Genetic Characteristics of Artificial Agents In FormAL

Lei Zhao

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

March 2004

©Lei Zhao, 2004

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisisitons et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-91157-8
Our file Notre référence
ISBN: 0-612-91157-8

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Genetic Characteristics of Artificial Agents in FormAL

Lei Zhao

This work addresses several issues regarding Artificial Life. It focuses on the behaviours of simulated organisms in such models, e.g. the evolutionary tendencies displayed through reproduction. The scope of Artificial Life is briefly discussed, as well as some arguable issues in this area. An Artificial Life model, called FormAL, is presented. FormAL is a platform capable of running simulations with thousands of organisms (agents), and tracing their evolutionary properties through the changes of their genomes. The design principle of FormAL is to allow the simulated agents maximum freedom possible for their behaviours, with fewest possible 'law's to govern them. The ultimate goal of this design is to give rise to emergent properties of the agents governed with very simple rules. Some interesting results came out of extensive experiments conducted with FormAL. Although there is no explicit fitness function in this work, the selection pressure from competing for the finite energy supply drives the agents to evolve into optimized forms. The test results presented in Chapter 4 and 5 indicate that even in a fairly simple environment, premature convergence can be avoided, and that the mutation mechanism plays a crucial role in evolution. The moment the mutation mechanism is lost, the system stops evolving and the course of evolution reaches a dead-end. This project is still young in its development, and a number of future research directions are discussed at the end of this thesis.

Acknowledgements

I have only inadequate words to express my gratitude to Prof. Peter Grogono, who has helped me throughout this work in many profound ways, with guidance, advice, encouragement and patience.

I also owe much to my family, Ming, Lawrence and Cara, who supported me with great understanding and patience. This would not have been possible without their sacrifice.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Throughout history, the origin of life, and more importantly, nature's ability of inventing infinite novelties, have been regarded as the most mysterious, thus the most researched topics by scientists, and yet, to this day, a large proportion of questions in this field remains unanswered. The invention of the computer brought about a new horizon in this field, accompanied by the research field called Artificial Life, which encompasses a broad range of topics. The very nature of life fascinates computer scientists as well, and using computer as a tool to shed light on many biological issues has captured much attention. And vice versa, a better understanding of the seemingly endless creativity of the biological world has provided inspirations in solving many compurer-science related problems, e.g. optimization in searching, creativity in arts, and improving software designs.

What is Artificial Life? Is it possible to simulate, to study, and furthermore, to better understand the infinite properties of life through a computer-based model? How important are reproduction strategies in the process of evolution for simulated agents? The goal of this thesis is to address these fundamental questions, in an empirical manner, through an Artificial Life model simulating thousands of agents.

An overview of Artificial Life is presented in Chapter 2, together with a review of relevant previous works of Artificial Life models. Arguments about what Artificial Life is, why computer scientists study it, how it has been developed from the beginning of computer-age, what branches Artificial Life encompasses, as well as arguments about the weak points of Artificial Life are discussed. The rest of this Chapter is devoted to the descriptions of the general structural designs and algorithms of the important models in this field, with emphasis on works that do not have an explicit fitness function.

In Chapter 3, our Artificial Life model, called FormAL, is analyzed in details. The reasons for its development, as well as its structures are discussed. Details including its spatial structure, energy and atom distribution, communication among agents, configuration of an agent, structure of the genome, states of the agent, and most importantly, reproduction mechanisms, are explained, together with its global algorithm and user interface.

Chapter 4 is devoted to various results from experiments conducted using FormAL as the platform. Special attentions are given to the problem of early convergence, together with other strong tendencies in the evolution of genomes. In section 4.1.1, reasons for early convergence in simulated agents are discussed and possible solutions are explored. Agents clearly displayed various tendencies in these runs, and they are analyzed and interpreted in section 4.1.2. Four sets of runs were conducted, and the configurations for these runs as well as results are presented in section 4.2. The overall results from these tests are summarized and relevant issues are discussed in section 4.3.

In Chapter 5, results from another group of tests are presented. Agents were simulated in a more complex environment, in order to see whether the conclusion in chapter 4 is valid, e.g. with the mutation mechanism available, agents can adapt to any change of environment for their survival.

There were interesting problems encountered in the process of the program development. These problems and their solutions are presented in Chapter 6. Artificial Life issues presented in this paper, as well as some conclusions from the tests I conducted, are summarized in Chapter 7, and further research work need to be conducted in the future are discussed. Appendix contains the list of parameters used in runs presented in this thesis.

Chapter 2

Artificial Life

"You know, the universe is the only thing big enough to run the ultimate game of life. The only problem with the universe as a platform, though, is that it is currently running someone else's program."

Ken Karakotsios [18](P.37)

2.1 Overview

Artificial Life, defined by Christopher Langton, is "a field of study devoted to understanding life by attempting to abstract the fundamental dynamical principles underlying biological phenomena, and recreating these dynamics in other physical media—such as computers—making them accessible to new kinds of experimental manipulation and testing."[10].

Since the very beginning of the computer age, creating artificial life and studying the evolution of living organism through computer simulations have fascinated computer scientists like Alan Turing, John von Neumann, Norbert Wiener, Christopher Langton, and others [12]. Many Artificial Life researchers seek to find answers to some fundamental questions like "What is life" [6], as that is the base for further defining

Artificial Life. They were "motivated in large part by visions of imbuing computer programs with intelligence, with the life-like ability to self-replicate, and with the adaptive capability to learn and to control their environments. It should be no surprise, then, that from the earliest days computers were applied not only to calculating missile trajectories and deciphering military codes but also to modeling the brain, mimicking human learning, and simulating biological evolution."[12]. Like many biologists, researchers of artificial life attempt to develop a truly universal theory of biology, i.e. not only life as it is, but life as what it could be. "While biological research is essentially analytic, trying to break down complex phenomena into their basic components, Alife (Artificial Life) is synthetic, attempting to construct phenomena from their elemental units. As such, ALife complements traditional biological research by exploring new paths in the quest toward understanding the grand, ancient puzzle called life."[15]. The theoretical base of artificial life research is that the defining feature of life is independent of matter, not limited to the carbon-based material that supports the naturally occurring organisms that we see everyday. These researchers seek to understand how life-like processes can be embodied in computer programs, and Artificial Life models "pursue a two-fold goal: increasing our understanding of nature and enhancing our insight into artificial models, thereby providing us with the ability to improve their performance. An example of the first goal is seen in Von Neumann's research. An example of the second goal is John Koza's work involving software development through evolution."[15]. It is widely believed that by modelling particular natural processes such as evolution, the understanding of those processes and their relevance to the biological world can be improved. The features making computer-based Artificial Life models favourable include

speed and control. It is very helpful to have the ability of creating organisms of arbitrary simplicity, as a tool for studying biology, ecology, the intermediate states that occur on the path to life, and the origin of life. The study of Artificial Life also provides an opportunity of examining and studying emergent properties of life forms that were not conceivable with more traditional methods [17], due to the speed factor.

However, there are arguments against the possibility of creating artificial life. The major argument is that due to the vastness of life and the billions of years it took for biological evolution to proceed, it is not feasible to compress such a process into a computer simulation. Another formidable argument states that, when we simulate evolution, only the important components of the process (according to what the designers opinion) are modeled, leaving out irrelevant details. However, the primary driven force of evolution might very well lie in irrelevant details (they seem irrelevant to us because we do not understand it completely). Even if we could model organisms with tremendous complexity, the complexity of their environment required to provide sufficient selection pressure for the agents might not be possible to be integrated into artificial life systems [18].

Despite these arguments, it is still widely believed that definite answers can only be achieved by investigating into these issues, as Eric Bonabeau and Guy Theraulaz stated: "AL (Artificial Life) is precisely a constructive way of checking whether these limitations are real obstacles"[3](P.314) [18](P.42).

Tremendous amount of research work have been dedicated to this field, especially since the early 1980s, when computers became faster and more powerful, although many important works were completed much earlier. The interest in Artificial Life is still growing.

Artificial Life today encompasses a broad range of research. Genetic algorithms (GA), Evolutionary Programming (EP), cellular automata (CA), synthetic evolutionary models, artificial chemistries, self-organizing systems, Neuro-network and robotics, to name a few, are all branches of artificial life [5] [18] (p18). Most of these forms share some common characteristics. There is a population of organisms; these individuals are able to reproduce by various means; in many cases there is a fitness-function as a selection criterion during reproduction; as the population evolves, the average fitness also evolves; and the behaviour of an individual organism (phenotype) is defined by its genome (genotype).

Artificial Life can be further divided into two classes of models, the first with an explicit fitness function, the second without *(endogenous fitness models [12])*. Since the FormAL project belongs to the latter, I will be focusing on this class.

In the following section, a review of some earlier works is presented. With focus only on issues relevant to the subject of this thesis, it is not intended to be a comprehensive analysis of research works in Artificial Life.

2.2 History Review

John von Neumann, a Hungarian mathematician, was one of the earlier Artificial Life researchers. He devoted much time to the problem of how complicated machines could evolve from simple machines. He believed that life was not a special arrangement of organic molecules, but rather a process that exhibited certain behaviour. If life could be understood and formulated as a process - in particular as a Finite State Machine (FSM), then a computer (which can in principle emulate any FSM) could exhibit lifelike behaviour and ultimately be deemed as alive as any other organism. He claimed that self-replicating computers were the key to intelligence and would ultimately constitute a new form of life. After devoting much time on designing the self-reproducing machines, which was composed of a general constructive machine, a general copying machine, and a control machine [22], he switched his focus and started to devise rules in order that a computer could be composed of appropriately arranged cells, along with other cellular elements that would enable the whole entity to build a copy of itself in the cellular universe. Such entities and systems were dubbed cellular automata (CA) [22]. In his work, von Neumann focused on the logic, rather than the biological, aspects of self-replicating machines, and its ability in evolving increased complication. He was not concerned with problems like fuel and energy, or the self-maintenance ability of biological organisms in the face of environmental changes. Later works on CA share the lack of biological concern as well [18]. "The majority of work (CA) still models purely logical self-reproduction, where the reproducing entities are configurations of states with no material grounding; in such systems, no collection of 'raw materials' is required for reproduction, and from this it follows that there is no competition for raw materials

between individual reproducers. It is likely that only when such considerations are included in these models can we expect there to be selection pressure for self-reproducers with the ability of self-maintenance, potentially leading to the evolution of autopoietic organisms." [18] (P. 50).

Von Neumann's work inspired a great many computer scientists in the search of the origin, evolution, and the very meaning of life, using computer simulation as a powerful tool which enabled them to run experiments otherwise impossible to achieve with traditional research methods. Game of Life, created by John Conway, is one of the best-known examples of CA. The rules governing the states of the cells in Game of Life are very simple, and yet it can give rise to very complex structures.

In the group of Artificial Life where there is not an explicit fitness function, the best-known model is Tierra, created by Tomas Ray. It is an implementation of a virtual computer, simulating a computational environment for population of self-replicating computer programs. The 'organisms' in Tierra, which include genome strings, are computer programs, and they are written in a "specially designed language that is both robust and simple. Programs written in this langrage can be mutated without causing the computer to crash"[19]. The system is evolvable because of this robustness. "The robustness of the language is achieved chiefly through the use of relative or template-driven addressing in branching instructions (rather than absolute addressing), and by avoiding the use of explicit memory addresses as operands to instructions." [18] (P.53). There is no fitness function defined by the creator. The only intrinsic selective pressure

for the organisms is to compete for CPU time (memory space). Self-replication occurs in Tierra when an individual executes instructions that copy its own genome. These copies are influenced by mutation that can change replication process and thus give rise to new organisms. Sexual reproduction is not applied in Tierra's organisms' replication mechanisms. Interesting results have been observed from Tierra experiments, for example, the emergence of 'parasite' programs that cannot replicate by themselves, but utilise its neighbouring programs' code in order to reproduce. Hyperparasites (parasites of parasites) have also appeared [19]. However, when we analyse the model more carefully, it seems clear that the observed emergence of parasites in Tierra has a lot to do with the particular design of the language. Specifically, it is the ability of one agent to execute code in another agent that allowed parasites to evolve, as the flow of control in one program can jump easily to a point in a neighbouring program [20]. The questionable emergence of these complex biodiversity leaves space for further research works.

One drawback of Tierra lies in its user interface. The model does not allow interactive watching and analyzing of the simulation, thus experiments with the system requires a lot of raw data handling. Nonetheless, Tierra has profound influence on the study of Artificial Life, and inspired many models that came after it.

Another artificial life model, called 'Avida', by Chris Adami and Titus Brown, is based on the Tierra design. Cells in Avida can only interact with their nearest neighbours [1], and these local interactions between individuals are closer to real life systems. Self-replication in Avida happens when a segment of computer code copy their genome into a

child string. Once the copying process is completed, the original genome is separated into two identical pieces. The new organism with the new genome replaces the oldest cell within the nearest neighbourhood. Like Tierra, genomes are subject to mutation, resulting in imperfect copies, and " this is the driving force of evolutionary change and diversity in the system" [1]. However, unlike Tierra, programs in Avida can gain more CPU-time by successfully completing user-specified tasks [18]. This important feature means that the programs are given incentive to perform other functions, as well as reproduction. However, selection pressure from these externally defined functions is arguably not analogous to that of the natural world, in which the most important features of evolution comes from the interactions among organisms.

Another model related to Ray's Tierra was developed by Michael Conrad and Howard Pattee. Organisms in this model compete for the possession of chips, which is needed for reproduction and self-repair [18]. This is an extension to Tierra, in the sense that it has concern over matter, which is closer to biological system Yet the authors were not totally satisfied with the model, saying "It is evident that the richness of possible interactions among organisms and the realism of the environment must be increased if the model is to be improved.... the process of variation and natural selection alone, even when embedded in the context of an ecosystem, are not necessarily sufficient to produce an evolution process...the most profound and significant processes of evolution---the innovations, the origins of new hierarchical levels of organization---are still outside the scope of this type of program and remain to be discovered." [18](P. 60).

