From Artificial Evolution to Artificial Life

Timothy John Taylor

PhD
University of Edinburgh
1999

To Pete

Abstract

This work addresses the question: What are the basic design considerations for creating a synthetic model of the evolution of living systems (i.e. an 'artificial life' system)? It can also be viewed as an attempt to elucidate the logical structure (in a very general sense) of biological evolution. However, with no adequate definition of life, the experimental portion of the work concentrates on more specific issues, and primarily on the issue of open-ended evolution. An artificial evolutionary system called Cosmos, which provides a virtual operating system capable of simulating the parallel processing and evolution of a population of several thousand self-reproducing computer programs, is introduced. Cosmos is related to Ray's established Tierra system [Ray 91], but there are a number of significant differences. A wide variety of experiments with Cosmos, which were designed to investigate its evolutionary dynamics, are reported. An analysis of the results is presented, with particular attention given to the role of contingency in determining the outcome of the runs. The results of this work, and consideration of the existing literature on artificial evolutionary systems, leads to the conclusion that artificial life models such as this are lacking on a number of theoretical and methodological grounds. It is emphasised that explicit theoretical considerations should guide the design of such models, if they are to be of scientific value. An analysis of various issues relating to self-reproduction, especially in the context of evolution, is presented, including some extensions to von Neumann's analysis of self-reproduction [von Neumann 66]. This suggests ways in which the evolutionary potential of such models might be improved. In particular, a shift of focus is recommended towards a more careful consideration of the phenotypic capabilities of the reproducing individuals. Phenotypic capabilities fundamentally involve interactions with the environment (both abiotic and biotic), and it is further argued that the theoretical grounding upon which these models should be based must include consideration of the kind of environments and the kind of interactions required for open-ended evolution. A number of useful future research directions are identified. Finally, the relevance of such work to the original goal of modelling the evolution of living systems (as opposed to the more general goal of modelling open-ended evolution) is discussed. It is suggested that the study of open-ended evolution can lead us to a better understanding of the essential properties of life, but only if the questions being asked in these studies are phrased appropriately.

Acknowledgements

Many people have contributed in a variety of ways to the production of this thesis over the last three and a half years.

First of all, I would like to thank John Hallam, my supervisor. John took me on as a student, and has always been ready to offer guidance, discussion and constructive criticism of my work. He has also allowed me the freedom to follow my own academic interests in the course of my work, which I greatly appreciate. Thanks John!

The interdisciplinary nature of my work has required me to seek advice and discussion from a wide variety of people. Their generosity in giving up their valuable time to answer my questions and read drafts of some of the following chapters has consistently surprised me (especially those whom I know only by email exchanges).

Chief among my list of 'extra supervisors' is Mark Bedau from Reed College in the USA. Since first meeting Mark in early 1997 at the Philosophy of Artificial Life conference in Oxford, he has provided vital encouragement, suggestions and discussion of my work. From a practical point of view, he also developed a range of analysis and visualisation techniques for evolutionary systems, which I have used extensively. I would also like to thank Emile Snyder, who implemented the analysis software and has helped me to use it. Thanks, Emile, you saved me a lot of time!

Tom Ray read through an early draft of the Cosmos design document, and provided useful comments and suggestions. The work reported in Section 6.1 was originally published in the proceedings of the Sixth International Conference on Artificial Life in 1998, and I would like to thank Chris Adami and the four anonymous reviewers who offered comments on the original paper. Towards the end of my research I have had some invaluable discussions, and comments on drafts of various chapters, from a number of other people. I especially thank Nick Barton, John Holland, Howard Pattee and Moshe Sipper for reading and providing useful comments on Chapter 7 (and Chapter 2 as well in Nick Barton's case). Thanks also to Barry McMullin for providing clarification on some issues, and to Daniel Mange for some interesting discussion.

I have had a great time at a number of conferences over the last few years, and these have been made even more enjoyable by a number of people I've met at them. In particular, thanks to Jason Noble and Alastair Channon—see you at ECAL'99?

During my research I have also benefited from some much broader discussions about artificial life, artificial intelligence, life, the universe and everything. These discussions have often been lubricated with plenty of beer, and fed with copious amounts of chicken korma and egg foo yung. I am indebted to the other members of the Mobile Robots Group in the (former) Department of Artificial Intelligence, for making it such a fun place to work. In particular (at the risk of offending the others), I'd like to say a special thank you to Simon Perkins, Richard Reeve, Arturo Espinosa-Romero, John Demiris, Nuno Chagas, Sandra Gadanho and Heba Al-Lakany. Thanks also to Judith Good for the therapeutic moaning sessions about our theses, and to Bernard Goffin for putting up with our constant droning about them during dinners at Teviot.

There are many other people at the department who have made my life a lot easier. The librarians, Olga Franks and Janice Gailani, have always been more than helpful. To name a few more, Ken Dawson, Karen Konstroffer, Margaret Rennex, David Wyse and

Dougie Howie have also been very helpful in a number of ways. I'd also like to thank the organisers of the EUCS Perl course, for what must be the most useful half-day course I've ever attended.

I would not have been able to conduct this research without the financial assistance of a grant from the Engineering and Physical Sciences Research Council (grant number 95306471), for which I am very grateful.

I don't have enough room to adequately thank my parents for everything they have done, in so many ways, to enable me to be in the position of submitting this thesis. In retrospect, it was a Christmas gift from them some years ago of a paperback book—on the face of it insignificant in comparison to everything else they have done for me—that planted the seed for my growing interest in evolution and artificial life and ultimately led me to write this thesis. The book was *The Blind Watchmaker* by Richard Dawkins. Thanks, Mum and Dad, and thanks also to Anne, Katie, Rich and Tamsin.

Of course, it hasn't all been work over these last three and a half years, and I'd like to thank those who have kept me away from artificial life and helped to make my real life so enjoyable. Apart from various friends already mentioned, thanks to all the (ex-)Strathfillan Road gang—Claudio, Trish, Gordon, Jen, Matt and Wakako. A big thank you to Simon and Ögmundur for all the squash matches and pints in the bar afterwards—I hope we keep in touch. Thanks also to Stu for the regular and very welcome coffee breaks, and to Chris and Alan for really taking my mind off work!

But most of all, there is one person who has been with me through every hour of this time, and without whom I can't imagine having done this. Thank you Pete, for your constant love, affection, friendship and support.

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Tim Taylor
Edinburgh
May 29, 1999

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved."

Charles Darwin, The Origin of Species ([Darwin 59] pp.459–460)

1.1 The Big Picture

This thesis is about life: what it is, how we recognise it, and whether it is possible to recreate it on a computer. The world-view introduced by Darwin asserts that the existence of living organisms can fundamentally be explained in terms of a small number of basic processes [Darwin 59]. Darwin argued that the most important of these are the processes of reproduction, variation and natural selection. Neo-Darwinism, the modern version of Darwin's position, asserts that these are in fact the *only* relevant processes for explaining the adaptations of organisms.

On the face of it, these arguments suggest that it might be possible to build an artificial world that exhibits these simple processes on a computer, such that entities evolve within it which could be regarded as living organisms. Most people would agree that the chances of success of such an endeavour are slim. However, there is less agreement over the reasons why the enterprise might fail. Some would argue that life is a phenomenon which is fundamentally associated with the material world; for example, they might argue that some of the processes associated with living organisms, such as metabolism, could be simulated on a computer, but that a computer program could never be regarded as *really* metabolising. On the other hand, others would argue that life is fundamentally a process (or set of processes) and is quite independent of its particular medium of implementation; they would be quite happy to accept that artificial life could evolve on a computer. However, even some of these people would predict the failure of the endeavour, but for more practical reasons such as the lack of processing speed, memory, etc., of today's computers.

An attempt to create artificial life might therefore seem ill-fated from the start, but I believe it is still worth pursuing for a number of reasons. First, the approach of building a synthetic evolutionary system is very different to the traditional approach taken in theoretical biology of analysing evolution by tracking changes in population-level measures (e.g. gene frequencies) using models based upon differential equations. Different approaches involve looking at systems in different ways; one approach might suggest answers (and, indeed, questions) whose significance is not apparent from another approach. In this way, the synthetic approach can complement the more traditional approaches of theoretical biology, and lead us to ask different sorts of questions about evolution and life. This being the case, the endeavour might be worthwhile pursuing even if the attempt ultimately fails in achieving its grand goal. Secondly, we do not know how much of a barrier the practical limitations imposed by today's computers, mentioned above, really are. These are issues which can be resolved empirically.

The potential benefits of this work therefore include a contribution to our scientific knowledge of biological evolution. Even if we are ultimately unsuccessful in reaching the grand goal of synthesising artificial life, the nature of the ways in which the attempt fails will be instructive. In addition, any success in the endeavour would have tremendous implications in the longer-term for many kinds of computer applications. The ability to evolve an unlimited variety of complex adaptations is of obvious significance to areas such as machine learning systems, evolutionary design, computer games and genetic art, to name but a few.

1.2 Organisation of the Thesis

I will now give an overview of the thesis. I will briefly describe the content of each chapter, and emphasise the flow of thought connecting the chapters together. Where relevant, I highlight the key problems addressed in the chapters, and what I regard as the key contributions of the work.

Relevant issues from the biological literature are discussed in Chapter 2. The distinction between evolutionary and ecological views of life is emphasised, and attempts to produce definitions of life from each of these views, as well as hybrid definitions, are discussed. Similar distinctions in origin-of-life models are described. Current debates concerning various preconceptions about life and evolution are then discussed. These include debates over the notion of progress in evolution, and in particular the common assumption that evolutionary processes necessarily lead to an increase in the complexity of organisms. This leads to a discussion of the term 'complexity', and of the various issues it encompasses. The nature of the major evolutionary transitions that have occurred in the history of life on Earth is described. Current debates in the biological literature over the relative importance of general principles versus contingency ('historical accidents') in determining the course of evolution are also discussed. At the end of the chapter, definitions are provided for the term 'open-ended evolution' and for various sorts of complexity, but an adequately precise and satisfactory definition of life is still lacking. In this light, a decision is made to frame the research questions addressed in the rest of the thesis in terms of more specific concepts. In particular, the experimental work will concentrate on the issue of open-ended evolution.

A review of the relevant artificial life literature is presented in Chapter 3, together with an introduction to the more important issues and debates of the subject. The distinction between weak and strong artificial life is discussed, as are some of the theoretical and practical hurdles that the subject faces. Various views of the relationship between artificial life and theoretical biology are also presented. The majority of the chapter is devoted to a description of previous work with artificial life models. Particular attention is given to studies addressing the issues of self-reproduction and open-ended evolution, including work by John von Neumann, Nils Barricelli, John Holland and Tom Ray. More briefly, a number of models addressing issues of self-organisation and the origin of life are also discussed. Having devoted some time to describing the literature, I end the chapter by talking about the various methodology and design issues for artificial life platforms that have been highlighted by the preceding review.

In Chapter 4 the platform used for the experimental portion of the thesis is introduced. This platform, called Cosmos, is based upon Ray's Tierra system [Ray 91]. Reasons for experimenting with such a system are discussed at the start of the chapter. As the approach pioneered by Ray with Tierra is fairly widely used in the artificial life community, and its validity is fairly widely accepted, I claim that it is a valuable endeavour to investigate such models more thoroughly. Although Cosmos shares the general Tierra approach of modelling evolving individuals as self-reproducing computer programs, there are a number of important differences in the design of the two systems. These include: a closer analogy between individual programs (or more accurately, processes) in Cosmos and biological cells, including features such as regulation of the genome; the introduction of the notion of energy (potential CPU-time) as a commodity that cells must collect, store and use to pay for the execution of instructions; the introduction of a two-dimensional discrete spatial environment in which the programs can move and interact; and finally, the potential for parallel programs (which are considered analogous to multicellular organisms) to evolve. (Tierra has also been modified to allow for parallel programs, e.g. [Thearling & Ray 96], but the implementation details are different.) These and other differences between Cosmos and Tierra are discussed throughout the chapter, and summarised in the final section.

The results of a wide variety of Cosmos runs are presented, analysed and discussed in Chapters 5 and 6. These two chapters together constitute possibly the fullest and most systematic investigation of a Tierra-like system to have been reported.

Chapter 5 is devoted to the detailed analysis of a single Cosmos run. The various measures and visualisation techniques used are first described, including a collection of measures suggested by Mark Bedau and colleagues (e.g. [Bedau *et al.* 98]). The results of the run were not spectacular; programs generally increased in length due to the acquisition of more instructions to collect energy from the environment (which thereby increased their chances of survival). No parallel programs emerged, neither were any parasites, or similar ecological phenomena, observed. Reasons for this behaviour are discussed. The lack of emergence of ecological phenomena reinforces the suspicion that their appearance in runs of Tierra is due to some fairly specific details of the system's design rather than to any more general principles. The phylogenetic tree of the significant types of program to have emerged during the run is reconstructed, and suggests that chance events may have played a significant role in determining the outcome of the run. This issue is investigated in more detail in the following chapter.

Chapter 6 describes the results of a wide variety of Cosmos runs, designed to explore the system's parameter space. This issue of contingency is investigated in detail, in a group of 19 experiments in which the system is run under identical conditions except for the number used as a seed for the random number generator. It is found that each run performed significantly differently, in a number of measures, to at least a third of the other runs. This result is used in a calculation which suggests that the system should be run at least nine times in future experiments to be sure of observing a variety of results due purely to contingency rather than any other factors. A range of other experiments are also reported. Some of the more interesting results from these include: the runs can generally by categorised into one of two classes proposed by Bedau and colleagues [Bedau *et al.* 98], where one class represents active evolutionary behaviour and the other represents a degenerative state where the system's evolutionary potential is effectively lost; and, in one set of experiments (reported in Section 6.5.3) 'speciation' is observed, i.e. the coexistence in the population of programs of different lengths, in different areas of the environment. A gradient in the energy distribution to the environment is required to achieve this, although no physical boundaries are necessary. Some experiments with a sexually reproducing program are also described, although asexual varieties quickly invaded the population and displaced the sexual programs in all of the reported runs. A summary and discussion of all of the results is presented at the end of the chapter.

In Chapter 7, with the benefit of the practical experience gained by designing, implementing and using Cosmos, I step back and reconsider the scientific value of the approach. A number of areas are identified in which Cosmos and similar Tierra-like systems are lacking, from both theoretical and methodological points of view. The major problem with using these systems for scientific purposes is the inadequacy of the theoretical grounding upon which their design is based. Various other problems are discussed, including the fact that these systems still do not seem to have the ability to evolve *fundamentally new* types of innovations. These discussions lead on to a detailed analysis of various issues involved in the concept of self-reproduction, particularly within an evolutionary context. The reproduction process employed by the programs in these systems is analysed in terms of von Neumann's genetic self-reproduction architecture [von Neumann 66], and compared to the replication processes employed by von Neumann's self-reproducing automata, by DNA, and by a hypothetical 'proto-DNA' structure which has properties making it suitable for acting as a seed for an open-ended evolutionary process. It is argued that the explicit encoding of the self-reproduction process on programs in Tierra-like systems is a liability in terms of open-ended evolution, and that these systems would have greater evolutionary potential if the copying process was, initially at least, implicitly encoded in the operating system. In this situation, the focus of the design of such systems then shifts to the consideration of the sorts of phenotypic properties the reproducers should possess. The domain of interaction of the phenotypes should be within the evolving system itself, to allow for intrinsic rather than extrinsic selection (e.g. [Packard 88]), but beyond this there would seem to be few restrictions. Some ways in which the study of open-ended evolution in artificial life models may be improved are discussed in Section 7.3, where a number of open questions are also listed. A paradigm for studying biological evolution, introduced 30 years ago by C.H. Waddington [Waddington 69], is described, and it is argued that this represents a useful starting place for an improved synthetic approach. As an example of a future research direction, I show how the important question of how organisms evolve fundamentally new measuring instruments can be addressed within Waddington's framework (see Section 7.3.2). In the final section of Chapter 7 I return to the issue of studying life as opposed to open-ended evolution. I suggest that artificial life models of open-ended evolution and of other processes associated with the biosphere can usefully contribute to a broader understanding of the nature of life, but only if the questions being addressed are expressed in terms of precise concepts.

