Université de Toulouse **École Doctorale Mathématiques Informatique et Télécommunications**

Un modèle de développement artificiel pour la génération de structures cellulaires

THÈSE

présentée et soutenue publiquement à l'Université de Toulouse 1 le 17 décembre 2007

en vue de l'obtention du

Doctorat de l'Université de Toulouse

spécialité Informatique

par

Arturo CHAVOYA PENA

Devant le jury composé de :

Directeur de recherche :	Yves Duthen, professeur, Université de Toulouse 1
Rapporteurs :	Jean-Claude Heudin, professeur, Pôle Universitaire Léonard de Vinci André Stauffer, Senior Lecturer, Ecole Polytechnique Fédérale de Lausanne
Examinateurs :	Marc Schoenauer, directeur de recherche INRIA, Université Paris Sud Marie-Pierre Gleizes, professeur, Université Paul Sabatier, Toulouse III Claude Lattaud, maître de conférences (HDR), Université René Descartes Hervé Luga, maître de conférences, Université de Toulouse 1

Remerciements

Je tiens à remercier Yves Duthen pour m'avoir accueilli dans son laboratoire et pour m'avoir proposé un sujet de thèse passionnant. Son soutien et ses conseils ont été inestimables pour l'aboutissement de cette thèse.

Je remercie Jean-Claude Heudin, professeur au Pôle Universitaire Léonard de Vinci et André Stauffer, Senior Lecturer à l'Ecole Polytechnique Fédérale de Lausanne, pour avoir accepté d'être rapporteurs de cette thèse et pour avoir eu la gentillesse de faire partie du jury.

Je remercie également Marc Schoenauer, directeur de recherche à l'INRIA, Marie-Pierre Gleizes, professeur à l'Université Paul Sabatier, Toulouse III, Claude Lattaud, maître de conférences (HDR) à l'Université René Descartes, Paris V et Hervé Luga, maître de conférences à l'Université de Toulouse 1 pour avoir eu l'amabilité de participer au jury.

J'exprime mes remerciements à l'ensemble de l'équipe VORTEX et particulièrement aux membres de l'équipe à l'Université de Toulouse 1 avec qui j'ai pris plaisir à travailler.

Je remercie vivement ma femme Rebeca pour son soutien inconditionnel et sa compréhension pendant toutes ces années. Mes remerciements vont aussi à mes enfants Yael, Astrid et Zair pour la joie de leur présence dans ma vie et pour m'avoir donné la force et la motivation pour faire aboutir ce travail.

Je dédie cette thèse à mon fils
Zair Alejandro

Table des matières

Table des figures

Liste des tableaux

1

Introduction

Résumé

Le Développement artificiel est l'étude des modèles de simulation de croissance cellulaire, avec pour objectif de comprendre comment des structures et formes complexes peuvent émerger d'un petit groupe de cellules initiales indifférenciées. Dans les systèmes biologiques, le développement est un processus fascinant et très complexe qui implique l'exécution d'un ensemble de segments de programme codé dans le génome de l'organisme.

Une des étapes cruciales dans le développement d'un organisme est celle de la génération de la forme générale où l'organisation corporelle fondamentale de l'individu est esquissée. Il est reconnu maintenant que les réseaux de régulation génétique jouent un rôle central dans le développement et le métabolisme des organismes vivants. Des chercheurs ont découvert ces dernières années que les diverses structures cellulaires créées pendant les étapes du développement sont générées principalement par l'activation et l'inhibition sélectives de gènes régulateurs très spécifiques. Sur ce principe, les Réseaux Artificiels de Régulation (RARs) sont des modèles qui tentent d'imiter les réseaux de régulation génétiques trouvés dans la nature. D'un autre côté, des techniques de calcul évolutionnaires ont été amplement utilisées dans une grande gamme d'applications pour faire évoluer un codage (génotype) afin que son image (phénotype) possède des propriétés particulières. Il en a été ainsi en particulier pour faire évoluer des RARs afin qu'ils exécutent des fonctions particulières.

Au cours des années, plusieurs modèles de croissance cellulaire artificielle ont été proposés pour comprendre les mécanismes du processus de développement. Ce travail propose un modèle de développement artificiel qui produit des structures cellulaires par activation et inhibition sélectives de gènes du développement, sous les contraintes des gradients morphogénétiques. La croissance cellulaire est accomplie à travers l'expression de gènes structurels qui sont eux mêmes contrôlés par un RAR qui a été évolué par un Algorithme Génétique (AG). Le RAR contrôle le temps de reproduction des cellules et il détermine également le gène à utiliser pour la reproduction à chaque moment. Parallèlement, des gradients morphogénétiques contraignent les positions sur lesquelles les cellules peuvent se reproduire. Le RAR et les gènes structurels constituent le génome de la cellule artificielle.

Afin de tester les fonctionnalités des RARs trouvés par l'AG, un modèle de croissance cellulaire basé sur le paradigme des Automates Cellulaires (ACs) a été développé dans un premier temps. Cela a permis d'évaluer si les chromosomes de l'AG (qui représentent les modèles des RARs) possèdent la capacité de produire des structures désirées. Les ACs fournissent un cadre excellent pour modéliser des interactions locales qui donnent lieu à des propriétés émergentes dans les systèmes complexes.

Artificial Development is the study of computer models of cellular growth, with the objective of understanding how complex structures and forms can emerge from a small group of undifferentiated initial cells. In biological systems, development is a fascinating and very complex process that involves following an extremely intricate program coded in the organism's genome. To present day, we still marvel at how from a single initial cell, the zygote, a whole functional organism of trillions of coordinated cells can emerge.

One of the crucial stages in the development of an organism is that of pattern generation, where the fundamental body plans of the individual are outlined. It is now evident that gene regulatory networks play a central role in the development and metabolism of living organisms [Dav06]. It has been discovered in recent years that the diverse cell patterns created during the developmental stages are mainly due to the selective activation and inhibition of very specific regulatory genes.

Artificial Regulatory Networks (ARNs) are computer models that seek to emulate the gene regulatory networks found in nature. ARNs have previously been used to study differential gene expression either as a computational paradigm or to solve particular problems [Egg97b, Rei99, Ban03, KB04, STK05, FHB+05]. On the other hand, evolutionary computation techniques have been extensively used in the past in a wide range of applications, and in particular they have previously been used to evolve ARNs to perform specific tasks [Bon02, KLB04].

Over the years, artificial models of cellular growth have been proposed with the objective of understanding the intricacies of the development process [Lin68, Mei82, FB92, Kit94, KB03c]. In this work an artificial development model that generates cellular patterns by means of the selective activation and inhibition of development genes under the constraints of morphogenetic gradients is presented. Cellular growth is achieved through the expression of structural genes, which are in turn controlled by an ARN evolved by a Genetic Algorithm (GA). The ARN determines when cells are allowed to grow and which gene to use for reproduction, while morphogenetic gradients constrain the position at

which cells can replicate. Both the ARN and the structural genes constitute the artificial cell's genome.

In order to test the functionality of the ARN found by the GA, a cellular growth testbed based on the Cellular Automata (CA) paradigm was first developed, so that the GA chromosomes representing the proposed ARN models could be evaluated in their role to produce the desired patterns [CD06b]. Cellular automata have previously been used to study form generation, as they provide an excellent framework for modeling local interactions that give rise to emergent properties in complex systems [de 99, DD05].

2

Objectives

Résumé

Objectif général

L'objectif général de ce travail est de proposer un modèle de développement artificiel qui soit capable de générer des structures cellulaires reposant sur des mécanismes d'auto-organisation et d'interaction avec un environnement artificiel.

Objectifs particuliers

- *Développer un modèle de croissance cellulaire pour produire des formes simples.*
- *Utiliser la combinaison de formes simples pour créer des structures plus complexes.*
- *Appliquer le modèle au problème de la génération de la structure d'un drapeau français (French flag problem).*

2.1 General Objective

To propose an artificial development model that is able to generate cellular patterns through the use of self-organizing properties and interaction with an artificial environment.

2.2 Particular Objectives

- To develop a cellular growth testbed for generating simple shapes.
- To use the combination of simple shapes to create more complex patterns.
- To apply the model to the problem of generating a French flag pattern.

3

Artificial Development

Résumé

Ce chapitre présente un état de l'art des principales recherches relatives au développement artificiel après une brève introduction aux principaux concepts biologiques associés. Un des premiers modèles du processus de la morphogenèse est la Réaction-Diffusion proposée par Turing, où des substances appelées morphogènes réagissent entre elles dans un milieu initialement homogène. Des structures peuvent se développer lors de ces réactions provenant de l'instabilité de l'équilibre homogène, déclenchée par des petites perturbations aléatoires. De leur côté, Meinhardt et Gierer ont proposé un système similaire, appelé le Modèle d'auto activation et d'inhibition latéral, où l'interaction d'un activateur et un inhibiteur peut altérer la symétrie d'un milieu homogène. Ces auteurs ont considéré que la formation des structures nécessite la présence d'une réaction auto-catalysante et d'un inhibiteur de longue porté pour arrêter la croissance infinie de ce processus (inhibition latérale).

Dans les Systèmes de Lindenmayer, ou L-Systèmes, un organisme est produit par un assemblage de structures ou modules discrets répétés. Un L-Système est une grammaire formelle avec un ensemble de symboles et un ensemble de règles de récriture. Les règles sont appliquées de manière itérative en commençant sur le symbole initial. Une des applications principales des L-Systèmes a été dans la modélisation du développement de plantes, en utilisant des visualisations graphiques avancées pour les simulations. D'un autre côté, les modèles des Biomorphs de Dawkins utilisent les micromutations et la sélection cumulative pour tenter d'expliquer l'évolution des structures complexes chez les êtres vivants.

Les modèles d'Embryogenèse artificielle proposés par Hugo de Garis sont parmi les premiers systèmes qui ont exploité l'évolution d'automates cellulaires par algorithme génétique afin d'obtenir des structures cellulaires prédéfinies en 2D. Kitano a été également l'un des premiers chercheurs à expérimenter l'évolution de systèmes de développement artificiel. Cet auteur a en particulier réussi à évoluer des grands réseaux neuronaux en utilisant des AGs.

Fleischer et Barr ont présenté un cadre pour la modélisation et la simulation de la formation de structures multicellulaires en 2D. Ils ont conclu que la combinaison de facteurs chimiques, de forces mécaniques et de mécanismes de contrôle de la lignée cellulaire (cell lineage) est au cœur du développement cellulaire. D'un autre côté, Eggenberger a utilisé une approche évolutionnaire pour étudier la création des réseaux neuraux et la morphogenèse simulée d'organismes en 3D basé sur l'expression différentielle des gènes.

Le modèle de Réseau Booléen Aléatoire (RBA) proposé par Kauffman considère l'interaction de nœuds connectés aléatoirement en utilisant des tables des règles également produites au hasard. Depuis plusieurs années, des modèles associés des Réseaux Artificiels de Régulation (RARs) ont également été proposés. Ce chapitre en présentera plusieurs possédants des concepts communs tels que génomes, gènes, sites

promoteurs, sites régulateurs, facteurs de transcription, séquences de correspondance, et degré de correspondance. Quelques-uns de ces modèles présentent trois types de comportements, qui peuvent être classés comme statique, dynamiquement structuré et chaotique.

Le Modèle de Développement Évolutionnaire (MDE de Kumar et Bentley) est initialement un système d'étude des processus de développement des organismes multicellulaires et leur application potentielle à l'informatique. L'EDS contient l'équivalent de beaucoup d'éléments clefs dans le développement biologique tels que : cellules, cytoplasme cellulaire, parois cellulaires, protéines, récepteurs, facteurs de la transcription et gènes. L'EDS contient deux types de génomes : le premier assure l'initialisation des cellules et le deuxième contrôle la croissance cellulaire. On peut citer également le système artificiel de développement cellulaire METAMorph où les règles morphogénétiques sont introduites par l'utilisateur dans un outil de simulation.

Enfin, le problème académique de la génération de la structure d'un drapeau français (French flag problem) est présenté avec un état de l'art des principaux travaux informatiques sur ce sujet.

This chapter covers the main research areas pertaining to artificial development after a short introduction to biological gene regulatory networks and their relationship with development. In the sections that follow, the work more directly related to the model presented in this thesis will be reviewed in more detail.

3.1 Development and Gene Regulatory Networks

The mechanism of biological development can be viewed conceptually as consisting of a series of concentric layers [Dav06]. On the outer layer, development is achieved through the spatial and temporal regulation of expression of myriads of genes coding for all the different proteins of the organism, which catalyze the creation of other constituents. A deeper layer is characterized by a dynamic progression of regulatory states, defined as the presence and activity state of the set of regulatory proteins that control gene expression. Finally, at the core layer is the genomic machinery consisting of all the modular DNA sequences that interact with the regulatory proteins in order to interpret the regulatory states. These DNA sequences are known as *cis*-regulatory elements (*cis* is Latin for "this side of"), as they refer to regulatory elements usually located on the same DNA molecule as the genes that they control. *Cis*-regulatory elements are the target of active diffusible proteins known as transcription factors or *trans* (Latin for "far side of") elements, which define the regulatory state.

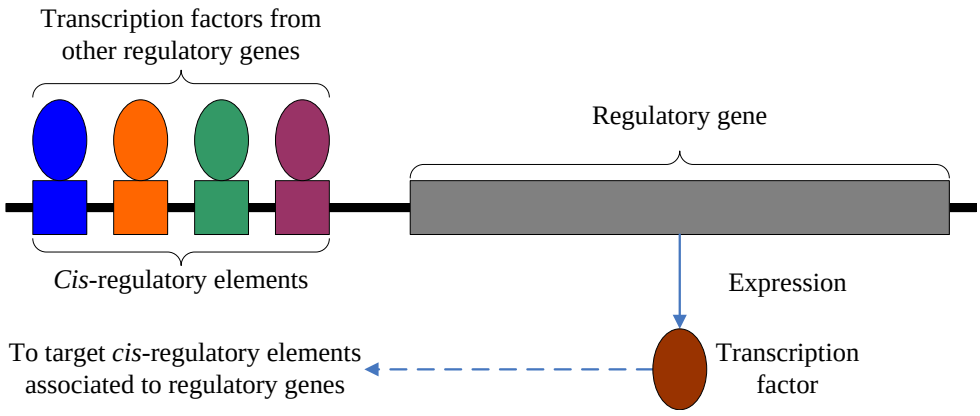


FIG. 3.1 – Relationship between *cis*-regulatory elements, transcription factors and regulatory genes.

Cis-regulatory elements read the information contained in the regulatory state of the cell, process that information, and interpret it into instructions that can be used by the cellular biochemical machinery to express genes contained in the genome. The term "regulatory genes" refer to those genes encoding the transcription factors that interact with *cis*-regulatory elements. Figure 3.1 shows a simplified illustration of the relationship between *cis*-regulatory elements, regulatory genes and transcription factors. Transcription factors bind to specific *cis*-regulatory elements, and this interaction can enhance or inhibit the expression of the associated regulatory gene into a transcription factor, which can in turn interact with its target *cis*-regulatory element on other genes.

The spatial and temporal expression of regulatory genes is central to development, as they determine to a great extent the fate and function of all cells in the developing organism. Developmental control systems take the form of gene regulatory networks (GRNs) ; when genes in a GRN are expressed, they produce transcription factors that can affect multiple target genes (through their associated *cis*-regulatory elements), which can in turn express transcription factors that affect their target genes. Each regulatory gene can have multiple inputs from other regulatory genes and multiple outputs to other regulatory genes. Thus a regulatory gene can be viewed as a node in a network of interactions.

The periphery of a developmental GRN is defined by the absence of outputs to other

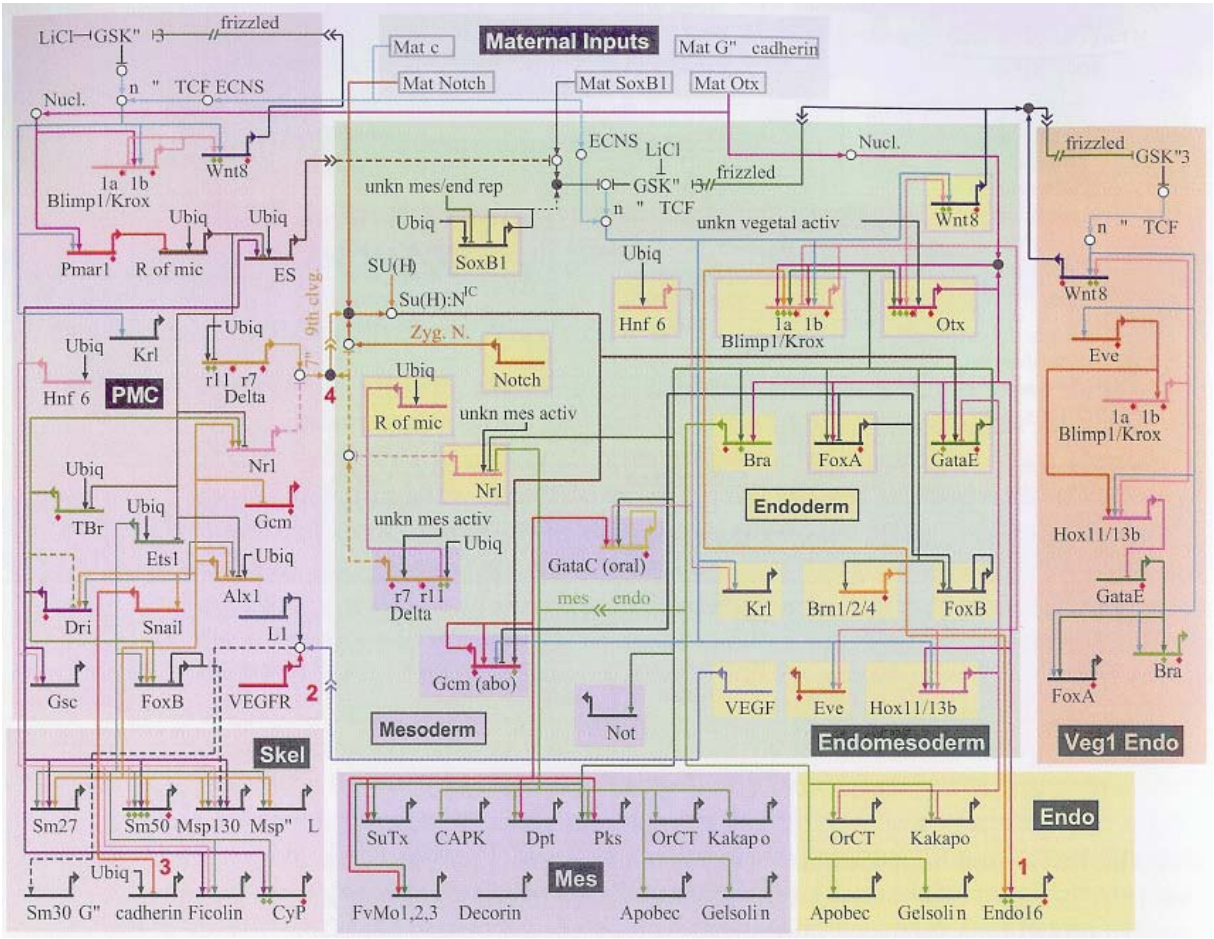


FIG. 3.2 – GRN for the endomesoderm specification at the 30 hours of development of the sea urchin embryo. Short horizontal lines represent a node in the network with arrows indicating the direction of interactions. (Taken from [Dav06].)

genes in the network, i.e. transcription factors that affect other regulatory genes. We mainly find at the outskirts of the GRN the sets of developmental genes that code for proteins that lead to cellular differentiation. Actual developmental GRNs are extremely complex systems that involve other elements not shown in Fig. 3.1, such as intracellular and intercellular signaling molecules, cellular receptors and lineage proteins that are present in one cell type and not in others.

Biological development is evidently an extremely complex process that involves interactions both in time and space of cells in the growing organism. To give an idea of the complexity of developmental GRNs, Fig. 3.2 presents a network corresponding to current

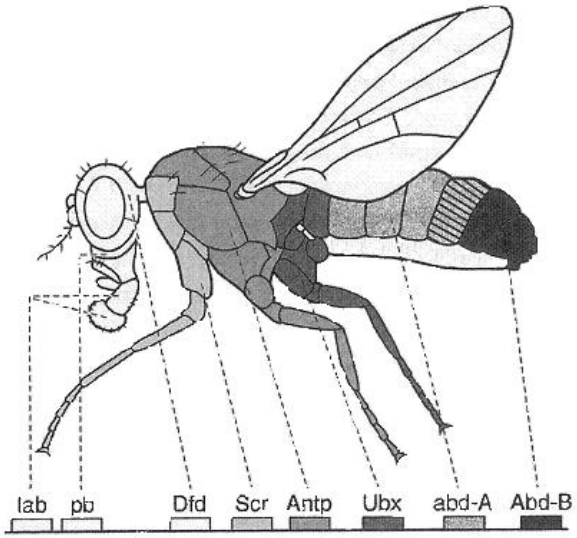


FIG. 3.3 – Regulatory genes of the fruit fly *Drosophila melanogaster*. Each gene controls development of different body regions along the body axis. (Taken from [Car06].)

knowledge on the endomesoderm specification of the sea urchin embryo at 30 hours of development. This GRN is considered incomplete and it is the subject of ongoing research with new nodes and relations being added as they are discovered.

Much of the knowledge on the development of organisms has been obtained from studying animals that are easy to maintain and reproduce in captivity. One of the most studied species in genetic and developmental research is the fruit fly *Drosophila melanogaster*. Over the years, researchers have been able to identify a number of proteins directly involved in the development of *D. melanogaster*. Figure 3.3 shows eight regulatory genes that are responsible for the development of specific body segments on this insect. Mutation of these genes severely affects the development of the associated body segment.

Researchers continue to elucidate the mechanisms underlying development at the molecular level and much work on the subject remains to be done. However, it is increasingly evident that GRNs play an essential role in the development of multicellular organisms, from the simplest to the higher species of plants and animals.

3.2 Reaction-Diffusion Systems

It is usually attributed to Turing the founding of modern research on artificial development. He suggested in his seminal article on the chemical basis of morphogenesis [Tur52], that an initially homogeneous medium might develop a structured pattern due to an instability of the homogeneous equilibrium, triggered by small random perturbations.

Using a set of differential equations, Turing proposed a reaction-diffusion model where substances called morphogens, or form generators, would react together and diffuse through a medium, which could be a tissue. In this model, one of the substances is autocatalytic and tends to synthesize more copies of itself. At the same time another substance also promotes synthesis of the first substance, but the latter inhibits the synthesis of the former. One key element of the model is that the two substances have very different diffusion coefficients, with one of them diffusing much more rapidly than the other. The system can be fine-tuned with the proper parameters such that at some point the slightest disruption in the equilibrium can be amplified and propagated through the medium generating unpredictable patterns.

For simplification purposes, Turing built his model using a few elemental cellular structures. He considered the cases of an isolated ring of cells, a hollow sphere of cells, and a 2D single layer of cells. He was particularly interested in investigating what caused the initial instability that led to the formation of patterns. He observed that there were six main patterns in the distribution of morphogens. One of the most interesting was the appearance of stationary waves on a ring of cells. He suggested that this could explain certain radial patterns that appeared during the morphogenesis of some organisms. Using the other cellular structures, he demonstrated how the gastrulation process could be generated in a sphere of cells by the reaction-diffusion mechanism. Other patterns, such as dappling, could be generated in a single layer of cells, which could account for the skin patterns seen in many animals.

Even though his model was based on an oversimplification of natural conditions, Turing

succeeded in demonstrating how the emergence of a complex pattern could be explained in terms of a simple reaction and diffusion mechanism using well-known physical and chemical principles.

3.3 Self-Activation and Lateral Inhibition Model

Experiments with biological specimens have demonstrated that development is a very robust process. Development can continue normally even after a substantial amount of tissue from certain parts has been removed from an embryo. However, there are small specialized regions that play a crucial role in the organization of the development process. Such organizing regions are usually very small and they control pattern generation in the surrounding tissues. When these organizing regions are transplanted to other parts of the embryo, they start to generate the structures that they would normally form in the original region [CGW04, Car06].

In order to explain the long range effect of these small organizing regions on the larger surrounding tissue and the robustness of their influence even after induced interferences, Wolpert introduced the concept of "positional information", whereby a local source region produces a signaling chemical [Wol69, Wol81]. This theoretical substance was supposed to diffuse and decay creating a concentration gradient that provided cells with information regarding their position in the tissue.

Nevertheless, the problem remained as to how a local differentiated source region could be generated from a seemingly homogeneous initial cluster of developing cells. Even though many eggs have some predefined structure, all the patterns developed after a number of cell divisions cannot initially be present in the egg. A mechanism must exist that allows the emergence of heterogeneous structures starting with a more or less homogeneous egg.

Pattern generation from an almost homogeneous condition can often be observed in the inanimate world. Dunes, for example, can form from an initially homogeneous sand

surface under the influence of wind. A small initial ripple in the sand can be amplified (positive feedback) until bigger and bigger sand deposits are formed.

Gierer and Meinhardt proposed that pattern formation was the result of local self-activation coupled with lateral inhibition [GM72, Gie81, Mei82]. In this model, which has some resemblance to Turing's model, a substance that diffuses slowly, called the activator, induces its own production (autocatalysis or self-activation) as well as that of a faster diffusing antagonist, the inhibitor. These authors suggest that pattern formation requires both a strong positive feedback (autocatalysis) and a long-ranging inhibitor to stop positive feedback from spreading indefinitely (lateral inhibition).

The concentration of the activator and the inhibitor can be in a stable state, since an increase in the activator concentration can be compensated by a corresponding increase in the inhibitor concentration, bringing the activator concentration back to the initial concentration. This equilibrium is nonetheless locally unstable. A local increase in the activator concentration will continue to increase by autocatalysis as the inhibitor diffuses more rapidly than the activator into the surrounding area. The smallest random fluctuations are sufficient for breaking the homogeneity so that pattern generation can be initiated.

One possible interaction between the activator a and the inhibitor h is suggested by the authors with the definition of the following equations :

$$\frac{\partial a}{\partial t} = \frac{\rho a^2}{h} - \mu_a a + D_a \frac{\partial^2 a}{\partial x^2} + \rho_a \tag{3.1}$$

$$\frac{\partial h}{\partial t} = \rho a^2 - \mu_h h + D_h \frac{\partial^2 h}{\partial x^2} + \rho_h \tag{3.2}$$

where t is time, x is the spatial coordinate, D_a and D_h are the diffusion coefficients and μ_a and μ_h are the decay rates of a and h, respectively. Parameter ρ is the source density that expresses the ability of cells to perform autocatalysis. A small activator production

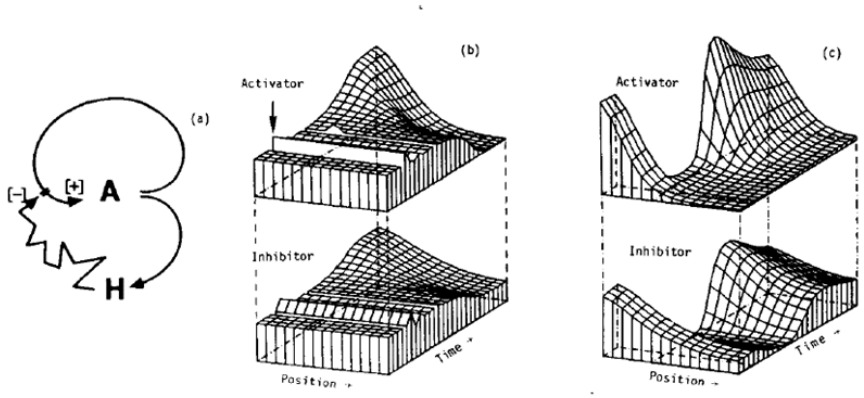


FIG. 3.4 – Pattern generation and regeneration. (a) Reaction diagram (A = Activator, H = Inhibitor) ; (b) Pattern generation ; (c) Pattern regeneration after perturbation. (Taken from [Mei82].)

ρ_a can start the disruption of the homogeneous condition at low activator concentrations.

For the formation of stable patterns in this model, the diffusion of the activator has to be much slower than that of the inhibitor, i.e. $D_a << D_h$. Furthermore, the activator must have a longer time constant than the inhibitor, $\mu_a < \mu_h$, otherwise oscillations will be produced [Mei03].

These equations were approximated by difference equations for discrete cells and used for computer simulations. Figure 3.4 shows a simulation of pattern generation and regeneration in a linear array of cells. A reaction diagram is shown in Fig. 3.4(a). Activator A catalyzes both itself and inhibitor H, while the latter diffuses more rapidly than A and suppresses production on long ranges. Figure 3.4(b) shows how a concentration pattern is formed over time starting with a homogeneous concentration of activator and inhibitor [Mei82]. The arrow in Fig. 3.4(b) shows where a uniform increase in the activator along the cell array has no effect on the homogeneous distribution of both types of molecules, since an equally uniform increase in inhibitor concentration suppresses the increase in activator concentration. However, a small local increase in the activator concentration, or even a random fluctuation, cannot be compensated by the inhibitor, as the latter diffuses more rapidly into the surroundings. In the meantime, the activator concentration increases

steadily until a stable state is achieved, where the production, and the diffusion and decay of the activator reaches an equilibrium. The pattern thus obtained is very robust, as it has self-regulating properties. In Fig. 3.4(c), after a stable concentration pattern is formed, most of the activator is manually depleted in the simulation. With the diminished catalytic action of the activator, the inhibitor concentration decreases by decay, but then the remnant activator molecules self-activate to reach again a stable condition after a number of time steps.

These simulations suggest how in an undifferentiated medium, a small perturbation could be amplified and then disrupt the homogeneous state. For instance, in an apparently homogenous egg, a small perturbation in the activator concentration at one end of the egg could lead to the accumulation of activator molecules at this end. The activator molecules could have other properties such as a signaling role that could induce cells near this zone to differentiated into the head (or tail) of the developing organism.

Gierer and Meinhardt have proposed many other examples of biological systems where their model could be applied [Mei82, Mei96, MG00, Mei03]. One well-known application of their model is in the simulated generation of shell patterns in seashells [Mei98]. With the appropriate parameter values, these simulations generate patterns that are very close to those found in the shells of certain species of seashells.

