeably from now on; also, the term *cell* will be used instead of always referring to *model cells*, if no confusion is possible) corresponds to the intermediate stages between egg and final cell type. Thus, the way the cell state evolves over time can be seen as roughly analogous to the developmental pathways that biological cells go through.

The specific properties of the random Boolean Networks, i.e. a limited number of states and the synchronous updating, make it very easy to analyze their dynamical properties. In what follows, we will introduce some concepts from dynamical systems theory in the context of random Boolean Networks, and after that we will proceed by discussing their connection with the developmental pathway our model cells will traverse.

Concepts

State space

To know what happens to the state of the cell's regulatory network as it changes over time, we must consider the space of all possible state vectors of a given network, the *state space*. Because random Boolean Networks are updated synchronously, the next state of the network is always completely determined for any given state that the network finds itself in. This enables us to list the successor of each state in a state transition table. For example, in Fig.4-16 the state transition table is given for a small network with N=3 and K=2, first in binary format and then in decimal format. The table must be read as follows: if the network at time t is 000, the state at time t+1 will also be 000. For each of the eight possible states a successor state is listed.



1	3	2
0	0	0
0	1	1
1	0	1
1	1	1

2	3	1
0	0	0
0	1	0
1	0	0
1	1	1

1	2	3
0	0	0
0	1	1
1	0	1
1	1	1

t	t+1
123	123
000	000
001	010
010	001
011	111
100	011
101	011
110	011
111	111

t	t+1
dec	dec
0	0
1	2
2	1
3	7
4	3
5	3
6	3
7	7

Fig.4-16. The simple random Boolean Network example from before and its state transition table. The state transitions are also shown using the decimal values of the state vectors.

State transition diagram

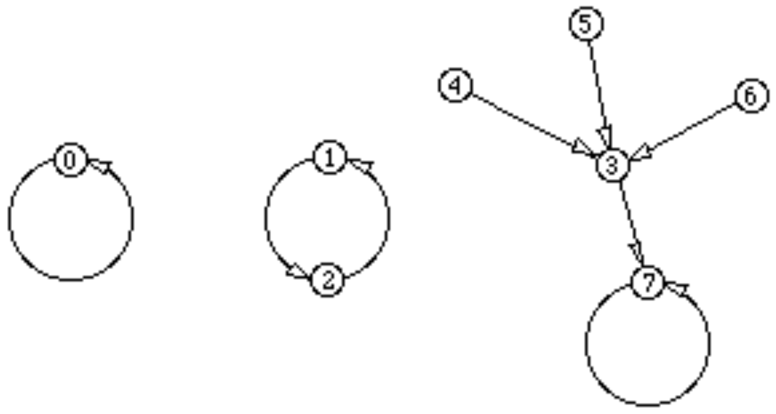


Fig.4-17. The state transition diagram of the example network. The small circles represent states (displaying the decimal values of the binary state vectors), the big circles attractors. The arrows on the edges could be dropped in the transient tree as the flow is always towards the attractor.

Moreover, if we represent the sates in some graphical way, in effect creating a graphical representation of state space, we can construct a *state transition diagram* for any particular network: in such a graph, we represent each state by a node, and connect it to its successor state by means of a directed arc. One state can have only one successor, but can have many predecessor states. In this manner, we get a directed graph representing the changes of the network's state vector over time. In Fig.4-17 the state transition diagram for the example of Fig.4-16 is depicted.

Trajectories in state space

Now we view the changing state of the system over time as the traversing of a trajectory through state space: when a cell is in a certain initial state, it will always follow the same trajectory. Thus, when given the state transition diagram and this initial state, we can easily follow and predict what states the cell will traverse. This makes state transition diagrams an excellent tool to analyze how the expression of genes in the cells changes over time, as the graphical representation allows us to see large scale structure in the dynamics of these networks much more easily (see below).

Attractors and basins of attraction

One of the most important properties of the dynamics of random Boolean Networks is the existence of *attractors* in state space. Since the number of states is finite (it is simply 2^N), it follows that at a certain moment the network will reach a state it has already traversed, after which it will simply repeat that same trajectory over and over again: the system has settled into a *state cycle*. The number of states it traverses on the cycle before reaching a previously visited state is called the *period* of the state cycle. Since the network will always fall into some state cycle, eventually, these state cycles are attractors in state space. The dynamics of random Boolean Networks allow for only two types of attractors: state cycles and *fixed point attractors*. The latter category is in fact only the special case of a state cycle with period 1. In other dynamical systems, other types of attractors may exist, for example chaotic ones.

Each attractor has an accompanying *basin of attraction*. Since there may be more than one attractor present in state space, it follows that for any given initial state of the network, one particular attractor is reached, and always the same one (since random Boolean Networks behave in a deterministic way). The collection of all these initial states (including the states on the attractor itself) that will lead to the network reaching a certain attractor, is termed the basin of attraction for that attractor. Once the state of the system has landed on an attractor, it cannot leave that attractor anymore. It follows that these basins are non-overlapping and thus define a partition on the state space.

When looking at Fig.4-17, we can see that the simple network has three attractors: two fixed point attractors (0 and 7), and one state cycle with period 2 (cycling between states 1 and 2). Moreover, the attractor 7 has a basin of attraction consisting of states 3, 4, 5 and 6. The longest transient trajectory the state of the network can traverse is when the initial state is 4, 5 or 6: then 2 transitions will take place before the fixed point attractor is reached.

Phase portraits

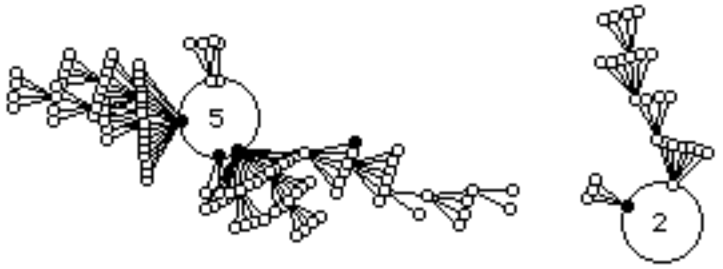


Fig.4-18. The phase portrait of a larger network with N=7 and K=3.

The collection of attractors, their basins of attraction and the particular trajectories that lead to these attractors are collectively called the *phase portrait* of the system, and it is this phase portrait that is immediately apparent from the state transition diagram. By one look at the graph one can assess the number of attractors, the relative size of their basins of attraction, and the length of the transient trajectories that may have to be traversed before a state cycle or a fixed state attractor is reached. All this can be done without having to know the underlying updating rules or wiring configuration of the network considered, which is one of the great advantages of working with phase portraits.

Consider for example the phase portrait of a larger network (N=7 and K=3) in Fig.4-18: the small circles represent the 128 different states the network can be in, whereas the two larger circles represent the attractors, in this case both state cycles. Inside the attractor circles the period of the attractor is shown, i.e. resp 5 and 2. The attractor states, i.e. the states that form the state cycle, are the ones that lie on the attractor circle, and their angular position is proportional to their decimal value. In this way, the angular position represents a way of identifying a state, albeit non-unique since for larger values of the state the angle will wrap around and overlap with positions that can be taken by lower-value states. Also, one can not deduce from these diagrams in what order the state cycle is traversed. I chose to relate angular position to state vector to be able to view mutations more easily (see below), as opposed to knowing how the state cycle is traversed.

Relating these Dynamical Systems Concepts to the Model

In what follows we will associate fixed point attractors with the intermediate (and final) stages that a cell traverses while it goes through its 'developmental pathway'. At each step of the developmental process, as will be explained in detail in the next chapter, we will let the state evolve towards a fixed point attractor. Say that a cell is in a fixed point attractor, i.e. a state that is stable and will not change anymore over time until something happens. During development, this is like a 'precursor' stage, in the adult organism it is the 'cell type'. Now, assume that for some reason, one of the elements changes from inactive to active. For example, the activity of that element was also dependent on some variable outside the cell (we will model intercellular communication in this manner, see below). If this genetic element happens to regulate other elements, a cascade of changes will propagate through the network. This will happen until a new stable state is reached, i.e. a transient state space trajectory is traversed until a new fixed point attractor is reached. Thus, the simulated developmental pathway that will be traversed by the cell is the succession of fixed state attractors that the cell will attain, separated by transient trajectories in state space.

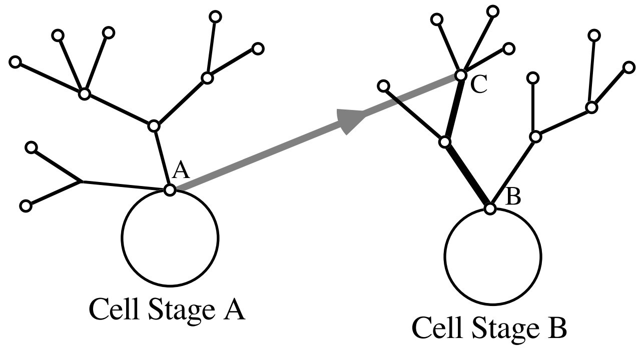


Fig.4-19. A cell is thrown from a stable state A into an unstable state C, after which it traverses a transient trajectory (indicated in bold) and settles down in the stable state B.

We will not be interested in random Boolean Networks where the state reaches a cyclic attractor: instead of flowing into a fixed state attractor, the state of the network could reach a state cycle, i.e. the network oscillating between a number of states. Already in the simple example we discussed above, there was a state cycle with period 2. We are only interested in the fixed point attractors, because these correspond to stable expression of a combination of regulatory genes/gene products, determining the cell type (or intermediate stage in development). Although it is certainly not the case that biological cells are static entities without time varying quantities in them, the state cycles in our networks would correspond to cycling on and off of regulatory elements, something that does not seem to be very useful. Moreover, the state of the system would traverse these cycles with a frequency that is totally arbitrary, as it depends on our - arbitrary - choice of the time delay we assume between successive updates of the network. If we assume, as we did, that the regulatory network serves to activate structural gene sets, it makes no sense that they turn on and off all the time, but they should, on the contrary, be activated in a reliable and constant fashion when the right preconditions are present[4].

Quantitative Analysis

It is of interest to examine (1) how many different possible networks there are for a given N nodes and K inputs; (2) how many different phase portraits there are for the same N and K; (3) how these numbers are related. Knowledge about these quantitative properties of random Boolean Networks and their phase portraits will help to gain insight in the size and degeneracy of the search spaces we will encounter later when working with genetic algorithms to evolve regulatory circuits.

The number of different random Boolean networks

A random Boolean network is completely specified by its topology and the updating rules of all nodes:

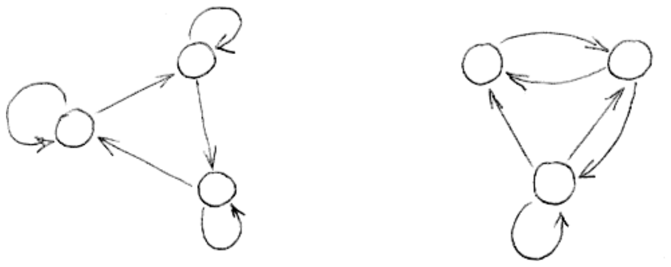


Fig.4-20. Examples of wiring configurations in a Boolean network with N=3 and K=2.

The topology information specifies what nodes serve as input for which nodes, i.e. defines the pseudo neighborhood for each node. As a particular node has K inputs, and connections can be recurrent, there are N^K different pseudo neighborhoods possible for each node. As there are N nodes in total, the total number of possible wiring configurations is thus $(N^K)^N$. This number can become very large even for small N and K! For example, Fig.4-20 shows some of the $(3^2)^3 = 729$ possible random Boolean Network configurations with parameters N=3 and K=2.

[4] An alternative view is proposed by Kauffman, who relates cell type to large cyclic attractors in state space. He also considers networks with many magnitudes more participants than the ones are normally considered here (order of 100000 as opposed to 10-100 regulatory genes).

The updating rule of a node can be any Boolean function of its K inputs and can be specified by a lookup table with 2^K entries, one for each of every possible combination of input values. As each entry can contain a value of 1 or 0, there are 2^{2^K} possible such lookup tables.

So, all the possible random Boolean Networks for a given *N* and *K* can thus be calculated by multiplying the number of different wiring configurations for each node, N^K, with the number of possible updating rules, 2^{2^K}, and raising the result to the power of N (as each node may have a different wiring scheme and updating rule). This yields a total of $\left[(N^K)(2^{2^K})\right]^N$ different random Boolean Networks for a given *N* and *K*, a number that can become very large indeed. For example, for *N*=3 and *K*=2, there are $\left[(3^2)(2^{2^2})\right]^3 \cong 3\cdot 10^6$ possibilities.

Another but less instructive way to arrive at this formula is to calculate the number of bits needed to describe a random Boolean network: for each of the N nodes, K inputs need to be specified (each ln N bits) and one lookup table (2^K entries), yielding a total of $b = N(K\cdot \ln N\cdot 2^K)$ bits. The number of possible networks is then 2^b, yielding the same formula as before.

Although we will always have an astronomical number of possible networks even for small N and K, there are two ways in which this number is reduced. First of all, there is degeneracy in the space of possible networks, as the nodes in the random Boolean Networks are *unordered*. Because of this, all permutations of the nodes for a given network actually yield equivalent networks. Of course this changes once a meaning is attached to certain nodes, as then a node becomes *labeled*. For each labeled node, the degeneracy of the space decreases accordingly. The second consideration is that only a certain class of network interests us, i.e. those with (at least some) fixed point attractors, so this will reduce the number of networks that we will have to work with. However, note that even when only this class is useful in a developmental context, the networks that are not in this class will still have to be reckoned with by the genetic algorithm.

The number of different phase portraits

To calculate how many different phase portraits can possibly exist for a given N is much simpler. This number will be independent from K, as will be seen shortly. Indeed, each phase portrait is defined by a state transition table in which there are 2^N entries, i.e. for each of the possible network states the next state is listed. As this next state can be any of the other states (including the state itself), there are 2^N choices for each entry, yielding a total of $(2^N)^{(2^N)}$ different phase portraits. The number of bits needed to describe a phase portrait is $N2^N$.

Again the same argument can be made about the degeneracy of the space, as the states are also unordered, and if only phase portraits with fixed point attractors are considered, the number is reduced even more.

Relating phase portraits and random Boolean networks

The number of different phase portraits rises much more rapidly with increasing N than the number of different networks, as can be readily deduced from the formulas calculated above (since the number of phase portraits is doubly exponential in N). Related to this, the number of bits needed to describe a phase portraits is much larger than the number of bits needed to describe a network, as already calculated. This is one reason why we do not evolve phase portraits directly.

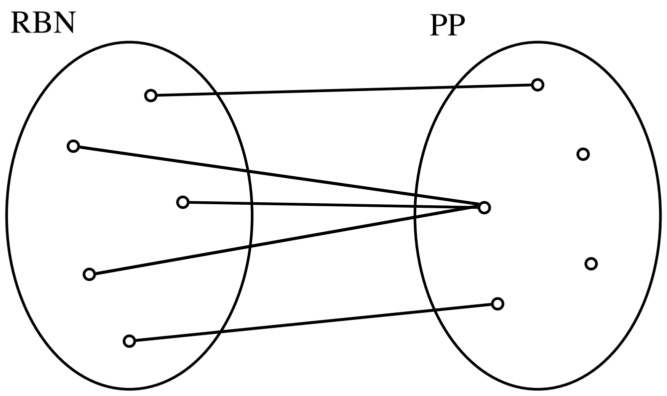


Fig.4-22. The relation between phase portraits and random Boolean networks

The relation between phase portraits and random Boolean Networks is illustrated in Fig.4-22. The essential characteristics of this relation are the following:

- Every random Boolean network has a phase portrait.
- Two different random Boolean Networks might have exactly the same phase portrait. As a simple example, the Boolean function 1 (the lookup table filled with 1) will always yield 1, irrespective of the inputs to the node. Hence, all the networks that differ only in the wiring to the node that realizes this function will have the same phase portrait.
- For a given N and K, it is entirely possible that some phase portraits cannot be realized by any random Boolean Network. A trivial example is when K=0. In this case, there are only 2^N different random Boolean Networks, and they all have a trivial phase portrait with one fixed point attractor. All other phase portraits (for example containing a state cycle) are impossible for K=0.

From these considerations alone, a number of conclusions can be drawn about searching these spaces by means of the genetic algorithm (see below). It is evident that, when searching the parameter space of random Boolean Networks of given N and K, we are at the same time exploring a *subspace* of all possible phase portraits for that N. However, mutation and crossover operators (see later) will have a totally different effect than when mutating phase portraits directly. In addition, mutation of a network might be neutral with respect to its phase portrait.

Evolving Regulatory Circuits

Now that we have proposed a model for genetic regulatory networks, it is useful to examine how this model behaves under evolution.

Evolving Regulatory Circuits in a 'Unicellular' Context

Development is a process characteristic of multicellular organisms, but genetic regulatory networks existed before that and were equally subjected to evolutionary pressures, namely in the unicellular organisms like bacteria and protists. One of the classic examples in bacteria is that of the lactose operon: in reaction to a change in the medium in which it resides, it starts expressing the genes that code for an enzyme that is more suitable to process the 'food' now present in the environment. In fact, since these animal kingdoms, protista and monera, still make up the bulk of all life, even at this very moment most evolutionary processes take place at the unicellular level.

It is convenient at this point to define the notion of a *regulatory circuit*, i.e. a small subset of interconnected elements in a genetic regulatory network, together with the edges that connect them. The lactose operon is an excellent example of such a circuit. Although it might influence other events inside the cell, when we talk about the lactose operon we mean only the *circuitry* that is directly relevant to the production of the lactose enzyme. Said in another way, a regulatory circuit is a sub-network of the genetic regulatory network, independent with respect to a particular function.

One can speculate that multicellular development is in fact a novel way of using the mechanisms of long term behavioral selection (by means of gene regulation) that were already present in unicellular organisms. Evolution proceeds this way in many cases. Indeed, the choices that must be made during development are the result of the appropriate expression of regulatory genes in response to signals from the cell's environment (or internal signals), reminiscent of how for example the lac operon works. Regulatory circuits in a unicellular context can thus be seen as the evolutionary precursors for the regulatory circuits in development.

We may not know whether this is what actually happened during evolution, but we know that in our model (since we make it that way) the random Boolean Networks will be used in the model of multicellular development. Because of this, it is useful to examine how they perform at the unicellular level as a regulator of long term behavior.

Not only does the random Boolean network underlie the developmental pathways we will see in the multicellular model, but at the same time it provides the vehicle of information that is responsible for the heritable character of our simulated developmental process. Indeed, a change in the developmental sequence can only be brought about if these networks change. Thus, it is of importance to examine whether our model behaves gracefully under evolutionary pressure: do mutations in our model of the regulatory mechanisms result in behavior totally uncorrelated with what was there before, or is there some structure in the way the behavior is changed ? Is it easy to evolve crucial regulatory circuits that make sense in a unicellular context, but also in the context of development ?

If we can not use the random Boolean network model to evolve simple cellular behavior, it is unlikely that we will obtain the coordinated cell behavior that is the hallmark of development. In addition, it is beneficial to be able to study the evolution of simple regulatory circuits in isolation,. Once the random Boolean network model is embedded in a model of multicellular development, one has to infer much more what is going on. In contrast, in a unicellular context where we have no other phenomena to detract our attention.

Artificial Evolution and Phase Portraits

Seen from a different angle, evolving a regulatory circuit can be viewed as modifying the phase portrait of the genetic regulatory network such that a *performance function* is optimized. Indeed, the phase portrait describes what happens to the state of the cell over time, i.e. how the regulatory circuits of the model genetic regulatory network influence the differentiation of the cell. However, note that we will not try to modify the phase portrait directly, but instead we will modify the underlying random Boolean network.

Wuensche's learning method

One way of shaping phase portraits by modifying random Boolean Networks is discussed in (Wuensche 1994): here Wuensche hypothesizes that transient trajectories can serve as content-addressable memories, where the retrieved memory is represented by the attractor the system eventually settles into[5]. In the paper he explores ways of ‘learning’ new basins of attractions in the state space of these networks, basically by means of two mechanisms: either by mutating the wiring scheme of the network (the edges between the nodes) or by mutating the rule scheme (the lookup tables). In both of these methods, the learning algorithm serves to learn a new pre-image for a given state. The two states, aspired pre-image and target state, must be specified manually.

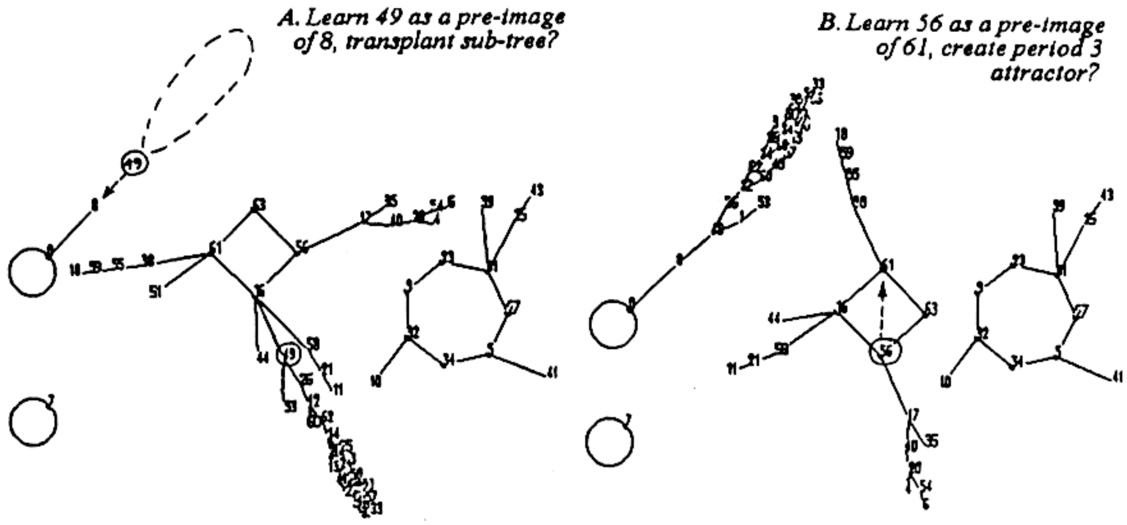

A. Learn 49 as a pre-image of 8, transplant sub-tree?
B. Learn 56 as a pre-image of 61, create period 3 attractor?

Fig.4-23. Left: phase portrait where we want to make state 49 a pre-image of state 8. Right: the result after a rule has been changed to achieve that effect. From (Wuensche 1994).

As an example, consider Fig.4-23 the aim is to make state 49 a pre-image of state 8, so that the entire transient tree culminating in state 49 is transplanted to another part of the phase portrait. This can be done by examining the rules and connections of the network and looking which one can be mutated to achieve the desired result. The right panel of Fig.4-23 shows the result when such a change is effected: as you can see, the transient tree is indeed moved to a different attractor, but it is also apparent that side-effects are created because of this. As can be intuitively expected, Wuensche has established that mutating connections has much more side effects than mutating rules only.

Using genetic algorithms

In contrast with this manual approach, we will make use of evolutionary processes to shape the phase portraits of the regulatory networks, using a genetic algorithm to find the networks with the

[5] Although he also considers what he calls ‘memory, far from equilibrium’, i.e. the transient trajectories themselves being a repository for memories.

right phase portrait. As in Wuensche's method, by mutating the wiring configuration and the update rules that specify the random Boolean Networks, changes will occur in their phase portraits. However, in our case there will be no manually specified pre-images or target states: instead the aim is to optimize a given performance function. Some of the changes in the phase portraits might be drastic, like the appearance of new attractors or disappearance of old attractors, or a fixed state attractor that suddenly becomes cyclic. These sudden changes are analogous to bifurcations in dynamical systems theory. Other mutations might be less dramatic: some states might move from one basin of attraction to another one, for instance. By evaluating how these changes affect performance, we will be able to select the better networks and use them as the parent population in the next GA generation. This process will be repeated over and over again until, in this case, an optimal state cycle is found.

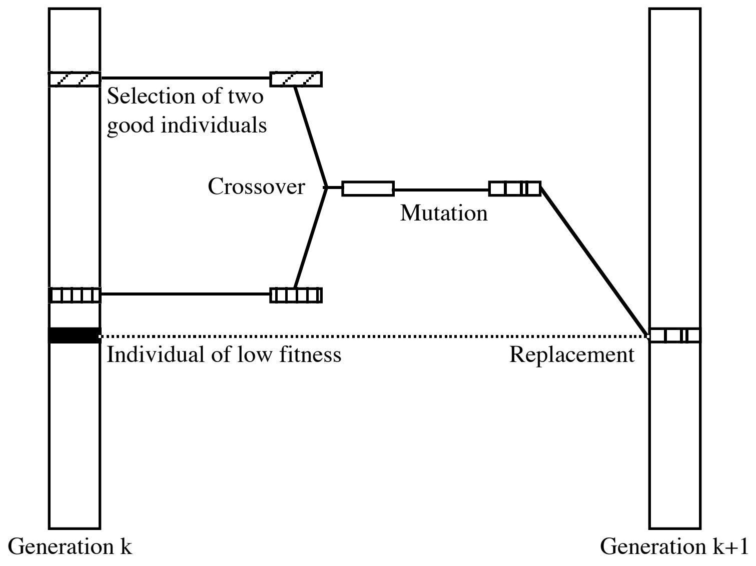
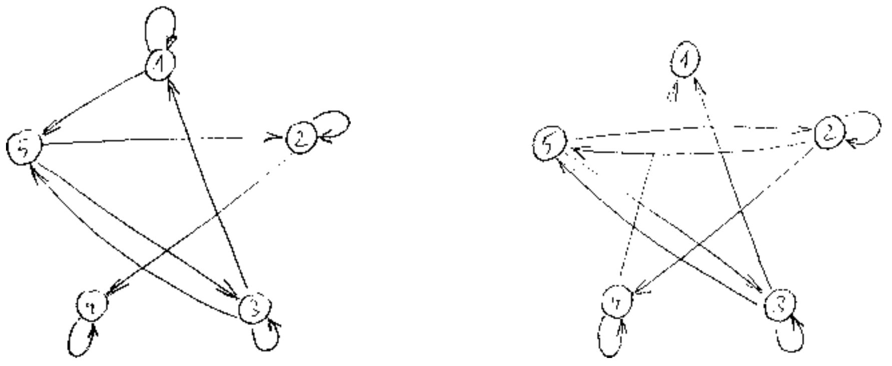


Fig.4-24. The steady state GA (mutation and crossover are schematically indicated).

The genetic algorithm used is a steady state genetic algorithm (see Chapter 2) and is illustrated in Fig.4-24: at each generation two of the fittest individuals in the population are taken to generate one offspring, using a tournament selection scheme. The single offspring organism then replaces one of the individuals in the population, with a probability dependent on the fitness of the individual considered for replacement (see also Chapter 2). In the following experiments, the population size was 100 and the tournament size was 7 (the number of individuals pitted against each other in the tournament selection algorithm). As explained before, steady state genetic algorithms are very good for maintaining adequate variability in small populations and thus avoid premature convergence.

Encoding and mutation of random Boolean networks

The genetic encoding used in the genetic algorithm is a straightforward description of the random Boolean network: for each node, labeled from 1 to N, the Boolean function is specified as an array of bits, representing the lookup table of the function. To specify the connectivity, K labels indicate for each node from what other nodes it receives input. The lookup table is specified in ascending order, the first bit in the array denoting the lookup entry for (decimal) input value 0, the last for input value 2^K. The input labels are given in low-to-high bit order, i.e. the first input will be multiplied by 1 to arrive at the decimal input value, the Kth input by $2^{(K-1)}$.



Node	Inputs	Lookup Table
1	1 3	1011
2	2 5	0000
3	5 3	0001
4	4 2	0110
5	3 1	1100

$\xrightarrow{\text{3 Mutations}}$

Node	Inputs	Lookup Table
1	**4 3**	1011
2	2 5	0000
3	5 3	**0101**
4	4 2	0110
5	**3 2**	1100

Fig.4-25. Left: The genetic encoding of a random Boolean network with N=5 and K=2. Right: the way it is modified using a mutation rate of 3 mutations per genome.

An example for a simple network with N=5 and K=2 is given on the left side of Fig.4-25, where a wiring configuration is shown along with a genetic description of the network. Node 1 for example, takes input from nodes 1 and 3, as indicated by the arrows in the figure. The Boolean functions can be inferred from the lookup tables in the genetic description: for example, node 3 will be active only if 3 *and* 5 were active in the previous time step. Node 2 will never be active, and node 5 will only be active if 3 is active. In a similar way, you can deduce Boolean functions for all of the nodes.

Both the connection parameters and the Boolean function are subject to mutation: in the case of the input labels, a new random value between 1 and N is chosen. In the case of the lookup table entries, one of the bits is flipped. The *mutation rate* is specified in mutations per genome: in most experiments, a mutation rate of 5 is used, unless indicated otherwise. An example for a mutation rate of 3 is shown in Fig.4-25, with 1 lookup entry bit flip and 2 input labels randomized, but any other combination yielding a total of 3 mutations is also valid. *Crossover* is implemented as simple two point crossover and is done with respect to the boundaries of the nodes: for example, node 5 to 8 of one genome might be exchanged with those of another genome. The *crossover rate* is specified as the probability that two chosen progenitors will exchange genetic material. I have found crossover to be very useful for this particular genetic encoding and so most of the time it is set to 100%.

Effect of mutation on the phase portrait

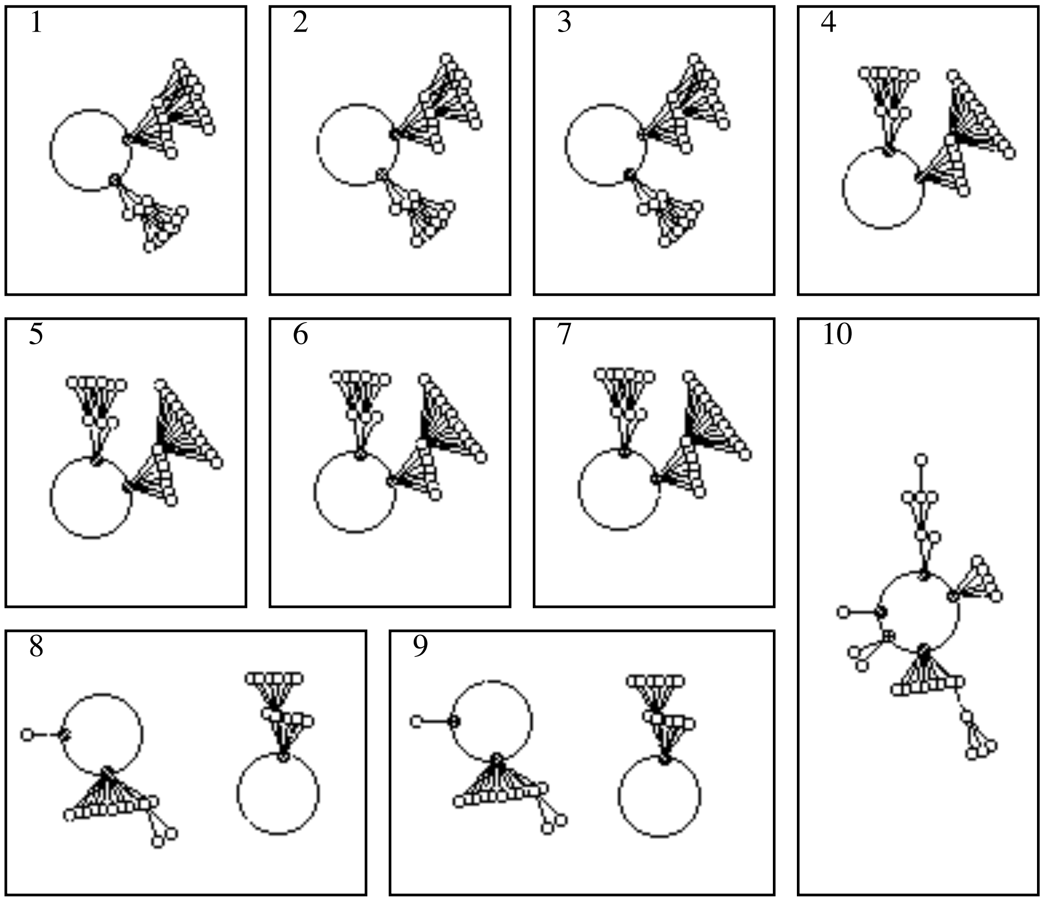


Fig.4-26. The changing phase portrait of a random Boolean network as it is mutated. Mutation rate is 1 mutation per genome.

To illustrate the way the phase portraits are changed by mutation, consider Fig.4-26. Here a random Boolean network was randomized (shown in panel 1) and then subsequently mutated 9 times, each time with a mutation rate of 1 mutation per genome, i.e. either one bit of a Boolean function was flipped or one of the wires was randomly reconnected (see below for detailed explanation). In the figure, the small circles represent states, and the larger circles attractors. As explained before, the angular position of the states on the attractor circles is proportional to the decimal value of their state vector, so if they stay at the same place after mutation one can assume (but not know for sure, since there is some redundancy in this representation) that these states have not been affected by the mutation.

As time progresses and mutations accumulate, the phase portrait changes in several ways. At first, there is only one attractor, a state cycle with period 2. Mutations do not seem to affect much until at panel 4, one of the attractor states is replaced by another one, at a different angular position. In panel 8, suddenly a second attractor appears: it can be readily seen that it is not unrelated to what was there before. In panel 10, a drastic rearrangement leaves us with one state cycle of period 5, but again a relation with the previous phase portrait can be readily detected. An interesting point to make here is that a mutation does not necessarily change the phase portrait: neutral mutations exist, although they can of course synergize with mutations that happen down the line and produce a larger than expected result later.

Evolution of a State Cycle

Mimicking a Cell Cycle

To test the evolvability of random Boolean networks, I tried to evolve a network that had a state cycle with some interesting characteristics. Although we will not be interested in state cycles per se in the context of the multicellular model, I wanted a test problem that was as simple as possible, preferably without having to supply any external input to the network. However, a random Boolean network without any inputs is not very interesting in itself: it will always settle down either in a fixed point attractor or in a state cycle. Since I could not see much challenge in evolving a particular fixed state attractor in itself, I decided on a state cycle as the next more interesting problem.

In addition, a state cycle is interesting on another plane as it is reminiscent of a biological cell cycle. A state cycle is a succession of different states of the random Boolean network, just as a biological cell cycle can be seen as a succession of genetic regulatory network configurations. However, this is where most comparison ends: unlike in the biological example, there are no complex non-linear dynamics here, nor is there a concept of timing the duration of the different cycle states: the different phases of a real cell cycle may be very different in length, and the transitions between them may be swift or rather slow, depending on the kinematics of the underlying reactions. In the state cycles we will evolve here, each transition will take the same amount of time, as determined by the time constant we choose to update the networks.

The biological cell cycle is an important phenomenon, both from a unicellular as well as from a developmental viewpoint. Indeed, in development, cell division is obviously an important mechanism of growth, but it is also an important way to accomplish certain morphological properties: by dividing unequally in different parts of the embryo, changes in shape can be accomplished. In unicellular organisms, evolving an optimal cell cycle is a matter of life and death (for the species, not the individual): if you do not have a successful strategy to propagate your genes into the population (and this while making optimal use of the resources at hand) somebody else that is better at producing progeny will take over your habitat.

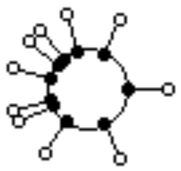


Fig.4-27. A cell cycle can be implemented in random Boolean networks by means of a cyclic state attractor in state space.

Thus, we can make the problem more interesting by devising a simple cell model, and letting a state cycle play the role of cell cycle. To implement cell division, we can simply assume that one of the genetic elements is a regulatory gene that triggers the process of cell division. Thus, the cell will divide when that genetic element becomes active. If a state cycle is now evolved in such a way that the ‘dividing’ element is turned on in a regular fashion, we have mimicked a simple cell cycle.

To make the problem even more interesting, we will assume that a cell needs energy to divide, and that cells dividing more than others (in a given time span) have a better performance (they would produce more progeny, as it were). In this way, performance will increase if division takes place at a rapid pace. However, if a cell divides too quickly, it will die for lack of energy. Thus, there is an optimal value for the period of the state cycle, and we will try to find an optimal state cycle using the genetic algorithm.

At this point some clarification may be necessary: although we have argued before that cyclic attractors are not really of interest to us, we are now trying to evolve them. Why ? Indeed, in the model of multicellular development we will actually incorporate an explicit ‘cell cycle’ in our simulation algorithm, so that we have no need for these cyclic attractors. The point that we are exploring here, however, is evolving phase portraits to optimize a given performance function: the cell cycle problem is a simple and informative example of how state cycles of different periods and with different properties can be arrived by an evolutionary process. It is ideal to illustrate the shaping of the phase portrait, as state cycles are readily detected in the state transition diagrams and it is very clear how they change in character as evolution proceeds.