Perhaps the most interesting Alife model, ECHO, created by John Holland [9], [18](P. 61) is closer to the natural ecological world than most other models, and has the potential of shedding more light on the subject of evolution and organism's adaptive behaviour, than Tierra-like platforms. ECHO simulates the evolution of simple 'agents', instead of computer programs. These agents are modelled at a higher level than those in the Tierra-like platforms. They have a predefined structure, and can interact by mating, fighting, and trading, depending on the conditions and tags of agents encountering each other. At each time step, a fixed amount of resource is pumped into the two-dimensional sites populated by agents, and the competition for this resources, together with mutation while reproducing, cause agents in ECHO to evolve. Sexual reproduction in ECHO means that two agents can mate and reproduce if their mating tags and conditions match, and if they have acquired enough resources in their reservoir [15]. Once they have obtained sufficient raw materials, agents can reproduce by copying, which is subject to point mutation and insertion-deletion, as well as two-point crossover [9]. After crossover, two offsprings replace their parents, and their parents die. If an agent reproduces asexually, by copying its own chromosome, its offspring receives a percentage of the parent's resources. Agents can die if they do not have sufficient amount of resources, and they can also be killed randomly, with a probability proportional to their resources reservation. Agents that died have their resources returned to the environment. Like other models we mentioned previously, ECHO does not have explicit fitness functions coded into its agents [12].

It is widely acknowledged in the Artificial Life field that ECHO has demonstrated promising potential of giving rise to complex ecological phenomena.

Another computational ecology that is in many aspects similar to FormAL is PolyWorld, developed by Larry Yaeger [23]. In this work, "predation, mimicry, sexual reproduction, and even communication are all supported in a straightforward fashion" [23]. Organisms' behaviours are controlled by neural networks, and the organisms have a colour vision, can move around in a two-dimensional environment, eat food or each other, and reproduce sexually. Their physical and neural characteristics are all coded in the genome. When two organisms meet spatially, and if they both are willing to mate, a child organism is produced with a genome resulting from crossover (could be mutated). Tim Taylor gave credit to PolyWorld as saying "Yaeger's model is one of the very few artificial worlds in which distinct species of organisms have evolved and coexisted."[18] (P. 63). One of the problems with PolyWorld is its complexity, which makes it a difficult model to perform extensive experiments [18]. In this model, the simplicity, favoured by Packard as saying "I make every attempt to strip down most of the complexity of real biological systems, with the aim of discovering a minimal model that displays evolutionary behaviour"[18](P.62), is missing.

An ad hoc fitness function is used in PolyWorld, similar to a GA, until its population can sustain its numbers through births [23].

Another agent-based package, called LEE, written by Rich Belew et al, combines neural networks, genetic algorithms, and organisms (agents). Organisms in LEE have sensors, neural network, motors and energy stock. They live in a two-dimensional grid, into which resources (atoms) are distributed. Agents can sense or move, and all actions cost certain amount of energy. Instead of an explicit fitness measure, selective pressure among the LEE population comes from intrinsic competition for resources. The behaviour of an agent is allowed to improve through experience (learning) [11]. When agents in LEE obtained enough energy, they can reproduce asexually. Mutation can occur at time of reproduction, and reproduction is controlled by parameters in the GA [6].

M. Epstein and R. Axtell developed a computer model in 1995, called Sugarscape, for studying human social and economic behaviour. The agents of Sugarscape live in a two-dimensional landscape, with some built-in attributes, which direct them to perform certain actions. They move around to look for resource, called sugar [14], and after many steps, emergent phenomenon somewhat resembling human economic society appears: a few agents have most of sugar and the rest live in relative poverty. These agents also have cultural attributes [2], and they could engage in cultural interactions. Agents in Sugarscape have gender programmed in them, and they select a neighbour at random for mating. A child will be produced if the neighbour is of opposite sex and of reproductive age.

A different model worth mentioning here, called swarm, is a general-purpose platform, on which computer simulations can easily be built for studying in various disciplines [8].

Users from these fields can write code specific to their needs, and not have to worry about general foundations like user-interface or analysis tools. Typical Swarm simulations involve large amount of agents interacting with each other and with the dynamic environment (most Swarm experiments use two-dimensional lattice space) [8]. Swarm agents usually reproduce and die according to a given probability, and most actions by the agents cost energy. Agents search the neighbourhood to find another suitable agent for mating.

The last simulator I want to talk about is called Gaia, designed for the study of certain aspects of ecology and biology [4], and was inspired by L. Yaeger's PolyWorld, with some added features. There are two types of organisms in the two-dimensional world of Gaia: Heterotrophs (critters) and autotrophs [4]. Critters can move, eat, fight, and mate. They have a "nervous" system and their genomes define some structural and physiological characteristics [4]. Autotrophs can grow only according to a predefined distribution and rate. They are food for heterotrophs and are the only energy source. An offspring is produced when two critters meet and both desire to mate. The newborn organism is located close to its parents, and the parents transfer a certain amount of energy to the offspring. Only mutation and crossover are used as reproduction operators.

Chapter 3

Design Document of FormAL

3.1 Design Philosophy

Inspired by artificial life models reviewed in Chapter 2, especially by ECHO, an artificial life model called FormAL (stands for 'Formal Artificial Life) was developed. It is implemented using C++, and OOP method. The intent of this work was to explore issues in Artificial Life, especially in the study of evolution, with a model based on simple rules (laws), and expect to see emergent behaviour (life) in the simulated organisms. It is an evolving system that simulates living organisms capable of reproducing, accumulating energy, and absorbing information from their environment. A population of organisms, called agents, reside in a 3D world, reproduce, and compete for resources. The goals of this project included the following:

1. Providing a platform with rules as simple as possible, to explore various issues in Artificial Life (the 'laws' and 'life') [5].

For ensuring precision, a careful distinction between 'laws' and 'life' was made. The unchangeable features of the environment are defined by laws, and the changeable factors occurring in the process of evolution constitute life. In FormAL, properties defined by the

designers are 'laws', and the adaptive features by the agents are 'life'. The goal was to provide laws as simple as possible, thus giving the agents maximum liberty to evolve on their own, without predefined rules. Giving the simulated agents as little law as possible, yet at the same time aiming to give rise to complex behaviour, has proven to be the most challenging task while designing and implementing this project. Through the experiments, it was revealed that a compromise had to be made regarding this issue, as too few rules would not allow agents to display any evolution within practical time period. The current platform is fairly simple and can be understood easily, yet provides a base allowing the agents to evolve, according to the living conditions of the environment.

2. Exploring ways of providing simulated agents with means to develop strategies, rather than with strategies directly.

This goal is related to the first, in the sense that as few rules as possible should be provided to the agents as they evolve. The challenge is to determine a simplest set of rules with which the agents can evolve strategies for their survival, and adapt to any changes occurring in the environment.

3. Providing insight into the Gould versus Wolfram debate [5].

It is widely believed that evolution increases complexity. "Stephen Jay Gould argued that evolution does not take the form of a monotonic increase in complexity from amoeba to plant to ape to Charles Darwin. The tree of life is a broad bush in which most branches lead to extinction, not a spindly Victorian tree of progress. Gould did argue, however, that evolving organisms will tend to increase in complexity as time passes...Stephen Wolfram disagrees with this argument. He believes that complexity arises from very simple mechanisms . .... Wolfram argues that evolution is a simplifying mechanism: if we start with a collection of simple automata with rich behaviour, evolution will find the simplest and most economical mechanism that suffices to solve the problem." [5] (P. 2).

In the experiments conducted (Chapter 4 and 5), the agents were given a genome that was more complex than was required to exist in the environment, and during these runs, the majority agents evolved to a genome much simpler than the ancestor genome, which seemed to favour Wolfram's opinion. However, this was probably due to the fairly simple environment, and as this environment got more complicated, as in Chapter 5, agents did show signs of adapting by the utilization of other genes. Complexity as an emergent property could evolve under suitable conditions, therefore more precise insight into this issue requires further research.

4. Avoiding the 'mere optimization' problem.

This model was designed with the goal of avoiding the limitations of optimization. Through the development of this work, it has proved difficult to avoid optimization completely. As mentioned in Chapter 2, no fitness functions were imposed on the agents. In long runs, majority genomes became extinct, indicating that the genomes left was in some sense more fit than other genomes created during that run. Thus, although there is no explicitly programmed fitness function, the simulated environment imposes constraints on agents that can survive, and agents evolve to satisfy those constraints as efficiently as they can. It is hoped that with further modifications to the system, with the environment implemented with more complexity, the agents will develop "both a variety of strategies and also the ability to apply them in appropriate situations" [5] (P. 3).

5. Providing better understanding of the evolutionary behaviours, and of the relation between early convergence and reproduction strategies.

FormAL is a viable platform for conducting experiments, in order to observe the evolutionary characteristics of agents in a simulated system, and discover the relation between convergence (end of open-ended evolution) and reproduction strategies (see Chapter 4). The second issue is very important in the sense that a mutation rate analogous to that of nature, which is very low, yet providing a possibility for adaptations should the need surface with changes of environment, has to be determined, in order to see the effect of evolution in a context bearing similarity to that of the biological environment, and to provide results of runs that are meaningful in a biological sense.

3.2 Structures of FormAL

The current design details of FormAL are presented in this section, as well as the rationale behind them. Since this is an open-ended project, changes in the design may occur in future development.

3.2.1 The Environment

3.2.1.1 The Space

The spatial structure in FormAL is a three-D space with a Euclidean metric. A 3D space offers more potential for the interactions among agents and their movements than does a 2D space. Other spatial structures were considered, e.g. Hamming distance, in which the position of an agent is defined by a string of bits, and the distance between two agents is represented by the number of bits that differ. This method would have been computationally efficient, but not as intuitive as a Euclidean space. This 3D space is continuous, divided into a finite grid of cubical cells [5] (P.3), in which agents reside. A population of agents of a predefined number (which can be set by users) is created by the system during initialization period until a threshold is reached, and they are assigned randomly to the cells. Once the minimum population is reached, the system no longer creates new agents and the population is on their own for survival.

3.2.1.2 Atoms and Energy

The concept of matter in FormAL is defined by 'atoms', which are symbolically represented by the 26 lower case letters of the alphabet. The environment in the simulation makes changes of its atom collection in the form **a→f→k→p→u→z**. This effectively adds energy to the environment. An agent gains energy by collecting high-energy atoms, converting them to low energy ones, and discarding the low energy atoms into the environment. Agents can use the inverse transformations (**z→u→p→k→f→a**), as in the current simulation, but they are not restricted to these transformations. The higher the letter in the alphabet, the more energy it contains. The total number of atoms in the environment is conserved. Upon its death, the atoms of the individual agent are transferred back to the environment or to another agent (depending on the cause of death, see chapter 5), which could give rise to selection pressure for the evolution of agents who kill other agents to get their atom collection.

During each time step in a simulation, an amount of energy is distributed into each cell by converting the low-energy atoms to high-energy ones stored in the bag of the cell, and agents can transfer high-energy atoms from their cell into their own storage (bag), and obtain energy by converting them into low-energy atoms. The amount of energy each cell receives could be different, as the case in chapter 5.

Much thoughts were given to the design of atoms and energy, since in many earlier AL works, consideration with matters is lacking, e.g. in Tierra, the only resource for organisms to compete for is CPU time, and memory. It is hoped that with this material

grounding, the system will provide selection pressures for the agents that is more analogous to that in biological evolution.

3.2.1.3 Communications

In FormAL, there are two types of signals an agent can send or intercept: ready to kill and ready to mate, controlled by their genes respectively. In future development, we might allow arbitrary messages to be exchanged among agents, thus providing the agents with a richer environment. An agent may send a certain type of pheromone, by converting a small number of the 'z' atom into the atom representing this type of pheromone, and release them into the local environment (transfer these atoms into the cell's storage). An agent may also detect the cell containing the most pheromone, and move toward or away from it. Once an agent locates another agent containing the same mating pheromone as its own, they will mate and produce offspring by crossing over their genomes.

The act of eating in the basic version of FormAL remains very simple: an agent may detect another agent with a part that contains the most atom, and eat this part (cut it off from its parent, and transfer this part onto its own body as a new part). In Chapter 5, a more complicated behaviour is described with tests results. Agents can kill others and obtain their atom collection, adding complexity to the surviving environment of agents.

An agent may move around during each step, if it has enough energy, and consume a required amount of energy. The agent chooses a force and direction, and then

Newton's law is used to compute the new motion. An agent may also move toward the cell that contains the most atoms, and move toward (or away from, depends on the value of the argument) the cell with the most agents. It can also approach an intended prey, or move away from a cell containing more hostile signals.

3.2.2 The Agent

All the actions an agent can perform are controlled by its genes and their arguments, therefore mutatable, and to certain extent are within the agent's control, instead of being set by the programmer. Therefore they belong to 'life'.

3.2.2.1 Structure

An individual organism in FormAL, called an agent, may be a body or a part, and is represented graphically by a cube. There are two reasons for thinking of it as a cube:

(1) Since cubes are easy to draw;

(2) A body or part can "grow" by adding parts on any of the six faces that a cube has, and may continue growing in that direction.

A body is an independent agent, with zero to six direct parts. A part has to be attached to a body or other parts, and therefore is not physically independent. An agent also contains a bag, which is the storage for its atoms, analogous to a stomach. During initialization period of simulations, as mentioned in section 3.2.1.1, a minimum number of agents are created by the system, and they are all bodies, with no atom and some initial energy. When a body reproduces, its offspring is a part attached to the body, like an embryo (The

detailed process of reproduction is explained in section 3.2.2.2.). An agent has an age property (see section 3.2.2.2) that is measured by time steps, and controlled by a gene.

The most important property of an agent is the genome (genotype), which is the encoded instruction for the agent's behaviours (phenotype), following the conventional biological model. It plays a double role in the simulation. It is treated as data when being copied from a parent to a child, and as instruction when being decoded to determine the agent's behaviours.

The structure of a genome is a string of atoms (represented by the 26 letters of the alphabet). The genome is composed of genes, where each gene is represented by a group of three letters, also called triplet [5]. In the group of three letters, the first stands for the probability for this gene to be activated during each time step, the second is the action of the gene, and the third is the argument of the gene, which affects the performance of the gene in various ways. This structure provides a potentially vast behaviour space, with up to 26 genes, each combined with up to 26 arguments, coming out with 676 different actions. Currently only a portion of the genes are implemented (See table 3.1 for the genes used in FormAL). From the definitions of these genes, it is obvious that their performances are defined at a fairly high level, and we hope that through future development their effect can be simplified and they can dictate the behaviours of the agents at a lower level, thus leaving more space for the agents to evolve on their own.

During mutation, any one of the three letters of any gene could be changed into a randomly selected letter, and an existing gene could be deleted or a new gene added, thus varying the length of the genome. In crossover, the first part of the mother's genome is combined with the second part of the father's genome (one point crossover), the crossover point could be anywhere in the genome, and the new genome has a slight chance of having a different length than that of its parents.