These results suggest how a relatively simple mechanism of coupled biochemical interactions can account for the generation of very complex patterns. The components of the model are based on reasonable assumptions, since mutual activation and inhibition of biochemical substances and molecular diffusion actually exist in the real world. In recent years, molecular biology and genetics experiments have given support to many elements of the model. Possibly the weakest assumption the authors made for the model was the use of simple diffusion for cell signaling. Modern biological techniques have shown that intercellular communication is indeed very complex and it usually involves the expression of ligand-specific receptors at the cell's surface and intracellular transportation of certain

signaling molecules to the cell's nucleus. Nevertheless, the logic behind the model seems to be sound.

3.4 Lindenmayer Systems

Lindenmayer systems, or L-systems, were originally introduced as a mathematical formalism for modeling development of simple multicellular organisms [Lin68, Lin71]. The organism is abstracted as an assembly of repeating discrete structures or modules. The formalism is independent of the nature of the module, which can be an individual cell or a whole functional structure such as a plant branch. A module is represented by a symbol from an alphabet, and the symbol represents the module's type. Additionally, there can be parameters associated to a symbol in order to define its state, in which case the formalism is extended to what is known as Parametric L-systems [Han92].

An L-system is a formal grammar with a set of symbols and a set of rewriting rules. The rules are applied iteratively starting with the initial symbol. Unlike traditional formal grammars, rewriting rules are applied in parallel to simulate the simultaneous development of component parts of an organism.

L-Systems were initially conceived to derive theoretical results applicable to biological development. However, with the introduction of state-of-the-art computer graphics techniques and the widespread use of computer equipment, L-Systems were used to program and dynamically visualize development systems [Smi84, PHM97].

One of the main applications of L-systems has been in the modeling of the development of higher plants [PL90]. The modeling does not take place at the cellular level. Instead, it is based on a modular construction of discrete structural units that are repeated during the development of plants, such as branches, leaves and petals [Pru93, Pru97]. Initial models did not consider the influence of the environment on development. However, as organisms in nature are an integral part of an ecosystem, an extension to the modeling

framework that considered interaction with the environment was introduced [MP96].

The use of L-Systems has been extremely fruitful in modeling the development of organisms at a high structural level. Implemented models of plant development that use L-systems are visually striking because of their resemblance to growth seen in real-life plants and trees.

3.5 Biomorphs

Richard Dawkins' well-known Biomorphs were first introduced in his famous book "The Blind Watchmaker" to illustrate how evolution might induce the creation of complex designs by means of micro-mutations and cumulative selection [Daw96]. Dawkins intended to find a model to counteract the old argument in biology that a finished complex structure such as the human eye could not be accounted for by Darwin's evolution theory.

Biomorphs are the visible result of the instructions coded in a genome that can undergo evolution. The original genome consists of nine genes coded as integers, where the first eight genes determine the length and branching direction of growing lines in a 2D plane, whereas the last gene controls the depth of branching. Dawkins introduced a constraint of symmetry around an axis so that the resulting forms would show bilateral symmetry, as in many biological organisms.

The construction algorithm that grows biomorphs from their genome is recursive in nature, as Dawkins considered that actual embryological processes could to a large extent be considered recursive. The idea being that the shape of an adult individual emerged after a number of local cellular interactions in the whole developing body and that these effects consisted of simple divergencies such as binary cellular divisions. Figure 3.5 shows the recursive construction of a basic biomorph with a recursion depth of 4.

Initially Dawkins thought that the forms produced would be limited to tree-like structures. However, to his surprise, the forms generated were extremely varied in shape and

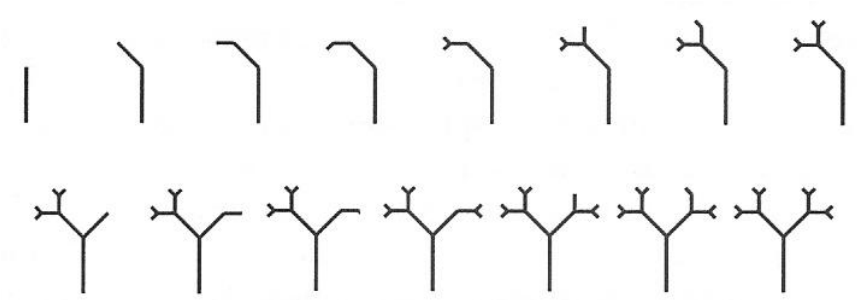


FIG. 3.5 – Construction of a biomorph with a recursion depth of 4. (Taken from [Ren02].)

detail. There were biomorphs that roughly resembled insects, crustaceans or even mammals.

This author proposed next an "interactive" evolutionary algorithm, where the user played the part of the selection force. The algorithm implementation presents the user with an individual biomorph and the eighteen biomorphs that result from the addition of subtraction of one unity (micro-mutation) on each of the nine genes in the genome. Initially the user has to decide which form he/she wants to evolve, such as a spider or a pine tree, and in each step of the algorithm he/she chooses the biomorph that best resembles the target form (cumulative selection).

The algorithm was further improved by adding genes that could activate or disable the symmetrical growth of the generated forms in the 2D plane, both horizontally and vertically. The algorithm was also extended to take into account the segmentation process, which is considered one of the greatest innovations in biological evolution. Segmented bodies are present in at least three of the major phyla : vertebrates, arthropods and annelids. Figure 3.6 shows some of the biomorphs obtained using an implementation of this algorithm.

Dawkins showed with his models that the evolution of complex structures was indeed feasible in a step by step manner by means of the cumulative selection of the individual that best approached the final structure.

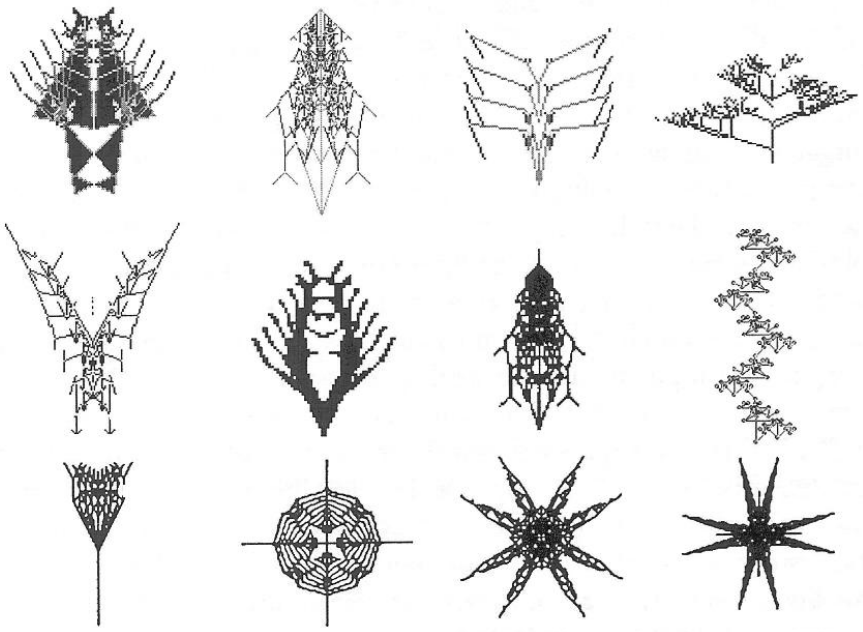


FIG. 3.6 – Biomorphs obtained with an implementation of the interactive algorithm. (Taken from [Ren02].)

3.6 Artificial Embryogenesis

Hugo de Garis worked on the creation of a self-assembly process that he called "artificial embryogenesis". His motivation was that he believed that in the future, machines would have so many components that a sequential mechanical assembly would not be feasible. He theorized that highly complex machines should be self-assembled in a similar way as biological organisms are developed.

He worked on artificial "embryos" as 2D shapes formed by a colony of cells using the CA paradigm. The basic idea is that a GA can evolve the CA rule set to control reproduction of artificial cells [de 91]. The rule sets are encoded in the chromosomes used by the GA and they can be switched on or off, depending on whether the state of a cell matches the gene's "condition field". The test in the condition field determines whether or not the "action field" is activated. If the action field is activated, then cells can reproduce.

De Garis used what he called "differentiable chromosomes", which consisted of several genes or "operons", where in general each operon contained a condition field C and an

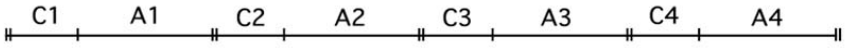


FIG. 3.7 – A differentiable chromosome with four operons.(Taken from [dIF92].)

action field A, as shown in Fig. 3.7.

Each cell in a particular cellular automaton model contains the same chromosome and at every time step all cells calculate their current state by looking at the states of the neighboring cells. Next, each cell compares its present state with the condition fields C_i in the chromosome. If one of the condition C matches, then the corresponding action field A is activated and the reproduction instructions coded in this field are executed.

This author developed a model that evolved reproduction rules for CA, with the goal that the final shape of a colony of cells was as close as possible to a predefined simple shape such as a square or a triangle [de 92, dIF92]. In this model, cells can only reproduce if there is at least one adjacent empty cell, i.e. only edge cells are allowed to reproduce. Assuming that only edge cells can reproduce and that no isolated cells are generated, then cells can be in one of fourteen possible states in a 2D lattice (Fig. 3.8).

If one state is coded by one bit, then only fourteen bits are necessary to code all states, with a predefined position for each state in the bit string. If there is a "1" in the bit corresponding to a state, then all edge cells in that state are allowed to divide and generate a daughter cell, whereas a "0" means that cells in that state are not permitted to reproduce.

When edge cells are allowed to reproduce, it has to be determined the direction at which the daughter cell is to be placed. For states where there is only one empty adjacent cell, there is no option but to place the new cell at that position, so in this case there is no need to code the direction of reproduction. However, when there are two or three empty adjacent cells, a choice has to be made as to where to place the newly produced cell. For states with two empty adjacent cells, only one bit is necessary to determine the direction

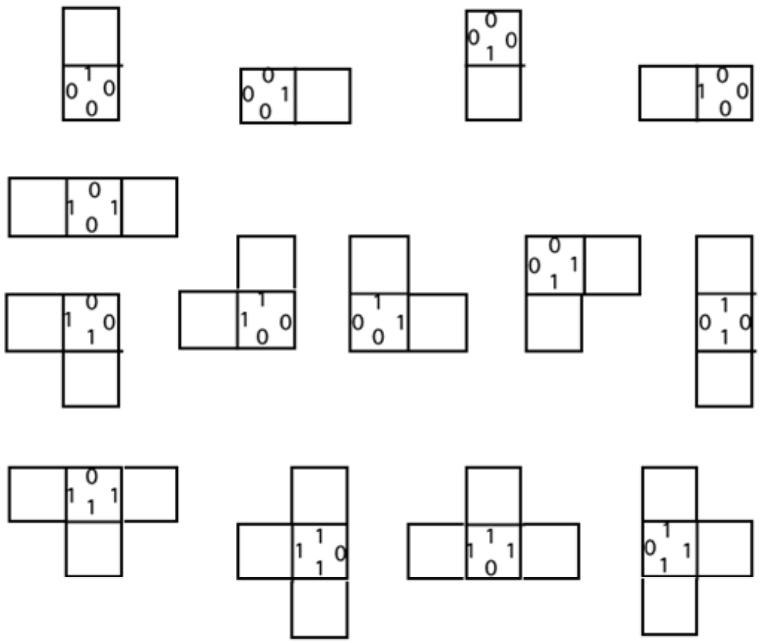


FIG. 3.8 – The 14 neighborless states for 2D cells. (Taken from [dIF92].)

of reproduction, with a predefined direction for each bit value and each state. For states with three empty cells, two bits are sufficient to code the direction of reproduction, again with a predefined direction of reproduction for each state and each combination of the two bits. As a result, since there are 6 states with 2 empty adjacent cells, and 4 states with 3 vacant adjacent cells (see Fig. 3.8), it is necessary to use $6 \times 1 + 4 \times 2 = 14$ bits to specify the direction of reproduction for all states.

The combination of the two 14-bit strings determines for one reproduction cycle which cells are allowed to divide and at which relative position to place their daughter cells. In order to evolve the reproduction rules for more than one cycle with a possibly different reproduction rule for each cycle, it is necessary to introduce in the differentiable chromosome as many instances of the two 14-bit strings as iteration steps are desired. Even the number of iterations necessary to create a predefined shape can be evolved. The chromosome proposed in [dIF92] for this purpose is shown in Fig. 3.9.

The chromosomes thus defined were evolved by a genetic algorithm using the following fitness function :

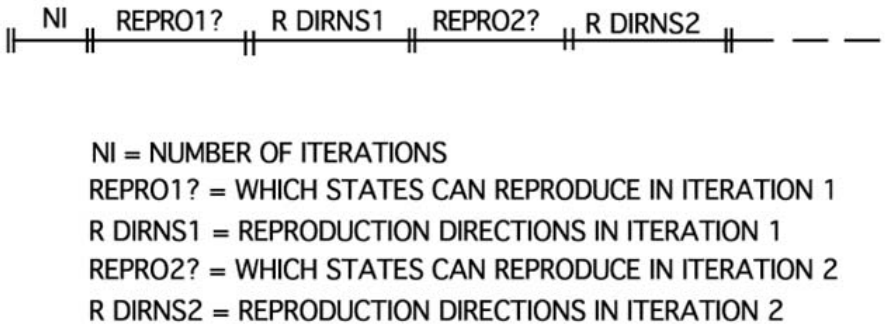


FIG. 3.9 – Format of a simplified chromosome. (Taken from [dIF92].)

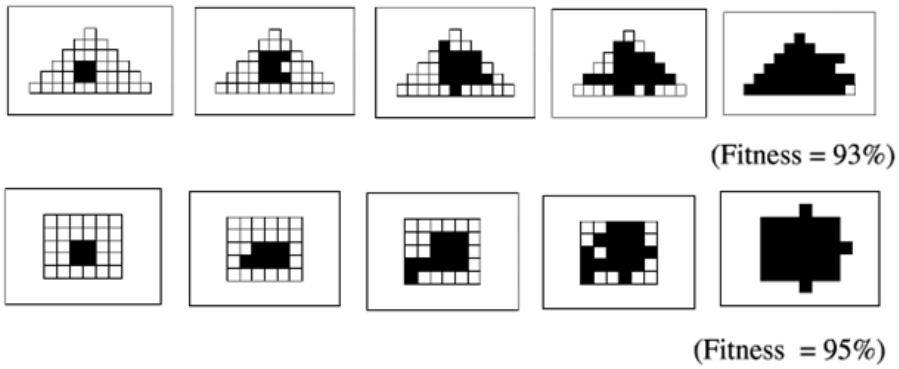


FIG. 3.10 – Evolved triangular and rectangular shapes. The desired shapes are outlined by the empty cells in the background. (Taken from [de 99].)

$$Fitness = \frac{ins - \frac{1}{2}outs}{des}, \tag{3.3}$$

where *ins* is the number of filled cells inside the desired shape, *outs* is the number of filled cells outside the desired shape, and *des* is the total number of cells inside the desired shape. Thus, a fitness value of 1 represents a perfect match.

Using this setup, the generation of several target shapes was tried. Two examples of shapes evolved are presented in Fig. 3.10. Since isolated cells are not allowed in the model, reproduction started with a 2×2 cluster of cells. In both cases, the number of iterations (NI in the chromosome) evolved to be 4.

Several other target shapes, both convex and non-convex were tested. Results showed

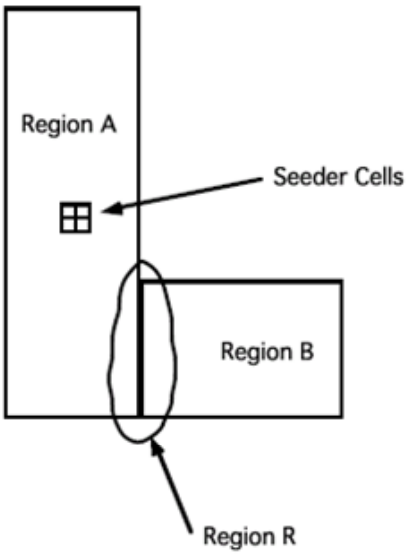


FIG. 3.11 – Desired L shaped and component regions. (Taken from [dIF92].)

that convex shapes could be obtained with a fitness value around 95%, but non-convex shapes evolved poorly, with low fitness values.

After these initial results, de Garis concluded that evolving an artificial embryo implies a type of sequential, synchronized unfolding of shapes. For example, after the main body is grown, then the head and limbs can be grown, followed by the emergence of more detailed shapes, such as those corresponding to fingers and toes [dIF92].

To put this idea to the test, de Garis attempted the generation of an L-shaped form, which is a non-convex shape. Figure 3.11 shows the target shape and its decomposition in regions. The basic idea is to use two genes to generate the L shape, where the first gene would generate region A, and it then would be shut off to let cells in region R express the second gene in order to produce region B.

For this approach to work, it is necessary that cells can somehow determine their position in the grid. This author decided to use a concentration gradient of a theoretical chemical to provide cells with positional information. Each cell has a certain quantity of this chemical, which is replicated and a fraction of it is transmitted to the daughter cell. Thus a concentration gradient is formed with its peak at the position of the initial

cells. Each cell also has an 8-bit storage for determining the direction of the highest concentration of the chemical. At each time step, concentration is measured at each cell position averaging the concentrations from the cell and the three cells towards the direction of the eight main "cardinal points" (N, NE, E, SE, S, SW, W and NW), thus obtaining eight average concentration values for each cell. The bit corresponding to the direction with the highest concentration is then given a value of 1, and the other bits are set to 0. In this manner, by measuring the concentration values from neighboring cells, a cell can determine the concentration gradient and have an estimate of its relative position. For example, cells in the south-east of region A would have their highest gradients towards the north-west direction and would therefore have their north-west bit set to 1.

On the other hand, it is necessary that cells know when to switch gene A off so that gene B can be expressed. The solution proposed by de Garis was to let cells carry an internal generation count, so that a parent cell with a count value of g would produce a daughter cell with a count of $g + 1$.

Under these assumptions, the two-operon chromosome shown in Fig. 3.12 was proposed. Field X codes for the iteration count to determine when the first operon is turned off and the second operon becomes activated. When the iteration count reaches $X + Y$, the second operon is switch off. The *NEWS DIRNS* field has 8 bits and determines which cells can reproduce after X iterations. Only cells that have their direction bit set to 1 when the same direction bit is set to 1 in this field can replicate. For example, if a cell's highest gradient is NW, and the bit corresponding to NW in the *NEWS DIRNS* field is 1, then the cell is allowed to reproduce after X iterations. Finally, the *REPRO ?/DIRN PAIRS* fields are similar in function to the *REPRO ?* and *R DIRNS* fields presented in Fig. 3.9.

Figure 3.13 shows the results obtained using this approach when trying to produce an L shape. The fitness value of the resulting shape shown in Fig. 3.13b was about 80%, which was not as good as when generating convex shapes.

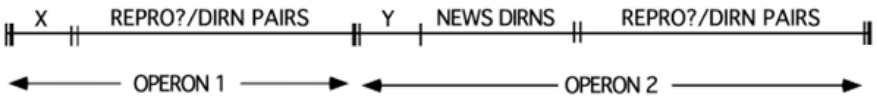


FIG. 3.12 – Two-operon chromosome for the generation of an L shape. (Taken from [dIF92].)

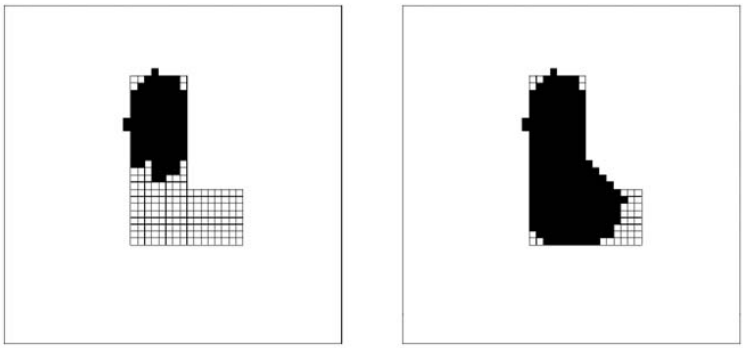
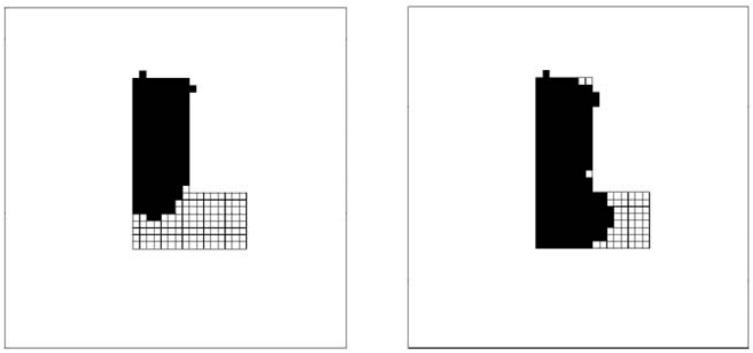


FIG. 3.13 – L shape produced by an evolved two-operon chromosome. The target shape is outlined by empty cells. (Taken from [dIF92].)

In an attempt to obtain better results, the author used a technique that he termed "shaping", which consisted in dividing the evolutionary process in phases with intermediate targets. In this case, the second operon was disabled and evolution was initially conducted with the objective of producing chromosomes that could generate region A of the L shape (see Fig. 3.11). The resulting chromosomes were then fed as a starting chromosome population for the GA, this time with both operons fully functional. Results from these experiments are shown in Fig. 3.14.

Despite the bias towards chromosomes that were already conditioned to start with the correct region, the shaping technique did not produce better result, as the final fitness value of the most successful experiments was again about 80%. This author went on and tried to generate other non-convex shapes, such as a snowman shape and a turtle shape with a combination of circle shapes, but he again ran into limitations in the maximum fitness values that he was able to obtain.

Even though the approach used by de Garis proved the potential of the application



(a) OPERON 1 ONLY (b) AFTER OPERONS 1 & 2

FIG. 3.14 – L shape produced by an evolved two-operon chromosome using the shaping technique. The target shape is outlined by empty cells. (Taken from [dIF92].)

of evolutionary techniques to the growth of artificial cells in order to generate desired shapes, his results were of limited success. However, he was one of the first researchers to use the concept of sequential gene activation for the production of artificial cellular structures using the CA paradigm.

3.7 Evolutionary Neurogenesis and Cell Differentiation

Kitano was another of the first researchers that conducted experiments towards evolving an artificial development system. This author was successful at evolving large neural networks using genetic algorithms [Kit90]. He encoded into the GA chromosome the neural network connectivity matrix using a graph generating grammar. Instead of using a direct encoding of the connectivity matrix, a set of rules was created by a grammar overcoming the scalability problem on the cases tested. Previous attempts saw how convergence performance was greatly degraded as the size of the neural network grew larger. The grammar used was an augmented version of Lindenmayer's L-System and used matrices as symbols.

Kitano later developed a model of neurogenesis and cell differentiation based on a simulation of metabolism [Kit94]. The idea was to see if artificial multicellular organisms

could be created using genetic algorithms evolving the metabolic rules in the genome of the cell. Although all cells carry the same set of rules, individual cells can express different rules because of differences in their local environment, thus producing a sort of cell differentiation. Metabolic rules define which kind of metabolite can be transformed into another kind and under what conditions of metabolite concentration and enzyme presence. These rules are coded in the genome of the cell, and are of the form "If the level of metabolite A is less (or greater) than n, then start a reaction that converts metabolite B into C using the enzymatic properties of metabolite D". These rules are applied to all the metabolites in every cell of a developing organism at each iteration of the simulation. One of the metabolites represents DNA molecules, and when its concentration is above a specified threshold, cells can divide and produce daughter cells with a certain amount of random fluctuation. This randomness provides the base of a chaotic process that can lead to symmetry breaking, which is a widespread feature of developing organism. In addition, cell death can occur in the model when the overall metabolism is too low or too high.

Additionally, Kitano *et al.* developed a project to simulate the development of the soil nematode *C. elegans* [KHKL98]. They chose to model the embryogenesis of *C. elegans* because of its relatively simplicity in structure and because it is one of the best studied multicellular organism in biology. They used data on cell lineage and cell location published in [SDT+92] in order to generate a 3D computer graphics image from the division of the first cell to approximately 600 minutes after the first cellular cleavage. They tried to match as much as possible the data from the actual organisms and the results from the simulation. When there was missing simulation data regarding the actual position of cells, their system could calculate forces between cells such as the force that pushes adjacent cells. Their long-term goal was to produce a complete synthetic model of *C. elegans* cellular structure and function.

3.8 Evolutionary 2D/3D Morphogenesis

Fleischer and Barr presented a simulation framework and computational testbed for the study of 2D multicellular pattern formation [FB92]. Their initial motivation was the generation of neural networks using a developmental approach, but their interest soon shifted towards the study of the multiple mechanisms involved in morphogenesis.

Their approach combined several developmental mechanisms that they considered important for biological pattern formation. Previous work from other researchers had individually considered chemical factors, mechanical forces, and cell-lineage control of cell division to account for some aspects of morphogenesis. These authors decided to combine these factors into one modeling system in order to determine how the interactions between these components could affect cell pattern development.

The modeling framework consists of discrete cells capable of independent movement and controlled by an artificial genome. The latter is a set of differential equations that depend on the cell's current state and its local environment. The changes in the environment are in turn determined by differential equations which implement mechanical forces and the diffusion of extracellular substances. The computer implementation of the model was able to simulate a number of simple multicellular behaviors, such as cells following gradients, cell clustering, cell differentiation, pattern formation, and network generation.

Fleischer and Barr emphasized that it was the interactions between the developmental mechanisms that were at the core of the determination of multicellular and developmental patterns, and not the individual elements of the model.

On the other hand, Eggenberger used an evolutionary approach for studying the creation of neural network and the simulated morphogenesis of 3D organisms based on differential gene expression [Egg97a, Egg97b]. His model for simulating morphogenesis includes a genome with two types of elements : regulatory units and structural genes. The regulatory units act as switches to turn genes on and off, while structural genes code for specific substances that are used to modulate developmental processes. Every gene is defined as

having the same number of integers, with the last integer, called the marker, indicating the type of gene. The integers composing the genome are taken from the set $\{1, 2, 3, 4, 5, 6\}$. The fist gene of the genome is assumed to always be a regulatory unit. All genes from the first to the one ending in the marker 5 are defined as regulatory units. Next, all genes after the last regulatory unit and until the gene with the marker 6, are defined as structural genes, and the activity of the latter depends on the regulatory units that precede them. After the last marker 6, the next marker 5 is searched and all the genes between these markers are again considered regulatory units, which control the structural genes found next, until the gene with marker 6, and so on until the end of the genome (see Fig. 3.15).

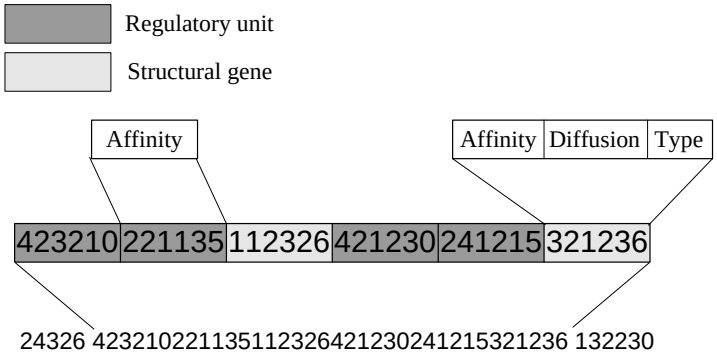


FIG. 3.15 – Example of a segment of the genome used in Eggenberger's model.

From the above definition, it follows that one or more adjacent regulatory units can control expression of one or more adjacent structural genes. Gene expression control is based on its biological counterpart, where sequences in the genome, called *cis*-elements, can show affinity with soluble factors, typically proteins, and the degree of matching determines the strength of the effect, either as enhancement or as inhibition of gene expression.

Structural gene expression is regulated by the concentration and affinity of transcription factors. Each cell contains a list of transcription factors, which consist of string of integers that can be compared for matching with the regulatory units in the genome. Affinity is calculated subtracting in the appropriate base the first n integers (from 6 to 8 in the implementation) of both strings. This difference represents the degree of affinity and

its sign designates its role : positive for enhancement and negative for inhibition. On the other hand, every transcription factor in the cell has a concentration value. The product of the affinity and the concentration of each transcription factor at a regulatory unit is calculated and the resulting values are added. The same procedure is performed for every regulatory unit of a gene. The resulting sum is then fed to a sigmoidal function and if predefined thresholds are crossed, the corresponding gene is activated or inhibited. The associated equations are shown next :

$$r_j = \sum_{i=1}^{n} aff_i \times conc_i \tag{3.4}$$

$$a_k = \frac{1}{1 + e^{-\sum_{j=1} r_j}} \tag{3.5}$$

$$g_k = \begin{cases} -1.0 & if\ a_k < 0.2 \\ 1.0 & if\ a_k > 0.8 \\ 0.0 & otherwise \end{cases} \tag{3.6}$$

where aff_i is the affinity of transcription factor i with regulatory unit j, $conc_i$ is the concentration of transcription factor i, r_j is the activity of regulatory unit j of a structural gene, a_k is the total sum of the activities of all regulatory units of gene k, and g_k is the activity of gene k [Egg97b].

An active structural gene can perform a number of functions, depending on its type, which is determined by some of its integers. A structural gene can be translated into a transcription factor, a cell adhesion molecule that can connect cells with the corresponding adhesion molecule, or a receptor used to regulate communication between cells. A structural gene can also elicit a function such as cell division or cell death.

Eggenberger implemented cell signaling in several ways : with intracellular substances which regulate the activity of its own gene, with specific receptors on the cell surface

which can be stimulated by other substances, and with substances that can penetrate cell walls and diffuse to neighboring cells. These diffusing substances can provide cells with positional information.

In order to direct morphogenesis towards a 3D shape with desired properties, the artificial genome was evolved by means of a genetic algorithm, with mutation and single-point crossover as genetic operators. In the implementation of the model, a series of 8 genetic elements with 2 regulatory units and 2 structural genes each were used. Figure 3.16 shows the shapes grown by evolved genomes when a single morphogen-producing cell is allowed to reproduce. A spherical shape tends to be formed due to the modulating effect of the morphogen concentration gradient on cell replication.