The Cell Model

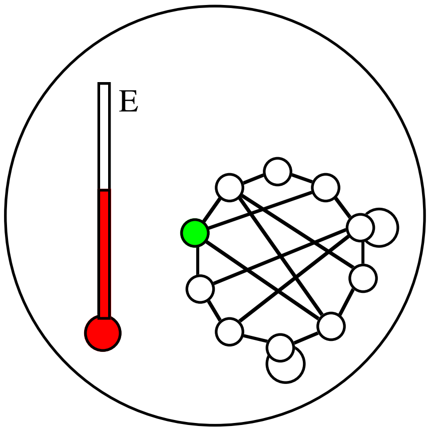


Fig.4-28. The simple cellular model used to try and evolve a ‘cell cycle’. It has a regulatory network and an energy level. The cell is assumed to be in a rich medium so that at each time step, its energy level is replenished by a constant amount (see text).

Here we will explain the simple cell model mentioned above. Each cell has a random Boolean network and an energy level (see Fig.4-28). Only two possible actions are possible: to calculate the next state of the regulatory network and/or go through cell division. Both of these actions take up a certain amount of energy: the next state calculation reduces the energy level by an amount of energy proportional (by a constant EC) to the number of nodes in the regulatory network, and cell division always takes up a constant amount of energy, hereafter referred to as ED. If a cell signals it wants to divide (by means of the ‘dividing’ element) it first checks whether enough energy remains: if not, the cell dies. At each time step the energy level of the cell is replenished by a constant amount ER. The cell dies after 50 time steps or when its energy runs out.

```
step (k+1) :  0  1  2  3  4  5  6 7
Ek+ER      : 17 27 37 47 57 67 77 87
Ek+ER-EC*N : 10 20 30 40 50 60 70 80
```

Fig.4-29. At each time step, E is increased by ER, then the state of the system is calculated, reducing E by EC*N. This is the energy the cell could use to divide. In the example ER=17, EC=1, N=7.

The optimal cell cycle is the one that yields the most offspring for the cell, and this will depend on the parameters EC, ED and ER, but also on N, the number of nodes in the random Boolean network. For example, in the first experiment N=7, EC=1, ER=17 and ED=40. Thus, at each step the energy E just before dividing will be Ek+1 = Ek+ER-EC*N = Ek+10 (see Fig.4-29). Thus, after every 4 time steps enough energy is left to divide safely, yielding an optimal cell cycle of period 4. A smaller period will lead to premature death when the cell tries to divide without having enough energy, and a larger period is non-optimal as more offspring could be produced.

Experiments

Cycle with period 4

```
Cell model:

Energy ER received per time step     17
Energy EC needed per node update      1
Energy ED required for division      40

Steady State GA:

Population size                     100

Random Boolean Network:

N = Number of nodes                   7
K = Number of inputs                  3
```

Table 4-1. The parameters for the experiment. Note that crossover and mutation parameters are irrelevant because an optimal solution was found immediately after randomization.

In the first experiment, the parameters are such that an optimal state cycle would have a period of 4 time steps, as discussed above. The relevant parameters for the experiment are listed in Table 4-1.

```
step     :  0  1  2  3  4  5  6  7  8  9
division :  0  0  0  1  0  0  0  1  0  0
E before : 10 20 30 40 10 20 30 40 10 20
E after  : 10 20 30  0 10 20 30  0 10 20
```

Fig.4-30. The energy over time in the best solution found, a ‘cell cycle’ with period 4.

As it happened, the problem turned out to be so easy that an optimal solution was found in the first generation, i.e. by pure random sampling. As expected, the best cycle found has a period of 4, and the energy fluctuation in the cell is shown in Fig.4-30.

A more complex cycle

Cell model:	
Energy ER received per time step	17
Energy EC needed per node update	1
Energy ED required for division	25
Steady State GA:	
Population size	100
Number of mutations per genome	1
Crossover probability	1
Size of tournaments	7
Randomize probability	0
Random Boolean Network:	
N = Number of nodes	7
K = Number of inputs	3

Table 4-2. The parameters for the second experiment.

In the second experiment, I tried to evolve a less trivial state cycle than the one above. In the first experiment the performance function (maximum number of division events) could just be realized by first finding a state cycle where the dividing bit is switched on and off regularly, and then tuning its period (by mutation of the random Boolean network) to the optimal length. As we have seen, such an optimal cycle was found right away, indicating that there are many such networks with 'good' cycles and in fact the genetic algorithm is not needed: random sampling would do just as well. In contrast, we will now set the parameters in such a way that a simple state cycle with one division per period cannot be optimal, thus making the problem less trivial.

```
step     :  0  1  2  3  4  5  6  7  8  9
division :  0  0  1  0  1  0  0  1  0  1
E before : 10 20 30 15 25 10 20 30 15 25
E after  : 10 20  5 15  0 10 20  5 15  0
```

Fig.4-31. A potential strategy to yield maximum offspring with ED=25.

As listed in Table 4-2, we will set the amount of energy that the cell gets per step to 10 units (in fact ER=17, of which 7=EC*N are subtracted right away when the next state is calculated). The amount of energy needed for division (ED) is set to 25. So, a strategy has to be devised which is more complex that a simple period with one division, as otherwise the energy allotted is not used optimally. An example of a working strategy is shown in Fig.4-31, where the state cycle has a period of 5 but two divisions are made per state cycle.

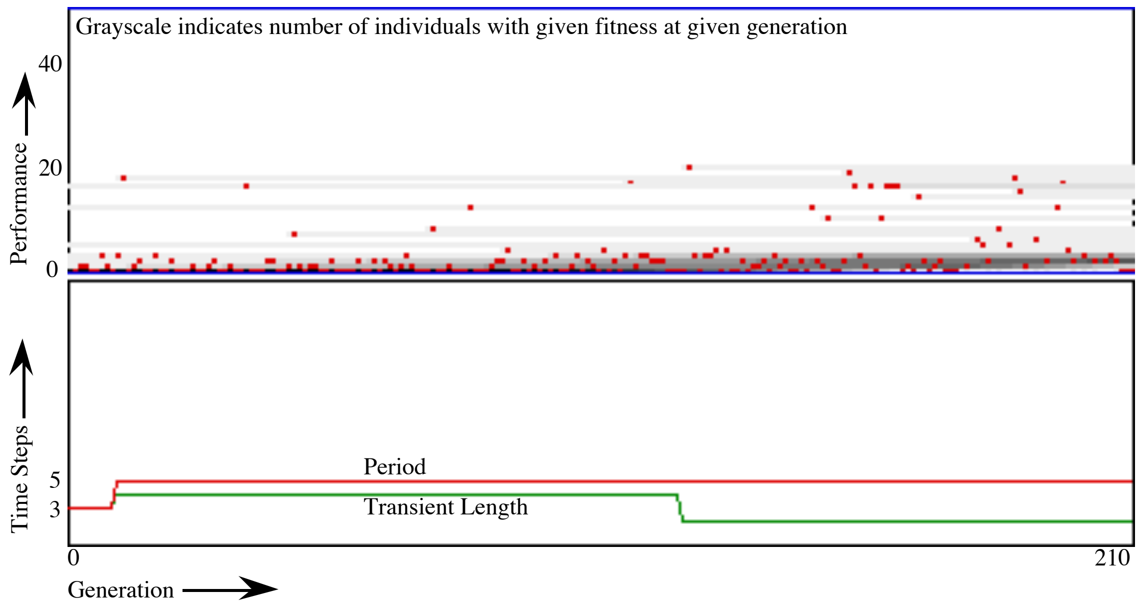


Fig.4-32. Top panel: Fitness distribution over time in the second experiment (see text for an explanation). Bottom panel: transient length and period length of best state cycle of each generation.

```
step    :   0  1  2  3  4  5  6  7  8  9
division:   0  0  0  1  1  0  0  0  1  1
before  :  10 20 30 40 25 10 20 30 40 25
energy  :  10 20 30 15  0 10 20 30 15  0
```

Fig.4-33. The optimal strategy found by the GA after 116 generations.

The solution found by the GA, is shown in Fig.4-33. Fig.4-32 shows how the distribution of fitness in the population changes over time, and the characteristics of the state cycle with the best performance in each successive generation. The fitness distribution is shown as a histogram that changes over time: the vertical axis is divided into bins of increasing fitness, and a gray scale pixel indicates how many individuals in the population are in that bin. The horizontal axis represents time. The solution found by the GA is different from the one that was proposed before, but just as good. This result is not surprising, since the maximum offspring criterion specifies nowhere that this offspring has to be obtained as soon as enough energy is available, as it was in the proposed solution. What matters only is the amount of offspring at the end of the lifetime of the cell, which is the same in both cases. This serves as another illustration that the GA will be opportunistic in the interpretation of the performance function, which may be different from what we *thought* it was.

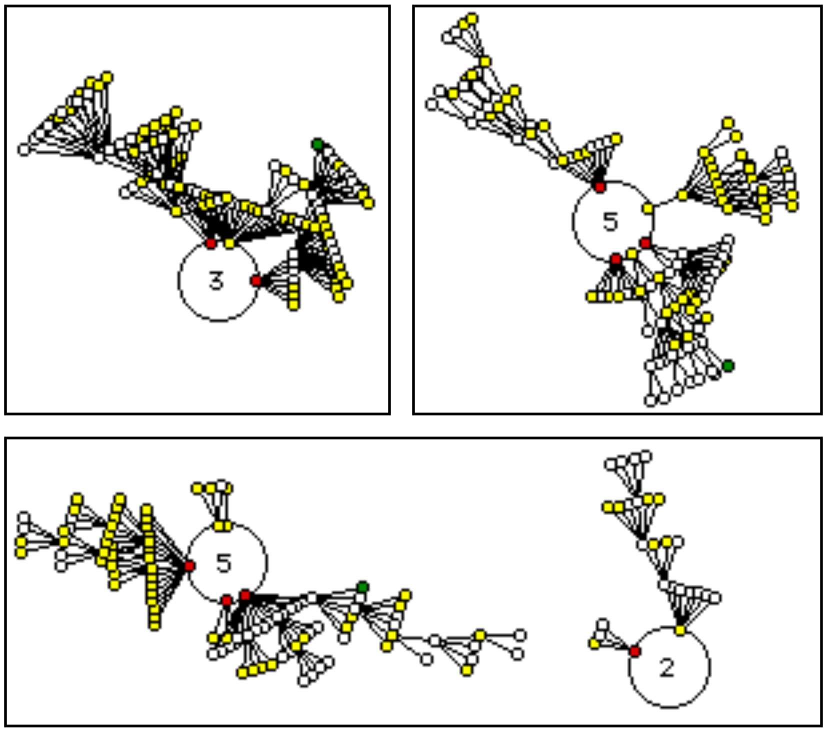


Fig.4-34. The phase portrait of two intermediate solutions and the best solution found by the GA in the second experiment.

In analyzing the intermediate solutions found during the experiments, the dynamical systems concepts developed before proved to be a powerful tool. Immediately from looking at Fig.4-34, showing the phase portraits of the intermediate solutions found during the second experiment, we can get a qualitative understanding of what these evolved networks are about: the first intermediate solution (top left panel) has a state cycle of period 3, with one of the three states signaling division (light gray states have the division bit set to TRUE), which is sub-optimal. Indeed, this network (found in the initial population) produced 16 offspring instead of the optimal 20, having 1 division event per three steps as opposed to the optimal 2/5 ratio. The next intermediate solution finds the appropriate 2/5 diving ratio, with its cell cycle of period 5 and two division events per cycle. The transient length from the initial state (the dark gray state in the transient tree) is still sub-optimal, however, and when this is shortened to 2 instead of 4 the optimal solution is the result, as shown in the bottom panel of the figure.

Evolving Switching Circuits

One of the most important phenomena in development is that of induction, by which at a certain moment in time, a cell can change its long term developmental fate in response to a signal from another cell. Naturally, this change can only be effected by a change in the pattern of gene expression inside the cell in question. In addition, this change in gene expression should be stable, as it is known from developmental biology that the inductive signal is not needed to keep the cell on the new developmental pathway it has been ‘asked’ to go down. Thus, what is needed is some kind of genetic switch: in response to a transient signal, one or more genes should switch on or off, and remain that way even after the signal has disappeared.

It is important that such switching behavior can be easily evolved as part of the behavioral repertoire of a single cell. Indeed, if is relatively easy, we are in good shape. If it is not, difficulties may be expected when trying to evolve complicated developmental programs. Thus, the ability to evolve regulatory circuits that display this switching capability is very important for the evolution of development itself.

Experiments

In what follows, we will try to evolve regulatory circuits exhibiting three different types of switching. The first one is the simplest one, and corresponds more or less directly to the inductive process: we will give the network an input line on which we supply a signal, and we ask that in response to that signal a certain genetic element turns on and stays on thereafter irrespective of what happens next on the input line. In the second experiment, we will now supply two lines, and ask that in addition to the switching on, the genetic element will also switch off in response to a pulse on the second line. Finally, in the third experiment, the second pulse that should turn the genetic element off will come in on the same line, in effect realizing a simple flip/flop.

In the experiments, the input to the network is given by adding a dummy node (or two, in the case of the second experiment) to the random Boolean network, with a negative index, in this case -1 (We will use the same trick to implement intercellular communication during development of our artificial organisms, see below). This node’s output will be clamped to the input signal, and the other nodes can wire their inputs to the input signal by wiring to the dummy node -1 (and -2 in the case of the second experiment).

In testing the performance of the evolved regulatory networks, we will look at the behavior of one of the genetic elements, the *output node*, in response to the input signal. We want the output node to switch on and stay on in response to a pulse on the input line, as shown in Fig.4-35. The performance function of the network on one trial sequence is the length of the time sequence subtracted by the hamming distance between the desired sequence and the sequence actually produced, as indicated in the figure.

```
t       : 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
in      : 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
desired : 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
out     : 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
correct : 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1

-> score = 13/15
```

Fig.4-35. Example of incorrect switching behavior in the first experiment: *out* should become high at the same moment the pulse appears on the input line, and stay on afterwards. The performance of the net for this particular sequence is 13 out of 15, as it produced an incorrect output for two time steps (t=3,4).

Each individual network of the population used in the GA is evaluated 100 times, where each trial will be different because of the introduction of two random factors: (1) before each sample, the state vector of the net is randomized (except for the state of the output node, which starts off consistently as 0, i.e. inactive); (2) the pulse width of the input varies between 1 and 3, randomly: this is to keep the GA from finding solutions that depend on the pulse width being 1, as I found to be the case in earlier experiments. The onset of the pulse is also variable, but not in a random way: it sweeps from step t=3 to t=12 in the 10 consecutive trials.

```
Steady State GA:

Population size                     20
Number of mutations per genome       5
Crossover probability              100
Size of tournaments                 10
Randomize probability                0

Random Boolean Network:

N = Number of nodes                  5
K = Number of inputs                 4
Initial Ratio of External inputs   50%
```

Table 4-3. The parameters for the second experiment.

The parameters that were used in all three experiments are shown in Table 4-3. The number of nodes in the random Boolean network was chosen to be fairly small (N=5) leading to a reasonable size of the search space. However, the number of inputs per node was fairly high (K=4), increasing the search space but at the same time leading to a large number of regulatory circuits per given randomized network. Each one of these circuits can possibly be part of the solution, and thus the role of the genetic algorithm is to select the good circuits out of there and increase their robustness with respect to the introduced noise (as discussed above). The last parameter, initial ratio of external inputs, signifies what fraction of the inputs will have a negative value when the networks are first randomized.

Although a mutation rate of 5 mutations per genome seems fairly high, consider that a N=5, K=4 network is described by 100 numbers (for each of the 5 nodes, 4 input tags are specified plus one lookup table with 16 entries, totaling 100 numbers) and thus in each mutation step only 5% of the genome is changed. In addition, some of these mutations will have no phenotypic effect, i.e. they will not change the phase portrait of the system or, alternatively, they may change the phase portraits but still have no consequence on performance (which is determined only on basis of the network's switching capabilities).

On switch

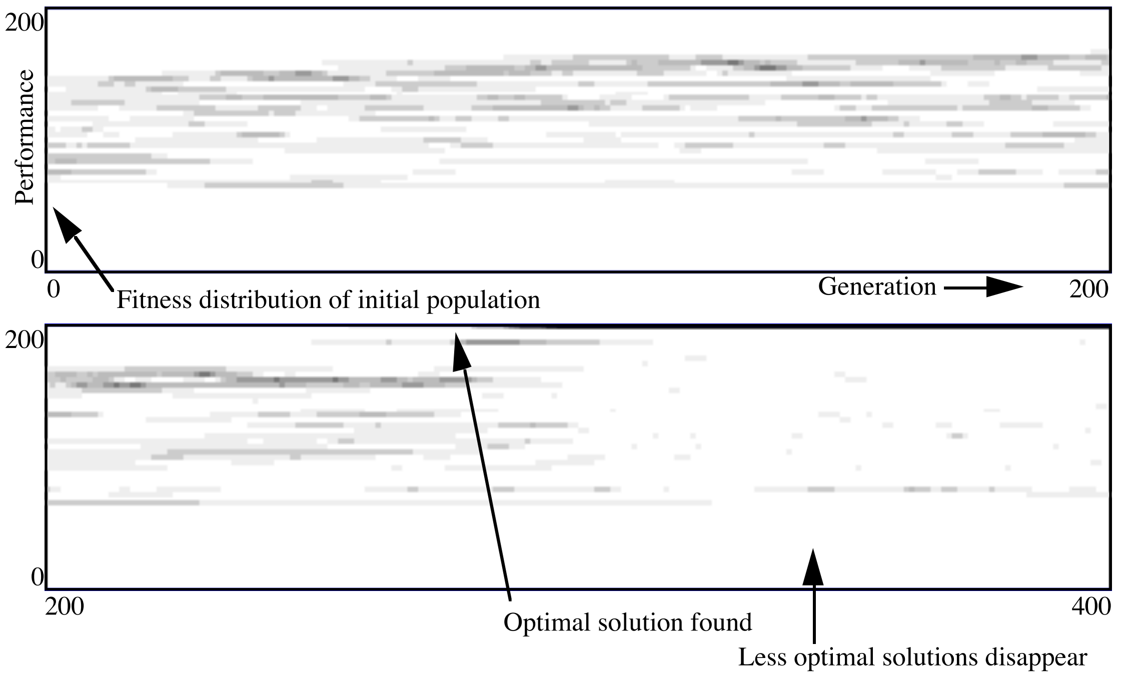


Fig.4-36. Fitness distribution over time for the first experiment. Maximum (and optimal) fitness is 200.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
     *%*% % % % %*%*% % % % % % % % % %
       *%*%*% % % %*%*%*% % % % % % % %
         *%*%*% % % %*%*%*% % % % % % %
           *%*%*% % % %*%*%*% % % % % %
             *% % % % % %*% % % % % % %
               *%*% % % % %*%*% % % % %
                 *% % % % % %*% % % % %
                   *% % % % % %*% % % %
                     *%*%*% % % %*%*%*%
                       *%*%*% % % %*%*%
```

Fig.4-36bis. The output of the evolved regulatory network for the first experiment (see text).

The fitness distribution of the population over time for the first experiment, the on-switch, is shown in Fig.4-36, and the output of the evolved regulatory network in Fig.4-36bis. 10 trial sequences are shown, each representing one sweep of pulse onset. In the first sequence the pulse starts at t=3, and lasts for 2 steps. The '*' symbols represent the input pulse, and the '%' is shown when the output signal is high. Thus, as you can see, right when the pulse appears, the output signal goes high. The network is not fooled by variable pulse width, nor by the second pulse that comes in after a while. Interestingly, if we evolve a network with the same performance function but without the second control pulse, networks are evolved that will be fooled by a second pulse. Thus one lesson learnt is that, if you want a network behavior to resistant to noise, it has to be present during the evolutionary process.

Two-line on/off switch

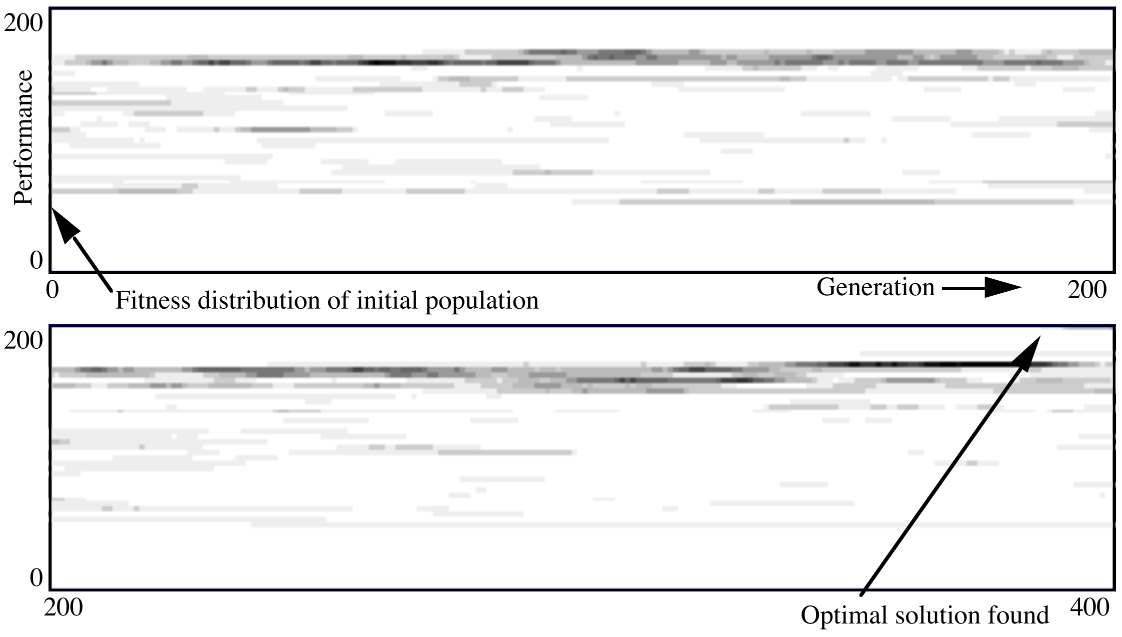


Fig.4-37. Fitness distribution over time for the second experiment.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
     *%*%*% % % %* * *
       *% % % % % %*
         *%*%*% % % %* * *
           *%*% % % % %* *
             *%*%*% % % %* * *
               *% % % % % %*
                 *% % % % % %*
                   *% % % % % %*
                     *%*% % % % %* *
                       *% % % % % %*
```

Fig.4-37bis. The results for the second experiment, in which a pulse on a second line signals the output node to turn off.

In the second experiment the network is given two inputs, and the goal is to evolve a network which will switch on (i.e. the output node becomes high) in response to a pulse on the first line (first input), and switches off when a pulse comes in on the second line.

One-line on/off switch (flip-flop)

