Toward an Evolvable Model of Development for Autonomous Agent Synthesis

Frank Dellaert[1] and Randall D. Beer[1,2]

Department of Computer Engineering and Science[1]
Department of Biology[2]
Case Western Reserve University
Cleveland, OH 44106
Email: dellaert@alpha.ces.cwru.edu, beer@alpha.ces.cwru.edu

Abstract

We are interested in the synthesis of autonomous agents using evolutionary techniques. Most work in this area utilizes a direct mapping from genotypic space to phenotypic space. In order to address some of the limitations of this approach, we present a simplified yet biologically defensible model of the developmental process. The design issues that arise when formulating this model at the molecular, cellular and organismal level are discussed, and for each of these issues we describe how they were resolved in our implementation. We present and analyze some of the morphologies that can be explored using this model, specifically one that has agent-like properties. In addition, we demonstrate that this developmental model can be evolved.

1. Introduction

Our long term goal is the co-evolution of bodies and control systems for complete autonomous agents. The design of autonomous agents is a complex task that typically involves a great deal of time and effort when done by hand. 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. A number of people (Beer and Gallagher 1992; Harvey, Husbands and Cliff 1993) have argued it might be advantageous to use evolutionary methods.

However, most of the current attempts at solving the design problem using genetic algorithms employ some form of a 'direct mapping' between genotype and phenotype. Here, typically there is a one-to-one correspondence between some substring on the genome and an associated parameter or feature in the final design. Examples can be found in (Beer et al. 1992; Lewis, Fagg and Solidium 1992; Harvey et al. 1993). In addition, in the majority of these cases the authors try to optimize a fixed number of parameters in some chosen architecture that a priori determines which designs are possible.

One can identify a number of problems with this direct mapping: (1) The designs to be explored are essentially limited by the chosen architecture, because of the fixed dimensionality. (2) One of the most obvious problems is one of scaling: while evolving small networks works quite well, the approach scales badly for larger networks. (3) In many cases it is desirable that the final design exhibits bilateral symmetry, which is hard to come by using a direct mapping approach.

In contrast, there is no direct mapping between genotype and phenotype in biology. Rather, plant and animal morphologies are the result of a growth process that is directed by the genome. We believe that there are intrinsic properties to this developmental process that, when used in conjunction with genetic algorithms, may enable it to address some of the difficulties with the direct mapping approach: (1) 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 those achievable with direct mapping. (2) There is some hope that a developmental process can reduce the scaling problems as well, as it essentially builds upon previous discoveries in an incremental fashion. (3) Symmetry comes for free in a developmental model.

Apart from promising to address some problems, there are some additional advantages to using a developmental model in its own right: (1) Development naturally provides a way to sample a spectrum of genetic operators, ranging from local hill-climbing operations to long jumps beyond the correlation length of the genetic search space (Kauffman and Levin 1987). (2) With a developmental model, morphologies and behavioral control mechanisms can graciously co-evolve to obtain optimal performance.

This paper describes our first step toward a simplified but biologically defensible model of development that is efficient enough to be used in conjunction with a genetic algorithm. This preliminary model enables us to explore body patterns and relative placement of essential components. After discussing related work is Section 2, in Section 3 we will highlight some of the design issues and tradeoffs that come up when modeling development in general along with the way we have resolved these issues in our particular model. In Section 4 we show the results that can be expected from it and how the model behaves when used in conjunction with genetic algorithms. In Section 5 we analyze the developmental sequence of a simple agent, using that as an example to show the detailed workings of the model.

Finally, in the Section 6 we will discuss it's strengths and weaknesses, along with directions for future work.

2. Related Work

There is already a body of work wherein the authors both understand and appreciate the importance of incorporating a developmental process into the picture. (Wilson 1989) discusses a general representional framework to set the stage for simulations of development. A number of people have implemented some models, and they can be roughly categorized into two groups:

Much work involves some kind of growth-model coding for evolving neural networks: (Kitano 1990) and (Gruau and Whitley 1993) use grammatical encoding to develop artificial neural networks. (Harp, Samad and Guha 1989) try to evolve the "gross anatomy" and general operating parameters of a network by encoding areas and projections onto them into the genome. (Nolfi and Parisi 1991) uses an abstraction of axon growth to evolve connectivity architectures. Most of these models do not aspire to be biologically defensible, however. Also, they have not been applied as such in the area of autonomous agents.

