A Developmental Model for the Evolution of Complete Autonomous Agents

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

[1] Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213
[2] Dept. of Computer Engineering and Science, Dept. of Biology
Case Western Reserve University, Cleveland, OH 44106
[2] Santa Fe Institute, 1399 Hyde Park Rd, Santa Fe, NM 87501
e-mail: dellaert@cs.cmu.edu, beer@alpha.ces.cwru.edu

Abstract

Development is an important, powerful and integral element of biological evolution. In this paper we present two models of development that can be used to evolve functional autonomous agents, complete with bodies and neural control systems. The first and most complex model is more biologically defensible in its details. It has been used to hand-design a genome for the development of complete agents capable of executing a simple avoidance task. These agents were then incrementally improved through evolution. The second model is simpler and uses a random Boolean network model for the genome and cell state that is somewhat more removed from the biological realm, making it easier to analyze and more amenable to artificial evolution. Using this model, we have successfully evolved complete agents from scratch that are capable of following curved lines.

1. Introduction

In this paper we present a model of development that has been used to evolve functional autonomous agents, complete with a morphological structure and a neural control system. Earlier work that involved a more biologically defensible but more complex model is contrasted with a new and simplified approach that performs surprisingly better.

Development is an important and integral part of biological evolution. Genetic changes are not directly manifested in phenotypic changes, as is often assumed both in population genetics and in most autonomous agent work involving evolution. Rather, a complex developmental machinery mediates between genetic information and phenotype, and this has many consequences. It provides a certain robustness by filtering out genetic changes (i.e. some genetic changes make little or no difference to the final phenotype; there is an equifinality to development). It also provides a natural way to try out a spectrum of mutations, i.e. the same type of genetic mutation can produce anything from no effect to a very large effect in the phenotype, depending on when the affected gene acts during development. Furthermore, development provides a compact genetic encoding of complex phenotypes, allows incremental building of complex organisms, and supports symmetry and modular designs.

For these reasons, there is a growing interest in modeling development (Lyndenmayer and Prusinkiewicz 1989; Wilson 1989; Mjolsness, Sharp and Reinitz 1991; deBoer, Fracchia and Prusinkiewicz 1992; Fleischer and Barr 1994; Kitano 1994). Many ongoing efforts aimed at including simple developmental models in evolutionary simulations can be found in the literature (Belew 1993; Cangelosi, Parisi and Nolfi 1993; Gruau and Whitley 1993; DeGaris 1994; Kodjabachian and Meyer 1994; Nolfi, Miglino and Parisi 1994; Sims 1994; Jakobi 1995).

Much of the latter work has focused on modeling *neural* development, however. But biological bodies and nervous systems co-evolve. Body morphology and nervous systems can constrain and shape one another. Somatic and genetic factors can interact, and this occurs not only during development, but on an evolutionary scale as well. Problems posed by evolution can be solved by a combination of body and neural changes. Allowing both body and nervous system to co-evolve can provide a smoother and more incremental path for substantial changes.

Also, most of this work is highly abstracted from biological development (e.g. using grammars). While there are good reasons for this (familiarity, simplicity, computational speed, emphasis on performance not biology, etc.), too little is currently understood about development to know what are the right abstractions to make. Development completely transforms the structure of the space that is being searched. If we're lucky, this transformation will allow us to evolve interesting agents more easily. But if we're unlucky, we could actually make the search problem *harder*. Because so little is currently understood about the overall 'logic' of development, it is important that we explore many different levels of abstraction to get a sense of the tradeoffs involved. An important aspect of this exploration should be to explore developmental models that are more biologically realistic in their basic structure than the highly abstract models that have currently been explored. That is the aim of this paper.

In the next section we give a brief overview of the general approach that we have adapted to model a developmental process for autonomous agents. It is common to both models we discuss in the paper. Section 3 presents a biologically defensible model of development, complete with a mechanism for the emergence of a nervous system inspired by axonal growth cones. The expressiveness of the model is demonstrated by means of a hand-designed genome, able to direct the development of a functional and complete agent that can execute a simple task in a simulated world. In Section 4, we discuss a simplified model that addresses some of the problems of the earlier model, and show that it can be used to evolve functional agents from scratch. We

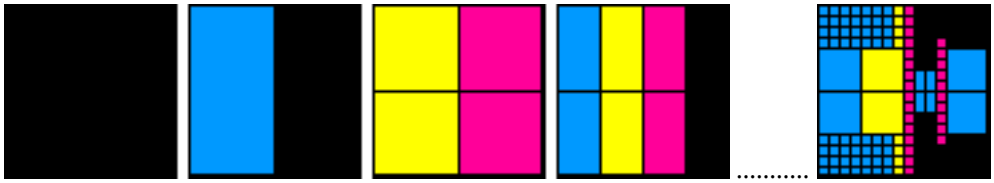

Figure 1: Starting with an 'egg cell' the developmental simulation yields a multicellular square as the adult organism.

show the example of an agent evolved to execute a line following task. Section 5 discusses some of the lessons we learned from this work, and suggests some avenues for further research.