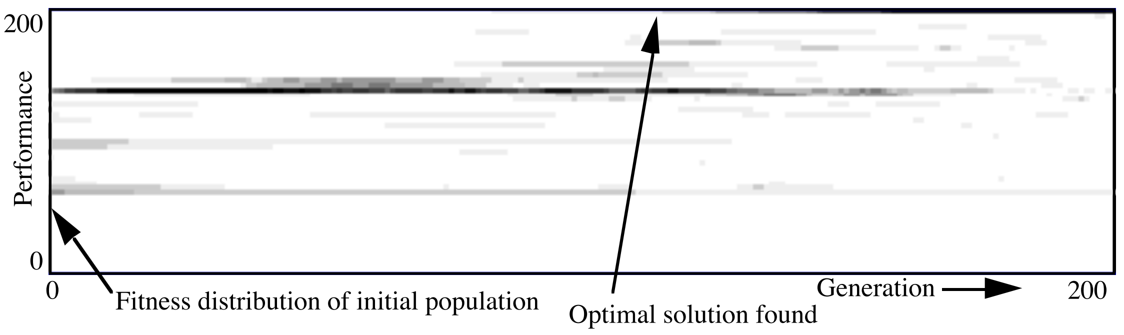


Fig.4-38. Fitness distribution over time for the third experiment, the flip-flop.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
     *%*% % % % %* *
       *%*%*% % % %* * *
         *%*%*% % % %* * *
           *%*%*% % % %* * *
             *% % % % % %*
               *%*% % % % %* *
                 *% % % % % %*
                   *% % % % % %*
                     *%*%*% % % %* * *
                       *%*%*% % % %* *
```

Fig.4-38bis. The results for the third experiment, in which the off-pulse comes in on the same line.

The last experiment is similar to the previous one, but now the off-signal comes in on the same input line. Thus, the input is the same as in the first experiment, but now we want the network to switch off, not stay on. The distribution of the fitness over time for this experiment is given in fig.4-38, and the input-output diagram in Fig.4-38bis. Note that it took much less time to evolve this ‘flip-flop’ behavior than was needed in the other experiments. This indicates that this is easier to evolve a circuit that switches off than one that stays on in response to a second pulse.

Discussion

The GA was able to find correct solutions in all three experiments, even though the networks were required to overcome the variability in their initial state and in the input signals presented to them. Of course, noise of a type other than that was present during the evolutionary process may still disrupt the operation of the networks as they were not evolved to cope with such disturbances.

Non-Autonomous Random Boolean Networks

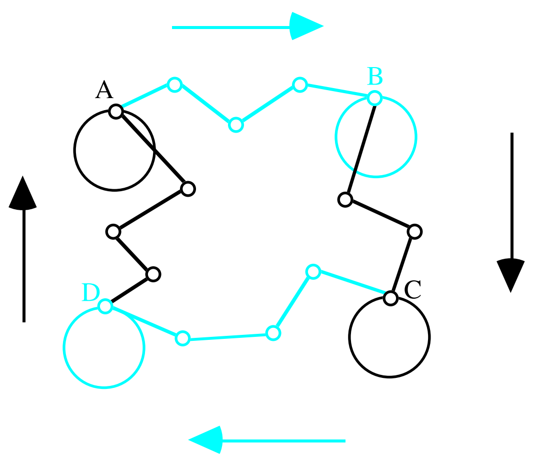


Fig.4-39. Two alternative phase portraits of a network are shown here superimposed, describing a bistable network. If the system starts in attractor A, and then the input is applied, the second (gray) phase portrait is selected, and the system evolves towards B. When the input is gone, the net evolves to C. Likewise, we can switch from C back to A via D.

It is interesting to view these evolved networks from the dynamical systems perspective we have developed above. In fact, during the switching, the phase portrait of the network is changed by the incoming signal, so that the system evolves into a different fixed point attractor (Fig.4-39). Then after the signal disappears, the original phase portrait appears, but the state of the system should stay in the newly reached attractor (or at least one that is not the same as the original attractor it was on before the induction signal). It will be informative to analyze the evolved networks that are capable of switching in this way.

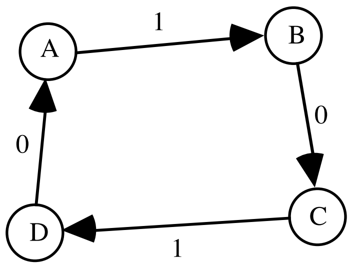

$$\begin{tikzcd} A \arrow[r, "1"] & B \arrow[d, "0"] \\ D \arrow[u, "0"] & C \arrow[l, "1"] \end{tikzcd}$$

Fig.4-40. The higher order state transition diagram for the example given in Fig.4-39. The numbers next to the edges indicate for what value of the input the transition will occur.

It is useful to introduce a new concept at this point: *non-autonomous random Boolean networks*. Until now we had not really considered inputs to the regulatory networks, i.e. the networks were completely autonomous. However, when you add an input to the system, the phase portrait, and thus the way the state of the network will evolve, will change as a function of the value of the input. If you have two (binary) inputs, there are four possible phase portraits etc... The input in fact serves as a phase portrait selector. This concept is very useful in a developmental context, since cells are constantly receiving inputs from each other and the environment.

We can develop a *higher order transition diagram* for these non-autonomous random Boolean Networks. Given that we are only interested in fixed point attractors, the complete phase portraits - shown by means of the state transition diagram with all the transient trees on them - will give us more information than we will actually need. However, we can construct a higher order state transition diagram, that only displays these fixed point attractors. The higher order state transition diagram for the network of Fig.4-39 is shown in Fig.4-40. Now we can look how the system evolves when - with the system state on a fixed point attractor - the phase portrait is suddenly changed by an input change. There are two possible reactions the system can have. One possibility is that the old attractor is also an attractor in the new phase portrait, in which case the system will stay put. Alternatively, the system will evolve to a new attractor, the one that happens to own the basin of attraction in which the state finds itself after the input change.

Analysis of the Evolved Switching Circuits

In this section the concept of non-autonomous RBNs is applied towards the evolved networks that were evolved above to display switching behavior. Only the on-switch and the flip-flop are analyzed, but a similar analysis could be made for the two-line switch of the second experiment.

On switch

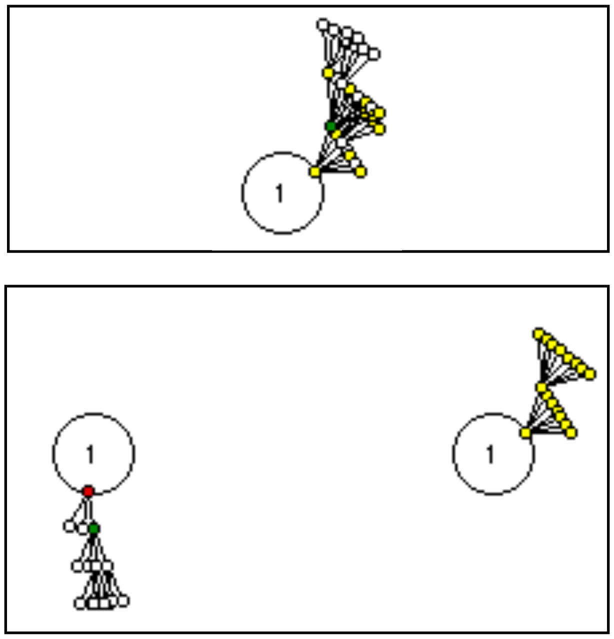


Fig.4-41. The two phase portraits for the on-switching regulatory network. Top: input 1. Bottom: input 0.

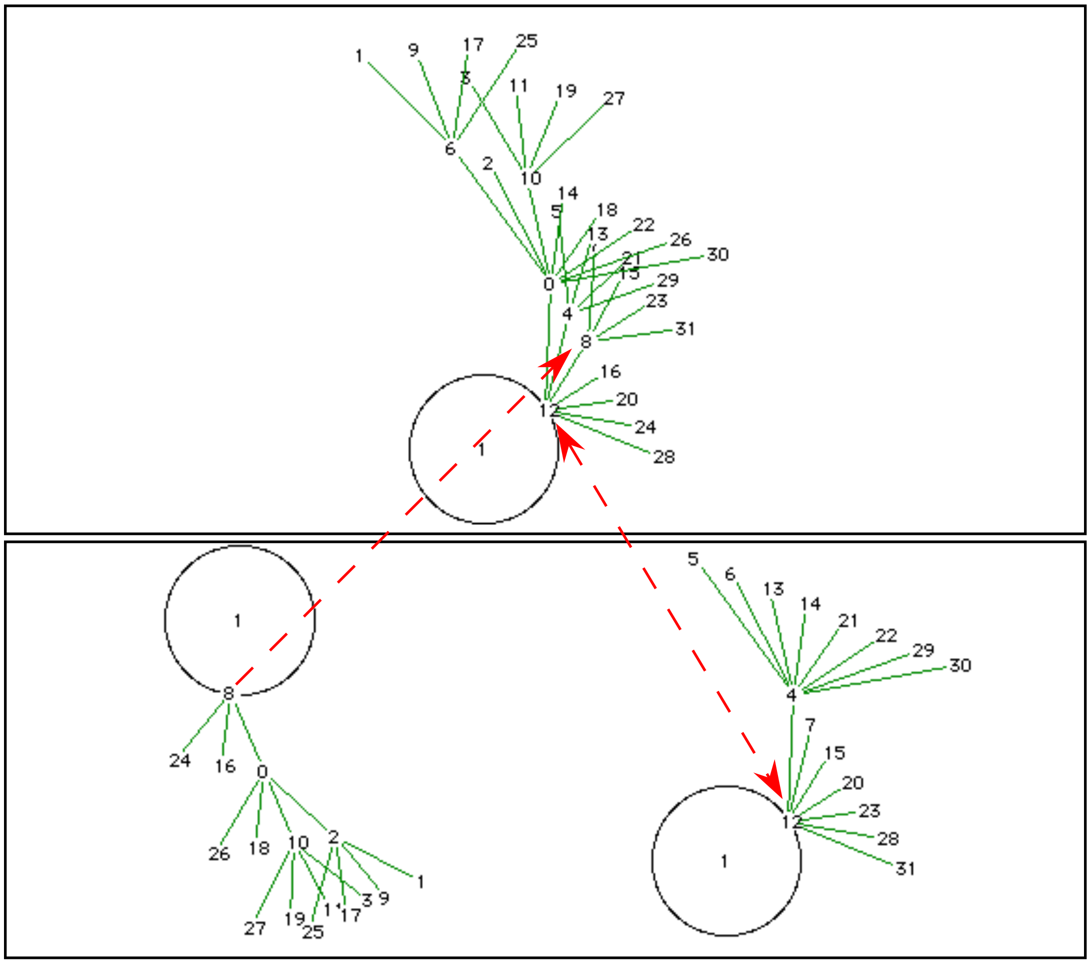


Fig.4-41bis. A more detailed rendition of Fig.4-41, with the decimal values of the states shown instead of open circles.

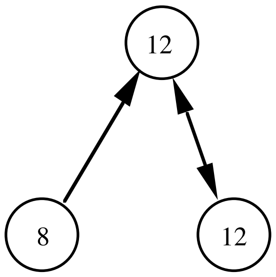


Fig.4-42. The higher order state transition diagram that can be deduced from the above phase portraits.

In Fig.4-41 and 4-41bis I have shown the two phase portraits for the network that was evolved in the first experiment (on switch), and the corresponding higher order state transition diagram is shown in Fig.4-42.

Initially the state of the network is 0, with input 0, and thus the phase portrait is the one in Fig.4-41 on the bottom. As you can see from Fig.4-41bis, the network will traverse the transient 0 -> 8 and settle down into the fixed point attractor 8. Then, when the input pulse comes in, the top phase portrait is selected, and here 8 is no longer a fixed state but one of the states in the transient tree of the fixed point attractor 12. This transition is indicated by the dashed arrow connecting the state 8 in the bottom phase portrait with that one of the top. Subsequently, the network evolves towards the state 12 and stays there. When the pulse goes away, the bottom phase portrait is valid again, but nothing changes as the state 12 is also a fixed point attractor in the bottom phase portrait. Any subsequent pulses will not affect the state of the system, which stays in state 12, with output high. Thus, the pulse had as effect to switch the system from the fixed point attractor 8 into the attractor 12. A higher level state transition diagram which shows only the fixed point attractors is shown in Fig.4-42.

Flip flop

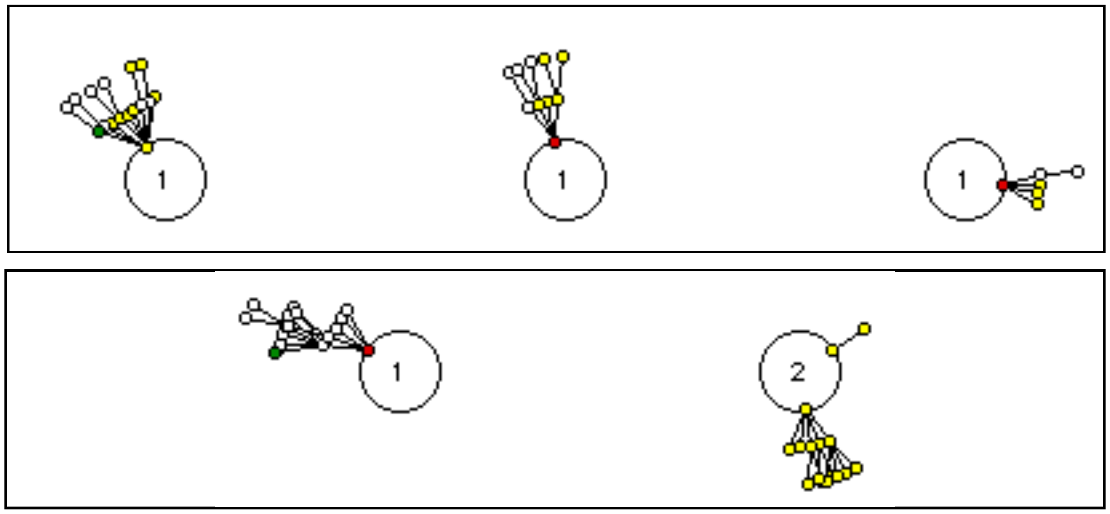


Fig.4-43. The two phase portraits for the flip/flop. Top: input 1. Bottom: input 0.

In Fig.4-43, 4-43bis and 4-44 the same kind of figures are shown for the flip/flop that was evolved. The same analysis of how the state of the system evolves in response to input changes can be made by looking at the first two figures. The initial state is 0 in the bottom phase portrait, after which the state evolves towards the fixed point attractor 10. When the on-pulse comes in, the system moves towards fixed point attractor 23. Then, when the pulse disappears, the system settles down in the state cycle 14-12. For the output of the system this does not matter, however: in both states the output of the system is 1, and when the off-pulse (the second pulse on the input line) comes in, both of the states will lead to a transition which eventually brings the system back to the attractor 10. Thus, from the higher order state transition diagram it is clear that the network implements a bistable system that switches between the attractors 10 and 14-12 in response to the input pulses.

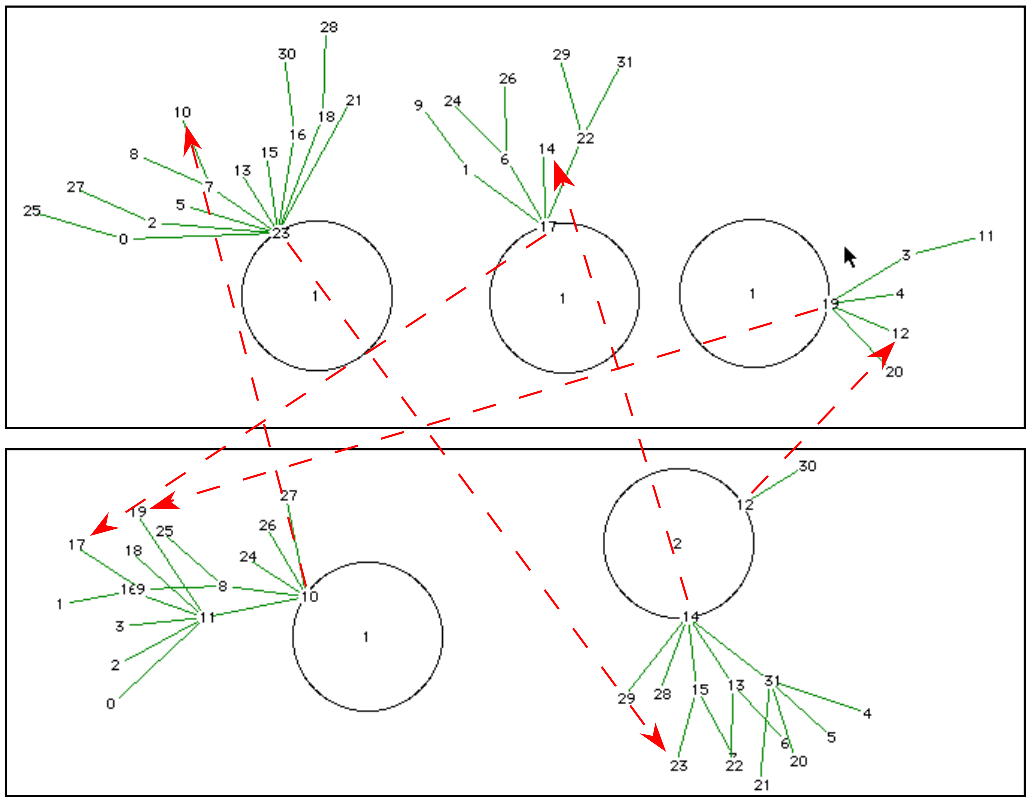


Fig.4-43bis. A more detailed rendition of Fig.4-43, with the decimal values of the states shown instead of open circles.

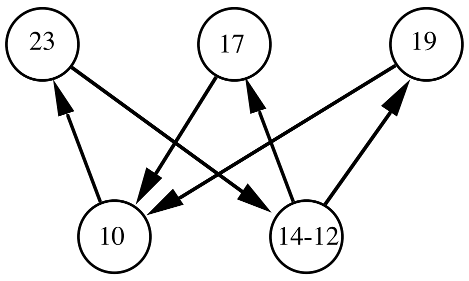


Fig.4-44. The higher order state transition diagram for the flip/flop.

Discussion

Thus, in this section we have seen how alternative phase portraits are selected by the inputs to the network, and we have illustrated the use of higher-order state transition diagrams. This is important, as we will correlate fixed-point attractors to cell type, and these diagrams provide valuable insight into how, in reaction to external signals and in interaction with each other, transitions from one cell type to the other occur.

Chapter 5

Modeling Multicellular Development

In the previous chapter a model has been developed for genetic regulatory networks, and we examined how circuits that may be important for development could be evolved in a unicellular context. Here we will look at a simplified, yet biologically defensible model for multicellular development. Parts of this material have been presented earlier in (Dellaert and Beer 1994).

Introduction

In biology, each multicellular organism starts out as a single cell, the zygote, originating when egg and sperm mix their genetic contents by the process of fertilization. After a series of rapid and synchronous divisions, a complex process of development unfolds, resulting in the emergence of an adult organism (see Fig.3-1).

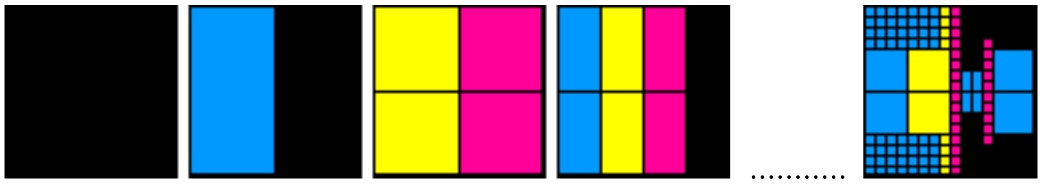


Fig.5-1. Starting from a 'zygote square', the developmental simulation discussed in this chapter yields a complex, spatially organized square as the 'adult' organism.

In parallel with the biological process, the model of development that will be presented here starts with a simple two-dimensional square representing the fertilized egg. This so-called 'zygote square' will start to divide as well, and after a *simulated* developmental process a complex multicellular square will emerge, as shown in Fig.5-1. The cells in this 'adult' square organism will not all look the same, however, but will exhibit a spatial pattern of organization. It is the mechanisms by which these patterns emerge that will interest us here and which we will try to capture in the model.

Like in biology, each cell in the developmental model will contain a simulated genome, being a random Boolean network as developed in the previous chapter. It is clear, however, that in addition to genetic regulatory networks, processes at the level of the cell and phenomena that involve the coordination of multiple cells will have to be modeled, too. In what follows we will attempt to do just that, by analyzing the process of development in terms of three levels of complexity, and addressing how each of these three levels is represented in the model.

Modeling Development

The developmental process unfolds simultaneously at three different levels, each of which will need to have a counterpart in the model: at the level of the organism, of the cell and at the biomolecular level. At the topmost level, a single zygote develops into a multicellular organism by a complex epigenetic process: eventually, groups of cells will stick together and coordinate their actions to form tissues and organs that make up the entire organism. This happens because at the cellular level, individual cells go through a sequence of determination and differentiation events that enable them to take up their specific role in the developing embryo. Ultimately responsible for this unfolding sequence, however, is the genetic information contained within each cell, which brings us down to the level of molecular biology. Although each cell has the same copy of the genome, different genes are expressed in different cells, which in turn leads to their difference in behavior. Thus, this pattern of differential gene expression lies at the heart of the developmental process.

The *genetic regulatory network* will be the first principal component of the model. As fleshed out in the previous chapter, I believe that, for the purposes at hand, the essence of the unfolding pattern of differential gene expression at the genome-level is best captured by modeling a network of interacting genetic elements. Each element in this network corresponds to the existence of a gene product or the expression of some gene. We have modeled such genetic regulatory networks by means of random Boolean networks, where the expression of each genetic element is governed by a Boolean function, taking input from the activity of other genes. A dynamical systems framework has been presented for these networks, and we have looked at how this model can be used to evolve simple regulatory circuits, of the kind that will be needed in a developmental context. The higher order state transition diagrams developed for non-autonomous random Boolean networks will be especially useful when analyzing developmental phenomena, as in development the cells must be viewed as a collection of closely coupled dynamical systems, rather than as independent entities as we did before.

The second component of the model consists of a very simple cellular simulator to model development at the *cellular level*. Eventually, every action directed by the genome should have a consequence at the level of the cell, if it is to have an effect on development. It follows that we will need to construct a model for how a cell behaves and the way the genome can influence this behavior. In the previous chapter, I have already presented a very simple cell model when trying to evolve cell cycles. However, to model multicellular development, a more sophisticated model is needed. One approach might be to build a complex, three dimensional model mimicking the way biological cells behave inside a multicellular organism, but modeling at that level of detail is currently unfeasible from a computational viewpoint. In addition, there is no need to have such a complex cellular model to be able to simulate the fundamental aspects of development. Thus, in what follows a very simple two-dimensional cellular simulation will be presented, that nevertheless captures many of the essential properties of cells that are crucial to development.

Finally, the last aspect of the model will cover all phenomena at the *organismal level*. As we know, during development cells interact continuously: in biological development, cells communicate by touch as well by chemical signals (Walbot et al. 1987, page 4). This intercellular communication is extremely important, as it can change the pattern of gene expression in the participating cells. Thus, it will have to be taken into account in the model. Another issue that transcends the cellular level is that of external influences, such as how symmetry is somehow broken at the very first stages of development. These aspects will be modeled at the level of the organism.

While implementing these components, one is confronted at each step by the trade off between simplicity and biological defensibility, as ultimately the model is to be used in conjunction with the genetic algorithm. Typically, when you want to evolve autonomous agents for instance, populations of hundreds of individual organisms are used. In the developmental model, each of these consists of hundreds of cells, and in each cell a genetic regulatory network must be evaluated hundreds of times. It is obvious that with such numbers you want to keep the model as simple as possible to keep the computational demands reasonable. Although many aspects of biological development are important and even crucial for biological life forms, the standpoint taken here is that some of them can be left out in a simplified model without invalidating the results obtained, depending on the questions you are addressing, obviously.

Genetic Regulatory Networks

As stressed many times before, the essence of development is to be found in differential gene expression. In biology, any multicellular organism is in fact a highly organized collection of cells, where cells in different parts of the body have differentiated into specialized cell types. For example, a liver is composed of liver cells whereas the brain is composed of neurons. These cell types are very different, and this is the case despite the fact that in both cell types, the same genetic information is present. Indeed, the difference lies not in the genome, but in the set of genes that is being actively expressed inside each cell: genes that code for special enzymes only needed in liver cells will not be expressed in neurons. Conversely, genes coding for neurotransmitters will remain unexpressed in liver cells.

Similarly, in the developmental model a simulated multicellular organism will consist of a collection of cells that will all contain the same genome, but might display different patterns of gene activity. As said before, the genome will be modeled by random Boolean networks, and by consequence we can restate this in the familiar terms of the previous chapter: each cell will have an identically configured random Boolean network - i.e. the same wiring and update rules in each node - but *the state vector that describes the pattern of activity in each cell will be different* in the various ‘cell types’.

The state of each cell can differ in different parts of the organism because each cell may have received different inputs from other cells and the environment during its developmental history. Cells will influence each other by means of intercellular communication, and they might also receive clues as to their position in the organism. For example, information will be available to the cell on whether it is on the inside or the outside of the organism (see below).

The genome of the simulated organism, i.e. the random Boolean network (specified by giving the wiring and the Boolean function for each node), will completely determine how and when the cells will diverge from each other in the value of their state vector.

Indeed, the net determines three things:

1) The way the cell reacts to signals. How a cell reacts to these signals will depend on the state the cell is in: as the activity of particular genes can modulate the way another gene reacts to a change in one of its inputs, it is clear that the total pattern of gene activity will determine the way the network will react to a change in external information.

2) The signals this cell gives to other cells.

3) The way the cell behaves. How the state vector is interpreted to yield a different behavior for the respective cell types is a problem that will be discussed below when we talk about the model at the level of the cell. Dividing can yield different positional information.

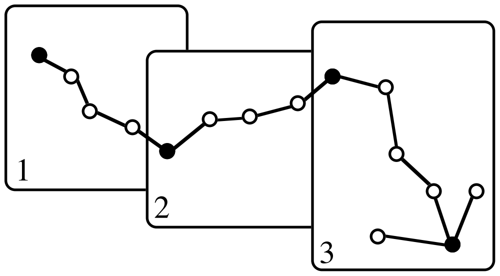


Fig.5-2. The developmental history of a simulated cell will be a succession of trajectories that will move the state of the system form attractor to attractor, each time the phase portrait of the network changes as a consequence some change in the external inputs to the cell.

This becomes much clearer when viewed from the dynamical systems perspective that we have developed before: at each moment in time, the state vector of a cell eventually settles into a fixed point attractor, and it can only be thrown out of this state by a change in its environment. Indeed, as the inputs to the cell assume new values, the phase portrait of the random Boolean network will change, and the attractor that the state resided on may disappear in this new situation (see Fig.5-2). If this is the case, the state of the cell will evolve towards a new attractor. Thus, the way the state vector will evolve in response to a change in the inputs depends both on the state the cell was in (i.e. its developmental history) and on the phase portrait that is selected by the new inputs (i.e. the nature of the signals received by the cell), since that determines what attractor the state will evolve to after the input change.

The configuration of the random Boolean network genome completely determines the multicellular organism, as all the information that influences the unfolding developmental process is contained in it. Because of this, a change in the developmental process can be brought about by a change in the configuration of the network. Thus, when using the genetic algorithm to evolve multicellular organisms, it will be these networks that will be mutated and combined by the mutation and crossover operators, respectively.

Cellular Level

At the cellular level we will have to model the properties of the cell that play a role in the developing organism at the higher level and that can be influenced by the genetic regulatory networks at the lower level. These properties include the physical characteristics of the cell, the cell cycle controlling the cell's behavior and how a cell differentiates into a particular cell type, and they will be discussed here. The aspects of biological cells that were not modeled will also be touched upon, along with the justification for leaving them out.

The Physical Characteristics of the Cell

When modeling the physical characteristics of the cell, we are looking for a model that is both simple enough to be efficiently simulated, and yet captures enough aspects of the biological cell so that it will work within a developmental model. Two properties of cells seem crucial in this respect. First, a cell has to have some form of physical extent: for the simpler cellular model we used in the previous chapter, we did not need to know anything about the spatial layout of the cell, because it was irrelevant to the evolution of a state cycle. However, much of what development is about is the formation of spatial patterns, hence here it is important that we know where a cell is relative to its neighbors and what space it occupies. Second, a cell has to be able to undergo cell division, a process underlying both growth and cellular differentiation. These two properties constitute an extreme simplification of what a real cell is actually capable of, but the goal here is to reduce development to its bare bones underlying mechanisms and, if necessary, build up from there (as we will do when extending the model towards neural development, see below).

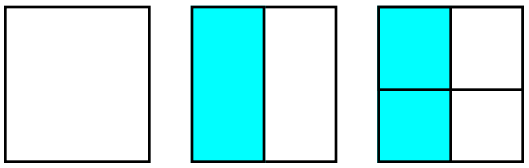


Fig.5-3. Zygote square dividing two times to yield 4 ‘cells’, or squares

Thus, we will simulate the physical appearance of a cell by a simple, two-dimensional, square element that can divide in any of two directions, vertical or horizontal. If division occurs it always takes place in such a way that the longest dimension is halved, and the two resulting daughter cells together take up the same space as the original cell. This very simple approach has as a consequence that we do not have to deal with cells changing shape as a result of cell division: after two cleavages the shape is again a square. See Fig.5-3 for an illustration.

The Cell Cycle

A simulated cell cycle consisting of two phases, interphase and mitosis, coordinates the updating of the random Boolean network state and cell division, respectively. In one organism, each cell has a copy of the same random Boolean network constituting the genetic information of that organism (see section about organismal level below). However, the state of the network, corresponding to the pattern of gene expression in a particular cell, may be different in each cell, as they underwent different influences during their life span or started out with a different initial state. In Fig.5-4 the cell cycle is depicted graphically, and we will discuss each phase here:

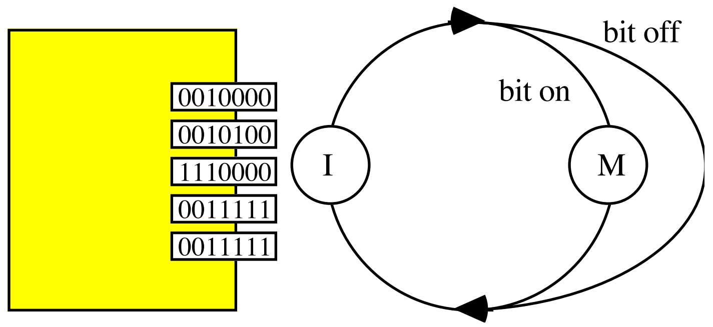


Fig.5-4. The cell cycles between interphase and mitosis. The cell will only go through mitosis if the gene governing cell division is active, otherwise nothing happens.

During the first phase, *interphase*[6], the network state is synchronously updated until a stable pattern of gene expression has been reached, corresponding - as explained in the previous chapter - to a fixed point attractor in state space. The assumption is made that each cell has enough time to reach a fixed point attractor, i.e. the transient trajectory through state space will not be too long. When evolving developmental programs, any organism where the transients are too long or the network state falls onto a *cyclic* attractor will be discarded. In the actual implementation of the model, we can conveniently test this in the same way: in each interphase period, we limit the number of time steps available to reach the next fixed point attractor. If the fixed point attractor is not reached in the allotted time interval, the state is either still on a transient or it is on a state cycle, and we can discard the organism. Note that the transient behavior of the network depends both on the original state vector as on the new environmental stimuli each cell is confronted with at the beginning of interphase (see below), and thus might be different in each cell: in fact, that is what *differential* gene expression is all about!

The activity of a particular, a priori designated gene will determine what the second phase of the cell cycle will look like: either this ‘mitosis’ gene is active, in which case the cell goes through *mitosis*[7] and divides into two daughter cells. If the ‘mitosis’ gene is inactive, the cell stays intact and just waits for the next interphase to start. Note that this makes the ‘cell cycle’ presented here somewhat different from its biological equivalent, as the cell does not necessarily have to go through mitosis. If a cell divides, however, the state of the cell is inherited, i.e. its state vector is copied to its two daughter cells. Note that in the following interphase, unless something in the

---

[6] We will use the term *interphase* to refer to this phase of the *model* cell cycle in what follows. Although this phase is based on the interphase of biological cells, it is important not to confuse them.

[7] Again, we will use *mitosis* to refer to the *model* cell cycle phase.

environment changes or there is some other external interference, the state vector of the cell will remain on the same fixed point attractor and nothing will happen to it, i.e. that pattern of gene expression will be passed on unchanged to the next generation of cells.

Differentiation into Distinct Cell Types

There are two ways in which we could model how the genome determines the final differentiation of a cell: by combinatorial specification or by using ‘master genes’. In biology, the combinatorial gene regulation theory hypothesizes that the cell can ‘detect’ a particular combination of regulatory proteins and thus is able to differentiate into the corresponding cell type. For instance, this type of mechanism is thought to underlie the division of the imaginal discs in *Drosophila* into sharply demarcated compartments (Alberts et al. 1983, page 846-847). In principle, three different genes would be sufficient to specify a unique address for each of the eight compartments formed. Alternatively, there could be several regulatory ‘master’ genes whose expression determines the expression of a whole gene batteries needed in a particular cell type. See (Davidson 1990) for a comparative overview of a number of cell fate specification mechanisms.

combinatorial	master genes	cell type
000	00000001	0
001	00000010	1
010	00000100	2
011	00001000	3
100	00010000	4
101	00100000	5
110	01000000	6
111	10000000	7

Fig.5-5. Final differentiation tables according to combinatorial specification or master gene approach.

In the implementation of the model I had provided for both these mechanisms and I could choose between them when running simulations. In combinatorial mode, a subset of genetic elements is chosen to determine the final differentiation of the cell. Every distinct combination of activity in these elements then corresponds to a particular cell type. In the other, ‘master gene’ mode, differentiation is related to the activity of one specific genetic element, with the additional constraint that there should be no conflict between competing cell types. The two approaches are illustrated in Fig.5-5. All the results reported in this thesis use combinatorial specification, as I have found that it takes considerably longer to evolve the additional mapping between the state of the network and the different ‘master genes’.

In the model color is used as an abstraction for cell type. Of course, when applying the model towards the synthesis of autonomous agents, the final differentiation of a cell will correspond to that cell being a sensor, actuator or control-neuron, or other cell types relevant in that domain. For the time being, however, it will be sufficient to simulate cell differentiation by the different colors that the cell can take on. This will enable us to demonstrate the different spatial architectures that can be explored using the model. In the combinatorial mode, a color can be assigned according to the settings of $\ln_2 C$ specific bits in the state vector, where C is the number of colors. In this chapter, C=8 most of the time and the three bits used are bit[0], bit[1] and bit[2].

Biological Properties not Included in the Model

Although many aspects of biological development at the cellular level are important and even crucial, some of them can be left out in a simplified model without invalidating the results obtained. Cell movement and coordinated cell sheet deformations, for instance, lie at the basis of all but the simplest morphologies encountered in multicellular organisms. However, they would make the model quite complex and much more difficult to implement, and we are probably better off to start exploring what is possible with 'simple' intercellular communication (see below) and genetic regulatory networks, rather than make the model too complicated from the start. Once it is clear what can be achieved with a simple model, it is certainly worthwhile to incorporate more complex mechanisms, as will happen when we get to neural development.

Organismal level

In the subsequent paragraph I will discuss how individual cells function within the organism and how two very important aspects of development, symmetry breaking and intercellular communication, are implemented.

The Organism as a Collection of Cells

The organism itself is implemented as a two-dimensional square consisting of many cells. Development starts out with one single square that represents the zygote (the fertilized egg), and whose random Boolean network state is initialized to zero except for the dividing bit, which is set to 1. Since this dividing element is now active, the zygote will always divide at least once. As discussed, whenever a square divides the two daughter cells take up the same space as the original one: there is no pushing away of neighboring cells or shape change involved, except that after an odd number of cleavages, cells may be rectangular in shape rather than square. The organism is then the collection of squares that originated from the 'zygote' square.

Symmetry Breaking in the Early Stages of Development

The way we will break the symmetry between the first cells at the early stages of development will have to be addressed, otherwise we will end up with an uninteresting, homogeneous collection of cells: because of the deterministic, synchronous updating and as all cells are descendants of the same ‘zygote’, they will have the same state vectors at each step unless something disturbs this symmetry. Biological development faces the same problem, and there are diverse mechanisms by which in early development the correct spatial pattern of differential gene expression is imposed (Davidson 1990).

Here the symmetry will be broken at the time of the first cleavage by assuming the existence of a ‘maternally’ imposed asymmetry in the ‘zygote’ square, that can lead to different patterns of gene expression in the first two daughter-cells. This is certainly biologically defensible, as in many organisms this anisotropic distribution of some entity is actually observed (Walbot et al. 1987, page 340-353). This will be simulated by consistently flipping a bit of the random Boolean network state vector in only one of the two daughter cells after the first cleavage event. If you will, the genetic element of which the state is flipped corresponds to an asymmetrically distributed determinant in the zygote.

A second spatial clue is introduced by supplying the developing organism with the notion of a midline. As will be explained below, it was sometimes necessary to provide more spatial clues than only the first cleavage symmetry breaking. Thus, the cells are also provided with information on whether they are adjacent to the horizontal midline of the organism, according to a bit flipping scheme similar to the one used in the first cleavage step (although this time a bit in the neighborhood vector is flipped, see below). Actual biological embryos get this midline notion for free because of the three dimensional topology in which they develop: as an example, in the frog embryo neurulation takes place along the dorsal midline of the embryo, which after gastrulation lies closest to the mesodermal germ layer that is responsible for the initiation of the process (Walbot et al. 1987, page 368-375).

Intercellular Communication

One of the key elements of the developmental process and consequently of the model is how the cells communicate. Indeed, following the initial symmetry-breaking the cells now have a rough plan for the positioning of major body structures. In all but the simplest organisms, however, a great deal of fine-tuning is necessary, and this can be achieved by intracellular communication or induction, i.e. the way in which one group of cells can alter the developmental fate of another group by providing it with some signal (Walbot et al. 1987, page 366). This can be done via a number of ways, one of which relies on the diffusion of small soluble molecules through specialized cell-to-cell junctions (gap junctions). Another way is for the cell to secrete 'signal' molecules that bind to receptors (specialized cell-membrane proteins that are specifically tuned to bind to certain molecules) on the surface of other cells; this binding then activates a chain of chemical reactions leading to the regulation of genes inside the receiving cell.

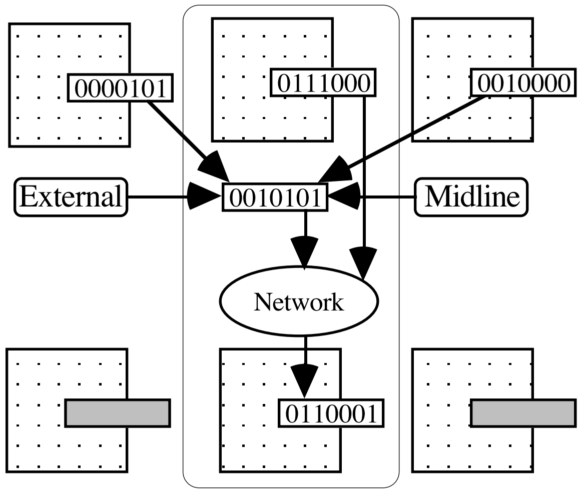


Fig.5-6. The state vectors of two neighbor cells are combined (OR) together to yield a neighborhood vector that is combined with the cell's state vector to determine the next state.

A lot of thought has been put into whether to model this by actually simulating the existence of cell-surface receptors and chemical 'signal' molecules. As an alternative, one could link the genetic regulatory networks in a more direct way, by letting their next state depend not only on their own state, but also on that of surrounding cells.

Thus, to implement induction a modified version of the random Boolean network was used. Whereas normally each node has K incoming edges from other nodes in the network (or recurrent), we now allow for some of these incoming edges to connect to nodes in an abstract 'neighborhood vector'. The latter is the logical OR of all the state-vectors of the neighboring cells. Fig.5-6 shows this arrangement. In the genetic description of the network, a negative connection parameter will imply that the corresponding input is taken from the neighborhood state vector, instead of from the cell's state vector.

This implementation implies that we have to keep track of the neighbor relationships of cells. Although this may sound an easy thing to do, it does actually complicate things somewhat, as the cells are not static entities but instead divide all the time. Thus, a scheme must be devised by which topological relations are constantly kept up to date. However, it becomes soon intractable to let each cell poll every other cell in the organism, because the number of cells rises exponentially in each organism. This has been solved by letting each cell pass on a list of its neighbors at the time of division and then letting each daughter cell poll these neighbors to check whether they are still adjacent. It is interesting to note that this was implemented using object-oriented techniques, thus allowing to easily substitute a more complex geometry for the 2-D square one, e.g. a three-dimensional geometry. A similar approach was taken in (Fleischer et al. 1994).

Another type of induction is the influence exercised by the external environment. This was modeled by reserving one bit in the neighborhood state vector for that purpose. It is forced to ON if the cell in interphase is at the border of the organism, otherwise it is OFF (See Fig.5-6).

Developmental Examples

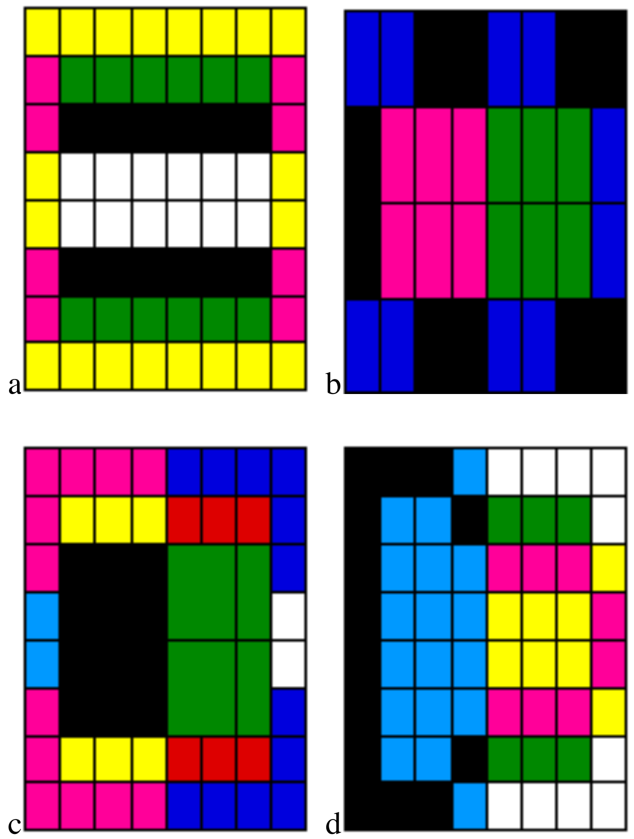


Fig.5-7. Four different examples that demonstrate the range of organisms that can be attained using the model of multicellular development.

In Fig.5-7 you will find some examples that demonstrate the range of organisms that can be attained using the developmental model. They were found using ‘biomorph mode’ (sitting down at an X-terminal and selecting the fittest individual according to subjective taste, see (Dawkins 1989)), and they exhibit interesting features that can conceivably be put to use in the context of autonomous agents.

Fig.5-7a displays an interesting 'layered' characteristic, with cell-types at the sides of the organism (it is facing towards the right) different from those in the middle, and with an intermediate layer in between. Note that in biological development the three germ layers exhibit the same spatial order: ectoderm to face the outside, endoderm at the inside and mesoderm in between them.

The organism in Fig.5-7b was selected because it has a segmentation property: you can easily discern a bilaterally symmetrical repeated structure at the sides of the organism. Finally, Fig.5-7c and d represent more complex morphologies, both asymmetric with respect to the vertical axis and having more detailed patterning at the rostral side.

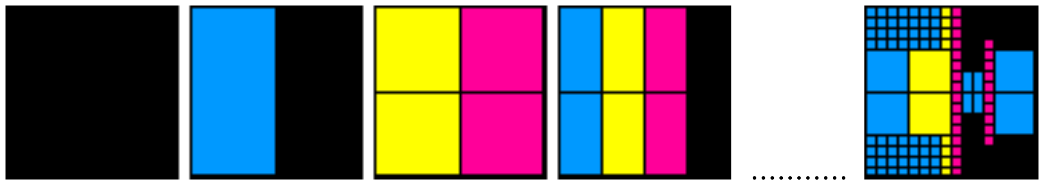


Fig.5-7bis. An example where different rates of cell divisions during the course of development can lead to quite complex 'adult' organisms.

In Fig.5-7bis a more complex organism is shown, along with its developmental sequence. Because certain cells divide more often than others during the course of development, some parts of the organism are more finely divided than other parts.

Evolvability

One of the things to look at is how the model behaves when used in conjunction with a genetic algorithm. To investigate this, I devised a simple *performance function* that maximizes the number of colors, taking care that no color is represented more frequently than any other. Although this particular criterion has no direct relevance to autonomous agents design, it is nevertheless useful to examine how the discoveries made by evolution serve in maximizing this function. The formula used to calculate the score for a particular organism was the following:

$$score = 2^p \sum_{c=0}^{7} \sqrt{n_c} \ ,$$

where n_c represents the number of occurrences of color c, and p is the total number of different colors present. This is a multilevel performance function, meaning that there are two levels at which selection occurs: (1) large scale jumps in performance occur when a new color is discovered, as then p increases and the score is doubled; (2) after each large scale jump, a smaller scale fine-tuning component will try to balance the relative occurrences of the colors present:

taking the square root of n_c for each color ensures a diminishing return when n_c becomes large, so less represented colors become more important. The perfect score for a 64-cell organism and 8 possible colors is

$$2^8 \sum_{c=0}^{7} \sqrt{8} = 5792.6.$$

Genetic algorithms did a good job in finding random Boolean networks that could steer the developmental model so that the fully developed organism optimized some performance function. This is a strong result: there is no obvious relationship between the setting of a bit in an update rule of the genetic regulatory network and the performance function to be optimized. The color of a square in the final design is quite far removed from the particular wiring of the network. In addition, the organism is evaluated only at the end of the full developmental process, so that any mutation in the model genome (i.e. the random Boolean network) must not only be beneficial from a performance function viewpoint to be incorporated in the population, it must also take care not to interfere with the existing developmental process in a 'wrong' way.

Also, the computational overhead induced is not as bad as one might expect when introducing a model with so many different elements. A typical simulation with population size of 20, network parameters N=6 and K=2, and a maximum of 64 cells per organism takes about 10 minutes to at most half an hour on a Sparc 10, depending on the performance function and the mutation rate. Typically, mutation rates of 0.1 and crossover probabilities of 0.5 were used.

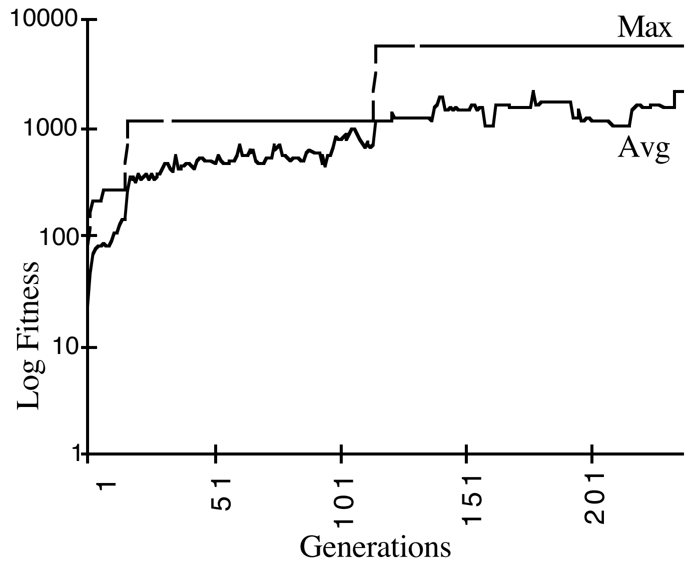


Fig.5-8. Maximum fitness and average fitness for about 240 generations.

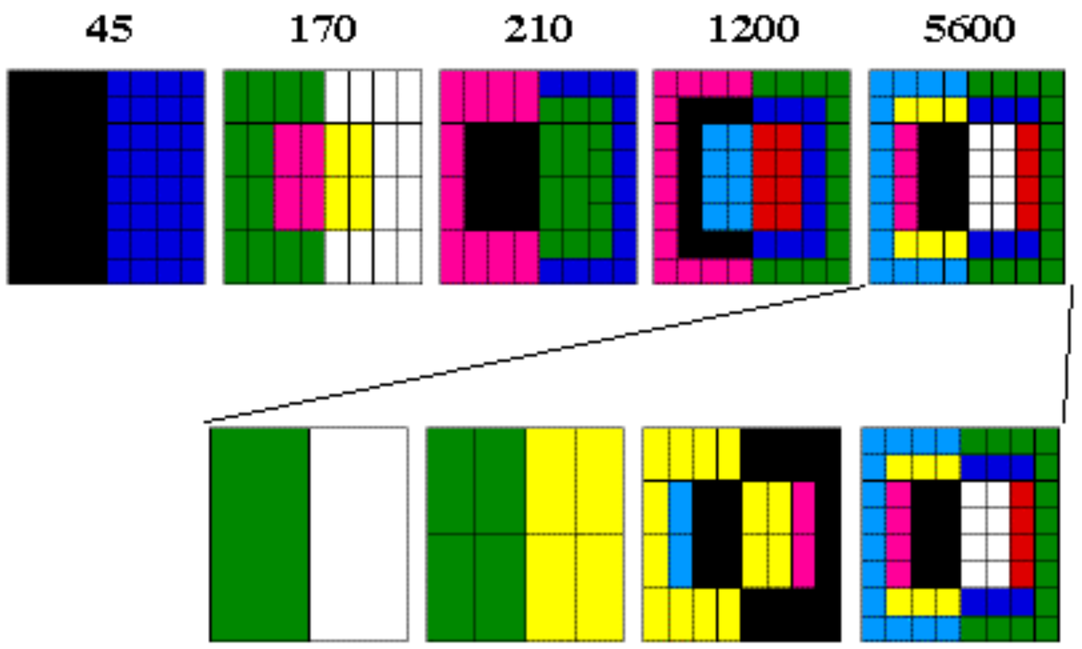


Fig.5-9. The best individual after each fitness jump during evolution. The respective performance values of these individuals are 45, 170, 210, 1200 and 5600. Of the last individual, the developmental stages are also shown.

Fig.5-8 and Fig.5-9 show the results for a typical run. Fig.5-8 shows the maximum and average fitness of subsequent generations during the run of the genetic algorithm. The GA used for this experiment had elitist selection, i.e. the best individual is never thrown away, which explains the step-like manner the maximum fitness evolves. In Fig.5-9 five organism are shown, each a snapshot of the best individual of its generation. The snapshots were taken just after a jump in fitness occurred: the way the performance function was constructed, this corresponds to the discovery of a new color. The last organism has discovered all eight colors.

In the first individual, one can immediately see evidence of the asymmetry that was introduced at the time of the first division: all individuals that did not make use of that were discarded from the first sample, as it is very easy for the GA to 'discover' this asymmetry. The score of this first organism is easily calculated, as we can see that there are p=2 colors and 32 cells per color, yielding:

$$2^2(\sqrt{32} + \sqrt{32}) \approx 45$$

Note that all evolved organisms shown are bilaterally symmetrical. This is a direct consequence of how the model is set up: the only asymmetrical stimulus is the first bit-flip, and the other external stimulus is the environment, which is symmetrically introduced at all sides[8]. Because of the synchronous updating of the networks, no other asymmetries are introduced. Thus, in this model symmetry is gotten for free.

The next discovery made by the GA is that of the external environment. Notice that in the second square there is a difference between the center and the border cells of the organism. Together with the asymmetry, the developmental process is able to specify 4 colors, yielding a score of 170. However, the cells in the organism with score 170 have stopped dividing after 5 times, yielding a sub optimal score for 4 colors. This is partially corrected a couple of generations later, when the organism with score 210 emerges: here all the cells at the anterior side of the organism divide 6 times, yielding more cells and thus a better score.

In the further course of the evolutionary process, the previously formed layers themselves provide information for new cells to assume different colors, by the mechanism of induction. This is evident in the fourth organism, with score 1200. In this organism, 6 colors are present, although not optimally distributed, as some colors are more prominent than others.

Finally, the last organism shown has discovered all eight colors, and its developmental sequence is reminiscent of the discoveries made by the GA during the time span of the experiment. In Fig.5-9, the successive stages of development that this individual goes through are also shown. As you can readily observe, the steps that development goes through follow the 'discoveries' made in the course of evolution: asymmetry, external environment, induction. Thus, although this ought to be examined much more systematically, there is some evidence here of 'recapitulation': the developmental sequence reflects the evolutionary history of the organism. This is one of the great strengths of the developmental model: evolution is able to gradually build on previous discoveries, and extend them towards fitter organisms.

Development of a Simple "Agent"

To demonstrate how the model of multicellular development operates in detail, this section will analyze the development of a multicellular organism with agent-like properties. A simple organism has been evolved that exhibits the relative placement of sensors, actuators and control system of the kind one would like to see in a simple chemotactic agent. Any attempt at the design of autonomous agents using a developmental model will have to deal with morphological features such as

[8] In this particular run there is no notion of a midline, as introduced in the previous section.

these. For the simple task of chemotaxis, for example, one would wish for a bilaterally symmetric organism with sensors and actuators placed sideways at the front and the back, respectively, and a control structure or ‘neural tissue’ connecting them.

4	**4**	1	1	1	1	**2**	**2**
4	**4**	1	1	1	1	**2**	**2**
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
4	**4**	1	1	1	1	**2**	**2**
4	**4**	1	1	1	1	**2**	**2**

6	*6*	*6*	*6*	*6*	*6*	2	2
4	*0*	*0*	*0*	*0*	*0*	*0*	2
4	*0*	*0*	*0*	*0*	*0*	*0*	*4*
1	1	1	1	1	1	1	*1*
1	1	1	1	1	1	1	*1*
4	*0*	*0*	*0*	*0*	*0*	*0*	*4*
4	*0*	*0*	*0*	*0*	*0*	*0*	2
6	*6*	*6*	*6*	*6*	*6*	2	2

color	1	2	4
wanted	48	8	8
match	16	6	2
perc	33.33	75.00	25.00
sqrt	5.77	8.66	5.00
bonus	1300.00	1300.00	1300.00
cells	640.00		
score	4559.43		

Fig.5-10. The ‘seeker’ organism, the goal template and the calculated score.

The performance function used by the genetic algorithm to evolve the organism was a simple pattern matching algorithm. In Fig.5-10 the goal template is shown, alongside with the best organism found, which has been termed ‘seeker’, and a calculation of its score. A number-coded representation is used for the different cell types or colors, 4 representing the actuator cells, 2 the sensors and 1 the ‘neural tissue’. The performance function used to calculate the score of an organism was the following:

$$\text{total score} = \text{number of cells} * 10 + \text{sum over colors} (s_c)$$

where s_c is calculated for each requested color c:

$$s_c = [\text{anywhere}]*300 + [\text{right place}]*1000 + \text{sqrt}(\text{match percentage})$$

Thus, for each requested color the square root of the percentage of cells with matching color was taken and added together, with bonus points if a color occurs in the organism but not in the right place, and even more bonus points if a color did occur in the right place. The total score is also incremented with a 10 point bonus for each cell. This multilevel score calculation ensures that (1) the cells will divide as often as possible; (2) a ‘good’ color will not be thrown away, even if it is in the wrong place; (3) the square root calculation will tend to balance matching percentages when all colors have been found, and is in fact the fine-tuning part of the performance function. As you can see from the figure, the matching for ‘seeker’ was far from perfect: the value of its fine-tuning score was only 19.43, as compared with a value of 30 for a perfect score.

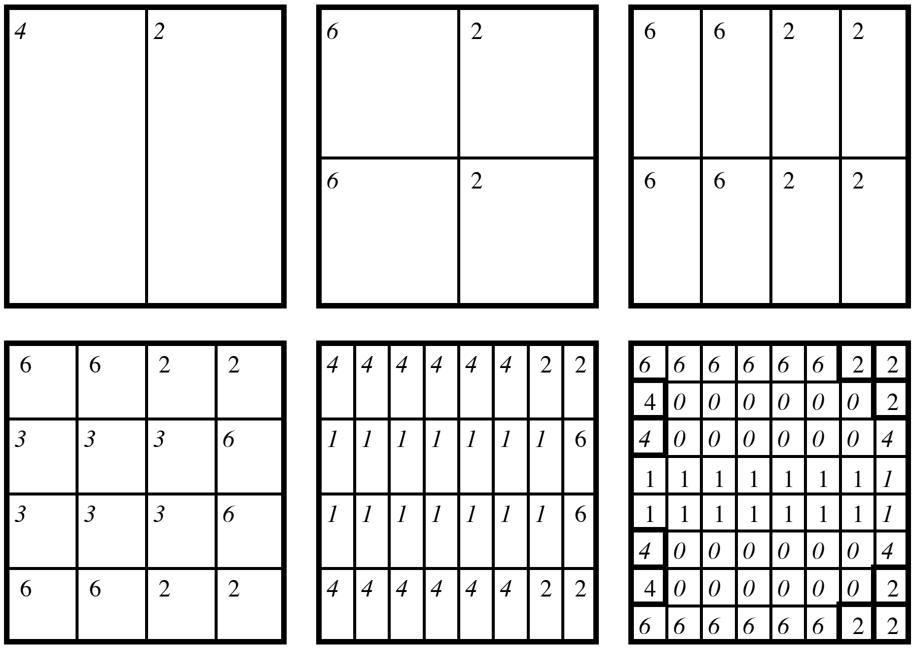


Fig.5-11. The six consecutive stages of development in the 'seeker' organism, with digits denoting cell types. Italic digits indicate that in the corresponding cell the color /cell type changed relative to the previous developmental stage.

In Fig.5-11 the developmental sequence of 'seeker' is shown. This figure will often be referred to during the following explanation, although in itself it does not tell us much about the underlying process of development. We have access however - in contrast to researchers in biology - to every variable at every stage of the developmental process, from the outward appearance of the cells up to and including the complete description of the genome. In the subsequent paragraphs this information will be analyzed to show what it can tell us about the sequence of events in the development of 'seeker'.

a)	0 1 0 1	
node	0 0 1 1	inputs
1	0 0 1 0	3 -6
2	1 1 0 0	-2 -1
3	0 0 0 1	-5 5
4	1 1 0 1	4 4
5	0 1 1 0	6 -6
6	0 1 1 1	6 -1

b)	Equivalent
node	Boolean function
1	~3 AND mid
2	~(-1)
3	ext AND 5
4	~4 OR 4 = TRUE
5	6 XOR mid
6	6 OR -1

Fig.5-12. a) the actual genome of the 'seeker' organism. b) The Boolean functions in a more readable form. 'mid' and 'ext' refer to the midline and external environment, respectively (see text for an explanation).

The genome, shown in detail in Fig.5-12a, specifies the wiring of the random Boolean network that models the regulatory network of 'seeker' (Fig.5-13) and the updating rules of each of the nodes (Fig.5-12b). The network has parameters of N=6 and K=2, so the genome consists of 6 update rules and 12 indices to designate the inputs to each node. Induction from other cells is modeled by a negative node index, corresponding to an incoming edge from outside the cell (actually from the neighborhood vector, see above explanation). The numbers -5 and -6 are reserved for conveying the influence of the external environment and the midline, respectively: if a cell is on the perimeter of the organism the value of bit '-5' will be TRUE and FALSE if not. Likewise, the value of bit '-6' is TRUE when the cell borders the midline of the organism, which runs horizontally across[9].

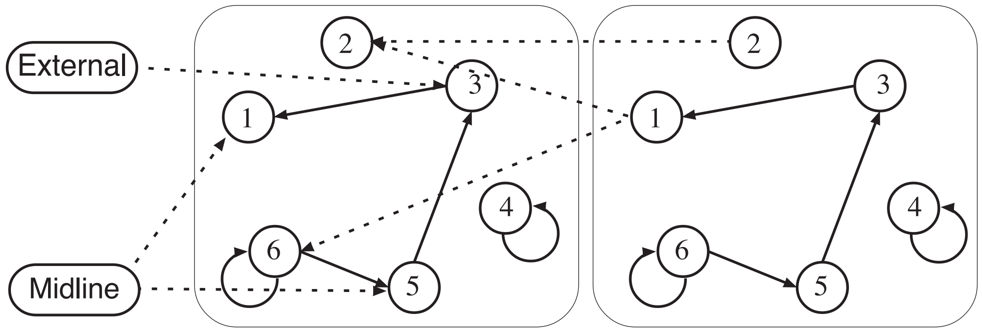


Fig.5-13. the 'seeker' wiring diagram: the dashed lines represent extracellular inputs. The 'midline' and 'external' have value 1 when the cell in question is on the midline respectively the perimeter of the organism.

The wiring of the network can serve very specific purposes: one thing that immediately catches the eye when looking at these figures is that both inputs to node 4 are recurrent connections and that the updating rule (~4 OR 4 = TRUE) ensures that the corresponding genetic element will be permanently active. This can be explained by the particular performance function that was used to evolve the organism, i.e. it rewarded a high number of cells in the final design: as bit 4 is used to decide whether to enter mitosis or not, the genetic algorithm found this positive feedback loop to ensure that division would take place at every step, resulting in a maximum number of cells.

At the very first stage, it is ensured that the zygote will divide at least once and that the resulting daughter cells are not completely alike, so that the symmetry is broken. The organism starts out as a single 'zygote' square with all but one genetic element inactive, i.e. zero state vector, except for the 'dividing bit' node 4, which is forced to 1. This will ensure that the cell division will take

[9] In the simulated evolution that led to this particular organism, a midline notion is only present from the 16-cell stage and onwards. Up to and including the 8-cell stage, bit 6 in the neighborhood vector is always 0.

place, dividing the zygote into two cells L and R (Left and Right). In addition, at the time of that first cleavage, bit 1 is set to 1 in one daughter cell and to 0 in the other, in order to introduce the asymmetry. We then have two cells with state vectors 000100 and 100100 respectively, as depicted in Fig.5-14. As the cell type or color is determined by the first three bits (least significant bit at left) this corresponds to color [0] at the left and color [1] at the right.

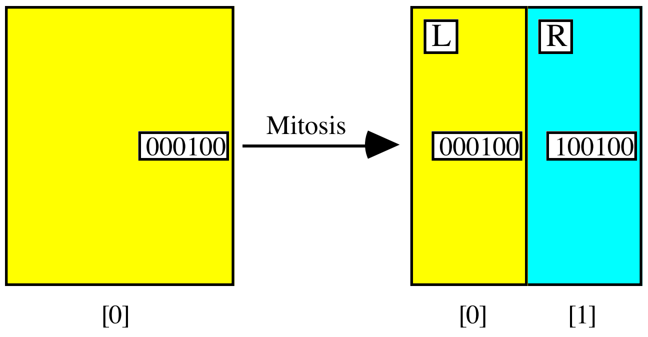


Fig.5-14. The zygote square divides at least once because the dividing bit 4 is forced to TRUE prior to the first mitosis phase. Colors are read from the first three bits in the state vector and are indicated in square brackets.

From now on interphase and mitosis will alternate until the final design of the organism is reached after stage 6 of the developmental process. We will look at the first stages in detail and then paint the broader picture when a detailed explanation becomes both tedious and too space-demanding. To understand the detailed picture, keep in mind that at each developmental stage a cell does three things: (1) it determines its neighborhood vector, (2) it repeatedly updates its state vector in interphase until a steady state is reached, and (3) it assumes a color and decides whether to divide in the next stage.

Cell L:
Neighborhood vector: 100110
Interphase: 000100 -> 000101 -> 001111
Color = [4], divide

Cell R:
Neighborhood vector: 000110
Interphase: 100100 -> 010100
Color = [2], divide

Fig.5-15. Before, during and after interphase in each of the daughter cells L and R.

After the first interphase, we will have reached the 2-cell stage of the developmental sequence depicted in Fig.5-11. As described in Fig.5-15, the two daughter cells of the zygote each go through the three steps described above before entering mitosis, and you can see that the colors now match up with the colors shown in Fig.5-11. The behavior of cell L will be examined in somewhat more detail: bit 6 switches to 1 because its updating rule (6 OR -1), as we know from Fig.5-12b evaluates to TRUE because of the positive induction from cell R: bit 6 is 0 but bit -1, i.e. bit 1 in the neighborhood vector, is 1, so 6 OR -1 = 1. To put this into more general terms, the activity of the genetic element 1 in cell R induces a change in the pattern of gene expression of cell L, whose perturbed stable state will now elicit a transient behavior in the regulatory network. The final pattern of gene expression is not reached until after a steady state is reached, however, which happens after one more synchronous update. The color of cell L can now be read from the first three bits, i.e. color [4].

Because cells inherit steady state vectors after mitosis, some change in the environment is needed to trigger a change in behavior and/or color, as otherwise the state vector will just stay on the fixed point attractor it has reached earlier. A change in the neighborhood vector will change the phase portrait of the system and the state will flow to an attractor in the new phase portrait. State cycles will not occur, as they were selected against by the performance function used in the genetic algorithm. Recall from before that we only allow a certain time to reach a fixed point attractor, and if not reached by then, the organism in question is given a performance of zero.