	Body	Part
a	aging	
b		
c	convert high-energy atoms to low-energy ones to acquire energy	convert energy
d	reproduce by cloning	grow in new direction
e	regulate energy	regulate energy
f		grow in current direction
g	get atoms from the environment	get atoms
h	kill other agents	
i	start interaction	take atoms from parent
j	respond to pheromone	
k	emit pheromone	
l		
m	regulate mass	regulate mass
n		

o		
p	dispose atoms to the environment	dispose atoms
q	mutate (currently used as another copy gene)	
r	reproduce with crossover	
s	detect the cell with the most atoms, and move toward it	
t	terminate interaction	
u	detect the cell with the most agents, and move toward or away from it	

Table 3.1: Genes in FormAL

3.2.2.2 States

An agent can be in one of three states: active, inactive and dead, determined by the amount of energy or atoms it contains. When its energy or atoms stay below some thresholds for a certain amount of time, the agent is declared dead (with the exception of the agent interacting with another one) and will be removed from the agents' list. The agent also has a higher chance of dying as its age approaches a threshold, but this mechanism is controlled by one of its genes and agents could learn to mutate that gene, as they always seemed to do (see Chapter 4), as an effort of improving their longevity.

3.2.2.3 Reproduction

Reproduction is doubtlessly the most important action an agent performs, as in most of AL works, since it is the means of preserving the population. The reproduction mechanisms in FormAL are similar to those used in Genetic Algorithm models. The

original design had reproduction activated by 3 genes: reproduce by exactly copying the agent's genome into its offspring; reproduce by mutating one of its genes; and reproduce by combining the mother's and father's genomes (crossover). The mother agent is the one bearing the offspring as its parts. During the experiments, this configuration revealed some drawbacks (see Chapter 4, set 4), e.g. since most mutations are harmful, evolution tends to eliminate the mutation gene. When the population no longer contains mutation genes, evolution is restricted to crossover, which shuffles existing genes but does not introduce new genetic material. The usual effect of losing the mutation gene is that the gene pool converges to a single genome. To solve this problem, the current configuration is modified, and reproduction is performed by either the copy gene or crossover gene, each with a low probability (0.3%) of activating the mutation mechanism.

As mentioned in section 3.2.2.1, when an offspring is reproduced by one of the means above, it is a part, attached to its parent. This method of reproduction is unusual for GA and AL work, in which offspring are normally launched directly into the world. It is analogous to mammalian reproduction, in which the offspring is supported by the mother for a while before it becomes independent. Also unlike many AL works, a parent in FormAL is not replaced by its offspring, but continuously living with its offspring. The mechanism works as this: When an agent is about to reproduce, it checks to see if it has enough energy. If so, one of its 6 part-locations is selected randomly and checked. If it is unoccupied, the newborn will be attached at this location as a part. If it already contains a part, the old part will be promoted into a body, detached from its parent, become independent, and leave space for the newborn offspring. The new part gets half of its

parent's atoms, and is inserted into the agent list as a part. A part is created without energy, and when being promoted into a body, it receives a proportion of the parent's energy (determined by the value of argument for this gene, therefore evolvable), and keeps its own atoms.

A part may also grow new parts (called divide). The problem with the "growing cube" model is that time is required to compute whether a new offshoot overlaps an existing one. Here is one possible algorithm:

(1) Choose a possible location L for the new part

(2) Find the body that owns the parent of the new part

(3) Recursively find the set S of locations of all parts owned by the body

(4) If $L \in S$, go back to step (1)

This could get very slow, especially if agents grow to have hundreds of parts. Consequently, we use a heuristic method that makes it unlikely, but not impossible, for a new part to have the same location as an existing part. The heuristic is that the location of a new part is usually such as to extend the agent in the same direction as before, thus reduce the chance of overlapping with other parts (see Chapter 5), with a small chance of growing the part in other directions, in which case one of the remaining four locations is selected randomly, and if it is free, it will be occupied by the new part, which gets half of the original part's atoms, and has the exact genome (reproduction by copying).

A related problem is that agents can overlap each other in the space. There does not seem to be an efficient way of preventing this, but the limited energy supply from the system helps to prevent overcrowding in each cell.

When a part is promoted into a body, its entire offspring parts are promoted into bodies as well provided it has enough energy to do so, and each promoted part gets a proportion of the original part's energy. If it does not have enough energy, its parts will remain parts of the newly promoted body.

The design of agents as bodies and parts adds biological meaning to the system (simulates embryo development), and provides potential complexity for the agents' behaviours.

3.2.3 The Algorithm

Simulation of FormAL advances in discrete time steps, during which all existing agents' behaviours, encoded in their genomes, are executed. The main algorithm of FormAL is given as pseudo-code, in table 3.2 and 3.3. The initialize function is activated only once, at the beginning of runs. The step function is executed at each time step.

```
Initialize:
{
  record the starting time of the simulation
  set the location of each cell
```

```
  insert a predefined number of atoms into each cell
  locate all the neighbours of each cell
  prepare the logfile for output data
}
```

Table 3.2: The Initialization function

```
Step:
{
  if(duringInitializationPeriod)
     if(population<minPopulation)
     {
        randomly select a cell
        create a new body in this cell, with the starting genome (or genomes read from
                the genebank) and some initial energy
     }
     insert a predefined amount of energy into each cell
     execute each agent's actions encoded in its genome (if its prob is right)
     each agent makes a random move
     update each agent's position in case it has moved to another cell
     check each agent's state, and remove dead agents
     update the list of agents
     output data to logfile
}
```

Table 3.3: The Step function

3.2.4 User Interface

FormAL has an intuitive user interface, providing a visual presentation of the simulation, and allowing users to set many parameter values other than the default ones, to record various data to logfile, as well as to watch the changes occurring in the process of a run.

There are three graphic windows in the user interface (see figure 3.1). The model window provides the visual effect of the run, showing the agents in their world (a cube), where different colours represent different states of the agent, e.g. pink for the active state, blue for less active, black for dead, yellow for interacting, and green is for a part, with different shades representing different levels of energy. The colours are arbitrary and easily changed: a different choice of colours might provide more information. We could also use shape to convey something about an agent's state. The visualization was important during the early stages of development so that we could get a quick idea of what was happening.

The statistics window displays various statistic data and their changes, including sizes of population, energy level, time-steps, energy per step, total number of bodies, total number of parts, atoms in each cell, atoms in agents, number of clone performed, number

of mutate, crossover, and number of eating performed, number of interactions, and the current active genes, etc.

The control window provides an interface for users to set values for the minimum population the system is required to generate during initialization, the birth rate, the energy and atoms injected into each cell during each time step, and allows users to input random values for these parameters as well. Users could also interrupt the run temporarily. The viewing angle for the model window can be adjusted, and help instructions are available. Reports on information about agents, cells, genes, vectors, calculation of probabilities, and coordinates can all be generated into the output logfile as extra data (a default genome report is automatically generated by the system).

The rich and intuitive user-interface of FormAL makes it easy to conduct experiments and analyse the simulations.

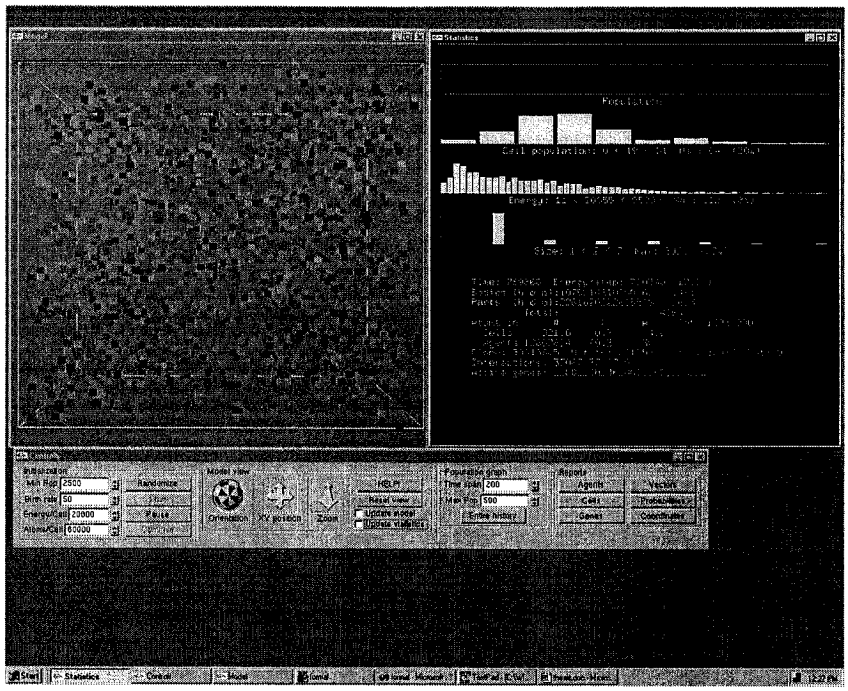

Figure 3.1: The Graphic Windows

3.2.5 The Output File

Test results regarding genomes, information about agents and cells, coordinates, and probabilities can be written into the output logfile for further analysis.

Chapter 4

Experiments in FormAL

In this chapter, a variety of formAL experiments are described. First, the relevant system behaviours of interest are listed, and then different results from these experiments are presented and analysed in details.

Most of the simulations were conducted for 1,000,000 time steps, except earlier tests in set 1 and 2, where early convergence happened and tests were terminated when agents stopped evolving. Four types of graphs are presented in this chapter. In the first type, the number of genomes is plotted against time steps. Second type shows the total number of genes in the majority genome, including repetitive ones. The third displays the number of actual distinct genes in the dominating genome through out the runs, and the fourth type shows the resultant majority genomes.

4.1 System Behaviours

The relevant system behaviours discussed in this chapter include the rate of convergence (if they do converge), traced by the change in the number of genomes in the process of runs, and solutions for early convergence; the noticeable tendencies in genome evolution;

and the method of distinguishing contingent (chance) factors from those of generic features, factors that may affect the end results of these runs.

4.1.1 Genome Convergence

In FormAL runs, like those of some other AL models, early convergence of genomes was encountered, which prevented any possibility of open-ended evolution (although this word is generally used with a more specific meaning, such as evolutionary arms races and sexual selection, here it implies that "new, adaptively successful individuals continuously appear in the populations—evolutionary activity does not peter out" [19] (P.80). The convergence, usually caused by the disappearance of the mutation gene, seemed inevitable at times, and led to a halt of further evolution and prevented the formation of any interesting gene pools.

Once left with very few numbers of genomes, the crossover gene would work just as the copy gene, in the sense that it wouldn't provide any varieties. However, the tests showed that for the crossover gene to be activated, the mutation mechanism had to be effective so that the agent could get hold of enough get-atom genes and thus obtain enough atoms in order to be qualified as a mate.

Even if agents all evolved to a converging point, as long as they kept the mutation gene in their genomes, they still retained the ability of changing and evolving, as noticed in run 3 of set 1 and run 1 in set 2, where agents converged into one genome, then started to diverge again.

It seems that the relation between the simulated population's ability of evolution and the frequency settings of reproduction methods has not been explored by other AL models. In FormAL, after many test runs with parameter adjusting, it was revealed that the frequency of the activation of the copy gene had a strong effect on the agents' ability of retaining the mutation gene. A little restraint on the frequency of the copy gene slowed down the rate of convergence significantly. That is, agents in FormAL tended to reproduce by copying themselves, if given a choice, and shed off the mutation gene as soon as possible. They achieved this in the course of mutation, during which they sooner or later got hold of a copy gene with high probability. When agents got hold of another copy gene, their reproduction rate by copying was fast enough that it was no longer advantageous (they could afford to lose the mutation gene) to keep the mutation gene as a reproduction method. This result indicated that the mutation mechanism was not favoured by evolution in this current setting of FormAL, and that the agents preferred to reproduce faithfully. The benefit of mutation might manifest only when the environment becomes more complicated, at which time the agents will have to keep evolving to adapt to the changes of the environment.

In the efforts of achieving open-ended evolution in a rather simple environment, emphasis was given to the fine-tuning of the frequency of all three reproductive methods at first. However, extensive tests showed that as long as mutation was implemented as a gene, there was always a chance for it to get lost in the process of evolution.

The current solution to this problem was that the mutation mechanism was implemented as a small possibility in other reproduction genes, rather than being a gene in the genome. This design prevents agents from getting rid of mutation all together. This proved to be a sound solution for keeping open-ended evolution in a fairly simple system, where selection pressure from the environment is minimal. Even in a more complex environment, it might still be necessary to impose mutation onto the system, in order to guard off any chance event and ensure the retaining of this mechanism

4.1.2 Tendencies in Genome Evolution

Genomes in most of the runs appeared to have a surge of divergence at the beginning, and then quieted down to a small number, or to a single genome.

In all of the runs in which the agents had a chance to evolve, the following functional genes were shed off:

'A', and 'P'.

They were for aging and dispose atoms to the environment respectively. Clearly they were not advantageous properties and not favoured by the process of natural selection. Many runs produced genomes that included very few genes from the ancestor genome. The aging gene is usually the first to be lost, meaning the agents did not like to have a life expectancy imposed on them, and liked to increase their life span.

The lengths of genomes in most of these runs appeared to grow longer at first, later settled down to shorter ones (e.g. in most of the runs of set 4, the resultant genomes were 70% shorter than the ancestor genome). The number of total genes in the graphs showed the overall numbers of genes, which could include duplicates. The number of distinct genes was usually smaller than the total number of genes, since there were repetitive ones.

In most of the runs, the resultant genomes showed that agents learned to acquire many of the 'M' genes, for obtaining atoms and regulating mass. This could be regarded as another sign of evolution, as this gene allowed agents to get high-energy atoms from the environment, and so did the 'G' gene. It was also noticed in most of the runs that agents mutated low-probability letter of the gene for getting atoms from the environment into a high probability letter and thus accumulated more energy and atoms from their environment. This could explain the fact that the population size would not shrink towards the end of runs, when agents were usually left with only one reproduction gene, because they would have had enough energy to perform reproduction every time that gene was activated.

In runs which convergence happened early, the genomes did not have a chance to evolve much from the ancestor genome other than the loss of the mutation gene.

It was also revealed through these experiments that at the early stage of runs, agents did not have enough atoms (energy) to perform all reproductions even if they were activated,

therefore in most cases three reproduction genes were required in the current configuration for keeping the agents from extinction (except for a few runs in section 4.2.2). However, once agents started to evolve and learned to acquire more atoms from the environment, they would accumulate enough energy for any reproduction gene that was activated, thus sustaining a stable population with only one reproduction gene in their genomes.

The population sizes were fairly constant in these runs, therefore no graph were presented for this matter. The reason for the stability of population size must be the constant energy supply from the environment, which was sufficient for supporting certain number of agents, and creating a ceiling effect on the population. This might affect the end results of the genomes, since many new off springs could not survive, due to the lack of atoms available in the environment.

4.1.3 Contingent Factors

It was widely believed that the course of evolution on earth was affected by generic factors as well as contingent events.