A summary of the thesis message is presented in Chapter 8.

Further details of the design and implementation of Cosmos are given in Appendix A, and further details of the runs reported in Chapters 5 and 6 can be found in Appendix B. A glossary of the biological terms used in the thesis is included after these appendices.

1.3 Major Contributions

In the following list I summarise what I regard as the major contributions of this work.

1. The design and implementation of Cosmos, a Tierra-like artificial life platform.

2. The description and analysis of a wide variety of experiments with Cosmos, constituting the fullest and most systematic investigation of the behaviour of such a system to have been reported. An important part of this investigation concerns the role of contingency in determining the course of evolution.

3. The identification of a number of problems, some theoretical and some methodological, that limit the utility of Tierra-like platforms as tools for the scientific investigation of open-ended evolution in general, and of the evolution of life in particular.

4. An analysis of the logic of reproduction in terms of more specific issues, with particular emphasis on processes of self-reproduction in the context of evolution. This includes topics such as: the superiority of genetic reproduction over reproduction by self-inspection; discussion of self-reproducing programs in Tierra-like platforms in terms of von Neumann's analysis of the logic of self-reproduction; issues relating to the explicit versus implicit encoding of the various components of von Neumann's genetic architecture; an emphasis on the role of phenotypes; the desirability of allowing unrestricted interactions between individuals to promote open-ended evolution; and discussion of the evolution of symbolic information and of fundamentally new measuring instruments, and how these may be achieved.

5. Suggestions for how the approach to modelling open-ended evolution, and the evolution of life, may be improved. In particular, it is emphasised that careful consideration must be given not only to modelling individuals, but also to modelling the environments in which they exist, and the sorts of interactions allowed. Waddington's paradigm for an evolutionary process [Waddington 69] is suggested as a possible unifying framework within which to develop a better approach to the synthetic modelling of evolution and life.

1.4 Typographical Conventions

Various typographical conventions have been used throughout the thesis, as listed in Table 1.1.

Example	Signification	Described in
`rng_seed`	Name of a Cosmos parameter, parameter value, instruction or file.	Apdx. A
`ExportData`	Name of a routine in Cosmos top-level algorithm.	Fig. 4.6
find_end	Label for a section of code on a predefined Cosmos ancestor program.	Figs. 5.1, 6.30
Translator	Structure within a Cosmos cell, or a class of object in the underlying implementation.	Sct. 4.3
A	Component of von Neumann's genetic architecture.	Sct. 3.2.1
$\mathcal{A}$	Object in Löfgren's definitions of production, reproduction and self-reproduction, or in Waddington's work.	Scts. 7.2.1, 7.3.2
ax	Register in a Cosmos cell.	Sct. 4.3.8

Table 1.1: Typographical Conventions.

Chapter 2

Evolution and Life

"I may look happy, healthy and clean . . .

but behind the smile there is a Zerox machine"

Adam Ant, 'Zerox'[1]

In this chapter I present a review of some of the current viewpoints and debates on evolution, life and their interrelationship. This is by no means an exhaustive review, but serves to place the work that follows in some context. I also use it as a starting point from which to define more carefully the sense in which I will use words such as 'life' and 'open-ended evolution' throughout the rest of this thesis.

2.1 Two Views of Life

As emphasised by the evolutionary biologist John Maynard Smith, there are two distinct ways in which living organisms may be viewed. "One is of a population of entities which, because they possess a hereditary mechanism, will evolve adaptations for survival. The other is of a complex structure which is maintained by the energy flowing through it" [Maynard Smith 86] (p.7). In the following two subsections I will give a quick overview of these two pictures.

[1] From the album *Dirk Wears White Sox* by Adam and the Ants ©1979 Do-It Records.

2.1.1 The Evolutionary View

The first view sees organisms as players in an evolutionary process. This view originated from Charles Darwin's ideas [Darwin 59], and has been gradually refined by advances in knowledge to the current 'neo-Darwinist' position.[2]

Neo-Darwinism

Darwin's theory originated with the observation that organisms are exquisitely adapted to their environments. He further pointed out that organisms have the properties of multiplication, variation and heredity. Essentially, he showed that organisms with these three properties necessarily become adapted to their environment, and that there is therefore "a necessary connection between one set of observations, concerning reproduction, and another set, concerning adaptation" [Maynard Smith 86] (p.6).

Darwin claimed that all living things on Earth are descended, with modification, from one or a small number of simple original forms.[3] He further claimed that the major cause of evolutionary change was natural selection, i.e. those modifications which help the evolving entities to survive and reproduce will be maintained, and those which do not help will not. While Darwin himself was willing to accept that evolutionary changes may also be brought about by other causes (such as the effects of use and disuse), the neo-Darwinist position sees natural selection as the *only* relevant mechanism by which evolution leads to adaptation. The position does not rule out other causes of evolutionary change, but it does contend that any cause other than natural selection will result in essentially random changes.

An extreme neo-Darwinist definition of life might therefore be "any entity with the properties of multiplication, variation and heredity". There are a number of complications to this picture, which, while not contradicting the neo-Darwinist position, at least suggest other things to be taken into account in our efforts to build an artificial evolutionary system. Some of these (self-organisation and development, sex, and

[2] The major extension that produced the neo-Darwinist position was the synthesis of Darwin's original theory with Mendelian genetics.

[3] Darwin did not claim his theory accounted for the *origin* of life. However, more recently, a number of theories have been proposed which put the original transition from chemistry to biology in an evolutionary context (see, for example, [Eigen 71], [Maynard Smith & Szathmáry 95]).

symbiogenesis) are briefly described below, and will be returned to in later chapters.

Self-Organisation and Development

Although some people might accept this extreme neo-Darwinist definition, few would claim that it can give a full explanation of life on Earth. After all, life is embedded in the physical world, and the rules of physics and chemistry must surely impose important constraints on the evolutionary process.

In recent years there have been a growing number of attempts to integrate theories of self-organisation with evolutionary theory. Stuart Kauffman, professor at the Santa Fe Institute, argues that the laws of physics generate general principles of self-organisation which must be considered when studying the evolution of life (although he believes that science does not yet have an adequate conceptual or mathematical framework to deal with such issues) [Kauffman 93]. He says on the subject:

> "The building blocks of life at a variety of levels from molecules to cells to tissues to organisms are precisely the robust, self-organized, and emergent properties of the way the world works. If selection merely molds further the stable properties of its building blocks, the emergent lawful order exhibited by such systems will persist in organisms. The spontaneous order will shine through, whatever selection's further siftings.
>
> Can selection have reached beyond the spontaneous order of its building blocks? Perhaps. But we do not know how far. The more rare and improbable the forms that selection seeks, the less typical and robust they are and the stronger will be the pressure of mutations to revert to what is typical and robust. That natural order, we may suspect, will indeed shine through." [Kauffman 95] (p.189).

Many have argued that the developmental processes of organisms also impose significant constraints upon organismal design (e.g. [Wolpert 93], [Goodwin 94], [Raff 97]). Note that the nature of these constraints is subtly different to the general principles of self-organisation proposed by Kauffman and others; developmental constraints are somewhat arbitrary with respect to underlying physical laws, but they are nevertheless real constraints for organisms employing any kind of developmental process. Maynard Smith says on the issue:

> "For 50 years to my knowledge, and maybe for much longer than that, people have been saying that our ideas about evolution will be transformed when we have an adequate theory of development." [Maynard Smith 96].

The distinguished geneticist C.H. Waddington remarked that:

> "The hereditary differences which arise in animals are not quite random, like the differences between two heaps of bricks. They result from changes in orderly systems of development, and each new variety has an order of its own, maybe less, but sometimes more, complex than that of the original form from which it was developed." [Waddington 57] (p.7).

Developmental biologist Lewis Wolpert argues that the self-organisational and feedback mechanisms operating during the development of an embryo to an adult organism radically restrict the space of morphologies that evolution is free to explore. In his book on animal development, *The Triumph of the Embryo*, Wolpert says:

> "Developmental mechanisms, together with their genetic control, put a severe constraint on the evolution of animal form. It is not selective pressures that have kept the basic pattern of the vertebrate arm the same, but the fact that altering the basic pattern is almost impossible. Therefore, not all imaginable animals are possible. Any theory of evolution must incorporate an appreciation of developmental mechanisms." [Wolpert 93] (p.195).

There are many arguments over the relative importance of the spontaneous order inherent in the physical world, of the constraints imposed by developmental processes, and of natural selection in determining morphology (e.g. [Maynard Smith 86] p.43), but all these factors clearly have some part to play in the process. These considerations indicate that we cannot understand life purely through studying the genes, but we must also think about their *interactions* with the physical medium in which they are embedded. Richard Belew and Melanie Mitchell warn that "common metaphors for the genes (e.g., programs, blueprints, etc.) as the *information*-carrying component in evolution all rest on an inappropriate 'preformationist' view of information, as if information '... exists before its utilization or expression.' " [Belew & Mitchell 96] (p.14, original emphasis).[4]

Another important point about evolution taking place in a medium with some inherent self-organisational properties is that, although the possible forms achievable by evolution may be restricted, the 'instructions' for making those forms that *are* achievable can be compactly described on the organism's genome. In other words, the genome does not have to encode information about every aspect of the adult organism's design, because some features will just fall into place 'for free' as the developmental process unfolds, due to the self-organisational properties of the constituent matter.

Sex and Species[5]

Common conceptions of life are rooted firmly in our experience of organisms being grouped into species, with sexual reproduction within a species, and crosses between species being either infertile or impossible.[6] With the highly dissected flow of genetic information that results from the existence of different species, the possibility emerges for the specialisation of life forms into plants, animals, and, within these kingdoms, countless further subdivisions. Without species, evolution would be a continuous divergence—evolutionary novelty appearing in the genome of one individual would be passed on to its descendants, but would never spread to other lineages if there is no sex (in the most general sense of the mixing of genetic material from two or more sources in a single individual).

However, the existence of species, and of sexual reproduction, is not a universal feature of life. In their book *Origins of Sex*, Lynn Margulis and Dorion Sagan argue that "The fact that we are descendants of that minority of organisms that is sexually reproducing has led us to a very unbalanced and unrealistic view of biological sexuality" [Margulis & Sagan 86] (p.15). Indeed, the very existence of sexual reproduction in nature is a great puzzle from a neo-Darwinian point of view, as organisms which

[4] The enclosed quotation is from [Oyama 85].

[5] The ideas in this section are partially based upon a talk given by Richard Dawkins at the 'Digital Biota 2' conference in Cambridge, England on 10th September 1998.

[6] In fact, it is still not entirely clear whether the existence of species is a consequence of sexual reproduction ([Maynard Smith & Szathmáry 95], Chapter 9), but this does not affect the argument being made in this section.

engage in it (and therefore only pass on half of their genes to their offspring) would appear to be at an evolutionary disadvantage to those which reproduce asexually. The question of how sexual reproduction evolved, and the related, but separate, question of how, once it appeared, it has been maintained in the face of apparent selection pressure to remove it, constitute major research topics of modern evolutionary biology (see, for example, [Maynard Smith 86], [Margulis & Sagan 86]).

The reason for mentioning this subject here is that our common notions of what it is to be alive, through our experience of biological life, may include some aspects which are by no means universal laws of biology, but rather contingent features of the particular way in which life has evolved on our planet. This raises the question of whether we would be able to recognise artificial life if we ever succeeded in creating it, and to what extent we might want to try to replicate the course of *biological* evolution in our artificial systems. These issues will be discussed in Chapter 3.

Symbiogenesis

Another complication to the neo-Darwinist picture is the fact that many species enter into close ecological relationships with other species. The general term for this is 'symbiosis'. Unfortunately this term is used in various different ways in the biological literature,[7] but I use it as an umbrella term to cover three distinct types of inter-species relationship: *parasitism*, where one organism benefits from the relationship to the detriment of the other organism(s) involved; *commensalism*, where one organism benefits with no significant detriment or benefit to the other organism(s); and *mutualism*, where both (all) species benefit from the relationship.

The intimacy of such symbiotic relationships varies considerably. In 'ectosymbionts' the associations between organisms are purely external, whereas in 'endosymbionts' one of the organisms is actually incorporated internally into the other (the 'host').

In the extreme, it is possible that the relationship between host and (endo)symbiont becomes so strong that the previously independent organisms are totally dependent on each other for their survival and reproduction, and effectively become a single organism. This process, the evolutionary origin of new morphologies and physiologies by symbiosis,

[7] For discussion of this, see, for example, [Margulis 91], [Daida *et al.* 96].

is called ‘symbiogenesis’, a term coined by the Russian biologist Merezhkovsky and subsequently developed by others such as Kozo-Poliansky (see [Khakhina 79]).

It is now widely accepted that the process of symbiogenesis played a key role in the transition from prokaryotic[8] cells to eukaryotic[9] cells (e.g. [Margulis 81], [de Duve 96], [Maynard Smith & Szathmáry 95]). This has led a growing number of people to question whether symbiosis also played a key role in other major events during the evolutionary history of life [Maynard Smith 91].

The significance of symbiogenesis with respect to neo-Darwinism is that Darwin’s arguments concerning descent with modification seem to imply that evolutionary novelty comes about ‘vertically’, by mutations in the genes passed on from parents to offspring. In contrast, symbiogenesis is a mechanism for ‘horizontal’ gene transfer, where the genetic material from essentially unrelated organisms can be brought together in a single descendant. Whether this process is considered to be consistent with neo-Darwinism is a matter of debate. If one thinks about Darwinian evolution occurring at the fundamental level of the genes, it can certainly be argued that it is consistent (e.g. [Dawkins 76], [Maynard Smith 91]). For example, Maynard Smith says:

> “Whether [symbiosis constitutes a challenge to neo-Darwinism] depends, of course, on what one understands by ‘neo-Darwinism’. I interpret it to mean the hypothesis that mutation (change in the genetic material) is not, except occasionally and by accident, adaptive to its causative agent, and therefore, insofar as organisms are adapted to their ways of life, that adaptation must have arisen by natural selection acting on originally nonadaptive genetic variation. In this sense I am a neo-Darwinist. However, the ways in which genetic material from different ancestors is united in a single descendant has profound effects.” [Maynard Smith 91] (p.26).

Even if we accept that symbiogenesis is consistent with neo-Darwinism, we should recognise that this particular mechanism for producing evolutionary novelty has apparently played a key role at certain stages in the history of life on Earth. In terms of

[8] Present-day bacteria are examples of prokaryotic cells.

[9] All present-day animal, plant and fungi cells are eukaryotic. They are far more complex than prokaryotic cells, housing a wide array of specialised structures.

building artificial evolutionary systems, then, we should certainly give consideration to how symbioses and symbiogenesis might be achieved.