```
Cells LT and LB:
Neighborhood vector: 011110
Interphase: 001111 -> 011111
Color = [6], divide
```

Fig.5-16. The two left cells at stage 2 undergo a transition from color [4] to color [6].

In the ‘seeker’ organism, after cell L and R go through mitosis, we get 4 cells which will be denoted by LT, LB, RT and RB, where T and B stand for ‘Top’ and ‘Bottom’. As it happens, a perturbation of the stable state only occurs in the cells LT and LB, where the resetting of bit 1 in the neighborhood vector causes bit 2 to switch on (rule ~(-1)), resulting in color [6] for both cells after interphase settles down. The details are given in Fig.5-16, and the colors can be verified by looking at Fig.5-11: only the left cells have changed color from [4] to [6].

Now that we have looked in detail at the mechanism that underlies the transitions in cell color at a given developmental stage, we can at least qualitatively understand the subsequent stages of the developing organism of Fig.5-11. In stage 3, it looks as if all state vectors remain unperturbed because the colors are unchanged: when looking at the log of the simulation, it was found that this was indeed the case. To make this difference between ‘active’ and ‘inactive’ interface apparent, the colors in Fig.5-11 are printed in italics when they resulted from a triggered transient, i.e. the attractor that the state vector had reached disappeared when the input to the cell changed.

A ‘neurulation-like’ event takes place at the 16-cell stage: suddenly all cells lying around the midline of the organism undergo a color change. It is clear that this resulted from the influence of the ‘midline bit’ 6 in the neighborhood vector, that has value 1 for these cells but value 0 for the cells at the sides of the organism. This inductive step sets the stage for the specification of sensors and actuators away from the midline, and for ‘neural tissue’ in the middle. The reminiscence of neurulation is not altogether surprising as the midline concept was implemented with just that phenomenon in mind (see above).

A secondary induction event occurs at the 32-cell stage: all the cells of color [3], created by the ‘neurulation’ event in the 16-cell stage, in turn induce a perturbation in the cells around this group. Indeed, it can be verified from Fig.5-12b that bit 2, with rule ~(-1), will switch off in response to the now active genetic element 1 in the middle of the organism. This at least accounts for the change to color [4] respectively [1], for the cells that had color [6] respectively [3].

Eventually, via qualitatively similar interactions and influences from the external environment, the more complex picture at the last stage of development emerges.

The Case for a Three-State Gene Model

Although in most of this work a genetic element is modeled to be either on or off, i.e. it can assume a Boolean value, I have also experimented with a three state model. My hypothesis is that such a model might more accurately reflect the situation in eukaryotic versus prokaryotic cells, and one could speculate that the evolution of a similar mechanism as the one I am about to describe might explain why only eukaryotes are involved in the formation of multi-cellular organisms.

Eukaryotes are cells with organelles, i.e. the nucleus and many other specialized structures, for example mitochondria, that supply energy to the cell. Prokaryotes are cells of a much simpler design, without a cell nucleus: all bacteria are prokaryotes. Most of what we know about gene regulation has been learned by studying bacterial models, and in these systems the simple operon model holds true. However, such a simple mechanism has not been found for eukaryotic gene regulation. Although there is evidence for regulatory proteins (transcription factors), their interaction with the genome is apparently more complicated.

One of the big differences between prokaryotic and eukaryotic cells is in the way genes are kept around. In the prokaryote the genome is essentially a circular thread on which all the genes reside, openly exposed to the regulatory proteins. In the eukaryote the genes are packed into a complex structure called *chromatin*, consisting of specialized proteins (the histones). In their normal state, the genes are very densely packed, and the resulting structure is called *heterochromatin*. However, before a gene can be transcribed (i.e. converted into RNA) the dense heterochromatin has to be unpacked to result in euchromatin, which is partially unfolded. Only in this state, the RNA polymerase (the enzyme responsible for transcription) can interact with the DNA and transcribe the genes into RNA.

The implication is that there is an additional degree of regulatory control in eukaryotic cells that is not present in prokaryotes, and this might play a role in assuring that during development, the potencies of a cell are progressively restricted. Indeed, one of the key observations that has been made regarding development is that the potential for a cell to differentiate into a certain cell type is maximal in the egg (a totipotent cell) but as time goes by, the cells become more and more focused on a particular pathway. Quoting from (Walbot et al. 1987):

> *Commitment to a specific differentiation pathway is a stepwise process; as it takes place, a series of decisions are made between alternative fates, and the cell's developmental potential is increasingly restricted.*

There has been speculation that the stable packaging of genes in dense and inaccessible regions of the genome plays a role in this central aspect of development, although I am certainly not in a position to either deny or endorse that view. However, in (Walbot et al. 1987, page 134) it is said:

> *[Facultative heterochromatin] is condensed in some cell types and dispersed in others. Presumably, the facultative heterochromatic regions are transcriptionally active in some cell types but permanently shut down in others.*

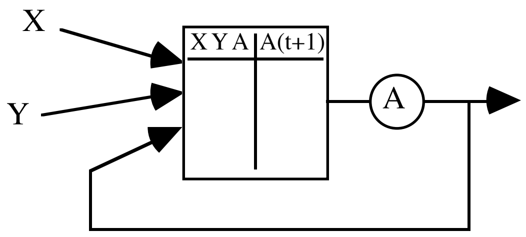
In addition, there seems to be evidence for other mechanisms by which an irreversible state of inactivation can be attained and passed on to daughter cells (for example methylation, see (Walbot et al. 1987, page 181).

Thus, it seems that there are several mechanisms by which irreversibly turning off genes can yield a progressive restriction of cell fate. It is not clear how such a stable phenomenon is easily come by in the case of prokaryotic cells, nor in the pure random Boolean network model for that matter. Also interesting to note is that, along with the cells having a richer structure and specialized organelles, the existence of these mechanisms is precisely what differentiates eukaryotes from their prokaryotic ancestors.

Be that as it may, I would like to proceed by building an argument for a three-state model of the gene that is not entirely dependent on the speculative content of the previous paragraphs, although they do provide the motivation and an essential background for what follows.

To start, let me refer back to the switching circuits that have been evolved earlier. Remember that switching behavior is something that would be very useful to have in a cell that participates in development: a cell can be switched to a different type of cell by the mechanism of induction, and we want that cell type to be stable, i.e. persist after the initial stimulus for differentiation has gone. For an example in biology we have to look no further than neurulation. As explained in Chapter 2, the presumptive skin cells are induced by underlying cells to become neurons, and, once embarked on this path, they have no further need for the original signal to be present.

It is evident that such a behavior requires some kind of cell memory, and we have seen earlier that when selecting for regulatory mechanisms displaying switching behavior, a mechanism was discovered by the GA to implement this memory. In effect, were we to design it ourselves, the easiest way to implement memory for a gene is to endow it with a self-connection. In that way, the genetic element can take its own state into account to determine its next state. Since switching is so important for regulatory genes, it would be interesting if one could give all the genes (remember that only regulatory genes are simulated, not structural genes) a self connection by default, so that it need not be explicitly evolved.



X(t)	Y(t)	A(t)	A(t+1)	Interpretation
0	0	0	0	REST
0	0	1	1	
0	1	0	1	ON
0	1	1	1	
1	0	0	0	OFF
1	0	1	0	
1	1	0	1	OSCILLATE
1	1	1	0	

Fig.5-17. The different ways in which an input signal (XY) can make a node A with memory behave.

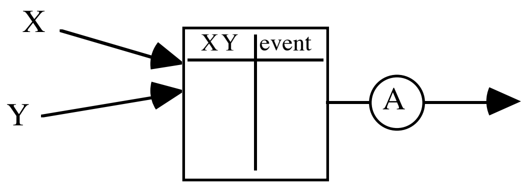
If we did that, each gene's behavior (in the case of the random Boolean network, its Boolean function) could be analyzed in a rather simple fashion. This is illustrated in Fig.5-17. Here a node A has three inputs to its Boolean function: X, Y and A (a self connection). The combination of X and Y can select 4 different regimes in the example: (00) REST : the input stays the same; (01) ON : since there are all 1 in the lookup table, the gene A will always switch to 1; (10) OFF, always to 0; (11) OSCILLATE: the state of the node A is reversed.

Although from a developmental designer point of view it is interesting to have the three first events, the OSCILLATE event results in oscillations of the gene that last until the stimulus disappears. The eventual state of the gene depends on when exactly the stimulus is gone, and the behavior of the cell is thus extremely sensitive to the precise duration of the stimulus. However, one might argue that this is in fact an artifact of the discrete synchronous updating that we have adopted in the multicellular model. In development these periods of exposure to stimulus will never be determined with such exactness, but they will in fact vary between organisms: yet development is able to proceed along its predicted path. Thus, we would like to prevent the OSCILLATE bit-from occurring.

In addition, since we are interested in switching behavior, it would be nice that mutations between all remaining events would take place with equal probability. As it is now, a mutation of one bit of the Boolean function will never be able to mutate from an ON event to an OFF event in one step, but has to proceed via either the REST or OSCILLATE bit-pair.

Summarizing, if we were to modify the model to accommodate these considerations, the following issues should be addressed:

1. all genes should have memory by default
2. the OSCILLATE event should not occur
3. equal probability of mutation between ON, OFF and REST events.



X(t)	Y(t)	Interpretation
0	0	REST
0	1	ON
1	0	OFF
1	1	OSCILLATE (not allowed)

Fig.5-18. New lookup table for the example of Fig.5-17.

If we now drop all references to the self-connection and provide a new lookup table that would contain ON, OFF or REST instead of respectively 11, 00 and 01, and not allow the OSCILLATE entry, we have resolved all issues at once. The genetic description would contain these bit-pairs as atoms, as it were, and mutation would automatically be equally probable between all three possibilities. The new lookup table of the example would look like Fig.5-18: the OSCILLATE event is not allowed though, so it would not occur. This is a much more compact way of specifying a genetic element with memory.

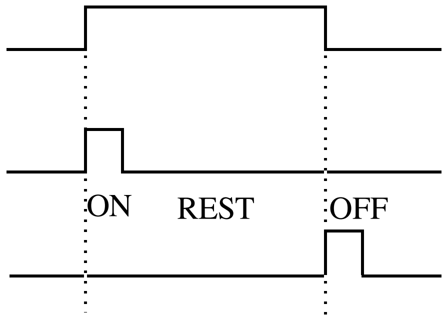


Fig.5-19. New behavior of Boolean gene.

Thus, as shown in Fig.5-19, we can simply switch a genetic element between states by providing the necessary input signals: first ON, thus XY=01, then REST XY=00 and finally OFF, XY=10. Evolving switching networks with this model would be a trivial problem, almost.

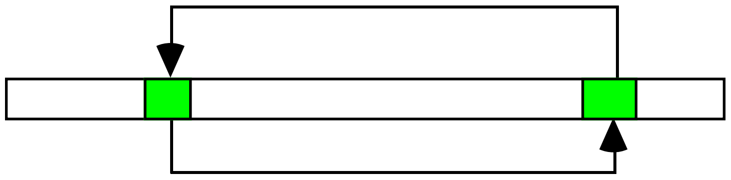


Fig.5-20. two gene oscillation.

Now, by implementing this ‘atomized events model’, the oscillation in response to stimulus is gone, but oscillation by the interaction of two genes is still possible (Fig.5-20). It is this kind of interaction that clouds the descendance of a certain cell type. From experience with the model, I know that cells often fall back into states that have been assumed earlier in the artificial developmental sequence, in contrast with biological development, where the lineage of a cell is one way only. This point is illustrated in Fig.5-21. These gene oscillations are worrisome also because of this: if they have the same at two different moments in developmental time, it implies that they have the same potentialities, and this seems to contradict the observed facts of progressively restricted potential in biological development we talked about earlier (and which are so consistently observed in biological development).

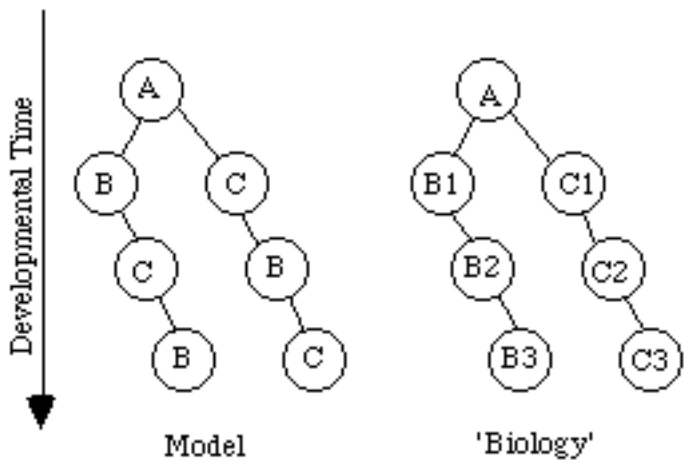


Fig.5-21. In the model, a 'dance between cells' can exist, unlike in biology, where the lineage are one way streets.

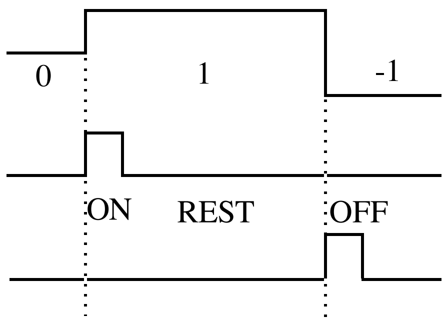


Fig.5-22. The 3-state model.

We can resolve this problem by implementing a three state model, where each genetic element would have three states: at first, it is in a 'sleep' state, it has never been expressed before. Then, the second state, the 'active' state, is reached when an ON event switches the 'sleeping' genetic element on. Finally, an OFF event switches an active element into the 'off' state, from which it can never become active again. These three states can for example be implemented by letting a genetic element take on the values {0, 1, -1}, instead of just {0, 1}. By integrating the three state model immediately with the 'atomized event' model discussed above, we have a new and consistent model.

By using this model, we also achieve a very clear cut view on the developmental process, since the graphs describing cell type transitions will reduce to the nice tree-like graphs we would find in biology. Consider for example the cell type transition graphs for the lineages of Fig.5-21, shown in Fig.5-23.

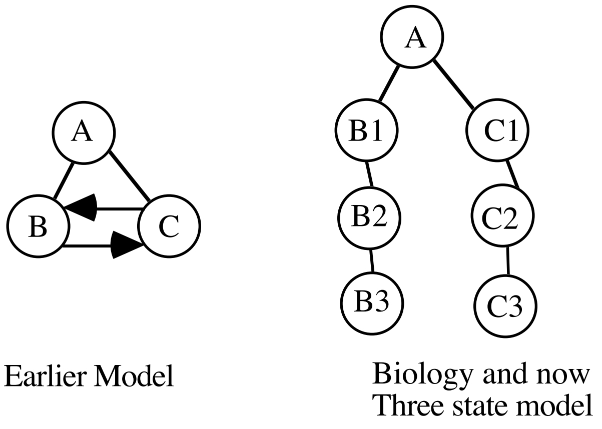


Fig.5-23. a tree instead of graph.

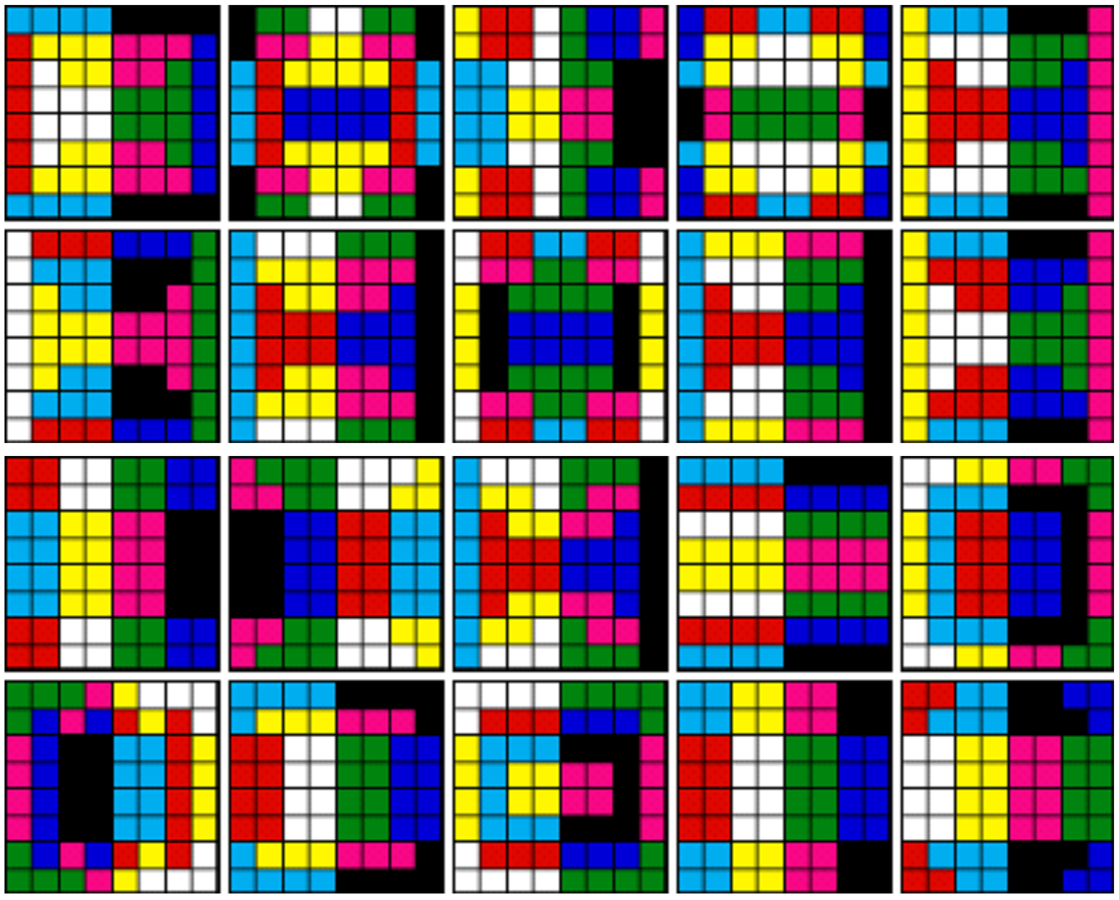


Fig.5-24. The best organisms obtained in each of the 20 runs of the comparison experiment: the top 2 rows have been obtained using the binary model, the bottom two with the three-state model.

To test the hypothesis that this three-state model actually has some advantages, I have repeated the maximum color experiment with both the binary model and the three-state model. I have run the steady state GA 10 times for each model, each time with a different random seed. This with a population size of 100 and this for 100 generations. For the steady state GA, that comes down to 10000 evaluations of genomes per run.

Crossover was set at 100%, mutation at 5 mutations per individual, and the regulatory networks evolved had 20 nodes with two inputs each. To be able to compare the range of organisms evolved for this particular performance function, the best organisms of each run are shown in Fig.5-24. It is not obvious from that picture, however, which of the two models is actually performing better. Indeed, the adult organisms are qualitatively the same. To learn which is the better performing model, we will have to resort to the statistics we can obtain from the experiment that indicate how fast the optimal adult organisms are found.

Some caution is required in interpreting the results of this experiment because of the following reasons: (1) the sample is small, only 10 runs each, (2) the three-state model has the additional advantage that no cycles can occur, and thus more genomes will effectively be evaluated, (3) the specific problem, maximizing the colors, might favor one or the other model.

Maximum performance		
	A	B
0	**22176**	22153
1	22153	22127
2	**22176**	**22176**
3	**22176**	22077
4	22153	22153
5	**22176**	22153
6	22153	22153
7	22153	22127
8	**22176**	22153
9	22153	22153
Avg.:	22165	22143
Max.:	5	1

Generation of max.		
	A	B
0	**21**	12
1	9	20
2	**40**	**38**
3	**2**	56
4	30	39
5	**43**	9
6	13	89
7	8	64
8	**27**	37
9	13	70
Avg.:	20.6	43.4

Nr. of Steps to Max.		
	A	B
0	**13**	8
1	5	6
2	**11**	**13**
3	**4**	8
4	6	8
5	**10**	7
6	3	11
7	9	10
8	**11**	12
9	6	17
Avg.:	7.8	10

Fig.5-25. Comparison of 10 runs with the three-state model (A) and pure RBN's (B). Left: Maximum performance obtained in one run (bold means optimal). Middle: generation in which this maximum was reached. Right: the number of intermediate solutions before the maximum was found.

Nevertheless, I would like to present the following statistics derived from the experiment: the three tables in Fig.5-25 show respectively the maximum performance obtained in each run, the generation in which the best organism was obtained, and the number of performance steps taken to get to the best organism. In addition, we know that the maximum score obtainable is 22176, as that represents an organism displaying all eight colors in equal proportions. The number of times that this maximum is found is also indicated at the bottom of the first table.

With the above caveat in mind, four trends are readily gleaned from these data:

(1) The average performance at the end of the run is higher for the three state model.

(2) The perfect organism is found only once using the RBN, but 5 times using the three-state model.

(3) The maximum performance is found earlier in the case of the three state genome.

(4) On average, using the three-state genome model the best organism is found using fewer jumps in performance.

Thus, the advantages we have summed up in the argument above seem to be confirmed by the experiment.

Chapter 6

Modeling Neural Development

In previous chapters we discussed the behavior of cells and the emerging phenomenon of development when cells coordinate their behavior. In this chapter, I will try to extend this model towards neural development. In this way the whole spectrum from unicellular to organismal behavior in these artificial organisms will have been explored. Parts of this material have been presented earlier in (Dellaert and Beer 1994).

To extend the model, we will first discuss the phenomena that are important in *neural development*, the process by which in nature the nervous system of an organism develops. This will give us a road map to extend the model. Because the random Boolean network genome does not lend itself easily towards implementing these functionalities, a modified model for the regulatory networks is proposed that will make it easier to implement different behaviors of the model cells, and that has some other interesting properties as well. This new model is then capitalized upon when we subsequently implement the neural functionalities to be discussed. One of the crucial elements of the extended model will be the simulation of axon growth, which will be implemented using a model reminiscent of a finite state machine. Finally, I present a handcoded autonomous agent that is capable of simple behavior in a simulated world, and I show that genetic algorithms can be used to incrementally improve upon its performance in the world.

Modeling Neural Development

In this section we focus our attention in turn on some of the phenomena that play a major role in (biological) neural development. No attempt is made to explain these aspects of development at length, but instead I would like to refer the reader to (Purves et al. 1985), whose order of presenting these topics I will loosely follow here. Here I intend to examine what aspects are important to incorporate in the model; how they are implemented will be addressed in detail in the next section.

Differentiation and Induction

The processes of differentiation and induction have been sufficiently covered in what came before. Suffice to say here that induction plays a large role in neural development: one of the most important events in vertebrate development, for example, is the process of neurulation, as shown earlier in Fig.2-9. However, most of the work described in this chapter is inspired by simpler animals with simple nervous systems. It is of importance to note that in these animals, induction plays a lesser role with respect to simple lineage events like asymmetric division of cells. This will be discussed in detail below.

Migration of Neural Precursors

In many animals (particularly in vertebrates) cell movement, i.e. cells that move from one place to another inside the embryo, plays a large role in neural development. For example, in the development of our own nervous system, a special class of neural precursors (those that will give rise to the neurons of the peripheral nervous system) migrates from a specialized region associated with the spinal chord (the neural crest) to the regions of the body where they will take up their function in adult life. Migration plays a role in the formation of the brain also, where neurons migrate extensively during the formation of the different layers of the cortex.

However, it is reasonable not to include cell movement in a first attempt at modeling an analog to neural development. In invertebrates cell movement is not nearly as important, and in many simple organisms it does not occur at all. Invertebrates certainly have interesting nervous systems, much more complex than what is achieved with the simple model presented here. Since invertebrates don't seem to require cell movement, we can safely leave it out for our purposes. In addition, cell movement is complicated to model and would substantially increase the computational complexity of the simulation, thus making it less tractable (which is not what we want, as we still want to use the model in conjunction with genetic algorithms).

Axon Outgrowth

In contrast, the processes (axons and dendrites) that are sent out (and sometimes retracted) by the neurons during development are crucial to the proper development of the nervous system, even in the simplest animals. Thus, we will have to make sure that outgrowth and subsequent navigation of these neurites is adequately modeled. Indeed, what makes neurons so special is their ability to communicate with each other over long distances via these processes that somehow manage to innervate targets in a remarkably robust way. The outgrowing axons find their way across the extracellular matrix by means of the *growth cone*, a complicated structure at the tip of the axon. This growth cone makes its way towards the target cells across the embryo, as shown in Fig.6-1.

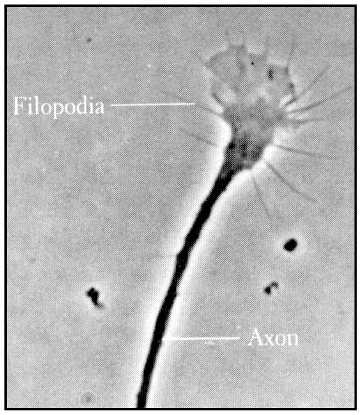


Fig.6-1. A growth cone at the tip of a growing axon. From (Walbot et al. 1987).

For axon outgrowth I propose a model based on the displaying of cellular adhesion molecules (CAMs) on the surface of the cells upon which the growth cone travels. In this proposed mechanism, the growing axon senses these CAMs and only grows where they are laid out, and moves in the direction where it can find more[10]. In (Purves et al. 1985) a number of other possible mechanisms are discussed; I chose a model inspired by the CAM mechanism because that was easily implemented using the model of multicellular development that already existed. Maybe this will limit the range of developmental patterns we will be able to develop, but on the other hand we can then get an idea of why these other mechanisms could be important, rather than ending up with a melting pot of diverse mechanisms that complicates understanding.

Taking inspiration form lower invertebrate nervous systems, I only consider axons that directly synapse on another cell's soma (the cell body), without modeling dendritic trees.

Cell Death

One way to get the numbers of neurons right is to grow many of them and then prune away those that are not needed, by letting some cells undergo programmed 'death'. This happens not only in the case of neurons, but in organisms like *C. elegans* it is an important mechanism of morphological change. In the model, I do not explicitly address cell death, but the phenomenon of a neuron being pruned away can be simulated by turning off the genetic elements that are

[10] This is a very simple rendering of the theory of differential adhesion, as found in the textbooks of neural development.

responsible for its neural character, i.e. sending an axon. Since in the model that is taken care of by one gene (see below) it effectively has the same result.

Trophic Dependencies

I did include the notion of a *trophic factor* in the model, as this seems to be a major mechanism whereby the fan-in, fan-out properties of neurons (and other cells innervated by them) are regulated. A trophic factor is a substance that neurons or axons need to survive or to stay connected, respectively. It is normally supplied by the target cells, i.e. the cells that neurons innervate by the axons they send out. For example, when a certain group of neurons is to be innervated by another group, the target neurons can provide a trophic factor to the innervating (incoming) axons of the projecting neurons. By modulating the amount of factor available, they can modulate the number of connections made to each neuron.

Formation of Synapses

It seems reasonable then to use the modeled trophic factor simultaneously as a signal for growing axons to stop wandering and form synapses at the location where this trophic factor is available. It may not be the case in biology that this is mediated by the same molecule, but I did not want to make the model needlessly complicated in the first modeling attempt. In any case, it can always be incorporated in subsequent models if necessary. The amount of ‘trophic factor’ available will also be used to specify the strength of the synaptic connection: one could draw the analogy with biology by imagining many synaptic boutons (places where synaptic contact is made) where there is a lot of trophic factor available, and conversely less where there is a lower concentration of trophic factor.

Nature of the Synaptic Interactions

I made the nature of the synaptic interactions dependent on two variables: the type of ‘neurotransmitter’ expressed in the pre-synaptic ‘neuron’, and the nature of the ‘receptors’ expressed in the post-synaptic cell. Both these variables can be under genetic control. The receptors should be selective to a certain neurotransmitter only, so that innervated cells can selectively tune in to signals provided by a certain group of neurons only.

The Cytoplasm Model

To extend the model towards neural development, a framework must be provided first to enable genetic elements - that were represented by the nodes of the random Boolean network in the basic model - to exert an influence within the cell. After having done so, some general cellular functionalities will be discussed that are also present in the multicellular model but that now have to be implemented in the new framework. Then I will discuss the implementation of the specific neuronal functionalities discussed in the previous section. After that, in the next section, we will continue with a model for a single growth cone.

Incorporating Functionality of Genetic Elements

Extending the model

The previous model did not allow for the state of a particular genetic element to have any consequences other than specifying the color of the cell. This simple color-model only necessitated interpreting certain bits in the state vector: the only 'active' role played by the genetic elements was (1) to regulate the state of other elements, and (2) to influence neighboring cells via the neighborhood vector. Thus, induction was implemented by the fact that one genetic element could take input from a bit in the neighborhood state vector. This is like a cell-surface receptor in biology, only that was never made explicit.

Here, a framework will be provided to make 'cellular function' possible, e.g. sending out an axon or releasing a chemical agent. To that end, I have implemented the random Boolean networks using a new genome-cytoplasm model, cytoplasm model in short. As the name suggests, there are two components to this model: the genome and the cytoplasm. I will discuss each of these here in somewhat more detail.

Cytoplasm

The *cytoplasm* is a *set* of *gene products*[11] that is produced by the expression of some *genes* in the genome (see below). In the model a gene product is simulated by an integer number. The set is not ordered, and only supports the operations ADD, REMOVE and CONTAINS. This model 'cytoplasm' is only loosely inspired by the cytoplasm of a real cell: in fact, about the only thing they have in common is that they can both contain 'gene products'; however, in a cell the cytoplasm is infinitely more complex, being a highly organized and dynamical system far removed from being a static container.

[11] In the remainder of this chapter I will use, for convenience, the terms *gene*, *operon*, *gene product*, *cytoplasm* and *genome* to denote components of the model, not the biological entities.

The gene products will be things like ‘cell surface receptor’ or ‘axon provider’, that execute active functions inside the cell. A single gene product will for instance be responsible for sending out an axon from a cell. Other products can be released in the cytoplasm when the cell is at the perimeter of the artificial organism, or when a ‘receptor’ senses the right gene product on the ‘surface’ of another cell. All these functionalities and more will be discussed in detail below. Again, the ‘gene products’ of the model and gene products in biology are quite different: biological gene products can range from RNA molecules to enzymes and highly modified proteins that perform incredible feats within the cell. However, as in biology, the ‘gene products’ of the model will be responsible for the actions that have to be taken by the cell, whereas the genome will have no direct effect on the cell’s internal or external environment.

Genome

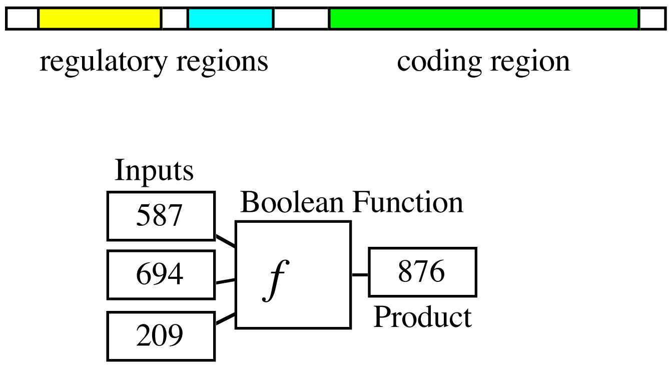


Fig.6-2. The ‘operon’ compared with (the schematic representation of) a biological operon.

The simulated genome consists of a set of *operons* representing the *genes* of the organism, mimicking the operation of a biological operon on an abstract level. Every operon specifies (1) one or more gene products that will be produced when the operon is active, (2) other gene products that can regulate its expression and (3) a Boolean function that specifies how the expression is regulated. In Fig.6-2 this is schematically depicted and compared to a biological operon. In the model, when the Boolean function is evaluated, a gene product will contribute a one as input to the Boolean function of another operon if it is present in the cytoplasm and zero otherwise.

Please note that a biological operon is more complicated than our very simple model that was inspired by it. In a biological operon, after the DNA has been transcribed into RNA, regions of RNA (introns) containing no genetic information are spliced out to yield the processed RNA, and only then the RNA is translated into proteins and enzymes that can function within the cell. In our simple model operon, there are no introns, there is no explicit transcription or translation, etc. In addition, in biology regulation of the expression of the genes in the operon is much more complicated.

Discussion

The picture produced by putting these two components together is shown in Fig.6-3. There are some obvious similarities but also some important differences with the straightforward random Boolean network implementation used earlier. A similarity is that one operon can in fact be seen as the description of one node in the random Boolean network. However, in the new implementation each operon can have a variable number of inputs, in contrast with the fixed K in the random Boolean networks. The number of gene products inserted in the cytoplasm when the Boolean function evaluates to true can also vary from operon to operon.

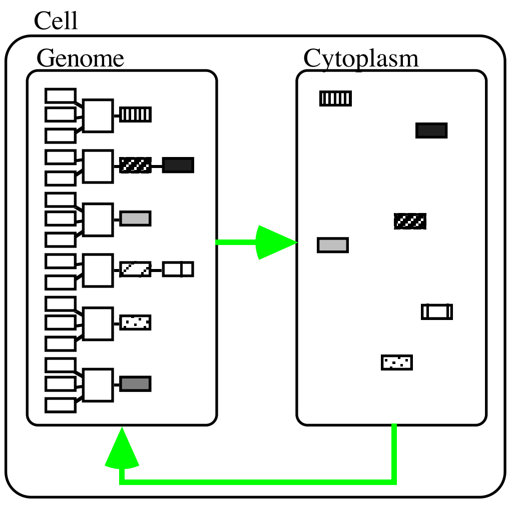


Fig.6-3. Schematic summary of the cytoplasm-genome model.

In addition, since the genome is a *set* of operons, the number of operons is not constant nor constrained: by ‘deletions’ and ‘insertions’ modeled after those we find in biological evolution, the genome can shrink or grow. If the mutation operator (i.e. the operation that mutates the genome when it is evolved using genetic algorithms) is accordingly modified to take advantage of this variable genome length, the acquired flexibility might be beneficial for evolving artificial organisms. For example, it is easier to evolve regulatory circuits when many operons are present as the number of circuits formed is simply greater, even in a random genome. It takes more time to simulate a big number of operons, however. If after initial pruning of non-favorable circuits selection pressure is put on the length (number of operons) of the genome, it will shrink and execution speed will improve. It is interesting that bacteria actually do something similar: when they are put under evolutionary pressure, they duplicate certain genes a large number of times in order to yield a large variation in genes, and when this has proven successful, the length of their DNA plasmids falls back to normal.

The cell-cycle

In the basic model, a simple simulated cell cycle is present, consisting of an interphase where the random Boolean network is evaluated, and a mitosis phase where the cell goes through division (or not). In the cytoplasm model, a new phase - the *functional phase* - is inserted to allow for the gene products to execute their function. This takes place before the mitosis phase is entered, and in it the cytoplasm is examined whether gene-products with an active function are present. If found, these functions are executed by the simulation program.

If you will, interphase now consists of a regulatory phase and a functional phase. In the regulatory phase, every time the operon’s Boolean function evaluates to true, its gene products are added to the cytoplasm, and, conversely, they are removed from it if the function evaluates to false.

Non-neural Functionalities

divide,	
assymgene, assym,	
massymgene, *massym*,	
symmgene, *symm*,	
a0,a1,a2,a3,a4,	// agents
r0,r1,r2,r3,r4,	// receptors
d0,d1,d2,d3,d4,	// detector
envsensor, e,	
midsensor, m,	

Table 1. The non-neural gene products used in the model.

Here a summary is given of gene products that implement basic developmental mechanisms (see Table 1), either extending the basic model or implementing mechanisms that were implicitly defined in the basic model but now made explicit by associating them with certain gene products. In the table, the words starting with ‘t’ are labels used to identify particular gene products, and they are taken directly from the implementation of the model. They will be discussed in some more detail in the following paragraphs.

Simple regulatory genes

Simple regulatory genes, not shown in Table 1, code for gene products that serve to regulate the expression of the other genes. They make up the vast majority of possible gene products.

Cell-cycle

A special gene product (*divide*) serves to signal whether a cell should go through mitosis or not, analogous to the 'dividing bit' we used in earlier experiments to evolve a 'cell cycle' in a unicellular context. By making use of this element, the genome could specify regions where great detail is needed in contrast with other regions where the cells can be coarser. This may play not such a large role in the square geometry that we work with right now, but might be an important regulator of form in future extensions of the model where more complex geometries could be used than the present one.

Asymmetric cell division

Events that influence the developmental lineage of the simulated cells used above were restricted to influences from outside the cell. In the previous chapter, it was discussed how a cell in the artificial organism can 'differentiate' into a certain cell type. A cell-type corresponded to a fixed state attractor in the phase portrait of the random Boolean networks that were used as a model for genetic regulatory networks. However, in that model, the cells could only be coerced into following a certain developmental pathway by either taking clues from the outside the cell (midline and external environment) or from other cells (by the mechanism of induction).

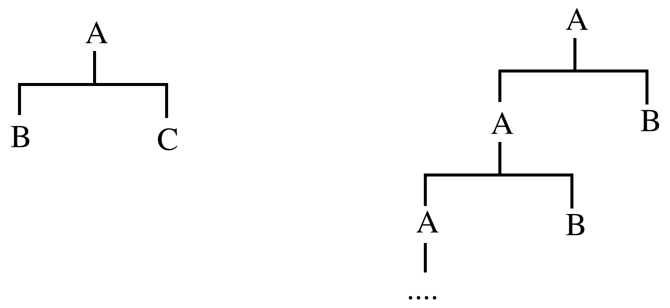


Fig.6-4. Asymmetric division and stem cell pattern (see text).

In biological development, especially in simple invertebrates like the leech or the nematode *C. elegans*, other 'lineage events' play a large role as well. For example, cells can divide asymmetrically, resulting in lineages as shown in Fig.6-4. At the left side of the figure, cell type A divides asymmetrically to yield two different cell types, B and C. At the right side, one of the cell types is again A, resulting in a typical stem cell pattern: here, the A cell type is preserved and serves as a continual supply of cells of type B. Such stem cells are very common in biological development, as well as in adult life. For example, in our bone marrow, hemapoetic stem cells continually produce new blood cells.

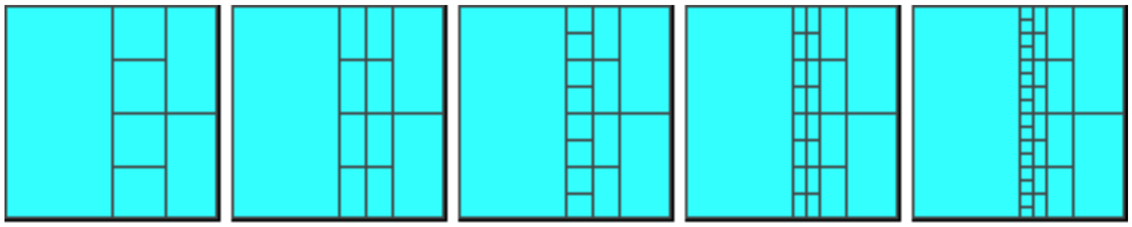


Fig.6-5. An artificial organism displaying a stem cell pattern.

It is possible to realize a stem cell pattern with an induction-like mechanism: for instance, a cell that lies on the midline and senses that can be like a stem cell, if the cells that it buds off will take a different developmental pathway, since they are not adjacent to the midline anymore. A biological example are the spermatogonia cells of Fig.2-1, that bud off from sperm-cell precursors. The inductive signal is given here by the basement membrane. Fig.6-5 shows an example of an artificial organism produced by the multicellular model where the posterior side of the body provides an inductive signal, so that cells at the border divide all the time, whereas the cells they bud off don't, resulting in a characteristic 'refinement' pattern.

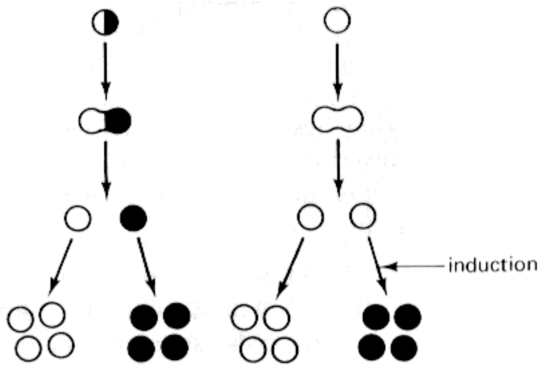


Fig.6-6. Left: intrinsic cell lineage specification. Right: Specification by induction. From (Brown, Hopkins and Keynes 1991).

However, in the invertebrate organisms mentioned above, induction usually plays a much less prominent role, and many of these lineage events are intrinsic to the cell, i.e. they will occur without any signal from outside the cell. The difference between the two mechanisms is illustrated in Fig.6-6. Thus, taking inspiration from that, when we would implement artificial 'genes' that can specify such events, we possibly make evolution of complex lineages easier. Instead of relying on these signal-induced asymmetric divisions to evolve[12], they could be coded on the genome directly.

In the model, a particular gene product (*assymgene*) can set up an asymmetric cell division: this is mimicked by letting a secondary gene product (*assym*) be distributed to only one daughter cell at the time of division, when *assymgene* is present in the cytoplasm prior to division. It is this kind of event that will lie at the basis of much of the cell-lineage events inspired by *C. elegans* development. However, this mechanism can only specify a stem cell pattern: indeed, the cell that does not have *assym* will have the same state as the mother cell, thus the typical stem cell configuration. To get the event depicted in the left panel of Fig.6-4, we need to make *both* daughter cells different from the mother cell: that is why the gene product *symmgene* is implemented, which will distribute a new component (*symm*) to both daughter cells after division.

Finally, one special gene product (*massymgene*, with *massym* as its secondary product) is reserved for a maternal effect event[13], i.e. the initial asymmetric distribution of some factor at the time of the first cleavage. To accomplish this, *massymgene* is injected automatically into the cytoplasm of the ‘zygote’ square.

Receptors

Some genes code for a ‘receptor’ (*rX*): these will look for a specific gene product (an ‘agent’ gene product) in the cytoplasm of neighboring cells (*aX*). When that is detected, a ‘detector’ product (*dX*) is inserted in the cytoplasm (the simulation program emulates a function similar to that of a cell surface receptor in biology, where the detector elements are analogous to second messengers), that can then regulate the expression of other genes. Note that this was done in the basic model by letting random Boolean networks influence each other across cell boundaries.

---

[12] In fact, it is entirely possible that some stem-cell patterns that rely on induction have evolved from intrinsic stem-cell patterns, where the role of the intrinsic signal is taken over by signals from the outside for some reason (for example greater control over position and timing).

[13] What is meant is just the symmetry breaking after the first cleavage event. It is called a ’maternal’ event in analogy with biology, where such asymmetries are often set up by the egg-producing machinery of the mother.

Sensing external stimuli

Very similar to receptors are the environment and midline sensor genes (*envsensor* and *midsensor*). When these are expressed, they will insert detector-products (*e* and *m*, respectively) into the cytoplasm to signal whether the cell lies at the perimeter of the organism or on the midline, respectively.

Implementation of Neural Functionalities

The neural developmental events depend on the coordinated expression of a number of gene products (see Table.2), which we will discuss now.

c0,*c1*,*c2*,*c3*,*c4*,	// CAMs
axon,	// Axon outgrowth
trophic,	// Trophic factor
n0,*n1*,*n2*,*n3*,*n4*,	// 'Neurotransmitter'
i0,*i1*,*i2*,*i3*,*i4*,	// inh. post-synaptic receptor
e0,*e1*,*e2*,*e3*,*e4*	// exc. post-synaptic receptor

Table.2. The neural gene products used in the model.

CAMs

Specific gene products represent the cellular adhesion molecules (CAM's) that will serve as the guidance for growing axons. They have no active function, but they just sit there to be sensed by the growth cone of the axon (see below). You can think of them as being displayed on the surface of the cells.

Axonal processes

A special gene product (*axon*) will be responsible for sending out a simulated axon into the environment of the cell. Whenever this gene is expressed, the cell will be checked to see whether it expresses a certain CAM. If it does, an axon will be sent out that is looking for CAM's of the same type. The detailed workings of the axonal growth cones (where all the action is) are discussed below.

Of course, in biology, there are many genes that cooperate to do this, not just one gene. For this simple model, however, we will not concern ourselves with the detailed execution of a task, just with the fact that it is being executed.

Trophic factor

The targets of neuronal processes express the gene coding for a trophic factor: this gene builds up an amount of factor available, that increases as long as the gene is being expressed. When a growth cone reaches a target, it will make synaptic contact and take some of the trophic factor away. The amount of trophic factor available and the number of axons trying to reach this axon can be significant factors in determining the type of innervation of a certain region. Also, the weight of the synaptic contact can be modulated by the amount of trophic factor available (see below).

Neurotransmitters and post-synaptic receptors

Three other kinds of genes will determine the nature of the synaptic contacts between neuron and innervated target:

- The neurotransmitter used must be expressed in the neuron and determines the selectivity of the neuron: only targets expressing the appropriate receptors will be able to respond to the neuron.

For each neurotransmitter there are corresponding receptors that can be expressed:

- Inhibitory receptors
- Excitatory receptors

It is conceivable to let these different choices that each cell can make determine the dynamical characteristics of the synaptic contact: one kind of receptor could have a fast response to released neurotransmitter, while others would respond slower, although this was not implemented in the model.

The Growth Cone Model

Central to the neural developmental model is how an axon sprouts from a simulated neuron, finds its way towards the target and innervates the target. This is implemented in the growth cone model.

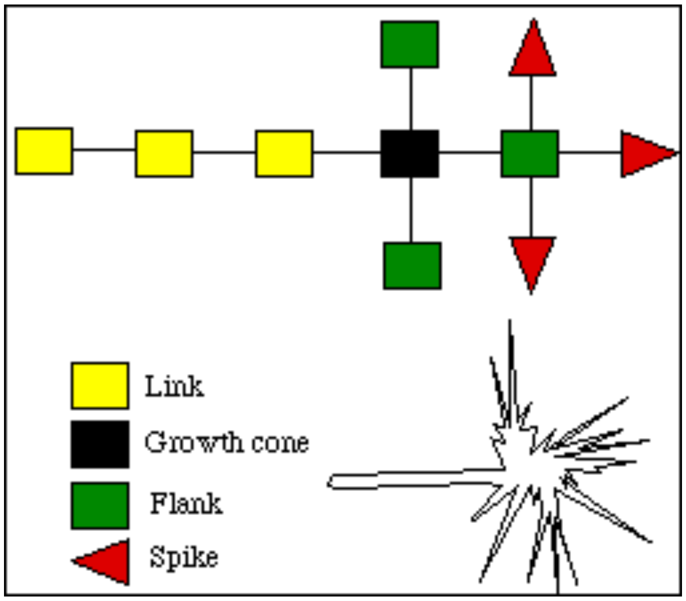


Fig.6-7. The different axon-element states.

The growth cone is the central component of the axon growth model. To be precise, the whole axon is modeled by a linked chain of ‘axon-elements’, each of which can be in one of four states. The different states are illustrated in Fig.6-7 and are summarized here:

- link : serves merely as a link between the neuron and the active growth ones.
- growth cone: this is the element that directs the active search through the organism.
- flank: every growth cone has several flanks, one in every possible direction (except backwards).
- spike: this is a process sent out by a flank, that can sense the presence of a CAM or a trophic factor.

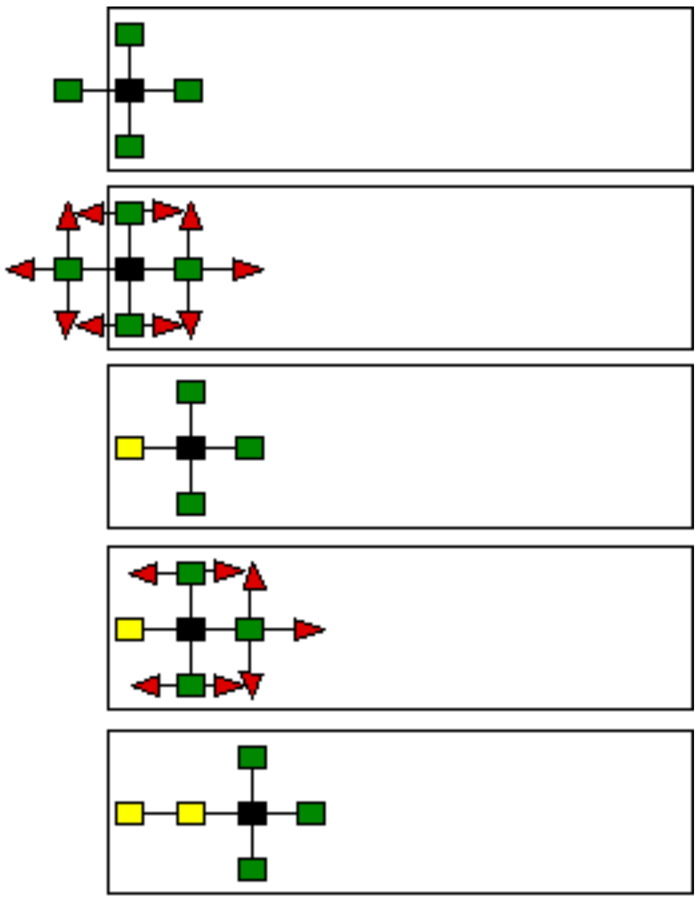


Fig.6-8. The consecutive steps of axon outgrowth. The rectangular panes represent a lane of CAM molecules laid out over the neighboring cells.

The algorithm for axon outgrowth is as follows (see Fig.6-8):

- The 'neuron' cell sprouts one growth cone, sending out flanks in every direction,

whereafter for each growth cone, the following steps are executed at each step:

- All flanks send out a number of spikes, again in every possible direction. The spikes that do not encounter a CAM molecule of the right type are pulled back.
- The flanks with the most spikes remaining will become the new growth cones, with axon branching if there is a tie. The original growth cone now becomes a link element.
- The new growth cones send out flanks in every possible direction, except backwards.

These three steps are executed until a trophic factor (*trophic*) is encountered: the growth cone stops looking for CAM's and instead synaptic contact will be made with the cells expressing this factor. It will still send out flanks and spikes to try to find more cells in the neighborhood expressing the trophic factor. The amount of trophic factor in the target cells is decreased for each increase in synaptic weight, and this will take place once per cell-cycle until the trophic factor is exhausted. Thus, the time that the factor had been expressed in the cell will determine the strength of the connection. Note that several neurons can conceivably innervate the same target and thus compete for available factor. A nice touch for future extension would be to implement a 'winner-take-all' mechanism, where only the most successful axon will innervate the target at the end of the process, while other axons pull away. Note also that the switch from CAM-searching mode to trophic-factor-searching mode is a local event and other branches of the same axon might still be happily feeling their way on some CAM pattern (the same CAM!).

Development of a Simple Nervous System

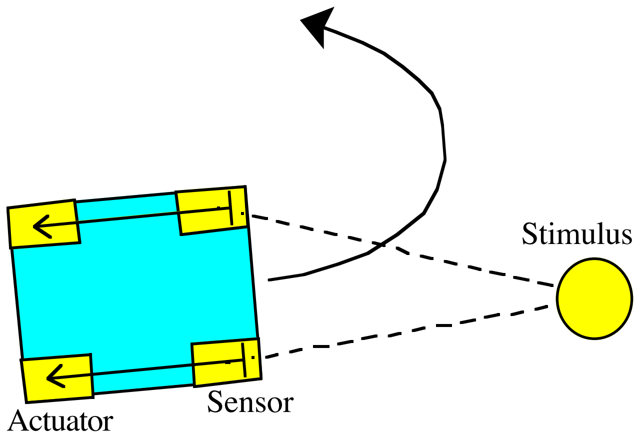


Fig.6-9. Schematic representation of an organism capable of avoiding a stimulus in a simulated environment.

To test whether this extended model can indeed be used to develop functional autonomous agents, I handcoded the genome of an artificial organism that is capable of executing a simple task. Here I describe that organism and the developmental sequence by which it originates, explaining how the genome constitutes a developmental specification for both body morphology and nervous system. We will also look at how this organism behaves in a simulated environment.

A Simple Braitenberg Hate-Vehicle

The organism that we want to hardwire should be capable of moving in a simulated world, and avoid a patch of 'chemicals' that it is capable of detecting by simulated 'smell'. I defined sensory regions in the front of the organism and actuator regions in the back. The actuators propel the organism, whereas the cells in the sensory regions are capable of detecting the patches of chemicals present in the environment. The simplest way for a nervous system to endow the organism with the correct behavior is to connect left-sensors with left-actuators and right-sensors with right-actuators. This is illustrated in Fig.6-9.

Morphological Development

In what follows it is shown how the handcoded genome leads to the development of a number of particular gene expression domains, that will serve to guide the development of the nervous system. This will provide some insight into the ways that patterns of gene expression can be regulated in our model. In fact, you will see that much is done with simple logic, just using AND functions. Of course, what is shown is a handcoded genome, i.e. it is designed. This explains the logical structure of how the spatial patterns of gene expression are set up. Note also that, although the developmental process is discussed here in two sequential sections, during the simulation these two steps are actually executing in parallel.

Overview

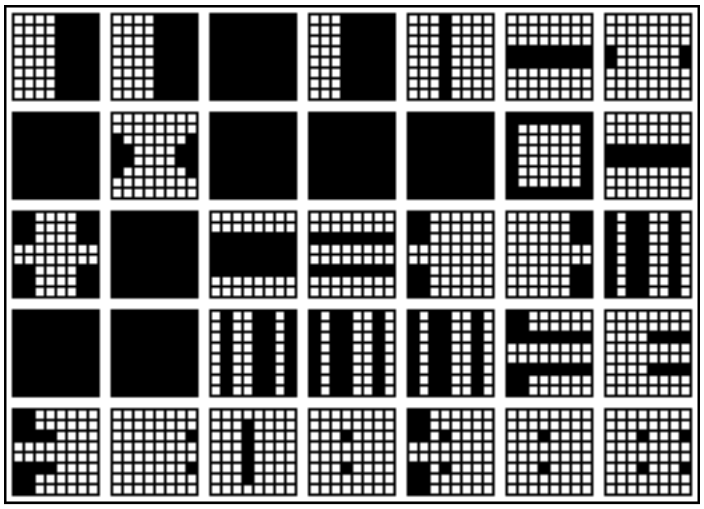


Fig.6-10. The expression of the relevant gene products in the adult organism. The order is taken from Table 3, starting top left, from left to right.

anterior	*a2*	*r2*	*d2*	*stripe*	*m*	*e*m*
*receptor e*m*	*next e*m*	*divide*	*envsensor*	*midsensor*	*e*	*m*
corner	*r0*	*d0*	*spinal*	*motor*	*eye*	*runt*
runt receptor	*detect runt*	*hairy*	*hairy recept.*	*detect hairy*	*wedge*	*c0*
c1	*sensory*	*stripey*	*inbetween*	*trophic*	*interneuron*	*axon*

Table 3. The names of the gene products in Fig.6-10.

In Fig.6-10 an overview is given of the expression patterns of all gene products relevant to the development of the handcoded organism. Do not attempt to understand them just yet, as we will explain these patterns in detail in the following paragraphs. However, the figure illustrates in an excellent manner that many different patterns of expression will be needed to specify our little 'nervous system'. Every square in the figure represents the body of the adult handcoded organism, 'stained' for the activity of one particular gene product. The legend to the figure is given in the form of Table 3. There are 35 gene products shown, among them the 28 gene products that are specified by the genome. The gene products *d0*, *d2*, *detect runt*, *m* and *e* are generated by the developmental process itself, as they are the detector elements (as discussed above). The names that the gene products are referred by are inspired by their function but have otherwise no importance: they are merely a convenient way to reference what would otherwise be numbers.

General gene products