In the effort of distinguish contingent factors from more generic features in FormAL, same runs were performed with different seeds for the RNG (generated by clock). Results showed that important features discussed in this chapter, like the shedding of apparently harmful genes and the addition of get-atom genes, happened in all of the runs, indicating

clearly that they were not contingent events. The rate of convergence differed among earlier tests, which might indicate the influence of chance factors.

4.2 Experiments

Four sets of runs are analyzed in this section. They all started with the same handwritten ancestral genome, designed in a random manner, which included all useful genes as well as some junk genes:

<aAb bBb vCo bDl fEw fFa dGf aHp oIi rJg nKm lLl yMo nNn jPe eQy dRq wSt tTt jUs>,

except for a few exceptions in set 2, which started with either genomes read from the genebank or a somewhat less complete genome:

<kAk qCn fDb xEv fFa uGp gHx oIp hJi fKi wMw kPd xQm eRy gTa wUx>,

and again in the run 4 of set 4, with a modified genome.

As mentioned in section 3.2.2.1, the genome is organized as groups of three letters. The first letter refers to the frequency of the gene being activated, e.g., 'Z' is 100% probability (i.e., always) and 'a' is about 0.13%, the second letter stands for the gene (see table 3.1), and the third is the argument letter, which influences the behaviour of agents when that gene is activated.

The reason for using only one ancestor genome to start these runs was that the effect of evolution would be clearly presented in the test results, by comparing the resultant genome with the ancestral genome, and it was inspired by the design in Tierra [13].

The agents in these runs could reproduce by copying themselves, copying with mutation, and crossing over with another agent's genome, in the same time step. The first set describes the standard runs, performed with the same parameters except for the seed for the random number generator; the second set includes a variety of miscellaneous experiments with small differences in their reproduction parameters; the third set shows results from the model after fine-tuning the reproduction genes' frequency, again keeping other parameters the same except the random seed; and the fourth set presents a different approach to mutation, which had mutation not implemented as a gene, but rather as a small probability in other methods of reproduction instead.

The experiments were focused on the reproduction behaviours of agents and their effects on the genomes, especially the convergence rate, represented by the change in number of genomes throughout the course of runs. Considering the vast parameter space of this model, these experiments are therefore by no means claimed to be comprehensive. Runs with different parameter settings for energy and atom consumption, tracing of the spatial behaviours of agents and life span of agents, etc. were beyond the scope of this paper and were not attempted.

4.2.1 Experiment Set 1---Standard Runs

In the standard runs, genomes of agents could be mutated in three ways during reproduction: (i) by randomly flipping any letter in the genome; (ii) by randomly deleting an existing gene (with a probability of 1/13, which comes to about 7.7%); or (iii) by inserting a new gene (again with probability of about 7.7%). While crossing over, the offspring's genome selected from the part of father's genome had 1/13 chance of being one gene shorter and 1/13 chance of being one gene longer, thus giving agents another chance of varying the length of the offspring's genome.

Four runs were conducted in this set, which differed only by the seed for the random number generator. Run 1 was generated with the default seed of the system, and the other three runs were seeded by clock. Random numbers are used in various places in the simulation, e.g. agents are assigned a random position when created by the system in the initialization period; the argument letter, which dictates certain behaviours of agents, are selected randomly; and as indicated in chapter 5, a randomly selected amount of energy is distributed into each cell for the purpose of making the environment more complex.

In run 1, agents somehow managed to retain the mutation gene (see figure 4.4), and their genomes had not fully converged by the end of the run (see figure 4.1). The majority genomes were 20% shorter than the ancestor genome (see figure 4.2 and 4.3).

In the other three runs, as we can see in figure 4.5, 4.7 and 4.11, at a fairly early stage agents lost the mutation gene in their genomes and started to converge at the same time,

after the initial period of divergence. The lengths of their genomes did not have enough time to evolve (see figure 4.8, 4.9, 4.12 and 4.13).

In run 2, an extra copying gene (gene 'D') was inserted and the mutation gene (gene 'Q') was no longer present at as early as step 10,000, which led to the complete convergence at step 30,000. Because the evolution process stopped quite early in this run, the resultant genome didn't have a chance to evolve much from the ancestor genome, and differed mainly by the addition of an extra copy gene and the disappearance of mutation gene. The lengths of the majority genomes did not have time to change either.

The mutation gene disappeared at step 150,000 in run 3. The genomes actually converged before this, but thanks to the mutation gene in the genome, it managed to diverge again for a little while after, until it lost the ability to mutate and came to a full convergence. Again the aging and dispose atom genes disappeared from the resultant genome, replaced by four regulating-mass genes (see figure 4.10).

Same scenario happened in run 4 at step 11,000, when it lost the mutation gene.

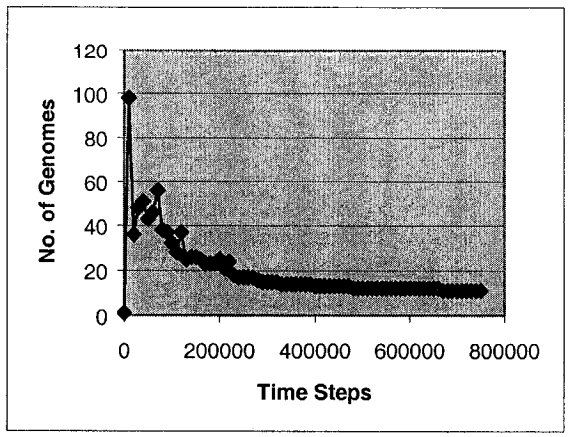

Figure 4.1: No. of Genomes in Set 1, Run 1

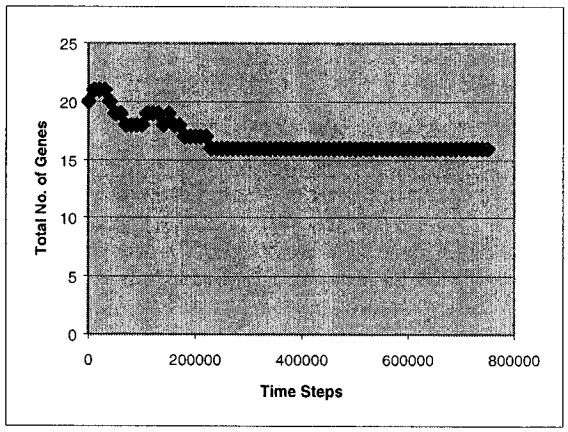

Figure 4.2: No. of Genes in Set 1, Run 1

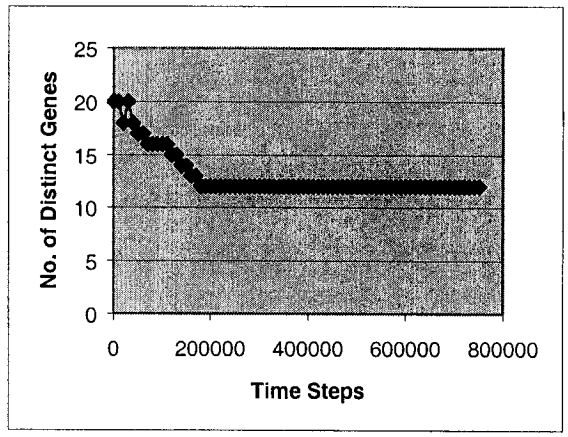

Figure 4.3: No. of Distinct Genes in Set 1, Run 1

4632 99% <aQb vCo fFa uGf aHp nIi wMm yMv nNn dRq wSt tMt xUs mDc yMm yMy>

2 0% <aQb vCo fFa uGf aHp nIi wMm yMv nNn rQy dRq wSt tMt xUs mVc yMm yMy>

Figure 4.4: Genome results in Set 1, Run 1

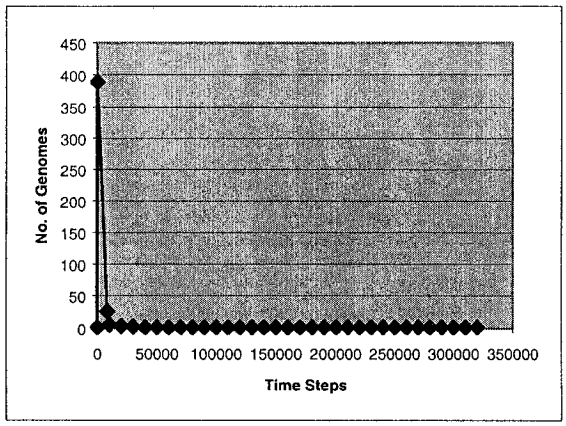

Figure 4.5: No. of Genomes in Set 1, Run 2

100% <aAs bHb vCo bDl fEw fFa pGf aHp oIi nKo lLl yMo nNn jPe wJy dRq wSt tHt jUs mDc>

Figure 4.6: Genome results from Set 1, Run 2

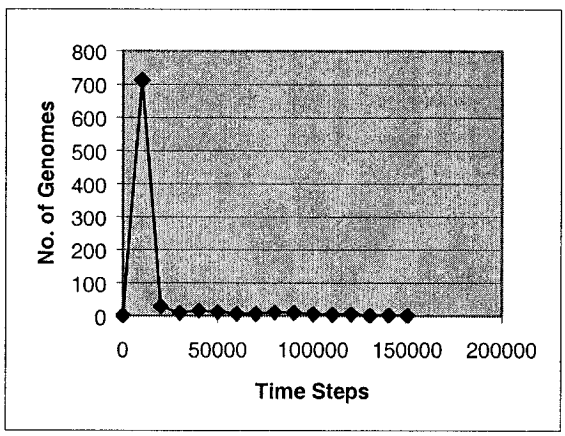

Figure 4.7: No. of Genomes in Set 1, Run 3

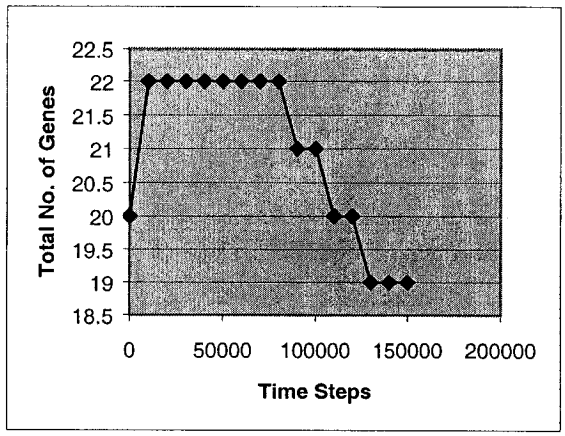

Figure 4.8: No. of Genes in Set 1, Run 3

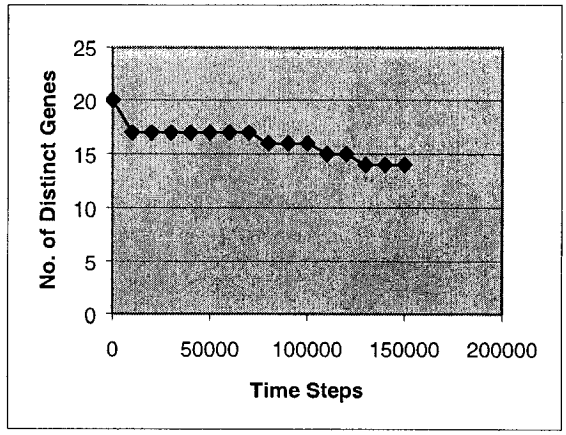

Figure 4.9: No. of Distinct Genes in Set 1, Run 3

100% <oDb vCo vMo bDl xEw fFa vGf aHi oIo xGg nMm yMo nNn wSy dRq wSw tTt xMn aLa>

Figure 4.10: Genome results from Set 1, Run 3

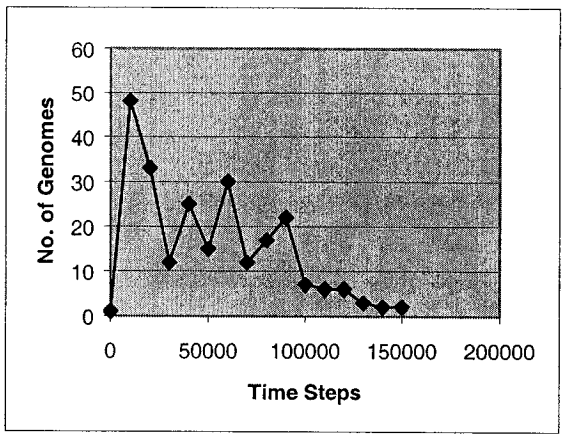

Figure 4.11: No. of Genomes in Set 1, Run 4

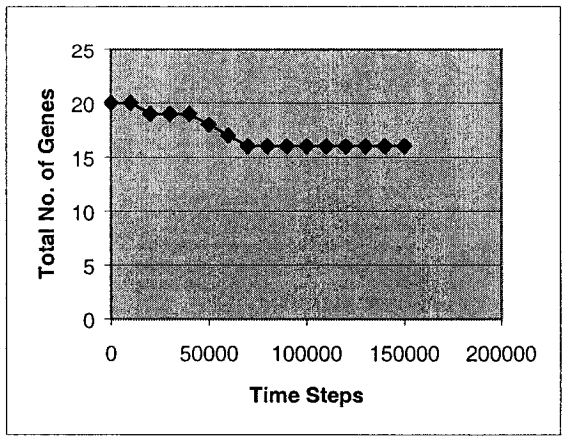

Figure 4.12: No. of Genes in Set 1, Run 4

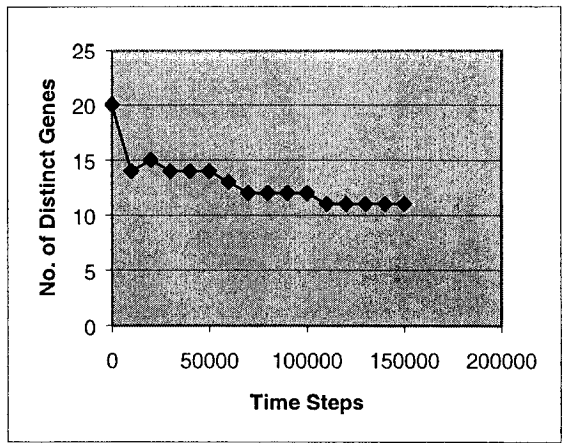

Figure 4.13: No. of Distinct Genes in Set 1, Run 4

99% <vCo bSl fEw fFa xGf oIi rGg nKm lLl yMo nDc uJy dGq xSw tMt jCs>

Figure 4.14: Genome results from Set 1, Run 4

4.2.1.1 Experiment Set 1.1---Standard Runs with Variations

In the effort of finding the desired balance for the frequencies of different reproduction methods, four more runs were conducted with certain parameters explored, in this subset of standard runs.