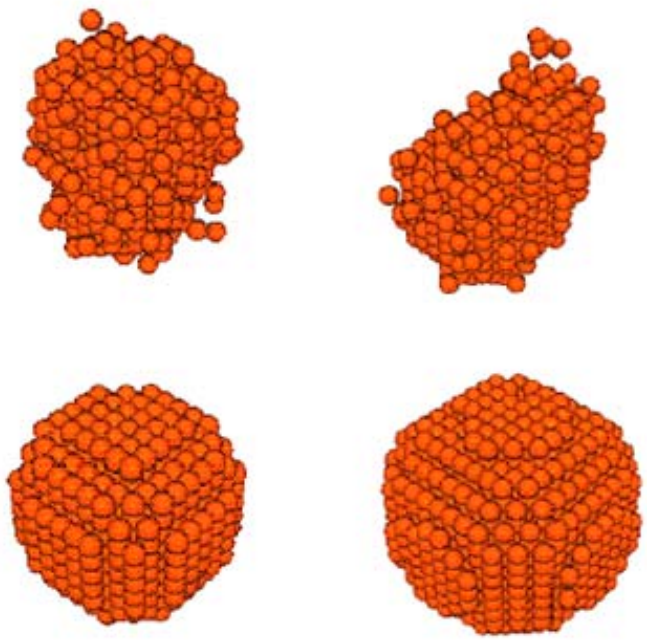


FIG. 3.16 – Upper cell clusters result from random growth. Lower cell clusters emerged after addition of one morphogen to modulate growth. (Taken from [Egg97b].)

In other series of experiments, a fitness function that rewarded bilateralism was used. Before the initial cell is allowed to reproduce, the artificial environment is conditioned with morphogen sources on each of the three axes at varying distances to the origin. All

three morphogens are different, so that they can regulate growth independently. Cells are able to read and respond to the varying morphogen concentrations. The fitness function was dependant on the total number of cells and their position with respect to one of the axes. Figure 3.17 shows some of the resulting evolved 3D shapes.

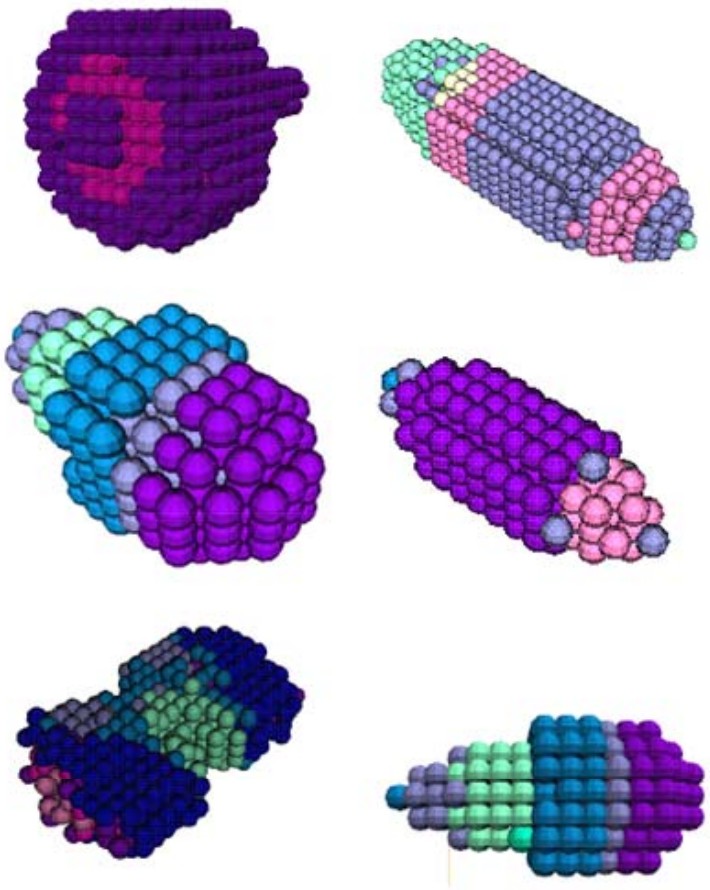


FIG. 3.17 – Examples of evolved 3D forms. The fitness function only evaluated the number of cells and the degree of bilaterality. (Taken from [Egg97b].)

In order to gain more flexibility, this author extended his model later on. In the extended model, structural genes have seven parameters that encode their properties. Among the new additions there is a field that stores the probability of interaction with ligand molecules, a field for storing threshold levels of activation, and a field that contains the decay rate of the expressed molecule. Regulatory units are also endowed with a field that stores the threshold level at which the associated structural genes should be activated

[Egg03, EH04].

The new model used floating point numbers, and for this reason Eggenberger decided to use the Evolutionary Strategy developed by Rechenberg as evolutionary algorithm, instead of the genetic algorithm used in the previous model. Figure 3.18 shows some of the simulation results obtained with this model. These forms represent six stages of a simulated invagination, induced by placing a source of morphogen at a random position in the cell cluster. This invagination process is similar to the gastrulation process seen at the initial stages of development in higher animals.

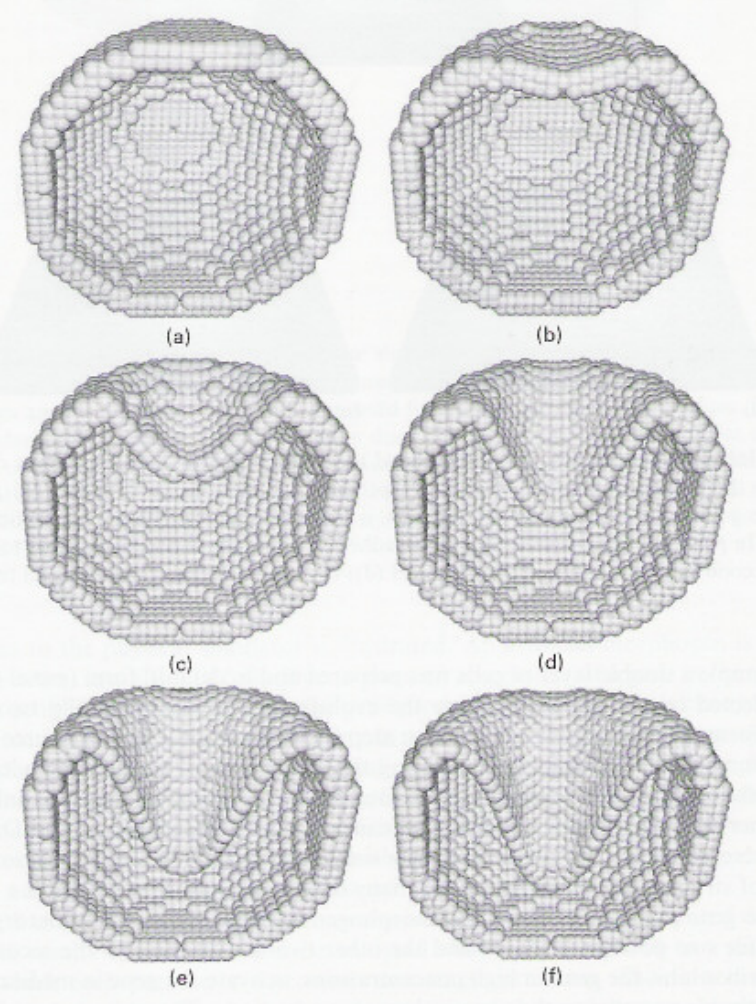


FIG. 3.18 – Stages of a simulated invagination. (Taken from [Egg03].)

Eggenberger's models showed that a number of mechanisms central to development

such as cellular growth, cell differentiation, axis definition, and dynamical changes in shape could be simulated using a framework not based on a direct mapping between a genome and the resulting cellular structure. The shapes that emerge in the models are the result of the interaction among cells and their environment.

3.9 METAMorph

METAMorph, which stands for *Model for Experimentation and Teaching in Artificial Morphogenesis*, is an open source software platform for the simulation of cellular development processes using genomes encoded as gene regulatory networks. The design is made by hand and it allows visualization of the resulting morphological cellular growth process [STK05].

As in higher organisms, cellular growth starts in METAMorph with a single cell (the zygote) and is regulated by gene regulatory networks in interaction with proteins. All cells have the same genome consisting of a series of genes. Each gene can produce exactly one protein, although the same protein can be produced by different genes.

A protein is defined by a unique name, a type (internal or external) and two constants (decay and diffusion). Internal proteins can only diffuse inside a cell, while external proteins can travel through cell membranes and can have a signaling function for communication among cells.

The concentration of each protein is stored in 12 sites inside every cell. As a result, proteins may not be uniformly distributed within the cytoplasm. On the other hand, genes can be expressed differently at each of these sub-cellular sites based on local protein levels. The diffusion constant determines the amount of protein that migrates from one site to another at each time step.

Protein production by a gene is dependent on the promoter sequences located next to the gene. The influence of the promoter is calculated using the sum of the products of

the weight and concentration of all proteins. The resulting value is then fed to a sigmoid function that determines the next protein concentration level.

In the model, cells are represented by spheres of a fixed radius and each cell occupies a position on an isospatial 3D grid, so that every cell can have up to 12 equidistant neighboring cells. Cell actions can be triggered when a specific protein concentration level rises above a threshold value. These actions are :

Cell division If there is an empty space available, a dividing cell produces a daughter cell placed in the space located in the direction of the mitotic spindle.

Mitotic spindle movement Cell orientation is achieved through the definition of a "mitotic spindle" pointing towards one of the 12 possible adjacent cell positions. The spindle orientation can be varied when specific proteins reach a threshold level.

Programmed cell death (apoptosis) The cell is removed from its position leaving an empty space.

Differentiation Cell type is visualized in the model as a distinct external color. The type of a cell is not related to its function.

The main disadvantage of this simulation platform is that the cellular development model has to be designed through a trial and error process that is limited by the designer's ability to introduce the appropriate parameter values. By the authors' account, this trial and error process typically involves a considerable amount of time, since simulation times are usually high due to the parallel nature of the morphogenetic process. To compound the problem, small changes in design can have substantial consequences on the final shape caused by "the butterfly effect".

METAMorph would greatly benefit from introducing in the platform a search process (possibly an evolutionary algorithm) so that the system could find by itself a suitable design for a desired cellular growth pattern.

3.10 Random Boolean Networks

Random Boolean Networks (RBNs) are a type of discrete dynamical networks that consist of a set of Boolean variables whose state depends on other variables in the network. In RBNs, time and state values take only integer values. The first Boolean networks were proposed by Kauffman in 1969 as a randomized model of a gene regulatory network [Kau69, Kau74, Kau93].

RBNs are also known as $N-K$ models or Kauffman networks. They consist of a set of N binary-state nodes, where each node represents a gene that can be on or off, and a set of K edges between nodes that represent relationships between nodes. The connections between nodes are randomly selected and remain fixed thereafter. The dynamics of the RBN is determined by the particular network configuration and by a randomly generated binary function, defined as a lookup table for each node.

RBNs are a generalization of CA, but unlike the latter, the state for each node is determined by nodes that are not necessarily in the immediate vicinity. However, in CA a node can take up any number of states, while a RBN is constrained to only two states. As in canonical CA, updating is synchronous in RBNs, so that the state of all nodes at time $t+1$ depends on the state of nodes at time t and are all updated at the same time.

Figure 3.19(a) presents an example of a RBN where all three nodes are connected to all the other nodes ($N = K = 3$). A possible update function is shown in 3.19(b), while the corresponding state space diagram is presented in 3.19(c) [Ger04].

Since the state space is finite, a state can eventually be visited more than once, and when that happens it is said that an *attractor* as been reached. If the attractor consists of a single state, it is called a *point attractor*, while if it contains two or more states, it is called a *cycle attractor*. The states that lead to an attractor are called the *attractor basin*. The RBN shown in Fig. 3.19 contains both a point and a cycle attractor, as can be seen in Fig. 3.19(c).

Depending on the behavior of the network dynamics, three different phases or regimes

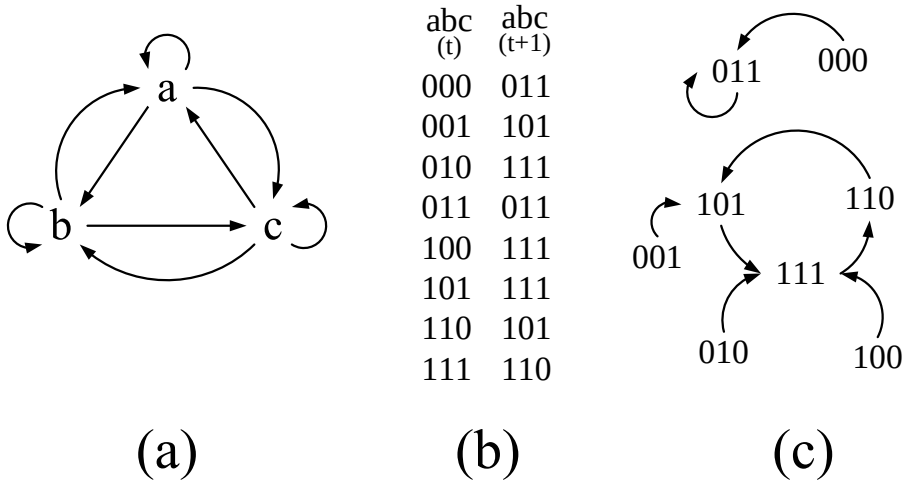


FIG. 3.19 – Example of a Random Boolean Network. a) Network configuration ; b) Lookup table for state transitions ; c) State space diagram.

can be distinguished : ordered, chaotic and critical [Kau04]. In order to identify these phases, a plot can be generated with a small black square for a node in state '1' and with all the nodes lined up at the top of the plot with time flowing downwards. Figure 3.20 shows plots from each type of RBN. Early studies of RBNs quickly revealed that parameter K had a great influence on the type of RBN generated. Broadly speaking, networks with $K \leq 2$ correspond to the ordered type, while the chaotic type is usually seen in networks with $K \geq 3$.

The critical type of behavior is usually considered by researchers as the most interesting of the three types. The ordered type is too static to derive useful observations applicable to dynamic systems, whereas the chaotic type is too random to study any kind of reproducible property. Of particular interest is what has been termed "the edge of chaos" [Heu98], which in RBNs means a condition in networks of the critical type where dynamically changing states are in a phase transition between ordered and chaotic.

Kauffman also discovered that these transitional phases were related to diverse aspects of the stability of a network. He conducted experiments where he manually perturbed or damaged nodes in a RBN in order to study the effect of these alterations. He found that in the ordered regime, damage to a node does not usually spread to other nodes. In the

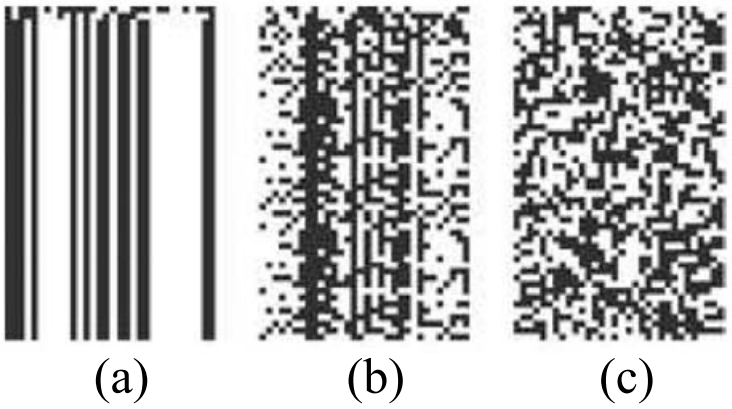


FIG. 3.20 – Examples of state dynamics for the three types of Random Boolean Networks. $N = 32$. Black square = state '1'; white square = state '0'. Initial states at the top and time flowing downwards. a) Ordered ($K = 1$); b) Critical ($K = 2$); c) Chaotic ($K = 5$). (Taken from [Ger04].)

chaotic phase a small change could have large consequences in the network dynamics. Finally, at the edge of chaos, changes can disseminate, but not necessarily affecting the whole network [Ger04].

It has been suggested that living systems evolve more naturally at "the edge of chaos", since they need certain stability in order to survive, while at the same time they need flexibility to explore their space of possibilities [Lan90, Kau04]. Furthermore, Kauffman suggested that biological entities could have originally been generated from random elements, with no absolute need of precisely programmed elements [Kau69]. This conjecture was derived from his observations of the complex behavior of some of these randomly generated networks and the inherent robustness he found in them.

3.11 Artificial Regulatory Networks

Over the years, many models of Artificial Regulatory Networks (ARNs) have emerged in an attempt to emulate the gene networks found in nature. Torsten Reil was one of the first researchers to propose an artificial genome with biological plausible properties based on template matching on a nucleotide-like sequence [Rei99]. The genome is defined as a

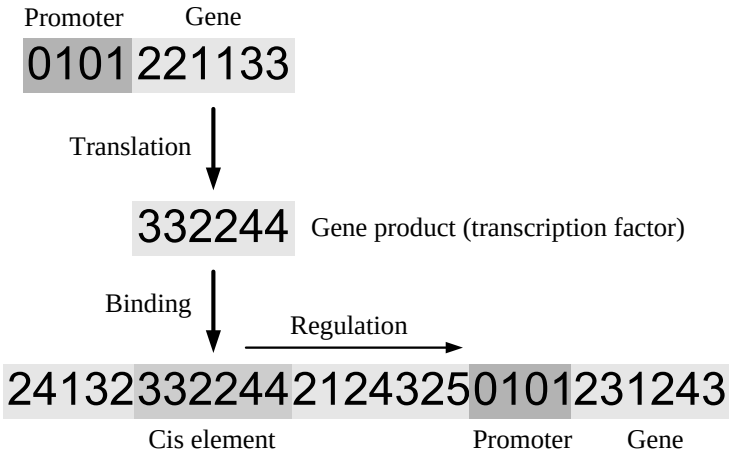


FIG. 3.21 – Gene expression and regulation in the artificial genome. Gene length N is equal to 6.

string of digits and is randomly created. Genes in the genome are not predefined, but are identified by a "promoter" sequence that precedes them. The string arbitrarily chosen for the promoter sequence is 0101, similar to TATA boxes found in biological genomes. The N digits immediately following the promoter sequence constitute the gene (Fig. 3.21).

After a gene is identified, it is translated using a simple transformation of the sequence. In the implementation used by Reil, the gene product is simply generated by the addition of one digit to the sequence, with the corresponding modulo operation. For each identified gene product, all direct matching sequences in the genome are searched and stored (see Fig. 3.21). These matching sequences are termed "*cis*-elements" in the model, as their biological counterparts, and they constitute the regulatory units of the genome. A regulatory sequence can behave either as an enhancer to activate a gene, or as an inhibitor to block its activity. The role of the gene product is defined by the value of its last digit. For instance, all gene products ending in '1' are inhibitors. Regulation is not concentration dependent ; it suffices to have one enhancer to activate a gene. However, inhibition is defined as having precedence over enhancement.

In order to test the model, after a genome was generated and all the above rules were applied, gene expression over time was studied. Initially all genes but one were set as

turned off. It was then determined which genes were regulated by the initial active gene, and they were labeled as *on* or *off*, depending on their role as enhancers or inhibitors. In the next step, the same procedure was applied using the newly activated genes, and so on for all subsequent time steps until a specified number of cycles was reached. The resulting pattern of expression is visualized in what the author called an *expression graph*, which is a 2D graph with time in the X-axis and a black dot for every active gene in the Y-axis.

The author varied the following model parameters : genome size, gene length, base (range of digits), and degree of inhibition measured as a fraction of the range of digits that determined the role of inhibitor in *cis*-elements. It was found that the behavior of the model was highly dependent on the parameter values.

As with RBNs and other dynamical systems, three basic types of behavior were identified : *ordered*, *chaotic*, and *complex*. Gene expression was called ordered if genes were continuously active or inactive throughout the run. If gene expression seemed to be random with no apparent emerging pattern, it was called chaotic. If the expression of genes was considered to be between ordered and chaotic with the formation of identifiable patterns, then it was called complex. The author broadly identified which ranges of parameter values gave rise to each type of behavior.

For genomes with a behavior of the complex type, it was found that gene expression converged to the same pattern from a number of different start genes. This was viewed as the cycle attractors found in the RBNs described in Section 3.10. Every genome of this kind typically contained several such attractors. This finding supported the notion proposed by Kauffman that cell types could be viewed as attractors of gene expression in natural gene networks [Kau71]. Thus, cell differentiation could be viewed as a dynamic system that moved from one attractor to another.

On the other hand, Reil observed that even after manual perturbations in the model, gene expression usually returned to the attractors that emerged previously. It must be emphasized that the artificial genomes endured no evolution. The behaviors observed were

the result of the properties of genomes entirely generated at random. Reil hypothesized that robustness in natural genomes might be an inherent property of the template matching system, rather than the result of the natural selection of the most robust nucleotide sequences [Rei99].

An important advancement in the design of an artificial genome model was made by Banzhaf, who designed a genetic representation based on ARNs [Ban03]. His genome consists of a randomly generated binary string. Similarly to other models, the "promoter" is a particular sequence that signals the beginning of a gene in the genome. The promoter was arbitrarily chosen as the string 'XYZ01010101', with 'XYZ' being any 3-bit combination. In a randomly generated bit string, the expected frequency of the pattern '01010101' is $2^{-8} \approx 0.0039 = 0.39\%$. The gene following the promoter was defined as a series of five 32-bit strings, for a total of 160 bits per gene. Immediately before the promoter sequence, two special 32-bit sites were defined : an enhancer and an inhibitor.

The five 32-bit regions after the promoter are translated into a protein using a majority rule, i.e. the first bit in the translated protein corresponds to the bit that is in majority in the first position of the five protein-coding regions, and so on until the end of the 32-bit sequence. In this model the transcription process seen in nature is completely disregarded. There is no intermediary element –such as the messenger RNA sequences found in biological systems– between the gene and the translated protein.

After a protein has been produced, it is then compared on a bit by bit basis with the enhancer and inhibitor sequences on all genes in the genome. The comparison is achieved through the use of an XOR operation, which renders a '1' if the bits compared are complementary. It is expected that a Gaussian distribution is found when measuring the match between a particular protein and all the 32-bit sequences of a randomly generated genome. Thus there will be few sequences with a high degree of matching and likewise there will be few poor-matching sequences. The majority of 32-bit strings will be average-matching sequences. Banzhaf confirmed through simulations that randomly generated

genomes of various sizes contained the expected number of genes. For example, a genome consisting of 100,000 bits contained 409 genes, which is consistent with the 0.39% rule.

The degree of matching between proteins and enhancer or inhibitor sites defines the degree of activation or inhibition of genes, respectively. The influence of a protein is exponential with the number of matching bits. How the degree of activation and inhibition of genes with the corresponding change in protein concentration are calculated will be discussed more thoroughly in Section 4.3 of Chapter 4, as the ARN proposed in this thesis is based on the model here described. It suffices to say for the moment that all translated proteins are compared against the regulatory sites of all genes and the concentration change for the next time step is then calculated for all proteins.

Using randomly generated genomes with no evolution involved, Banzhaf discovered several types of protein concentration dynamics over time. Figure 3.22 shows four examples of protein concentration variations over time. Protein concentrations are normalized so that total protein concentration is always the unity. In Fig. 3.22(a), protein concentration change exhibits a dampened oscillating pattern for several proteins, while in Fig. 3.22(b) protein concentrations follow a slow and smooth development. It can happen that a protein quickly dominates in concentration over the other proteins as in Fig. 3.22(c), or a protein can achieve higher values of concentration with a subsequent switch of expression to another gene (Fig. 3.22(d)).

It was soon discovered that the degree of matching between regulatory sites and a protein by one or two bits could sometimes induce dramatic changes in the dynamics. Figure 3.23 exemplifies this point my showing the results of manually modifying the genome from Fig. 3.22(d). In Fig. 3.23(a) the degree of matching between protein 7 and the inhibitor site in gene 4 was changed by one bit. The transition of expression to another gene is displaced to the right in the graph and this displacement is more pronounced by simply modifying another bit in the same gene (Fig. 3.23(b). Note the change in the time scale.)

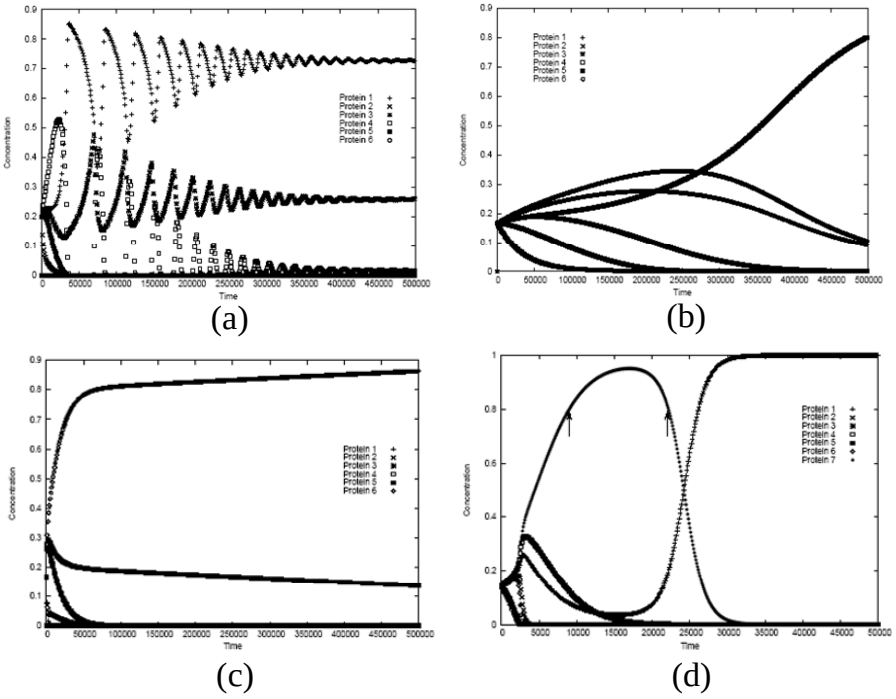


FIG. 3.22 – Examples of time development of protein concentrations. (a) Oscillating concentration change ; (b) Slow and smooth concentration change ; (c) Quick settlement into a point attractor ; (d) Transition concentration change between two proteins. (Taken from [Ban03].)

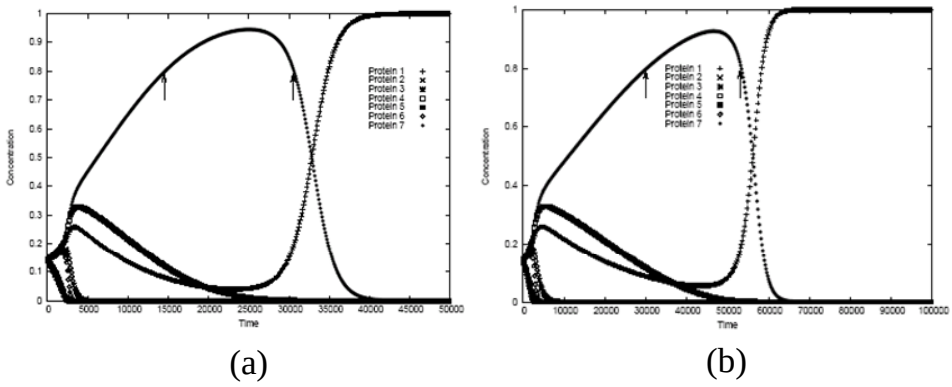


FIG. 3.23 – Effect of a single bit change in protein concentration behavior. The genome is the same as in Fig. 3.22(d).(a) Degree of matching between protein 7 and inhibitory site to gene 4 changed by one bit ; (b) Degree of matching increased by another bit. (Taken from [Ban03].)

After observing the dynamics of proteins from genomes that had experienced no evolution, Banzhaf used Genetic Programming (GP) in an attempt to drive the dynamics of

gene expression towards desired behaviors. He started by evolving the genome to obtain a target concentration of a particular protein. He found out that in general the evolutionary process quickly converged towards the target state.

In the search of other applications for this model, Banzhaf and his colleagues evolved genomes where the protein concentrations were used to modulate output functions such as sinusoids, exponentials and sigmoids [KLB04]. A randomly selected 62-bit string in the genome was chosen to function as an enhancer and an inhibitor site, and they were allowed to freely interact with expressed proteins. However, instead of controlling gene expression, these sites were used to calculate the output value of a function mapped to the $[-1, 1]$ interval. These authors found that it was feasible to evolve genomes for generating time-series for function optimization. They also used evolution of the artificial genome model to produce networks with small-world and scale-free topologies [KB04].

Another author that evolved an ARN in order to perform a specific task was Bongard [Bon02]. He designed virtual modular robots that were evaluated for how fast they could travel over an infinite horizontal plane during a time interval previously specified. The robots are composed of one or more morphological units and zero or more sensors, motors, neurons and synapses. Each morphological unit contains a genome, and at the beginning of the evolution a genome and a motor neuron are inserted into the initial unit. As in similar genome models, a gene is preceded by a promoter sequence. Transcription factor sources are placed at the poles of the initial unit to allow the ARN to establish its anterior/posterior axes. The unit then starts producing transcription factors that activate expression of genes in the genome. Transcription factors can have a direct influence in the external features of the developing unit. They can activate one of 23 predefined phenotypic transformations, such as increasing the length of the unit, causing a unit to divide in two, or adding, deleting or modifying neurons or synapses. The unit's behavior is dependent on the real-time propagation of sensory information through its neural network to motor neurons, which can actuate the unit's joints to generate movement. Using this model,

Bongard demonstrated that mobile units could be evolved in a virtual environment. His results suggest that a similar model might be applied in the design of physical robots.

Other authors have performed research on artificial regulatory networks using a number of approaches. Willadsen and Wiles designed a genome based on the model proposed by Reil [Rei99]. As in other models, the genome consists of a string of randomly generated integers where a promoter precedes a fixed-length gene. Gene products are produced, which can regulate expression of other genes [WW03]. While their genome model offered no major improvement over previous models, these authors succeeding in showing that there was a strong relationship between gene network connectivity and the degree of inhibition with respect to generating a chaotic behavior. Low connectivity gene networks were found to be very stable, while in higher connectivity networks there was a significantly elevated frequency of chaotic behavior. The same research group suggested that the synchronous updating of the network dynamics regularly used in genome models was not the most adequate since it was not biologically plausible [HW04b]. They suggested that asynchronous updating of the dynamics was more realistic since biological cells do not work synchronously. They found that using asynchronous updating, dynamics converged to an attractor under almost all conditions [HW04a].

Flan *et al.* used ARNs to construct 2D cellular patterns such as borders, patches and mosaics [FHB+05]. They implemented the ARN as a graph, where each node represents a distinct expression level from a protein, and each edge corresponds to interactions between proteins. A protein is influenced when its production or inhibition is altered as the function of other protein concentration levels. A set of differential equations was used to define the rate of production or inhibition. In their model, cell to cell contact signaling was sufficient to form a number of global patch patterns. On the other hand, they found it difficult to produce certain patterns with a single ARN, but they solved the problem by using disjoint ARNs run in parallel and combining their protein concentration levels. These authors conjectured that complex ARNs in nature might have evolved by combining

simpler ARNs.