```
divide = 1
envsensor = 1
midsensor = 1
```

First of all some general gene products need to be set up, as they are important for all other interactions. This is done by the portion of the genome shown in human-readable form above. Three gene products - *divide*, *envsensor* and *midsensor* - are set to be always present in the cytoplasm, by setting their Boolean function to TRUE. This ensures that (1) all the cells keep dividing; (2) all the cells have an environment sensor and (3) a midline sensor.

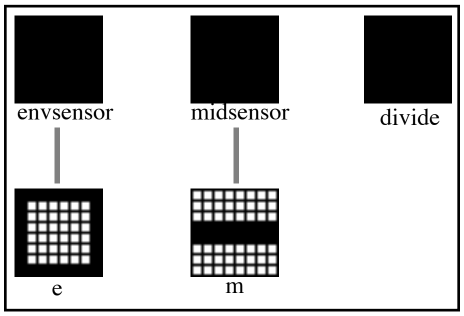


Fig.6-11. Some general gene products that are specified to be always on: *envsensor*, *midsensor* and *divide*. The gene products *e* and *m* are injected into the cytoplasm as a result of this. Such indirect relationships are indicated by the thick gray lines.

The activity patterns for these gene products are quite trivial, and are shown for the 64 cell stage in Fig.6-11. As you can see, *divide*, *midsensor* and *envsensor* are expressed in every cell. Also shown is the expression of *e* and *m*, the gene products that are injected into the cytoplasm by the action of the environment and midline sensors, respectively. The figure shows that *e* is only expressed in the cells bordering the external environment, and *m* is only expressed in cells next to the midline. The thick gray line between the *e* and *envsensor* patterns indicate that the gene product *e* is only injected by the simulation program on condition that *envsensor* is expressed. The same indirect dependency relation is valid for *m* and *midsensor*, and below for all *rX-aX* pairs.

stripe-spinal

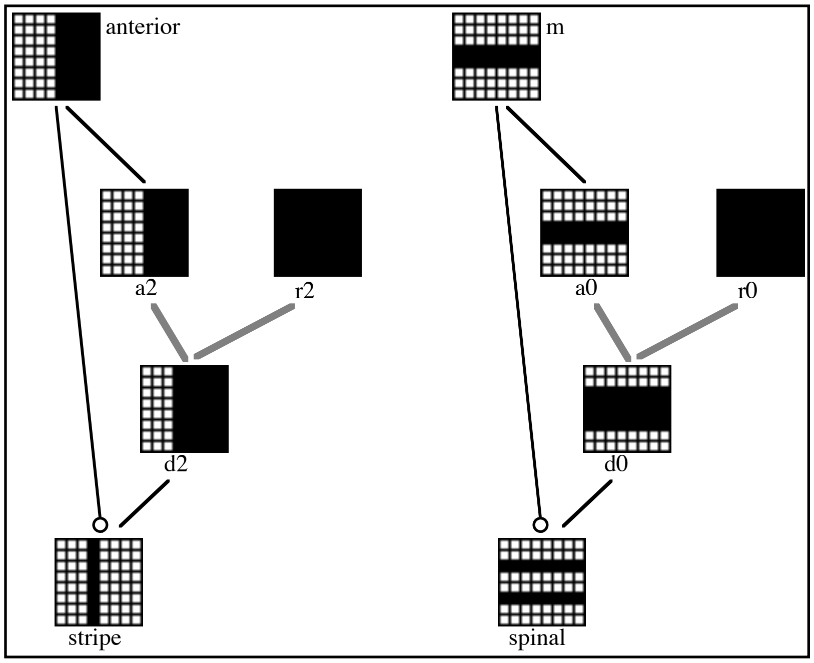


Fig.6-12. The interactions leading to the 2 topological gene products *stripe* and *spinal*.