2. Overview of the Developmental Model

In nature it is the performance of an adult organism in its environment that will determine whether its genetic material is propagated. However, in sharp contrast with the model usually assumed in the genetic algorithm literature, genes do not directly specify the traits of an animal. Rather, they specify the developmental sequence by which an animal grows out of a single egg cell to a fully developed phenotype. It is only after this process is complete that specific traits or behavior can be selected for or against[1].

For reasons specified in the introduction and elsewhere (Dellaert and Beer 1994a; Dellaert 1995), we believe that using a developmental model in conjunction with the genetic algorithm (GA) can help us to better evolve autonomous agents. Thus, we have tried to implement this in simulation, where a genetic algorithm will supply us with some genetic material, the genotype, that is then transformed by a developmental model into a fully grown organism, the phenotype. It is the performance of the phenotype that will determine whether its genotype is selected inside the GA.

In particular, we have implemented a model of development for simple simulated organisms that start out as a single cell but 'grow' into multicellular organisms. An example of one such developmental sequence is shown in figure 1. The fully developed agent will then be evaluated on how well it performs a simple task in a simulated world.

In contrast to the complex nature of real biological cells that are able to move and change their shape, our simulated cells are modeled as simple, rigid two-dimensional squares. This choice allowed us to implement cell-division in a particularly efficient way, keeping the computational cost of the simulation acceptable. Indeed, after two rounds of divisions, the resulting cells are again square cells. Our implementation does allow for more complex cell models to be substituted in place of the square cell model should that need arise in the future.

A range of different cell types coexist in any full-grown biological organism. This is so even though each cell possesses the same identical copy of the genome, a sequence of DNA unique to that particular animal. Yet, not all the cells have the same characteristics or behavior. Although the cells contain identical genes, different subsets of genes are expressed in different cells, giving them different properties.

In our model we have implemented a similar arrangement: each cell has an identical copy of a simulated genome inside it, but the subset of the 'genes' active at any given moment determines what type of cell it is and how the cell will behave during development. Exactly how this is implemented is at the core of both models we will present, and will be discussed in detail below.

During biological development, however, the subset of genes expressed in each cell is not static but rather in constant flux. The cell responds to its environment and to the instructions coded in its genome by changing its composition continuously, until it has differentiated fully into one of the 'adult' cell types. In turn, the instructions given by the genome are a function of the state of the cell. Thus, cell state and genome comprise an interwoven dynamical system, a genetic regulatory network, and it is the collective unfolding of the dynamics of many such systems -one for each cell- that constitutes development. On a higher level of abstraction, development can be seen as the sequence of events by which the cells in the body differentiate to perform the various functions inside the animal.

The heart of our developmental model is formed by exactly one such genome-state dynamical system that lives inside each of our model cells. The active subset of model genes inside a cell will be regarded as the cell state, and the possible state transitions from that state are governed by both the genome, the current state and the environment in which the cell finds itself. Qualitatively, this picture is similar to what we see in biological cells, albeit quite simplified in the details. Our developmental simulation can now similarly be viewed as the sequence of events by which the state of the cells differentiate from the initial 'egg cell' state -and from one another- to form a particular cell type arrangement that will suit a particular task.

As explained in the introduction, we are interested in having both a morphological component, i.e. the development of the physical extent of a simulated organism, and a neural component, i.e. how the nervous system of an organism develops. Together, these components should form a complete autonomous agent. To this end, we have associated certain simulated cell types with particular functions, e.g. sensors, interneurons, or actuators. We then also provide a model of how these control components get wired up in a working neural network i.e. the neural developmental component of the model.

This high level specification of the model needs to be complemented with quite a few implementation issues and choices to make it work. We need to specify which states will lead to cells dividing. Since we have chosen to update

[1]This is not entirely true: there will also exist mutations that prevent an organism from developing to maturity.

all the cell states synchronously, we need to break the symmetry after the first division to get interesting dynamical behavior. Some mechanism of intercellular communication must be implemented to make it possible for cells to influence each other's state. Meaning must be assigned to the different possible genes, so that we can interpret state as cell type. And finally, the detailed mechanism for neural development needs to fully specified.

In the following we will present two different implementations of the developmental model: a complex one and a simple one. The complex model is more biologically defensible, but suffers from its complexity. The simple model is further removed from biological reality in its implementation details, but has the advantage that it is computationally more tractable and easier to analyze. We will discuss both models in the next sections.

3. A Complex Model

The first implementation actually models simplified genome and cytoplasm entities inside each cell, and even has model 'proteins' that are produced by the genome and are collected inside the 'cytoplasm'. The set of proteins in each cell determines what kind of events the cell reacts to and which signals it emits while the developmental sequence unfolds. In addition, there is an elaborate model of neural development based on a growth-cone model that detects the presence of proteins in the cells and grows accordingly.

Genome-cytoplasm Model

In the complex model, each cell contains a 'cytoplasm' and a 'genome'. The cytoplasm contains 'proteins', and the proteins present in each cell determine its capabilities, i.e. cells with a different set of proteins can be thought of as having a different cell type. In this regard our model proteins can be thought of as having a similar role as biological proteins. We represent each model protein by a unique integer, and implemented the cytoplasm as a set of integers.