Life as Supple Adaptation

Mark Bedau has recently proposed an extreme evolutionary definition of life (e.g. [Bedau 96], [Bedau 98b]). Nearly all other definitions, whether evolutionary or ecological, see life primarily as a property of individuals (be they organisms, cells, or genes). In contrast, Bedau claims that a system capable of open-ended evolution (or 'supple adaptation' to use his phrase), *in its entirety*, should be considered as the primary form of life. A system exhibits supple adaptation by the "ongoing and indefinitely creative production of significantly new kinds of adaptive responses to significantly new kinds of adaptive challenges and opportunities" [Bedau 98b] (p.127). Bedau continues: "a supplely adapting system is an evolving population of organisms, or a whole evolving ecosystem of many populations, or, in the final analysis, a whole evolving biosphere with many interacting ecosystems" (*ibid.* p.131). In this view, particular components within the supplely adapting system, such as individual organisms, qualify as (secondary forms of) life by virtue of their specific relationship with the system. Exactly what relationships are required between a component and the system for the component to qualify as being alive is an open question. In suggesting this definition, Bedau emphasises that "natural selection is not sufficient for supple adaptation ... natural selection produces supple adaptation only when it is continually creative" (*ibid.* p.127).

At first glance, Bedau's suggestion might appear similar to James Lovelock's 'Gaia' hypothesis, that the whole Earth is equivalent to a single living organism [Lovelock 79]. However, the similarity is only superficial. Lovelock sees the Earth as a homeostatic, self-regulating entity; in other words, something which has the properties that we commonly associate with more standard biological organisms. Bedau, on the other hand, is certainly not claiming that the biosphere has these properties; what he is claiming is that the world must have a certain set of properties which enable it to act as a supplely adapting system if it is to be able to support the evolution of living (in the common sense of the word) organisms.

Bedau himself acknowledges that many obstacles must be overcome before a satisfactory definition can be produced in these terms, but claims that the approach has promise for providing good explanations for a number of fundamental puzzles about life. The primary interest of his proposal in the present context is that it defines (secondary forms of) life as being the product of the *right kind* of evolutionary process (i.e. one capable of supple adaptation), rather than of evolutionary processes in general. The precise characterisation of the kind of evolutionary process which might possess this capacity is an endeavour requiring much further work. The intriguing implication is that many of the specific phenomena related to biological life which are not usually associated with purely evolutionary definitions (e.g. self-maintenance, food webs, etc.: see Section 2.1.2) might actually be necessary features of any system capable of supple adaptation. In this way, such features might ultimately be incorporated into an evolutionary framework of the type suggested by Bedau. Of course, this is just a theory, like any other, but it is worth serious consideration. We will return to this topic in Chapter 7. In the following section we return to more traditional individual-based approaches, but those that have concentrated on ecological rather than evolutionary considerations.

2.1.2 The Ecological View

Distinct from the evolutionary approach to defining life, a separate, and older, tradition has sought to define life in more direct, ecological terms. This approach does not seek to answer the question of what an organism is in terms of the history of its predecessors, but rather in terms of its characteristic organisation; in plain terms, what it *is* and what it *does*.

Some have even argued that it makes no sense to define life in terms of evolution, as evolution is purely a mechanism for *change*. Francisco Varela points out that "reproduction requires a unity to be reproduced; this is why reproduction is operationally secondary to the establishment of the unity, and it cannot enter as a defining feature of the organisation of living systems" [Varela 79] (p.33). Similarly, in discussing the distinction between genealogical (evolutionary) and ecological hierarchies of nature, Stanley Salthe questions whether organisms (or similar entities) have a place in the genealogical hierarchy. "After all, much of population genetics deals directly with gene frequencies in demes and does not bother with genotypes *per se*" [Salthe 85] (p.235). An organism "has no significant extension in time; it cannot evolve. The organism as we usually think of it (ourselves) is clearly an ecological entity" (*ibid.*).

In looking at an individual organism, it is remarkable that there is very little that stays constant over its lifetime. Energy is continuously expended as it goes about its activities, and this lost energy is replaced by the regular ingestion of food or by the direct harnessing of sunlight. Even the very molecules from which an organism is made are generally in a state of flux.[10] In contrast to most other objects in the world, organisms also have a remarkable capacity for self-maintenance and repair in the face of environmental perturbations. As Waddington puts it, the characteristic form of living organisms is "not only, in many cases, a complex one, but the entity by which it is expressed is more nearly comparable to a river than to a mass of solid rock" [Waddington 57] (p.2).

The image of organisms as self-maintaining structures that are open to both energy and matter has led many to consider them as dissipative systems (e.g. [Haskell 40], [O'Neill *et al.* 86]). The vortex produced when the plug is pulled from a basin, a flame, and, to take a larger example, the Earth's atmosphere, are all examples of physical dissipative structures. They all require a continuous input of energy to maintain their structure, and dissipate this energy as a result. However, organisms differ from physical dissipative structures in their ability to exert *control* on the flux of energy and matter through them. We are drawn to the idea of organisms as *self*-producing and *self*-maintaining wholes.

This view of life dates back to Immanuel Kant in the late eighteenth century, if not earlier. Kant's view of an organism was one in which "each part existed both for and by means of the whole, while the whole existed for and by means of the parts" (from [Kauffman 95] (p.274)). The Chilean biologists Humberto Maturana and Francisco Varela have attempted to formalise this notion of self-production and self-maintenance in their definition of 'autopoietic systems'[11] (e.g. [Maturana & Varela 80], [Varela 79]). The formal definition of autopoiesis is:[12]

> "An autopoietic system is organized (defined as a unity) as a network of processes of production (transformation and destruction) of compon-

[10] Although the extent to which this is true varies greatly between different organisms, and between different parts within a single organism [Maynard Smith 86] (p.2).

[11] They invented the word 'autopoiesis' from the Greek $\alpha\upsilon\tau\acute{o}\varsigma$ = self, and $\pi o\iota\epsilon\iota\nu$ = to produce.

[12] Unfortunately the definition, at least in its English translation, is somewhat opaque.

> ents that produces the components that: (1) through their interactions and transformations continuously regenerate and realise the network of processes (relations) that produced them; and (2) constitute it (the machine) as a concrete unity in the space in which they exist by specifying the topological domain of its realisation as such a network." [Varela 79] (p.13).

In other words, an autopoietic organisation is one which actively produces and maintains its own structure. Maturana and Varela also suggest an informal definition:

> "an autopoietic machine is a homeostatic (or rather a relations-static) system that has its own organization (defining network of relations) as the fundamental invariant." (*ibid.*)

Kauffman claims that the intellectual lineage starting from August Weismann's view of the "germ line" has led to the loss of the earlier image of cells and organisms as self-creating wholes [Kauffman 95]. Varela agrees, saying that "the great developments of molecular biology have led to an overemphasis on isolated components, and to a disregard of questions pertaining to what makes the living system a whole, autonomous unity that is alive regardless of whether it reproduces or not" [Varela 79] (p.5).

Some computer models of autopoiesis are mentioned in Section 3.2.3, and I will discuss the relationship of concepts such as autopoiesis and self-maintenance to evolutionary processes in Sections 7.1.4, 7.3.2 and 7.3.3.

2.1.3 Hybrid Definitions

For non-biologists, much of what has been said above might appear to be fairly far removed from the popular concept of what it means to be alive.

High school biology textbooks usually present a list of properties which collectively constitute a definition of life. A typical list, from [Day 85] (p.34), is: Movement, Feeding (obtaining energy and raw materials), Respiration, Excretion, Growth (increase in size and complexity), Reproduction, and Sensitivity.

Many similar lists have also been proposed, each combining some aspects of organisms from the evolutionary perspective, and some from the ecological perspective. One of the major problems with check-list definitions, however, is that it is invariably possible to find an example of a real organism which fails to meet all of the requirements. A common problem case is the mule, which is certainly alive, although incapable of reproduction.

Considering the difficulties of such an endeavour, it is reasonable to ask whether we could *ever* arrive at a concise definition of life. Chris Langton, regarded by many as the founder of the scientific discipline of 'artificial life' (at least in its present incarnation), is skeptical that we will ever reach a consensus on this issue. From his experience of creating computer programs which exhibit many of the properties associated with living organisms, he remarks "every time we succeed in synthetically satisfying the definition of life, the definition is lengthened or changed."[13]

Attempts to fuse the ecological and evolutionary pictures in an illuminating and precise manner are complicated by the intricate, non-linear interactions involved. One of the major problems in developing a coherent picture of living systems is the description of the two-way interactions between processes occurring at vastly different time scales. Waddington says:

> "Perhaps the main respect in which the biological picture is more complex than the physical one, is the way in which time is involved in it ... [T]o provide anything like an adequate picture of a living thing, one has to consider it as affected by at least three different types of temporal change, all going on simultaneously and continuously. These three time-elements in the biological picture differ in scale. On the largest scale is evolution ... On the medium scale ... life history ... Finally, on the shortest time-scale ... rapid turnover of energy or chemical change." [Waddington 57] (pp.5–6).

In the 40 years since Waddington wrote this, little progress has been made in developing models which satisfactorily capture the interactions of processes happening at these different time-scales. Recent attempts to achieve such a synthesis are discussed by John Holland in his book *Hidden Order* [Holland 95] (esp. Chapter 5). Further efforts from the theoretical biology community at modelling the interface between ecology and evolution are discussed in [Travis & Mueller 89], [Stanley 89] and [Feldman 89].

[13] Quoted from [Kelly 94] (p.447).

In Section 2.1.2 some objections were raised to defining life in terms of evolution, as evolution is just a mechanism for change. Margulis and Sagan accept this, but claim that both autopoiesis *and* reproduction are distinguishing processes of living matter [Margulis & Sagan 86]. They argue that, while autopoiesis defines the organisation of an individual living entity, we still need to retain the idea that organisms reproduce in order to appreciate the larger picture of life:

> "Autopoiesis occurs, then, to maintain an organism during its own life, but by itself autopoiesis does not guarantee that an organism will show genetic continuity or that the characteristics of any given organism will persist faithfully through time. The process that ensures genetic continuity is reproduction. But autopoiesis remains the primary process. On the one hand, without it the organism would not survive to reach the stage at which reproduction becomes feasible. On the other hand, autopoiesis does not depend on reproduction, at least within a single generation. An infertile but healthy person with muscle, circulatory, excretory, and other organ systems in excellent running order is autopoietic even though he is unable to reproduce. In an evolutionary sense, however, such an individual has forfeited genetic continuity; he is already dead." [Margulis & Sagan 86] (p.13).

Any approach that redefines life using terms such as reproduction and autopoiesis has the great advantage that such terms can be precisely defined. Nils Barricelli, possibly the first person to implement and experiment with an artificial evolutionary system ([Barricelli 57], [Barricelli 62], [Barricelli 63]) clearly realised the problems with using words such as 'life'. In his work, Barricelli gave a precise definition of what he called a 'symbioorganism' (the work was based upon the concept of symbiogenesis, discussed earlier). Rather than asking whether the organisations that evolved in his system were alive, he turned the question upon its head:

> "As a matter of fact [the question of whether the evolved organizations are alive] has no meaning as long as there is no agreement on a definition of 'living being'. However, the reciprocal question 'whether the objects we are used to call [*sic*] living beings are a particular class of symbioorganisms'

has a meaning. This is the question we have been trying to answer in this paper ..." [Barricelli 63] (p.99).

This tactic of reversing the question, and using terms which can be precisely defined in preference to terms such as 'life', will be the one taken throughout most of this work. Clarification of some of the terms to be used is given in Section 2.5. We will return to the problem of providing an adequate characterisation of the concept of life in Section 7.3.3.

2.2 The Origin of Life

In the previous section, the distinction was made between definitions of life which emphasise the evolutionary perspective and those which emphasise the metabolic and/or ecological perspective. A similar distinction exists in models put forward to explain the *origin* of life on Earth.[14] The two approaches can be called the *replicator-first* approach and the *metabolism-first* approach.

The replicator-first approach assumes that the original seed for life was the existence of some sort of material that could exist in a large or infinite variety of forms, could reproduce more or less faithfully without the assistance of complicated machinery, and had some mechanism whereby specific other reactions or processes could become associated with specific forms of the material. This view was first explicitly expounded by Muller in the 1920s (see [Muller 66]). Candidates for the original self-replicating material of this type include RNA (for references see, for example, [Nuño *et al.* 95] and [Lazcano 95]), and clay minerals ([Cairns-Smith 71], [Cairns-Smith 85]). The presence of a simple self-reproducer of this nature is enough to begin a process of evolution. The idea is that some forms of the material may be such that they have processes associated with them (e.g. they may act as a catalyst for some reaction) that act to stabilise the material. Such forms will be favoured by natural selection (precisely because they are more stable), and evolution proceeds by selecting reproducers that catalyse more and more reactions that are beneficial to the stability of the replicator. At some point the reactions will effectively give the replicator complete control over the composition of its local environment, at which stage the network of reactions will probably fulfil our cri-

[14] Here I am only talking about models that assume a terrestrial origin of life.

teria for being a living organisation. Many prominent evolutionary biologists and chemists favour this approach, e.g. John Maynard Smith [Maynard Smith & Szathmáry 95], Richard Dawkins [Dawkins 76] and Graham Cairns-Smith [Cairns-Smith 85].

In contrast, the metabolism-first approach assumes that self-maintaining organisations were the seed of life. These models assume that the world is such that self-maintaining (collectively autocatalytic) organisations of chemical reactions occur spontaneously with reasonable probability. Being self-maintaining, they persist for reasonable durations. Another consequence of being self-maintaining is that they produce all of the components from which they are composed, so it is easy to imagine scenarios by which some organisations of this type might reproduce (e.g. by splitting in two). Such self-reproducing organisations will become more abundant, and will replace non-reproducers if there is competition for resources. With self-reproduction comes the possibility of evolution, so any variations of these self-reproducing and self-maintaining organisations that make them more stable will be selected for. By this process, the idea is that a genetic representation emerged by natural selection (in the right conditions)[15] to give the organisation a high degree of stability. With a genetic representation comes an unlimited range of hereditary variation, enabling the organisations to participate in prolonged evolutionary processes (see [Maynard Smith & Szathmáry 95] pp.67–72). This approach to the origin of life is favoured by Maturana and Varela (see, for example, Chapter 5 of [Varela 79]), and variations have also been suggested by, among others, Freeman Dyson [Dyson 85] and Stuart Kauffman [Kauffman 86].

2.3 The Pattern of Life

At this stage it is worth stepping back from the debate over definitions of life from the point of view of individual organisms, and considering the way in which life has evolved on Earth on a larger scale. We are interested in the fundamental question of which features of evolution we might expect to be universal (and therefore be apparent in any evolutionary system, natural or artificial), and which features are due to mere contingencies in the particular history of life on our planet (cf. the discussion of sex and species in Section 2.1.1). These considerations will affect the expectations we should

[15] One necessary condition is that the self-maintaining organisations were enclosed in some form of compartment, to enable natural selection to act *between* the compartments [Szathmáry & Demeter 87].

have for how an evolutionary process might unfold in an artificial system.

2.3.1 Evolutionary Progress

We might ask whether there is any underlying direction in evolution, such as a tendency for organisms to become more complicated. Although this might at first sight appear to be obviously so, the concepts of evolutionary progress and of the evolutionary increase in complexity are in fact the subject of great debate. (Note that an increase in complexity would be just one possible example of evolutionary progress; in other words, the concept of evolutionary progress is broader than that of complexity increase, and could be used in relation to many other measures.)