In contrast, a number of other authors have looked at more biologically inspired models of developmental processes: some work is based on the grammar based approach first developed by Lyndenmayer, such as (deBoer, Fracchia and Prusinkiewicz 1992). For instance, (Mjolsness, Sharp and Reinitz 1991) use grammatical rules to account for morphological change, coupled to a dynamical neural network to model the internal regulatory dynamics of the cell. (Fleischer and Barr 1994) have a hard-coded model for gene-expression that they combine with a cell simulation program. Many other biologically realistic models of different developmental processes are found in the theoretical biology literature. However, to our knowledge, none of these more complex models in the second category have as yet been used in conjunction with genetic algorithms.

It is the combination of a biologically defensible model of development with evolutionary methods that we would like to apply to the design of autonomous agents, something that at this point in time has not yet been addressed in the existing literature.

3. Model

In this section we will first give an overview of the principal components of our model. Then, in the subsequent sub-sections, we will raise some of the issues that came up when modeling each of these components. For every issue that we discuss, we will present the way we resolved it when implementing the model, together with a more detailed description of the actual implementation.

3.1. Overview

The developmental process unfolds simultaneously at three different levels, each of which will need to have a counterpart in our model: at the level of the organism, of the cell and at the bio-molecular level. At the topmost level, a single zygote develops into a *multicellular organism* by a complex epigenetic process: eventually, groups of cells will literally stick together and co-ordinate their actions to form tissues and organs that make up the entire organism. This happens because at the *cellular level*, individual cells unfold 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.

This *genetic regulatory network* will be the first principal component of our model. We believe that, for our purposes, 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 would correspond to the existence of a gene product or the expression of some gene. The total state of the network at a given time can then accordingly be viewed as the pattern of gene expression of a given cell at that time: because the state evolves over time this then corresponds to the unfolding of a developmental program in each cell.

The second component consists of a very simple *cellular simulator* to model development at the cellular level. Eventually, every action directed by the genome should first have a consequence at the level of the cell, if it wants 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. One way to do this could be to build a complex, three dimensional model of how biological cells actually work. However, our primary interest is not to mimic the actual biological developmental process, but to extract from it the essential beneficial properties. That is why we have opted for a simple, two-dimensional cellular simulation.

Finally, the last aspect of the model will cover all phenomena at the *organismal level*. During development cells interact continuously: in biological development, cells communicate by touch as well by chemical signals (Walbot and Holder 1987, page 4). This intracellular communication is extremely important, as it can change the pattern of gene expression in the participating cells. Thus, we will have to take this into account in our 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, we were confronted at each step along the way by the trade off between simplicity and biological defensibility, as ultimately the model was to be used in conjunction with the genetic algorithm. Typically, when you want to evolve autonomous agents you use populations of hundreds of individual organisms in parallel. With our developmental model, each of these would consist of from a hundred to a thousand cells, and in each cell a genetic regulatory network would be active. It is obvious that with such numbers you want to keep the model as simple as possible to keep the computational demands feasible. Although many aspects of biological development are important and even crucial for biological life forms, we feel that some of them can be left out in a simplified model without invalidating the results we get.

3.2. Genetic Regulatory Network

As we will model the patterns of gene expression in each cell as the state of a regulatory network, we will have to address each aspect of these networks, i.e. the nature of the elements and the way they interact.

The interactions between the genetic elements

The major issue here is at what level of detail one wishes to simulate the genetic elements and their interactions. 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 it's final gene product presents an opportunity to regulate the expression of that particular gene (Alberts et al. 1991, page 551-556). Do we want to make a distinction between transcriptional control and RNA degradation control and incorporate them as different building blocks in our model ? Chances are that doing so will yield some insight into the detailed workings of these processes, but it will also pose an enormous computational problem to simulate.

In our model we will assume the existence of one type of abstract genetic element and one way in which these elements can influence each other. Although the genetic elements in a biological cell include not only DNA sequences but also regulatory proteins, cell-surface receptors and a whole lot more, we will assume there is only one type. This assumption does also imply that we will model only one way of interaction between these elements, because most of the time the differences between different regulation strategies in the cell come down to differences in the type of players involved. Of course, if you decide on this course of action, you have to make the assumption that the essence of development is not to be found in the details of all the different regulatory mechanisms, but rather the interaction of mutually influencing elements.

The nature of the genetic elements

The genetic elements can be modeled by anything ranging from simple binary to complex quantitative models. One of the choices that must be made is between continuous and discrete state variables, and between continuous or discrete time models. In the literature this choice has been made in a number of ways, resulting in models that use essentially simple binary elements (Kauffman 1969; Jackson, Johnson and Nash 1986), models with multi-level logic (Thieffry and Thomas 1993), dynamical neural networks (Mjolsness et al. 1991) and fully quantitative models.