```
a2 = anterior
r2 = 1
stripe = d2 but not anterior

a0 = m
r0 = 1
spinal = d0 but not m
```

Then two crucial gene products are set up that will serve later to specify the spatial position of other gene products, e.g. where the neurons are. This will happen again and again: most of the gene products perform a topological function. In Fig.6-12, the interactions leading to the expression of *stripe* are shown on the left side of the figure. Above the relevant portion of the genome is shown in an understandable notation: 3 lines govern the expression of stripe and 3 lines that of spinal. Here I will step through the genetic specification of *stripe*, explaining each line in detail and relating it back to the Fig.6-12, so that the specifications of the other expression domains can be understood from the figures alone.

```
a2 = anterior
```

(1) The first line[14] makes the expression of *a2* dependent on the presence of *anterior* [15] alone. By using the identity function the 'agent' element *a2* will be expressed only in the cells lying on the anterior end of the organism, because it is there that *anterior* is expressed. In the figure this is indicated by a line, connecting *anterior* to *a2*.

```
r2 = 1
```

(2) In the second line we specify that the appropriate receptor *r2* to detect this gene product *a2* should be present in all cells, by setting the Boolean function of this gene product to 1. This in turn will lead to the expression of *d2*, as shown in Fig.6-12, in all cells in or bordering the expression domain of *a2*, because of the implementation of the intercellular communication mechanism discussed above.

```
stripe = d2 but not anterior
```

(3) Then, in the third line, we make sure that *stripe* is expressed only in those cells were *d2* is expressed but *anterior* is not. In this way, the expression domain is the region bordering the anterior part but not the anterior part itself. In the figure, this expression domain is shown, along with the two lines indicating that *stripe* takes its clues from two other elements, *d2* and *anterior*. Note the open circle at the end of one of the lines indicating that the NOT operator is applied first.

Thus, we can define a nice expression domain for *stripe* just by defining the appropriate Boolean functions and inputs in the genome. In the same manner, *spinal* is the expression domain defined by cells that are adjacent to the midline cells, i.e. those that contain *m*, as shown on the right side of Fig.6-12. In the figure, the lines indicate the Boolean interactions that lead to the expression of certain gene products, denoting which element receives input from which others. In this way, the figures can also be viewed as representing 'regulatory circuits' as discussed in Chapter 3.

Two other points should be noted about the figures: (1) As explained before, the thick gray lines indicate indirect relationships: for example, *d2* is expressed in those cells *r2* AND bordering the *a2* domain. However, no operon codes this rule for *d2* as it is done implicitly by the simulation program. (2) If two lines converging on a square are to be ORed together, that will be explicitly stated, as AND is more common.

[14] Although I discuss this line first, note that there is no particular order to the lines, as the developmental model operates on the operons in parallel. The order in which they are treated here is merely for convenience of explanation.

[15] The gene product *anterior* will be present automatically, by the maternal asymmetry mechanism.

corner

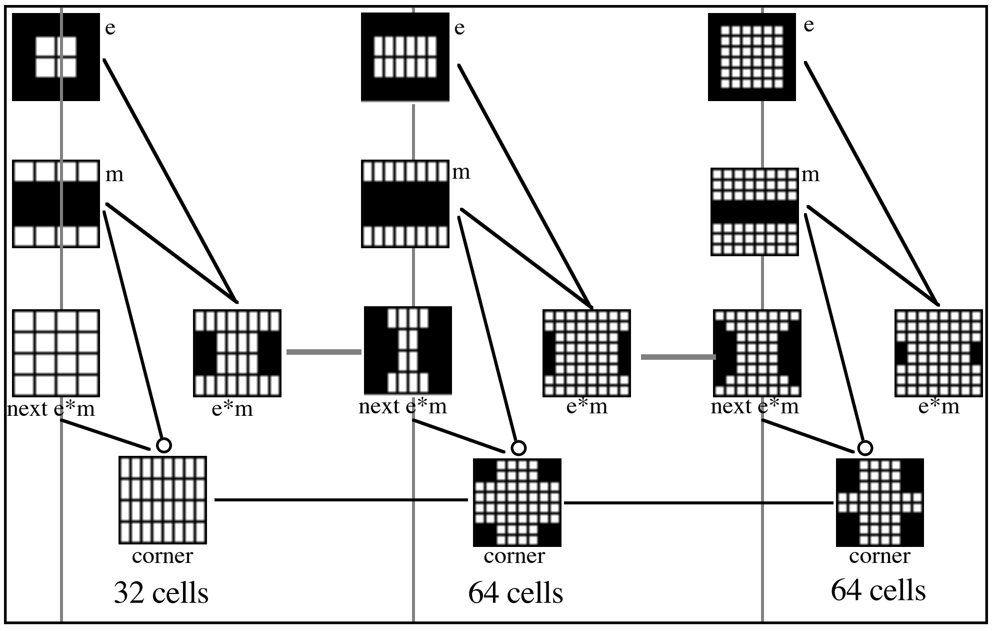


Fig.6-13. The interactions leading to the expression of the topological gene product *corner*.

```
a1 = e*m = e and m
r1 = 1
corner = (not m and next e*m) or corner
```

The interactions leading to the expression of the topological gene product *corner* are somewhat more complex (Fig.6-13), as they depend on early expression patterns of certain gene products (*e***m* and *m*) that change over time. Because of this, *corner* needs a self-reinforcement connection, so that its pattern remains stable even after the original cause for it disappears. This is shown in the genome code: the Boolean function for *corner* contains 'or *corner*', so that from the moment *corner* becomes active in a cell it will stay active. The *next e***m*, *e* and *m* patterns are drawn inbetween developmental stages, as they are formed in one stage but becomes available only in the next: they are detector elements, injected into the cytoplasm prior to mitosis, in the functional phase.

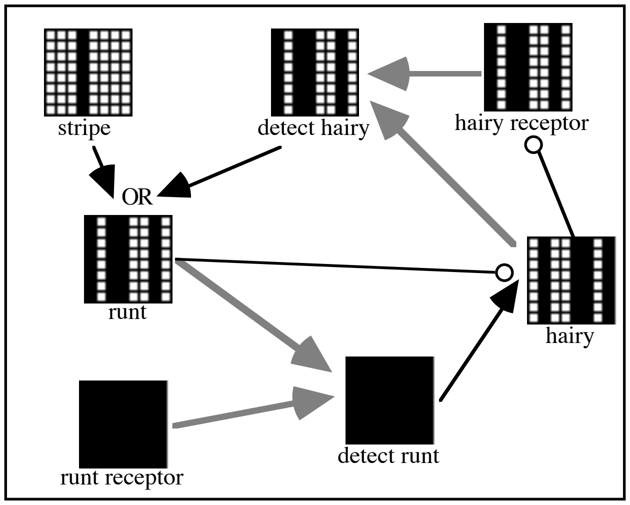


Fig.6-14. *stripe* seeds the interactions that will lead to a stable segmentation pattern formed by the *hairy* and *runt* gene products.

```
runt = stripe or hairy detect
runt receptor = 1

hairy = next runt but not runt
hairy receptor = not hairy
```

One of the most complex interactions will lead to the emergence of a stable segmentation pattern, formed by the two gene products *hairy* and *runt*. These gene products will set up a segmentation pattern by influencing each others expression, and they were inspired by the mechanisms by which such a pattern of segmentation originates in *Drosophila*[16]. In Fig.6-14 the situation of the 64 cell stage is shown. The process of pattern formation begins earlier, however: first, *runt* is seeded by *stripe*, and in successive interactions a striped pattern emerges. This is not shown in detail, but note that the two gene products inhibit each others expression, as indicated by the open circles in the figure. Again, indirect interactions as a result of the functional phase are indicated in thick gray lines, for example: *detect hairy* is injected into the cytoplasm of those cells neighboring *hairy* where a *hairy* receptor gene product is expressed.

[16] The inspiration is limited to the fact that there are sets of genes in early Drosophila development that inhibit each others expression and in that way help to set up a striped segmentation pattern, just like the elements *hairy* and *runt* will.

CAMs

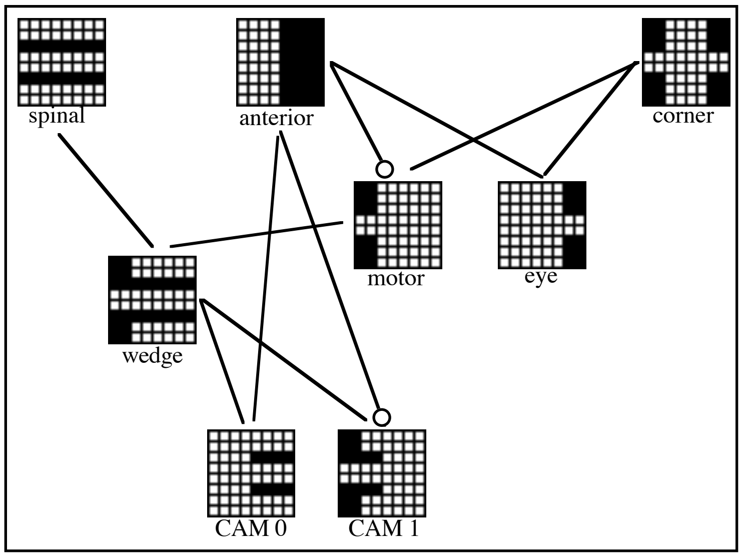


Fig.6-15. The gene products *spinal*, *anterior* and *corner* serve to set up the *wedge*, CAM (*c0* and *c1*) and *motor-eye* expression patterns.

```
motor = corner and not anterior
eye = corner and anterior
wedge : spinal or motor
c0 if wedge and anterior
c1 if wedge and posterior
```

In Fig.6-15 and the accompanying segment of genome code it is shown how the position of sensory and motor cells is determined. Also, an expression domain *wedge* is set up to provide a connecting lane from the *eye* region towards the *motor* region. Since the neural network that is coded for will have ‘interneurons’ along this pathway, this *wedge* pattern is used to specify two CAM patterns: one to lead the axons from the sensory neurons to the interneurons, and one from the interneurons towards the actuator region. These CAM patterns are indicated in the figure as *c0* and *c1* respectively.

axon

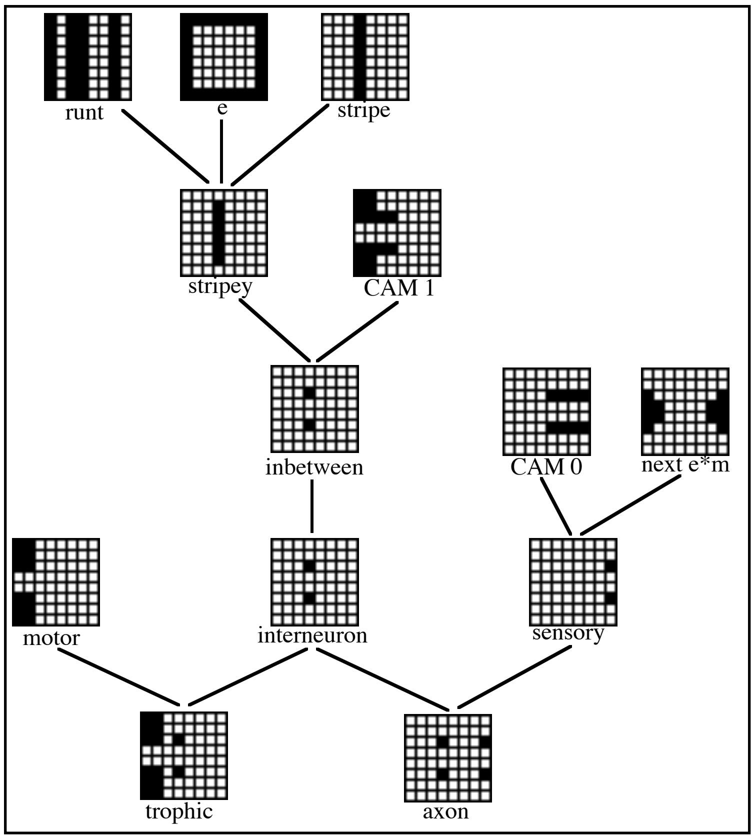


Fig.6-16. The interactions leading to specification of neurons and innervation targets.

```
sensory = c0 and detect e*m and e
stripey = runt and stripe and not e
inbetween = stripey and c1
trophic = motor or interneuron
interneuron = inbetween
axon = sensor or interneuron
```

Finally, the placement of the 'neurons' is specified: as you can see in Fig.6-16, first the topological helper gene products *stripey* and *inbetween* are set up, after which the neuron placement is attained by *sensory* and *interneuron*. Now we are in the position to say where the axons must sprout and where they should go to: *axon* is the gene product responsible for sending out an axon, and is expressed in *sensory* and *interneuron* cells. *trophic* is the gene product, as explained, that will serve to home in the axons, and it is expressed, in *interneuron* and in *motor*.

Summary

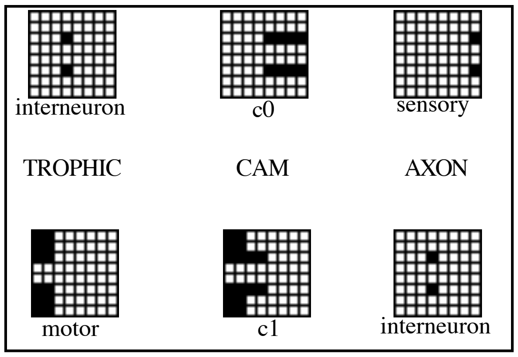


Fig.6-17. Summary of gene products specifying the axon pathways.

The patterns that are crucial for axon growth and guidance are summarized once again in Fig.6-17. At the anterior end of the artificial organism, the *sensory* neurons will send out axons towards the 'interneurons' over the cam lane *c0*. At the posterior end, the 'interneurons' send out an axon over *c1* towards the *motor* target area.

Neural Development

In the previous section you could see how the appropriate gene products that guide the outgrowth of axons are correctly set up by the developmental specification contained in the genome (essentially in the form of a random Boolean network, although a cytoplasm-genome implementation is used). In fact, that is the only thing that the genome is capable of setting up: topological domains of gene expression. The remainder of the developmental process is solely the result of the simulation of axon outgrowth by the simulation program[17].

[17] The developmental model, implemented by the simulation program, was also crucial for the correct topological expression of genes, as it implements the receptor-detector and asymmetric cell division mechanisms we discussed.

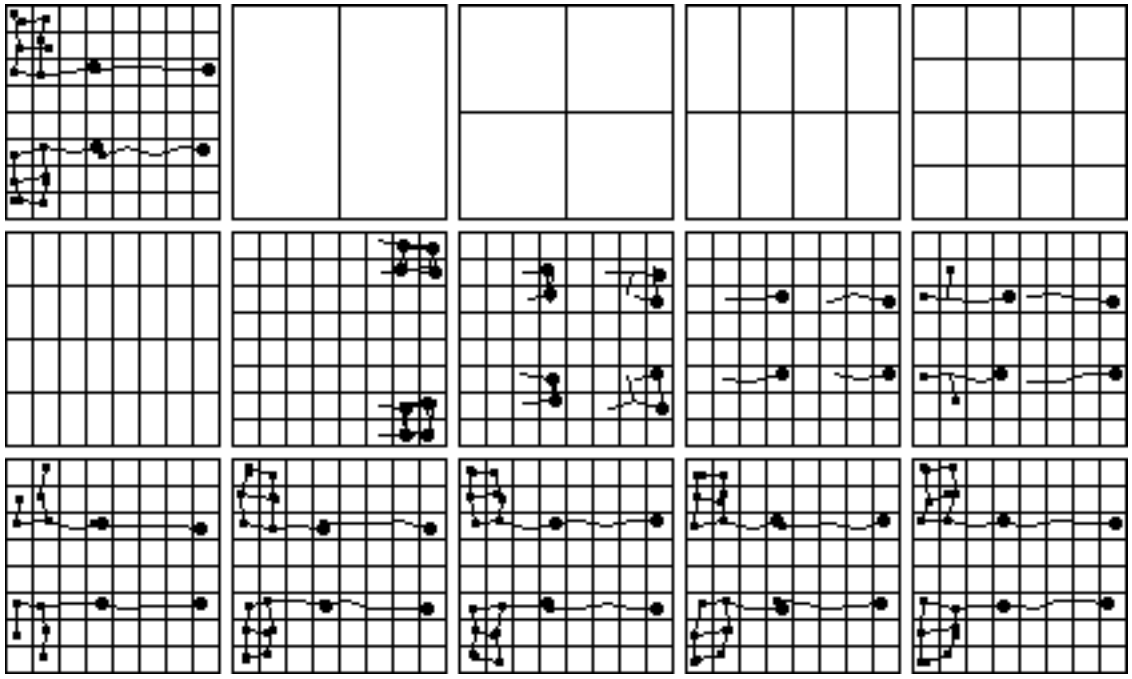


Fig.6-18. The consecutive steps in the development of the organism with the handcoded genome (see text for an explanation).

The consecutive steps in the neural development of the organism are shown in Fig.6-18. In it, the fully developed organism is shown on the upper left. The second square in the figure (to the right of the 'adult') is the two-cell stage, then the four-cell stage is shown etc. In this way the whole developmental sequence can be followed, in chronological order from left to right and from top to bottom. As you can see, the last square is identical to the 'adult' in the upper left corner. Cells that sprout axons have a larger black circle on them, and the innervation sites ('synapses') of the axons are represented by small black squares. The axon elements themselves are shown as line segments between the centers of the cells they reside on. However, to make it easier to recognize when two axon elements use the same pathway, a small random displacement is added to their starting and ending position so that they will not overlap completely.

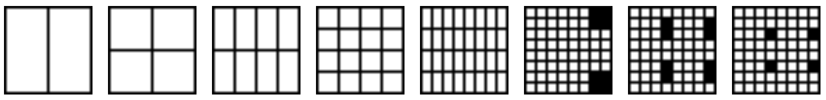


Fig.6-19. The expression of *axon* during consecutive developmental steps.

As discussed, two sensory cells at the anterior side (right side in the figures) of the organism will project to two interneurons in the middle of the body. These in turn innervate the actuator regions at each flank of the body on the posterior side (left side in the figures). The process of axon outgrowth is first seen in the 64 cell stage: there 8 cells have apparently started to express *axon*.

However, the *axon* expression domain has not yet reached its final form: in Fig.6-19 you can see that it only attains that after two more cell cycles. The cells that lose *axon* during this process, simply lose their axons.

When the simulated growth cones reach the target area, specified by *trophic*, they start looking for more trophic factor as opposed to CAM's. At the same time, connections are formed between the neuron and the cells underlying the growth cones that detect trophic factor, as indicated in Fig.6-18 by small black squares.

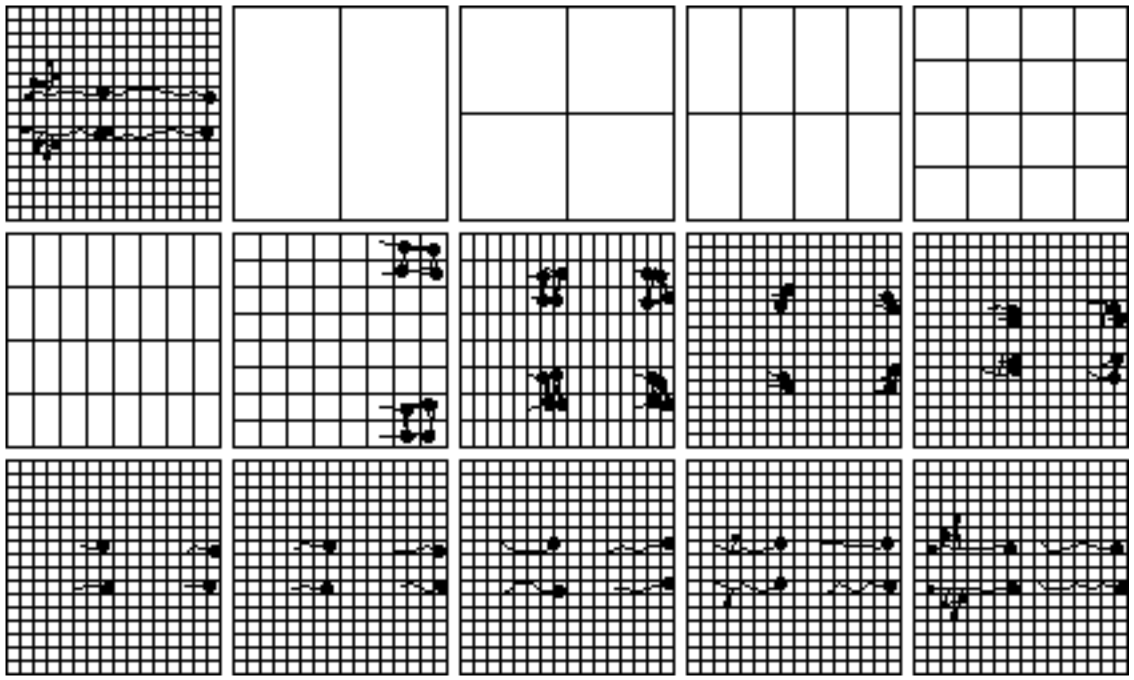


Fig.6-20. The developmental sequence directed by the same genome, but now with 8 divisions instead of 6. (Note that the sequence does not run to completion, but that the adult emerging can nevertheless be seen in the upper left corner).

Interestingly, if we let all the cells divide twice more to yield four times as many cells, the developmental process adapts and produces a qualitatively similar pattern. This indicates that using a developmental model does provide some robustness, even in the face of rather important changes in the underlying morphologies. Fig.6-18 and Fig.6-20 show the results for 6 and 8 divisions, respectively, yielding 64 and 256 cells in the final organism.

Behavior in a Simulated Environment

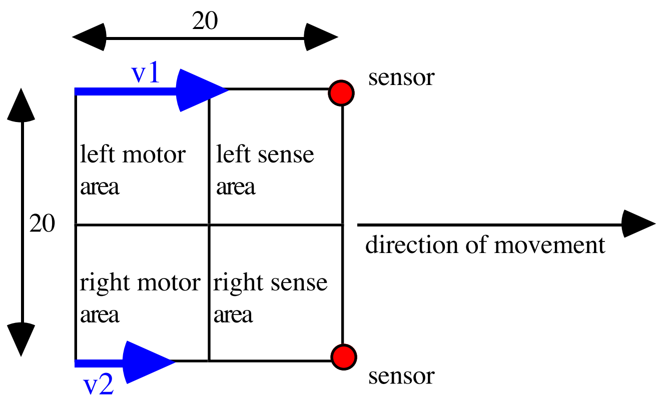


Fig.6-21. The body characteristics of the agent as measured in 'world units' (see text for a detailed explanation).

Finally a functional agent is extracted from the developed organism and put in a simulated world. We will first address the *body implementation* of the artificial organism. It has a body whose dimensions are 20 by 20 units. Two sensors are placed as indicated in Fig.6-21, i.e. to the front and at the extreme left and right side. These sensors will provide input to all neurons (see below for the discussion of the neural network) that are located in the appropriate 'sensory' region, also indicated in the figure. Conversely, the output of all the neurons located in the 'motor areas' is summed to yield a left and a right speed component that will cause the movement of the organism, as follows:

$$v1 = 5.0 \sum_{i \in left} out_i \qquad v2 = 5.0 \sum_{i \in rightt} out_i$$

$$v_{forward} = 10.0 \qquad v_{perpendicular} = (v1 - v2) / 2$$

v1 and v2 are the speed vectors caused by the motor neurons, and are indicated on Fig.6-21 by the thick arrows. There is no attempt at modeling wheels or any other detailed propulsion mechanism, but these speed vectors can be thought of as the result of spinning wheels, where the rotation of the wheels is proportional to the summed activity of the neurons in each motor area. These two speed vectors are then converted into relative speed vectors, v_{forward} and $v_{\text{perpendicular}}$, as shown. However, in these experiments v_{forward} is made to be constant: in this way, it is only the steering that is regulated by the neural network, not the velocity, which will be more convenient in the evolution experiment discussed below.

The *simulated environment* is modeled as an infinite plane, and the portion shown in Fig. 6-21bis (and 6-26) measures 2000 by 2000 'units', i.e. about 100 body lengths in both horizontal and vertical direction. A patch of ‘chemicals’ is positioned at coordinate (1000,1000), which will provide an input to each of the two sensors of the artificial organism according to the formula:

$$\text{Sensor activation } a = 0.1 \sum_{patches} Pe^{-(d/400)^2}$$

where d is the distance from the patch to the sensor, and P is the patch size. As can be deduced from the formula, each patches’ smell falls off exponentially with the squared distance from the patch to the sensor. The patch size used for the only patch present in this experiment was 50.

On top of the organism’s body, a *neural network* is constructed that will take input from the sensors and provide output to the motors, representing the nervous system of the organism. This is done by first checking which cells either send out axons themselves or are innervated by them. For each of these cells a neuron is then created and connected appropriately to the other neurons. The number of ‘synaptic contacts’ that each axon makes with a target cell is multiplied by 3.0 to get the weight from the neuron to the target cell.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	**6**	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**6**
8	0	0	**3**	**3**	**3**	**3**	**3**	**3**	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	**3**	**3**	**3**	**3**	**3**	**3**	0

Table 4. The weight matrix of the resulting neural network (see text for an explanation).

The resulting weight matrix that is formed for the handcoded organism is shown in Table 4: neurons labeled 0 and 1 are the cells associated with the sensory region, neurons 8 and 15 are the interneurons, and the target cells are numbered 2-7 and 9-14. The neurons are connected appropriately to the input from the sensors, and the output of selected neurons is directed towards the actuators of the simulated agent, as explained above. The thresholds and time constants in the extracted neural network are the same for all neurons[18], i.e. tau=1.0 and theta=2.7.

The resulting agent -complete with body and nervous system- was then deposited in the simulated environment and set on a course towards a patch of ‘chemicals’. Fig.6-21bis shows the results of the experiment, wherein the organism is put down to the left side of a large patch (that it will want

[18] In fact the values for threshold and time constant were optimized in the experiments described in the evolvability section, by the genetic algorithm. After these experiments, the optimal values found were used in the original organism as well.

to avoid) and set on a course due East. As you can see from the trace that is left behind, the organism does indeed steer away from the patch when its sensors pick up the smell.

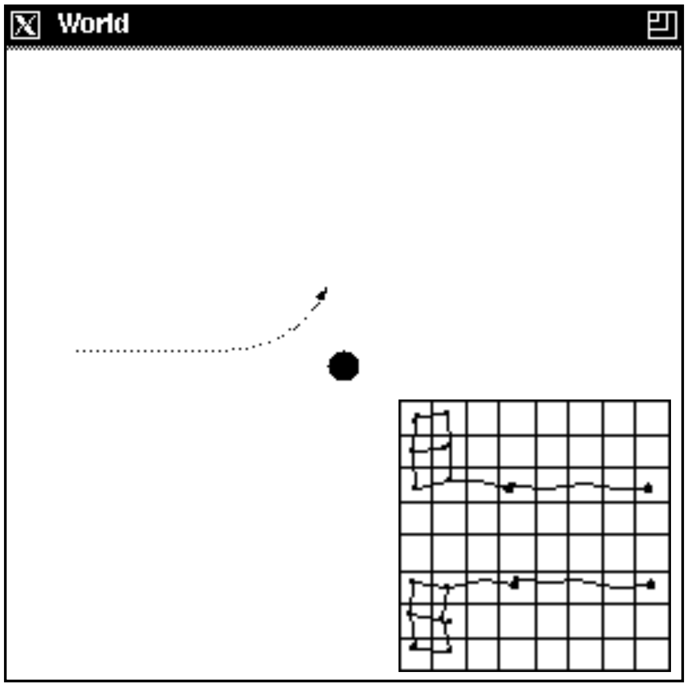


Fig.6-21bis. The behavior of the handcoded organism in response to a patch of chemicals. The portion of the world shown measures 2000 by 2000, with patch of chemicals placed in the middle. The organism is released on the left and leaves a trail as it moves towards the patch.

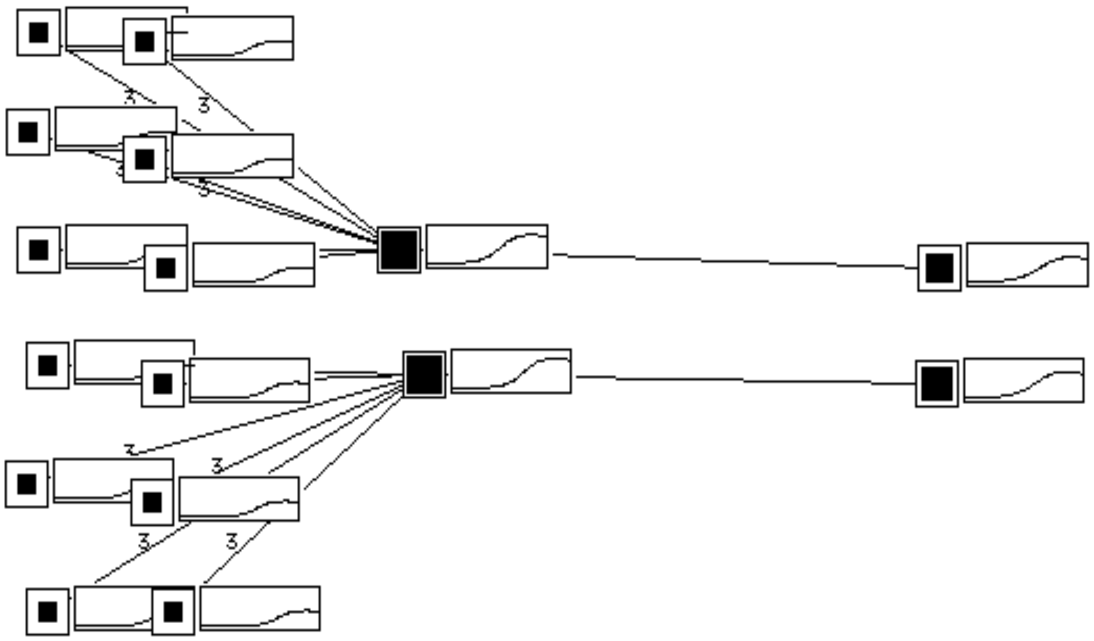


Fig.6-22. The 'intracellular recordings' of all the neurons of the handcoded organism during the experiment of Fig.6-exp1.

I used ‘software intracellular recordings’ to show the activities of the neurons while it was executing its behavior, as shown in Fig.6-22. The square elements represent the neurons[19], and next to each neuron its activity over time is displayed. As you can see, the activity in the sensor neurons is low at first, but as the organism nears the patch it rises quickly. Since the organism is slightly above the patch, however, the right sensor picks up more activity, and this difference is transmitted to the motor commands, causing the organism to veer to its left (i.e. towards the top of Fig.6-21bis). Note that most of the change in activity is caused *by the movement of the agent*, not by the dynamical behavior of the network itself.

Evolvability of the Extended Model

The next interesting question is whether we can now use the genetic algorithm to improve on the behavior of the handcoded organism, i.e. its evolvability. As already stressed, note that mutations and crossover will take place on the level of the genome, but that we are looking to optimize a performance function in the fully developed organism, which is not a trivial problem.

To this end we need an encoding for an organism, a way to mutate this encoding and a performance function by which the organism is evaluated. The encoding is somewhat different than the one used for pure random Boolean networks of Fig.4-26, as we now have to work with the cytoplasm model. To be precise, each operon is encoded on the genome as follows:

```
<2> 210 MAsymm  // input products
<1> 210         // output product
<4> 0 1 1 1     // Boolean function
```

In this example there are two input products, 210 and MAsymm, there is one output product, 210, and the Boolean function is OR. As you can see, the encoding is not that different from the one used in Chapter 4, although note that (1) the number of inputs can differ for each operon, (2) the number of output products can be greater than one and can also be different for each operon, and (3) the number of operons itself can vary.

Mutation is implemented in a similar way as with the pure random Boolean networks: either one of the gene products (in or output) is mutated by substituting it with a random gene product, or the Boolean function is mutated by flipping one bit in the lookup table. Note that in this way there is no correlation between gene products before and after mutation. When we would use binary

[19] The size of the filled square in each neuron represents the activity of the neuron, but since only the last frame of the dynamical display is shown, it represents the activity of the neurons in the last time step of the experiment.

encoding to encode the gene products, there would be transitions that are more probable than others (those transitions where the Hamming distance is small will occur more often).

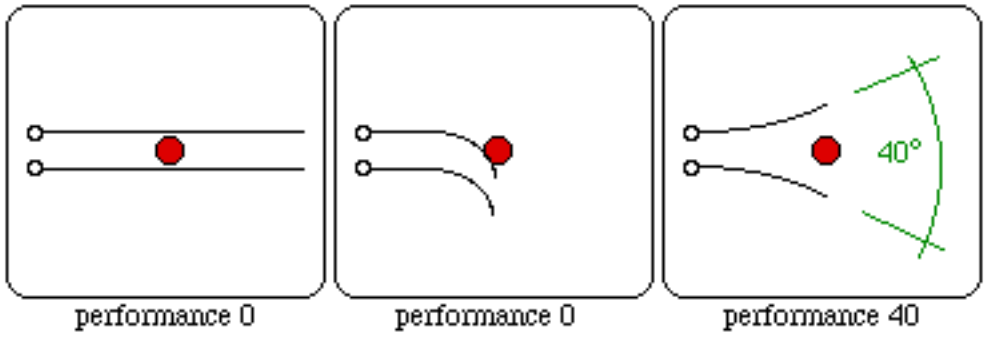


Fig.6-23. The performance function is calculated by measuring the angle between the final orientation vectors of the organism at the end of the double experiment.

The performance function used to test the evolvability was very simple: the organism is dropped in the world on two different starting points, both to the left of the patch, and both times facing East. However, in one case the organism is placed slightly higher than the patch, in the other case slightly lower, as illustrated in Fig.6-23. Thus, if it is a good organism, it will turn towards the top the first time and towards the bottom the second time. The performance is simply the angle between the final orientation of the organism in both experiments.

The handcoded organism was used to seed the starting population for the GA and I found that indeed, a better performing organism emerged, where the actuator regions were innervated in a different and stronger way, i.e. more ‘motor area’ cells were innervated by the interneurons. Because of this, the difference in input between the two sensory neurons was amplified more strongly and the agent turns away faster.

In this particular run of the GA I used a population size of 100 and the organism shown emerged after only 22 generations. Although I have not analyzed the detailed workings of the genome as with the handcoded genome, it is of interest to say that the newly evolved genome differed from the handcoded one on nine occasions, i.e. in the course of the evolutionary process nine mutations took place. Six of those were bit flips in the Boolean functions and three of them were changes in connectivity as a result of an input change.

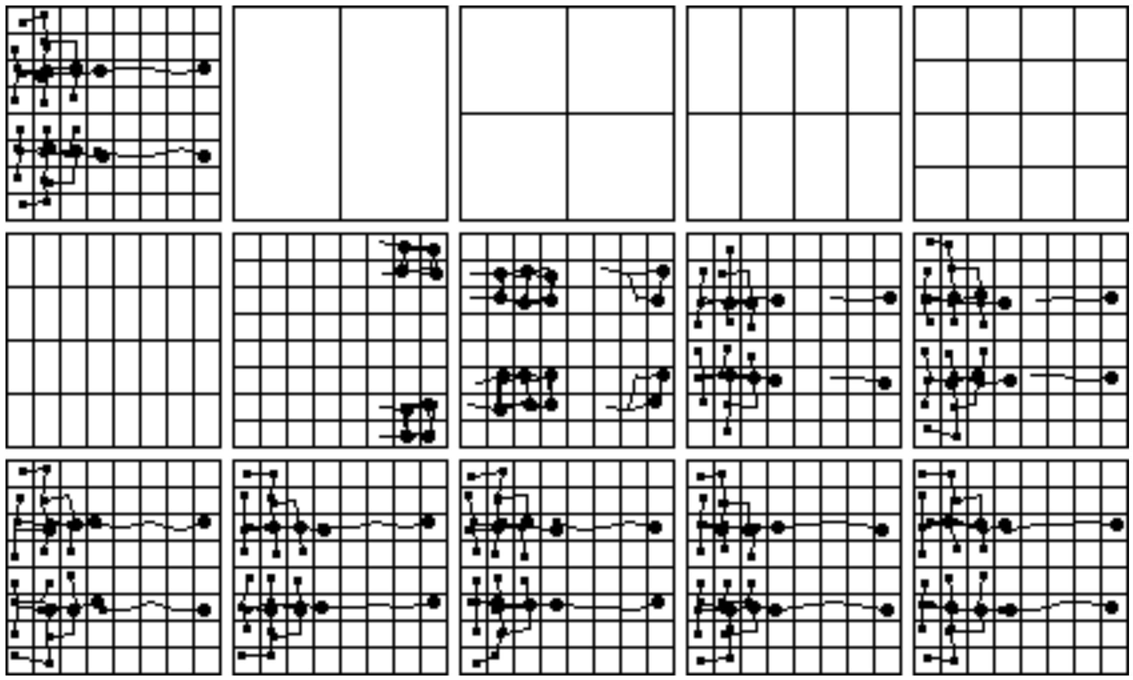


Fig.6-24. The developmental sequence of the newly evolved organism.

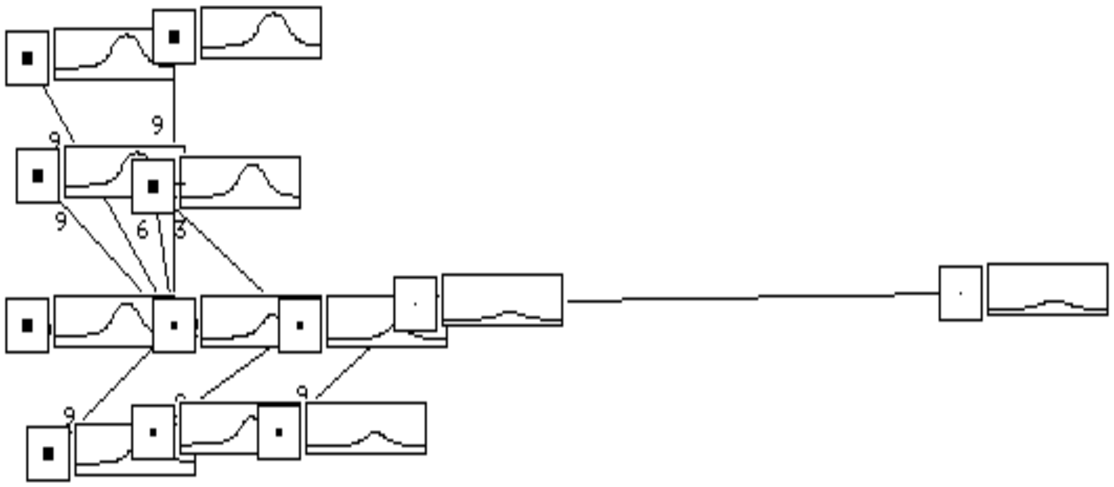


Fig.6-25. The ‘intracellular recordings’ of the topmost neurons of the newly evolved organism during the experiment.

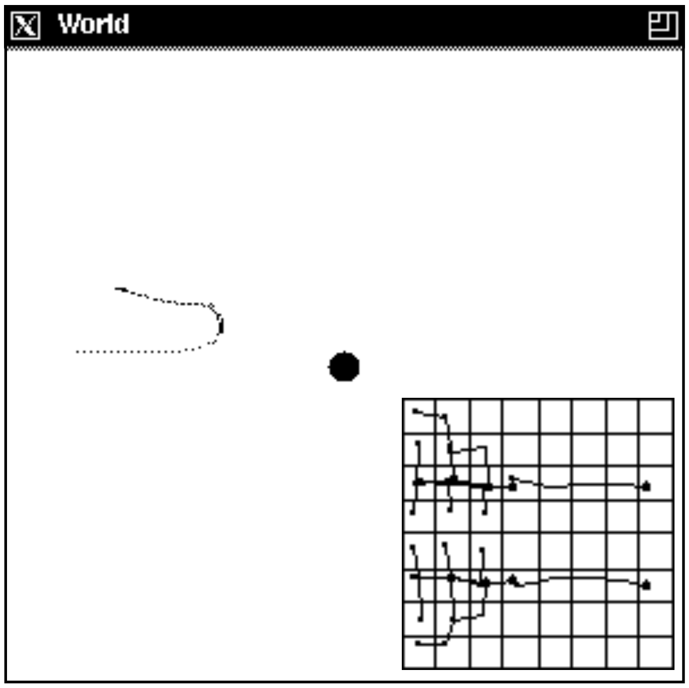


Fig.6-26. The markedly better behavior of the organism that was evolved from the original handcoded organism.

The developmental sequence of this newly evolved organism is shown in Fig.6-24, the intracellular recordings of its nervous system during the same experiment as before are shown in Fig.6-25. Note that the activity of the neurons over time is strongly correlated with the distance of the agent from the patch, and that this explains the more pronounced rise and fall of the activity in the neurons of Fig.6-25 as opposed to Fig.6-22. Because the agent turns away faster, the smell of the patch also weakens faster. Finally the behavior of the evolved organism in the simulated environment is shown in Fig.6-26.

In conclusion, we can state that at least incremental evolution is possible. This is of course not as strong a result as was hoped: ultimately, it must be possible to evolve organisms from scratch, something which we have not been able to do. In the concluding chapter, we will discuss this point more extensively.

Pattern Formation Mechanisms

As the last topic in this chapter, this section describes the development of a second organism with a handcoded genome, to illustrate the pattern formation mechanisms that can be attained using the model. Although no neural development is shown, it will nevertheless be discussed here for the following reasons: (1) it is implemented based on the cytoplasm model discussed at the beginning of this chapter; (2) the inspiration for the developmental sequence was taken from the formation of eye segment precursors in the development of the *Drosophila* eye, thus of interest to neural development; (3) the previous explanation of the mechanisms underlying the formation of particular expression domains in the handcoded Braitenberg organism provides a good starting point and background for explaining what happens in this case.

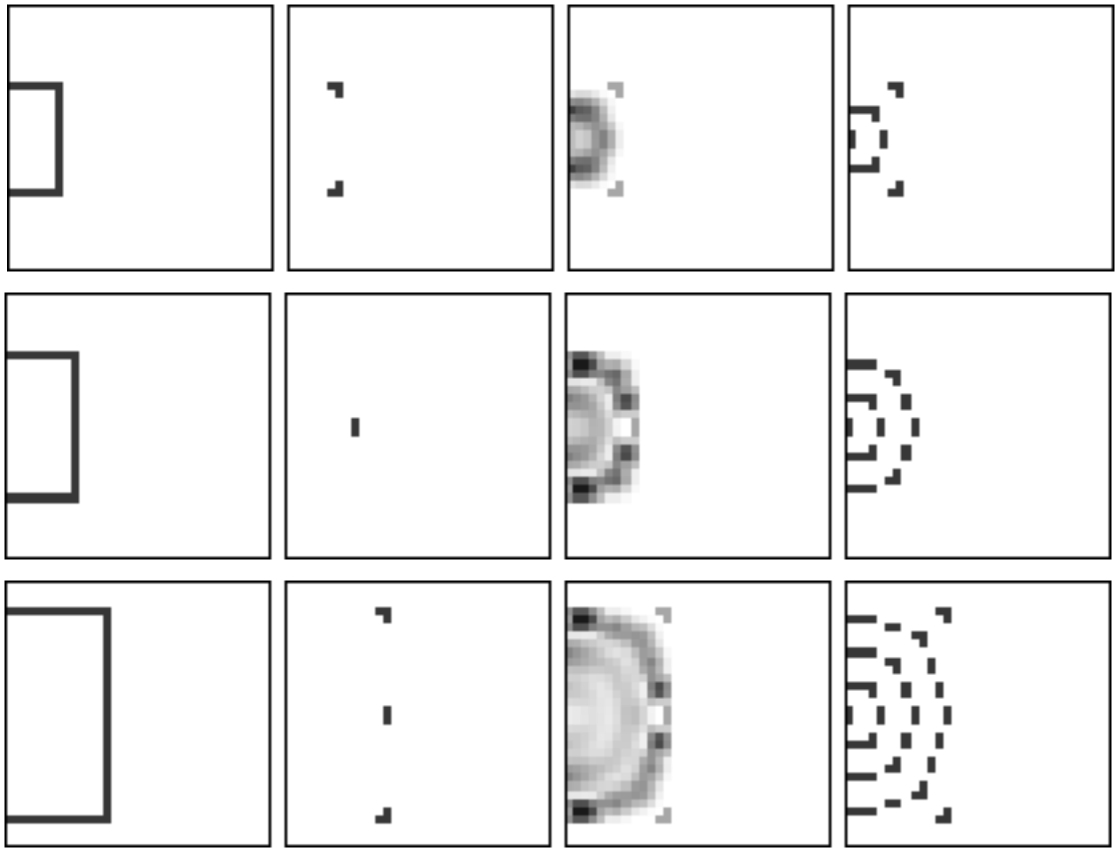


Fig.6-27. Selected snapshots from the developmental sequence of the second handcoded system, inspired by the *Drosophila* eye development. From left to right: the wave, new precursor cells, diffusing inhibitor and precursor pattern.

The essence of what happens is the following: during the formation of the *Drosophila* eye, a ‘wave’ travels over the surface of the eye, leaving in its wake the hexagonal pattern of precursor cells, where each precursor ‘patch’ will later give rise to one segment of the segmented eye. Essentially, the precursor cells at a certain spatial location are formed as the wave travels over that location. When a cell knows it is going to be a precursor cell, it inhibits it neighbors from becoming one, too. This effect plus the motion of the wave generates the pattern.

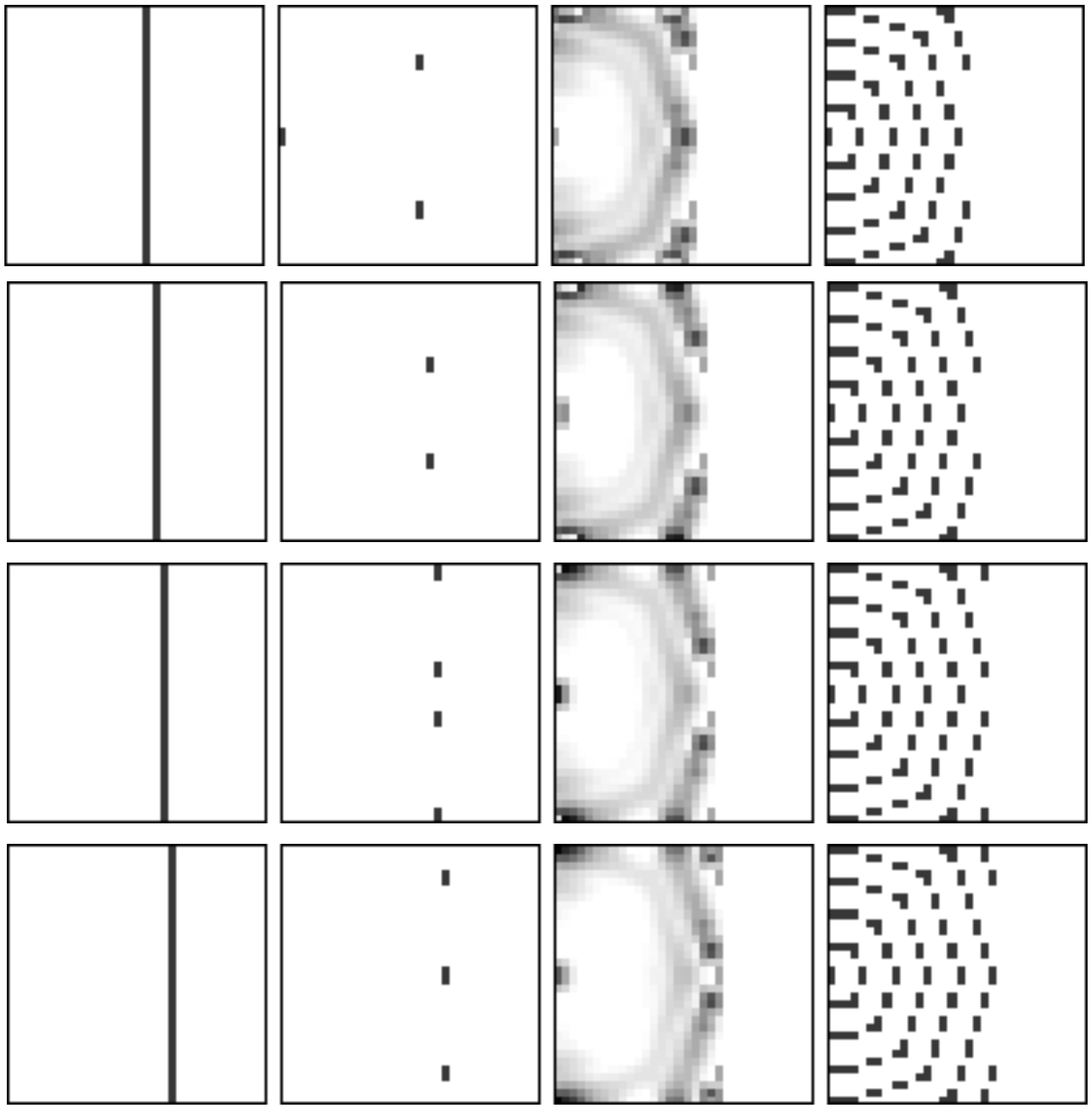


Fig.6-28. More snapshots: Here the wave spans the entire organism.

I have extended the model with a diffusion mechanism to be able to simulate this developmental sequence. One of the genetic products is modeled as a *morphogen*, a diffusable molecule, and at each time step this morphogen is moved from areas of high concentration to areas of low concentration. The morphogen transport between two cells is proportional to the difference in concentration and a diffusion constant.

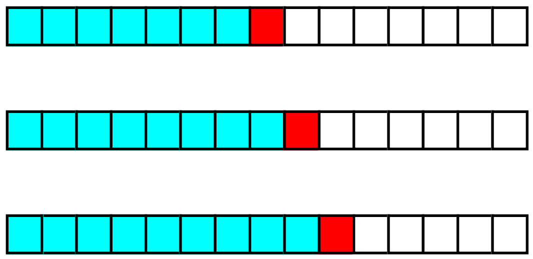


Fig.6-29. The traveling wave is easy to generate using three distinct cell types. The wave (represented by the cell in the middle) indices the cells to the right to become wave cells, and induces the cells to the left to become ‘has been’ cells.

The wave was the result of a second handcoded genome, making use of the cytoplasm-genome model that was already there. A traveling wave is very easy to generate, as at any given moment one can divide the cells up into three categories (see Fig.6-29): those that not have been visited, the ones that are visited by the wave at that very moment and the ones where the wave has already passed through. By letting the wave cells induce new wave cells only in those where the wave has not been, a traveling front can be established easily. All you need is an initial seed. The breaking up of the wave happens in a similar fashion.

The results of these efforts are shown in Fig.6-27 and Fig.6-28, respectively when the wave has not yet reached full maturity, and when it has. As you can see, a nice pattern is left behind. It must be remarked that the process was inspired by the *Drosophila* eye development, but the only thing they have in common is an emerging pattern of ‘dots’ by means of negative inhibition. No claim is made that the biological phenomenon is accurately modeled. It does serve as a nice illustration, however, of what could be achieved when diffusion is added to the model.

Chapter 7

Comparison with Related Literature

A selected overview of the use of developmental approaches in animat research is given in (Kodjabachian and Meyer 1994). Another overview, but with a stronger emphasis on biological relevance, can be found in (Prusinkiewicz 1994). Here we will discuss some of those efforts as well as some others that are not mentioned in the cited overview. The papers are discussed here in the chronological order in which they appear in the literature.

Turing 1952

In 1952, Turing wrote a seminal paper (Turing 1952) in which he suggested that many of the phenomena that are observed in morphogenesis could be explained on a physical and chemical basis alone. The specific question Turing wanted to answer in his paper was whether one could imagine a mechanism whereby spontaneous symmetry breaking would occur in a system of identical cells of the early embryo, giving rise to patterns of minimal and maximal concentrations of certain chemicals (morphogens) in the cells (Kauffman 1993, page 566). Thus, his question concerns pattern formation mechanisms.

The paper is important because it was the first attempt at modeling morphogenesis using a mathematical model. More precisely, he described several sets of differential equations that governed the interactions between different morphogens in one and two-dimensional arrays of cells. These morphogens reacted with each other and could diffuse between cells, giving rise to the term *reaction-diffusion model*. Fig.7-1 is taken from (Prusinkiewicz 1994) and shows how Young (Young 1984), using a variant of Turing's model, produced several different patterns by varying the model parameters that show resemblance to morphological patterns found in nature.

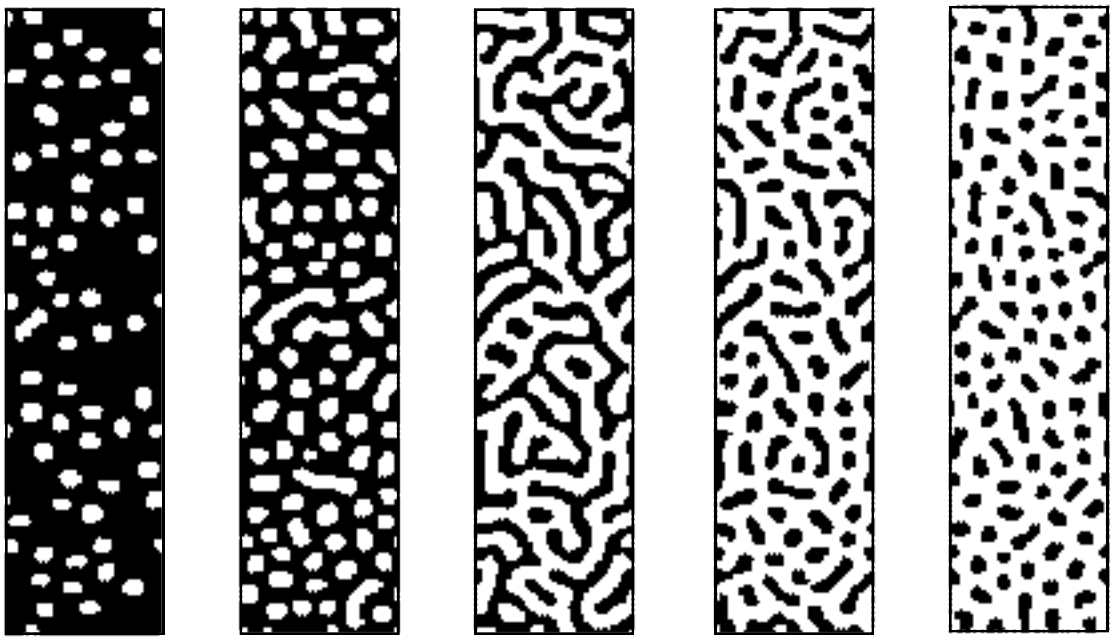


Fig.7-1. Different patterns obtained by Young (Young 1984) using a variant of Turing's reaction-diffusion model. Black are areas of high concentration of some chemical component. From (Prusinkiewicz 1994).

Although pattern formation was not addressed at length in this thesis, it was briefly discussed in Chapter 6, where it was also illustrated with an example inspired by the early development of the *Drosophila* segmented eye. However, in contrast with the reaction-diffusion models discussed above, I have not made use of continuous differential equations. Instead, diffusion is implemented using a simple difference equation rule in discrete time. Also, there is no reaction between the different morphogens in model, only a steady decay.

Another difference between models inspired by Turing's work and the model discussed in the thesis is that there is no attempt in the former to model mechanisms at the genetic regulatory level, or that they are assumed to follow the same form as the chemical reactions that are modeled. The model described here attempts to make a cut through the developmental process as a whole, and as such it was decided to forsake the detailed quantitative modeling that is exemplified by reaction-diffusion models. In addition, although Turing's approach can generate many interesting patterns, the continuum model for a sheet of cells is inappropriate for some of the behaviors that are interesting in development (Fleischer et al. 1994), e.g. the sprouting of axons.

Mjolsness et al 1991

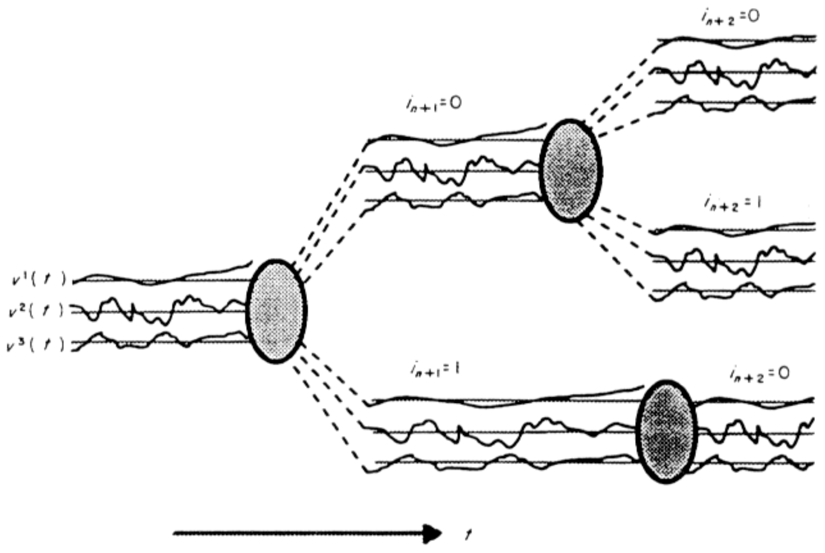


Fig.7-2. In (Mjolsness et al. 1991), the regulatory networks are implemented as dynamical neural networks.

In (Mjolsness et al. 1991) an approach is described whereby a grammar based model of macroscopic cellular events -cell division, cell death- is combined with a continuous neural network model for the regulatory interactions between genes during interphase. In Fig.7-2, you can see how in each *interphase*, the time between cell division or *mitosis* events, the continuous dynamics of the network play out. The continuous time variables represent the state variables that characterize the properties of the cell that one is interested in. In the paper, this work is presented as a modeling framework of which the purpose is, in their own words:

> *. . . to provide a systematic method for discovering and expressing correlations in experimental data on gene expression and other developmental processes.*

The model I have presented bears more than superficial resemblance to their model, as (1) the procedures which govern cell division in our model are implicitly defined in the simulation program but could conceivably be expressed in a grammar based framework, and (2) as in their work, I have explicitly modeled the interactions between genetic elements as a connectionist way, albeit not using continuous time dynamical neural networks, but discrete time random Boolean networks.

There are some important differences, however. (1) Their model is geared towards the discovery or description of actually occurring correlations between essential variables in biological cells, during biological development. This makes it necessary for them to operate in continuous time, at least in interphase, but this is not a stringent condition for our work. (2) Because of this, it is applicable in our case to use the much simpler random Boolean networks, which has important computational advantages. In addition, these discrete time networks lend themselves much more easily to dynamical analysis than continuous time dynamical neural networks, where the dynamics of more than a couple interacting neurons becomes soon intractable (Beer 1994). (3) I have not attempted to capture the macroscopic events in development in a grammar based theory. There might be some advantages in doing so, however, both to better understand and describe the events that take place, as possible benefits it might bring from a computational point of view.

Nolfi et al 1991

Nolfi and his colleagues (Nolfi and Parisi 1991; Nolfi, Miglino and Parisi 1994) have presented a developmental approach to the evolution of artificial neural networks that is based on the outgrowth of axons. Here, artificial organisms are presented that can move around in a two-dimensional world, looking for food. These organisms have sensors and actuators by which they can respectively perceive their environment and move around in them. The papers then discuss a developmental encoding of a neural network that will serve as the organisms' nervous system, and they use genetic algorithms to evolve organisms that are better at finding food.

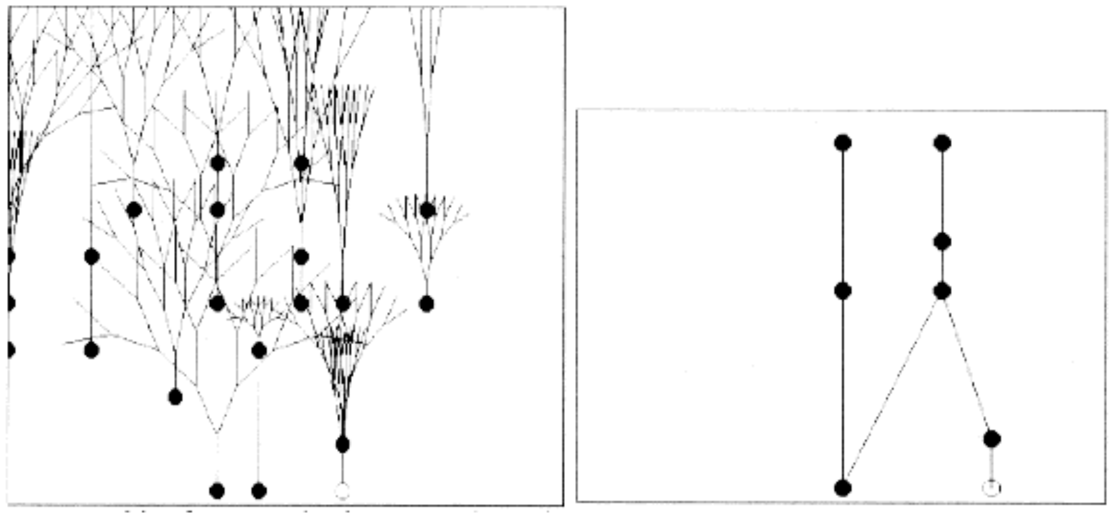


Fig.7-3. Development of a nervous system in the work of Nolfi et al. Left: the situation after axonal trees are fully grown. Right: the resulting network after elimination of non-functional connections. Taken from (Nolfi et al. 1991).

The development of the neural network proceeds as follows. First, neurons appear on an organism's body, which is assumed to be rectangular in shape. Then, these neurons proceed to send out axons in the form of branching trees. When one of the branches in an axonal tree hits another neuron, a connection is established. The situation after this has happened is shown in Fig.7-3, left panel. Subsequently, the resulting network is examined on non-functional connections, and the nervous system of the organism is the one that remains after elimination of these superfluous connections, as shown in fig.7-3, right panel.

This developmental process is encoded on an artificial genotype for use with the genetic algorithm. Each neuron has its own block assigned on this fixed length genotype, and each block specifies the position and properties of the neurons as well as the branching characteristics of the axonal trees. They have also examined the role of phenotypic plasticity, i.e. the further development of the neural networks while the organism is already functioning in the environment, in which case the genotype also contains information on when during development the neurons will be expressed. Please refer to (Nolfi et al. 1994)where this interesting question of phenotypic plasticity is examined more closely and tested with actual robots.

Although Nolfi et al. use a model of axonal growth, as the model presented in this text, there are some important differences between the two approaches. (1) There is no notion of a developing body morphology in Nolfi's work, thus co-evolution of bodies and brains cannot be addressed in his model. (2) The model of axon outgrowth in his model is quite mathematical, whereas the model presented here is more biologically plausible. However, a point that has already been made, this inspiration from biology can also be a disadvantage when interested in neural network design alone. In addition, Nolfi goes a long way towards the biological end of the spectrum compared to some other approaches. (3) The genetic representation he uses is developmental in the sense that neural connections are the result of a growth process, but neurons are still coded for using a direct mapping approach. In our model, the specification of neurons is a part of the developmental specification. (4) He uses an implicit fitness function. Organisms are deposited in the environment(s) and have to evolve to cope with competition of other organisms. There is no explicit fitness function that examines the behavior of the organisms and accords a fitness value to it: only the final amount of food gathered matters. This approach might actually be the better one, a point that I will return to in the discussion chapter.

Cangelosi, Parisi and Nolfi 1993

A extension of Nolfi's work very relevant to this thesis is found in (Cangelosi, Parisi and Nolfi 1993) where the authors provide Nolfi's model with cell division and cell movement mechanisms. The cell division mechanism is especially interesting to take a closer look at, as it is of course very closely related to the model we are proposing.

In the paper, a population of 100 organisms is subjected to evolutionary pressure to evolve the correct feeding and drinking behavior, depending on an internal motivation that specifies whether the organism is hungry or thirsty. This is the same artificial environment as the one used in

(Cecconi and Parisi 1991). The idea is to evolve the most successful nervous system using a developmental model.

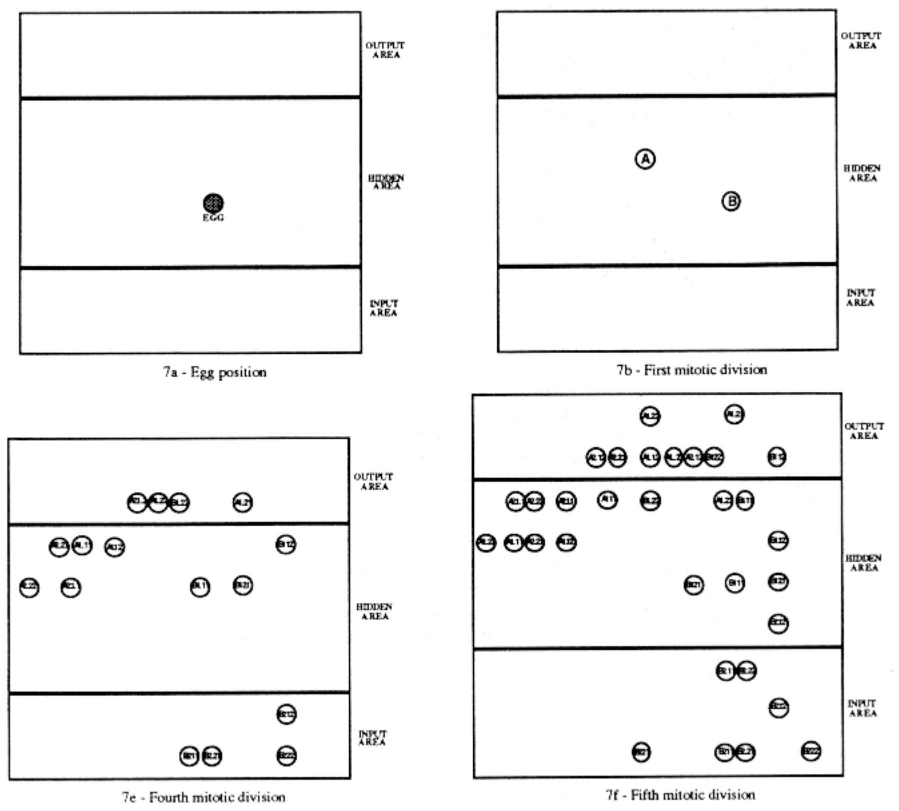

OUTPUT AREA
HIDDEN AREA
INPUT AREA
EGG
7a - Egg position
A
B
7b - First mitotic division
7e - Fourth mitotic division
7f - Fifth mitotic division

Fig.7-4. Four stages in the developmental sequence as discussed in (Cangelosi et al. 1993).

Development starts out with one cell of the egg-type, placed in the center of the space representing the artificial organism. This first cell will divide 32 times to yield a total of 32 cells in the adult organism. A grammar-like system determines what kind of cell-division events will take place: 16 different cell types are assumed, and for each cell type a rewriting rule is present in the genome. For example, a rule might be

Rewrite Type 5 as Type 2 + type 14

These sixteen rules thus provide a rule-based specification of the cell lineages that will exist during the developmental phase of the organism. Selected stages in the developmental sequence are shown in Fig.7-4.

The rules also specify how the daughter cells differ from their mother cell in neuronal properties and in spatial position. This last feature is the one used to specify 'cell movement', although it is in fact a very limited form of migration: only the relative position of the daughter cell with respect to its parent is indicated. There is no movement after the cell division effect has taken place.

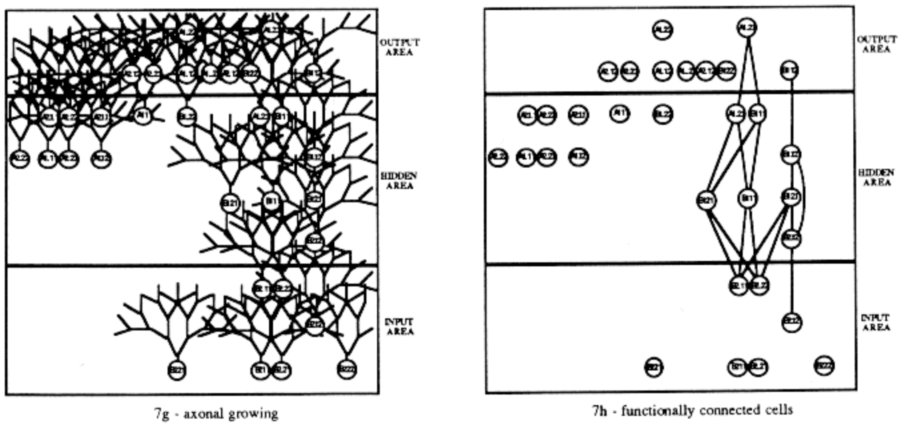


Fig.7-5. The neural developmental phase as discussed in (Cangelosi et al. 1993).

After this 'mitotic phase', the neural developmental phase takes place. In the model the same mechanism for axon outgrowth is used as in (Nolfi et al. 1991), with the genotype specifying branch length and angle. An example of an organism following this phase is shown in Fig.7-5, both right after axon growth but also after non-functional branches have been removed. The resulting network is used as the nervous system of the artificial organisms that are then evaluated on their behavior.

Gruau 1993

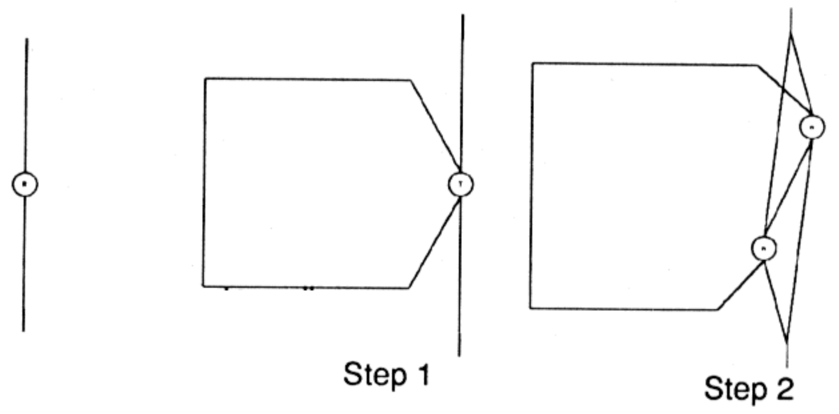


Fig.7-6. Illustration of the developmental process in the graph grammar based approach of Gruau. From (Gruau 1994).

Gruau (Gruau et al. 1993; Gruau 1994) describes an approach to the design of neural networks that is much less biologically inspired but is also based on the idea of using a developmental specification to encode neural networks. In his approach, the product of the developmental process is the directed graph that results from the successive application of graph rewriting rules, and this graph is then subsequently mapped onto a neural network. An example of the three first steps of a developmental process is shown in Fig.7-6, illustrating in step 2 an instance of a ‘cell division’ event.

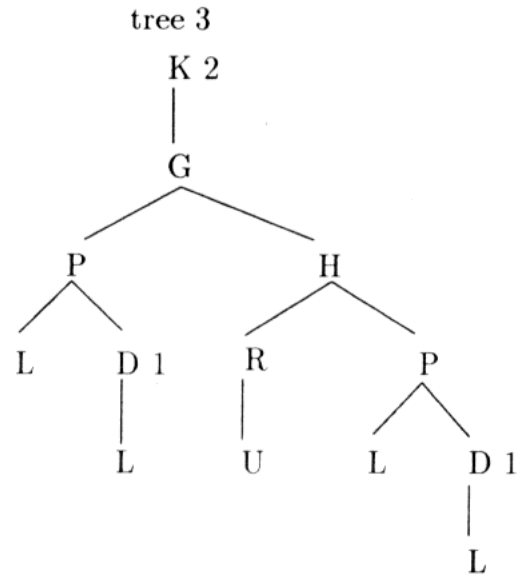


Fig.7-7. Example of a grammar tree, constituting a developmental specification of a neural network in Gruau’s approach. From (Gruau 1994).

The genetic encoding he uses to specify which rewriting rules to use at what point, constituting a developmental specification of a neural network, takes the form of a *grammar tree*, whose nodes represent particular graph transformations. An example is shown in Fig.7-7. Development starts with one cell in the graph and with the *reading head* at the root of the tree. Then, as developmental time proceeds, the reading head moves down the tree, generating a cell division event each time the grammar tree branches. Thus, after the first division, two reading heads will be present, making the algorithm intrinsically parallel. In fact, one can view the grammar tree as a lineage tree of cell types.

The developmental model in this thesis is of course far removed from Gruau’s paradigm both in implementation and in the perceived applications. This text stresses the biologically defensible aspect of the model, while Gruau explicitly rejects it. Also, our model is not grammar-based, and the neural development is rooted in a developmental model of body morphology, which is totally absent in Gruau’s model[20]. We have opted for a biologically defensible model because the desire to address questions in theoretical biology, and this goes a long way to explaining the different

[20] Although he does map to an situated agent body, a simulated six-legged robot inspired by (Beer 1990).

choices made in this text. It is interesting to note, however, that for our model an a posteriori grammar tree could be derived from the dynamics of the developmental process. Thus, it would be of value to compare this type of trees in both models, even though the directed process is totally different.

Fleischer and Barr 1994

In (Fleischer et al. 1994), the authors present a simulation framework to study developmental phenomena. A central role in their model is assumed by the model *cells*, whose behavior is controlled by *state variables*. A set of *cell state equations* make up the 'genome' of the cell and govern how the state variables change over time. These equations have a conditional part, loosely simulating the regulation of gene expression within the cell. In addition, rather computationally involved phenomena like diffusion and mechanical forces are simulated as well.

A defining characteristic of their model is that all changes in the environment, within the cells and the interaction between the two are implemented using a set of differential equations. These are numerically implemented using a piecewise-continuous ordinary differential equation solver. All discrete events occur when certain *behavior functions* cross zero. For example, a cell division will occur when the function TimeToSplit reaches zero.

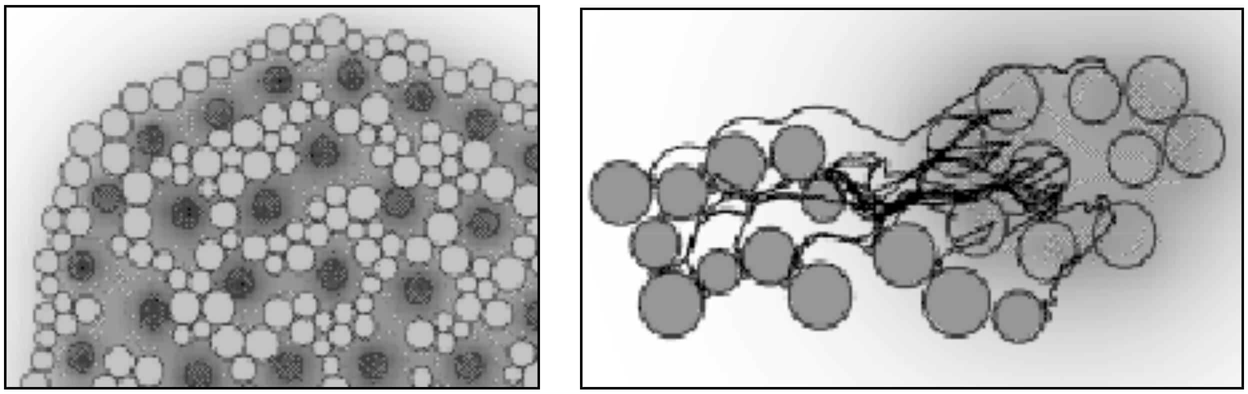


Fig.7-8. Left: differentiation of cells in the simulation framework of (Fleischer et al. 1994). Right: simple neurite growth. Used with permission. Copyright Kurt Fleischer and Alan Barr.

With this very general model, they have been able to simulate very interesting developmental phenomena by handcoding these cell state equations in a number of ways. Thus, see Fig.7-8, they could simulate simple behaviors, e.g. a cell climbing a gradient, to more complex behaviors, e.g. cell differentiation and even neurite growth, based on chemo-attraction. The way they implemented growth cones is particularly interesting: it is simply a miniature cell attached to the end of the neurite. Indeed, the movement of a growth cone uses many of the same mechanisms seen in cellular movement, and thus it makes sense to regard the growth cone as a mini-cell in this respect.

There are of course many differences between their model and the one proposed in this thesis, beginning with the representation and dynamics of cell 'state variables'. First of all, we have adopted a more simplified approach and have used binary variables instead of continuous variables. We have taken a discrete time approach, versus the continuous time approach taken in (Fleischer et al. 1994). And finally, we use a state transition table based approach (albeit distributed over the different Boolean functions) versus the differential equations that govern the dynamics of state variables in their model. Thus, the consequence of all this is that our model is less computationally expensive. However, the downside to this is that it is also much less general.

Another important difference concerns the genetic representation of the cell's behavior. We have conceived our model from the start to be useful in conjunction with genetic algorithms, something that was not yet present in their paper (where the genomes are always handcoded), although the intention is certainly there. Remember that this design goal motivated us to make the model of gene regulation and cell behavior as simple as possible. The generality of their model and the accompanying computational overhead will be a problem they have to deal with when they want to follow the same route.

Despite these differences, there are also some important similarities to note. One concerns the intended use of the model: both our models have been conceived first with the automatic design of neural networks in mind, but with an eye towards applicability in biology. In this respect, our model addresses phenomena at a higher level, i.e. behavior of organisms in a simulated environment, although their model is much more biologically plausible with respect to the origin of multicellularity. The second is in the relation to grammar based systems: in both models, random Boolean networks in one and conditional expressions in the other, assume a similar function to that of the grammar rules in grammar based approaches, however at a lower level of abstraction (paraphrased from (Fleischer et al. 1994)[21]).

Kitano 1994

In Kitano's revised model (Kitano 1994), the core of the model is a simulation of the cell's metabolism, described by a set equations modeling the change in concentration of the cell's

[21] Note that this statement was made in (Fleischer et al. 1994) with their model in mind only, and that the analogy with random Boolean networks is made by the author.

chemicals. These changes are governed by a set of *metabolic reactions* catalyzed by *enzymes*. Which enzymes are produced at each time step - the simulation is in discrete time - is in turn decided over by a set of *metabolic rules*. These rules play an equivalent role as the Boolean functions used in my model: they serve to interpret the changing conditions within a cell and respond with a certain behavior. As expected, it is the metabolic rules that are coded for in the genome, and thus mutated by the genetic algorithm.

A developmental process starts with a cell in a *medium* that also contains chemicals. In fact, in parallel with the metabolic reactions described above, equations modeling diffusion and active transport of chemicals to and from the medium are implemented as well. Intercellular communication is simulated by dividing the cells into regions: the chemical concentrations of the medium in a certain region then act as a kind of global variable for that region, and Kitano has indeed found that cells are able to synchronize their behavior using this mechanism.

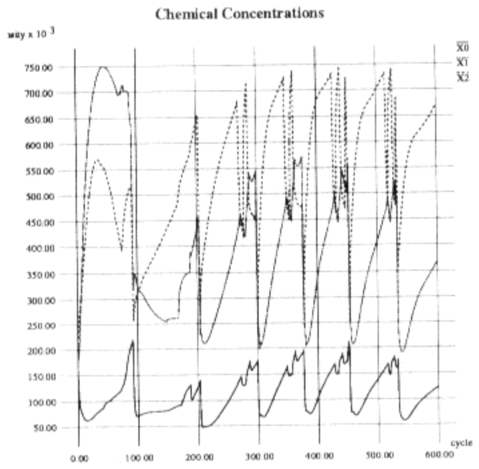


Fig.7-9. Trace of the ‘cell cycle’ from (Kitano 1994).

In the paper, Kitano describes a set of experiments in which he tries to evolve a simple cell cycle, using the model of metabolism described above. In the model, cells can divide or die depending on the result of the metabolic reactions. By using a performance function that rewards a large number of cells at the end of a simulated time period, a cellular metabolism that implements a cell cycle with regular cell divisions is evolved. The concentration of three chemicals over time in a cell with such an evolved cell cycle is shown in Fig.7-9: as can be seen, the cell cycle results from the highly complex and non-linear interactions of the chemicals by means of the metabolic rules and reactions.

Sims 1994

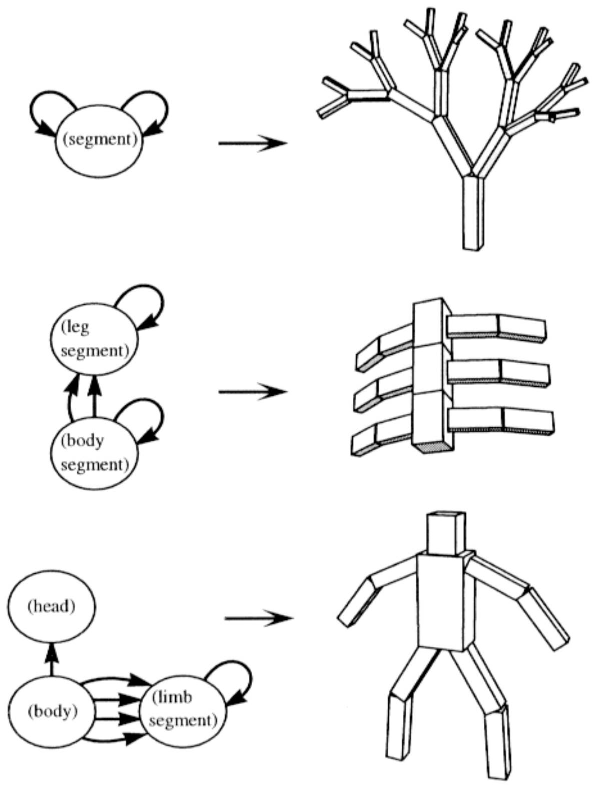


Fig.7-10. Examples of some genotypes and their corresponding body morphologies in Sims' work. From (Sims 1994).

A very successful approach to the co-evolution of body morphology and nervous system is taken by Karl Sims (Sims 1994; Sims 1994). Sims' creatures are hierarchical structures of rigid, three-dimensional body parts, connected by joints. They are built up according to a developmental specification contained in the genotype, which is a directed graph. Each node in the graph contains parameters that influence the developmental process. Examples of some genotypes and their corresponding body morphology are given in Fig.7-10. As you can see, the nodes correspond to body segments, and the arcs show how to build the creature morphology, starting at a root node. Recursive interpretation of the graph results in segmented body structures.