Countless putative examples of evolutionary progress exist in the literature, such as an increase in the maximum sizes of organisms [Bonner 88], an increased resistance of taxonomic groups to extinction [Raup & Sepkoski 82], and net progress in the expansion of life (e.g. increase in the number of species, the number of individuals, and in biomass and energy flow) [Ayala 88]. Daniel McShea criticises the evolutionary progress literature for concentrating on theorising rather than empirical inquiry, but he also warns that "marshaling cases does not document a pervasive trend either. The many increases could well be offset by an equal (or greater) number of decreases" [McShea 96] (p.477). Referring to the question of whether there has been an increase in the complexity of multicellular animals (metazoans) over the Phanerozoic Era, McShea concludes that "the evidence so far supports only agnosticism, indeed it supports an emphatic agnosticism" (*ibid.* p.489).

Almost as numerous as the number of examples of claimed evolutionary progress are the number of measures that have been used to track this progress. These measures are often intended to indicate the complexity of an organism, and include the length of coding DNA in an organism's genome [Cavalier-Smith 85], the number of different cell types in an organism [Bonner 88], and the number of species in a community (Bonner, *ibid.*).

Among the mechanisms which have been proposed to drive evolutionary progress, a common theme is the selective pressure and ecological niche opportunities that arise from other species in the ecosystem. Coevolution and symbiosis are general principles here, and have been applied to explain evolutionary progress in a number of specific ways. Examples include Van Valen's Red Queen hypothesis, in which it is postulated that even in a constant physical environment there will be indefinite evolutionary change as each species evolves to meet changes in others in the ecosystem [Van Valen 73]. A similar scenario was suggested by Waddington ([Waddington 69] pp.115–6). On a larger scale, consideration of population sizes and spatial heterogeneity are also common factors in explanations of differentiation between populations, such as allopatric models of speciation,[16] and Wright's "shifting balance" theory.[17]

Although many different mechanisms have been proposed to explain apparent evolutionary progress, McShea notices that most are based upon a small number of underlying principles [McShea 91]. These include some kind of ratcheting process, where a progressive step, once it occurs, is not easily lost (perhaps because the new development has become very tightly integrated with the rest of the system) and is thereby able to act as a stable base for further progression. Another common principle is the repetition-and-differentiation-of-parts theme (as noted by Darwin ([McShea 91] p.308) and Maynard Smith [Maynard Smith 86] (pp.45–47) among others).

However, many would claim that there is no consistent progress in evolution. D'Arcy Thompson, in his seminal book *On Growth and Form*, remarked "That things not only alter but improve is an article of faith, and the boldest of evolutionary conceptions ... I for one imagine that a pterodactyl flew no less well than does an albatross, and that Old Red Sandstone fishes swam as well and easily as the fishes of our own seas" [Thompson 17].[18] George Williams doubted that there was any consistent notion of complexity which can be said to have increased from the Paleozoic to the present day [Williams 66] (pp.42–43), and Stephen Jay Gould certainly agrees with these sentiments (e.g. [Gould 89]).

That Darwin's theory does not in itself predict evolutionary progress is emphasised by

[16] That is, speciation occuring as the result of the restriction (or total absence) of gene flow between two populations caused by some sort of physical barrier (e.g. a sea or mountain range). See, for example, [Maynard Smith 89].

[17] In the terminology of evolutionary algorithms, Wright's theory is a mechanism whereby an evolving population can avoid getting stuck in local (sub-optimal) fitness peaks. It involves the population being divided into a large number of small, partially isolated demes. Individuals in one deme may move to a higher fitness peak by chance (i.e. genetic drift), and subsequently spread throughout the whole population. See, for example, [Wright 31] and [Wright 82]; and also [Coyne *et al.* 97].

[18] Quoted from [Nitecki 88] (p.14).

Maynard Smith: “our theory of evolution does not predict an increase in anything. At first sight, Fisher’s [Fisher 30] ‘fundamental theorem of natural selection’ might seem to predict an increase in ‘mean fitness’, but it would be a mistake to think that there is any quantity that necessarily increases” [Maynard Smith 88] (p.220). Whether Darwin himself believed in the idea of evolutionary progress is itself a matter of debate (for an overview, see [Nitecki 88]). This is not surprising when some people quote passages such as: “as natural selection works solely by and for the good of each being, all corporeal and mental endowments will tend to progress towards perfection” [Darwin 59] (Chapter 14), and others quote: “After long reflection, I cannot avoid the conviction that no innate tendency to progressive development exists” [Darwin 72].[19]

At the root of much of this controversy lie the facts that the notion of progress is poorly defined, it can be used in a variety of different ways, and, above all, it is a value-laden notion. In an analysis of the concept, Francisco Ayala notes that progress “contains two elements: one *descriptive*, that directional change has occurred; the other *axiological* (= evaluative), that the change represents betterment or improvement” [Ayala 88] (p.78). It is the latter aspect of the concept that is the source of much of the disagreement. Many argue that we, as human beings, are not impartial observers, and that our value-judgements are easily influenced (even subconsciously) by cultural, political and religious considerations (e.g. [Ayala 88], [Hull 88], [Ruse 88], [Nitecki 88], [Gould 88], [McShea 91]). In stark terms, Gould says “progress is a noxious, culturally embedded, untestable, nonoperational, intractable idea that must be replaced if we wish to understand the patterns of history” [Gould 88] (p.319). The use of the term complexity is also problematic. McShea says “Complexity is more specific, more concrete, and therefore more tractable, but on account of its historical and still-commonplace association with progress, it carries the axiological taint” [McShea 91] (p.320).

2.3.2 Different Types of Complexity

McShea identifies a number of methodological problems with many previous studies of complexity increase, and remarks that “specifying an operational metric remains a difficult problem” [McShea 91] (p.318). In a later paper, he provides a useful analysis of the concept of complexity, and suggests that it can be broken down into a number

[19] Quoted from [Gould 89] (p.257).

of independent measures [McShea 96]. McShea's hope is that these measures are both operational (they can unambiguously be measured in real systems) and universal (they can be applied to all systems).

The definitions that McShea proposes are based upon two dichotomies: object versus process, and hierarchical versus nonhierarchical structure. These dichotomies yield four different types of complexity: (1) Nonhierarchical object complexity; (2) nonhierarchical process complexity; (3) hierarchical object complexity; and (4) hierarchical process complexity.[20] These different types are illustrated in Figure 2.1. In the context of his work on metazoan complexity, object complexity refers to morphological complexity, and process complexity to developmental complexity. McShea discusses various measures and proxies for obtaining these data from fossils. By taking this narrower, more specific view of complexity, a much needed degree of objectivity can be introduced. As a final remark on the notion of overall complexity, McShea says "Is a human more complex than a trilobite *overall*? The question seems unanswerable in principle because the types of complexity are conceptually independent ... Thus, it is hard to imagine how a useful notion of overall complexity could be devised" [McShea 96] (p.480).

2.3.3 Driven and Passive Trends

In his work, McShea also considers the possible causes that have been proposed to explain evolutionary trends, and highlights a broad distinction between *driven* and *passive* accounts (e.g. [McShea 94], [McShea 98]). A driven trend can occur when a directed, pervasive force exists in a system, causing a particular measure to move in one direction rather than another. However, McShea points out that trends can occur in measures even in the absence of such driving forces (i.e. passive trends). These may occur when some sort of boundary exists in the system, representing a minimum (or maximum) below (above) which the measure is unable to move. The distinguishing feature of such boundaries, compared to the biasing forces that cause driven trends, is that they only operate on a small subset of the state space. As a simple (if somewhat implausible) example, imagine a multicellular organism (consisting of a very small

[20] McShea also mentions a third possible dichotomy: differentiation versus configuration. The four types of complexity mentioned above are differentiational. Configurational complexity, on the other hand, is "irregularity of arrangement of parts and interactions, independent of their differentiation" [McShea 96] (p.480). McShea claims that configurational complexity has received little attention in the biological literature, so he does not consider it further in his work.

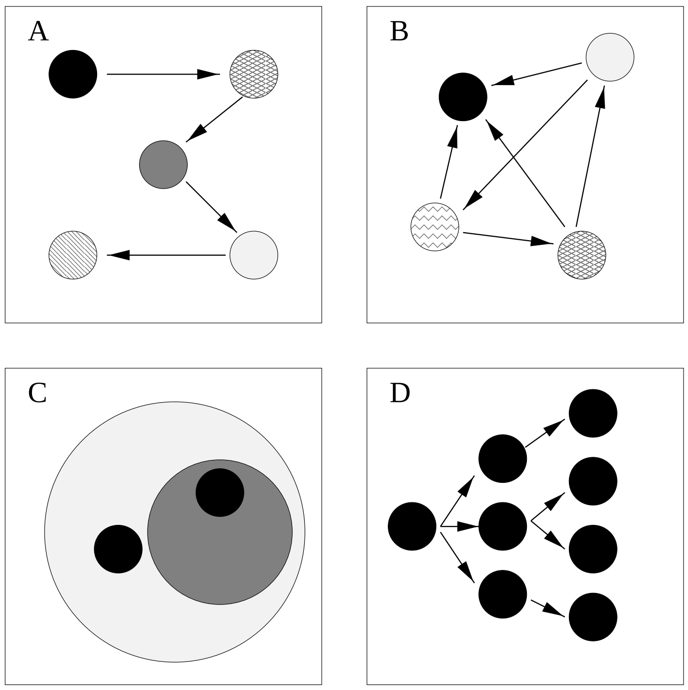


Figure 2.1: Different Types of Complexity. (A, B) Nonhierarchical complexity: A has greater nonhierarchical *object* complexity than B (it has more different parts); B has greater nonhierarchical *process* complexity than A. (C, D) Hierarchical complexity: C has a degree of hierarchical *object* complexity, D has a degree of hierarchical *process* complexity. Adapted from [McShea 96].

number of cells) which is free to evolve both in the direction of more cells and in the direction of fewer cells, with no selection pressure favouring evolution one way or the other. Both larger and smaller organisms arise by chance as evolution proceeds, so the diversity of observed sizes increases over time. Now, imagine that there is no particular upper limit on the size of the organisms which may evolve (at least, perhaps, until they consist of *millions* of cells). However, there must be a lower boundary on organism size—each organism must comprise *at least one* cell. In this situation, the mean observed organism size will increase, due to the existence of a boundary but in the absence of any directed force.

McShea suggests various ways by which these two causes may be distinguished, but remarks that “most explanations that have been proposed for complexity trends implicitly invoke biases and thus are driven ... In contrast, little has been said about possible causes of boundaries, a subject which is ripe for deeper theoretical investigation” [McShea 96] (p.486). In our work on artificial evolutionary systems it is therefore important to remember that some of the patterns of life observed in the biological world may not be due to pervasive directed forces, but rather to the existence of boundaries.

2.3.4 Major Evolutionary Transitions

Much of the debate over evolutionary progress, based upon evidence from the fossil record, is concerned with the pattern of evolution during the Phanerozoic Era (i.e. from the first occurrence of abundant fossils of multicellular organisms to the present day). However, multicellular organisms evolved from unicellular organisms, and those, in turn, must have evolved from simpler origins [Maynard Smith & Szathmáry 95]. The one measure of progress upon which there seems to be some agreement is that of life having passed through a succession of major hierarchical transitions. In his book *The Evolution of Individuality*, Leo Buss remarks that:

> "Life, beginning as self-replicating molecules, did not persist in this initial state. Rather, it subsequently elaborated 'vehicles' in which the original heritable units became increasingly distanced from direct interaction with the external environment" [Buss 87] (p.183).

One succession that has been proposed begins with self-replicating molecules as the first stage of the evolution of life (but see Section 2.2). These molecules later grouped together into larger collections of self-replicators, which became enclosed within a membrane. From these 'protocells', prokaryotic cells evolved, and collections of prokaryotes subsequently became associated with each other to form eukaryotes. From simple unicellular eukaryotes, multicellular organisms evolved, and single multicellular organisms then grouped together in various social patterns. Such a progression has been proposed by Maynard Smith and Szathmáry ([Szathmáry & Maynard Smith 95], [Maynard Smith & Szathmáry 95]). There is still some concern over the consistency of the criteria used to define each of the transitions (e.g. [McShea 96]), but the majority involve an increase in hierarchical object complexity at least.

In Buss' treatment of the subject, he says "At Darwin's strong urging, the notion of evolution by natural selection and the notion of progress have been divorced from the outset ... [However] in the limited sense that life evolves hierarchically ... progress is inherent in the evolutionary process" [Buss 87] (p.186).

If we restrict ourselves to considering only those transitions which *do* involve an increase in hierarchical object complexity, we see that each entails the absorption of a number of previously free-living individuals into a new, hierarchically higher, unit of selection. When a transition of this nature occurs, two major factors are involved [Buss 87]. The first is a synergism between the interests of the individuals at the lower level and the new 'individual' at the higher level. This synergism "will act to create novel organizations allowing exploitation of the external environment in ways that the lower unit alone could not accomplish" (*ibid.* p.185). However, the individuals at the lower level might evolve in ways that disrupt the higher level organisation. Thus, the second factor is a conflict between the two levels of selection. "The rate and magnitude of such conflicts must be limited, or the higher unit will perish. If variants arise in the lower unit whose effect is to limit the occurrence or magnitude of subsequent variation, then the higher unit will eventually become resistant to further perturbation" (*ibid.* p.185).

Once a transition has occurred, then, the new unit of selection must become stabilised if it is not to break down into its component subparts again. (Maynard Smith and Szathmáry also suggest that the genetic relatedness of individuals in the lower level may also play an important role in conferring immediate selective advantage at the time of transition [Maynard Smith & Szathmáry 95] (p.8).) If the new organisation *does* become stabilised, then it, in turn, is able to participate in the same sort of process, leading to yet higher organisational levels. Bronowski's term "stratified stability" captures this picture of progression well [Bronowski 73].[21]

Buss summarises: "the organization of any unit will come to reflect those synergisms between selection at the higher and the lower levels which permit the new unit to exploit new environments and those mechanisms which act to limit subsequent conflicts between two units" [Buss 87] (p.viii).

From this point of view, a new perspective on the pattern of evolution emerges. It is a picture of major hierarchical transitions, with long periods in between during which it is debatable whether any consistent notion of progress applies.[22] Buss claims that

[21] Although Bronowski used the term in a somewhat broader context.

[22] Note that I am *not* talking about the distinction between gradualism and punctuated equilibria here; this view of evolution *does* allow for evolution in the periods between major transitions. However, the view holds that evolution during these periods will not necessarily be in the direction of increased complexity.

"the major features of evolution were shaped during periods of transition between units of selection" [Buss 87] (p.188). In the period after a transition, an adaptive radiation of forms is possible to fill the novel evolutionary niches afforded by the new kind of organisation. In the context of organic evolution, Salthe suggests that this expansion continues until a saturation point of number of kinds is reached, "when the emphasis of the process ... shifts toward coevolutionary elaborations of pairs, guilds, and even more complex symbioses" [Salthe 85] (p.253). The saturation point can be released if, for example, a subset of the organisms become geographically isolated and able to participate in a new adaptive radiation, but none of these processes will necessarily lead to any general evolutionary progress.

This view of evolution is also consonant with Schwemmler's notion of a constant oscillation between divergent and convergent phases [Schwemmler 89], and, to some extent, with Gould's picture of diversification and decimation [Gould 89].[23]

2.3.5 Contingency in Evolution

Proponents of the general picture of evolution discussed in the previous section tend to emphasise the role of contingency ('historical accidents') in determining the course of evolution. In *The Evolution of Individuality*, Buss says:

> "the fact that life is hierarchical redirects attention to the effect of history on biological phenomena. The major features of genomic organization, of cell architecture, and of organismal ontogeny arose as the products of history-dependent variation at the time of the transition from one unit of selection to another. Units which persist today do so because variants which restrained further interaction between two units of selection arose and fixed the organization of the unit in question at a given state" [Buss 87] (p.187).