In run 1, only existing genes with the same arguments could be inserted through mutation, instead of random insertion as in the standard runs. This ensured that any addition of the reproduction gene would not have a very high probability. The system inserted an extra crossover gene and got rid of the mutation gene at step 15,000.

In run 2, no reproduction genes were allowed to be added to the agents' genomes, and it produced similar result. It did not have a chance of adding any mutation gene, and lost the existing one at step 39,000, and as usual, it converged at the same time.

Run 3 was conducted with the copy mechanism limited through mutation. It allowed the copy gene to replace an existing gene in the process of mutation, as long as the frequency letter prior to it was less than a threshold. However, although happened at a later stage, the mutation gene still disappeared at around 140,000 steps, leading to a convergence at the same time.

In run 4, the copy mechanism was further controlled in mutation, which did not allow the copy gene to replace any existing gene and the probability of the copy gene was not allowed to be changed. From figure 4.15 we can see the number of genomes was highly volatile, possibly indicating too frequent mutations for any meaningful evolution.

It was revealed through these experiments that when cloning was turned off or limited, it would affect the convergence rate. The Agents would need the mutation gene as one of the vital reproduction method, thus giving the mutation gene a better chance of sustaining for a long period of time.

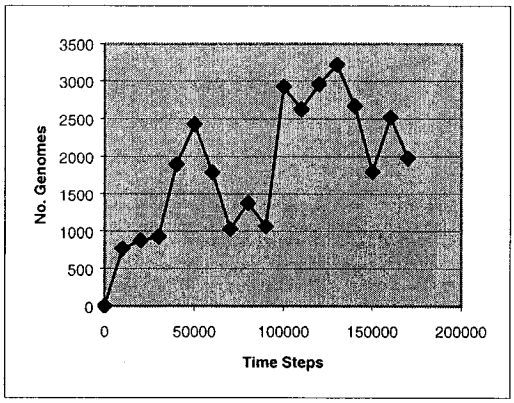

Figure 4.15: No. of Genomes in Set 1.1, Run 4

4.2.2 Experiment Set 2--- Miscellaneous Runs

A group of miscellaneous tests are included in this set, in the attempt of exploring the parameter space of reproduction.

The mutation mechanism in this set was different than that in other sets of experiments. Unlike the random-letter-replacement in other runs, in this set of runs the gene letter being mutated was replaced by the argument letter of the mutation gene, in order to give the agents more control of mutation, and the agents apparently used this control to mutate out many of the 'G' gene, which stands for the get-atom action.

Through these experiments, it revealed that as long as there was a 'g' as an argument of any gene, and the mutation mechanism was available in the system for a sufficient period of time, the agents would all end up with a huge number of 'G's in their genomes.

Run 1 was conducted with genomes read from the genebank, instead of with a single ancestor genome. The majority agents had six 'G' genes (60%) in their genomes at the end of the run (see figure 4.17). The fact that these genomes were read from the genebank, therefore there was a large variety of genomes to start with, but ended up, like those in some of the single genome runs, with many get-atom genes in their genomes, is interesting. The letter 'g' not only dominated the genes, but also the arguments of genes, as if agents learned to ensure that they would have many 'g' genes no matter what their starting genome was, and any further mutation would still produce this gene. The agents did have a high divergence at the start of the run, as shown in figure 4.16.

The lengths of the starting genomes varied, therefore it was not possible to see the effect of evolution on this matter.

Run 2 was started with a single, smaller genome:
<kAk qCn fDb xEv fFa uGp gHx oIp hJi fKi wMw kPd xQm eRy gTa wUx>.
The mutation gene ceased to be the dominating one at step 9000, then disappeared at step 11000, at which point the genomes converged into one (see figure 4.18). Like other runs which convergence happened early, the genomes did not have a chance to evolve much from the ancestor genome, other than the loss of the mutation gene.

Run 3 was performed with the same single genome like that in other sets, and the resultant genome was similar to that of run 1, in the sense that it had many get-atom genes (see figure 4.19). It was quite obvious that the agents favoured the 'get-atom' gene. It seemed that the agents had all turned into some atom-seeking monsters. It was unexpected in the case of this set, in the sense that the gene being mutated was set to be replaced by the value of the mutation argument, which was 'q', not 'g'. Yet it was no surprise that agents learned to retain their favourite gene, since they were given the chance of controlling the mutation with the argument letter.

In run 4 the mutation mechanism was turned off, in order to see the effect of crossover in the process of evolution. Genomes were constructed by reading from the genebank thus giving enough variations for crossover to work with. The crossover gene disappeared at time 11,000, and the system converged at that point (see figure 4.20). Obviously, in this setting, crossover alone was not enough to keep the agents evolving.

In a similar run, both the mutation and copy mechanisms were turned off, and as stated in 4.1.2, one reproduction gene was not enough for the population to survive the starting stage of runs, and agents ended up extinct at step 20,000. The extinction happened after the population size reached the threshold of 2500, at which point the system would no longer help to create any new agents, and the agents were left on their own to keep their population size stable.

The copy mechanism was turned off in run 5, and apparently agents performed more crossovers than other runs, since they had less choice of reproduction (no lazy alternative). The increased number of crossovers might also have been an attempt of preserving the genomes on the agents' part, since it worked just as the copy mechanism for the agents with the same genome.

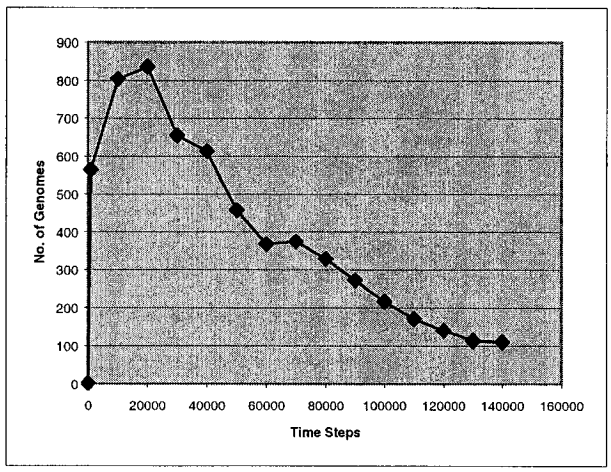

Figure 4.1 6: No. of Genomes in Set 2, Run 1

40% <uCg uGg gGg rGg gGg zMy gGg gQg ySo gGg>

13% <uCg uGg gGg rGg gGg zMy gGg gGg ySo gGg>

Figure 4.17: Genome results from Set 2, Run 1

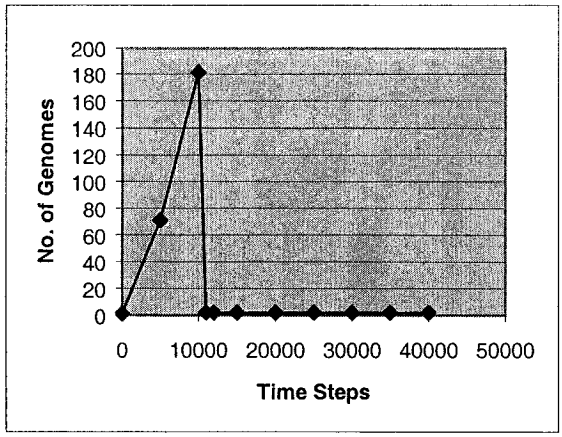

Figure 4.18: No. of Genomes in Set 2, Run 2

13% <gGg gGg vCo gGg gGg gGg gGg gGg gGg gQg gGg gGg yMo gGg gGg xGg zGg wSt tGg gGg>

12% <gGg gGg vCg gGg gGg gGg gGg gGg gGg gQg gGg gGg yMo gGg gGg xGg zGg wSt tGg gGg>

10% <aGg gGg vCo gGg gGg gGg gGg gGg gGg gQg gGg gGg yMo gGg gGg xGg zGg wSt tGg gGg>

4% <gGg gGg vCo gGg gGg gGg gGg gGg gGg gQg gGg gGg yMo gGg gGg xGg zGg wSt gGg gGg>

Figure 4.19: Genome results from Set 2, Run 3

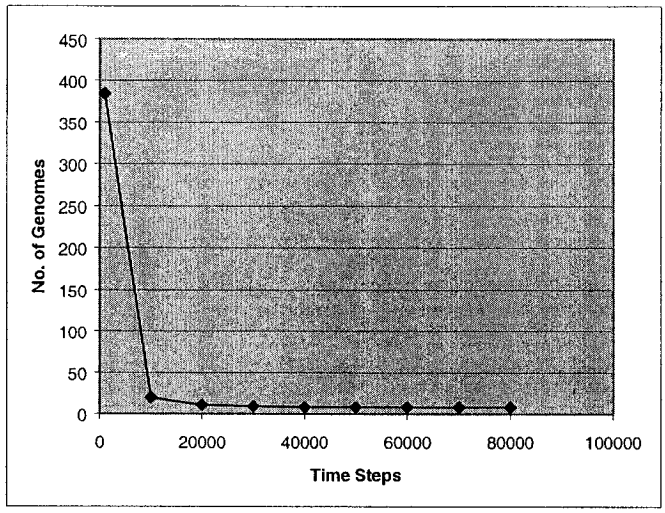

Figure 4.20: No. of Genomes in Set 2, Run 4

4.2.3 Experiment Set 3--- Runs with Fine-Tuned Reproduction Strategy

This set of runs was a continuance of the standard runs in set 1. The frequency of reproduction methods was fine-tuned in these runs, towards limiting the activation and duplication of the copy gene, for the purpose of avoiding the loss of the mutation gene and early convergence. It was revealed through numerous earlier experiments that setting the copy gene at a low frequency was not enough to guard off early convergence, since agents always learned to combine the copy gene with high probability letter through mutation, and add new copy genes on top of the existing one. In this set of tests, certain restraint was applied when a genome was being mutated. The genome was prohibited from mutating any gene into the copy gene, and any addition of the copy gene was also prohibited. The reason for this constraint was to not give agents the chance of getting hold of many copy genes and being able to reproduce only by cloning, which eliminated any possibility for open-ended evolution.

In one of the three runs conducted, the mutation gene was retained throughout the run and the system did not fully converge to one genome (see figure 4.23).

In run 1 and 3, however, the mutation gene was lost and agents reproduced only by crossover, and this led to a full convergence as well (see figure 4.21 and 4.25). Since this happened at a fairly early stage, there were not enough variations in the genome for crossover to have any mutating effect and the agents did not have time to evolve and shed off redundant genes, e.g. gene 'O' and 'P' (see figure 4.22 and 4.26).

This set of experiments revealed that imposing restraint on the copy gene alone might not be enough for ensuring open-ended evolution. It certainly helped agents to retain the mutation gene, but is not a 100% guarantee. Since simulated agents did not favour mutation as a reproduction method, they could always lose it during the process of reproduction, therefore man-made mechanism might have to be added into the system in order to ensure a small probability of mutation through out the runs. This led to the 4^{th} set of experiments, explained in section 4.2.4.

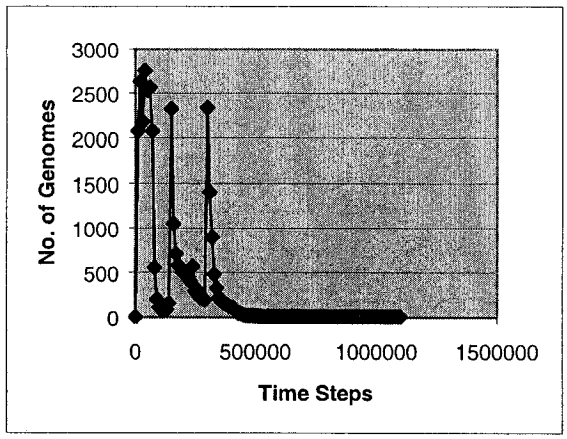

Figure 4.21: No. of Genomes in Set 3, Run 1

85% <xCk bKc wGf vIm qIv rGo yMt xMp hUw uQy vOv wEf tRs wSs yMc mSx yMt vLg yMs xSx wGf yMs wJd wGn>

14% <xCk bKc wGf vIm qIv rGo yMt xMp hUw uQy vOv wEf tRs wSx yMc mSx yMt vLg yMs xSx wGf yMs wJd wGn>

Figure 4.22: Genome results from Set 3, Run 1

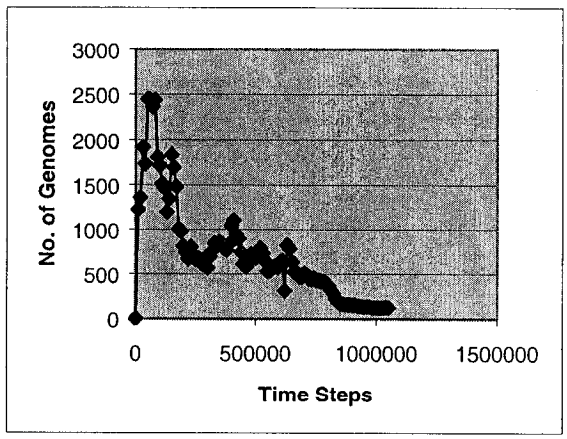

Figure 4.23: No. of Genomes in Set 3, Run 2

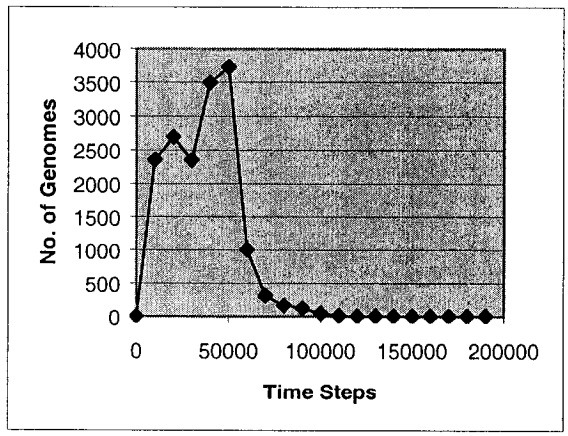

Figure 4.25: No. of Genomes in Set 3, Run 3

87% <mIg wCd wEc xRu eKi qHt rQy dEp wIw qKr mCo yUx gJn xGf yIr xGf qKn xGf nMf xJf yMr yUx mWg xGf xGl>

Figure 4.24: Genome results from Set 3, Run 2

100% <uHl vIg vCp xJl xGf aHv xMi yMo jPe qQq fCd wSu tCt jUs vLp uTn xMl yMo wSx yMo wSr yMo kEw iCt wSx xGf>

Figure 4.26: Genome results from Set 3, Run 3

4.2.4 Experiment Set 4---Runs with Mutation Adjusted

In previous experiments, mutation was implemented as a gene, and no matter how low its frequency was set at, the system still had a huge number of mutations, which did not resemble the mechanism in nature. At the mean time, by being a gene, the mutation mechanism was always at a risk of being lost in the process of reproduction, thus leading to a halt in the process of evolution.