Nehaniv's research group has worked on ARNs aiming at evolving a biological clock model [KNSQ06, KNS07]. They studied the evolvability of ARNs as active control systems that responded with appropriate periodic behaviors to periodic environmental stimuli of several types. Their genome model is based on the one proposed in [QNDR03], from the same research group. Unlike the model on which it is based, the genome in the biological clock model contains an evolvable number of complex *cis*-regulatory control sites. Each regulatory site in turn contains a number of activating or inhibitory binding factors. Although their model only considered the evolution of the genome of one single cell, their results with the biological clock model could be used to synchronize reproduction of cells in an artificial development model.

3.12 Evolutionary Development Model

Kumar and Bentley designed a developmental testbed that they called the Evolutionary Development System (EDS). It was intended for the investigation of multicellular processes and mechanisms, and their potential application to computer science. The EDS contains the equivalent of many key elements involved in biological development. It implements concepts such as embryos, cells, cell cytoplasm, cell wall, proteins, receptors, transcription factors, genes and *cis*-regulatory regions [KB03a, KB03b].

In the EDS, proteins are implemented as objects. Each protein has a field to identify it as one of the eight types defined. Protein objects contain a current and a new state object that are used to simulate parallelism in protein behavior. These state objects include information such as the protein diffusion coefficient. Protein diffusion in the medium is implemented by means of a Gaussian function centered at the protein source. It is assumed that proteins diffuse uniformly in all directions.

Unlike other models, there are two types of genomes in the EDS. The first genome

stores protein-related parameter values such as the rates of synthesis, decay and diffusion. The second genome encodes the architecture to be used for development by describing which proteins have a role in the regulation of the different genes. The second genome is contained inside every cell during the simulation of the developmental process. The first genome is only used to initialize proteins with their respective values.

The genome inside cells is represented by an array of genes, where each gene consists of a *cis*-regulatory region and a protein-coding region. The *cis*-regulatory region contains in turn an array of target sites for binding matching transcription factors. As in other models, the binding of transcription factors in the *cis*-regulatory region modulates the activity of the gene. By summing the product of concentration and interaction weight of each transcription factor, a value is obtained that is fed to a sigmoid function to determine whether or not the associated protein-coding region should be translated.

Cells in the EDS are autonomous agents that have sensors in the form of surface receptors capable of binding to substances in the environment. Depending on their current state, cells can exhibit a number of activities such as division, differentiation shown as an external color, and apoptosis or programmed cell death.

A genetic algorithm with tournament selection was used to evolve the genomes. One of the morphogenesis experiments consisted in evolving spherical embryos using the equation of a sphere as a fitness function. Results showed that evolution did not make use of many proteins and the evolved ARNs were not very complex. The authors considered that it was likely that their system had a natural tendency to produce almost spherical cell clusters and that it did not take much to achieve the goal desired.

The design of the EDS was probably too ambitious by involving many elements that introduced more variables and interactions in the system than desired. Results obtained with the EDS are meager considering the number of concepts involved. The system might prove its true potential with a more complex target cellular structure.

3.13 The French Flag Problem

The problem of generating a French flag pattern was first introduced by Wolpert in the late 1960s when trying to formulate the problem of cell pattern development and regulation in living organisms [Wol68]. This formulation has been used since then by some authors to study the problem of artificial pattern development.

Lindenmayer and Rozenberg used the French flag problem to illustrate how a grammar-based L-System could be used to solve the generation of this particular pattern when enunciated as the production of a string of the type $a^n b^n c^n$ over the alphabet $\{a, b, c\}$ and with $n > 0$ [LR72]. On the other hand, Herman and Liu developed an extension of a simulator called CELIA [BH70] and applied it to generate a French flag pattern in order to study synchronization and symmetry breaking in cellular development [HL73].

More recently, Miller and Banzhaf used what they called Cartesian genetic programming to evolve a cell program that would construct a French flag pattern [MB03]. They tested the robustness of their programs by manually removing parts of the developing pattern. They found that some of their evolved programs could repair to some extent the damaged patterns. Bowers also used this problem to study the phenotypic robustness of his embryogeny model, which was based on cellular growth with diffusing chemicals as signaling molecules [Bow05].

Gordon and Bentley proposed a development model based on a set of rules that described how development should proceed [GB05]. A set of rules evolved by a GA was used to develop a French flag pattern. The morphogenic model based on a multiagent system developed by Beurier *et al.* also used an evolved set of agent rules to grow French and Japanese flag patterns [BMF06]. On the other hand, Devert *et al.* proposed a neural network model for multicellular development that grew French flag patterns [DBS07]. Finally, even models for developing evolvable hardware have benefited from the French flag problem as a test case [TG07, HMB07].

4

Proposed Model

Résumé

Ce chapitre présente la contribution spécifique de l'auteur à un modèle de développement artificiel. Une série de gènes régulateurs codés au début de chaque génome artificiel constitue un RAR. Ces gènes de régulation sont suivis par une série de gènes structurels, dont chacun peut générer une forme particulière simple telle qu'un carré ou une ligne. Les différents modèles étudiés diffèrent principalement quant au type de RAR proposé.

Trois modèles de RARs différents ont été conçus et ils ont été évolués dans tous les cas au moyen d'un AG. Tous les modèles construits sont basés sur le modèle original de RAR proposé par Banzhaf.

- *Dans le **RAR basique**, chaque gène régulateur consiste en un site d'activation, un site d'inhibition et une série de régions qui peuvent produire une protéine régulatrice interagissant avec les sites régulateurs de tous les gènes.*
- *Dans le **modèle étendu de RAR**, le nombre de sites régulateurs peut être supérieur à 2 et ils peuvent se comporter en tant qu'activateur ou en tant qu'inhibiteur, selon la configuration des bits définissant le rôle du site régulateur.*
- *Dans la version finale du modèle, des structures cellulaires sont produites au moyen de l'activation et de l'inhibition sélectives de gènes du développement, sous les contraintes de **gradients des morphogènes**. Le RAR étendu détermine le temps de division cellulaire et il choisit le gène structurel à utiliser pour la reproduction, pendant que les gradients morphogénétiques contraignent la position sur laquelle les cellules peuvent se reproduire.*

Pour évaluer la performance des trois modèles proposés, leurs génomes ont été appliqués à un modèle de croissance cellulaire conçu pour produire des formes géométriques simples pour chaque gène structurel. Ce modèle est basé sur le paradigme des AC. Quatre voisinages d'interaction en 2D ont été utilisés dans l'AC : von Neumann, Moore, 2-Radial et Margolus. Dans le cas 3D, on a utilisé le voisinage Margolus en 3D. La table de transition des ACs fournit pour chaque voisinage local, l'état (vide ou occupé) de la position cible dans la grille. Les gènes structurels dans les génomes artificiels correspondent à une table de transition (résultat d'une évolution par un AG) qui produit une forme géométrique particulière. En commençant avec une cellule active au milieu de la grille, l'algorithme de l'AC est appliqué afin de permettre aux cellules actives de se reproduire d'après les règles trouvées par l'AG.

The proposed model of artificial development is presented in this chapter. In the final version of the model, cellular patterns are generated by means of the selective activation

and inhibition of development genes under the constraints of morphogenetic gradients. Cellular growth is achieved through the expression of structural genes, which are in turn controlled by an Artificial Regulatory Network (ARN) evolved by a Genetic Algorithm (GA). The ARN establishes the time at which cells can reproduce and determines which structural gene to use at each time step. At the same time, morphogenetic gradients constrain the position at which cells can replicate. The combination of the ARN and the structural genes make up the artificial cell's genome.

In the following sections, the successive phases of the model construction are presented. Each intermediate model is based on the preceding model, either to increase functionality or to overcome limitations presented in the previous model.

4.1 Cellular Growth Testbed

In order to evaluate the performance of the development program obtained with each successive incremental model, their evolved genomes were applied to a cellular growth testbed designed to generate simple geometrical shapes [CD06b]. This growth model is based on the extensively studied cellular automata (CA) paradigm.

Cellular automata are simple mathematical models that can be used to study self-organization in a wide variety of complex systems [Wol83]. CA are characterized by a regular lattice of N identical cells, an interaction neighborhood template η, a finite set of cell states Σ, and a space- and time-independent transition rule ϕ which is applied to every cell in the lattice at each time step [DD05].

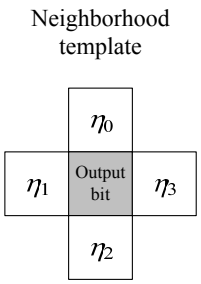
In the CA model used in this work, a cell can become active only if there is already an active cell in the interaction neighborhood. Thus, a new active cell can only be derived (reproduced) from a previously active cell in the interaction neighborhood, i.e. no spontaneous generation is allowed, as in actual biological systems.

In this work two different regular lattices with non-periodic boundaries were tried, a

2D and a 3D lattice. For 2D neighborhoods, a 33×33 cell lattice was used, while for the 3D neighborhood a cubic lattice of side length 17 was chosen. The set of cell states was defined as $\Sigma = \{0, 1\}$, where 0 can be interpreted as an empty cell and 1 as an occupied or active cell. For 2D shapes, four different interaction neighborhood templates η were considered, while only one neighborhood was studied in 3D. The interaction neighborhoods are described in the following subsections. The CA's rule ϕ was defined as a lookup table that determined, for each local neighborhood, the state (empty, occupied) of the objective cell at the next time step [MCD96, HCM98]. For a binary-state CA, these update states are termed the rule table's "output bits". The lookup table input was defined by the binary state value of cells in the local interaction neighborhood, where 0 meant an empty cell and 1 meant an occupied cell.

All the neighborhoods considered in the cellular growth testbed are *outer* interaction neighborhoods, since the objective cell is not considered as part of the interaction neighborhood. This simplification was made given that the CA rule is applied only to empty cells, implicitly assuming that all rules that have the state value 1 in the objective cell, also have the value 1 as output. That is, a cell that is already occupied by an active cell has no place to hold another active cell.

Figure 4.1 shows an example of the relationship between a CA neighborhood template and the corresponding lookup table. For each neighborhood configuration, the output bit determines whether or not a cell is to be placed at the corresponding objective cell position. In this example, if there is only an active cell at the objective cell's right position, then the objective cell is to be filled with an active cell (second row of the lookup table in Fig. 4.1). The actual output bit values used have to be determined for each different shape and are found using a genetic algorithm.



Lookup Table

η_0	η_1	η_2	η_3	Output bit
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
		⋮		⋮
1	1	1	1	0

FIG. 4.1 – Relationship between a cellular automaton neighborhood template and the corresponding lookup table. The output bit values shown are used only as an example.

4.1.1 2D Neighborhoods

Four types of 2D interaction neighborhoods were used in this work : von Neumann, Moore, 2-Radial, and Margolus neighborhoods (Fig. 4.2).

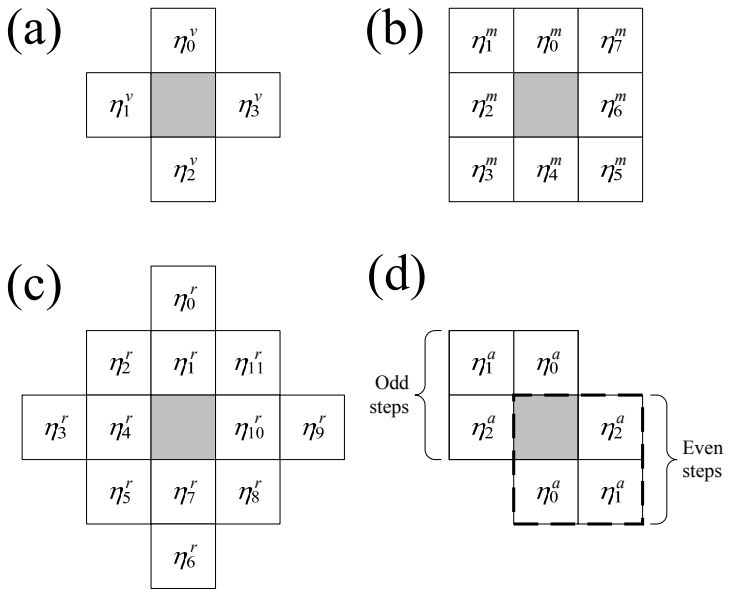


FIG. 4.2 – Interaction neighborhoods. (a) Von Neumann, (b) Moore, (c) 2-Radial, and (d) Margolus. The objective cell is depicted in gray.

Von Neumann Neighborhood

In the von Neumann neighborhood the cells at the top, left, bottom, and right of the objective cell make up the interaction neighborhood. The CA lookup table input ϕ is defined by the binary values of cells $\eta_0^v\eta_1^v\eta_2^v\eta_3^v$ of the neighborhood indicated in Fig. 4.2(a).

Moore Neighborhood

In the Moore neighborhood the nearest eight cells around the objective cell define the interaction neighborhood. The lookup table input is defined by the binary values of $\eta_0^m\eta_1^m\eta_2^m\eta_3^m\eta_4^m\eta_5^m\eta_6^m\eta_7^m$ of the neighboring cells shown in Fig. 4.2(b).

2-Radial Neighborhood

This interaction neighborhood is composed by all the cells within a radius of two cells of length from the objective cell. Formally,

$$\eta^r = \Big\{(c_x, c_y) : c_x, c_y \in \{-2,-1,0,1,2\} \wedge \sqrt{c_x^2 + c_y^2} \leq 2\Big\} - \{(0,0)\}$$

The lookup table input is defined by the binary values of $\eta_0^r\eta_1^r\eta_2^r\eta_3^r\eta_4^r\eta_5^r\eta_6^r\eta_7^r\eta_8^r\eta_9^r\eta_{10}^r\eta_{11}^r$, where η_0^r to η_{11}^r are as indicated in Fig. 4.2(c).

Margolus Neighborhood

In the Margolus neighborhood there is an alternation of the block of cells considered at each step of the CA algorithm. At odd steps, the cells at the top, upper left, and left of the objective cell constitute the interaction neighborhood, while at even steps the neighborhood is formed by the mirror cells of the previous block (see Fig. 4.2(d)). The lookup table input is defined by the binary values of $\varsigma\eta_0^a\eta_1^a\eta_2^a$, where ς is defined as 0 for odd steps of the CA algorithm and as 1 for even steps, and η_0^a, η_1^a and η_2^a are the binary values of the neighboring cells indicated in Fig. 4.2(d).

4.1.2 3D Neighborhood

The neighborhood chosen for working in 3D was the Margolus neighborhood, which has been previously used with success in modeling 3D shapes [ATTY99, WWW04]. As in the 2D case, each cell belongs to different blocks at odd and even steps. The input of the lookup table is defined by the values of $\varsigma\eta''_0\eta''_1\eta''_2\eta''_3\eta''_4\eta''_5\eta''_6$, where ς is defined as in the 2D case and η''_0 to η''_6 are as indicated in Fig. 4.3.

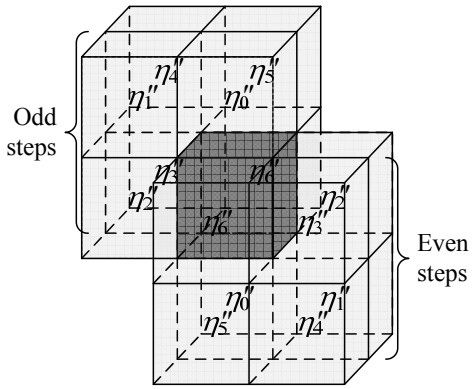


FIG. 4.3 – 3D Margolus neighborhood. The objective cell is depicted as a dark cube.

Results from simulations using these neighborhoods are presented in Chapter 5. From these results and for the sake of simplicity and shorter simulation times, only 2D cell patterns were considered for the rest of the model construction, using an outer Moore neighborhood template in the cellular growth testbed. However, all the principles that apply for the 2D case could be extrapolated to the 3D case in a straightforward manner, although with much longer simulation times.

4.1.3 NetLogo Models

NetLogo is a programmable modeling environment based on StarLogo that can be used to simulate natural and social phenomena [Wil99]. It works by giving instructions to hundreds or thousands of independent "agents" all operating concurrently. It is well suited to study emergent properties in complex systems that result from the interaction

of simple but often numerous entities. The version used in this work was *NetLogo 3-D Preview 1*, which has extensions from the regular NetLogo that allow modeling a virtual 3D environment.

For all simulations, the CA algorithm at study was implemented as a NetLogo model. For each of the neighborhoods studied and for each of the successive models proposed, a NetLogo model was built. Each cell position is defined by its Cartesian coordinates with the origin at the center of the lattice. Starting with an active cell in the middle of the lattice, the CA algorithm was applied allowing active cells to reproduce according to the CA rule table and until the indicated number of iterations was attained. Asynchronous updating of cells was originally chosen for the CA implementation, as it had been reported to give more biological-like results [SdR99]. However, in later models, synchronous updating was used for the sake of reproducibility.

4.2 Morphogenetic Gradients

Since Turing's influential article on the theoretical effect of diffusing chemical substances on an organism's pattern development [Tur52], the role of these molecules has been confirmed in a number of biological systems. These organizing substances have been termed *morphogens* due to their role in driving morphogenetic processes. In the final development model presented in this chapter, morphogenetic gradients were generated similar to those found in the eggs of the fruit fly *Drosophila*, where orthogonal gradients offer a sort of Cartesian coordinate system [CGW04]. These gradients provide reproducing cells with positional information in order to facilitate the spatial generation of patterns. The artificial morphogenetic gradients were set up as suggested in [Mei82], where morphogens diffuse from a source towards a sink, with uniform morphogen degradation throughout the gradient.

Before cells were allowed to reproduce in the cellular growth testbed, morphogenetic

gradients were generated by diffusing the morphogens from one of the CA boundaries for 1000 time steps. Initial morphogen concentration level was set at 255 arbitrary units, and the source was replenished to the same level at the beginning of each cycle. The sink was set up at the opposite boundary of the lattice, where the morphogen level was always set to zero. At the end of each time step, morphogens were degraded at a rate of 0.005 throughout the CA lattice. Two orthogonal gradients in the CA lattice were defined, one generated from left to right and the other from top to bottom (Fig. 4.4).

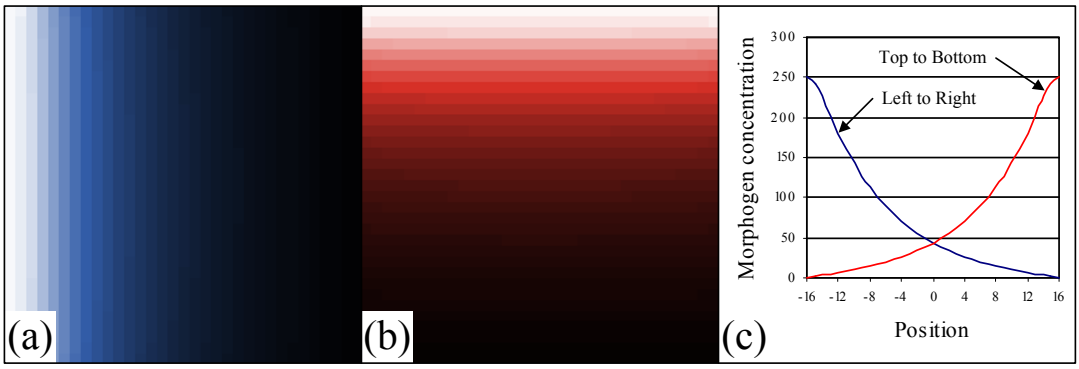


FIG. 4.4 – Morphogenetic gradients (a) Left to Right ; (b) Top to Bottom ; (c) Morphogen concentration graph.

4.3 Genomes

Genomes are the repository of genetic information in living organisms. They are encoded as one or more chains of DNA, and they regularly interact with other macromolecules, such as RNA and proteins. Artificial genomes are typically coded as strings of discrete data types. The genomes used in the following models were defined as binary strings starting with a series of regulatory genes, followed by a number of structural genes.

The series of regulatory genes at the beginning of each artificial genome presented in this chapter constitutes an Artificial Regulatory Network (ARN). For the sake of simplicity, the term "regulatory gene" is used in these models to comprise both the elements controlling protein expression and the regions coding for the regulatory protein. The mo-

dels presented next differ mainly on the type of ARN proposed. On the other hand, structural genes code for the particular shape grown by the reproducing cells and they will be described in more detail in Section 4.3.2.

4.3.1 Artificial Regulatory Networks

In nature, gene regulatory networks have been found to be a central component of an organism's genome. They actively participate in the regulation of development and in the control of metabolic functions in living organisms [Dav06]. Artificial Regulatory Networks on the other hand are computer models whose objective is to emulate to some extent the gene regulatory networks found in nature. ARNs have previously been used to study differential gene expression either as a computational paradigm or to solve particular problems [Egg97b, Rei99, Ban03, KB04, STK05, KNSQ06].

Basic ARN

The first ARN model presented here (shown as the series of regulatory genes of the genome in Fig. 4.5) is based on the ARN proposed by Banzhaf [Ban03]. However, unlike the ARN model developed by this author, the ARN implemented in the present work does not have promoter sequences and there are no unused intergene regions. All regulatory genes are adjacent and have predefined initial and end positions. Furthermore, the number of regulatory genes is fixed.

Each of the regulatory genes consists of an inhibition site, an enhancer site and a series of regulatory protein coding regions (Fig. 4.5). The latter "translate" these regions into a regulatory protein using the majority rule, i.e. for each bit position in the protein coding regions, the number of 1's and 0's is counted and the bit that is in majority is translated into the regulatory protein. The inhibition site, the enhancer site and the individual protein coding regions all have the same size in bits. Thus the protein translated from the corresponding coding regions can be compared on a bit by bit basis with the inhibitor and

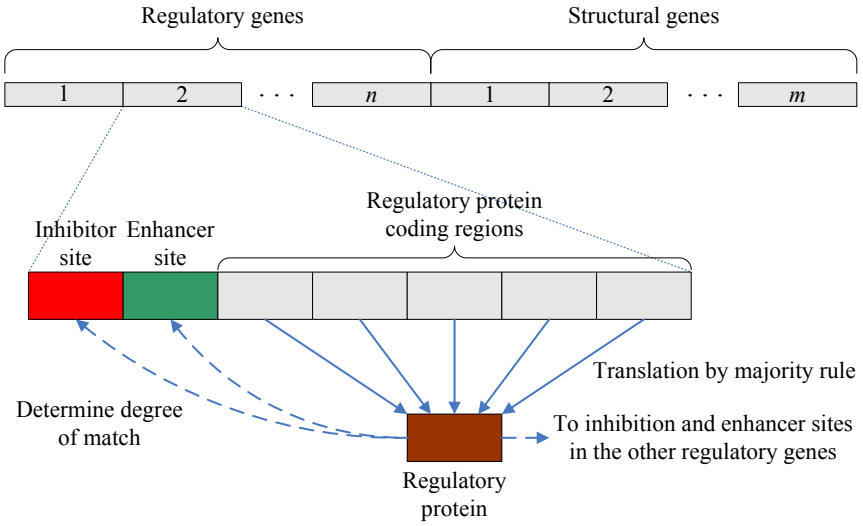


FIG. 4.5 – Genome structure of the basic model.

enhancer sites and the degree of matching can be measured. As in [Ban03], the comparison was implemented by an XOR operation, which results in a "1" if the corresponding bits are complementary.

Each translated protein is compared with the inhibition and enhancer sites of all the regulatory genes in order to determine the degree of interaction in the regulatory network. The influence of a protein on an enhancer or inhibitor site is exponential with the number of matching bits. The strength of enhancement en or inhibition in for gene i with $i = 1, ..., n$ is defined as

$$en_i = \frac{1}{n} \sum_{j=1}^{n} c_j e^{\beta\left(u_{ij}^{+} - u_{\max}^{+}\right)} \tag{4.1}$$

$$in_i = \frac{1}{n} \sum_{j=1}^{n} c_j e^{\beta\left(u_{ij}^{-} - u_{\max}^{-}\right)}, \tag{4.2}$$

where n is the total number of regulatory genes, c_j is the concentration of protein j, β is a constant that fine-tunes the strength of matching, u_{ij}^{+} and u_{ij}^{-} are the number of matches between protein j and the enhancer and inhibitor sites of gene i, respectively, and $u_{\max}^{+}$ and $u_{\max}^{-}$ are the maximum matches achievable between a protein and an enhancer or

inhibition site, respectively [Ban03].

Once the *en* and *in* values are obtained for all regulatory genes, the corresponding concentration change for protein i in one time step is found using

$$\frac{dc_i}{dt} = \delta \left(en_i - in_i\right) c_i, \tag{4.3}$$

where δ is a constant that regulates the degree of protein concentration change. Protein concentrations are updated and if a new protein concentration results in a negative value, the protein concentration is set to zero. Protein concentrations are then normalized so that total protein concentration is always the unity. Parameters β and δ were set to 1.0 and 1.0×10^6, respectively [CD07c], as will be explained in the chapter of results.

Genome size in bits is dependent on the number and size of its component genes, and in this basic model it was defined as

$$GenomeSize_1 = n \times [(2 + k) \times r] + (m \times s), \tag{4.4}$$

where n is the number of regulatory genes, k is the number of regulatory protein coding regions, r is the region size in bits, m is the number of structural genes ($m \leq n$), and s is the structural gene size in bits. For all simulations the following parameter values were used : $n = 10$, $k = 5$, $r = 32$ and $s = 256$. The number of structural genes m took values from 1, 2, 3 or 6, depending on the experiment performed, as explained in Chapter 5. The number of regulatory genes n was chosen as 10 because this figure was within the range of values previously reported for this kind of ARN [Ban03], and it was found that this value gave a desirable behavior in the protein concentration variations needed to control cell reproduction. Parameter values for k and s are equal to those used in [Ban03]. The value of 256 for parameter s results from the use of an outer Moore neighborhood for the CA lookup table that corresponds to the structural gene, that is $s = 2^8 = 256$, as will be described in Section 4.3.2.

Extended ARN

As explained in Chapter 5, the basic ARN genome could not reliably synchronize more than three structural genes, so it was decided that the model should be extended to overcome this limitation [CD07d, CD07b]. The extended genome is shown in Fig. 4.6.

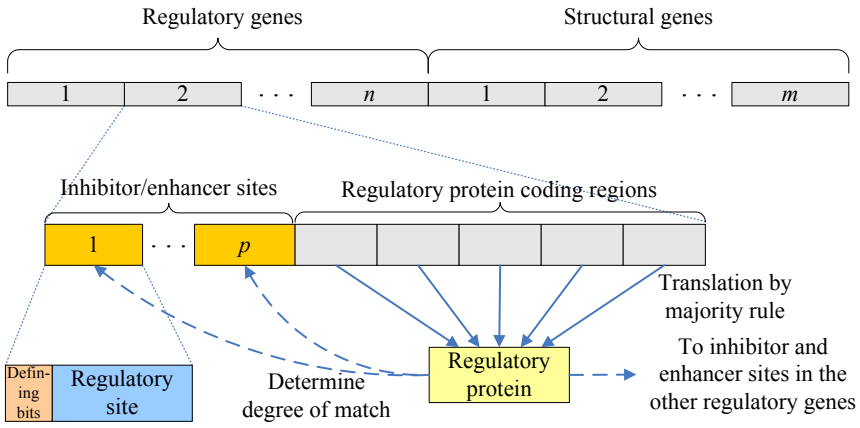


FIG. 4.6 – Genome structure of the extended model.

The basic ARN model only considered one inhibitor and one enhancer site for each regulatory gene. However, in the extended model the number of regulatory sites can be more than two and, more significantly, they have no predefined function. They can behave either as an enhancer or an inhibitor, depending on the configuration of the defining bits associated with the regulatory site (Fig. 4.6). If there are more 1's than 0's in the defining bits region, then the site functions as an enhancer, but if there are more 0's than 1's, then the site behaves as an inhibitor. Finally, if there is an equal number of 1's and 0's, then the regulatory site is turned off. This means that the regulatory site role as an enhancer or as an inhibitor can be evolved by the GA. Furthermore, if the number of function defining bits is even, then the regulatory site can be turned on and off. The number of regulatory sites was extended with respect to the original model, in order to more closely follow what happens in nature, where biological regulatory genes involved in development typically have several regulatory sites associated with them [Dav06].

Each regulatory gene in the extended model consists of a series of inhibitor/enhancer

sites and a series of regulatory protein coding regions (Fig. 4.6). As in the basic ARN, these regions translate a protein using the majority rule which can be compared bit by bit with the regulatory sites by means of an XOR operation.

The equations that calculate the degree of gene enhancement or inhibition had to be slightly modified to account for the variable number of inhibitor and enhancer sites. The strength of excitation en or inhibition in for gene i with $i = 1, ..., n$ is thus defined as

$$en_i = \frac{1}{v} \sum_{j=1}^{v} c_j e^{\beta\left(u_{ij}^{+} - u_{\max}^{+}\right)} \tag{4.5}$$

$$in_i = \frac{1}{w} \sum_{j=1}^{w} c_j e^{\beta\left(u_{ij}^{-} - u_{\max}^{-}\right)}, \tag{4.6}$$

where v and w are the total number of enhancer and inhibitor sites, respectively. The rest of parameters and constants are the same as in the basic model.

Once the en and in values are obtained for all regulatory genes, the corresponding change in concentration for protein i was calculated as in the basic model. Parameters β and δ were also set as in the previous model.

Genome size in bits for the extended model is calculated using

$$GenomeSize_2 = n \times \{[(p + k) \times r] + (p \times d)\} + (m \times s), \tag{4.7}$$

where n is the number of regulatory genes, p is the number of inhibitor/enhancer sites per regulatory gene, k is the number of regulatory protein coding regions, r is the regulatory/protein region size in bits, d is the number of function defining bits, m is the number of structural genes ($m \leq n$), and s is the structural gene size in bits. The number of structural genes m took values from 3, 4 or 8, depending on the experiment performed, and parameters p and d were varied as explained in Chapter 5. The rest of the parameters were set the same as in the basic model.

Extended ARN with Morphogenetic Fields

In this final ARN model, each regulatory gene consists of a series of eight inhibitor/enhancer sites, a series of five regulatory protein coding regions, and two morphogen threshold activation sites that determine the allowed positions for cell reproduction (Fig. 4.7). Inhibitor/enhancer sites are composed of a 12-bit function defining region and a regulatory site. As in the previous model, regulatory sites can behave either as an enhancer or an inhibitor, depending on the configuration of the function defining bits associated with them.