We have chosen to model the genetic elements as binary elements. Although the more complex approaches are certainly useful, 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 the model in conjunction with genetic algorithms. In addition, simplifying genetic elements to binary variables can be defended on a deeper ground: a lot of phenomena that occur during embryology and the life span of a cell have an on/off quality or 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. Kauffman (1993) presents additional arguments why the major features of many continuous dynamical systems can be captured by a Boolean idealization.

We will model the genetic regulatory networks by Boolean networks

Given all these considerations, we decided to model the genetic regulatory network by a Boolean network, as first pioneered in this context by (Kauffman 1969) and extended by (Jackson et al. 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 networks as used by (Mjolsness et al. 1991). The latter work is more focused on parameter identification of actual biological processes however, in which case this more complicated approach makes sense.

A Boolean network is much like a cellular automaton, but where in the latter the neighborhood of a node is fixed and consists of neighboring cells, there is no such restriction in the former. The basic elements are the N nodes, each with its own associated K-node neighborhood and updating rule. Each node can assume a state of 1 or 0, according to the state of its K inputs at the time it was updated. It is easy to see that there are $(N^K)^N$ possible wiring configurations. Fig.1 illustrates one possible instance of a Boolean network with N=3 and K=2 out of $(3^2)^3 = 729$ possibilities.

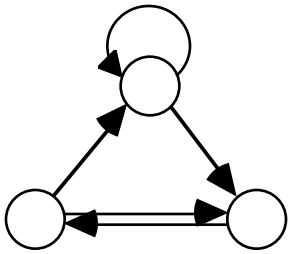

Fig.1. Example of a wiring configuration in a Boolean network with N=3 and K=2.

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. As each entry can contain a value of 1 or 0, there are 2^{2^K} possible updating rules for each node in the network. As an example, one out of the sixteen possible Boolean functions for K=2 would be the logical AND operator, which can be represented by the string <0001> in a lookup table.

The total number of possible Boolean networks with given N and K is then $\left[(N^K)(2^{2^K})\right]^N$, a number that can be very large. For the example of N=3 and K=2, there are $\left[(3^2)(2^{2^2})\right]^3 \cong 3\cdot 10^6$ possible networks.

Some additional issues

We have chosen to update these Boolean networks 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). Then again, they would be useful to examine phenomena like spontaneous symmetry-breaking interactions between cells, that can not occur with lock-step updating.

We 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 (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 actually a small number of genes might be responsible for the regulatory mechanisms within the cell.

The genome that we use in the genetic algorithm is a straightforward description of one such possible Boolean network. Both the connection parameters and the Boolean function are subject to mutation. In some experiments however, constraints can be imposed to explore restricted search spaces. Most of the time we used parameter settings of N=6 and K=2.

3.3. Cellular level

At the cellular level we will have to model all 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, which we will discuss here. We will also touch on the aspects of biological cells that we left out of the model and why we left 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 in large numbers, and yet captures enough of the aspects of a biological cell that makes it work within a developmental process. We think two properties are essential in this respect: it has to have some form of physical extent and it has to be able to undergo division. Granted that this is an extreme simplification of what a real cell actually constitutes: our ultimate goal is the synthesis of autonomous agents, however, not the modeling of biological development, and we think that these two properties suffice for our purposes.

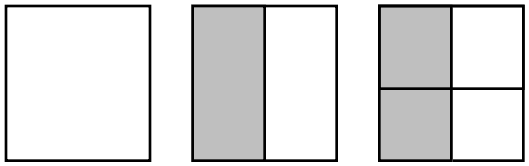

Fig.2. Zygote square dividing two times to yield 4 'cells'.

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.2 for an illustration.

The cell cycle

A simulated cell cycle consisting of two phases, interphase and mitosis, co-ordinates the updating of the Boolean network state and cell division, respectively. In one organism, each cell would have a copy of the same Boolean network constituting the genetic information of that organism. 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.3 the cell cycle is depicted graphically, and we will discuss each phase here.

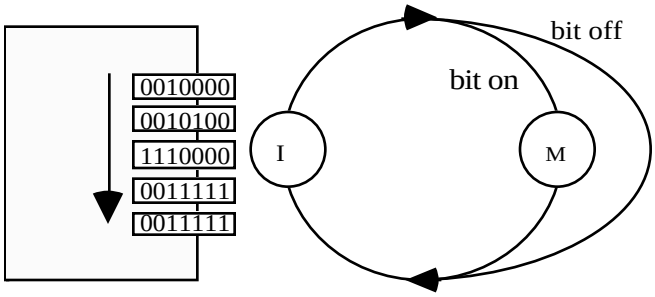

Fig.3. The cell cycles between interphase and mitosis. Mitosis is skipped if the 'dividing bit' is not set.