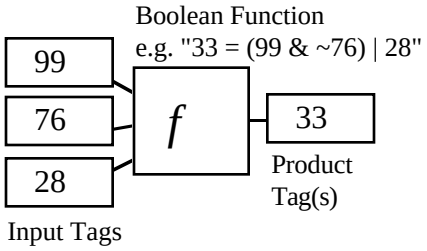

Figure 2: An artificial genome consists of artificial 'operons', one of which is shown here.

The genome consists of a set of 'operons', an example of which is shown in figure 2. As can be seen, each operon is made up of a set of input tags, a Boolean function and a set of output tags. In each time step of the simulation, the input tags determine the input to the Boolean function, and if the output of the Boolean function is evaluated to TRUE, the protein corresponding to the output tag is injected into the cytoplasm. An input tag will give a 1 to the Boolean function if its corresponding protein is present, and 0 otherwise.

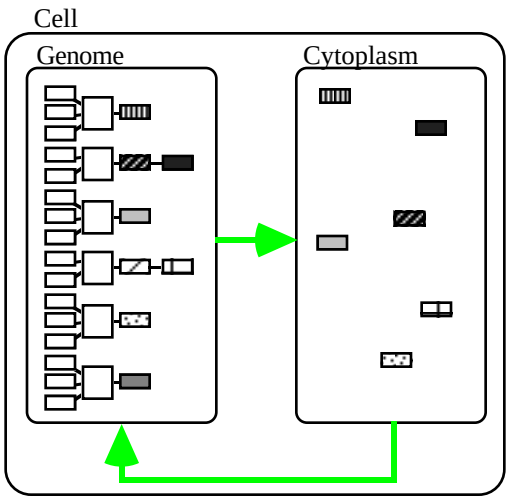

Figure 3: Schematic summary of the cytoplasm-genome model.

Thus, the content of the cytoplasm can be seen as the state of the cell or its cell type, and the genome will determine how this state will change over time. It is the interplay between genome and cytoplasm determines how the cell's cell type changes over time. This interplay is summarized graphically in figure 3.

Development

With the description of the cell's internals in the previous section, we can explain how the developmental simulation will proceed. The organism starts out as a single cell, with its genome given by the GA. The cytoplasm is initially the empty set. Then, for a constant number of iterations, all cells in the organism go through a simulated cell cycle in parallel. This 'cell cycle' consists of two phases. (1) During 'interphase' the cytoplasm is updated by evaluating all the operons in the genome, as explained above. (2) During mitosis, each cell checks for the presence of a special protein (which we will denote *tDividing*, where the leading t stands for 'protein tag') and divides if found.

The simulation ensures that the first cell always goes through division by injecting the *tDivision* protein into the cell's cytoplasm prior to starting the cell cycle. Thus, after the first division we end up with two cells, each with identical cytoplasm and genome that it inherited from their parent cell. This presents a problem: since all the cells obey the same deterministic rules set out above, the remainder of the simulation will yield identical cell types at each time step, and no interesting organisms will emerge.

Therefore we add two additional mechanisms, symmetry breaking and intercellular communication, to ensure that the developmental process exhibits interesting dynamics. Symmetry breaking happens just after the first division event and is implemented by injecting a special protein in only one of the first two daughter cells. This needs to happen only once: from now on these cells' descendants will also differ, as their dynamical trajectories start from different initial conditions. Cell communication, on the other hand, can happen at every cell cycle, and consists of a mechanism that allows one cell to cause the introduction of

a protein into another cell. This is modeled after the biological mechanism of induction, and involves proteins that represent morphogens, receptors and intracellular messenger proteins. The implementation details can be found in (Dellaert and Beer 1994a-b; Dellaert 1995).

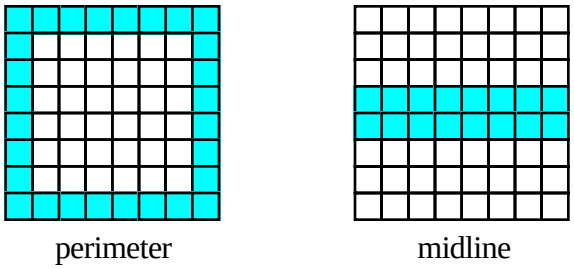

Figure 4: Two special receptors can lead to the differentiation of cells at the perimeter and adjacent to the midline of the organism, respectively.

The model for intercellular communication is also used to introduce two supplementary features that establish a perimeter and midline on the organism that can be detected by the cells, which can lead to local differentiation of cells as seen in figure 4.

A Detailed Neural Developmental Model

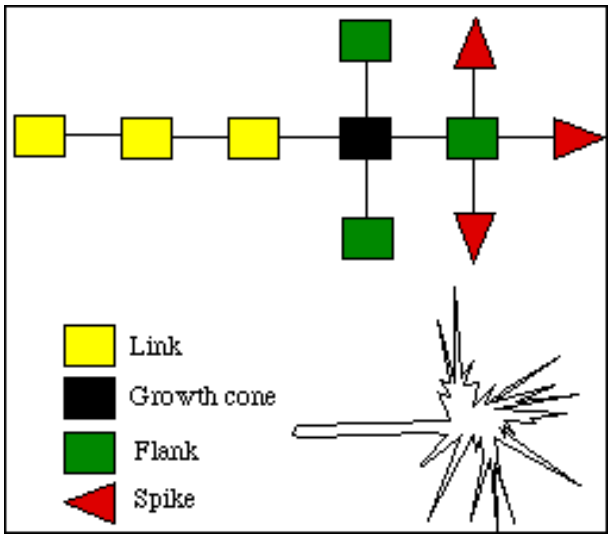

Figure 5: The different axon-element states.