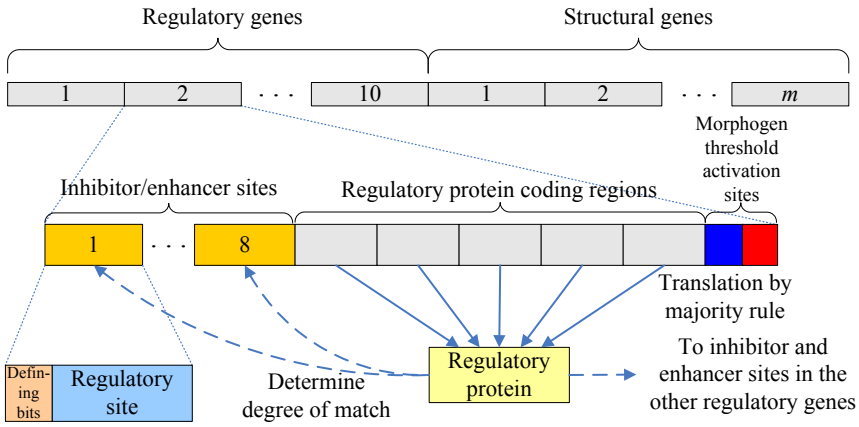


FIG. 4.7 – Genome structure of the extended model with morphogenetic fields.

The parameter values used for the number of inhibitor/enhancer sites and the number of function defining bits are those that gave the best results under the conditions tested (Chapter 5). The values for the rest of the parameters are the same as in the extended model [CD07a].

As in the previous models, the regulatory protein coding regions are translated into a protein that is matched against all the regulatory sites in the ARN. Protein concentration levels are calculated as in the extended ARN model.

The morphogen threshold activation sites provide reproducing cells with positional information as to where they are allowed to grow in the CA lattice. There is one site for each of the two orthogonal morphogenetic gradients described in Section 4.2. These

sites are 9 bits in length, where the first bit defines the allowed direction (above or below the threshold) of cellular growth, and the next 8 bits code for the morphogen threshold activation level, which ranges from 0 to $2^8 - 1 = 255$. If the site's high order bit is 0, then cells are allowed to replicate below the morphogen threshold level coded in the lower order eight bits; if the value is 1, then cells are allowed to reproduce above the threshold level. Since in a regulatory gene there is one site for each of the two orthogonal morphogenetic gradients, for each pair of morphogen threshold activation levels, the pair of high order bits defines in which of the four relative quadrants cells expressing the associated structural gene can reproduce. Quadrants can have irregular edges because morphogenetic gradients are not perfectly generated due to local morphogen accumulation close to the non-periodic boundaries of the CA lattice.

Genome size in bits for this model is calculated using

$$GenomeSize_3 = n \times \{[(p + k) \times r] + [p \times (d + 2h)]\} + (m \times s), \tag{4.8}$$

where h is the size in bits of a morphogen threshold activation site, and the rest of the parameters are the same as in the previous model. The number of structural genes m used in this model was 3 or 8, depending on the pattern desired.

4.3.2 Structural Genes

Structural genes code for the particular shape grown by the reproducing cells [CD06a] and they correspond to the CA rule table's output bits from the cellular growth testbed presented in Section 4.1. Previously to being attached to the regulatory genes to constitute the genome, structural genes were evolved by a GA in order to produce predefined 2D shapes.

Structural genes are always associated to the corresponding regulatory genes, that is, structural gene number 1 is associated to regulatory gene number 1 and its related

translated protein, and so on. A structural gene was defined as being active if and only if the regulatory protein translated by the associated regulatory gene was above a certain concentration threshold. The value chosen for the threshold was 0.5, since the sum of all protein concentrations is always 1.0, and there can only be a protein at a time with a concentration above 0.5. As a result, only one structural gene can be expressed at a particular time step in a cell. If a structural gene is active, then the CA lookup table coded in it is used to control cell reproduction. Given that the outer Moore neighborhood used in the cellular growth testbed consists of the eight cells surrounding the central cell, structural genes are all 256 bits in length ($2^8 = 256$) [CD06a].

In the series of simulations presented in Chapter 5, the number of structural genes used in the genome depended on the particular pattern grown and this number was always less than the number of regulatory genes. Thus, some regulatory proteins both regulated concentration for other proteins and directly controlled structural gene expression, while other proteins only had a regulatory role. Structural gene expression is visualized in the cellular growth testbed as a distinct external color for the cell.

4.4 Genetic Algorithm

Genetic algorithms are search and optimization methods based on ideas borrowed from natural genetics and evolution [Hol92]. A GA starts with a population of chromosomes representing vectors in search space. Each chromosome is evaluated according to a fitness function and the best individuals are selected. A new generation of chromosomes is created by applying genetic operators on selected individuals from the previous generation. The process is repeated until the desired number of generations is reached or until the desired individual is found.

The GA in this work uses tournament selection as described in [Mit96] with single-point crossover and mutation as genetic operators. Single-point crossover consists in ran-

domly selecting two chromosomes with a certain probability called crossover rate, and then randomly selecting a single bit position in the chromosome structure. From this point on, the remaining fragments of the two chromosomes are exchanged. The resulting chromosomes then replace the original ones in the chromosome population. On the other hand, mutation consists in randomly flipping one bit in a chromosome from 0 to 1 or vice versa. The probability of each bit to be flipped is called the mutation rate.

After several calibration experiments, the parameter values described next were considered to be appropriate. The initial population consisted of either 500 binary chromosomes chosen at random for evolving the form generating genes, or 1000 chromosomes for the simulations involving the ARN models. Tournaments were run with sets of 3 individuals randomly selected from the population. Crossover rate was 0.60 in all cases, whereas the mutation was 0.015 for the evolution of structural genes, and 0.15 for the evolution of ARNs. The crossover rate of 0.60 was chosen because it was reported to give the best results when trying to evolve a binary string representing a CA using a GA [BB05]. As for the mutation rate, it was decided to use a value one order of magnitude higher in the evolution of the ARN models than the one used in the same report, for the reasons that will be given in Chapter 5. Finally, the number of generations was set at 50 in all cases, since there was no significant improvement after this number of generations.

4.4.1 Chromosome Structure

The GA experiments were run with two different types of chromosomes, the kind used for the evolution of a form generating gene (structural genes in the artificial genomes), and the one used for evolving an ARN model.

Chromosome Structure for Form Generation

The chromosome structure used for evolving a form generating gene is shown in Fig. 4.8. The *control field* codes for the number of steps in base 2 that the CA algorithm is

TAB. 4.1 – Size in bits for chromosomes used in evolving a form generating gene. Parameters are as defined in the text.

Neighborhood	l	a	b	$2^l \times a + b$
Von Neumann	4	1	4	20
Moore	8	1	4	260
2-Radial	12	1	4	4100
2D Margolus	3	2	5	21
3D Margolus	7	2	4	260

allowed to run, whereas the *action field* represents the CA lookup table's output bits in lexicographical order of neighborhood.

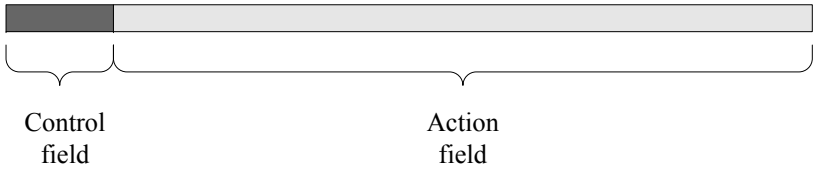


FIG. 4.8 – Chromosome structure for evolving a form generating gene.

For the initial simulations, when testing different neighborhood templates, chromosome size varied depending on the neighborhood type and the maximum number of iterations allowed for the CA. For this type of chromosome, size in bits was defined as

$$(2^l \times a) + b, \tag{4.9}$$

where l is the number of cells in the local interaction neighborhood (excluding the objective cell, which is always assumed to be 0, i.e. the CA rule is applied only to empty cells), a is the number of alternating steps in the CA algorithm (2 for the Margolus neighborhood and 1 for all the others), and b is the number of bits of the maximum number of iterations in base 2 that the CA is allowed to run. Table 4.1 shows the chromosome size for the various neighborhoods considered.

The control field size was chosen so that the shape formed on any CA run could not overflow the boundaries of the CA lattice. As mentioned in Section 4.1.3, the lattices

consisted of a square of 33×33 and a cube of $17 \times 17 \times 17$ cells, with the initial active cell at the central position. To have an active cell reach one of the lattice boundaries, the CA algorithm would be required to run for at least 16 steps in the square lattice, and 8 steps in the cubic lattice. On this ground, for all 2D neighborhoods, except 2D Margolus, the control field size was chosen to be 4, so that the CA algorithm would iterate for at most $2^4 - 1 = 15$ steps. For the Margolus neighborhoods, due to the alternation of the cell blocks forming the interaction neighborhood, the CA algorithm would require twice as many steps for an active cell to reach one of the lattice boundaries. For this reason, the chromosome's control field size was defined as 5 for the 2D Margolus neighborhood and 4 in the 3D Margolus case, so that the upper limit of iterations would be $2^5 - 1 = 31$ and $2^4 - 1 = 15$ steps, respectively.

The chromosome's action field coding for the CA rule table is of length $2^l \times a$. For the Moore neighborhood, that represents a rule space of $2^{256} \approx 10^{77}$, and for the 2-Radial neighborhood the rule space size is $2^{4096} \approx 10^{1233}$. In both cases the search space is far too large for any sort of exhaustive evaluation. And if we also take into account the bits introduced by the control field, the search space grows larger. Even for the smallest of the neighborhoods considered, the von Neumann and the 2D Margolus neighborhoods, the search space is not negligible, since it contains over one million possibilities.

In the case of the 3D CA, the 3D Margolus neighborhood was chosen over, for example, a 3D Moore neighborhood, since in the latter case the interaction neighborhood would consist of the nearest 26 cells, giving a CA lookup table of $2^{26} \approx 6.7 \times 10^7$ rows, as opposed to the $2^7 \times 2 = 256$ rows required by the 3D Margolus neighborhood.

Chromosome for evolving ARNs

When evolving the ARNs with the goal of synchronizing the expression of structural genes, the chromosomes used for the GA runs were simply the ARN chains themselves. Chromosome size in this case depended on the ARN model and the values of the parame-

TAB. 4.2 – Size in bits for chromosomes used in evolving an ARN.

ARN Model	Formula	Chromosome size
Basic	$10 \times [(2+5) \times 32]$	2240
Extended	$10 \times \{[(8+5) \times 32] + (8 \times 12)\}$	5120
With Morph. fields	$10 \times \{[(8+5) \times 32] + [8 \times (12 + (2 \times 9))]\}$	6560

ters chosen. Table 4.2 presents the chromosome sizes with the parameter values actually used. The formulas presented in the preceding sections for genome size were used to calculate chromosome size, only excluding the term $m \times s$ that represents the structural gene region size.

The basic ARN binary string, which is the smallest ARN tested, has a size of 2240 bits, which represents a search space of $2^{2240} \approx 2 \times 10^{674}$ vectors. Evidently, search space grows exponentially with the number of regulatory genes. But even for the simplest of ARNs, the one consisting of only two regulatory genes, the search space has a size of $2^{448} \approx 7 \times 10^{134}$, which is still too large to be explored deterministically. It should be evident that the search space for any of the ARN models considered is far too large for any method of exhaustive assessment. Therefore, the use of an evolutionary search algorithm for finding an appropriate synchronization of gene expression is amply justified.

4.4.2 Fitness function

As in the case of the chromosome structure, there were two different fitness functions used, depending on whether one o more structural genes were considered.

One structural gene

The fitness function for the simulations involving the evolution of one structural gene is the same as the function used by de Garis ([de 99]) and it has been already presented in Section 3.6, but is reproduced here for ease of consultation :

$$Fitness = \frac{ins - \frac{1}{2}outs}{des}, \tag{4.10}$$

where ins is the number of filled cells inside the desired shape, $outs$ is the number of filled cells outside the desired shape, and des is the total number of cells inside the desired shape. Thus, a fitness value of 1 represents a perfect match.

During the course of a GA experiment, each chromosome produced in a generation was fed to the corresponding NetLogo model, which was allowed to run for as many iterations as indicated in the chromosome's control field. Fitness was evaluated after the model stopped and a shape was formed. This process continued until the maximum number of generations was reached and then the best individual was selected.

Multiple structural genes

In the case of the evolution of ARNs that synchronized the expression of more than one structural gene, the fitness function used by the GA was defined as

$$Fitness = \frac{1}{c}\sum_{i=1}^{c}\frac{ins_i - \frac{1}{2}outs_i}{des_i}, \tag{4.11}$$

where c is the number of different colored shapes, each corresponding to an expressed structural gene, ins_i is the number of filled cells inside the desired shape i with the correct color, $outs_i$ is the number of filled cells outside the desired shape i, but with the correct color, and des_i is the total number of cells inside the desired shape i. In consequence, a fitness value of 1 represents again a perfect match. This fitness function is an extension of the one used in [de 99], where the shape produced by only one "gene" was considered. To account for the expression of several structural genes, the combined fitness values of all structural gene products were introduced in the fitness function used.

During a GA run, each chromosome produced in a generation was fed to the corresponding NetLogo model, where the previously evolved structural genes were attached and

the cells were allowed to reproduce controlled by the ARN found by the GA. Fitness was evaluated at the end of 100 time steps in the cellular growth testbed, where a colored pattern could develop. This process continued until the maximum number of generations was reached or when a fitness value of 1 was obtained.

5

Results

Résumé

Les quatre formes 2D choisies pour tester le modèle de croissance cellulaire basé sur des ACs étaient un carré, un triangle, un losange et un cercle. Les formes testées dans le cas 3D étaient un cube et une sphère. Pour produire une forme particulière, un AG a été utilisé pour trouver la table de transition et le nombre d'itérations appropriés. Après cette évolution, une cellule active est placée au milieu de la grille et l'AC s'exécute exploitant cette table de transition ainsi que le nombre d'itérations trouvés par l'AG. On présente les résultats moyens de 100 répétitions dans les ACs des meilleurs chromosomes trouvés.

Les trois modèles de réseaux de régulation (RAR basique, RAR étendu et RAR avec gradient de morphogène) ont subi une évolution par un AG afin d'obtenir les structures cellulaires désirées. Le but était de combiner des formes colorées simples générées par les gènes structurels afin d'obtenir des structures prédéfinies. Après qu'un RAR ait été obtenu et que les gènes structurels précédemment évolués aient été agrégés pour constituer le génome artificiel, une cellule active initiale au milieu de la grille de l'AC a exécuté son comportement de reproduction, contrôlée par la séquence d'activation des gènes structurels trouvés par l'AG.

*Une fois les valeurs appropriées des paramètres trouvés pour le **RAR basique**, plusieurs séries d'expériences ont été menées pour évoluer des RARs afin d'obtenir diverses structures colorées. Une structure carrée bicolore a été obtenue facilement. Cependant, afin de créer une structure carrée à trois couleurs, au lieu d'utiliser une série de trois gènes structurels qui convenait pour un carré, un tandem de deux séries de trois gènes structurels a dû être employé pour augmenter la probabilité de trouver un RAR approprié par l'AG. En utilisant la même approche de la duplication des gènes structurels, la structure d'un drapeau français a été obtenue. Dans le modèle de **RAR étendu**, pour tester un nombre variable de sites régulateurs ainsi qu'un nombre variable de bits codant les activateurs ou inhibiteurs, une série d'expériences a été menée. Les tests ont consisté à trouver un RAR apte à développer un carré à trois couleurs dans le modèle de croissance cellulaire, en faisant variant le nombre des sites régulateurs et le nombre des bits qui définit le rôle des sites. Les autres valeurs des paramètres pour l'AG et les RARs ont été fixées comme dans le modèle basique. Dans toutes les simulations exécutées ultérieurement, il a été retenu 12 bits pour le rôle des sites activateurs ou inhibiteurs et 8 bits pour le nombre de site régulateur. Ces valeurs ont fourni les meilleurs résultats dans les conditions testées. Les structures d'un carré à quatre couleurs et d'un drapeau français avec une hampe ont été obtenues en utilisant l'approche de duplications des gènes structurels. Enfin, pour tester l'effet de l'**addition de morphogènes** dans le modèle de RAR étendu, le modèle de croissance cellulaire a encore été appliqué à la création de la structure d'un drapeau français avec et sans la hampe. La duplication de gènes structurels ici aussi était rendue nécessaire afin d'obtenir la structure du drapeau avec la hampe.*

Results are divided in two main parts. In the first part, the framework for obtaining form generating genes was established, whereas in the second part the different ARN models were tested for their ability to synchronize the expression of structural genes in order to obtain a desired pattern.

5.1 Form Generation

In all cases, the GA described in Section 4.4 was used to evolve the lookup table and the number of iterations for the desired shapes. Starting with one active cell in the middle of the CA lattice in the NetLogo model, cells were allowed to reproduce (sprout an active cell from a previously active cell) using the lookup table found by the GA and for as many iterations as indicated in the chromosome's control field. Since the CA algorithm used asynchronous updating with the order of reproduction of cells randomly selected, a particular shape and fitness could slightly change on different runs of the CA algorithm for the same chromosome. For this reason, fitness mean and standard deviation from 100 runs of the CA algorithm are reported for all final chromosomes.

5.1.1 2D shapes

The desired shapes are shown in Figure 5.1. These shapes were chosen for their simplicity and familiarity. The square has a side length of 21 cells, the diamond has a length of 11 cells from the center to any of its corners, the triangle has a base and a height of 23 and 21 cells, respectively, and finally the circle has a radius of 11 cells.

Table 5.1 presents the fitness mean and standard deviation from 100 runs in the NetLogo model of the final chromosomes for all the shapes and neighborhoods considered. A comparative chart of the mean fitness values presented in Table 5.1 is shown in Figure 5.2, grouped by interaction neighborhood.

At the initial stages of this work, it was assumed that, since all shapes considered had

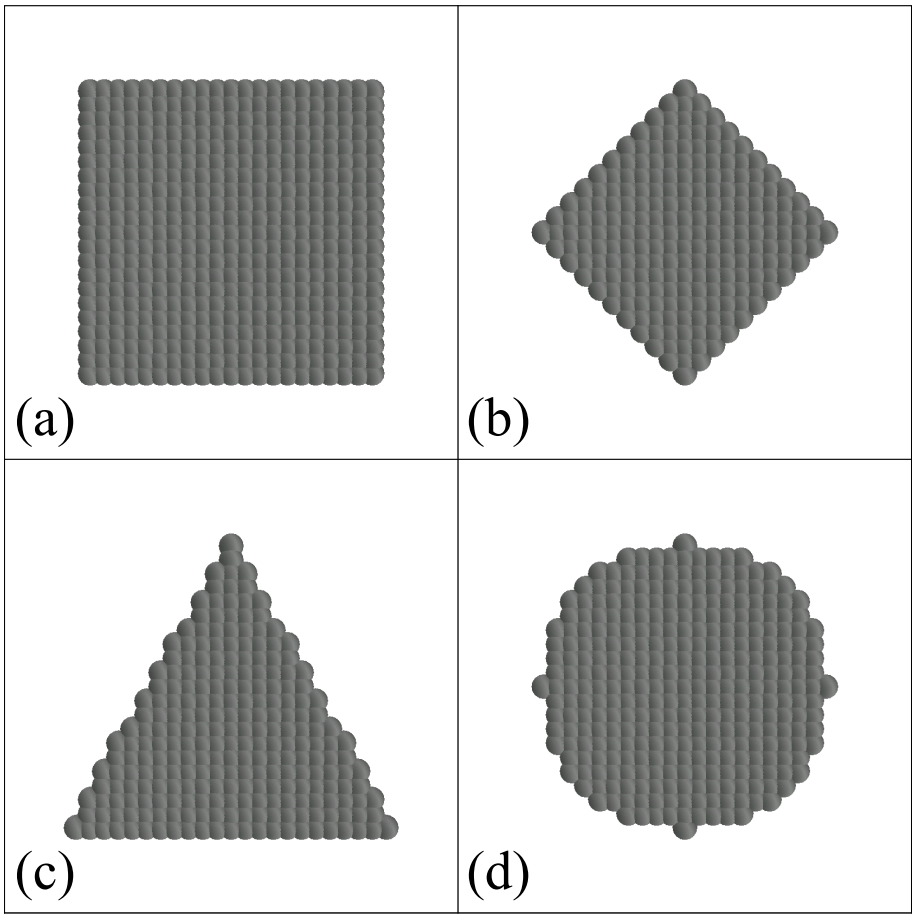


FIG. 5.1 – Desired shapes. (a) Square, (b) Diamond, (c) Triangle, and (d) Circle.

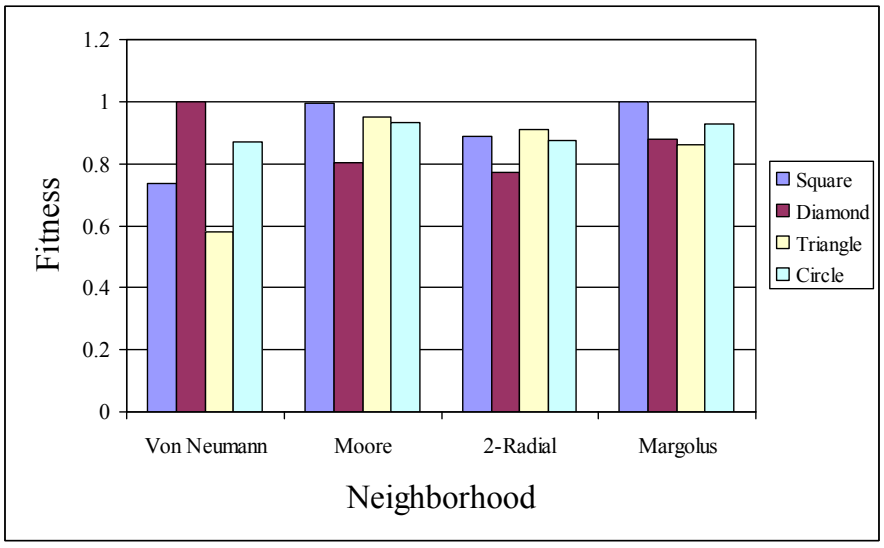


FIG. 5.2 – Mean fitness comparative chart for all neighborhoods and shapes.

TAB. 5.1 – Fitness mean ($\bar{x}$) and standard deviation (σ) from 100 runs of the CA algorithm for the final chromosomes.

	Von Neumann		**Moore**		**2-Radial**		**Margolus**	
Shape	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Square	0.738	0.008	0.993	0.003	0.887	0.015	1.000	0.000
Diamond	1.000	0.000	0.805	0.040	0.773	0.028	0.880	0.018
Triangle	0.580	0.000	0.950	0.011	0.909	0.023	0.860	0.010
Circle	0.868	0.000	0.932	0.013	0.875	0.017	0.928	0.006
Average	**0.797**	**0.002**	**0.920**	**0.017**	**0.861**	**0.021**	**0.917**	**0.008**

dimensions such that the outer active cells could be reached in 10 steps, the number of iterations should be fixed at 10 steps for the von Neumann, Moore and 2-Radial neighborhoods, and 20 steps for the Margolus neighborhood. However, it was later decided that, in order to avoid a preconceived notion of how the evolved chromosomes should work, the GA should also find the optimum number of iterations needed to generate a particular shape. For this reason the control field was introduced in the chromosome definition. Table 5.2 presents the evolved number of iterations (coded in the control field) of the final chromosomes for all shapes and neighborhoods.

TAB. 5.2 – Evolved number of iterations for the final chromosomes for all neighborhoods and shapes.

Shape	**Von Neumann**	**Moore**	**2-Radial**	**Margolus**
Square	15	10	11	20
Diamond	10	12	9	19
Triangle	12	10	10	19
Circle	13	9	10	18

Figures 5.3 to 5.6 show results from some of the best runs for the four types of shapes obtained using the four models, corresponding to each of the neighborhoods studied. For ease of visualization, cells that fall outside the desired shape are shown in light gray [CD06a].

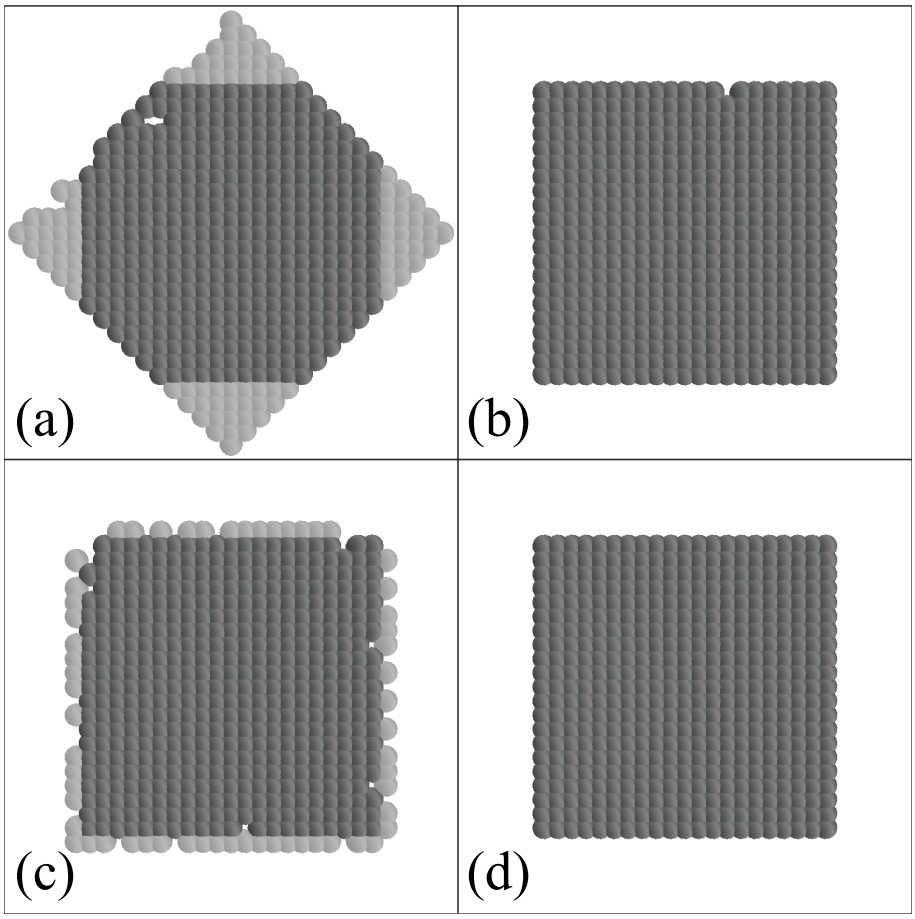


FIG. 5.3 – Square shape. Cells outside the desired shape are shown in light gray. Neighborhood description is followed by fitness value. (a) Von Neumann (0.751), (b) Moore (0.998), (c) 2-Radial (0.917), and (d) Margolus (1.000).

5.1.2 3D shapes

The desired 3D shapes are presented in Figure 5.7. The cube has a side length of 5 cells, while the sphere has a radius of 4 cells. Fitness mean and standard deviation from 100 runs in the NetLogo model for 3D shapes, as well as the evolved number of iterations for the final chromosomes, are presented in Table 5.3.

Figure 5.8 shows shapes from some of the best runs obtained using the 3D Margolus model. As in the 2D case, cells outside the desired shape are shown in light gray [CD06b].

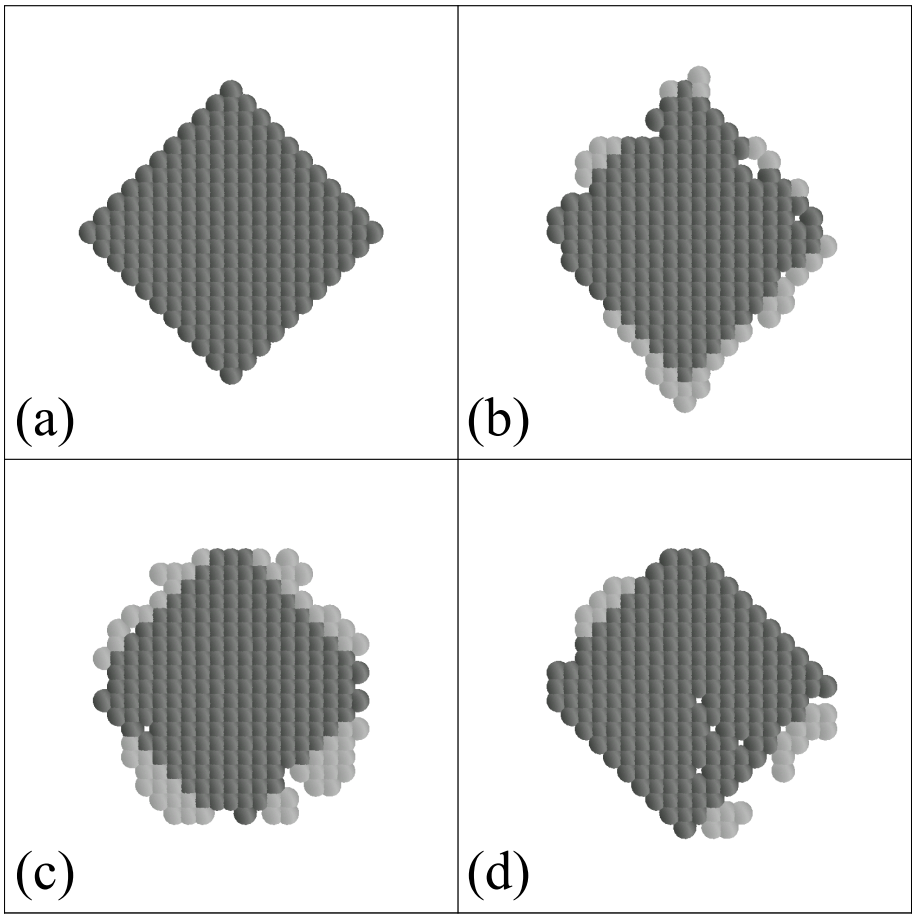


FIG. 5.4 – Diamond shape. Cells outside the desired shape are shown in light gray. Neighborhood description is followed by fitness value. (a) Von Neumann (1.000), (b) Moore (0.878), (c) 2-Radial (0.826), and (d) Margolus (0.912).