During the first phase, interphase, the state is synchronously updated until a steady state is reached: a steady state corresponds to a stable pattern of gene expression. We make the assumption that each cell has enough time to reach such a stable state, and when used with the genetic algorithm we will discard any organism whose regulatory network leads to a cyclic pattern. Waiting for a steady state also means that in every cell the number of updates may be different, because the transient behavior of the network depends both on the original state vector as on the new environmental stimuli (see below).

Then, in the second phase, according to the setting of a specific bit the cell either goes through mitosis and divides into two daughter cells, or it stays intact and waits for the next interphase to start. When the cell divides, it's state vector is inherited by the two daughter cells. Note that in the following interphase, unless something in the environment changes or there is some other external interference, that state vector will correspond to a stable pattern and nothing will happen, i.e. that pattern of gene expression will be passed on to the next generation. Also, our 'cell cycle' definition is not entirely comparable with the biological equivalent, as in our model the cell does not necessarily have to go through mitosis.

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 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. 1991, page 930). 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.

In our implementation we have modeled both these mechanisms and we can 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. This simply corresponds to a binary encoding of the cell types, e.g. for N=3 cell type 5 would be represented by the string <101>. In the other, 'master gene' mode, we relate differentiation to the activity of one specific genetic element, with the additional constraint that there should be no conflict between competing cell types. For encoding a cell type 5, we thus need at least N=6, and it would then be represented by the string <100000>. All the results reported in this paper use combinatorial specification, as we have found that it takes considerably longer to evolve the additional mapping between the state of the network and the different 'master genes'.

Our model uses color as an abstraction for cell type. As it is ultimately our goal to synthesize autonomous agents, the final differentiation of a cell will then correspond to it being a sensor, actuator or control-neuron. For the time being, however, it will be sufficient to simulate this by a different color that the cell can take on: it will enable us to demonstrate the different architectures that can be explored using the model. In the combinatorial mode, we assign a color according to the settings of lg(C) specific bits in the state vector, where C is the number of colors. Mostly, C=8 and the three bits used are bit[0], bit[1] and bit[2].

Biological properties we did not include in the model

Although many aspects of biological development at the cellular level are important and even crucial, we feel that some of them can be left out in a simplified model without invalidating our results. 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. We think it is better to start off exploring what is possible with 'simple' intracellular 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 add incorporate more complex mechanisms.

3.4. Organismal level

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

The organism as a collection of cells

The organism itself is a two-dimensional square consisting of many cells. Development starts out with one single square that represents the zygote, which then subsequently divides according to the state of the genetic regulatory network. 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

We will have to address a way to break the symmetry between the very first cells at the early stages of development, otherwise we will end up with a uninteresting, homogeneous collection of cells: because of the deterministic, synchronous updating and as they are all descendants of the same 'zygote', all the cells 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).

We will break the symmetry at the time of the first cleavage by assuming the existence of a 'maternally' imposed a-symmetry 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). We will simulate by flipping a bit of the Boolean network state vector in only one of the two daughter cells. 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 we will explain in Section 6, it was sometimes necessary to provide more spatial clues than only the first cleavage symmetry breaking. Thus, we also provided the cells 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).

Intracellular communication

One of the key elements of the developmental process and consequently of our 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). We have put a lot of thought 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.

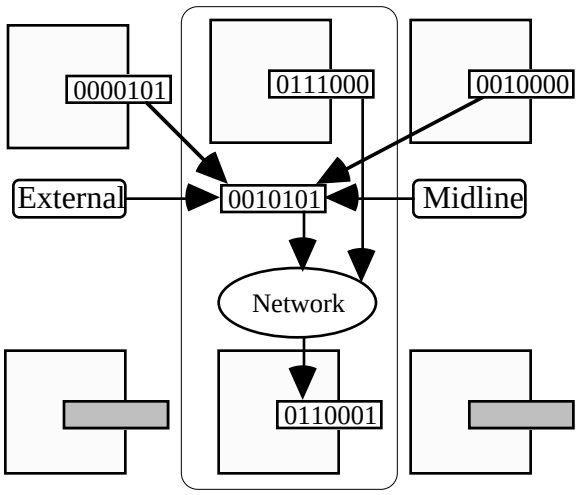

Fig.4. The state vectors of two neighbor cells are ORed together to yield a neighborhood vector that is combined with the cell's state vector to determine the next state.