Similarly, Maynard Smith says "My own view is of a series of historical accidents, subject to engineering constraints on the one hand, and to the conservatism of development on the other" [Maynard Smith 86] (p.45). The 'engineering constraints' arise when there may be only a small number of ways of solving a particular engineering

[23] Although Gould's view is also connected to the concept of punctuated equilibria, which is not specifically related to transitions between units of selection.

problem faced by a species. By 'conservatism of development', Maynard Smith is referring to the fact that "it seems to be a general feature of evolution that new functions are performed by organs which arise, not *de novo*, but as modifications of pre-existing organs" (*ibid.* p.46).

Finally, Gould's view on the subject is as follows:

> "Invariant laws of nature impact the general forms and functions of organisms; they set the channels in which organic design must evolve. But the channels are so broad relative to the details that fascinate us! The physical channels do not specify arthropods, annelids, mollusks, and vertebrates, but, at most, bilaterally symmetrical organisms based upon repeated parts ... When we set our focus upon the level of detail that regulates most common questions about the history of life, contingency dominates and the predictability of general form recedes to an irrelevant background ... Charles Darwin recognized this central distinction between 'laws in the background' and 'contingency in the details' ..." [Gould 89] (pp.289–290).

2.4 Playing the Game

Having discussed some of the issues that arise when considering how evolution has unfolded on Earth, it is time to ask how this affects our attempts at producing a definition of life. The distinction has already been made between evolutionary and ecological views of life (Sections 2.1.1 and 2.1.2 respectively), and in Section 2.1.3 it was noted that an adequate picture of living things would require a combination of these perspectives. However, it was also noted that it is hard to imagine a succinct definition which could capture this.

The discussion in the previous section only serves to emphasise the entanglement between the evolutionary and ecological pictures. Living organisms are taking part in the 'game' of life. Fundamental aspects of this game include ecological interactions with other organisms and with the abiotic environment, the exchange and transformation of resources, and the coevolution of all organisms, together with aspects of their abiotic environment. If we define life in terms of an individual organism, it is easy to underestimate the importance of that organism's interactions with the rest of the world. These interactions are inseparably linked to the concept we refer to as 'life'. Organisms exist both as products and as players in this game. It is only as both of these things that an individual organism can be said to be 'alive'.

2.5 Definition of Terms

Before continuing it is worth clarifying some of the terminology relating to evolutionary systems that will be used in subsequent chapters. It has been noted by various people that some confusion exists in the use of terms such as *genes*, *replicators* and *lineages* (e.g. [Hull 80], [Salthe 85], [McMullin 95]). The confusion arises because terms such as gene and replicator are sometimes used to refer to an *individual* instance of a specific form, e.g. a specific portion of DNA within one particular organism, and sometimes to refer to a *class* of forms, members of which can be considered identical from the point of view of their acting in an evolutionary process (e.g. it is in this sense that we refer to a gene being *passed on* (*replicated*) from one generation to the next).[24]

In an attempt to remove this ambiguity between referring to tokens (individual instances) and classes, McMullin introduces two prefixes, 'A-' (standing for *Actor*) and 'L-' (standing for *Lineage*) [McMullin 95]. The former signifies an individual token, and the latter signifies a class (read the 'L-' prefix as 'a lineage of ...'). For example, an A-replicator is an individual (molecule, strand of DNA, etc.), and an L-replicator is a lineage of A-replicators. McMullin goes on to single out lineages of replicators in which all members are identical to the founder of the lineage with respect to a particular selectively relevant character. These lineages, which he calls similarity lineages, or S-lineages, are, he argues, the fundamental units of selection in a Darwinian evolutionary process.[25] Phrases such as "genes are potentially immortal" refer to L-genes, and, in particular, to S-lineages. Although McMullin regrets the introduction of yet

[24] Salthe, in considering the hierarchical nature of life, argues that there is a clear distinction between the ecological hierarchy and the genealogical (evolutionary) hierarchy [Salthe 85]. The confusion in terminology under discussion here can also be seen as a confusion between entities in these two distinct (but related) hierarchies.

[25] That is to say, S-lineages are the fundamental *types of things* which act as units of selection. McMullin is *not* arguing here about whether it is more appropriate to look at evolution from the point of view of the gene or the organism, for example, but claims that "*whichever* of these viewpoints may be adopted, there will be a crucial distinction between actors and S-lineages, with only the latter being properly regarded as units of selection, or entities for whose benefit (Darwinian) adaptations may be said to exist" [McMullin 95] (p.166, original emphasis).

more terminology (as do I), he sees no other way to disambiguate confusing terms such as 'replicator'. I shall therefore use these prefixes throughout this thesis in situations where potential ambiguities may otherwise arise.

We are now in a position to be somewhat more precise about the meaning of a number of concepts that were used in Chapter 1.

Open-Ended Evolution. This term refers to a system in which components continue to evolve new forms continuously, rather than grinding to a halt when some sort of 'optimal' or stable position is reached. In other words, using the terminology just introduced, new, adaptively successful, S-lineages are continuously appearing and displacing existing S-lineages. Note that open-ended evolution does not necessarily imply any sort of evolutionary *progress*. Also note that this is still not a perfect definition; there are some more subtle issues involved in determining what counts as 'new' in this context. Also, by using the term 'open-ended' I wish to imply that an indefinite variety of phenotypes are attainable through the evolutionary process, rather than continuous change being achieved by, for example, cycling through a finite set of possible forms. I will return to some of these issues in Chapter 7 (for example, in Sections 7.1.2 and 7.3.2).

Complexity. The distinctions proposed by McShea, described in Section 2.3.2, will be adopted here, so we will not use the term 'complexity' unqualified. As the major evolutionary transitions described in Section 2.3.4 were critical points in the evolution of life on Earth, we will be especially interested in investigating hierarchical complexity (primarily hierarchical object complexity) in artificial evolutionary systems.

Life. Unfortunately we seem to be no closer to arriving at a satisfactory definition of life. Maturana and Varela's concept of autopoiesis may be adequate for the classification of individual organisms. However, as we are ultimately interested in *creating* artificial life, or at least in understanding why life exists on Earth, we need to look at a somewhat broader picture which considers the generative processes by which life has emerged. Both evolutionary and ecological considerations will be important for this purpose. We will keep these ideas in mind when developing our artificial models, but specific research questions will be phrased in terms of more specific concepts such as 'hierarchical object complexity' and 'open-ended evolution'. In Chapter 7 we will revisit the problem of providing an adequate characterisation of the concept of life, in the light of experience gained from building and experimenting with an 'artificial life' system (described in Chapters 4–6).

2.6 Summary

In this chapter we have discussed the two distinct ways of looking at life: the evolutionary and the ecological pictures. Although distinct from a conceptual point of view, both processes have contributed in an intricately interrelated manner to the pattern of life we see on Earth. Both views are therefore necessary to gain a satisfactory picture of life.

The concept of evolutionary progress was also discussed. Despite common preconceptions, evidence for evolutionary progress is only available in a relatively small number of fairly specific cases. Whether evolutionary progress, and in particular an evolutionary increase in complexity, has been a general feature of biological evolution is a subject of much debate. Despite the lack of evidence, many theories have been put forward to explain how progress may come about. Common mechanisms that have been utilised in these theories include some kind of ratcheting, and the idea of repetition-and-variation-of-parts. For explaining speciation and other macroevolutionary patterns, consideration of population sizes and spatial heterogeneity is also often employed.

One sense of progress on which there does seem to be broad agreement is that life has evolved in a number of hierarchical steps, starting with individual replicating molecules, and progressing through stages including unicellular organisms and then multicellular organisms. Although the details of each transition may be different, and depend on fairly particular, contingent factors, it is possible that some common features exist. The level of explanation at which these commonalities will be clearest is likely to be in the consideration of synergisms and conflicts between levels of selection. It has also been suggested that many of the major features of life are forged at the time of such transitions.

In the next chapter, a review is presented of the many ways in which people have applied the ideas discussed here to computer simulations of evolution and life, and to the creation of 'artificial life'.

Chapter 3

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[1]

3.1 What is Artificial Life?

Chris Langton, whom many regard as the founder of the research discipline known as Artificial Life, defines the subject as the study

> "of man-made systems that exhibit behaviors characteristic of natural living systems. It complements the traditional biological sciences concerned with the *analysis* of living organisms by attempting to *synthesize* life-like behaviors within computers and other artificial media. By extending the empirical foundation upon which biology is based beyond the cabon-chain [*sic*] life that has evolved on Earth, Artificial Life can contribute to theoretical biology by locating *life-as-we-know-it* within the larger picture of *life-as-it-could-be*." [Langton 88] (p.1, original emphasis).

As a brief aside, one might argue that the implication in this quotation that traditional theoretical biologists are concerned only with terrestrial biology is not entirely accurate.

[1] Quoted from [Kelly 94] (p.451).

Although the approach may be different, the goal of attempting to develop a truly universal theory of biology (i.e. a general theory which might, for example, predict the forms life might take on other planets, in contrast to our rather provincial view of life on Earth) is one that is shared by many biologists—for example, John Maynard Smith (e.g. [Maynard Smith 86] pp.21–23) and Richard Dawkins (e.g. [Dawkins 83]), to name but two.

Langton's publications in the mid-1980s in which he brought the term 'artificial life' into common usage (e.g. [Langton 86]),[2] together with his organisation of a workshop on the subject in Los Alamos in September 1987 (which has subsequently grown into the biennial International Conference on Artificial Life), were certainly the principal factors contributing to the current popularity of the subject (and to its crystallisation as a coherent subject[3]). However, research which would today certainly be classified as 'artificial life' began almost as soon as the modern digital computer was developed, with Nils Barricelli's work at the U.S. Institute for Advanced Study commencing in 1953, described in Section 3.2.2. The theoretical background for the subject was being developed even earlier, by von Neumann [von Neumann 66], for example, and by proponents of the cybernetics field (e.g. [Weiner 48]). A good history of much (but certainly not all) of this earlier work is provided by Langton himself in [Langton 88].

3.1.1 Weak Artificial Life versus Strong Artificial Life

The word 'artificial' is commonly used in two distinct ways. One carries the connotation that the thing being referred to is a fake—an imitation of something else (e.g. 'artificial flowers'), but the other merely suggests that the process by which the referent was produced is man-made or unnatural in some sense, but the thing itself is just as real as the original (e.g. 'artificial light').

These two uses of the term correspond to the contrasting philosophical positions of *Weak* and *Strong* Artificial Life respectively. Proponents of Weak Artificial Life view their work as an attempt to learn more about biological life (or to create artefacts

[2] Although others had used the term in earlier publications, such as [Overton 82] (in which it appears in the title without definition in the text, and in fact, by comparing the contents of the main text with that of the editor's note at the beginning, it appears that the title (including the term 'artificial life') was chosen by the editor, Edmund C. Berkeley, rather than by Overton himself).

[3] Although some would argue over the extent to which it *is* a coherent subject. See Sections 3.1.1–3.1.3.

such as robots) by creating synthetic models of various processes associated with living organisms (e.g. evolution) on computers and other artificial media, but do not claim that any parts of their models are actually living themselves. On the other hand, proponents of Strong Artificial Life claim that by instantiating such processes in artificial media, the end product will be just as deserving of the term 'living' as are biological organisms. In other words, the former group see their computer programs as *simulations* of life, while the latter group see them (potentially) as *realisations* of life [Pattee 88].

This distinction is related in some ways to the argument between those who claim that life depends crucially upon the physical medium in which it is implemented, and those who claim that life is fundamentally a *process* (or a collection of processes) and as such can be implemented in any physical medium which has the logical structure to support such processes. Arguments for the former view have been put forward by, for example, Elliott Sober [Sober 91], Claus Emmeche [Emmeche 92], Eric Olson [Olson 97], and Margaret Boden [Boden 99] (although she claims that this depends on how we define metabolism). Proponents of the latter point of view, that life is independent of matter, include Langton himself [Langton 86] and Tom Ray [Ray 91], creator of the 'Tierra' system (described in Section 3.2.1). A version of the former view which offers more hope for the endeavour of Artificial Life, and the view with which I have most sympathy, is that life must be embedded within a 'physical' environment, but that the symbolic environment provided by a computer program (that is, an 'artificial physics') *might* be sufficient for this purpose (e.g. [Pattee 88], [Rasmussen 91], [Pattee 95a], [Bedau 98b]). However, this view comes with the proviso that the artificial physics of the computer model must resemble the important aspects of the physics of the real world if the model is to be treated as science rather than just a computer game (see, for example, [Moreno *et al.* 94], [Morán *et al.* 97] and [Ruiz-Mirazo *et al.* 98]). A major challenge for the science of artificial life will be to develop theories of exactly which aspects of the real world are important in this sense.

A glance at any of the recent proceedings of conferences on artificial life (for example, [Adami *et al.* 98], [Husbands & Harvey 97]) reveals that the field now encompasses a very wide variety of research, including synthetic evolutionary models, artificial chemistries, models of autopoiesis, self-organising systems, evolutionary robotics and evolutionary hardware. The methods of artificial life are now also being employed for artistic purposes,[4] and work of this nature is also being integrated into these conferences. The majority of current work in the field (at least as published in scientific journals and conferences) does not postulate any more than the weak position. In other words, the premise is that by modelling particular natural processes such as evolution, we can improve our understanding of those processes and their relevance to the biological world, or, alternatively, we can profitably use such processes for various practical purposes (e.g. to develop robust control systems for robots).

However, a minority of published artificial life research explicitly takes the strong position. The protagonists, not surprisingly, include those who are most vociferous about life being independent of matter, such as Langton and Ray. For example, Ray says "I would consider a system to be living if it is self-replicating, and capable of open-ended evolution" (*ibid.* p.372).[5] Ray's requirement that living systems have the capacity of open-ended evolution is similar to Bedau's concept of life as supple adaptation, discussed in Section 2.1.1. However, the definition does not prescribe how such a capacity can be assured. In the following chapters, much time will be spent investigating the behaviour of an artificial life platform called 'Cosmos', based upon Tierra, in which populations of self-replicating computer programs evolve. I have developed this system largely to study its evolutionary behaviour, but I certainly do not equate self-replication with life. However, I think the relationship between open-ended evolution and life is more interesting. After reporting the lessons learned from running an extensive series of experiments with Cosmos (in Chapters 5–6), I will return to a further discussion of this relationship in Chapter 7.

3.1.2 Is Artificial Life Possible?

As mentioned in the previous chapter (Section 2.1.3), without a clear definition of the word 'life', questions such as "is artificial life possible?" really have no meaning. However, assuming that the question can be suitably rephrased in appropriately concrete

[4] For example, see Steven Rooke's online portfolio at `http://www.concentric.net/~Srooke/` and the list of genetic art-related sites at `http://gracco.irmkant.rm.cnr.it/luigi/alg_art.htm`.

[5] The concern with self-replication has also been a preoccupation for other artificial life researchers from von Neumann [von Neumann 66] to the present day, and can be related to the work of biologists such as Muller and Dawkins, discussed in Section 2.2.

terms,[6] there are still a number of philosophical and practical hurdles to be overcome.

A major practical barrier to creating artificial life is simply the vastness of life in the biological realm, both in terms of the numbers of molecules comprising the biosphere, and of the billions of years over which evolution has been proceeding. Even if artificial life were a theoretical possibility, why should we expect it to proceed any faster on a computer only capable of supporting a trivially small population?