When the development of the agent morphology has settled into a configuration in which divisions no longer occur (but the cell state can still continue to evolve) a control architecture or 'nervous system' develops on top of the arrangement of cells. This happens because specialized cells, i.e. those that express a specific protein (*tAxon*), will send out axons that will innervate other cells, and as such establish a neural network architecture. Only cells expressing the *tTarget* protein will be innervated, and axons will only grow on top of cells that express *tCAM* proteins. This is modeled directly after one postulated mechanism in biological development, in which neurons emit growth cones that can detect the presence of certain molecules (Cellular Adhesion Molecules or CAMs) on the surface of cells and adjust their direction of growth accordingly.

The central feature of this model is the 'growth cone' model, illustrated in figure 5. The black rectangle represents a growth cone. It is linked back to the cell from which the axon originated by 'link' elements, and it sends out 'flanks' that sample the neighborhood ahead by means of 'spikes'. The flank whose spikes detect more *tCAM* proteins will be promoted to a growth cone in the next time step, with the axon splitting in different directions in case of a tie. This fairly intricate finite state model is modeled on the workings of a real biological growth cone, albeit quite simplified.

After the process of axon growth is complete, a dynamical neural network (Beer and Gallagher 1992) is instantiated that connects sensor, interneuron and actuator cells according to the connections made during the neural developmental phase. Time constants and biases of the organism are global and are specified separately in the genome.

Thus, emergence and placement of sensor and actuator cells needs to be coordinated with appropriate neural developmental events to lead to interesting behavior. Sensor cells that did not send out an axon or actuators that were not innervated will have no effect on the agents' behavior.

A Braitenberg Hate Vehicle

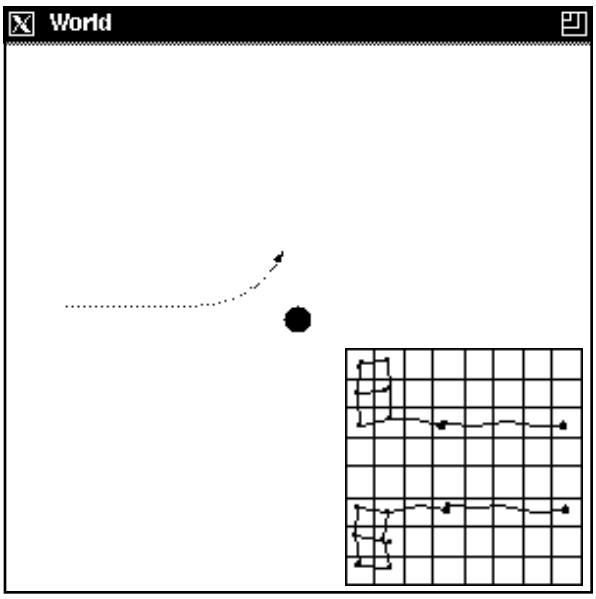

Figure 6: The behavior of the hand-designed Braitenberg Hate Vehicle in a simulated world.

In order to explore the expressiveness of our developmental model, we have hand-designed a genome capable of directing the development of both the body and nervous system of a simple Braitenberg-style “Hate Vehicle”. This agent executes a simple avoidance task in a simulated world. Figure 6 shows the adult form of the hand-designed organism and its behavior on the task; sensors are at the frontal side of the organism (on the right in the figure), and a simple network relays their activation to patches of actuator cells (on the left). Figure 7 shows the expression domains of the different

proteins in the final developmental stage of the organism. A complete analysis of the developmental sequence is beyond the scope of this paper, but can be found in (Dellaert 1995).

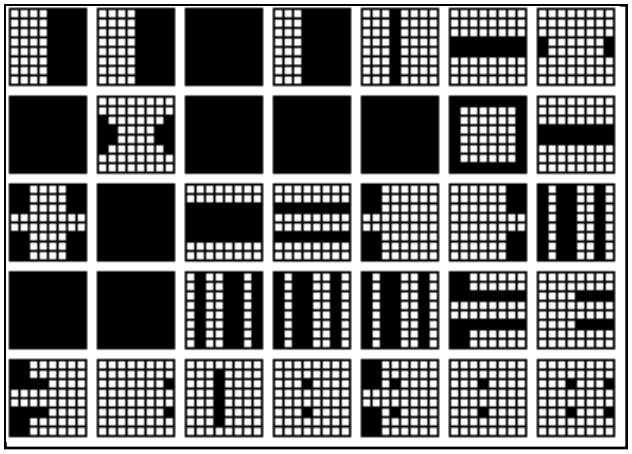

Figure 7: The different 'protein expression domains' in the adult organism. Each square in the matrix shows in which cells each of the 35 proteins is present. For example, the bottom right inset where the *tAxon* protein is expressed, and corresponds to the cells in figure 6 that send out an axon.