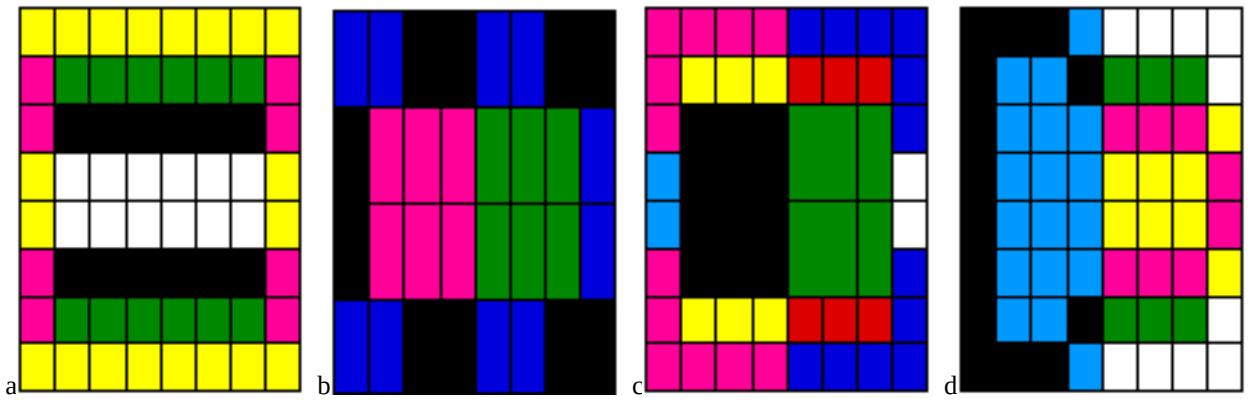

Fig.5. Four different examples that demonstrate the range of organisms we have been able to develop so far.

To implement induction we used a modified version of the Boolean Network: 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.4 shows this arrangement. In our genetic description of the network, a negative connection parameter implies that the corresponding bit in the neighborhood state vector is taken, in stead of from the cell's state vector. Note that an edge to the neighborhood vector can be interpreted as the existence of a cell-surface receptor, sensing the presence of specific chemical agents introduced by cells in its environment.

This implementation implies that our simulation has to keep track of which cells are neighbors. 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. We solved this problem by letting each cell pass on a list of it's 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 our particular implementation allows to easily substitute a more complex (even three dimensional) geometry for the 2-D square one. Indeed, we have already modeled one-dimensional 'string' organisms in this way, and plan to look at the more detailed topological framework model as proposed by (Matela and Fletterick 1979) and recently elaborated on by (Duvdevani-Bar and Segel 1994).

Another type of induction is the influence exercised by the external environment: we modeled this 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.4.).

4. Examples and Evolvability

4.1. Developmental Examples

Fig.5 shows several examples that demonstrate the range of organisms we have been able to evolve until now. Although these are preliminary explorations, mostly found using 'biomorph mode' (sitting down at an X-terminal and selecting the fittest individual according to subjective taste, see (Dawkins 1989)), they nevertheless exhibit interesting features that can conceivably be put to use in the context of autonomous agents.

Fig.5a 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.

We have selected the organism in Fig.5b because it has a segmentation property, as you can discern a bilaterally symmetrical repeat structure at the sides of the organism. Finally, Fig.5c and d represent more complex morphologies, both a-symmetric with respect to the vertical axis and having more detailed patterning at the rostral side.

4.2. Evolvability

Given that our basic goal is to efficiently evolve autonomous agents, one of the things to look at is how the model behaves when used in conjunction with a genetic algorithm. To investigate this, we have devised a generic performance function that maximizes the number of colors, taking care that no color is more represented than any other. Although this particular criterion has no direct relevance to autonomous agents design, it is nevertheless useful to examine the discoveries made by evolution in maximizing this function.

We have found that we can successfully evolve Boolean networks that can steer the developmental model so that the