In this set of tests, the above problems were avoided by imposing mutation onto other reproduction methods with a small probability (0.3%). So as long as agents were reproducing, their genomes always had a small chance of being mutated. Six runs were conducted with this configuration, to confirm the viability of this solution. Again these runs differed only by the seed for the RNG.

Results from these tests revealed that previous analysis was correct. As long as there was the mutation mechanism in the gene pool, the genomes would not converge, and would retain the ability of evolving (see figure 4.27, 4.31 and 4.35).

The lengths of genomes also evolved into much shorter ones, leaving only 5 genes (see Figure 4.30) plus 2 variations, out of which 4 were collecting atom genes and thus allowing agents to be able to reproduce efficiently with only one reproduction gene. Once the majority genomes reached the 7-gene length, they did not change for the next 500,000 steps (see figure 4.28, 4.29, 4.33 and 4.36), indicating that these 7 genes were vital for the survival of agents.

All three runs produced similar results, with the same genes left (see figure 4.30, 4.34, 4.37).

In run 2, the resultant genomes resembled the characteristics of a gene pool, in which four distinct genomes were dominating. However, this phenomenon might not qualify for speciation because they possessed basically the same genes, only differed in their gene-arguments. Similar results came out from run 3, which had only 6 genes left in the majority genome (with 5 distinct genes, as usual).

Run 4 and 5 were started with a different genome (with different seeds for the RNG), which had the frequencies of preferred genes set to low, and ended up having the same

result. It indicated that the agents learned to optimize their genome into the same one no matter what they were given to start with.

Run 6 was conducted with a modified genome, which lowered the frequency of the regulate-mass gene. Since in this system the agents were created without any atoms, and their regulate-mass gene was not activated enough for them to get enough atoms from the environment in this run, very few of them had enough energy to reproduce, therefore no sustained change in their genomes could be recorded. A low size of about 400 was kept by the system, since it was below the threshold of minimum population of 2500. The population started reproducing if the frequency of either the regulate-mass gene or get-atom gene was set higher. However, if only the get-atom gene was activated, they would reproduce at the beginning, pass the min population threshold, then would seem to stop obtaining atoms from the environment and thus stop reproducing, end up extinct at step 2000. This result indicated that the regulate-mass gene was very important for the well being of the agents. It allowed the agents to acquire atoms whenever needed, in order to replicate and preserve the population.

Another interesting point noticed was that if the frequency of the aging gene was set too high, the agents would not live long enough to reproduce, and a population size of about only 300 was kept by the system, similar to that of run 6. No doubt this was the reason that the aging gene was not selected in the process of evolution.

Results from this test set were satisfactory. They proved that open-ended evolution was possible in a fairly simple system and that a slight probability of mutation was vital in the process of evolution. From these test results, it was obvious that the current implementation was a sound solution for keeping simulated agents from converging, with just the right amount of mutation, which allowed them to adapt to their environment, and at the same time retained the vital part of their ancestor genome.

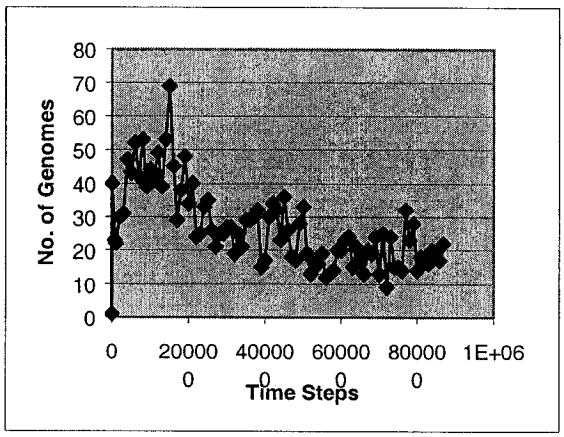

Figure 4.27: No. of Genomes in Set 4, Run 1

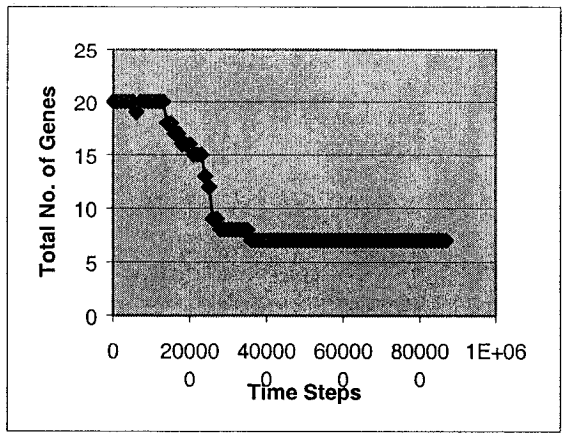

Figure 4.28: No. of Genes in Set 4, Run 1

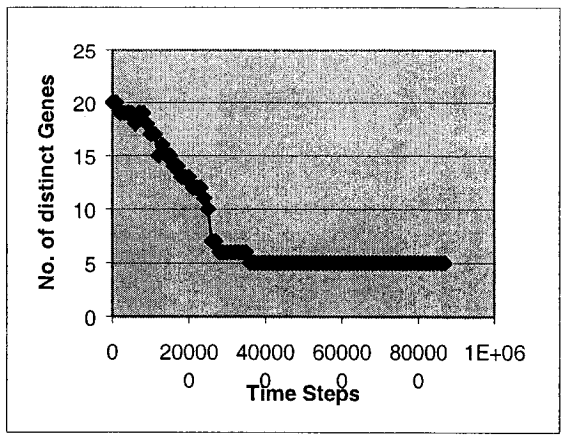

Figure 4.29: No. of Distinct Genes in Set 4, Run 1

96% <lDb vCo xGf xGg yMx xSx xMs>

1% <lDb vCg xGf xGg yMx xSx xMs>

1% <lDb vCo xGe xGg yMx xSx xMs>

Figure 4.30: Genome results from Set 4, Run 1

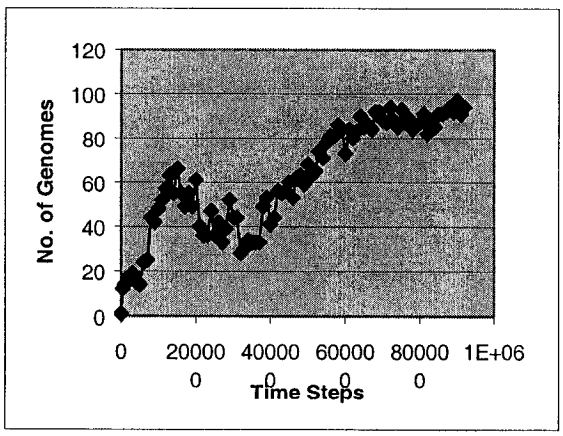

Figure 4.31: No. of Genomes in Set 4, Run 2

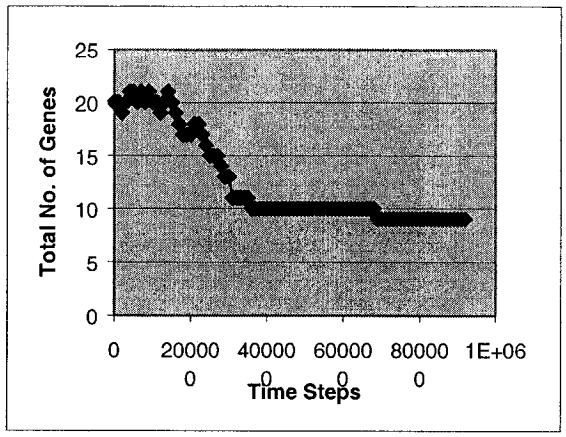

Figure 4.32: No. of Genes in Set 4, Run 2

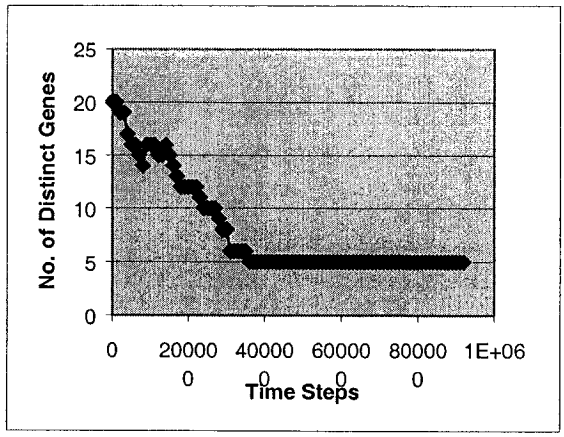

Figure 4.33: No. of Distinct Genes in Set 4, Run 2

42% <mQb vCl xGf yMt ySz xGe yMu xGg zMy>

26% <mQb vCm xGf yMt ySz xGe yMu xGg zMy>

17% <mQb xCm xGf yMt ySz xGe yMu xGg zMy>

8% <mQb vCm xGf yMt ySz xGe yMu xGg zMo>

Figure 4.34: Genome results from Set 4, Run 2

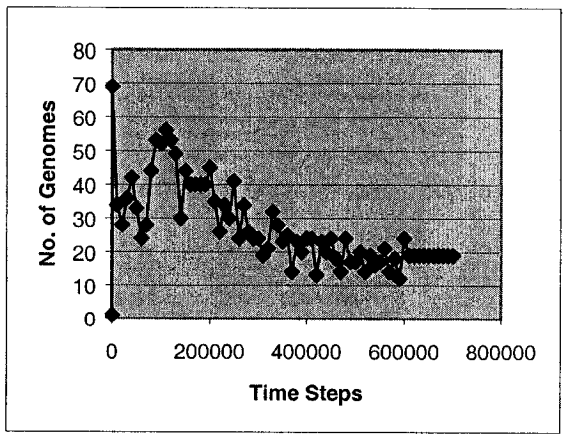

Figure 4.35: No. of Genomes in Set 4, Run 3

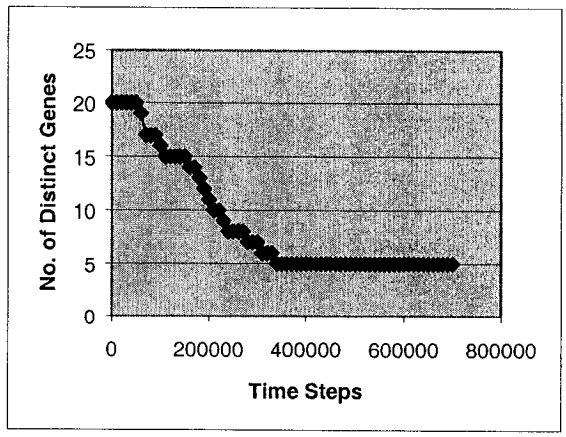

Figure 4.36: No. of Distinct Genes in Set 4, Run 3

85% <vCo xGf yMw xMt lQb xSy>

11% <vCo xGf yMw xMr lQb xSy>

Figure 4.37: Genome results from Set 4, Run 3

4.3 Summary and Discussion

To end this chapter, the more significant results from these experiments are summarized and further discussed in this section. Other than reporting the results from experiments performed, tests not yet have been done should also be considered.

The most interesting and significant result that came from these experiments could be the discovery that simulated agents in a fairly simple environment tended to lose the mutation mechanism, if given a choice, thus ending the process of evolution (which was not preventable with the crossover gene alone, since this gene was less likely to be activated and generally would not bring enough variations to the genome). The best solution discovered so far was to impose the mutation mechanism, with a small probability, onto the reproduction mechanism. This might just as well prove to be an ideal implementation with an environment of arbitrary complexity. In other words, the current mechanism should provide agents with enough adaptive capability to evolve in any environment.

Clearly, mutation played a crucial part in the process of evolution. "Mutation is a vital process from the evolutionary point of view, as it provides a continual source of genetic novelty for selection to work upon." [18]. However, from tests presented in this chapter, it appeared that simulated agents regarded mutation as a lethal mechanism, since too much of it could cause the agents to lose the vital part of their genome. That might be the reason behind the results from earlier experiments that the higher the mutation probability was, the faster the mutation gene was lost. In their attempt of retaining the vital part of their genomes, the agents forsook their chance of reaching the maximum optimization (adaptation to the environment), by choosing to copy themselves faithfully.

With mutation imposed on the agents with a low probability, in set 4, the system achieved maximum optimization, which was expected. For energy efficiency, the agents got rid of all redundant genes. The resultant genomes from the experiments in this set indicated that the only genes necessary for survival in the current environment were the ones for reproduction, for sensing the highest concentration of atoms in the environment and moving toward it, for getting atoms from the environment, for regulating mass (the result of run 6 in set 4 clearly indicated the importance of this gene) and for converting these atoms into energy. All of the genes left were combined with the highest frequency letter (except the reproduction gene, which obviously was activated enough for the survival of the population with medium frequency). In a minimal form, the optimized genomes focused on supporting the agents to replicate, as shown in set 4, in the most energy-efficient manner, without having to waste any on processing useless genes. With this configuration, the majority agents arrived at an optimized form, but with the

mutation mechanism built in, they would always be open for changes should the need come up.

Many runs were conducted with this configuration, in order to prove the viability of the above solution, and the results were satisfactory. The majority agents always ended up with the same optimized genes, regardless of the seed for RNG, or the starting genome. There were always a small number of agents having different genomes, indicating the readiness of the population to change and to adapt.

Another interesting feature displayed in the results of these tests was the fact that the genomes got much shorter compared to the ancestor genome. Tests should be conducted for discovering the reasons behind the loss of genes, why they were not useful for the agents, e.g. the loss of the crossover gene in almost all of the runs indicated that the agents did not favour sexual reproduction, and as a result, the sense-pheromone gene was also lost.

The addition of the get-atom gene in all the resultant genomes was clearly an evidence of evolution. This optimization allowed the population to survive with only one reproduction gene. This result was similar to that of Cosmos, by Tim Taylor, in which the et_collect instruction was repeatedly added to the ancestor program [18]. However, unlike the Cosmos model, where programs tended to get longer and accumulated many junk genes [18], the agents in FormAL evolved their genomes into much shorter ones, and got rid of any redundant genes. The probable reason behind this could be due to the

fact that processing genes consumes energy, and consequently there is evolutionary pressure to get rid of useless genes and to preserve energy. Presumably, agents that consume less energy are more likely to survive than "greedy" agents because the latter are more likely to starve. Thus the lack of junk genes strongly suggests that there are more selection pressure in FormAL than that of the Cosmos model, thus the organisms have to evolve into a more economic form, and that the simulation is working as intended – life is not *too* easy if you're an agent.