If we are eventually hoping to produce *intelligent* artificial life, the chances seem even slimmer. Gould remarks that "*Homo sapiens* is an entity, not a tendency" [Gould 89] (p.320). Expanding upon this, he says:

> "Life arose at least 3.5 billion years ago, about as soon as the earth became cool enough for stability of the chief chemical components ... [However], a good deal more than half the history of life is a story of prokaryotic cells alone, and only the last one-sixth of life's time on earth has included multicellular animals ... But cosmologists tell us that the sun is just about at the halfway point of existence in its current state; and that some five billion years from now, it will explode ... Since human intelligence arose just a geological second ago, we face the stunning fact that the evolution of self-consciousness required about half of the earth's potential time. Run the tape again, and even if the same general pathways emerge, it might take twenty billion years to reach self-consciousness this time—except that the earth would be incinerated billions of years before." [Gould 89] (pp.309–311).

Another argument against the possibility of artificial life concerns the nature of the major evolutionary innovations in biology, which seem to have provided organic evolution with an unlimited supply of phenotypic novelty. It has been suggested by Howard Pattee that most of these innovations involve the invention of techniques for measuring new aspects of the environment, and of arbitrarily mapping these patterns to symbols or to specific actions [Pattee 88]. Pattee emphasises that one of the major hurdles for

[6] This, of course, is a big assumption. Those who maintain that life is of necessity a biochemical phenomenon would argue that concepts such as autopoiesis do *not* capture all of the relevant features of life. However, this eventually boils down to a linguistic debate, and scientific progress can only be made when we do adopt precise definitions, even if these are not universally agreed upon.

(strong) artificial life will be to develop a satisfactory theory of measurement which suggests how we might equip our programs with the potential to evolve new measurement devices (*ibid.*). The evolution of new forms of action (e.g. new effectors) is equally problematic.

Furthermore, such innovations usually develop from existing phenotypic structures employed for unrelated functions, but which happen to perform (however crudely) the novel task as a side-effect. To the extent that this is true, this presents a problem for the designers of artificial life systems. A computer model, like any other scientific model, is an idealisation of the real world, designed to model (what the designer thinks are) the important components of a system, while ignoring irrelevant aspects. However, the present argument suggests that a primary source of evolutionary novelty might lie in precisely those aspects of the physical world which might ordinarily be deemed irrelevant for the purposes of constructing a model. We are again confronted by our lack of an adequate theory of measurement, as pointed out by Pattee. Even if we were to model a wide variety of properties for each component in our artificial life system, we might only expect complex forms to emerge if they existed in a very complex environment which afforded many possibilities for perception and action. Some steps which we might take to ensure a rich environment are discussed in Chapter 7, but the issue of evolving novel sensors and effectors in a computer model remains problematic.

These are formidable arguments, and there are a number of others which can also be made against the likelihood of success (see [Bonabeau & Theraulaz 94]). The number of points that computer models have in their favour seem to be few in comparison (speed of execution might be one example).[7] I take the pragmatic attitude that we simply do not know how successful the approach might be until we try it. Eric Bonabeau and Guy Theraulaz argue that "rather than true limitations, [issues such as these] constitute questions asked to AL [Artificial Life]. And AL is precisely a constructive way of checking whether these limitations are real obstacles" [Bonabeau & Theraulaz 94] (p.314). A detailed discussion of the philosophy of artificial life can be found in [Boden 96]. Even if the quest for strong artificial life ultimately ends in failure, the attempt can highlight questions and gaps in our knowledge which were not apparent, or did not seem particularly relevant, beforehand, and indeed it already has done.

[7] But note that artificial life does not restrict itself to using digital computers as the medium of implementation.

3.1.3 Relation to Theoretical Biology

Traditionally, most models used in ecology and evolutionary biology have been systems of differential equations. These equations track changes in macroscopic measures of the system being modelled, such as the frequency of an allele in a population's gene pool, or the amount of energy in a trophic level. This approach is appropriate for modelling a wide variety of systems, and has led to very successful models in numerous cases. It is, however, subject to a number of limitations. Charles Taylor (a biologist) and David Jefferson (a computer scientist) give some examples:

> "For example, in many models it is common to refer to the derivative of a variable with respect to population size N. This in turn implies the assumption of very large populations in order for such a derivative to make sense, which has the effect of washing out small population effects, such as genetic drift, or extinction. Another difficulty is that it would take tens to hundreds of lines of equations to express even a simple model of an organism's behavior as a function of the many genetic, memory, and environmental variables that affect its behavior, and there are simply no mathematical tools for dealing with equational systems of that complexity. Furthermore, equational models are generally poor at dealing with highly nonlinear effects such as thresholding or if-then-else conditionals, which arise frequently in the description of animal behavior." [Taylor & Jefferson 94].

Additionally, it is difficult to satisfactorily model spatial inhomogeneity and incomplete mixing with this approach, especially when considering continuous rather than discrete spatial structure (see [van Baalen & Rand 98], [van Baalen 98]). In discussing ecological modelling techniques, M.A.R. Koehl refers to models of macroscopic measures of a system as phenomenological models. He says of the shortcomings of such models:

> "The limitations of phenomenological models render them inappropriate for certain types of analysis. Whenever we make a prediction using a phenomenological model, we implicitly assume (1) that conditions do not change, and (2) that the phenomena that go into the model adequately sample the causal pattern of interest. Therefore, phenomenological models are best for making short-term predictions." [Koehl 89].

In other words, by their very nature differential equation models require that all components to be modelled are explicitly specified in the equations, so they cannot generally be used to model the emergence of *new* components.

In contrast, artificial life models dispense with equations, and represent individuals explicitly. That is, the artificial life approach is fundamentally synthetic rather than analytic (see Section 3.1). Artificial life models are similar to what Koehl refers to as mechanistic models in ecology [Koehl 89], where it is the essential processes governing components of the system that are modelled. As computers become faster and more powerful, it becomes possible to model larger populations of individuals, and to use a more sophisticated representation for each individual. Indeed, the growth of interest in artificial life in the late 1980s can be largely attributed to the fact that sufficiently powerful computers first became readily available at around that time. However, some interesting and worthwhile studies were conducted a long time before then, and will be reviewed in Section 3.2.

Artificial life models may be compared to microcosm experiments in ecology [Koehl 89], where a limited ecology of organisms with small size and short generation times are studied in the hope that they may reveal basic principles which are valid for larger systems as well. Such microcosm experiments are also easier to control and replicate than are larger-scale studies, and this capacity for controlled experimentation is even greater in computer-based artificial life models. In his discussion of microcosm models, Koehl warns that they "are certainly useful to test models, but are they so unrealistic that they tell us little about nature?" (*ibid.* p.43). One of his main concerns is the size of the microcosm compared to natural systems, and similar worries about the size of artificial life models were raised in Section 3.1.2. Koehl takes the pragmatic view that we should not forget about such issues, but use microcosm studies in order to "pursue sensible answers to these questions" (*ibid.* p.43). I have a similar view of artificial life studies, as discussed in Section 3.1.2.

In considering the scientific status of artificial life models, Holland argues that computer simulations (weak artificial life) can be seen as a bridge between experimentation and theory—a sort of half-way house on the road to a more rigorous mathematical treatment (which might not be possible until new mathematical tools are developed) [Holland 95]. Daniel Dennett suggests that they can also be used as a bridge between experimentation and philosophy [Dennett 94]. Jason Noble, adopting the philosophical viewpoint known as 'unrepresentative realism', argues that artificial life models *can* be treated as scientific theories in their own right, but *only* if appropriately formulated (e.g. they must be based upon explicit axioms)[8] [Noble 97]. In particular, he points out that they are useful for testing the logical coherence of a given set of assumptions. Pattee also emphasises the need for simulations to be based upon solid theories, and additionally discusses the conditions under which we might consider an artificial life program as a *realisation* of life rather than a *simulation* of life [Pattee 88].

A number of people have warned of the dangers of taking the concept of 'life-as-it-could-be' too far, and argue that artificial life can only be treated as science if it sufficiently reflects the constraints and boundary conditions operating in the real world (e.g. [Bonabeau & Theraulaz 94], [Morán *et al.* 97], [Noble 97]). Geoffrey Miller has suggested that the science of artificial life should restrict itself to tackling established problems of theoretical biology [Miller 95], although others have argued (and I agree, at least if Miller's suggestion is taken too restrictively) that this goes too far, and that the approach opens up new areas for research which were not amenable to more traditional methods [Di Paolo 96].

The nature of artificial life models makes them suitable for studying emergent phenomena and open-ended evolution, not least because they have the potential for new types of individuals or components to develop within them [Miller 95]. Spatial structure can often be modelled more easily using this approach than by using more traditional theoretical biological methods (see, for example, [Boerlijst & Hogeweg 91], [Collins 92], [Nuño *et al.* 95]).

As should be apparent from the above, the artificial life approach, like more traditional theoretical biological approaches, has both strong and weak points. The synthetic and analytic approaches are complementary, and ideally they can reinforce each other. Further discussion of the relationship between artificial life and theoretical biology can be found in [Collins 92], [Taylor & Jefferson 94], [Bonabeau & Theraulaz 94], [Miller 95], [Roughgarden *et al.* 96] and [Toquenaga & Wade 96].

[8] David Marr's [Marr 82] analysis of complex systems on three levels (the computational theory level, the representation and algorithm level, and the hardware implementation level), and his insistence that the theoretical distinction between these levels should be recognised when devising a model—especially the distinction between *what* is being computed (level 1) and *how* (level 2)—is also relevant for the formulation of scientific artificial life models.

3.2 Previous Work

In this section, a review is presented of some previous artificial life studies. This is not a comprehensive review of the whole field, but focuses on work related to the central topics of this thesis, such as self-reproduction, open-ended evolution and self-maintenance. Coverage of some of the more important contributions to the field that are not discussed here can be found in [Fogel 98].

3.2.1 Self-Reproduction

Nearly all of the practical implementations of artificial worlds to be described are related in some way to the seminal work of John von Neumann. In the late 1940s and early 1950s, he devoted considerable time to the question of how complicated machines could evolve from simple machines.[9] Specifically, he wished to develop a formal description of a system that could support self-reproducing machines which were robust in the sense that they could withstand some types of mutation and pass these mutations on to their offspring. Such machines could therefore participate in a process of evolution.

Inspired by Alan Turing's earlier work on universal computing machines [Turing 36], von Neumann devised an architecture which could fulfil these requirements. The machine he envisaged was composed of three subcomponents [von Neumann 66]:

1. A general *constructive* machine, $\mathbf{A}$, which could read a description $\phi(\mathbf{X})$ of another machine, $\mathbf{X}$, and build a copy of $\mathbf{X}$ from this description:

$$\mathbf{A} + \phi(\mathbf{X}) \rightsquigarrow \mathbf{X} \tag{3.1}$$

(where $+$ indicates a single machine composed of the components to the left and right suitably arranged, and $\rightsquigarrow$ indicates a process of construction.)

2. A general *copying* machine, $\mathbf{B}$, which could copy the instruction tape:

$$\mathbf{B} + \phi(\mathbf{X}) \rightsquigarrow \phi(\mathbf{X}) \tag{3.2}$$

[9] Von Neumann had difficulties in defining precisely what the term 'complicated' meant. He said "I am not thinking about how involved the object is, but how involved its purposive operations are. In this sense, an object is of the highest degree of complexity if it can do very difficult and involved things." [von Neumann 66].

3. A *control* machine, $\mathbf{C}$, which, when combined with $\mathbf{A}$ and $\mathbf{B}$, would first activate $\mathbf{B}$, then $\mathbf{A}$, then link $\mathbf{X}$ to $\phi(\mathbf{X})$ and cut them loose from $(\mathbf{A}+\mathbf{B}+\mathbf{C})$:

$$\mathbf{A}+\mathbf{B}+\mathbf{C}+\phi(\mathbf{X}) \rightsquigarrow \mathbf{X}+\phi(\mathbf{X}) \tag{3.3}$$

Now, if we choose $\mathbf{X}$ to be $(\mathbf{A}+\mathbf{B}+\mathbf{C})$, then the end result is:

$$\mathbf{A}+\mathbf{B}+\mathbf{C}+\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}) \rightsquigarrow \mathbf{A}+\mathbf{B}+\mathbf{C}+\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}) \tag{3.4}$$

This complete machine plus tape, $[\mathbf{A}+\mathbf{B}+\mathbf{C}+\phi(\mathbf{A}+\mathbf{B}+\mathbf{C})]$, is therefore self-reproducing. From the point of view of the evolvability of this architecture, the crucial feature is that we can add the description of an arbitrary additional automaton $\mathbf{D}$ to the input tape. This gives us:

$$\mathbf{A}+\mathbf{B}+\mathbf{C}+\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}) \rightsquigarrow \mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}+\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}) \tag{3.5}$$

Furthermore, notice that if the input tape $\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D})$ is mutated in such a way that the description of automaton $\mathbf{D}$ is changed, but that of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are unaffected—that is, the mutated tape is $\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}')$—then the result of the construction will be:

$$\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}+\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}) \overset{\text{mutation}}{\rightsquigarrow} \mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}'+\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}') \tag{3.6}$$

The reproductive capability of the architecture is therefore robust to some mutations (specifically, those mutations which only affect the description of $\mathbf{D}$), so the machines are able to evolve. Von Neumann pointed out that it was the action of the general copying automaton, $\mathbf{B}$, which gave his architecture the capacity for evolving machines of increased complexity, because $\mathbf{B}$ is able to copy the description of any machine, no matter how complicated [von Neumann 66] (p.121). This ability is clearly demonstrated in Equation 3.5 above.

The original implementation envisaged by von Neumann was a constructive system, which Burks has referred to both as the 'robot model' and as the 'kinematic model' [Aspray & Burks 87] (p.374). However, von Neumann decided that the system was too complicated to capture in a set of rules that were both simple and enlightening, so he turned his attention to developing the cellular automata (CA) framework with Stanislaw Ulam. Von Neumann described the detailed design of a self-reproducing machine in a cellular automata space, according to the architecture described above.[10] Recently, a slightly modified and simplified version of this design was successfully implemented on a computer [Pesavento 95]. One of the major achievements of von Neumann's work was to clarify the logical relationship between *description* (the instruction tape, or genotype), and *construction* (the execution of the instructions to eventually build a new individual, or phenotype) in self-replicating systems. However, as already mentioned and as emphasised recently by Barry McMullin (e.g. [McMullin 92a]), his work was always within the context of self-replicating systems which would also possess great *evolutionary* potential.

Von Neumann concentrated on the *logic* required for a self-replicating machine to be able to evolve increased complication. He therefore did not specifically deal with various biological concerns, most notably concerns of energy. In fact, von Neumann was well aware of these concerns and warned that by restricting attention to the *logic* of self-reproduction, "one has thrown half the problem out the window, and it may be the more important half" [von Neumann 66].[11]The other half of the problem to which he is referring is that of the molecular phenotype [Pattee 95a]. Burks says of the kinematic model that "von Neumann intended to disregard the fuel and energy problem in his first design attempt. He planned to consider it later, perhaps by introducing a battery as an additional elementary part" [Aspray & Burks 87] (p.485). Another major difference between biological organisms and von Neumann's self-replicators is the capacity of the former, but not the latter, for self-maintenance in the face of environmental perturbations. Alvy Ray Smith has pointed out [Smith 91] that some of the CA-based self-reproduction models developed by von Neumann and more recently by others are very non-biological in other ways as well; for example, reproduction in CA models does not occur by development from an 'egg', most models suffer from 'overcrowding' such that an individual self-replicator can only reproduce once (or a small number of times) before it runs out of space in which to place its offspring, etc.