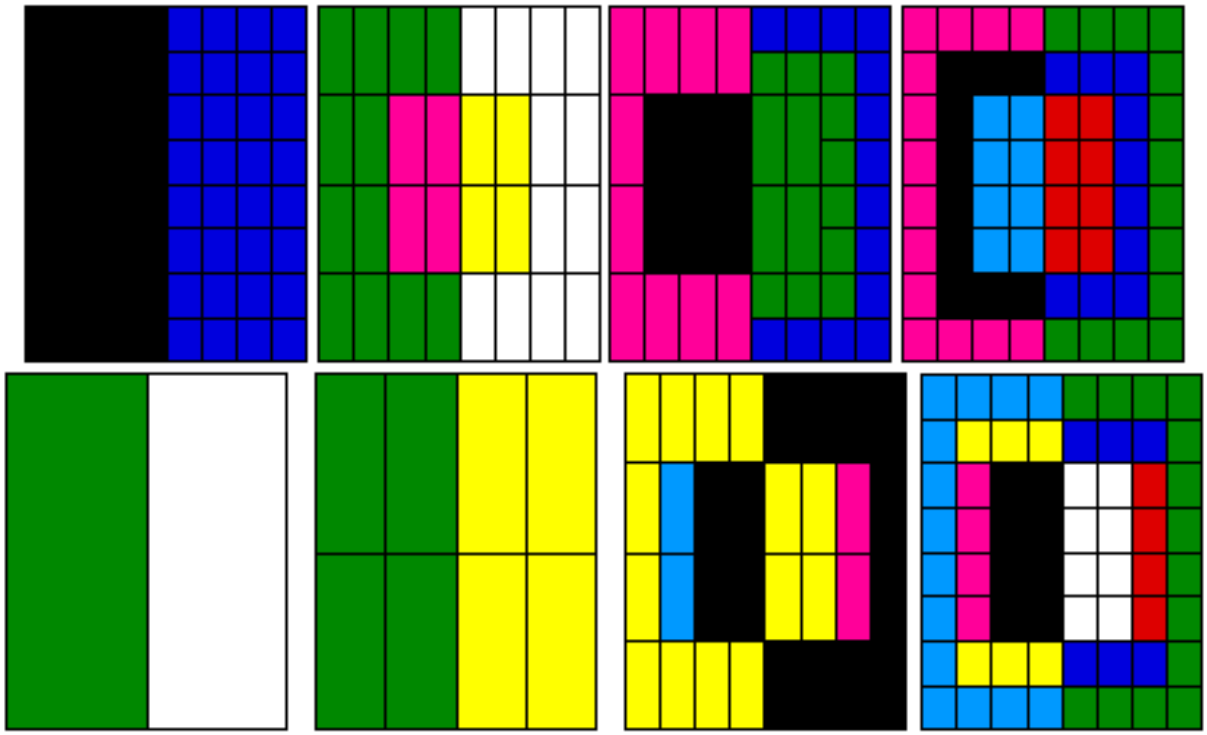

Fig.6. The best individual after each fitness jump during evolution. The respective performance values of these individuals are 45, 170, 210, 1200 and 5600. Of the last individual, the developmental stages are also shown.

fully developed organism optimizes 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 genome (= 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.

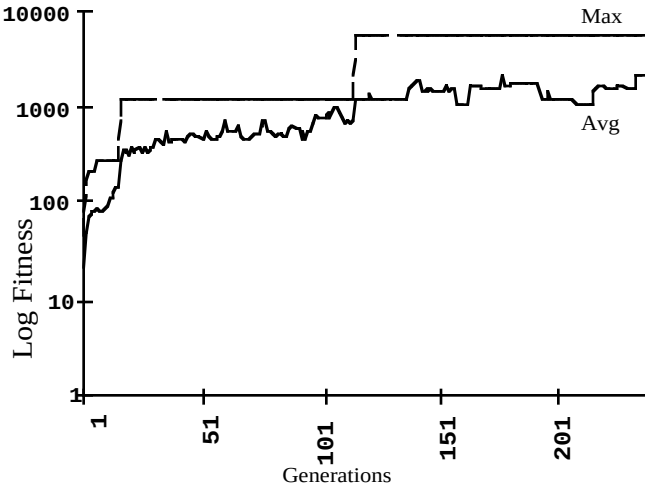

Fig.7. : Maximum fitness and average fitness.

Also, we have observed that 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 for 200 generations, depending on the performance function and the mutation rate. Typically, we used mutation rates of 0.1 and cross-over probabilities of 0.5.

Fig.6 and Fig.7 show the results for a typical run. In Fig.7 we plotted the maximum and average fitness. The GA we used for this experiment used elitist selection, i.e. the best individual is never thrown away, which explains the step-like manner the maximum fitness evolves. In Fig.6 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 our 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, we can immediately see evidence of the a-symmetry we 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 a-symmetry.

All evolved organisms shown are bilaterally symmetrical. This is a direct consequence of how the model is set up: the only a-symmetrical stimulus is our first bit-flip, and the other external stimulus is the environment, which is symmetrically introduced at all sides[1]. Because of the synchronous updating of the networks, no other a-symmetries are introduced. Thus, in this model we get symmetry 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 a-symmetry, the developmental process is able to specify 4 colors. In the further course of the evolutionary process, the previously formed layers themselves provide information for new cells to assume different colors. The next big discovery is six colors, then eventually eight.