The same mutation mechanism would certainly lead to different results in genomes once the environment changes. For example, the additions of the get-atom gene most likely would not have appeared if reproduction were not implemented as an energy-consuming activity. Similarly, if other activities were necessary for the agents to reproduce, then those genes surely would have sustained through evolution. There is still a vast parameter space for the complexity of the environment, waiting to be explored, and certainly will produce very different results in the genomes. In the next chapter, some attempts toward making the environment more complex are presented and discussed.

Chapter 5

More Experiments with Enriched Environments for FormAL

For the purpose of discovering whether agents can learn to evolve as the environment changes, with the reproduction strategy presented in chapter 4, another group of tests were conducted and presented in this short chapter. In these tests, the environment was changed toward complexity step by step, with tests run for each step in order to see each of its effects on the resultant genomes. The reproduction mechanism was kept unchanged from that of section 4.2.4, and so was the ancestor genome. The random number generator was seeded by clock, and the simulations were all conducted for 1,000,000 time steps.

5.1 Run 1

In run 1, the atom distribution mechanism at the initialization period was modified. Randomly selected amount of atoms was distributed into cells, instead of uniform distribution. The uneven energy supply mechanism was intended to drive agents to move and look for cells with more atoms. From the genomes at the end of the test (see figure 5.1), we can see that there are two sense-atom ('S') genes, compared to only one of this gene in previous runs, which could indicate the necessity for agents to have this gene in order to survive.

Freq Perc Gene

3027 91% <vCd xGg wMq wGg yMv lQc wSt xSx>

230 6% <vCd xGe wMq wGg yMv lQc wSt xSx>

9 0% <vCs xGe wMq wGg yMv lQc wSt xSx>

Figure 5.1: Genome results from Run 1

5.2 Run 2, 3 and 4

In run 2, 3 and 4, a killing gene was introduced. Agents were allowed to kill each other, and obtain the victim's atom collection. In the tests conducted in chapter 4, killing was not implemented, and agents could only grab other agent's parts and attach them to its own body. This mechanism apparently did not demonstrate any evolutionary advantage for agents, as the eating gene did not sustain in previous runs (agents learned to lose it at around step 70,000). Since killing is a very important mechanism in co-evolutional systems, it is a crucial factor to be added to the environment, with the aim of finding its effect on the evolving agents. The atoms of the agent got eaten were transferred into the bag of the eating agent, making it an incentive for agents to kill. With the new killing mechanism, activated by the killing gene (represented by the letter 'H'), a prey-predator relation was formed. Agents were motivated to kill others since this was a source of energy supply. The test results supported my expectation and the killing gene did sustain in all the tests (see figure 5.4, 5.5 and 5.6). Graphs from these tests showed that agents formed some clusters, and left the rest of the space empty (see figure 5.2 and 5.3), which could be the result of the unevenness in energy distribution. The resultant genomes also

revealed that the sense-atom gene was not needed any more, probably due to the fact that agents could obtain atoms by killing other agents, instead of getting them from the environment.

Agents send a signal to its cell's bag before killing another agent, and the signal is removed once the killing is over, which is inspired by John Hancock's remark:" The priorities for any signal are that it should be generated quickly, relay a specific message efficiently and then be removed when no longer needed. For these reasons, most biological signals are relatively small chemicals that are able to be moved efficiently, or can diffuse rapidly, to their site of action."[7].

In run 3 each agent had a gene for detecting this hostile signal from its environment, represented by the letter'L'. This gene allows agents to detect a cell with less hostile signals, and move toward it. Like the killing gene, this one did survive 1,000,000 time steps of evolution as well (see figure 5.5).

I attempted to add another gene for agents to detect hostile agents, and allow them to decide on their own whether to flee or to kill the hostile agent instead, based on its own energy level, which was intended to be a metaphor to the scenario that the stronger eat the weaker. Results from these tests were similar to those of run 2, 3, and 4, and the new gene was lost. So apparently it was another useless gene.

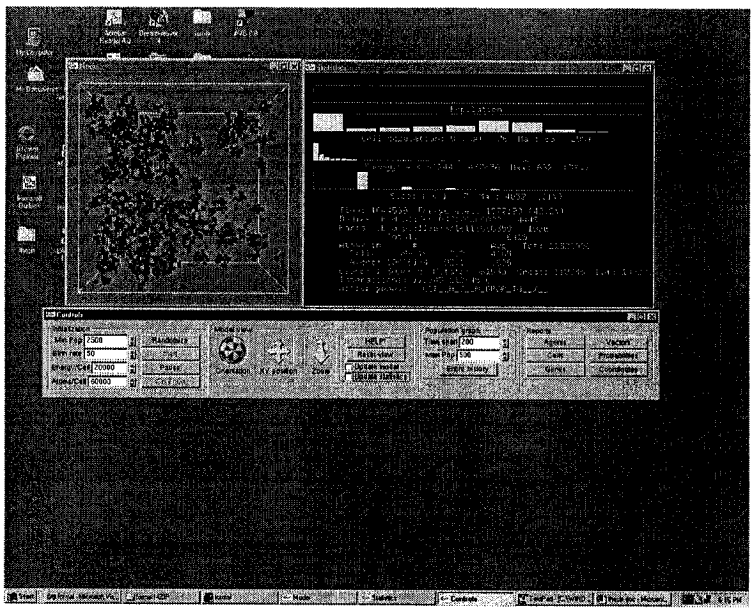

Figure 5.2: Graphic Window in Run 2

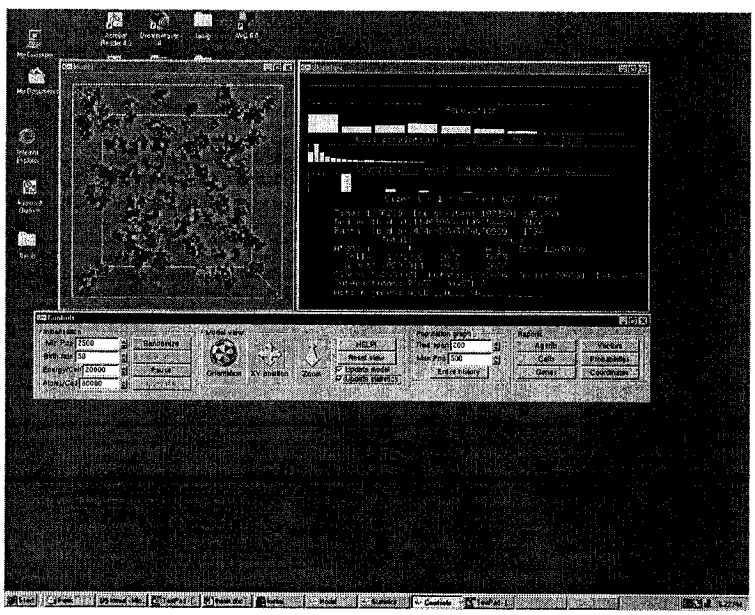

Figure 5.3: Graphic Window in Run 3

Freq Perc Gene

9584 71% <vQb xCx xCw xCw xHf wGe wMp xGf xMp xHf xQa xHa xQa>

888 6% <vQb xCx xCw xCw xHf wGe wMf xGf xMp xHf xQa xHa xQa>

792 5% <vQb xCx xCw xCw xHf wGe wMp xGf xMp xHe xQa xHa xQa>

Figure 5.4: Genome results from Run 2

Freq Perc Gene

3619 82% <lDb zCx zCx xCp cLr xHf wMs zMs xQa zMk>

220 5% <qDb zCx zCx xCp cPr xHf wMs zMs xQa zMk>

170 3% <oDb zCx zCx xCp cLr xHf wMs zMs xQa zMk>

Figure 5.5: Genome results from Run 3

5.3 Run 5 and 6

In run 5 and 6, another feature was added to the environment of agents. While obtaining energy by converting atoms, they also accumulated a small amount of poisoning atoms in their bag. Once the number of poisoning atoms reached a threshold, the agent died. A new gene for disposing lethal atoms was introduced and this gene (represented by the letter 'N') sustained 1 million steps as well (see figure 5.6). Before this gene was added, all agents had died before reaching 10,000 time steps.

2699 65% <xCx xCx xCx mDc xHf yCw wMs vNs xMu xDa wMq wMp>

696 16% <xCx xCx xCx mDc xHf yCw wMv vNs xMu xDa wMq wMp>

347 8% <xCx xCx xCx mDc xHf yCw wMs vNs xMu xDa wMq wMl>

260 6% <xCx xCx xCx mDc xHf yCw wMs vNs xMu xDa wMw wMp>

Figure 5.6: Genome results from Run 5

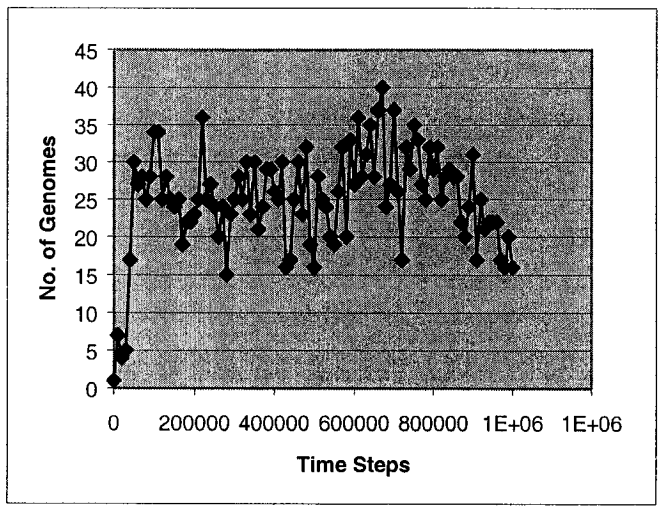

Figure 5.7: No. of Genomes in Run 5

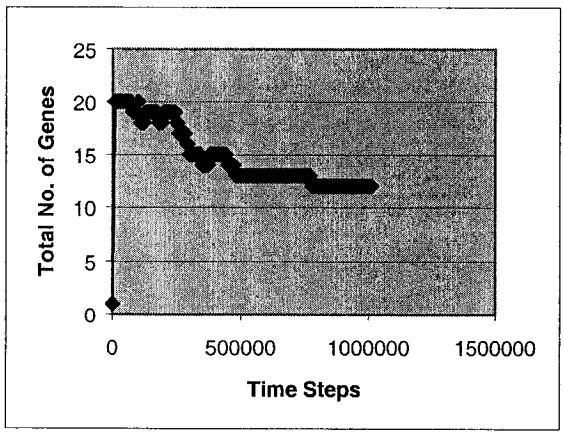

Figure 5.8: No. of Genes in Run 5

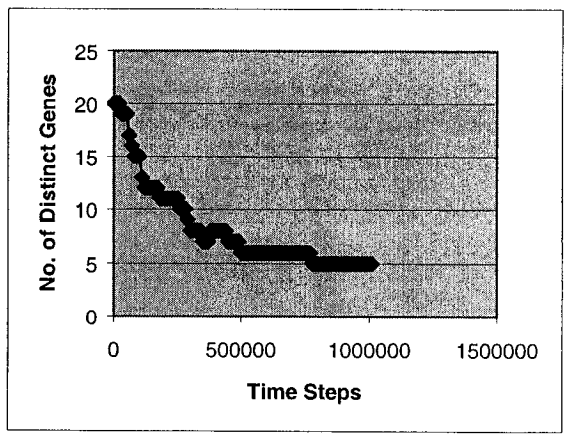

Figure 5.9: No. of Distinct Genes in Run 5

5.4 Conclusion

The additions of the killing, detecting signal and dumping poisoning-atom genes in the resultant genomes in this chapter are a clear indication that the previous speculation in Chapter 4 was valid, that with the current implementation, agents can evolve according to the demand of the surviving conditions of the environment, the population will not converge (see figure 5.7), and that when the environment becomes more complicated, so does the agents' behaviours. The trend of evolution in these runs was the same as those in section 4.2.4 in the sense that genomes became shorter (see figure 5.8 and 5.9) and the majority of them optimized. It took longer for agents to optimize their genomes in these runs, probably due to the more complicated environment.

Chapter 6

Problems Encountered and Their solutions

Some interesting problems were encountered during the development of FormAL. These problems and their solutions were presented in this Chapter.

6.1 Population Explosion

At an earlier stage of the development process, a mysterious problem came up, which resembled 'population explosion'. The program would run smoothly for a while, and then all of a sudden the population would start to explode, and very soon it crashed the memory. The black clusters in the modal window (see figure 5.1) illustrate this phenomenon.

I realized that the problem was caused by mutation. If one of the replication genes was mutated and placed with a high probability letter, all hell broke loose, and this agent would start to reproduce at a disastrous speed, since this agent was also subjected to cloning and every other type of replication. No matter how low the original probability of mutation was set, this was still inevitable at some point, since it took only one agent to cause it to happen. The program could not afford this kind of mutation at that time

because the energy consumption mechanism was not implemented, which meant that there were no restrains on the reproduction activities.

This problem was solved by a tentative algorithm, which only allowed mutation to happen according to the defined genebank. Only the action and argument letter of a gene could be mutated. In another word, when an action letter was being mutated, the program would read from the genebank and locate the probability associated with this gene and set this new action letter to be with this probability; the argument letter could safely be mutated into any letter; and if the probability letter was being mutated, nothing would be done, meaning that mutation would not be executed. If a new action letter were not located in the genebank, it would be allowed to be added as is, which worked as a noise in the genome.

The ultimate and desired solution to this problem came with the implementation of the energy consumption mechanism imposed on the reproduction activities. With the current design, an agent can only reproduce if it possesses enough energy, and any letter of its genome can be mutated. If this agent is reproducing too much, it will not have sufficient energy after a while and thus will have to slow down. The agents who over-produce will die of lack of resource, since reproduction is very resource-costly. If it can obtain energy (get atoms from the environment) to support its reproduction rate, other agents will be deprived of resources and may not be able to reproduce. Since the overall number of atoms in each cell is constant, the population size is kept stable.

6.2 Memory Leak

FormAL had a memory leak problem for a long time. It would run for, say, 40,000 steps, or 150,000 steps, or even 400,000 steps, but would always end up being out of memory and the program would inevitably crash. Much effort was given to the debugging, and finally the leaking point was identified in the program. It was a C++ problem, which allowed dynamic allocation and deletion of memory. When agents died, the memory space they occupied were freed up by deleting its dynamically allocated data member, an object of a class called code; however, hidden inside of this class there was another data member that was created dynamically, and was not deleted, thus the memory it occupied was not freed.