A good review of subsequent work on self-reproduction in CAs is presented in [Sipper 98].

[10] The general constructive machine **A** of this design is often referred to as a 'universal constructor'. However, this term should be used with caution; from the above description of the architecture it is clear that **A** can build any machine **X** *that can be described upon a tape* $\phi(\mathbf{X})$. For cellular automaton models it can be proved that there are some configurations that the universal constructor cannot build (e.g. [Moore 62], [Myhill 63]). These are referred to as 'Garden of Eden' configurations, as the only way they may exist is if they are programmed in as the initial state of the space at time zero.

[11] Quoted from [Pattee 88] (p.69).

Much of this more recent work (e.g. [Langton 84], [Ibáñez *et al.* 95], [Tempesti 95], [Perrier *et al.* 96], [Morita & Imai 97]) has the kind of 'non-biological' character mentioned above. In fact, as pointed out by Barry McMullin, much of this work does not even seem to share von Neumann's concern with the evolution of increased complication, but addresses the 'problem' of self-reproduction in and of itself as if this was all that von Neumann was concerned with [McMullin 92a]. For example, Chris Langton described a much simpler self-replicator in a CA, by dropping von Neumann's requirement for universal construction [Langton 84]. Langton, like von Neumann, asserts that the self-replicators must treat their stored information in two different manners: "*interpreted*, as instructions to be executed (translation), and *uninterpreted*, as data to be copied (transcription)" (*ibid.* p.137, original emphasis), but he only insists on these requirements in order to avoid the problem of 'trivial' self-reproduction[12] rather than from any concerns relating to evolvability. The resulting automaton, referred to as 'Langton's loop', is very fragile in that it cannot in general withstand perturbations and mutations, and is certainly not capable of heritable viable mutation. McMullin describes Langton's work as a "cruel (though of course unintentional) parody" of von Neumann's [McMullin 92a] (p.181). This 'myth' that von Neumann was concerned with self-reproduction in and of itself has by no means been dispelled even now; for example, an introduction to a paper at a recent European Conference on Artificial Life reads "The first steps in [developing a universal theory of evolution] were the cellular automata of von Neumann, *who only studied self-replication.* Among other [*sic*], Langton and Sipper *also* addressed the question of evolution." [de Dinechin 97] (p.560, emphasis added).

As already mentioned, these studies do not generally consider the ability of the automaton to actively maintain its own structure in the face of environmental perturbations. This deficiency has certainly been recognised for a long time (e.g. [Arbib 69] and, more recently, [McMullin 92a]), but very little work has so far been done to create more robust, self-maintaining, self-reproducing CAs (but see Section 3.2.3). Indeed, to anti-

[12] 'Trivial' self-reproduction, at least in the sense being used by Langton, occurs when reproduction of a particular sort of configuration happens purely due to the rules of the system rather than to anything explicitly encoded in the configuration itself. For example, a CA with a transition rule such as "if this cell is empty (in the quiescent state) and one of its neighbouring cells is in state $\mathcal{A}$, then change the state of this cell to $\mathcal{A}$" is an environment in which the state $\mathcal{A}$ trivially self-replicates. I suspect that Langton's work was the seed for the preoccupation of many artificial life researchers over the past decade with the 'problem of trivial self-reproduction'. However, I believe that this is not the most relevant distinction with respect to evolvability, as I will argue in Chapter 7.

cipate the discussion presented in Chapter 7, it may be that artificial life platforms will have to model the material grounding of self-reproducing entities in some way, rather than being purely logical models, if strong selection pressure of self-maintenance is to exist. Some extensions to the cellular automata framework in this direction have been reported, such as the brief description by Myhill of a model in which machine components moved discretely in a two-dimensional environment [Myhill 64]. Von Neumann's original 'kinematic model' also falls into this category [Aspray & Burks 87] (p.374). However, the majority of work 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.

A short time after von Neumann's death in 1957, Lionel Penrose developed a series of mechanical models of self-reproduction, which bear some similarity to von Neumann's kinematic model [Penrose 59]. Around the same time, a couple of other simple physical models of self-reproduction were also reported ([Jacobson 58], [Morowitz 59]), but Penrose provided the most discussion of his work. Furthermore, he also considered a self-reproduction scheme which was more like the relatively simple template-reproduction mechanism of the nucleic acids (DNA and RNA) than von Neumann's full self-reproduction architecture [Penrose 62]. Penrose mentions a number of deficiencies with this model (e.g. it was hard to get fully-formed copies of molecules to unbind at the right time, and there was no consideration of energy requirements), but was able to draw a number of conclusions about the properties required of a molecule which can form precisely self-replicated chains. He identified three necessary properties: (1) the molecules must have good alignment properties; (2) there must be a mechanism for strong permanent bonding down the chain (in order to form the copy); and (3) there must be a mechanism for cross-linkage between the original and the replica (which appeared to Penrose to be necessary to ensure stability). Although Penrose's approach was not particularly rigorous, his conclusions remain one of the few contributions to the 'logic' of template-reproduction to this day. This topic will be discussed further in Chapter 7.

The only other significant contribution to the subject, to my knowledge, is the formal system described by Douglas Hofstadter in his book *Gödel, Escher, Bach.* Hofstadter called the system 'Typogenetics', and explains that he "tried to capture some ideas of molecular genetics in a typographical system" ([Hofstadter 79] p.504). In this system, a strand of letters encodes a sequence of operations (such as copying, cutting etc.) which are to be performed upon the strand itself. Typogenetics was later developed and analysed by Harold Morris in his PhD dissertation [Morris 88], in which he showed that it was possible to produce several types of strand which were capable of self-reproduction. Louis Varetto has also reported work with self-reproducing strands in Typogenetics [Varetto 93]. In the state in which the system is presented, each strand can only inspect itself (i.e. there is no notion of a strand competing for resources with other strands in an environment), and the evolutionary potential of the self-reproducing strand is also unclear. These are both significant deficiencies from the point of view of using the system as a model of biological evolution (Hofstadter again was more interested in the process of self-reproduction in its own right). Modifications could conceivably be made to correct these—indeed, some were suggested by Varetto (*ibid.*)—but I am unaware of any implementations of such modifications.

The reproduction of structures in Penrose's and Hofstadter's models can be seen as the result of a self-inspection process; atomic subcomponents of the structure are inspected and copied, and these copies are then joined together to create a replica of the whole structure. More elaborate schemes for reproduction by self-inspection have been described in [Laing 77] and [Ibáñez *et al.* 95]. The relationship between these architectures and that proposed by von Neumann will be discussed in some detail in Section 7.2.3.

The next significantly different, substantial implementation of a system supporting self-reproducing components was reported by Steen Rasmussen and colleagues in the early 1990s.[13] They conducted a series of experiments using a general approach they named 'Coreworld'[14] (e.g. [Rasmussen *et al.* 90], [Rasmussen *et al.* 91]), inspired by an earlier

[13] Although Eigen and Schuster's work in the 1970s should also be mentioned. In the context of prebiotic evolution, they discussed ways in which the amount of selectively maintainable information in a system could be increased beyond the capacity of an individual replicator (see, for example, [Eigen & Schuster 77]). They introduced the 'hypercycle' (a cyclically catalytic group of replicators) as an functional organisation which could achieve this.

[14] Specific configurations of Coreworld were named Venus I, Venus II, and Luna.

computer game developed by A.K. Dewdney called 'Core Wars' [Dewdney 84].[15] In this approach, a one-dimensional address space is initialised with assembler-level instructions taken from a set of ten (or fewer). A number of execution pointers are distributed through the address space, determining which instructions get executed at any given time. Each address also has a certain 'resource' concentration associated with it, and execution of instructions is also conditional upon sufficient resources being available in the local neighbourhood. Noise is introduced into the system by new instruction pointers being introduced at random with low probability, and also by instructions being 'mutated' into randomly-chosen different instructions in certain circumstances. Although this work was concerned primarily with the emergence of self-organisation and cooperative structures (described in Section 3.2.3), experiments were conducted in which the memory was initialised by inserting a single copy of a hand-written self-reproducing program, and letting it reproduce. However, it was found that the self-reproducers soon started to malfunction and die out due to the noise in the system and to copying themselves on top of each other (but Rasmussen and colleagues still found this a useful method by which to create a biased initial distribution of instructions).

A major advance in the use of computer platforms such as Rasmussen's Coreworld to study the evolution of self-reproducing individuals was made by biologist Tom Ray with his 'Tierra' system ([Ray 91], [Ray 94b]). In his work, Ray places central importance upon the notion of self-replication:

> "Self-replication is critical to synthetic life because without it, the mechanisms of selection must also be pre-determined by the simulator. Such artificial selection can never be as creative as natural selection. The organisms are not free to invent their own fitness functions. Freely evolving creatures will discover means of mutual exploitation and associated implicit fitness functions that we would never think of. Simulations constrained to evolve with pre-defined genes, alleles, and fitness functions are dead-ended, not alive." [Ray 91] (p.372).

Ray's major development was to design an instruction set for his self-replicators which was robust to mutations and therefore evolvable, while at the same time retaining

[15] The ancestry of this approach can be traced back even earlier (e.g. [Bratley & Millo 72], [Burger *et al.* 80]), although these examples were presented in a fairly informal manner.

the property of computational universality.[16] The basic idea is that a virtual operating system is provided, complete with this robust and simple machine language, together with an address space (memory) of fixed size. The robustness of the language is achieved chiefly through the use of relative or template-driven[17] addressing in branching instructions (rather than absolute addressing), and by avoiding the use of explicit memory addresses as operands to instructions. In such an environment, the proportion of functional programs in the space of all possible programs is greatly increased, and the chances of mutating a functional program and coming up with another program that also works are much higher.

An evolutionary run commences with the introduction of an *ancestor* program into the otherwise empty memory. The ancestor is a hand-written self-replicator which produces another copy of itself in the computer's memory when it is run. At each iteration of the system, each program in the computer's memory is allowed to execute a certain number of instructions. By only running a limited number of instructions from each program at each time, the system does not run into the halting problem,[18] which is a potential problem for other kinds of program evolution system. A small element of stochastic behaviour is associated with the execution of the machine instructions, e.g. an ADD instruction which usually adds one to its operand may occasionally add zero or two instead, or a COPY instruction may sometimes mutate a byte of the data it is copying. The programs are also subject to point random mutations at a given low rate. In either of these ways, as a run proceeds variations of the ancestor program begin to appear. If a variation retains the ability to produce a copy of itself, then it too may be retained in the population of programs over a number of generations. As the available memory begins to fill, a 'reaper' operation is performed to kill off a number of the programs. Programs which perform operations which cause their flag to be set[19] are killed off quicker than others (by being advanced up the 'reaper queue'), but otherwise the order in which programs are killed off is largely determined by their age.

---

[16] A proof that the language has this property is presented in [Maley 94].

[17] With the template-driven branching scheme, a branching instruction is followed by a template (a sequence of bits). The operating system will search for the nearest matching template in the rest of the code and move the instruction pointer to that position.

[18] That is, some programs might never terminate.

[19] Examples include issuing a `jmp` instruction with a template pattern for which no match can be found, and attempting to write to a memory address for which the program does not have write access.

A number of interesting results have been obtained from such evolutionary runs, for example the appearance of "parasites"—short pieces of code which run another program's copying procedure in order to copy themselves. Hyper-parasites (parasites of parasites) have also been observed, along with a number of other interesting ecological phenomena [Ray 91].

It may be true to say that most of the interesting behaviour that has been seen to evolve has done so because facilities for these behaviours were engineered into the original language specification. For example, the fact that parasites emerge is a little less surprising when the mechanisms of template-driven branching are considered. I will return to this issue in Chapter 7. However, what these studies have demonstrated is that it *is* possible to build a robust operating system in which non-trivial self-replicating computer code can evolve.

The original goal of Tierra was to "parallel the second major event in the history of life, the origin of diversity" [Ray 91] (p.373). The event being referred to is known as the *Cambrian explosion*, and marks the first and sudden appearance in the fossil record, some 550 million years ago, of a great diversity of macroscopic multicellular organisms. Ray explains that "rather than attempting to create prebiotic conditions from which life may emerge, this approach involves engineering over the early history of life to design complex evolvable organisms, and then attempting to create the conditions that will set off a spontaneous evolutionary process of increasing diversity and complexity of organisms" (*ibid.*).

Ray admits that his early experiments with Tierra didn't actually achieve this goal, and suggests that the system is better viewed as "an artificial world which may roughly parallel the RNA world of self-replicated molecules (still falling far short of the Cambrian explosion)" (*ibid.*).

In analysing the broad patterns that emerge from evolutionary runs on Tierra, Ray notes that the observed evolutionary innovations fall into two broad categories: 'ecological solutions' and 'optimisations' [Ray 97]. The former include adaptations involving other programs in the environment (e.g. parasitism, immunity, etc.), and the latter are improvements within an individual organism to make the reproduction algorithm more efficient. Some experiments in which individual programs are allowed to create parallel processes have been reported ([Thearling & Ray 94], [Thearling 94], [Thearling & Ray 96]), and in these cases optimisations include the use of parallelisation to increase the speed of reproduction (but only when the system was inoculated with a two-process parallel ancestor). Parallelisation is achieved by each process owning its own instruction pointer, but these all refer to a single copy of the program's code. Speed-up is achieved by each parallel process copying a different section of this code to create an offspring.

More recently, Ray and his colleagues have been striving towards evolving *differentiated* parallel programs, where some of the processes may be performing other tasks (e.g. sensory), rather than all being dedicated to reproduction. These attempts have so far been largely unsuccessful, although the most recent milestone has been to create an environment in which a hand-written differentiated ancestor (comprising 10 parallel processes, two of which are devoted to reproduction, and the remaining eight to sensing the environment) is at least maintained over prolonged periods of evolution, rather than being degraded into a purely reproductive organism [Ray & Hart 98]. However, this partial success has only been achieved by adding some high-level operations to the system, such as insertion, deletion and crossover operators, and new forms of mutation. Ray writes that the inclusion of these operators represents "a change in the philosophy towards this kind of genetic operation. These operations are being imposed by the Tierra operating system, and are not under the control of the creatures themselves. The reason for the shift is that the primary goal and focus of the work is now the attainment of complexity increase. This is a very big, and probably very difficult objective. For this reason I am willing to try anything that might aid in achieving the goal. Among the rewards of achieving the goal, should be creatures that do many more things for themselves, which should make the imposed genetic operators seem to be a fairly trivial sin."[20]

Comparative studies using different instruction sets have also been reported [Ray 94a]. The results suggest that the evolvability of the system is very sensitive to the underlying genetic system (i.e. small changes to the instruction set lead to very different results).

A number of similar systems have been produced by other researchers. These in-

[20] From an online report on Tierra, available at `http://www.hip.atr.co.jp/~ray/tierra/netreport/netreport.html#Philosophy`.

clude 'Avida', developed by Chris Adami and Titus Brown [Adami & Brown 94], and 'Computer Zoo', written by Jakob Skipper [Skipper 92]. Programs in both of these systems live in a two-dimensional environment. This notion was introduced mainly to provide a form of genetic isolation (to promote heterogeneity), and also to facilitate efficient implementation on parallel hardware. Although the details of these systems differ somewhat, the general mode of operation is the same. Some attempts have also been made to extend the representation of individual programs into two dimensions (i.e. each instruction of a program is physically located at a unique position on a 2D grid), but these have generally proved too brittle to support prolonged evolution (e.g. [Davidge 92], [Maley 93], [de Dinechin 97]).