The last individual 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. We have shown the subsequent stages of development this individual goes through: as you can readily observe, the steps that development goes through follow the 'discoveries' made in the course of evolution: a-symmetry, external environment, induction. As we have argued in the introduction, 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.

5. Development of a simple "Agent"

We have evolved a simple organism 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 these. For the simple task of chemotaxis we specifically looked 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. The performance function we used tried to minimize the difference between the color-patterns in the fully developed organism and a template that to us represented the features needed in a chemotaxic agent. In Fig.8 the organism, which we have termed 'seeker', is shown along with its developmental sequence. A number coded representation is used for the different cell types. In the last stage of development, with 64 cells, you can observe how the different components are placed: cell type 2, prominent at the right-side corners of the organism, corresponds to sensors, whereas cell type 4 and 1 correspond to actuators and 'neural tissue', respectively. Note that 2 actuator-cells are out of place, but the overall relative placement of the components is quite good.

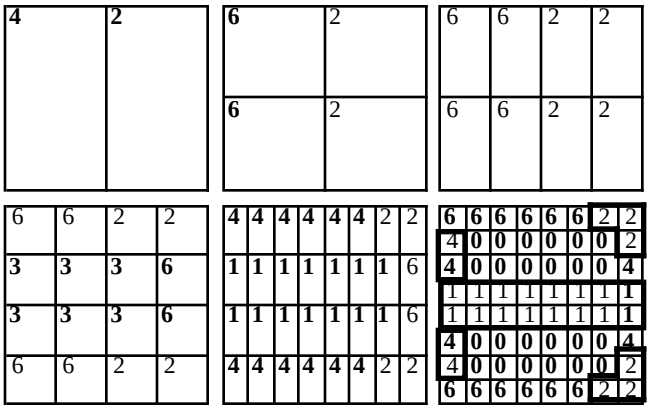

Fig.8. The six consecutive stages of development in the 'seeker' organism, with digits denoting cell types. Bold digits indicate the cell type changed relative to the previous developmental stage.

The 'seeker' organism will serve as an excellent example to illustrate in somewhat more detail just how the developmental model works. To do this we will frequently refer to Fig.8, but this in itself does not say much about the underlying process. 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 we will analyze this information and show what it can tell us about the sequence of events in the development of 'seeker'.

a) node	0 1 0 1 / 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) node	Equivalent 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.9. a) the actual genome of the 'seeker' organism. b) The Boolean functions in a more readable form.

The genome, shown in detail in Fig.9a, specifies the wiring of a Boolean network (Fig.10) and the update rules of each of the nodes (Fig.9b). The 'seeker' organism has network parameters of N=6 and K=2, so the genome consists of 6 update rules and 12 input addresses. Induction from other cells is modeled by a negative address, corresponding to an incoming edge from outside the cell. 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

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

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[2].

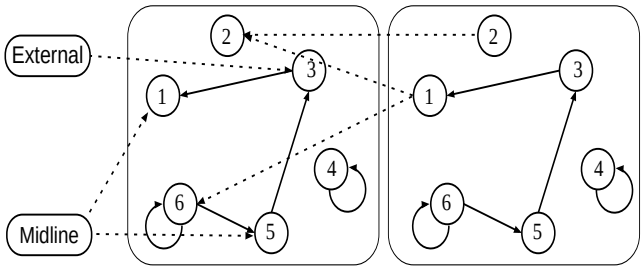

Fig.10. the 'seeker' wiring diagram: the dashed lines represent extracellular inputs. The 'midline' and 'external' have value 1 when the cell in question is on the midline resp. the perimeter of the organism.

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 fitness 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.

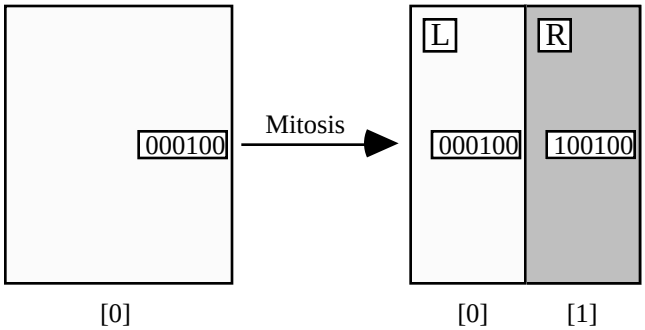

Fig.11. The zygote square divides at least once because we force the dividing bit 4 to TRUE prior to the first mitosis phase. Colors are read from the first three bits in the state vector and are indicated in square brackets.