6.3 Parts Overlapping

As mentioned in Chapter 3, parts are also allowed to divide by themselves, and new parts are attached to the old ones as their associated parts. A complicated problem arose from this: If new parts are allowed to grow at any location attached to the original part, how do we make sure the new part does not grow at a space already taken by other parts? I thought on this issue for quite a while, and came up with the idea of letting each cell keep a list of all the spaces already taken, and update the list each time a new part is grown or an agent moved its location. However, this algorithm proved to be too expensive and slowed down the program tremendously. The solution is to allow a new part to grow only

at the direction of the old part (with exceptions of a low probability), e.g. if a part is at the right side of the body, its subsequent parts also grow on the right side of this part. Therefore each part has an attribute of grow-direction, set when the first part is grown from its body. Each subsequent parts of this original part have the same grow-direction.

6.4 Growth in Single Direction

Originally, when a body was set to reproduce (which included cloning, mutation, and crossover), all its parts (6) were checked in order to locate a free space. If a space was found, it would be used to grow a new part; if all the parts spaces were taken, which meant this body already had 6 parts, then a random part, based on the value of argument of this reproduction gene, (arg - 'a')%6, was selected to be promoted into a body, and the newly available space was used to grow the new part. This caused another problem: agents tended to grow in one direction and become very large in size (see Figure 5.2). I rearranged the location selection algorithm so that when an agent is to reproduce, a location is selected randomly. If that location is already occupied, all the parts at that location are promoted into bodies, and the new part will grow at the newly available space. Parts have an added attribute, called layer, and each part can grow new part only if its layer number is smaller than the value of argument. This way, agents have a much smaller chance to grow large since their parts are promoted more frequently, and they have a chance to learn to control their lengths (see Figure 5.3).

6.5 Agents Not Mutating

The original mutation mechanism was to replace the letter in the genome decided by the value of the mutation argument, with the letter of the argument. This caused a problem: when the simulation was started with a single genome, (that is, all agents had the same genome), they all ended up with the same single genome after many long runs without any variation, despite that millions of mutations were recorded. Since the mutation argument was constant, therefore the mutation point was constant, and this clearly did not give the agents enough leeway to evolve. This problem was solved by randomly selecting the mutating letter from the genome.

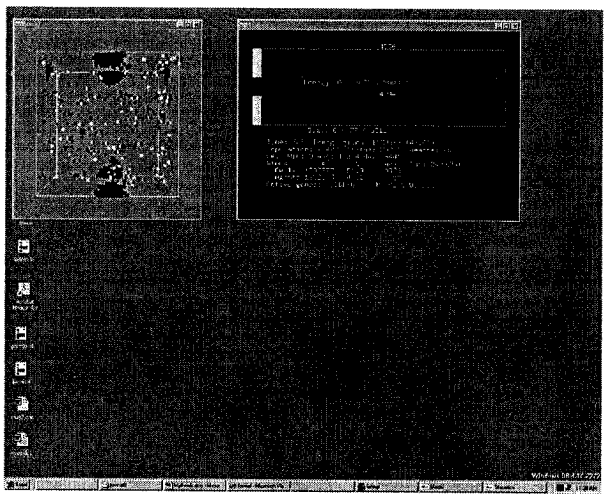

Figure 6.1: Population Explosion

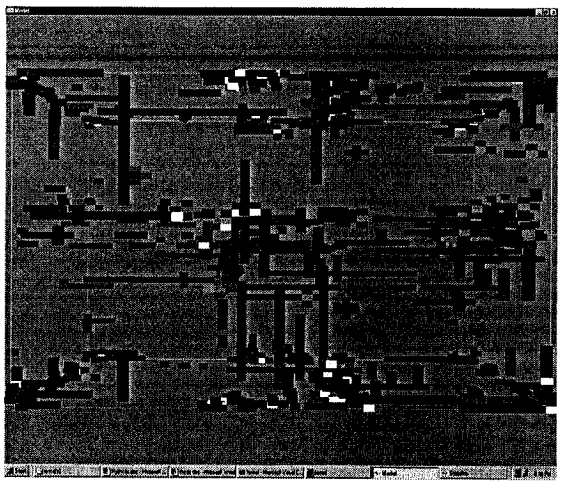

Figure 6.2: Single Direction Growth at Step 45,000

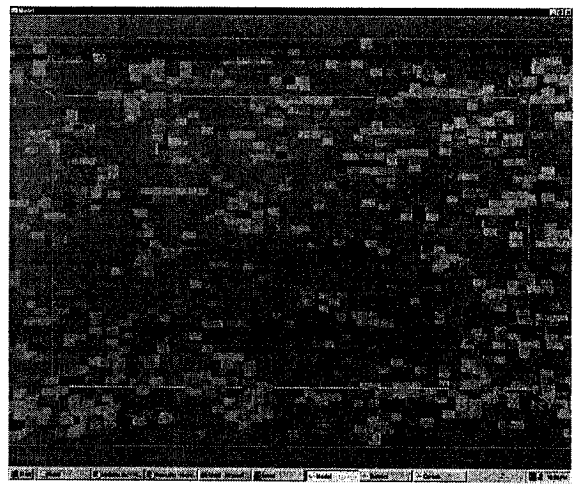

Figure 6.3: Growth after Modification at Step 45,000

Chapter 7

Conclusions and Further Work

Inspired by the origin and evolution of life, Artificial Life today encompasses a broad range of research fields, as mentioned in Chapter 2, and remains one of the fastest developing areas in computer science. Although whether it is possible to simulate life through a computer model remains an open question, it is widely accepted that Artificial Life models do provide a vital tool not available before, for simulating and studying certain aspects of life. For the purpose of studying and understanding life through simulated agents, a well-designed platform is required. FormAL was designed and implemented for this purpose, and its structures are presented and discussed in Chapter 3.

One of the most important issues in Artificial Life is the evolution of genetic characteristics of agents, usually through reproduction. The whole Chapter 4 and 5 is devoted to presenting numerous experiments, conducted with FormAL, for the purpose of tracing signs of evolution through adjustments with reproduction mechanisms and environment. The results of these experiments clearly indicate the important role that the mutation mechanism plays in the course of evolution, at least in simulated agents. The moment this mechanism is lost, the system stops evolving and evolution reaches a dead-end. The probability of mutation is crucial in the sense that if it is too high, it would not allow the organisms to maintain the vital part of their genomes, and a rate too low would take an unrealistically long time for the organisms to evolve. Early convergence is a

common problem encountered by Artificial Life models, and through my experiments, it showed that agents learned to get rid of the mutation gene if they had a choice, and as mentioned above, they converged quickly, thus ending the process of evolution. It was also discovered that reproduction by crossover did not prevent early convergence, since it needed a rich gene pool to be able to have an effect on the resultant genomes. It was obvious that the agents preferred to reproduce faithfully, unless more complicated environment was implemented. I discovered that even in a fairly simple environment, premature convergence could be avoided, by imposing the mutation mechanism onto the agents. With current configuration, the agents developed to a semi-convergence state, with the majorities having an optimized genome. However, with the probability of mutation, thus the probability of change, always present in the reproduction mechanism, the organisms are always ready to adapt to a new environment should the need come up. Chapter 5 is an extension of chapter 4, in which the conclusion in chapter 4 is further tested, and proved valid. With the mutation mechanism always available, agents did learn to adapt to the change and demand of their surviving conditions.

FormAL is work in progress. More complexity shall be added to the system in the future (they are out of the scope of this thesis), making it a more heterogeneous environment for the agents, in the hope of encouraging the evolution of diversity and complexity in the competing agents, obtaining a gene pool so that agents could survive drastic changes in environment conditions, and giving rise to the emergence of speciation and coevolution (host-parasite, prey-predator).

It is widely accepted that life is based on the ability to accept, process, and act on information. I believe that more complex interactions (signalling) among agents and with the environment are needed in order to give rise to emergent properties (evolution). I would like to see that not only the parameters of genes, but genes themselves can evolve, meaning that new, undefined genes could emerge and perform tasks not hand-coded previously.

More features analogous to those in nature might be accommodated in future development of FormAL, as life itself remains the ultimate inspiration for Artificial Life.

References

[1] C. Adami and C.T. Brown. *Evolutionary learning in the 2D artificial life system **avida.*** In R.Brooks and P. Maes, editors, Artificial Life *IV*, pages 377-381. MIT Press, 1994.

[2] R. Axtell, R. Axelrod, J. Epstein, and M Cohen. *Aligning Simulation Models: A Case Study and Results.* July 21, 1995.

[3] Eric W. Bonabeau and G. Theraulaz. *Why do we need artificial life?* Artificial Life, 1(3): 303-325, 1994.

[4] N. Gracias, H. Pereira, J. Lima, A. Rosa. *Gaia: An artificial life environment for ecological systems simulation.* Artificial Life V, pp. 124 134 1997 MIT press.

[5] Peter Grogono, GuoRong Chen, JunFeng Song, Tao Yang, Lei Zhao. *Laws and Life.* In Proceedings of the 7th IASTED Conference on Artificial Intelligence and Soft Computing (ASC 2003), pages 158—163. 2003.

[6] H. Gutowitz. *Artificial-Life Simulators and Their Applications.* 1995. http://www.santafe.edu/~hag/biblio.html.

[7] John T. Hancock. *The Principles of Cell Signalling.* In Sanjeev Kumar and Peter Bentley, editors. On Growth, Form and Computers. Elsevier Academic Press, 2003.

[8] D. Hiebeler. *The Swarm Simulation System and Individual-based Modeling.* 1994. http://citeseer.nj.nec.com/hiebeler94swarm.html.

[9] John H.Holland. *Hidden Order: How Adaptation Builds Complexity.* Addison-Wesley, 1995.

[10] Christopher G. Langton. *Preface.* In Christopher G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors. *Artificial Life II*, volume X of *Santa Fe Institute: Studies in the Sciences of Complexity.* Addison-Wesley, 1992.

[11] F. Menezer and R. K. Belew. *Latent Energy Environments.* http://www.informatics.indiana.edu/fil/Papers/pinep.ps.

[12] Melanie Mitchell. ed 1996. *An Introduction to Genetic Algorithms.* The MIT Press.

[13] Thomas S. Ray. *An approach to the synthesis of life.* In C.G. Langton, C.Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, Pages 371—408. Addison-Wesley, 1992.

[14] I. Peterson. *The Gods of Sugarscape, Digital sex, migration, trade, and war on the social science frontier.* ScienceNewsOnline, November 23, 1996.

[15] Moshe Sipper. *An Introduction to artificial life.* 1995. http://www.it.uom.gr/pdp/DigitalLib/ALife/Al_lect.htm, http://www.cs.unibo.it/babaoglu/courses/cas/tutorials/Artificial_Life.pdf.

[16] R. Smith and M. Bedau. *Emergence of Complex Ecologies in ECHO.* Proceedings of the International Conference on Complex Systems, 1997.

[17] Charles E. Taylor. *"Fleshing Out" Artificial life II.* In Christopher G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors. *Artificial Life II,* volume X of *Santa Fe Institute: Studies in the Sciences of Complexity.* Addison Wesley, 1992. Pp. 25 34.

[18] Timothy John Taylor. *From Artificial Evolution To Artificial Life.* PhD Thesis. May 1999. http://www.dai.ed.ac.uk/homes/timt/papers/thesis/.

[19] Timothy John Taylor. Creativity in Evolution: Individuals, Interactions, and Environments. In P. Bentley and D. Corne, editors, *Creative Evolutionary Systems,* pages 79-108. Academic Press, 2002.

[20] Timothy John Taylor and John Hallam. *Studying Evolution With Self-Replicating Computer Programs.* Fourth European Conference on Artificial Life, 1997.

[21] John von Neumann. *The Theory of Self-Reproducing Automata.* Edited and completed by Arthur Burks. University of Illinois Press, Urbana, 1966.

[22] Kai Wu. *Artificial Life.* Cornell's SciTech Magazine. 1994. http://www.rso.cornell.edu/scitech/archive/94fal/alife.html.

[23] L. Yeager. *Computational Genetics, Physiology, Metabolism, Neural Systems, Learning, Vision, and Behavior or PolyWorld: Life in a New Context.* 1994. http://www.beanblossom.in.us/larryy/polyworld.html.

Appendix

My Work in FormAL

FormAL was developed by a group of people, led by Prof. Peter Grogono who established a base framework for the program to run, and others worked on different aspects of the project, on top of the base program.

My contribution in FormAL can be divided into four parts:

1. Participated in the design phase of the project
2. Designed and implemented each and every algorithms related to the following subjects:

   Reproduction strategies, and including the design of offspring as parts

   Sending, detecting and reacting to signals

   Hazardous effects on agents, and agents releasing poisoning atoms

   Agents being killed both as bodies and as parts, and transferring atoms

   Parts dividing

   Separating parts from their parents (promoting parts into bodies)

   Modifying many parts of the program base to fit to my implementation
3. Solved problems in the project (see chapter 6)
4. Performed extensive and long runs of FormAL, and analysed their results

Default Parameter Values For FormAL

Parameter	Value
Number of cells	216
(The number of cells in the universe)	
Atoms/cell	60000
(The number of atoms in a cell at the beginning of the simulation)	
Energy/cell	20000
(The amount of energy added to each cell during each step of the simulation)	
Energy/agent	100000
(The amount of energy given to an agent at start-up)	
Energy/step	4320000
(The amount of energy added to the universe during each step of the simulation)	
Promote_Energy	20000
(The amount of energy given to a newly-promoted part)	
Part_Energy	20000
(Energy needed to grow a new part)	
Prob_A	0.001
(The value of a as a probability in the genome. z gives probability 1)	
Num_Cell_Neighbours	27
(The number of neighbours of a cell)	

Max_Init_Pop	5000
(The largest permitted value for the initial agent population)	
Life_Expectancy	100
(The expected lifespan of a body depends on this constant but is also controlled genetically)	
Gene_Energy	10
(The energy used by processing one gene)	
Energy_Factor	5000
(The desired energy of an agent is obtained by multiplying this factor by the argument of the gene)	
Gene_Rep_Frequency	10000
(Frequency of gene reports)	
Mass_Factor	250
(The desired mass of an agent is obtained by multiplying this factor by the argument of the gene)	
Min_Energy	5000
(An agent with this amount of energy is considered INACTIVE)	
Min_Size	1000
(An agent of this size is considered INACTIVE)	
Part_Energy_Factor	0.25
(The proportion of its energy that a body gives to a new part)	
MinPopulation	2500
(During initialization, agents are created until the population exceeds this value)	

BirthRate	50
(During initialization, this number of agents are created at every step, until the required minimum population is attained)	

Table A.1: Default Parameter Values For FormAL