5.2 Pattern Generation

For all models, the GA described in Chapter 4 was used to evolve the ARN for the desired colored patterns. The goal was to combine different colored shapes expressed by structural genes in order to obtain a predefined pattern. After an ARN was obtained and the previously evolved structural genes were attached to constitute the artificial genome, an initial active cell in the middle of the CA lattice was allowed to reproduce controlled by the structural gene activation sequence found by the GA. In the desired patterns studied, each color represents a different structural gene being expressed. In order to achieve the desired pattern with a predefined color for each cell, the genes in the ARN had to evolve

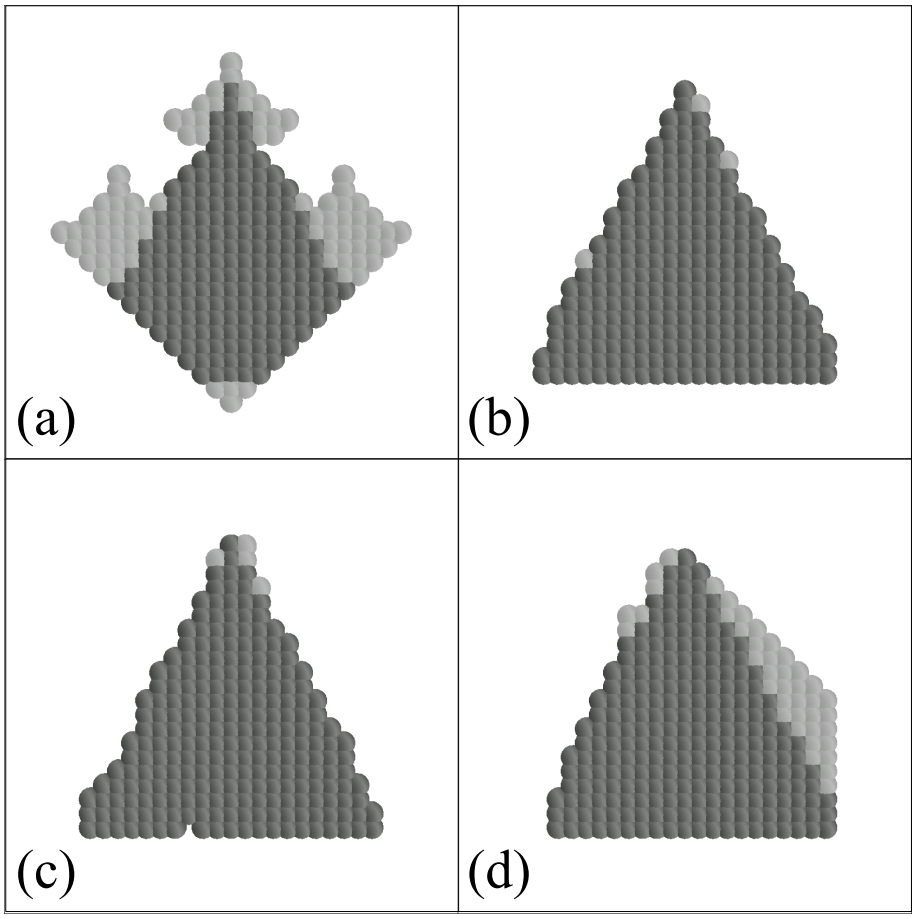


FIG. 5.5 – Triangle shape. Cells outside the desired shape are shown in light gray. Neighborhood description is followed by fitness value. (a) Von Neumann (0.580), (b) Moore (0.973), (c) 2-Radial (0.951), and (d) Margolus (0.879).

to be activated in a precise sequence and for a specific number of iterations.

5.2.1 Basic ARN

As a first step, in order to find the appropriate values for parameters β and δ used in the ARN, a series of GA experiments was performed. The tests consisted in finding an ARN to grow a 21×21 square in the cellular growth testbed in at most 10 generations, using 10 regulatory genes and one structural gene that coded for the square. The other parameters values for the GA and the ARN were set as described in Chapter 4.

Being a factor to an exponent, parameter β could not be varied widely. Thus, a range

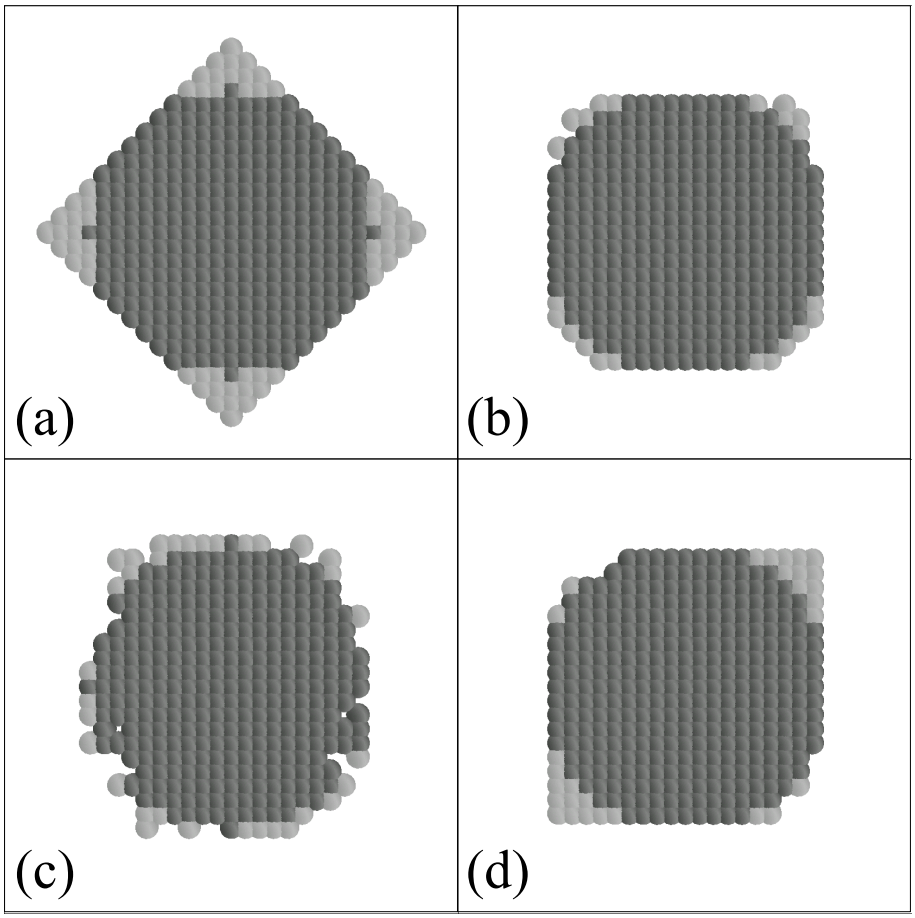


FIG. 5.6 – Circle shape. Cells outside the desired shape are shown in light gray. Neighborhood description is followed by fitness value. (a) Von Neumann (0.868), (b) Moore (0.953), (c) 2-Radial (0.901), and (d) Margolus (0.939).

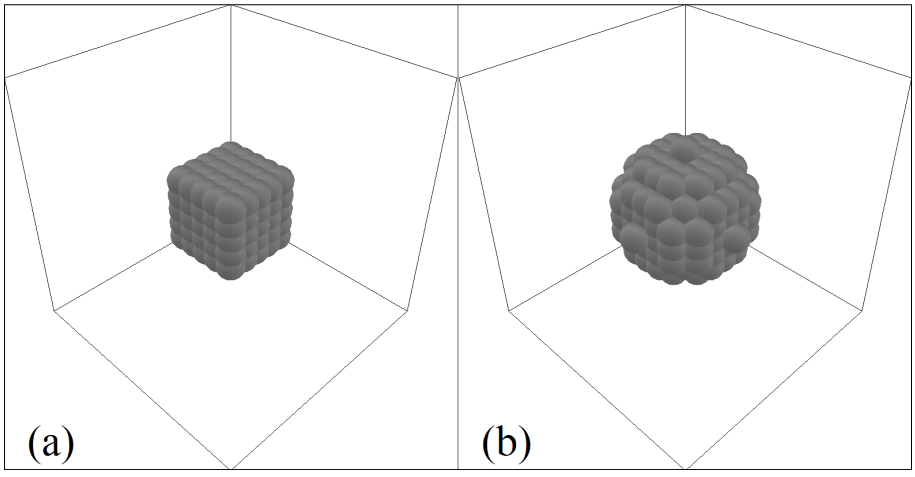


FIG. 5.7 – Desired 3D shapes. (a) Cube, and (b) Sphere.

TAB. 5.3 – Fitness mean ($\bar{x}$) and standard deviation (σ) from 100 runs of the CA algorithm for 3D shapes. The evolved number of iterations (*Iter.*) is also presented.

Shape	$\bar{x}$	σ	*Iter.*
Cube	0.9690	0.0165	4
Sphere	0.8579	0.0163	6

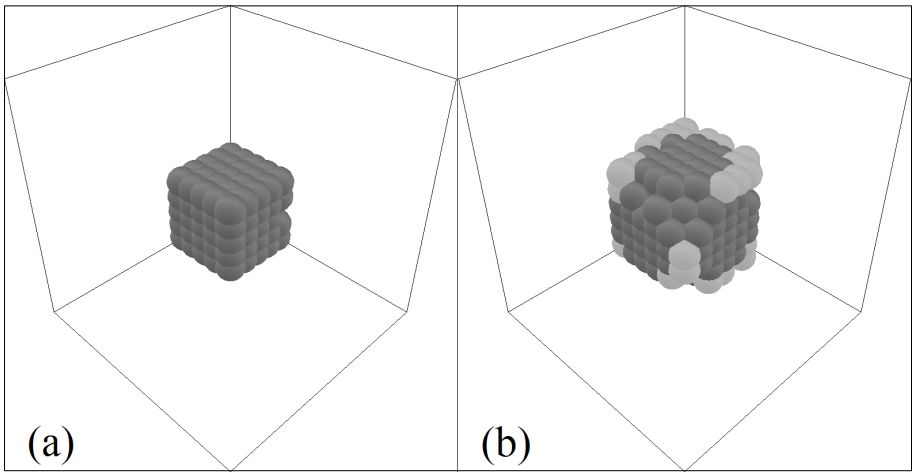


FIG. 5.8 – Shapes obtained using the 3D Margolus neighborhood model. Cells outside the desired shape are shown in light gray. Shape description is followed by fitness value. (a) Cube (0.9920), and (b) Sphere (0.8911).

of 0.5 to 5.0 in increments of 0.5 units was tried. For parameter δ, a range of 1.0×10^0 to 1.0×10^{20} with exponent increments of one unit was used. Surprisingly, parameter δ was found to be very flexible in the range of values that could successfully find the desired pattern. For a β value of 1.0, runs with parameter δ in the range from 1.0×10^6 to 1.0×10^{20} found the correct pattern under the conditions described. At the end, the values for β and δ were set to 1.0 and 1.0×10^6, respectively [CD07c].

Once the appropriate parameter values were found, several series of experiments were performed in order to evolve ARNs for various colored patterns. Not all GA experiments rendered an ARN capable of forming the desired pattern. Furthermore, for some desired patterns involving the expression of more than three structural genes, no appropriate ARN could be evolved. The graphs presented next correspond to some of those experiments where ARNs with fitness function values equal to 1.0 were found by the GA.

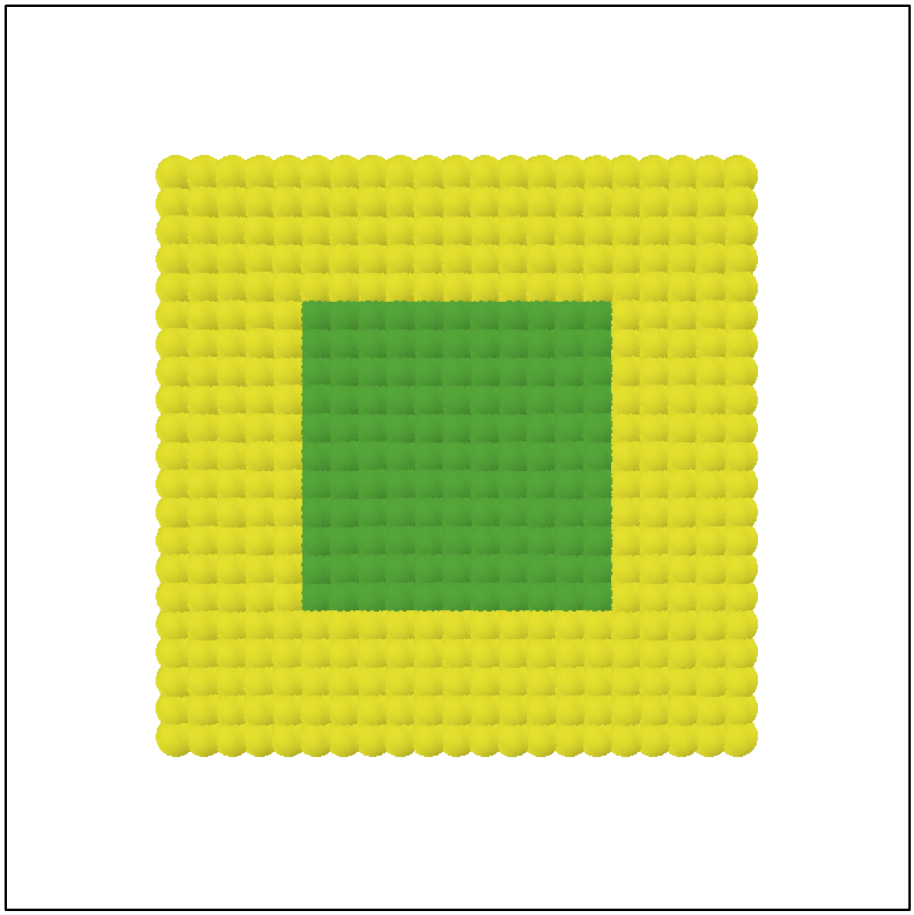


FIG. 5.9 – Two-color square.

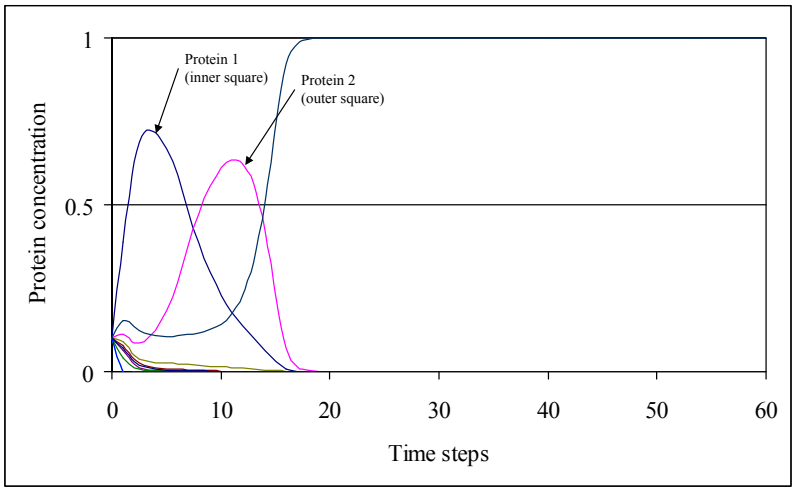


FIG. 5.10 – Graph of protein concentration change from an ARN expressing the two-color square.

Fig. 5.9 shows a two-color 21×21 square built by the expression of two structural genes, both coding for a square. The graph corresponding to the expression of the regulatory proteins of the evolved ARN is presented in Fig. 5.10. For some of the graphs shown, only the first 60 time steps are presented, as there is no significant change in these graphs after this point and until the end of the 100-step run.

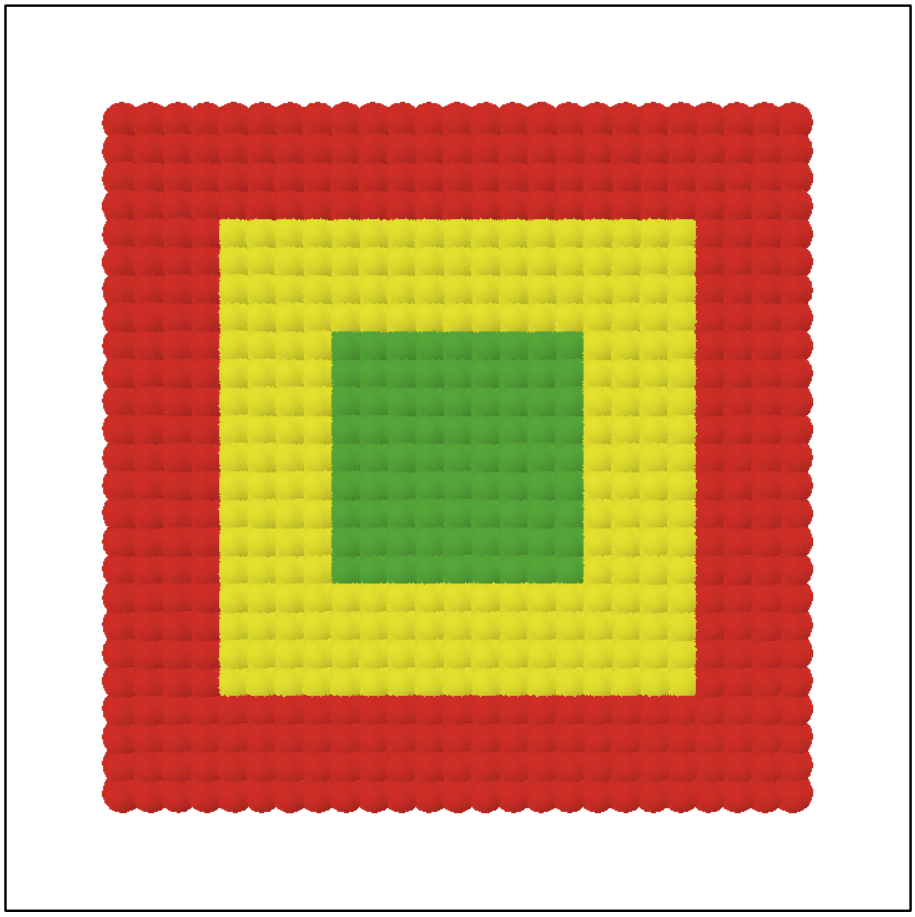


FIG. 5.11 – Three-color square.

When trying to create a three-color square expressing three identical structural genes each coding for a square, it proved difficult to synchronize the expression of the three regulatory genes in a specific sequence. In order to increase the likelihood for the GA to find an appropriate ARN, instead of using a series of three structural genes coding for a square, a tandem of two series of three structural genes was used, for a total of six structural genes. In that manner, for creating the inner square, the ARN could express

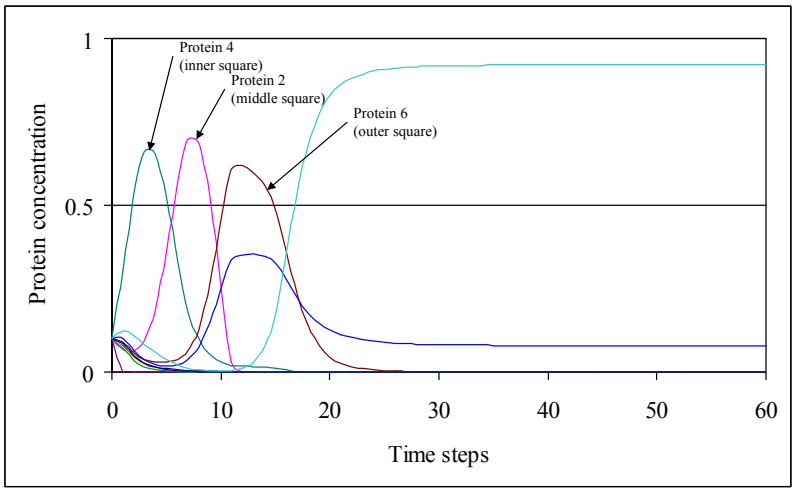


FIG. 5.12 – Graph of protein concentration change from an ARN expressing the three-color square.

either structural gene number 1 or gene number 4, for the middle square it could use genes 2 or 5, and finally for the outer square it could make use of structural genes 3 or 6. Thus, the probability of finding an ARN that could express a three-color square with a particular color order was substantially increased. Using an ARN with this configuration of structural genes, the three-color 25×25 square shown in Fig. 5.11 was obtained. The graph representing the expression of the proteins for the corresponding ARN is shown in Fig. 5.12. Note the order in which the structural genes were expressed, starting with the first gene from the second series of structural genes, followed by the second gene from the first series, and ending with the third gene from the second series of structural genes.

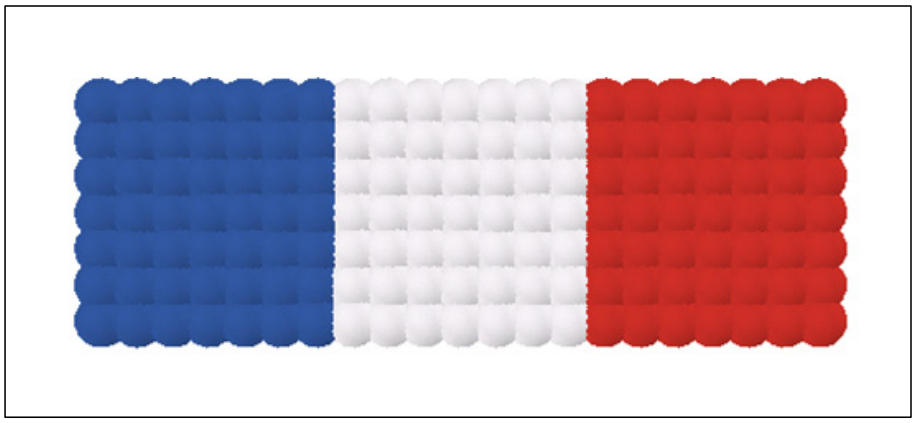


FIG. 5.13 – French flag (21×7).

These patterns were chosen so that the different structural genes were expressed for the same number of time steps in the cellular growth model. For the two-color square, each structural gene is expressed for five time steps, whereas for the three-color square, each of the three genes involved is activated for exactly four time steps.

In order to explore the result of combining different structural genes that are expressed for a different number of time steps, three different genes were used to grow a French flag pattern. One gene drove the creation of the central white square, while the other two genes extended the central square to the left and to the right, expressing the blue and the red color, respectively. The last two structural genes do not code specifically for a square, instead they extend a vertical line of cells to the left or to the right for as many time steps as they are activated, and they were obtained using the framework for evolving form generating genes.

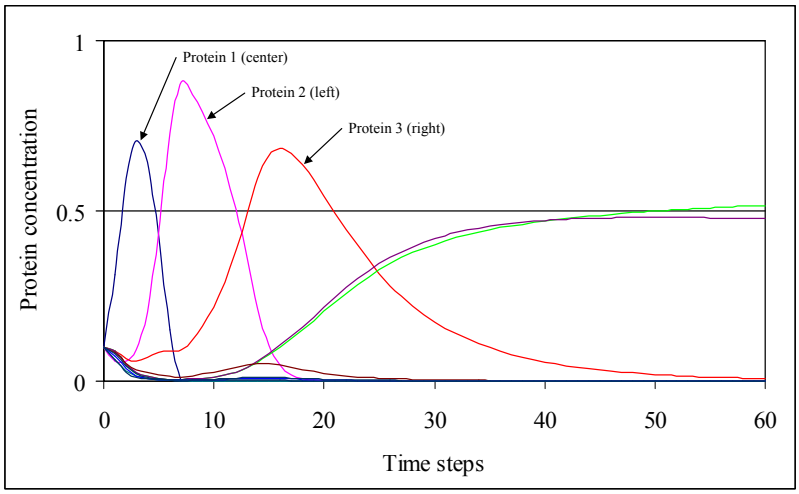


FIG. 5.14 – Graph of protein concentration change from an ARN expressing the 21×7 French flag.

Unlike the two- and three-color squares, where each gene had to be activated in a precise sequence, to create the flag pattern the central square could be extended to the left or to the right in any of the two orders, that is, first extend to the left and then to the right, or vice versa. This allowed more flexibility for the GA to find an appropriate ARN. Figure 5.13 shows a 21×7 French flag pattern grown from the expression of the

three structural genes mentioned above. The graph of protein concentration change from the corresponding ARN is shown in Fig. 5.14. After the white central square is formed, the left blue square is grown, followed by the right red square.

To illustrate a different sequence of gene activation, the 27 × 9 French flag pattern shown in Fig. 5.15 was created. The corresponding protein concentration graph is presented in Fig. 5.16. Note that in this case, the right red square is formed before the left blue one.

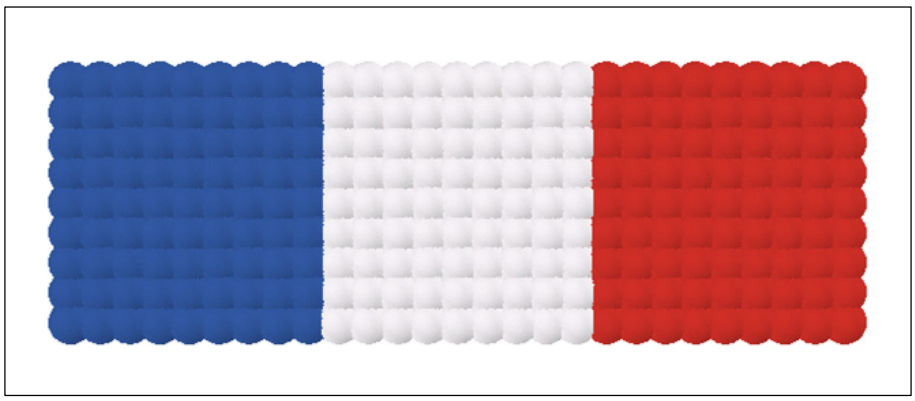


FIG. 5.15 – French flag (27 × 9).

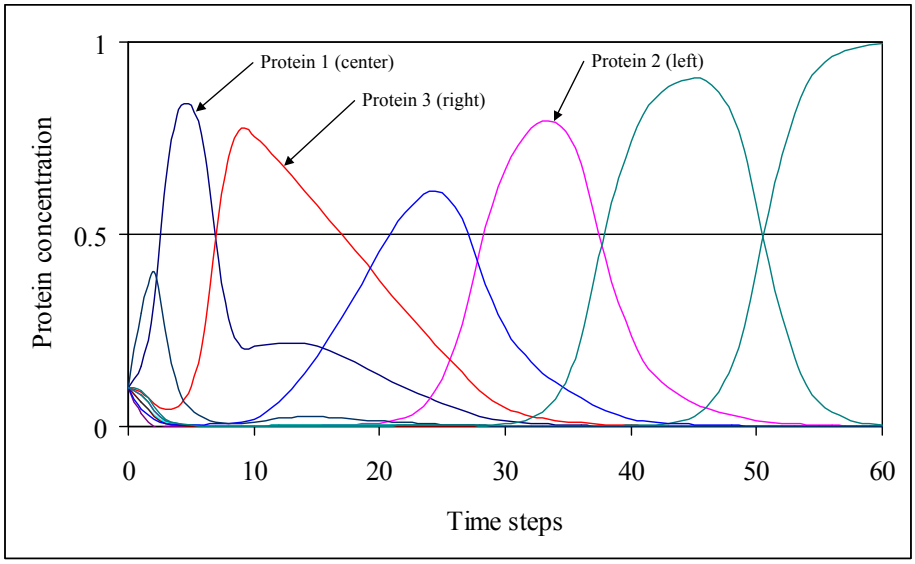


FIG. 5.16 – Graph of protein concentration change from an ARN expressing the 27 × 9 French flag pattern

When the patterns generated do not require strictly sequential gene activation, structu-

ral genes do not necessarily have to complete one pattern before forming another pattern. Take as example the case of the French flag pattern, where some structural genes can interrupt their expression and then resume activation at a later time after the expression of another gene. One of the genes that extend cells to one side can be interrupted to allow the gene that extends cells to the opposite side to be activated, and then resume its activation to finish the pattern. Fig. 5.17 shows the protein concentration graph of one of these cases, where a 27×9 French flag pattern is generated. After the white central square is formed, the gene that extends the red cells to the right is activated for only 5 time steps generating a 5×9 red rectangle. Then the gene that extends the blue cells to the left is activated for 9 time steps until completion of the left blue square. Finally, the gene that extends cells to the right becomes activated again for another 4 time steps to finish the generation of the red right square.

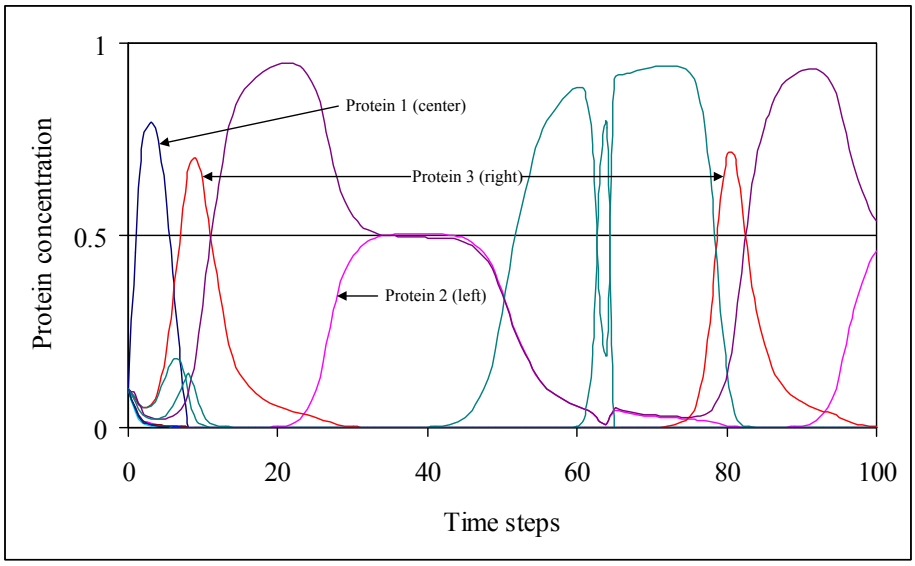


FIG. 5.17 – Graph of protein concentration change from an ARN expressing the 27×9 French flag pattern built from the alternating activation of structural genes.

One point worth noting regarding the GA parameters is that a relatively high mutation rate was used. This choice was made since it was found that single bits could have a considerable influence in the final behavior of the ARN, as previously noted by Banzhaf [Ban03]. As an example, consider the two graphs shown in Fig. 5.18 from an experiment

where a two-color square was grown. The corresponding ARNs for these graphs differ only in the value of bit position 952 on the enhancer site of regulatory gene number 5. While the upper graph of Fig. 5.18 corresponds to a chromosome with fitness value of 0.50, the lower graph corresponds to a chromosome with a fitness value of 0.93. In fact, further mutation of this latter chromosome on bit 599 in regulatory gene number 3, gets the ARN to achieve a fitness value of 1.00.