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 elements 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 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, so that the symmetry is broken. We then have two cells with state vectors 000100 and 100100 respectively, as depicted in Fig.11. 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.

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) 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.12. 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.10. As described in Fig.12, 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.10. We will examine the behavior of cell L in somewhat more detail: bit 6 switches to 1 because its updating rule (6 OR -1, as we know from Fig.9b 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].

[2]In the simulatied 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 neighbourhood vector is always 0.

Because cells inherit steady state vectors after mitosis, some change in the environment is needed to trigger a change in behavior and/or color. After cell L and R go through mitosis, we get 4 cells which we will denote by LT, LB, RT and RB, where T and B stand for Top and Bottom. Because the inherited state vectors represent a steady state of the regulatory network, nothing will happen unless some value changes that triggers a perturbation of this steady state. As it happens, this 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.13, and the colors can be verified by looking at Fig.10: only the left cells have changed color from [4] to [6].

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

Fig.13. The two left cells at stage 2 undergo a transition from color [4] to color [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.10. In stage 3, it looks as if all state vectors remain unperturbed because the colors are unchanged: when looking at the tracefiles of the simulation, we have found that this was indeed the case. To make this difference between 'active' and 'inactive' interface apparent, we have marked the colors in Fig.10 bold when they resulted from a triggered transient, i.e. 'active' interphase.

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 we implemented the midline concept with just that phenomenon in mind (see Model Section).

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.9b 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] resp. [1], for the cells that had color [6] resp. [3]. The picture is more complicated for some other cells, and we will not get into it here.

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

6. Discussion and Conclusions

We have built a simple yet biologically defensible model of the developmental process. We have shown that it can account for a range of morphologies and that it is evolvable, i.e. 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 agent-like morphology.

The work described in this paper is thus fully in line with our longer-term goal to use this developmental model for co-evolving body and control system in autonomous agents. Although to reach that goal, with computational simplicity in mind, we do not intend to modify the model all that much, our initial exploration with the model has raised a number of questions and suggested some issues that may be worthwhile to investigate further:

(1) It would 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 a-synchronously updating Boolean networks.

(2) Many important aspects of biological development that we have excluded from our model provide rich developmental possibilities and could be taken into account.

(3) Instead of binary induction between neighboring cells, it might be advantageous to model gradients of morphogens, as they are hypothesized to underlie both the expression of segmenting genes (Walbot et al. 1987, page 641) as the pattern formation in limbs (Wolpert 1977).

(4) One might want to incorporate a less direct mapping from genome to genetic regulatory network, using instead one that lends itself more naturally to operators that splice out or insert genes, affecting the size of the regulatory network.

Our future work involves, as suggested, extending the model towards actually functioning autonomous agents. We will examine whether it is possible to co-evolve sensor/actuator placement in an organism together with a control structure - or nervous system, if you will - based on non-linear neural networks. To that end, we will associate colors with real functional 'cell-types' like neuron, sensor and actuator, and then let evolved organisms perform some task in a simulated environment, evaluating them on basis of performance of that task. An obvious candidate, and easy to implement, is chemotaxis. Evolving non-linear neural networks for controlling a chemotactic agent has already been done within our research-group (Beer et al. 1992), and it will be of considerable interest to compare the two approaches.

Acknowledgments

We would like to thank everyone in the autonomous agents research group at CWRU for the fruitful discussions that helped shape this work. Special thanks to Leslie Picardo, Hurkan Balkir and Katrien Hemelsoet for their comments on an earlier draft of this paper. This work was supported in part by grant N00014-90-J-1545 from the Office of Naval Research.

References

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

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.

deBoer, M. J. M., F. D. Fracchia and P. Prusinkiewicz. 1992. "Analysis and Simulation of the Development of Cellular Layers." In *Artificial Life II*., edited by C. G. Langton, C. Taylor, J. D. Farmer and S. Rasmussen. 465-483. Reading, MA: Addison-Wesley.

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

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.

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.

Harp, S. A., T. Samad and A. Guha. 1989. "Towards the Genetic Synthesis of Neural Networks." In *Proceedings of the Third International Conference on Genetic Algorithms*., edited by J. D. Schaffer. 360-369. San Mateo, CA.: Morgan Kaufmann.

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.

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

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.

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

Lewis, M. A., A. H. Fagg and A. Solidium. "Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot." In *IEEE International Conference on Robotics and Automation*, Nice, France. 1992.

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.

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

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

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.

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

Wilson, S. W. 1989. "The Genetic Algorithm and Simulated Evolution." In *Artificial Life*., edited by C. G. Langton. 157-166. Reading, MA: Addison-Wesley.

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

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.