It is a demonstration of the power of our model that the initial genetic specification can direct the simultaneous development of both morphology and nervous system, leading to a *complete* autonomous agent. We want to stress that only the genome has been manually specified (i.e. a set of fully specified operons) and that all subsequent development follows from the model without intervention. Note also that no learning takes place in the agent. All the behavior it exhibits is solely a function of its evolved architecture.

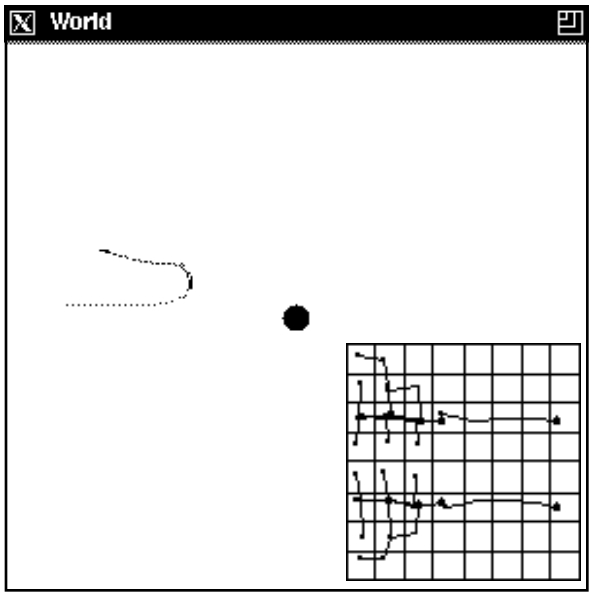

Figure 8: The behavior of the incrementally evolved agent in a simulated world.

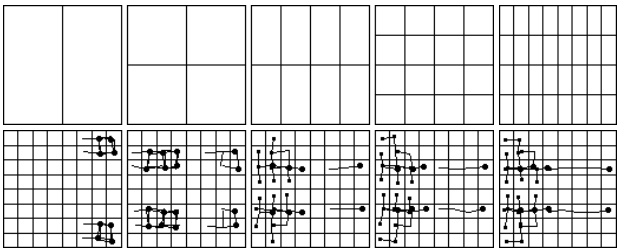

Figure 9: The consecutive steps in the development of the incrementally evolved agent. See text for an explanation.

Although we were unable to evolve such an agent from scratch, we were able to significantly improve its performance through incremental evolution. We started out with the hand-designed genome as a primer for the genetic algorithm, and used a performance function to evaluate each agent's aptness for the avoidance task. The result was an agent that was markedly better at executing the task, as shown in Figure 8. The consecutive steps in the development of this incrementally evolved agent are shown in figure 9. In this figure, the big black dots represent axon-emitting cells, whereas the smaller black squares represent innervation of target cells. Time goes from top left to bottom right, the last square representing the adult organism.

Lessons Learned

Alas, the expressiveness of the developmental model as illustrated in the previous paragraphs is also its downfall. There is a vast space of possible genomes with their associated phenotypes. Many of the protein expression domains need to be tightly coordinated with one another to have a working nervous system emerge. Thus, both the size and structure of the search space make the problem hard. Although the Braitenberg example convinces us that viable organisms (with respect to the task) exist, it does not give us a path to them starting from random genotypes. Although we *were* able to incrementally evolve better performing agents starting with the hand-designed genome (see also Dellaert and Beer 1994b), we have not been able to obtain convincing results when starting evolution from scratch. In addition, the sheer complexity of the model makes it quite hard and error-prone to implement. To cope with both these problems, we have experimented with a drastically simplified model.

4. A Simplified Model

The simplified model uses a random Boolean network (RBN) as an abstraction for the genome, where the state of each cell is equal to the state of its RBN. The topology and rules of the RBN are the same for each cell and can be regarded as the genetic specification of the organism. The model for neural development has been simplified considerably, and is now based on the range and position of the interacting cells. In this section we will discuss each of these aspects in more detail.

The Random Boolean Network Model

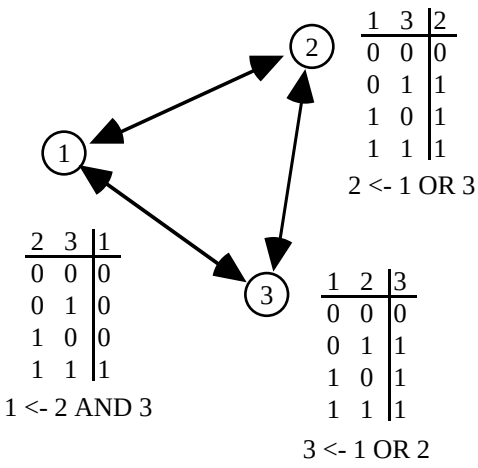

Figure 10: A simple RBN. Boolean functions are specified as truth tables. Example from (Kauffman 1993).

When we first started this work, we used a simpler model than the genome-cytoplasm model sketched above, based on random Boolean networks. RBNs were first thought of as an abstraction for genetic regulatory networks by (Kauffman 1969; Kauffman 1993) and extended by (Jackson, Johnson and Nash 1986) to systems of multiple, communicating networks as needed in the context of development.