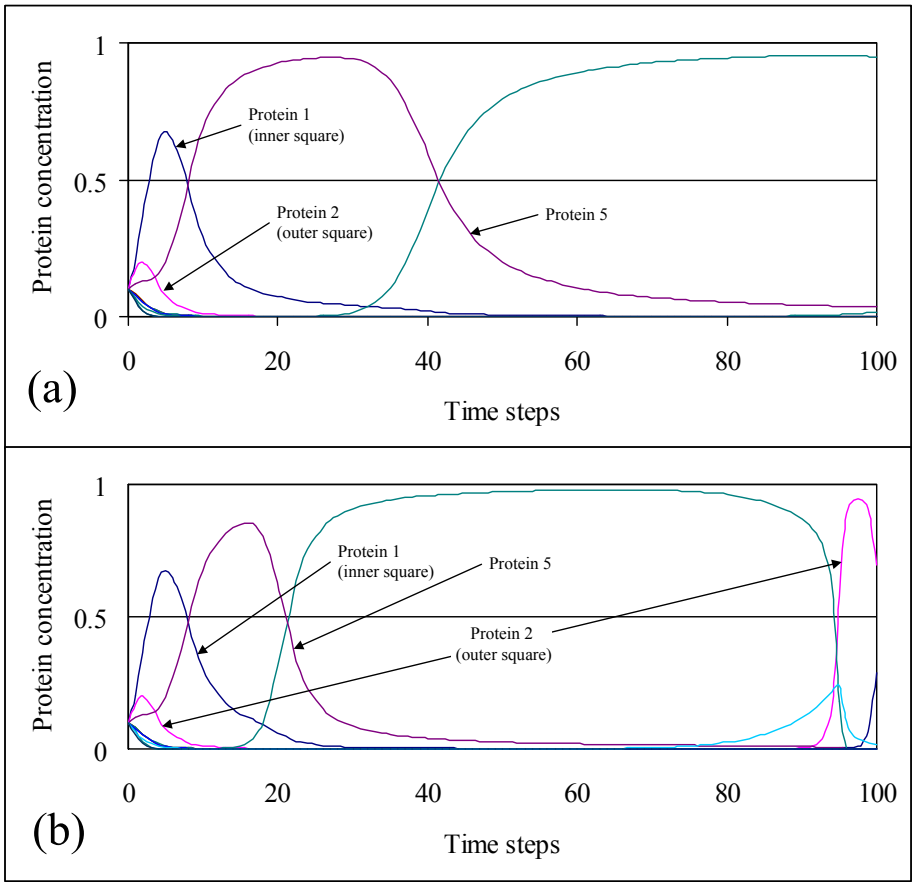


FIG. 5.18 – Effect of a single different bit in the ARN. (a) Fitness 0.50 ; (b) Fitness 0.93

5.2.2 Extended ARN

In the extended ARN model, in order to test the effect of different values in the number of function defining bits and regulatory sites, a series of GA experiments was performed. The tests consisted in finding an ARN to grow the three-color square shown in Figure

5.11 in the cellular growth testbed, while varying the number of function defining bits and regulatory sites. Three structural genes each coding for a square of different color were used. The other parameters values for the GA and the ARN were set as described in Chapter 4.

The number of function defining bits was varied from 1 to 16, whereas the number of regulatory sites was increased from 2 to 10, in 2-unit steps. Results are presented in Figure 5.19 as a graph of accumulated fitness values over the parameter ranges tested. These results represent single trial experiments for each pair of parameter values tested. Due to the lengthy simulation times, typically lasting several hours, it was difficult to run several simulations for each one of the parameter values tested. Even though an average fitness of several simulations over each pair of parameter values would have been more statistically sound, the results presented here can give us an overall estimate of the behavior of the ARN for the parameter values considered [CD07b].

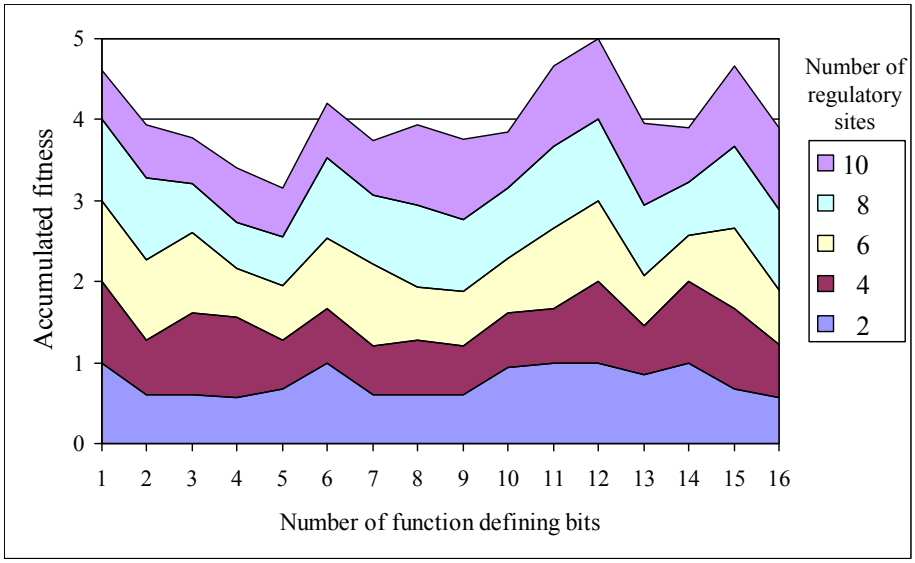


FIG. 5.19 – Accumulated fitness values for varying numbers of function defining bits and regulatory sites in the extended model.

From the results, it is not evident that there is a difference between the use of an odd and an even number of function defining bits on the final fitness values obtained. Therefore it is not clear whether or not there is an advantage in giving the regulatory

sites the ability to be turned on and off. Likewise, there is no clear improvement on the final fitness values when increasing the number of regulatory sites in the ARN. Figure 5.20 presents an average of the fitness values from all 16 simulations corresponding to the varying function defining bits, for each of the regulatory site values tested.

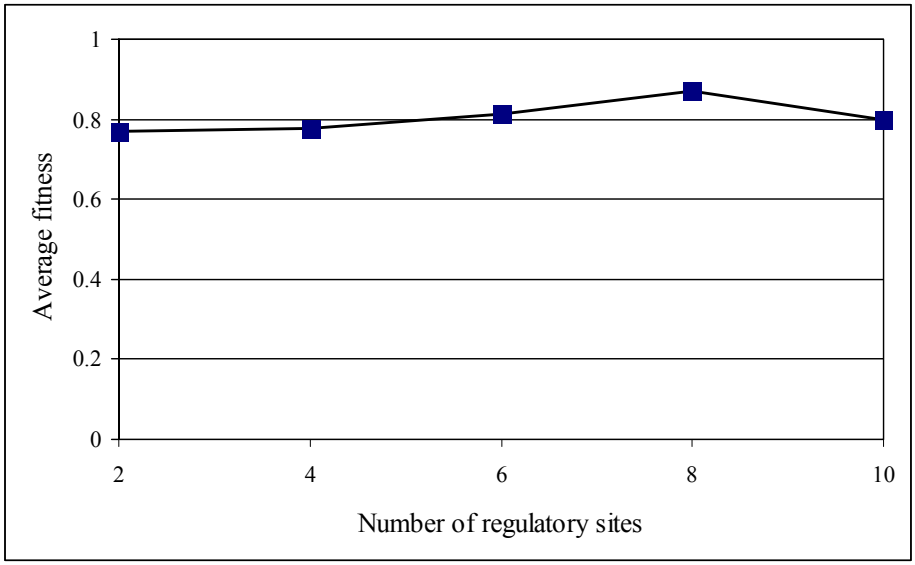


FIG. 5.20 – Average fitness values from the 16 simulations corresponding to the different values of function defining bits tested in the extended model.

In all the simulations described from now on, the values for the number of function defining bits and the number of regulatory sites were chosen as 12 and 8, respectively, since they provided the best results under the conditions tested. However, as mentioned before, these values are to be taken with reserve, as there is no definite proof that they correspond to the optimal values for these parameters.

Once suitable parameter values were chosen, a number of simulations were performed in order to evolve ARNs for other colored patterns. It should be mentioned that as in the basic model, not all GA experiments produced an ARN capable of finding the desired pattern. Furthermore, some difficulties were found when trying to evolve appropriate ARNs for developing patterns involving four structural genes.

The graphs presented next correspond to some of those experiments where ARNs with fitness function values equal to 1.0 were found by the GA. Once again, for some of

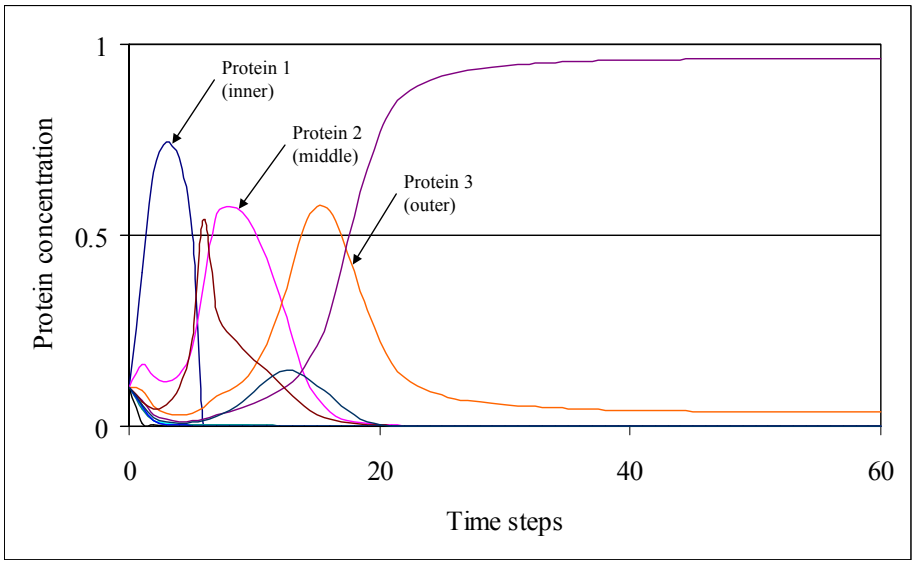


FIG. 5.21 – Graph of protein concentration change from an ARN expressing the three-color square.

the graphs shown, only the first 60 time steps are presented, as there is no significant change in these graphs after this point and until the end of the 100-step run. Figure 5.21 presents the graph of protein concentration change corresponding to one of the evolved ARN expressing the three-color square shown in Figure 5.11.

When trying to grow a four-color square through the expression of four structural genes each coding for a square of different color, it proved difficult to synchronize the expression of the four regulatory genes in a specific sequence. Using the same approach as in the case of the basic ARN model, in order to increase the likelihood for the GA to find an appropriate ARN, instead of using one series of four structural genes coding for a square, a tandem of two identical series of four structural genes was used, for a total of eight structural genes. In that manner, for creating the innermost square, the ARN could express either structural gene number 1 or gene number 5, for the next two squares it could use genes 2 or 6, or genes 3 or 7, respectively, and finally for the outermost square it could express structural genes 4 or 8. Then again, the probability of finding an ARN that could express a four-color square with a particular color order was significantly increased.

Using an ARN with this configuration of structural genes, the four-color 25×25

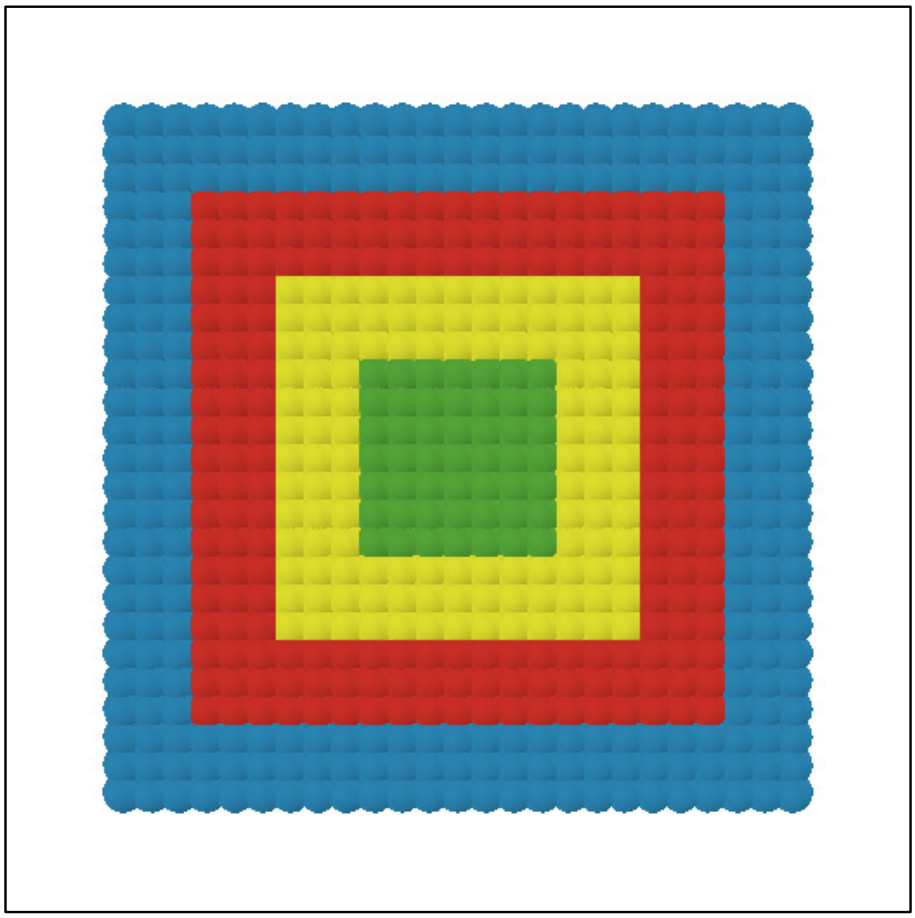


FIG. 5.22 – Four-color square.

square shown in Figure 5.22 was obtained. The graph representing the expression of the proteins for the corresponding ARN is shown in Figure 5.23. Note that in this case, only the structural genes from the second series were expressed, from gene number five to gene number eight. Other experiments found a different sequence of activation combining genes from the two series (data not shown).

As in the case of the three-color square, the four-color square pattern was chosen so that the different structural genes were expressed for the same number of time steps in the cellular growth model. Each of the four genes involved is to be activated for exactly three time steps to grow this pattern.

In order to explore once again the result of combining different structural genes that are expressed for a different number of time steps, four structural genes were used to

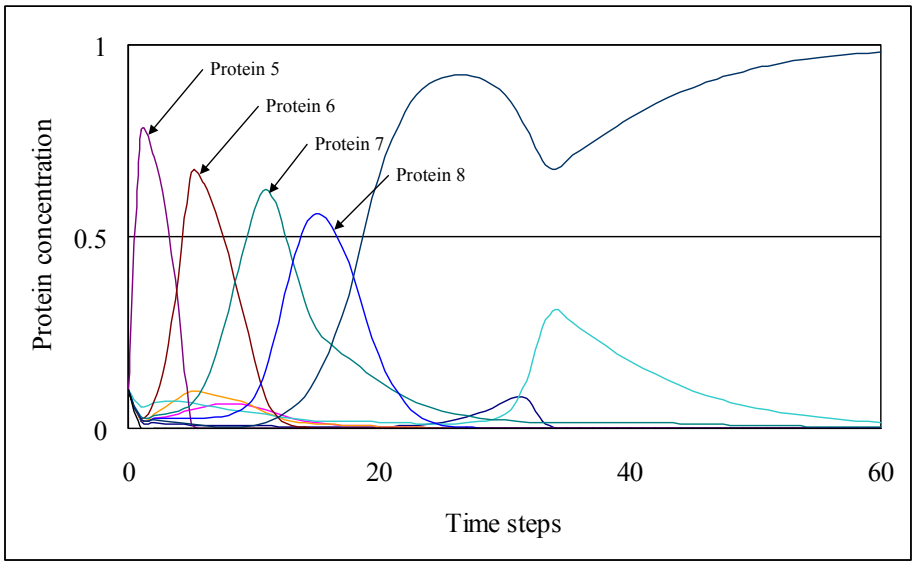


FIG. 5.23 – Graph of protein concentration change from an ARN expressing the four-color square.

grow a French flag with a flagpole pattern. Unlike previous reports where only the French flag itself was produced, the flagpole was added in order to increase the complexity of the pattern generated. The same three structural genes used previously for growing the French flag pattern were used. The fourth gene added created the brown flagpole by means of growing a single line of cells downward from the lower left corner of a rectangle.

When trying to evolve an ARN to produce the French flag with a flagpole pattern, it was found that, as in the case of the four-color square, the GA could not easily evolve an activation sequence that produced the desired pattern. In consequence, it was decided to use the same approach as before of setting a tandem of two identical series of the four structural genes that could produce the desired pattern. Figure 5.24 shows the 21×7 French flag with a flagpole pattern produced by the expression of the configuration of structural genes mentioned above. The graph of protein concentration change for the corresponding ARN is shown in Figure 5.25. After the white central square is formed, the right red square and the left blue square are sequentially grown, followed by the creation of the flagpole. Note that the white central square is formed from the activation of a gene from the first series of structural genes, while the other three genes are expressed from

the second series of the tandem.

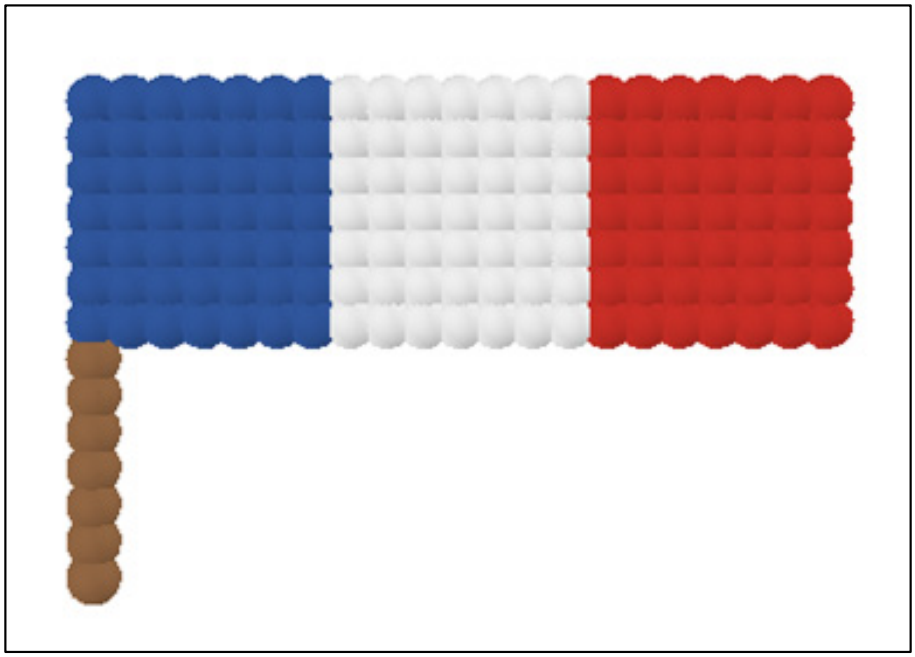


FIG. 5.24 – French flag with a flagpole pattern.

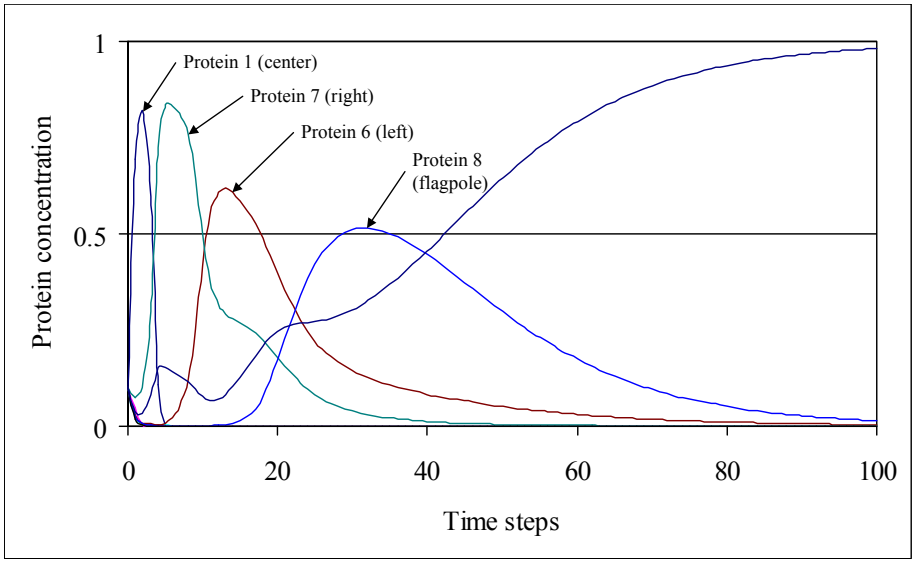


FIG. 5.25 – Graph of protein concentration change from an ARN expressing the 21×7 French flag with a flagpole pattern.

Unlike the problem of growing the four-color square pattern, where one gene had to finish forming the corresponding shape before the next gene could become activated, there is more flexibility in the activation sequence needed to grow the French flag with a flagpole

pattern. In particular, after the white central square is fully formed, the genes that extend the central square to either side can be activated in any order, and their corresponding activations can even alternate before either one has finished growing [CD07c]. However, it is essential that the flag is fully formed before the flagpole can begin to grow. It is evident that the left blue square has to be complete in order to start growing the flagpole at the correct position, but consider the case where the right red square is not fully formed after the flagpole, or part of it, was grown. In this case, if the gene that extends a vertical line of cells to the right is activated, it would not only produce the cells required to finish the red right square, but it would equally start to extend the flagpole to the right, since it also consists of a vertical line of cells.

5.2.3 Extended ARN with Morphogenetic Fields

In order to test the effect of the addition of the morphogenetic fields to the extended ARN model, the artificial development model was again applied to the creation of the French flag pattern with and without the flagpole. The same structural genes used in the previous models were used for growing the desired patterns [CD07a].

Figure 5.26 shows a 27×9 French flag pattern grown from the expression of the three structural genes used previously. The graph of the corresponding regulatory protein concentration change over time is shown in 5.26(e). Starting with a single white cell (a), a white central square is formed from the expression of gene number 1 (b), the left blue square is then grown (c), followed by the right red square (d). The evolved morphogenetic fields are shown for each of the three structural genes. Since the pattern obtained was exactly as desired, the fitness value assigned to the corresponding ARN was the unity.

In the case of the French flag with a flagpole pattern, the approach of using two tandems of the four appropriate structural genes was again used. The 21×7 French flag with a flagpole pattern produced by the expression of this configuration of structural genes is shown in Fig. 5.27. The graph for the corresponding regulatory protein concentration

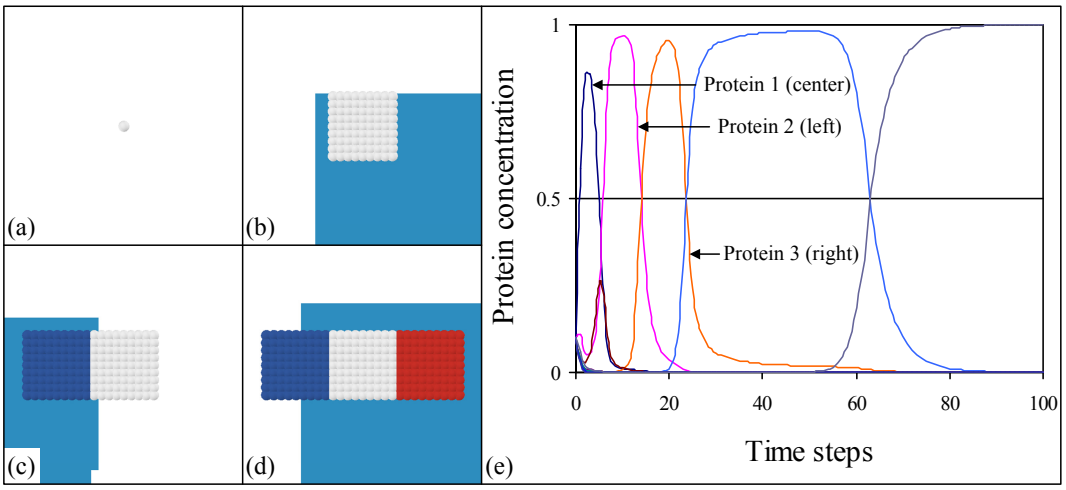


FIG. 5.26 – Growth of a French flag pattern. (a) Initial cell ; (b) Central white square with morphogenetic field for gene 1 (square) ; (c) White central square and left blue square with morphogenetic field for gene 2 (extend to left) ; (d) Finished flag pattern with morphogenetic field for gene 3 (extend to right) ; (e) Graph of protein concentration change from the genome expressing the French flag pattern.

change is shown in 5.27(e). After the white central square is formed (a), a right red pattern (b) and the left blue square (c) are sequentially grown, followed by the creation of the flagpole (d). The evolved morphogenetic fields are shown for each of the four structural genes expressed. Note that the white central square is formed from the activation of the first gene from the second series of structural genes, while the other three genes are expressed from the first series of the tandem. It should also be noted that the last column of cells is missing from the red right square, since the morphogenetic field for the gene that extends the red cells to the right precluded growth from that point on (Fig. 5.27(b)). On the other hand, from the protein concentration graph in 5.27(e), it is clear that this morphogenetic field prevented the growth of red cells all the way to the right boundary, as gene 3 was active for more time steps than those required to grow the appropriate red square pattern. The fitness value assigned to this pattern was 0.96, which corresponded to the most successful simulation obtained when trying to grow this particular pattern.

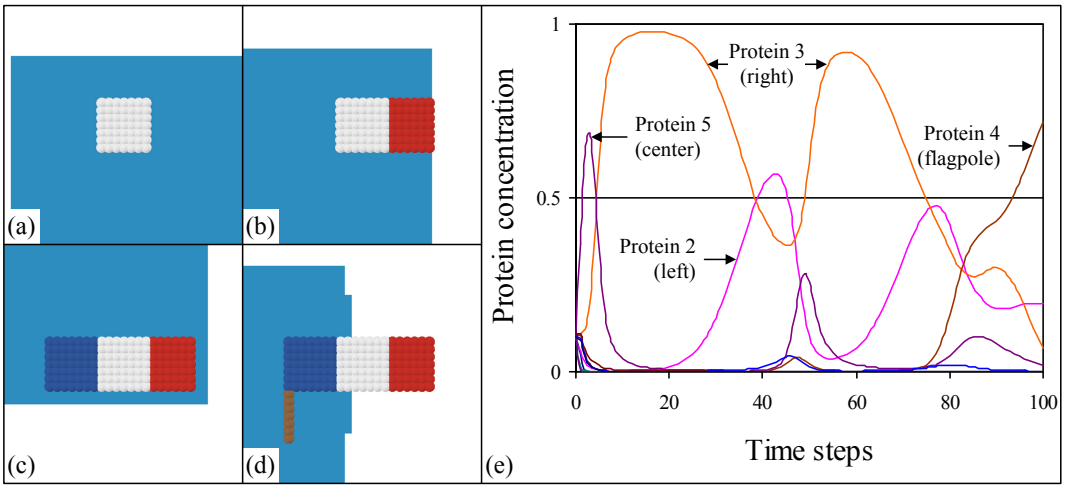


FIG. 5.27 – Growth of a French flag with a flagpole pattern. (a) Central white square with morphogenetic field for gene 5 (square); (b) White central square and right red pattern with morphogenetic field for gene 3 (extend to right); (c) White central square, right red pattern and left blue square with morphogenetic field for gene 2 (extend to left); (d) Finished flag with a flagpole pattern with morphogenetic field for gene 4 (flagpole); (e) Graph of protein concentration change from the genome expressing the French flag with a flagpole pattern.

6

Discussion and perspectives

Résumé

La génération de plusieurs structures convexes et non convexes a été tentée au début. On a rapidement découvert que les formes non convexes étaient très difficiles à obtenir en utilisant un AC avec une seule table des règles et sans mécanisme pour permettre aux cellules actives de revenir à l'état vide. Toutes les formes simples sélectionnées à la fin, tant en 2D qu'en 3D, étaient du type convexe. Le problème de générer une structure non convexe dans le modèle de croissance cellulaire a été résolu plus tard à travers l'utilisation de la différentiation cellulaire, au moyen de l'expression sélective de gènes structurels qui contiennent des tables de règles différentes.

Bien qu'il n'y eût pas un modèle qui puisse produire toutes les formes 2D avec un haut degré d'exactitude, c'était évident que quelques modèles étaient plus aptes à générer une forme particulière. En moyenne, pour les quatre voisinages étudiés, les modèles les plus prometteurs pour la génération des formes prédéfinies étaient les voisinages Moore et Margolus, avec pratiquement la même aptitude moyenne. Le voisinage Moore a été choisi à la fin dans le modèle de croissance cellulaire pour l'évaluation des modèles de RARs. Dans le cas 3D, les résultats ont montré que la combinaison d'un AG et d'un AC avec un voisinage Margolus en 3D était un bon choix pour modéliser la génération de formes tridimensionnelles. Bien que les modèles de RARs aient été évolués pour générer des formes en 2D, les résultats de ces modèles pourraient être extrapolés au domaine 3D, mais avec des temps de simulation plus longs.

Quant aux modèles de RARs, on a trouvé qu'en utilisant le modèle de RAR basique on pourrait synchroniser deux ou trois gènes structurels avec fiabilité. Cependant, quelques problèmes ont été trouvés en essayant de synchroniser l'activation de plus de trois gènes structurels dans une séquence précise. Avec le modèle de RAR étendu, les résultats ont montré que c'était relativement facile de synchroniser avec fiabilité jusqu'à trois gènes, mais des problèmes ont encore été rencontrés lorsque l'on a essayé de synchroniser l'activation de quatre gènes dans une séquence prédéfinie. D'après les résultats obtenus avec le RAR étendu qui utilise des gradients morphogénétiques, c'était apparemment plus difficile pour l'AG de trouver une séquence d'activation pour la création de la forme d'un drapeau français avec une hampe, que dans le modèle de RAR sans les gradients. Une explication possible c'est qu'avec l'addition dans le RAR des sites du seuil d'activation des morphogènes, l'espace de recherche de l'AG était plus grand que dans le modèle étendu original, en le rendant plus difficile pour l'AG de trouver une séquence d'activation appropriée.