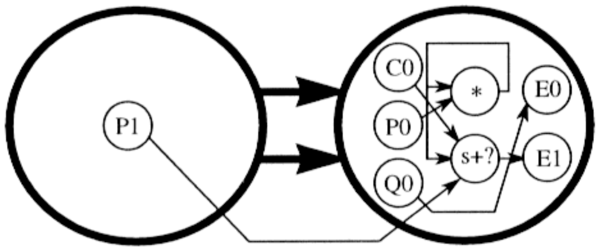


Fig.7-11. Nested graph describing body and nervous system. From (Sims 1994).

The nervous system of the creature is a dynamical system accepting sensory inputs and providing outputs to effectors, the creature's muscles. It consists of 'neurons' (nodes implementing a variety of functions, not really like neural-network neurons), sensors and actuators and the connections between them. The genetic specification of the nervous system is also a graph, which is nested in the higher level morphology graph. There is no developmental component here, however: the nervous system graphs are just copied into the body parts and connected appropriately (possibly to adjacent body parts or to a central control module; the latter is also specified using a graph). An example of the nested graph genotype is shown in Fig.7-11.

Although not biologically realistic as far as development goes, the introduction of a realistic world model goes a long way to produce creatures that exhibit remarkably life-like behavior. Since the dynamics of the model world mimics that of the real world, the strategies for propulsion invented by the evolutionary process mimic solutions that have been evolved in biology. For example, the swimming behavior of some creatures is very like that of certain sea-animals.

Chapter 8

Discussion and Conclusions

Contribution

We have built a simple, evolvable yet biologically defensible model of the developmental process.

At the basis of this model lies a model for *genetic regulatory networks*, which we have essentially modeled using random Boolean networks. Developing a dynamical systems perspective for this model brought us a powerful tool for analysis of development at the level of gene regulation, but it has also proven useful at higher levels of complexity. In addition, we have shown that we could evolve simple regulatory circuits relevant for development.

It was shown, at the *multicellular level*, that the model can generate a range of morphologies in the square-cell organism context, and that it is evolvable: the genetic regulatory networks can be evolved to optimize some performance function *for the fully developed organism*. Moreover, we have analyzed in some detail the developmental sequence of an artificial organism that was evolved to exhibit some agent-like properties.

We have presented an extension to the multicellular model to account for *neural development*, demonstrated it by handcoding a functional agent, and we have shown that the extended model can be evolved as well to yield better performing agents. In addition, the developmental process used is robust in the face of changes in the morphology of the agent. For example, we have seen this when we quadrupled the number of cells: the resulting phenotype was qualitatively the same, and we get functional agents without any change in the genome, which is remarkable. This opens the way for the co-evolution of body and brain in the field of Simulated Adaptive Behavior.

Throughout, we have discussed no less than three different models for gene regulation. One was based on a ideal paradigm, random Boolean networks, available in the literature and intuitively appealing. Problems of cell lineage in multicellular development and the desire to have gene-memory as a rule prompted us to look into the three state model. And, the (historically earlier) cytoplasm model was used to implement cell behavior and to have some control over timing (see below).

Directions for Future Work

The process of building the model has not proceeded without difficulties, although their solutions always seem so obvious in retrospect. Along the way, we have learned some lessons, and these can serve to guide us in future work. The approach of exploration that has been taken while building the model has sparked a multitude of ideas on how and where to proceed further down the road in this exciting area of research. In this section I will discuss some of these ideas.

General Considerations

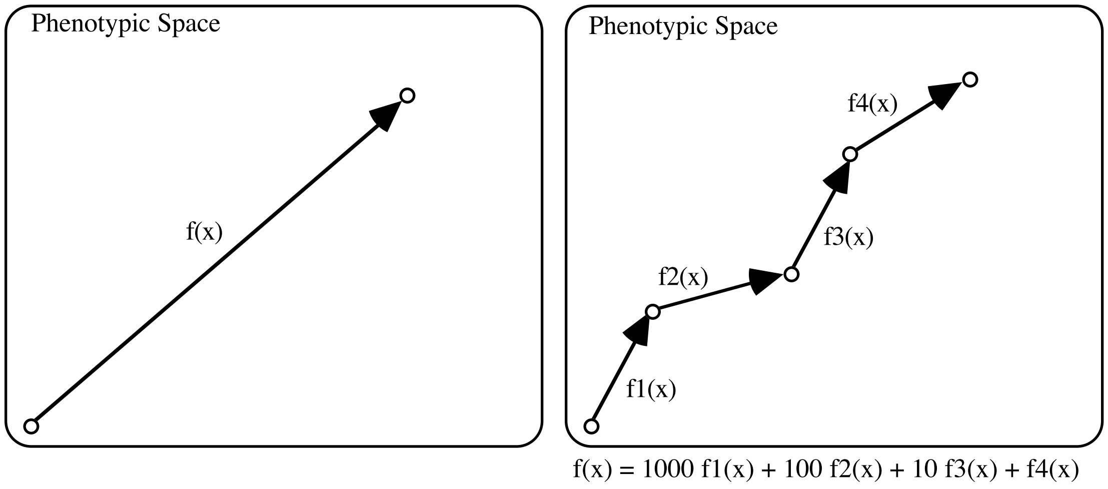


Fig.8-1. A multilevel performance function might make the problem of finding a particular point in phenotypic space more tractable.

One of the problems with evolving these artificial organisms is that there is only one particular performance function at the end of the process. In nature, each step during evolution was the result of an implicit performance function which never relaxed. Each cell, for instance, needs a metabolism. This has evolved early on in evolution, and no cell can afford to lose it. In this way, many earlier accomplishments of evolution are maintained and provide a constant basis from which to evolve new mechanisms. Even if the new mechanisms take over the function of older systems, the earlier structures often retain a function as developmental precursors for the new structures, i.e. they assume an interphene role. In contrast, as our model is currently conceived, the performance function is explicit and geared towards one particular goal. This might help to explain why we were never able to evolve an organism that was able to exhibit the avoidance behavior from scratch: the problem that the GA has to solve might simply be too complex to evolve in one step, and there is no incentive or reward to evolve intermediate steps[22].

One way to address this problem might be to construct multilevel performance functions that also select for intermediate steps. The problem and this possible solution are illustrated in Fig.8-1: here, instead of posing a formidable challenge to the genetic algorithm, for example to evolve a complex neural controller, we might decompose the problem into successive sub-problems. In fact, we have already done this in Chapter 5, where we used a multilevel performance function to evolve colorful organisms: the problem was decomposed into two sub-problems: (1) finding all colors, and (2) balancing the colors. However, decomposing the problem might be as complex or even more difficult than simply solving the problem itself. In the example, the design of neural controllers, which we wanted to automate using genetic algorithms, has simply been shifted to the design of adequate performance functions, with no guarantees that this is easier.

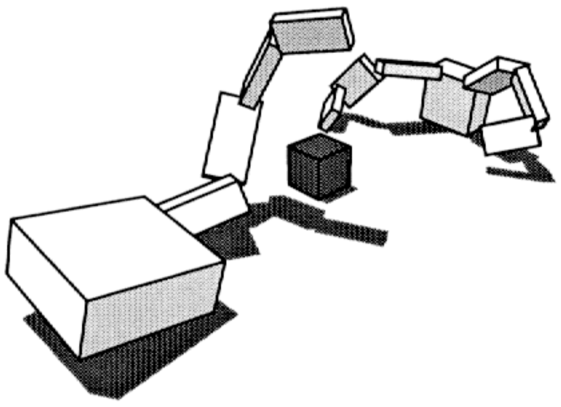


Fig.8-2. Two competing creatures from (Sims 1994).

Maybe the best way to address this problem, is to work with implicit performance functions, for example by constructing a toy world in which simulated agents are evolved. This approach is very common in Adaptive Behavior research. The idea is the following: if the simulated environment in which adaptive agents are evolved is rich enough (for example many sensory clues and enough of a challenge for organisms to survive) a basic behavioral repertoire can be built up by simply evolving creatures that successfully cope with the world. This is based on the idea of co-evolution: as organisms in this world evolve and interact, surviving in the world becomes progressively harder since organisms have to compete with one another. The potential success of

[22] Other reasons have to do with the neural developmental component of the process and are discussed further in the text.

this approach is illustrated by Karl Sims' work (Sims 1994), where organisms are pitted against each other in 'tournaments' (Fig.8-2), and stasis in evolution is quickly punished be extinction. The creatures evolve increasingly complex strategies to regain the evolutionary advantage, and different 'species' specialize in different strategies.

Genetic Regulatory Networks

At the regulatory level, the cytoplasm model was implemented to enable genetic elements to exert a function within the cell. One of the main reasons to modify the pure RBN model was intercellular communication. A problem with RBNs is that communication is explicit by graph edges to other cells. In the cytoplasm model, this function is taken up by the agent-receptor-detector triad. These are regular gene products, opening up the possibility to have genetic control over communication, for example, dependent on cell type. This was possible with RBN but in a much less natural way: a Boolean function has to be explicitly evolved to control the reaction of a gene to a signal from outside the cell. The 'modulating' element can then be seen as the 'receptor presence' gene. However, if you want conditional communication to be the rule rather than the exception, making that explicit as in the cytoplasm model seems better.

One of the main limitations of both models, however, is that there is no mechanism to control timing of the expression of genetic elements and/or gene products. We know from biology that timing considerations play a very important role in development and, even more so, in the evolution of a developmental process, see for example (Gould 1977; Raff et al. 1983). Thus, we can definitely improve the model by providing a mechanism for timing.

I have already implemented this for the cytoplasm model (although it has not yet been used in the experiments): instead of simply recording the presence of a gene product in the cytoplasm, a new attribute represents the *concentration level* of the gene product. Thresholding is then used to convert that level for use in the Boolean regulation functions. Since both *threshold* and *production rate* for the gene products can be coded on the genome, we effectively have control over timing. More work needs to be done here. As in (Nolfi 1994), we could examine whether phenomena like heterochrony and recapitulation will arise in our artificial phylogeny, making it interesting from a biological point of view. In addition, from a pragmatic standpoint, I am convinced that this additional degree of freedom will buy us a great deal in the way of evolvability.

Another promising start was made in the discussion of the three-state model, an extension of the random Boolean networks that gave rise to more stable cell lineages. These are beneficial in the development of multicellular organism and make development easier to analyze. However, I feel that a more thorough examination of this subject is in order. It would be very interesting, for example, to go to the biology and take a closer look at the patterns of gene expression over time in development, get a better feel for what is happening, and try to relate that back to the model. Also, the three state idea was never extrapolated to the historically earlier cytoplasm model.

Thus, one way to proceed at the level of genetic regulatory networks is to combine the advantages of both the cytoplasm model and the three-state model in one. For example, one could (1) replace the Boolean functions in the cytoplasm model operons by the event lookup tables as discussed in Chapter 5, and (2) turn the expression of gene products irreversibly off once an OFF event is generated. In addition, this new model should be capable of exerting control over developmental timing: careful consideration should be given whether the mechanism sketched out above is adequate, or whether another approach might be better suited to the combination of these two models.

Finally, one of the contributions of this thesis was to develop a dynamical systems perspective for the genetic regulatory networks, although it has only been thoroughly worked out for the pure random Boolean network model. It has provided us with an excellent tool for analysis but maybe even more importantly, a frame of reference in which to discuss higher level events that happen during the course of development. Whatever direction is taken as far as the model is concerned, I feel that it is important to extend this dynamical view of development beyond the pure RBN networks in one dimension, and towards higher level phenomena in the other. In addition, and more ambitiously so, I hope that one day such an approach might be extrapolated for use by developmental biologists, perhaps as one of the tools in the field of theoretical developmental biology.

Multicellular Level

At the multicellular level, many important aspects of biological development that we have excluded from our model provide rich developmental possibilities and could be taken into account. Here we will discuss some of the problems that arose in our model and suggest some directions to address them.

Perhaps the most important problem with the current square-cell model is the lack of adequate positional specification. In fact, the only spatial clues that are provided to the cells are (1) the external environment, (2) the midline, and (3) the asymmetry in the 'zygote'. All positional specification must follow by interaction and elaboration on these three primitives. This makes it difficult to have such common phenomena like dot patterns, or discrete lumps of cells that express the same genes and that might represent 'organs'. In contrast, in real biological development a number of mechanisms exist to organize the body in complex ways: one of the most important ones is probably coordinated cell movement, i.e. deforming cell sheets that give rise to pouches, vesicles, tubes and rods as illustrated in Fig.2-8.

Another way to address this is to model gradients of morphogens, i.e. diffusing regulatory molecules, instead of just binary induction between neighboring cells: such morphogens are hypothesized to underlie both the expression of segmenting genes (Walbot et al. 1987, page 641) as the pattern formation in limbs (Wolpert 1977). A glimpse of what that might buy us was provided in Chapter 6, where a system was set up that mimicked pattern formation in the *Drosophila* eye, and which used a diffusable substance that inhibited expression of certain genes in the neighboring cells. It is clear that we have barely scratched the surface of what is possible here.

It would also be of value to look at a model where symmetry breaking is the norm, rather than the exception. Continuous time networks with some introduced noise component are an option, as are asynchronously updating Boolean networks.

However, as already suggested, a more sophisticated model of cells at the physical level might be the most important change that is required, i.e. make it possible to assume changes in shape and cause morphological change by division, for example. Luckily, the implementation details of the model make it easy to substitute a different model of the cell in place of the simple square geometry used.

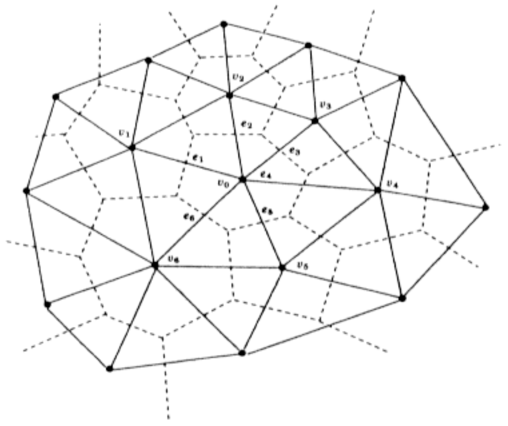


Fig.8-3. The topological cell model. Taken from (Duvdevani-Bar and Segel 1994).

I have briefly experimented with linear organisms, i.e. cells connected together as a simple filament, which worked well. Just these might already have a number of interesting applications: there are such organisms in biology that exist as filament like structures, e.g. blue-green algae. The extreme simplicity of such a system, nevertheless exhibiting many fundamental characteristics of development, has the great advantage of making all the issues very clear. Such work has already been undertaken using grammar models, but there an underlying genetic regulatory mechanism is not addressed. A model for pattern formation in blue-green algae that takes genetic information into account has been proposed in (Wilcox, Mitchison and Smith 1973).

A model that looks very promising is the graph-based topological model described in (Matela and Fletterick 1979; Duvdevani-Bar et al. 1994), where the cell system is represented by a triangulated graph and the vertexes of the graph represent the cell centers, as shown in Fig.8-3. The model is quite sophisticated in its capabilities (cell movement, division, cell-sorting, etc..) but it is still computationally tractable. Many mechanisms of morphological change depend on such capabilities that simply can not be captured by the square model. Thus, to enhance the model's capabilities with respect to development of morphology, this model is definitely worth looking into.

Neural Development

Finally, as far as neural development is concerned, although we have shown in Chapter 6 that we can incrementally improve on a working design using the genetic algorithm, we have not yet been able to evolve a functional agent from scratch. There may be several reasons why this is true:

A valid criticism of the model might be that, when primarily interested in evolving neural networks, we have simply taken the analogy with biology too far. For example, although we speculate at one point at the benefits of modularity that a developmental specification might bring us, it might be better to encode modularity explicitly, as is done for example in the grammar based approach of (Gruau et al. 1993; Gruau 1994). However, one of the premises set out in the introduction is that we *are* interested in the biological mechanisms of development. Adopting a biologically defensible model might provoke some thought on such problems as the evolution of early nervous systems, for example. In addition, it is difficult to ignore the successes of biology with respect to nervous systems: it seems not unreasonable that we can learn a lot by looking more closely at the underlying mechanisms of biological neural development, even if we are only interested in potential engineering applications.

Another problem is with the model underlying 'axon' outgrowth in our artificial development: there are several reasons why one could criticize using the CAM model as the only mechanism for axon guidance. For instance, in biology, CAM molecules seem to play a role as a general 'glue' and to facilitate axon growth in general (Brown et al. 1991, page 55), rather than provide guidance about direction. Also, chemotaxis, 'stepping stone' mechanisms and the following of other axons have been all been shown to be important in one system or another. Thus, the mechanism that we have modeled might have to be re-evaluated.

Often many more neurons and connections are present in early neural development that are needed in the adult organism, which are then later pruned away according to some selection mechanism, e.g. synchronized activity stabilizing a connection, or trophic factors etc. Implementation of such mechanisms might make genetic search easier. More often than not it was the case that in the model, neurons and axons grew indiscriminately without any direction to them. A selection mechanism might provide a way to structure the resulting mess. Interestingly enough, the mess originates because 'CAM' is expressed all over or in large groups, and that is exactly what the previous point was about: maybe CAM molecules serve only to group neurons, not to provide directional clues. An interesting discussion on these ideas including some computer models can be found in (Edelman 1987).

Finally, some issues have been touched upon but have not yet been completely implemented. For example, although the genome can code for different kinds of neurotransmitters and receptors, at this point there is not yet any functional significance attached to them in our implementation. The same argument holds for trophic deficiencies and resulting neuronal death.

Conclusion

Computer modeling is becoming an important tool in Biology, and it should not be restricted to the detailed modeling of isolated phenomena. Approaches like the one presented in this thesis, that make a cut through many different levels of complexity (albeit in 'artificial' organisms) will grow to become increasingly important. Questions about the interaction between evolution and development, for example, can now be posed in a new context, and new insight can be gained. To borrow an analogy from astronomy: the colliding of galaxies cannot be studied by looking at the stars alone.

In Artificial Intelligence research, 'intelligence' is increasingly looked upon not as deliberative reasoning processes alone, but as the ability to exhibit adaptive behavior in a complex world. The synthesis of autonomous agents is seen as one of the most promising ways to approach the problem of intelligence from the bottom upwards. Much work in the field of Simulated Adaptive Behavior is inspired and driven by findings from Ethology, i.e. the study of how behavior in animals has evolved to cope with an increasingly complex world. I believe that evolution and development are inseparable, however, and this has prompted me in turn to look towards Developmental Biology as a source for inspiration.

These two points sum up the main reasons why I pursued the research that led to this thesis. I have presented what I hope is an interesting attempt at modeling the process of development as a whole, with applications both in Theoretical Biology and Simulated Adaptive Behavior. In addition I have laid out some questions which I feel are important to address in the future, and provided my own perspective on how those questions may be answered. As such, this thesis can be viewed as a road map for future work, and I hope that it will effectively serve that purpose.

References

Alberts, B., D. Bray, J. Lewis, M. Raff, K. Roberts and J. D. Watson. 1983. *Molecular biology of the cell*. New York: Garland Publishing.

Alberts, B., D. Bray, J. Lewis, M. Raff, K. Roberts and J. D. Watson. 1991. *Molecular biology of the cell*. New York: Garland Publishing.

Beer, R. D. 1990. *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology*. San Diego, CA: Academic Press.

Beer, R. D. 1994. "On the Dynamics of Small Continuous-Time Recurrent Neural Networks." Technical Report Report CES-94-18, Dept. of Computer Engineering and Science, Case Western Reserve University.

Beer, R. D. and J. C. Gallagher. 1992. "Evolving dynamical neural networks for adaptive behavior." *Adaptive Behavior* **1** : 91-122.

Brown, M. C., W. G. Hopkins and R. J. Keynes. 1991. *Essentials of Neural Development*. Cambridge, UK: Cambridge University Press.

Buss, L. W. 1987. *The Evolution of Individuality*. Princeton, New Jersey: Princeton University Press.

Cangelosi, A., D. Parisi and S. Nolfi. 1993. "Cell Division and Migration in a 'Genotype' for Neural Networks." Technical Report PCIA-93, Institute of Psychology, C.N.R.-Rome.

Cecconi, F. and D. Parisi. "Neural Networks with Motivational Units." In *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, Honolulu, Hawaii. 1991.

Cliff, D., P. Husbands, J.-A. Meyer and S. W. Wilson, ed. 1994. *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press.

Davidson, E. H. 1990. "How Embryos work: a comparative view of diverse modes of cell fate specification." *Development* 108: 365-389.

Dawkins, R. 1989. "The Evolution of Evolvability." In *Artificial Life.*, edited by C. G. Langton. Reading, MA: Addison-Wesley.

Dellaert, F. and R. D. Beer. 1994. "Co-evolving Body and Brain in Autonomous Agents using a Developmental Model." Technical Report Report CES-94-16, Dept. of Computer Engineering and Science, Case Western Reserve University.

Dellaert, F. and R. D. Beer. 1994. "Toward an Evolvable Model of Development for Autonomous Agent Synthesis." In *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems.*, edited by R. Brooks and P. Maes. Cambridge, MA: MIT Press.

Duvdevani-Bar, S. and L. Segel. 1994. "On Topological Simulations in Developmental Biology." *Journal of theoretical Biology* 166: 33-50.

Edelman, G. M. 1987. *Neural Darwinism.* New York: Basic Books, Inc.

Fleischer, K. and A. H. Barr. 1994. "A Simulation Testbed for the Study of Multicellular Development: The Multiple Mechanisms of Morphogenesis." In *Artificial Life III.*, edited by C. G. Langton. 389-416. Reading, MA: Addison-Wesley.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, Mass.: Addison-Wesley.

Goldschmidt, R. 1940. *The Material Basis of Evolution.* New Haven, Connecticut: Yale University Press.

Gould, S. J. 1977. *Ontogeny and Phylogeny.* Cambridge, MA: The Belknap Press of Harvard University Press.

Gould, S. J. 1980. *The Panda's Thumb, More Reflections in Natural History.* New York, NY: W.W. Norton & Company.

Gruau, F. 1994. "Efficient Computer Morphogenesis: A Pictorial Demonstration." Technical Report 94-04-027, Santa Fé Institute.

Gruau, F. and D. Whitley. 1993. "The cellular development of neural networks: the interaction of learning and evolution." Research Report 93-04, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon.

Harvey, I. 1994. "Evolutionary Robotics and SAGA: The Case for Hill Crawling and Tournament Selection." In *Artificial Life III.*, edited by C. G. Langton. 299-326. Reading, MA: Addison-Wesley.

Harvey, I., P. Husbands and D. Cliff. 1993. "Issues in evolutionary robotics." In *Proceedings of the Second International Conference on Simulation of Adaptive Behaviour.*, edited by J. Meyer, H. Roitblat and S. Wilson. Cambridge, MA.: MIT Press.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press.

Jackson, E. R., D. Johnson and W. G. Nash. 1986. "Gene Networks in Development." *Journal of theoretical Biology* 119: 379-396.

Kandel, E. R., J. H. Schwartz and T. M. Jessell. 1991. *Principles of neural science.* New York: Elsevier.

Kauffman, S. 1969. "Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets." *Journal of theoretical Biology* 22: 437-467.

Kauffman, S. and S. Levin. 1987. "Towards a General Theory of Adaptive Walks on Rugged Landscapes." *Journal of theoretical Biology* 128: 11-45.

Kauffman, S. A. 1993. *The Origins of Order*. New York: Oxford University Press.

Kauffman, S. A. and S. Johnson. 1992. "Co-Evolution to the Edge of Chaos: Coupled Fitness Landscapes, Poised States, and Co-Evolutionary Avalanches." In *Artificial Life II.*, edited by C. G. Langton, C. Taylor, J. D. Farmer and S. Rasmussen. 325-370. Reading, MA: Addison-Wesley.

Kitano, H. 1990. "Designing neural networks using genetic algorithm with graph generation system." *Complex Systems* **4** : 461-476.

Kitano, H. 1994. "Evolution of Metabolism for Morphogenesis." In *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems.*, edited by R. Brooks and P. Maes. 49-58. Cambridge, MA: MIT Press.

Kodjabachian, J. and J.-A. Meyer. 1994. "Development, Learning and Evolution in Animats." In edited by

Koza, J. R. 1980. *Genetic Programming*. Cambridge, MA: The MIT Press.

Langton, C. G. 1989. "Artificial Life." In *Artificial Life.*, edited by C. G. Langton. 1-47. Reading, MA: Addison-Wesley.

Matela, R. J. and R. J. Fletterick. 1979. "A topological exchange model for self-sorting." *Journal of theoretical Biology* 76: 403-414.

McGinnis, W. and M. Kuziora. 1994. "The Molecular Architects of Body Design." *Scientific American* **270** 2: 58-66.

Meyer, J.-A. and A. Guillot. 1994. "From SAB90 to SAB94 : Four Years of Animat Research." In *From Animals to Animats 3, Proceedings of the Third International Conference on Simulation of Adaptive Behavior.*, edited by D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson. 2-12. Cambridge, MA: MIT Press.

Miller, G. F., P. M. Todd and S. U. Hegde. "Designing neural networks using genetic algorithms." In *Proceedings of the Third International Conference on Genetic Algorithms*, George Mason University. 1989.

Mjolsness, E., D. H. Sharp and J. Reinitz. 1991. "A Connectionist Model of Development." *Journal of theoretical Biology* 152: 429-453.

Nolfi, S., O. Miglino and D. Parisi. "Phenotypic Plasticity in Evolving Neural Networks." In *First Conference From Perception to Action*, Lausanne. 1994.

Nolfi, S. and D. Parisi. 1991. "Growing neural networks." Technical Report Report PCIA-91-15, Institute of Psychology, C.N.R.-Rome.

Nolfi, S. and D. Parisi. 1993. "Phylogenetic Recapitulation in the Ontogeny of Artificial Neural Networks." Technical Report Report PCIA-18-93, Institute of Psychology, C.N.R.-Rome.

Prusinkiewicz, P. 1994. "Visual Models of Morphogenesis." *Artificial Life* **1(1/2)** : 67-74.

Purves, D. and J. W. Lichtman. 1985. *Principles of Neural Development*. Sunderland, MA: Sinauer Associates.

Raff, R. A. and T. C. Kaufman. 1983. *Embryos, Genes, and Evolution*. New York, NY: Macmillan Publishing Co., Inc.

Sims, K. 1994. “Evolving 3D Morphology and Behavior by Competition.” In *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems.*, edited by R. Brooks and P. Maes. Cambridge, MA: MIT Press.

Sims, K. “Evolving Virtual Creatures.” In *SIGGRAPH ‘94*, 1994.

Thieffry, D. and R. Thomas. “Logical synthesis of regulatory models.” In *Proceedings, Self-Organization and Life: From Simple Rules to Global Complexity, European Conference on Artificial Life (ECAL-93)*, Brussels, Belgium. 1993.

Turing, A. 1952. “The chemical basis of morphogenesis.” *Philosophical Transactions of the Royal Society B* **237** : 37-72.

Walbot, V. and N. Holder. 1987. *Developmental Biology*. New York: Random House.

Wilcox, M., G. J. Mitchison and R. J. Smith. 1973. “pattern formation in the blue-green alga Anabena.” *J. Cell.Sci.* **12** : 707-723.

Wolpert, L. 1977. “Pattern Formation in Biological Development.” *Scientific American* **239** 4: 154-164.

Wuensche, A. “Memory, Far from Equilibrium.” In *Proceedings, Self-Organization and Life: From Simple Rules to Global Complexity, European Conference on Artificial Life (ECAL-93)*, Brussels, Belgium. 1993.

Wuensche, A. 1994. “The Ghost in the Machine: Basins of Attraction of Random Boolean Networks.” In *Artificial Life III.*, edited by C. G. Langton. 465-501. Reading, MA: Addison-Wesley.

Young, D. A. 1984. “A local activator-inhibitor model of vertebrate skin patterns.” *Mathematical Biosciences* **72** : 51-58.

Index