A random Boolean network can be represented by a graph, for example the simple RBN of figure 10. In an N node network, each node is defined by K incoming edges, defining a pseudo-neighborhood, and a particular Boolean function[2]. The edges can be recurrent, i.e. nodes can connect to themselves. In addition, each node has an associated state variable assuming a value of 0 or 1. Each node synchronously updates its state in discrete time steps, and the state of a node at time t+1 is the value of the Boolean function using the state of the input nodes at time t.

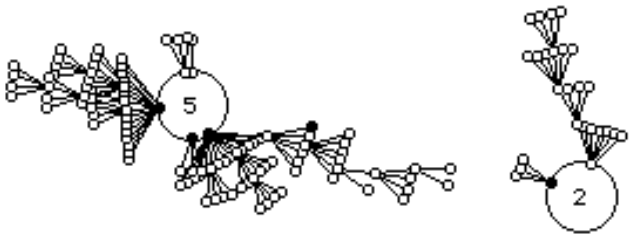

Figure 11: The phase portrait of a random Boolean network with N=7 and K=3. See text for an explanation.

An RBN can then be used as an abstraction of the genetic regulatory network inside a cell. The state of the cell can be equated to the state of the RBN, and each node can be seen as equivalent to a particular 'protein', although we will not use that terminology here. The topology and the particular Boolean functions that specify the RBN will then determine how the state evolves over time, and thus correspond to the 'genome' of the complex model. The incoming edges for each node indicate which other nodes influence its state, and can be seen to correspond to the input tags of an operon in the earlier model. If we complement this with a mechanism for intercellular communication, allowing for edges to occur between cells, we have a very similar picture to the more complex model sketched above.

One nice additional property of RBNs, however, is that their dynamics can be made explicit using phase portraits (Wuensche 1993; Wuensche 1994). This provides us with a powerful tool to analyze the dynamics of development (Dellaert 1995). An example is shown in figure 11. In the figure, each small circle represents one of the 128 states of the 7-node RBN, and the edges between them represent state transitions. The transitions always occur in the direction of the state attractors, in this case state cycles respectively with period 5 and 2.

Development

The developmental simulation unfolds similarly as with the more complex model. Each cell contains an identical copy of the RBN, but the state of the RBN can vary between cells. Instead of a given protein signaling division, a cell will now divide if a prespecified bit in the state vector is set. This bit is set in the 'egg cell' to ensure at least one division event. Symmetry breaking occurs by deterministically perturbing the state (by flipping one bit) of one of the daughter cells resulting from this first division. Intercellular communication is accomplished by calculating neighborhood state vectors that serve as external inputs to the intracell RBNs, perturbing their phase portrait. Details can be found in (Dellaert and Beer 1994a); the final result is a developmental sequence that is qualitatively similar to the ones obtained by the more complex model, but at a considerably reduced computational cost.

Simplified Neural Developmental

To cope with the problems that arose when using the complex model described above, we have implemented a much simplified neural developmental model.

As in the earlier model, the final differentiation of a cell determines whether it will send out an axon or not, and/or whether it is a target for innervation. This is done simply by associating a prespecified node of the RBN with these respective properties. In addition, three bits in the RBN state vector determine whether the cell will be a sensor, an actuator or an interneuron, and one bit specifies whether any innervation will be inhibitory or excitatory.

The development of the neural network is simple and straightforward: each cell that has the 'axon bit' set will innervate all target cells (with the 'target bit' set) within its range. The constants specifying the connection strength and the range at which cells innervate each other are evolved together with the genome, and are identical for all cells.

Thus, unlike before, the developmental process does not need to specify elaborate pathways of CAM molecules on which the axons can grow, but merely needs to make sure

[2]The 'random' refers to the fact that each node can have a different Boolean function and pseudo-neighborhood, in contrast to cellular automata which represent a special case of RBNs.

that it places cells that need to be connected within each other's range. In addition, it can adjust the architecture by specifying for each cell individually the sign of the connection and the cell type. The range and weight factors can be co-evolved with the placement of the cells to obtain the desired behavior.

A Line Follower

Using this simpler model, we have not only evolved agents that execute the earlier avoidance task quite well, but have also succeeded in evolving agents that perform more difficult tasks, in this case following a line made up of circular segments.

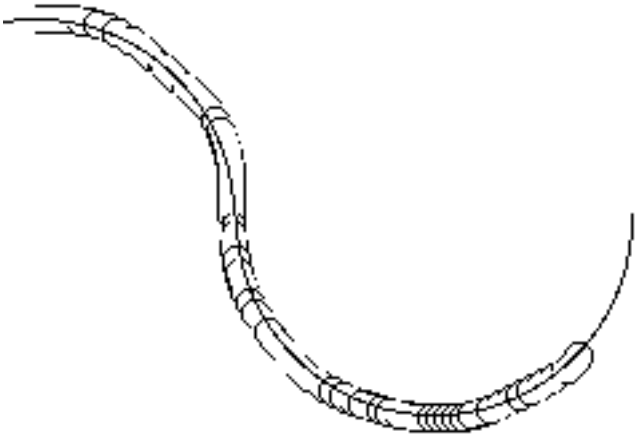

Figure 12: The outline of the line following agent as it executes the task. The agent moves from left to right.