En général, le cadre présenté a prouvé être convenable pour la génération de structures cellulaires qui impliquent jusqu'à quatre gènes structurels. Cependant, il est nécessaire d'approfondir ce travail afin d'explorer la formation de structures plus complexes en 2D et en 3D. Il est également souhaitable d'intégrer dans ce modèle la disparition et le déplacement des cellules. En outre, afin de construire un modèle plus

exact du processus de croissance cellulaire, l'utilisation d'un environnement physique plus réaliste peut être nécessaire.

6.1 Form Generation

At the initial stages of the work, the generation of several shapes both convex and non-convex was tried. In geometry, a convex shape has the property that the line segment that joins any two points in the shape is contained within the shape. Given the discrete nature of cell positions in the lattice, convex shapes are harder to define in a CA. We could say that convex shapes in a CA are those where all the filled cells in the shape are circumscribed by the perimeter of a real-valued convex shape with no space left for empty cells.

Among the non-convex shapes that were tried, there was a star shape and an L-shaped form. However, it was soon discovered that non-convex shapes are difficult to obtain in a CA with a single rule table and with no provision for allowing filled cells to revert to the empty state. Basically the problem in forming non-convex shapes in a CA with these restrictions is that the same rule table is applied to all cells. Since cells forming a single shape in a CA with a single transition rule are all the same and they only have limited local information, there is no coordination among cells to differentiate into cells with different roles. Given that cell death and cell displacement are not allowed in the model for the sake of simplicity, once an empty cell becomes occupied or filled, it remains in that state for the rest of the simulation. To compound the problem, a filled cell can only be introduced in the lattice if there is already a filled cell in the neighborhood template.

For the above reasons, all single shapes selected in the end both in 2D and 3D were of the convex type. However, the problem of forming a non-convex pattern in the cellular growth testbed was later solved through the use of cell differentiation, by means of the selective expression of structural genes containing different rule tables.

Results obtained in setting up the framework for single shape generation in 2D showed

that although there was not one model that could generate all four shapes with a high degree of accuracy, it was evident that some models were more appropriate than others in building a particular shape. For instance, the von Neumann model was particularly efficient in generating the diamond shape, which is no surprise given the spatial disposition of the neighborhood template itself. However, for the other shapes, the von Neumann model had the worst performance in form generation, possibly due to its "blindness" towards the adjacent diagonally positioned cells.

As for the other neighborhoods, even though the Moore neighborhood template could be viewed as a subset of the 2-Radial neighborhood template, the final chromosomes obtained with the Moore model had higher fitness values for all four shapes. One possible explanation is that, as mentioned in Chapter 4, the 2-Radial model had a far larger search space than the Moore model, which could make it difficult for the GA to find the fitness maxima. Furthermore, unlike the other neighborhoods where all cells in the outer neighborhood were directly adjacent to the central cell, in the 2-Radial template some cells could be introduced in the lattice without an intermediate cell to be present. In particular the four cells farthest from the neighborhood center could in principle generate a filled cell at the objective cell. This nonetheless gave no apparent advantage to this neighborhood over the other templates.

Although some of the shapes evolved in the cellular growth testbed are not strictly convex, the cells that are not inside the convex shapes are mainly generated by means of the random asynchronous selection of cell reproduction. The shapes shown in the figures of Chapter 5 are only the product of individual runs. Different runs normally rendered slightly different shapes, and that is the reason for using a probabilistic approach of summarizing the results from 100 runs.

On the average, of the four neighborhoods studied, the most promising models in shape generation were those corresponding to Moore and Margolus, both with practically the same average fitness (see Table 5.1). The results obtained with the Margolus template were

in most cases comparable to those derived with the Moore neighborhood, even though the former had a chromosome size that was less than a tenth in length than that of the latter. The only case where the Moore neighborhood showed some advantage over the Margolus neighborhood was in the generation of the triangle. Nevertheless, it is possible that the Moore neighborhood would be better at forming more complex shapes than the Margolus template, given its much larger search space. It is for this reason that the Moore neighborhood was chosen as template for the evaluation of the ARN models in the cellular growth model.

In the 3D case, results showed that the combination of a GA and CA with a 3D Margolus interaction neighborhood was a good choice for modeling 3D shape generation. Even though the ARNs models were evolved for developing 2D patterns, results from these models could be extrapolated to 3D. However, simulation times would be considerably longer and it would probably be more difficult for the GA to find an appropriate gene activation sequence in 3D, given the additional dimension to take into consideration.

As for letting the number of iterations evolve in the cellular growth model when generating a predefined shape, this decision turned out to be appropriate, as the number of iterations needed to create the shapes was not always the same as those intuitively expected. However, in the simulations that best approached the shapes desired, the number of iterations evolved were close to or exactly the same as those expected. It was decided nonetheless that one should not to influence the results with preconceived notions of the expected outcome.

In general, the framework developed proved to be suitable for generating simple shapes, but more work is needed to explore single shape formation of more complex forms, both in 2D and 3D. It is also desirable to study shape formation allowing cell death and cell displacement, as in actual cellular growth. Furthermore, in order to build a more accurate model of the growth process, the use of a more realistic physical environment may be necessary.

6.2 Pattern Generation

Simulations involving the basic ARN model show that a GA can give reproducible results in evolving an ARN to grow predefined simple 2D cell patterns starting with a single cell. In particular, it was found that using this ARN model it was feasible to reliably synchronize two or three structural genes. However, some problems were found when trying to synchronize the activation of more than three structural genes in a precise sequence.

Despite its limitations, this basic model demonstrated that the synchronization of structural genes similar to the gene expression regulation found in nature was feasible. However, it was necessary to find a way to synchronize more than three structural genes if generation of more complex patterns was to be achieved.

As is often the case, by studying how nature works, insight can be gained that aid in proposing approaches for solving a particular problem. In this case, it was decided that the number of enhancer and inhibitor sites in the regulatory network could be increased as in biological gene regulatory networks. Likewise, the role as enhancer or inhibitor of the regulatory sites was allowed to be evolved, as is the case in biological genomes, where the role of regulatory sites depends on the particular nucleotide sequence present at the appropriate places.

With the extended ARN model, results showed that it was relatively easy to reliably synchronize up to three genes, but problems were still encountered when trying to synchronize the activation of four genes in a precise sequence. For solving this problem, it was necessary to use the approach of defining a tandem of two series of the structural genes necessary to develop the pattern.

As for the extensions added to the original ARN model, the results suggest that the number of function defining bits had more influence than the number of regulatory sites in evolving an ARN to produce a desired shape. One possible explanation is that a bigger defining bits region could offer a larger target on which the mutation operator could act.

On the other hand, there was no apparent advantage in providing the regulatory sites with the ability to be turned on and off, since there was no significant difference between an even and an odd number of function defining bits. Similarly, there was no clear-cut advantage in increasing the number of regulatory sites above the original number. Although more work is required to confirm these results, the extended ARN model could arrive to solutions that the basic ARN model was not able to find under the conditions tested.

In the extended ARN model, apart from the gene activation sequence coded in the genome, cells only had local information to determine whether or not to reproduce. In particular, cells had no global positional information, since the shape grown was mainly due to a self-organizing mechanism driven by the ARN. However, in order to achieve more complex shapes, it was considered necessary to allow cells to extract information from their environment through the use of diffusing morphogens.

Morphogenetic fields should in principle assist in the creation of more complex patterns by providing positional constraints to cellular growth. However in the results obtained with the extended ARN with morphogenetic fields, it was apparently harder for the GA to find an activation sequence for the creation of the French flag with a flagpole pattern. One possible explanation is that with the addition of the morphogen threshold activation sites to the ARN, the search space grew even larger than in the original extended ARN model, making it more difficult for the GA to find an appropriate activation sequence. However, since individual simulations times usually took several hours to complete, it could be that the number of simulations essayed with both models was not high enough to draw an unambiguous conclusion.

On the other hand, there is evidence that the fitness landscape on which the GA performs the search to evolve the ARNs is very rugged. This was illustrated with the influence of single bits on the fitness values of an evolving model. It took the shift of one bit value in the genome string of the basic ARN model to go from a fitness value of 0.50 to 0.93, and one additional single bit shift led the fitness value to a perfect match.

In this particular case, that meant that adjacent vectors in the search space had very dissimilar values in fitness evaluation. It is conjectured that this behavior is widespread in the search spaces defined in the models developed, given the difficulties encountered in synchronizing what could be considered just a handful of structural genes. Most likely, a change of representation would aid in the search process. However, this would mean an extensive change in the basic model design, which is not always easy to accomplish.

In all the ARN models presented, some patterns were easier than others to produce. The more independence there was in the sequence in which the individual shapes were generated, the easier it was for the GA to find an appropriate gene activation sequence in the ARN. The hardest patterns to be generated were those where there was a fixed structural gene activation sequence on which the shapes were to be created.

One restriction of the three ARN models presented is that all cells synchronously follow the same genetic program, as a sort of biological clock. This has obvious advantages for synchronizing the behavior of developing cells, but it would also be desirable that cells had an individual program –possibly a separate ARN– for reacting to local unexpected changes in their environment. Morphogenetic fields provide a means to extract information from the environment, but an independent program would lend more flexibility and robustness to a developing organism. After all, living organisms do contain a series of gene regulatory networks for development and metabolism control. One could even envision either a hierarchy of ARNs, where some ARNs could be used to regulate others ARNs, or a network of ARNs, where all ARNs could influence and regulate each other.

7

Conclusion

Résumé

En général, les modèles de développement artificiels présentés dans ce travail ont prouvé leur capacité à produire des structures 2D simples qui impliquent l'activation de quatre gènes structurels au maximum. Les modèles les plus faciles à obtenir étaient ceux où l'activation des gènes ne devait pas suivre un ordre strictement séquentiel.

Malgré la difficulté à trouver un modèle d'AC pour produire régulièrement des formes convexes avec une seule table des règles, il a été trouvé qu'il est aisé de le faire à travers la synchronisation d'une séquence de l'activation dans un modèle de RAR.

Cependant, des travaux supplémentaires sont nécessaires afin d'explorer la formation des structures cellulaires plus complexes dans les domaines 2D et 3D. Il faut également de trouver un modèle de développement pour synchroniser de manière fiable l'activation de plus de quatre gènes. Pour accomplir la séquence d'activation de cinq gènes structurels (ou plus) dans l'approche présentée de synchronisation d'un RAR, il est probablement nécessaire de changer la représentation du modèle, afin qu'un paysage d'aptitude plus lisse puisse être obtenu. En outre, pour augmenter l'utilité du modèle, l'interaction avec d'autres entités artificielles et l'extraction d'information d'un environnement physiquement plus réaliste peuvent être nécessaires.

Jusqu'à maintenant ce travail a été consacré à produire des structures prédéfinies dans une espèce d'évolution dirigée. Cependant, il serait désirable de laisser évoluer les cellules dans une structure fonctionnelle sous contraintes de l'environnement, sans aucune notion préconçue du résultat final.

À la connaissance de l'auteur, ceci est le premier rapport de l'utilisation d'un AG pour évoluer un comportement spécifique dans un RAR pour former une structure cellulaire artificielle. Le but à long terme de ce travail est d'étudier les propriétés émergentes du processus du développement artificiel. Il peut être envisagé qu'un jour il sera possible de construire des structures très complexes qui surviennent principalement de l'interaction de myriades d'entités plus simples.

In general, the artificial development models presented proved to be suitable for obtaining simple 2D patterns involving the activation of up to four structural genes. The easiest patterns to obtain were those where the activation of genes did not have to follow a strictly sequential order.

Although it was hard to find a CA model that consistently generated convex shapes with a single rule table and no reversion to the empty cell state from the filled cell state, it was found that the expression of different rule tables through the synchronization of an activation sequence in an ARN model could readily do it.

However, additional work is needed in order to explore pattern formation of more complex forms, both in 2D and 3D. It is also desirable to search for a development model that can reliably synchronize the activation of more than four genes. In order to achieve the activation sequence of five or more structural genes using the approach presented of ARN synchronization, it is probably necessary to change the representation of the model, so that a smoother fitness landscape could be obtained. Furthermore, in order to increase the usefulness of the model, interaction with other artificial entities and extraction of information from a more physically realistic environment may be necessary. Until now this work has been devoted to generating predefined patterns in a kind of directed evolution. However, it would be desirable to let cells evolve into a functional pattern under environmental constraints without any preconceived notion of the final outcome.

To the author's knowledge, this is the first report of the use of a GA to evolve a specific behavior in an ARN to grow an artificial cell pattern. A simple GA was chosen in this work for evolving the ARN due to the discrete and fixed-size nature of the artificial genome used. Moreover, it was considered that the GA was the evolutionary computation paradigm that resembled the most the actual evolutionary mechanism seen in nature.

The long-term goal of this work is to study the emergent properties of the artificial development process. It can be envisioned that one day it will be feasible to build highly complex structures arising mainly from the interaction of myriads of simpler entities.

Bibliographie

[ATTY99] Hideki Arata, Yoshiaki Takai, Nami K. Takai, and Tsuyoshi Yamamoto. Free-form shape modeling by 3d cellular automata. In *Shape Modeling International (SMI '99)*, pages 242–264. IEEE Computer Society, 1999.

[Ban03] Wolfgang Banzhaf. Artificial regulatory networks and genetic programming. In Rick L. Riolo and Bill Worzel, editors, *Genetic Programming Theory and Practice*, chapter 4, pages 43–62. Kluwer, 2003.

[BB05] Ron Breukelaar and Thomas Bäck. Using a genetic algorithm to evolve behavior in multi dimensional cellular automata : emergence of behavior. In *GECCO '05*, pages 107–114, 2005.

[BH70] R. W. Baker and G. T. Herman. Celia - a cellular linear iterative array simulator. In *Proceedings of the fourth annual conference on Applications of simulation*, pages 64–73. Winter Simulation Conference, 1970.

[BMF06] G. Beurier, F. Michel, and J. Ferber. A morphogenesis model for multiagent embryogeny. In Luis Mateus Rocha, Larry S. Yaeger, Mark A. Bedau, Dario Floreano, Robert L. Goldstone, and Alessandro Vespignani, editors, *Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems (ALife X)*, pages 84–90, 2006.

[Bon02] J. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, pages 1872–1877. IEEE Press, Piscataway, NJ, 2002.

[Bow05] C.P. Bowers. Simulating evolution with a computational model of embryogeny : Obtaining robustness from evolved individuals. In M. S. Capcarrere, A. A. Freitas, P. J. Bentley, C. G. Johnson, and J. Timmons, editors, *Advances in Artificial Life, Proceeding of the 8th European Conference on Artificial Life : ECAL 2005*, pages 149–158. Springer, 2005.

[Car06] Sean B. Carroll. *Endless Forms Most Beautiful : The New Science of Evo Devo*. W. W. Norton, April 2006.

[CD06a] Arturo Chavoya and Yves Duthen. Evolving cellular automata for 2D form generation. In *Proceedings of the Ninth International Conference on Computer Graphics and Artificial Intelligence 3IA'2006*, pages 129–137, 2006.

[CD06b] Arturo Chavoya and Yves Duthen. Using a genetic algorithm to evolve cellular automata for 2D/3D computational development. In *GECCO '06 : Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 231–232, New York, NY, USA, 2006. ACM Press.

[CD07a] Arturo Chavoya and Yves Duthen. An artificial development model for cell pattern generation. In *ACAL '07 : Proceedings of the 3rd Australian Conference on Artificial Life*, December 2007.

[CD07b] Arturo Chavoya and Yves Duthen. A cell pattern generation model based on an extended artificial regulatory network. In *IPCAT '07 : Proceedings of the 7th International Workshop on Information Processing in Cells and Tissues*, pages 149–158, 2007.

[CD07c] Arturo Chavoya and Yves Duthen. Evolving an artificial regulatory network for 2D cell patterning. In *Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife'07)*, pages 47–53. IEEE Computational Intelligence Society, 2007.

[CD07d] Arturo Chavoya and Yves Duthen. Use of a genetic algorithm to evolve an extended artificial regulatory network for cell pattern generation. In *GECCO '07 : Proceedings of the 9th annual conference on Genetic and evolutionary computation*, page 1062, New York, NY, USA, 2007. ACM Press.

[CGW04] Sean B. Carroll, Jennifer K. Grenier, and Scott D. Weatherbee. *From DNA to Diversity : Molecular Genetics and the Evolution of Animal Design*. Blackwell Science, 2nd edition, 2004.

[Dav06] Eric H. Davidson. *The Regulatory Genome : Gene Regulatory Networks in Development And Evolution*. Academic Press, 1st edition, 2006.

[Daw96] Richard Dawkins. *The Blind Watchmaker : Why the Evidence of Evolution Reveals a Universe Without Design*. W. W. Norton, September 1996.

[DBS07] Alexandre Devert, Nicolas Bredeche, and Marc Schoenauer. Robust multicellular developmental design. In *GECCO '07 : Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 982–989, New York, NY, USA, 2007. ACM.

[DD05] Andreas Deutsch and Sabine Dormann. *Cellular Automaton Modeling of Biological Pattern Formation : Characterization, Applications, and Analysis*. Birkhäuser, Boston, 2005.

[de 91] Hugo de Garis. Genetic programming : artificial nervous systems artificial embryos and embryological electronics. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN 1*, volume 496, pages 117–123, Dortmund, Germany, 1-3 1991. Springer-Verlag, Berlin, Germany.

[de 92] Hugo de Garis. Artificial embryology : The genetic programming of an artificial embryo. In Branko Souček, editor, *Dynamic, Genetic, and Chaotic Programming*, pages 373–393. John Wiley, New York, 1992.

[de 99] Hugo de Garis. Artificial embryology and cellular differentiation. In Peter J. Bentley, editor, *Evolutionary Design by Computers*, pages 281–295. Morgan Kaufmann Publishers, Inc., San Francisco, USA, 1999.

[dIF92] Hugo de Garis, Hitoshi Iba, and Tatsumi Furuya. Differentiable chromosomes : The genetic programming of switchable shape-genes. In Reinhard Männer and

Bernard Manderick, editors, *Parallel problem solving from nature 2*, pages 489–498, Amsterdam, 1992. North-Holland.

[Egg97a] Peter Eggenberger. Creation of neural networks based on developmental and evolutionary principles. In W. Gerstner, A. Germond, M. Hasler, and J. D. Nicoud, editors, *Seventh International Conference of Artificial Neural Networks (ICANN'97)*, pages 337–342. Springer, 1997.

[Egg97b] Peter Eggenberger. Evolving morphologies of simulated 3D organisms based on differential gene expression. In I. Harvey and P. Husbands, editors, *Proceedings of the 4th European Conference on Artificial Life*, pages 205–213. Springer, 1997.

[Egg03] Pater Eggenberger Hotz. Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes. In Sanjeev Kumar and Peter J. Bentley, editors, *On Growth, Form and Computers*, chapter 16, pages 302–318. Academic Press, New York, NY, USA, 2003.

[EH04] Peter Eggenberger Hotz. Asymmetric cell division and its integration with other developmental processes for artificial evolutionary systems. In *Proceedings of the 9th International Conference on Artificial Life IX*, 2004.

[FB92] Kurt Fleischer and Alan H. Barr. A simulation testbed for the study of multicellular development : The multiple mechanisms of morphogenesis. In C. Langdon, editor, *Proceedings of the Workshop on Artificial Life ALIFE'92*, pages 389–416. Addison-Wesley, 1992.

[FHB+05] Nicholas Flann, Jing Hu, Mayank Bansal, Vinay Patel, and Greg Podgorski. Biological development of cell patterns : Characterizing the space of cell chemistry genetic regulatory networks. In *ECAL*, pages 57–66, 2005.

[GB05] Timothy G. W. Gordon and Peter J. Bentley. Bias and scalability in evolutionary development. In *GECCO '05 : Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 83–90, New York, NY, USA, 2005. ACM.

[Ger04] Carlos Gershenson. Introduction to random boolean networks. In M. Bedau, P. Husbands, T. Hutton, S. Kumar, and H. Susuki, editors, *Worshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)*, pages 160–173, 2004.

[Gie81] Alfred Gierer. Generation of biological patterns and form : Some physical, mathematical, and logical aspects. *Prog. Biophys. Molec. Biol.*, 37 :1–47, 1981.

[GM72] Alfred Gierer and Hans Meinhardt. A theory of biological pattern formation. *Kybernetik*, 12 :30–39, 1972.

[Han92] J. S. Hanan. *Parametric L-Systems and their Application to the Modelling and Visualization of Plants*. PhD thesis, University of Regina, June 1992.

[HCM98] Wim Hordijk, James P. Crutchfield, and Melanie Mitchell. Mechanisms of emergent computation in cellular automata. In *PPSN V : Proceedings of the*

5th International Conference on Parallel Problem Solving from Nature, pages 613–622, London, UK, 1998. Springer-Verlag.

[Heu98] Jean-Claude Heudin. *L'évolution au bord du chaos*. Hermes, Paris, France, 1998.

[HL73] G. T. Herman and W. H. Liu. The daughter of celia, the french flag and the firing squad. In *WSC '73 : Proceedings of the 6th conference on Winter simulation*, page 870, New York, NY, USA, 1973. ACM.

[HMB07] Simon L. Harding, Julian F. Miller, and Wolfgang Banzhaf. Self-modifying cartesian genetic programming. In *GECCO '07 : Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1021–1028, New York, NY, USA, 2007. ACM.

[Hol92] John H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[HW04a] J. Hallinan and Janet Wiles. Asynchronous dynamics of an artificial genetic regulatory network. In *Artificial Life IX : Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, pages 399–403. The MIT Press, 2004.

[HW04b] Jennifer Hallinan and Janet Wiles. Evolving genetic regulatory networks using an artificial genome. In *CRPIT '04 : Proceedings of the second conference on Asia-Pacific bioinformatics*, pages 291–296, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[Kau69] Stuart A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22 :437–467, 1969.

[Kau71] Stuart A. Kauffman. Gene regulation networks : A theory for their global structure and behavior. *Current Topics in Dev. Biol.*, 6 :145, 1971.

[Kau74] Stuart A. Kauffman. The large-scale structure and dynamics of gene control circuits : An ensembre approach. *Journal of Theoretical Biology*, 44 :167, 1974.

[Kau93] Stuart A. Kauffman. *The Origins of Order : Self-Organization and Selection in Evolution*. Oxford University Press, May 1993.

[Kau04] Stuart A. Kauffman. *Investigations*. Oxford University Press, 2004.

[KB03a] Sanjeev Kumar and Peter J. Bentley. Computational embryology : past, present and future. In *Advances in evolutionary computing : theory and applications*, pages 461–477. Springer-Verlag New York, Inc., New York, NY, USA, 2003.

[KB03b] Sanjeev Kumar and Peter J. Bentley. An introduction to computational development. In Sanjeev Kumar and Peter J. Bentley, editors, *On Growth, Form and Computers*, chapter 1, pages 1–44. Academic Press, New York, NY, USA, 2003.

[KB03c] Sanjeev Kumar and Peter J. Bentley. *On Growth, Form and Computers*. Academic Press, London, 2003.

[KB04] Paul Dwight Kuo and Wolfgang Banzhaf. Small world and scale-free network topologies in an artificial regulatory network model. In J. Pollack, M. Bedau, P. Husbands, T. Ikegami, and R. Watson, editors, *Proceedings of Artificial Life IX, (ALIFE-9), Boston, USA*, pages 404–409. MIT Press, Cambridge, 2004.

[KHKL98] Hiroaki Kitano, Shugo Hamahashi, Jun Kitazawa, and Sean Luke. The perfect *C. elegans* project : an initial report. *Artificial Life*, 4(2) :141–156, 1998.

[Kit90] Hiroaki Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4 :461–476, 1990.

[Kit94] Hiroaki Kitano. A simple model of neurogenesis and cell differentiation based on evolutionary large-scale chaos. *Artificial Life*, 2(1) :79–99, 1994.

[KLB04] Paul Dwight Kuo, Andre Leier, and Wolfgang Banzhaf. Evolving dynamics in an artificial regulatory network model. In X. Yao, E. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J. Rowe, P. Tino, A. Kabán, and H.-P. Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature Conference (PPSN-04)*, LNCS 3242, pages 571–580, Birmingham, UK, September 2004. Springer, Berlin.

[KNS07] Johannes F. Knabe, Chrystopher L. Nehaniv, and Maria J. Schilstra. The essential motif that wasn't there : Topological and lesioning analysis of evolved generic regulatory networks. In *Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife'07)*, pages 47–53. IEEE Computational Intelligence Society, 2007.

[KNSQ06] Johannes F. Knabe, Chrystopher L. Nehaniv, Maria J. Schilstra, and Tom Quick. Evolving biological clocks using genetic regulatory networks. In Luis M. Rocha, Larry S. Yaeger, Mark A. Bedau, Dario Floreano, Robert L. Goldstone, and Alessandro Vespignani, editors, *Proceedings of the Artificial Life X Conference (Alife 10)*, pages 15–21. MIT Press, 2006.

[Lan90] C. Langton. Computation at the edge of chaos : Phase transitions and emergent computation. *Physica D*, 42 :12–37, 1990.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18 :280–315, 1968.

[Lin71] Aristid Lindenmayer. Developmental systems without cellular interaction, their languages and grammars. *Journal of Theoretical Biology*, 30 :455–484, 1971.

[LR72] A. Lindenmayer and G. Rozenberg. Developmental systems and languages. In *STOC '72 : Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 214–221, New York, NY, USA, 1972. ACM.

[MB03] Julian F. Miller and Wolfgang Banzhaf. Evolving the program for a cell : from French flags to Boolean circuits. In Sanjeev Kumar and Peter J. Bentley, editors, *On Growth, Form and Computers*, pages 278–301. Academic Press, October 2003.

[MCD96] Melanie Mitchell, James P. Crutchfield, and Rajarshi Das. Evolving cellular automata with genetic algorithms : A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA'96), 1996.*, Moscow, Russia, 1996.

[Mei82] Hans Meinhardt. *Models of Biological Pattern Formation.* Academic Press, London, 1982.

[Mei96] Hans Meinhardt. Models of biological pattern formation : common mechanism in plant and animal development. *Int. J. Dev. Biol.*, 40 :123–134, 1996.

[Mei98] Hans Meinhardt. *The algorithmic beauty of seashells.* Springer-Verlag, 1998.

[Mei03] Hans Meinhardt. Models for pattern formation and the position-specific activation of genes. In Sanjeev Kumar and Peter J. Bentley, editors, *On Growth, Form and Computers*, chapter 7, pages 135–155. Academic Press, October 2003.

[MG00] Hans Meinhardt and Alfred Gierer. Pattern formation by local self-activation and lateral inhibition. *BioEssays*, 22(8) :753–760, 2000.

[Mit96] Melanie Mitchell. *An introduction to genetic algorithms.* MIT Press, Cambridge, MA, USA, 1996.

[MP96] Radomir Mech and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceedings of SIGGRAPH 96*, pages 397–410, 1996.

[PHM97] Przemyslaw Prusinkiewicz, Jim Hanan, and Radomir Mech. An l-system-based plant modeling language. In *Proceedings of AGTIVE 1999*, Lecture Notes in Computer Science, pages 395–410, 1997.

[PL90] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants.* Springer-Verlag, 1990.

[Pru93] Przemyslaw Prusinkiewicz. Modeling and Vizualization of Biological Structures. In *Proceeding of Graphics Interface '93*, pages 128–137, 19 May 1993.

[Pru97] Przemyslaw Prusinkiewicz. Visual models of morphogenesis. In Christopher G. Langton, editor, *Artificial life : an overview*, pages 61–74. The MIT Press, 1997.

[QNDR03] T. Quick, Chrystopher L. Nehaniv, K. Dautenhahn, and G. Roberts. Evolving embodied genetic regulatory network-driven control systems. In W. Banzhaf, T. Christaller, P. Dittrich, J. Kim, and J. Ziegler, editors, *Advances in Artificial Life, 7th European Conference (ECAL-2003)*, Lecture Notes in Artificial Intelligence, LNAI 2801, pages 266–277, Berlin, September 2003. Springer, Berlin.

[Rei99] Torsten Reil. Dynamics of gene expression in an artificial genome - implications for biological and artificial ontogeny. In *Proceedings of the 5th European Conference on Artificial Life (ECAL)*, pages 457–466, New York, NY, 1999. Springer Verlag.

[Ren02] Jean-Philippe Rennard. *Vie Artificielle : Où la biologie rencontre l'informatique.* Vuibert Informatique, Paris, France, 2002.

[SdR99] B. Schonfisch and A. de Roos. Synchronous and asynchronous updating in cellular automata. *BioSystems*, 51 :123–143, 1999.

[SDT+92] J. Sulston, Z. Du, K. Thomas, R. Wilson, L. Hillier, R. Staden, N. Halloran, P. Green, J. Thierry-Mieg, and L. Qiu. The *C. elegans* genome sequencing project : a beginning. *Nature*, 356(6364) :37–41, March 1992.

[Smi84] Alvy Ray Smith. Plants, fractals, and formal languages. In *SIGGRAPH '84 : Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 1–10, New York, NY, USA, 1984. ACM Press.

[STK05] Finlay Stewart, Tim Taylor, and George Konidaris. Metamorph : Experimenting with genetic regulatory networks for artificial development. In *ECAL*, pages 108–117, 2005.

[TG07] Andy M. Tyrrell and Andrew J. Greensted. Evolving dependability. *J. Emerg. Technol. Comput. Syst.*, 3(2) :7, 2007.

[Tur52] Alan M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641) :37–72, 1952.

[Wil99] Uri Wilensky. NetLogo. http ://ccl.northwestern.edu/netlogo/, 1999. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

[Wol68] Lewis Wolpert. The French flag problem : a contribution to the discussion on pattern development and regulation. In C. Waddington, editor, *Towards a Theoretical Biology*, pages 125–133. Edinburgh University Press, New York, NY, USA, 1968.

[Wol69] Lewis Wolpert. Positional information and the spatial pattern of cellular differentiation. *J. Theor. Biol.*, 25 :1–47, 1969.

[Wol81] Lewis Wolpert. Positional information and pattern formation. *Phil. Trans. R. Soc. Lond. B*, 295(1078) :441–450, 1981.

[Wol83] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55 :601–644, 1983.

[WW03] Kai Willadsen and Janet Wiles. Dynamics of gene expression in an artificial genome. In *Proceedings of the IEEE 2003 Congress on Evolutionary Computation*, pages 199–206. IEEE Press, 2003.

[WWW04] Pengfei Wu, Xiuping Wu, and Gabriel A. Wainer. Applying cell-devs in 3d free-form shape modeling. In *Cellular Automata, 6th International Conference on Cellular Automata for Research and Industry, ACRI 2004*, pages 81–90, 2004.