In this experiment, each agent in a genetic algorithm population (Steady state GA with tournament selection, population size 100, mutation rate 2.5%, crossover rate 100%, tournament size 7, run for 10000 evaluations) was put into a simulated world and evaluated on how well it could follow the curve sketched in figure 12, made up of two semicircles. In the neural developmental phase, standard static neurons were used instead of the dynamical neurons of before.

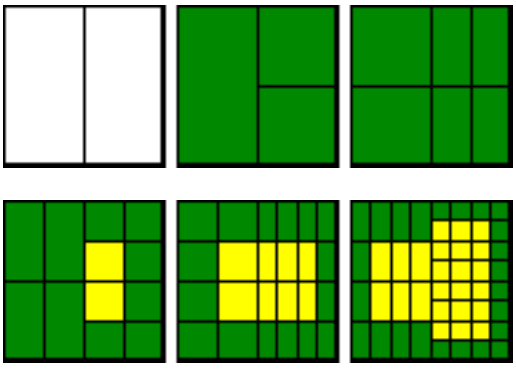

Figure 13: The developmental sequence, from the two cell stage onwards, of the line-following agent.

The evaluation function used was the integrated squared error to an ideal position along the curve. This can be easily calculated since the velocity of each agent is kept constant: only the steering angle varies. The response of each sensor cell in the agent is a Gaussian function of its closest distance to the line. The output of the actuators is averaged for each side, added together and multiplied by a constant factor to become a steering output. In the following paragraph we describe the structure of one such agent, the best of one particular run of the GA, whose behavior in the simulated world is shown in figure 12.

The developmental sequence of the evolved agent is shown in figure 13. In this figure, the dark cells represent sensors while the lighter ones are the actuators. As you can see, development unfolds asymmetrically as not all cells divide an equal number of times. Also, you can see that the sensor cell type is being induced by the perimeter (as shown before in figure 4). No interneuron type cells were present.

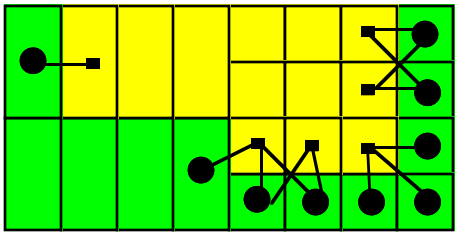

Figure 14: The innervation of actuators by sensors in detail. Only the bottom half of the agent is shown.

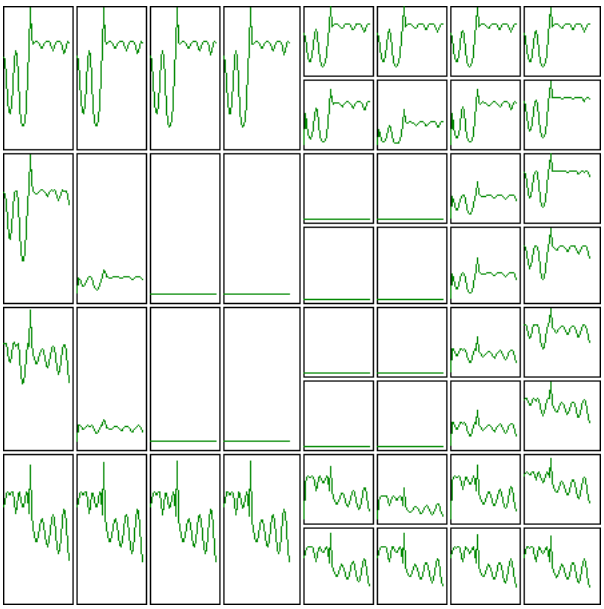

Figure 15: The activation of each neuron/cell during the execution of the task. Time is on the x-axis of each cell, the bottom and top of the cell represent an activation of 0 and 1

If we look at the detailed innervation of the cells, we get the picture sketched in figure 14, where all the connections turned out to be excitatory. This makes sense when you consider how movement is implemented. By this innervation, the agent steers in the direction where it can sense the line. Note that this particular innervation depends on the range factor: with a bigger innervation range more cells would be

innervated. With this picture in mind, it is easy to understand figure 15, which shows the activation of each neuron/cell during the execution of the task. You can see that the actuators not innervated by any sensors remain inactive throughout the whole task.

When evaluating the evolved line-followers on novel lines of different curvatures, their control strategy turned out to be unstable. They were able to follow the line for a while, but they eventually overshot and lost track of the line. Given that they have not been presented with different types of lines during the time that they were evolved it is reasonable that they experience these problems. We are investigating whether we can evolve dynamically stable controllers by exposing them to many different line-following tasks during evolution.

5. Discussion

In this paper we have tried to incorporate a developmental model in the artificial evolution of *complete* autonomous agents, i.e. with a 'body' *and* a 'nervous system'. We presented two models that operated at different levels of abstraction from the biological phenomenon of development, although we feel that both possess many of the properties that are at the core of development, most notably the dynamic interplay between genome and cell state. In addition, they are both able, albeit in quite different ways, to account for the emergence of a nervous system on top of a developed multicellular body arrangement.

The first and most complex model is more biologically defensible in its details. It represents an initial attempt at extending our earlier work (Dellaert and Beer 1994a) to include neural development. We were able to demonstrate its expressiveness by showing that it can account for the development of agents complete with a neural control architecture, capable of executing a simple avoidance task in simulation. We were also able to incrementally evolve better agents starting from that hand-designed agent.

The second and simplified model was adopted in the hope that we could use it to evolve agents from scratch. It used a model for the genome and cell state that was somewhat more removed from the biological example, random Boolean networks. These have the advantage of being computationally cheaper and they lend themselves to analysis using tools from dynamical systems theory. In addition, the second model used a considerably simplified model for neural development. We have used it successfully to evolve agents from scratch that can execute simple tasks. The line-following agent was presented as an example.

We believe that both models can be useful in future research. Both can conceivably be used in the quest to understand more about the logic of development. In the case of the RBN model (and possibly also for the genome-cytoplasm model), it is possible to use analysis tools like phase portraits to visualize what is going on during the simulated developmental process. We might one day be able to use the abstractions that are explored here to visualize the dynamical trajectories being traversed by cells in actual biological development. In addition, we can try and understand more by trying to synthesize observable aspects of development using these models. Some of this has already been explored in (Dellaert 1995). We have, among other things, examined the use of phase portraits to visualize the dynamics of interconnected RBNs (as used in our model for intercellular communication), and we have tried to synthesize pattern formation mechanisms like those found underlying the development of compound insect eyes.

In the domain of autonomous agents, we would like to see whether more complex tasks are within reach of the model. A first simple extension would be to evolve line followers with a stable controller for any type of line, given some smoothness constraint. We have also experimented with introducing learning into the process. If an agent is capable of learning during its lifetime, we might benefit from the Baldwin effect to speed up evolution (Whitley, Gordon and Mathias 1994). Finally, it would be of interest to investigate the behavior of the model under selection of an implicit fitness function, i.e. where agents are placed in a world in which their only task is to outsmart their peers. Here, we expect the complexity of the simulated world to be reflected in the complexity of the agents, and perhaps this brings out the potential of a developmental model better than using explicitly designed fitness functions.

Development is an important, powerful and integral element of biological evolution. It is our hope that explorations such as those we have presented here will contribute to its understanding, both in its own right and as an element of autonomous agent research.

Acknowledgments

Special thanks to James Thomas, Katrien Hemelsoet, and Shumeet Baluja for their helpful comments. This work was supported in part by grant N00014-90-J-1545 from the Office of Naval Research.

References

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

Belew, R.K. 1993. Interposing an Ontogenic Model between Genetic Algorithms and Neural Networks. In *Advances in Neural Information Processing Systems (NIPS) 5*, edited by Hanson, S.J., J.D. Cowan, andC.L. Giles. Morgan Kauffman: San Mateo.

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

de Boer, 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 Langton, C.G., *et al.* Addison-Wesley: Reading, MA.

De Garis, H. 1994. CAM-Brain: the genetic programming of an artificial brain which grows/evolves at electronic speeds in a cellular automata machine. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, IEEE: New York.

Dellaert, F. 1995. Toward a Biologically Defensible Model of Development, Masters Thesis, Case Western Reserve University.

Dellaert, F. and R.D. Beer. 1994a. 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 Brooks, R. and P. Maes. MIT Press: Cambridge, MA.

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

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 Langton, C.G. Addison-Wesley: Reading, MA.

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

Jackson, E.R., D. Johnson, and W.G. Nash. 1986. Gene Networks in Development. *J. theor. Biol.* 379-396.

Jakobi, N. 1995. Harnessing Morphogenesis, Technical Report School of Cognitive and Computing Sciences, University of Sussex.

Kauffman, S. 1969. Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets. *J. theor. Biol.* 437-467.

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

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 Brooks, R. and P. Maes. MIT Press: Cambridge, MA.

Kodjabachian, J. and J.-A. Meyer. 1994. Development, Learning and Evolution in Animats. In *Proceedings of PerAc '94: From Perception to Action, Lausanne*, edited by Gaussier, P.;.N., J.-D. IEEE Comput. Soc. Press: Los Alamitos, CA.

Lyndenmayer, A. and P. Prusinkiewicz. 1989. Developmental Models of Multicellular Organisms: A Computer Graphics Perspective. In *Artificial Life*, edited by Langton, C.G. Addison-Wesley: Reading, MA.

Mjolsness, E., D.H. Sharp, and J. Reinitz. 1991. A Connectionist Model of Development. *J. theor. Biol.* 429-453.

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

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 Brooks, R. and P. Maes. MIT Press: Cambridge, MA.

Whitley, D., V.S. Gordon, and K. Mathias. 1994. Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature - PPSN III*, edited by Davidor, Y., H.-P. Schwefel, andR. Manner. Springer-Verlag: Berlin, Germany.

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

Wuensche, A. 1993. Memory, Far from Equilibrium. In *Proceedings, Self-Organization and Life: From Simple Rules to Global Complexity, European Conference on Artificial Life (ECAL-93)*, edited by Brussels, Belgium (pp. 1150-1159).

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