John Koza has proposed a system of self-replicating programs using a slightly different approach [Koza 94]. Koza's programs are boolean trees whose nodes may be standard logical functions (specifically, AND, OR and NOT) or a number of extra functions which not only produce a boolean result but also have side effects which act upon the program itself or upon other programs in the computer's memory. For example, there is an ADD function which will search the rest of the memory for a program matching a given template, and, if one is found, will have the side effect of substituting this new program in the position where the ADD instruction resided in the original program.

One result of this approach of using boolean functions with side effects is that a program may perform a boolean calculation as well as having the side effect of producing another copy of itself somewhere else in memory. Koza encouraged his programs to perform specific boolean calculations by imposing a fitness function on the system—a program has a good chance of proceeding to the next generation only if it is able to self-replicate *and* performs well on the target boolean function.

In this respect, Koza's system is very similar to Avida, where programs can gain more CPU-time by successfully completing user-specified tasks. In this important sense, Avida and Koza's system differ from Tierra and the other related platforms; they have an *externally specified* function (or set of functions) which the programs are encouraged to perform *as well as* having to be able to reproduce. In this way, there are effectively two fitness measures, the former being completely unrelated to a program's ability to self-replicate.

Although this technique could conceivably be used to evolve programs which perform useful tasks (although it has not been demonstrated that it can scale to larger problems), it restricts the capacity of the system for open-ended evolution. By concentrating on applying selection pressure through the abiotic environment (i.e. by the externally defined functions), the programs are all evolving towards solving the same static problem rather than facing an indefinite variety of environments. Although biological organisms are themselves constrained by their abiotic environment, it is widely agreed (see Chapter 2) that the selection pressure for the most impressive forms of open-ended evolution (such as evolutionary arms races, symbiogenesis, and sexual selection) comes from the ecological relationships *between* organisms (i.e. the biotic environment). It is likely that only those systems in which organisms are able to *interact* with each other, and in so doing to directly affect each other's chances of survival, will display truly open-ended evolution. This issue will be discussed further in Chapter 7.

3.2.2 Open-Ended Evolution

The work reviewed in the previous section focussed on self-reproduction. This is clearly related to the topic of open-ended evolution (as defined in Section 2.5), and indeed that was also a major concern to many of the researchers mentioned, such as von Neumann and Ray. In this section, other important contributions to the topic of open-ended evolution will be reviewed. In many cases, the decision to review the work in this section rather than in the previous one is fairly arbitrary, and is based largely upon the relative emphasis given to the two topics by the authors of the work.

Nils Aall Barricelli was, to my knowledge, the first person to actually run artificial evolution experiments on computers [Barricelli 57], [Barricelli 62], [Barricelli 63]. He conducted a series of experiments, starting in 1953, to investigate the evolution of life (and not, as many others after him, purely to investigate the use of evolution as an optimisation technique). Barricelli explicitly discussed in his work what it would take in addition to reproduction and random variation in order for us to consider his evolved organisms 'alive'. In fact, he decided that the term 'life' was too poorly defined to be of use in the context, as mentioned in Section 2.1.3. Barricelli introduced the concept of symbiogenesis[21] in his work as an additional requirement for his organisms (which he

[21] According to Barricelli, the symbiogenesis theory claims that if genes are to evolve into "relatively

then called 'symbioorganisms'), and, instead of asking whether these symbioorganisms were alive, asked the reciprocal question of "whether the objects we are used to call[ing] living beings are a particular class of symbioorganisms" [Barricelli 63] (p.7).

The basic model used by Barricelli can be considered as a one-dimensional cellular automaton, where each state persists from one time step to the next depending upon the state of other cells in certain neighbouring positions. In this way, cooperative configurations of states can arise. Among the phenomena that Barricelli observed in this system are: self-reproduction of certain collections of states (i.e. symbioorganisms); crossing of material between two symbioorganisms; spontaneous formation of symbioorganisms; parasitism; self-maintaining symbioorganisms; and, evolution [Barricelli 62]. Barricelli's method and results are similar in a number of ways to those of many of the more recent studies described in the previous section, such as Rasmussen's and Ray's. The fact that these experiments were run using the computing equipment available in the 1950s makes them all the more impressive. It is highly regrettable that Barricelli's work has been largely forgotten or ignored by today's artificial life community,[22] although there are a few signs that it is at last beginning to achieve some recognition (e.g. [Dyson 97], [Fogel 98]). Although many might argue over the central role that Barricelli gave to the process of symbiogenesis in his work, his experiments are strengthened by the fact that he at least *proposed* an explicit list of the assumptions behind the model, and also by the simplicity of the model he developed. In these respects, his work is of more scientific value than much of the more recent work on these subjects.

In addition, Barricelli conducted some experiments in which the individual symbioorganisms were also decoded into a strategy for playing a simple game (i.e. they had a phenotype) [Barricelli 63]. When two symbioorganisms were competing to reproduce into the same space, they played the game according to their individual strategies, and the winner was allowed to reproduce. This is similar in some ways to the dual selection pressures in Avida and Koza's work reported above, but has the advantage (from the point of view of open-ended evolution) that the additional selection pressure is de-

higher forms of life", they must only be able to reproduce through a symbiotic relationship with other genes [Barricelli 57] (p.145). A group of genes that collectively reproduces in this way (a 'symbioorganism') can be considered a special case of a hypercycle [Eigen & Schuster 77], in which each component in the group is absolutely *required* for the reproduction of the next component, rather than just acting as a catalyst.

[22] His papers contained many ideas on subjects which have since become popular research areas, such as the use of real DNA for performing computations ([Barricelli 63] pp.121–122).

termined by other organisms within the system, rather than being externally defined. Barricelli met with mixed results in these experiments ([Barricelli 63], [Barricelli 72]), partly because symbioorganisms which had developed good strategies for playing the game often became infected by parasites. I suspect that Barricelli's system was more open to these types of phenomena than are systems such as Avida, because the individual cells within a symbioorganism reproduce by themselves rather than *en masse*. As such, they behave more as individually 'selfish genes' [Dawkins 76], and the symbioorganisms as a whole should therefore not necessarily be expected to be good at finding 'optimal' solutions to given problems.

There has been some more recent work within the artificial life community on the subject of symbiogenesis. Some of the issues involved in incorporating symbiogenesis (and symbioses in general) in artificial life models are discussed in [Daida *et al.* 96], and studies of the general conditions under which symbioses may occur are reported in [Bull & Fogarty 95]. Some practical implementations, incorporating the ideas of symbiogenesis, have been successful at solving particular problems (for example, [Ikegami & Kaneko 90], [Numaoka 95]), but as organisms in these systems have a limited behavioural repertoire and are solving a specific, externally defined task, their capacity for truly open-ended evolution is restricted.

Gordon Pask also describes some early work with artificial life models [Pask 69], although the evolutionary potential of his automata is limited by the rather small set of actions which they may perform. A rather more interesting system was developed by Michael Conrad and Howard Pattee [Conrad & Pattee 70]. In this model, individual organisms, with a genotype representation and a phenotype obtained by interpreting the genotype as instructions (cf. Barricelli's game-playing symbioorganisms), compete in a one-dimensional world for the possession of 'chips' which they use for self-repair and reproduction. It is closely related in many ways to Ray's later Tierra model, but with a smaller number of instructions representing a limited set of possible interactions between organisms, rather than a computationally complete instruction set as in Tierra (although even in Tierra there is still only a limited number of types of interaction between organisms). Also, it has a notion of conservation of matter, lacking in Tierra, to model ecosystem interactions. Having said this, there is only one type of matter in the model (a 'chip'), and it has a fairly arbitrary connection to the structure of an organism. For example, an organism's genome is represented as a string of 'states' rather than a string of matter—an organism's store of chips is only used to determine when it can repair itself and when it can reproduce. The major consequence of disassociating the structure of the organism from the 'matter' in the world is that the structure must therefore be predefined and is not able to evolve, whereas, had it been embedded in the material world, new organism structures could emerge from new organisations of the matter. This problem of predefining a non-material structure for organisms is shared by Tierra and other models described in this and the previous section, and will be discussed further in Chapter 7. The organisms in Conrad and Pattee's model had the potential to engage in symbiotic relationships with other organisms (by sharing chips), and also to reproduce sexually. The results showed that symbiosis was often selected for, but sexuality generally was not. However, in summarising the results, the authors conclude that:

> "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 ... One point is clear, that the processes of variation and natural selection alone, even when embedded in the context of an ecosystem, are not necessarily sufficient to produce an evolution process ... Experience with the present model re-enforces our feelings that 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." [Conrad & Pattee 70] (pp.407–409).

Many people claim that these processes remain outside the scope of artificial life models even today (e.g. [Stewart 97], [Bedau 98a]).[23]

In the mid-1970s, John Holland proposed a collection of models he collectively referred to as the 'α-Universes' as a suitable environment in which to study the spontaneous emergence of self-reproducing systems [Holland 76]. The design is similar in many ways to Conrad and Pattee's model, but was influenced by an analogy with the spontaneous emergence of life in a 'primordial soup' of biochemical molecules. A feature shared

[23] But see [Mayer & Rasmussen 98] for a recent study in which emergent hierarchical structure has been successfully simulated.

by both models (and by some of Barricelli's work) is that individual components are acted upon by certain predefined operators (the 'physics' of the world), but (adjacent collections of) components can also be interpreted as encodings of further operators (phenotypes, or 'emergent operators' in Holland's terminology), giving the models a more open-ended quality. Also, as in Rasmussen and colleagues' work discussed in the previous section, the configurations which are reproduced in the α-Universes are not individual structures, but *collections* of structures. In Barry McMullin's detailed analysis of the α-Universes [McMullin 92a], he suggests that these collective structures might alternatively be regarded as autopoietic (roughly, self-producing and self-maintaining) organisations. However, McMullin observes that "the 'higher-level', properly autonomous entities, are not, in general, self-reproducing in any sense, and are *certainly* not genetically self-reproducing in the von Neumann sense of permitting an open-ended growth in complexity" [McMullin 92a] (p.269, original emphasis).

Holland's original work was based upon a mathematical analysis of phenomena that he expected to emerge in the system. Fifteen years later, the α-Universes were implemented as a computer program by McMullin ([McMullin 92a], [McMullin 92b]). He found a number of problems with the design that were not anticipated by Holland and which meant that it did not produce the 'life-like' behaviour that he postulated. As many of the problems were ultimately due to components in the world being unable to control their local environment and maintain their own structure, McMullin has subsequently gone on to investigate software implementations of autopoiesis [McMullin & Varela 97].[24]

After the α-Universes, Holland developed the 'Echo' model of complex adaptive systems ([Holland 95], [Hraber *et al.* 97]). Echo places more emphasis on ecological interactions and exchange of resources than do most of the other models reviewed. In particular, in Echo Holland takes the view that it is the 'market' that emerges from exchanges of resources between individual agents that is the source of much of the interesting behaviour of a complex system. Individual agents are modelled at a fairly high level, with a predefined structure. In the basic model, agents can participate in a limited set of interactions with other agents (e.g. to exchange resources), and reproduce auto-

[24] Interestingly, Barricelli also predicted that his symbioorganisms would need to develop means of controlling their local environment (such as a membrane) if they were to evolve past a certain level of complexity [Barricelli 63] (pp.122–124).

matically when they have collected sufficient raw materials [Holland 95]. Interactions are governed by each agent's collection of 'tags'—short strings of symbols which are encoded on the agent's genome (or 'chromosome' in the Echo terminology). Each tag is used for a specific set of interactions, where the outcome of the interaction is partially determined by comparing corresponding tags in the interacting agents in some predefined way. Various extensions to this basic model are also described, to add features such as multicellularity, metabolism and selective mating (*ibid.*). The design is based upon a core set of principles which Holland believes are common to all complex adaptive systems. One feature that distinguishes Echo from other models reviewed in this chapter is that it has been successfully used for several ecological studies (e.g. [Schmitz & Booth 96], [Hraber & Milne 97]).

I have now discussed what I consider to be the most important practical examples of artificial life models aimed at self-reproduction and open-ended evolution. To end this section, I will briefly mention a number of other studies which have also made relevant contributions.

In my discussion of Avida and Koza's system in the previous section, I made the distinction between selection pressure that originates from other programs (organisms) within the environment, and that which originates from externally defined functions. This distinction was emphasised by Norman Packard, who used the terms *intrinsic* and *extrinsic* adaptation respectively [Packard 88]. He argues (as have many others mentioned in this section and the last) that models with intrinsic adaptation are more appropriate for modelling biological evolution. Packard described a model which he used to study evolutionary dynamics, which is distinguished by its simplicity. He says "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 behavior" (*ibid.* p.142). However, as his organisms only have two genes, the relevance of the model in the current context of studying open-ended evolution is limited.

Larry Yaeger has described a system called PolyWorld [Yaeger 94], which is in many ways the antithesis of Packard's minimal approach. PolyWorld models many features of biological life, such as a simple 'metabolism', a nervous system and vision. Yaeger describes it as an attempt to evolve Artificial Intelligence through the evolution of nervous systems in an ecology. In PolyWorld, organisms controlled by (genetically determined) neural networks move around a two-dimensional environment, collecting energy, fighting and mating. Yaeger's model is one of the very few artificial worlds in which distinct species of organisms have evolved and coexisted.[25] Unfortunately, the results of PolyWorld have not been analysed in sufficient detail to enable many useful scientific conclusions to be drawn from it, and the complexity of the model does not help in this respect.

Early attempts at allowing variable length genomes in an artificial life simulation are described by Robert Collins in his PhD thesis [Collins 92], which extends previous work by David Jefferson, Charles Taylor and colleagues (e.g. [Taylor *et al.* 88], [Jefferson *et al.* 91]). In this work, an organism's genome encodes a neural network which controls its behaviour. By allowing variable length genomes, larger networks are able to evolve, which are capable of producing more complex behaviour. Along the same lines, Inman Harvey has worked on extending the theory and the design of the standard genetic algorithm to allow open-ended evolution by permitting the length of genomes to increase over time [Harvey 93]. The approach is called SAGA, and, like Collins *et al.*'s work, is concerned with extrinsic adaptation (i.e. situations where an external fitness function is applied). This method has proved useful for a number of practical purposes, mostly concerned with evolutionary robotics, but it is not clear how relevant it is for many of the systems I have described in the last two sections. I suspect that some of the assumptions upon which Harvey based his analysis of the need to extend the 'schema theorem' of genetic algorithms for open-ended evolution (*ibid.* Chapter 6) might not be valid for these models. In particular, it is not clear how the notion of fitness used in the analysis can be related to models with intrinsic adaptation. Furthermore, Harvey's analysis assumed a system with low epistasis, and it is questionable whether models such as Tierra meet this requirement.

Finally, it is worth noting that some of the most spectacular examples of artificial evolution that have been produced to date model *co*-evolutionary processes of one form or another (e.g. [Hillis 90], [Sims 94a], [Sims 94b], [Miller & Cliff 94], [Cliff & Miller 96], [Floreano *et al.* 98], [Nolfi & Floreano 98]). In these studies, the success of organisms in one population depends upon the success of organisms in another, coevolving population. However, these studies have all been geared towards producing organisms which

[25] Another is Lindgren's model of a population of individuals playing a variation of the iterated Prisoner's Dilemma [Lindgren 91].

are good at performing a particular task. To this end, the coevolving organisms are still generally competing in some pre-specified (extrinsically defined) game, and they are not given the potential for truly open-ended evolution in which they could develop entirely *new* games to play.

3.2.3 Self-Organisation and Origin of Life Models

In this final review section, I will briefly mention some work on topics such as self-organisation, the origin of life, self-maintenance and autopoiesis. This work is not so relevant to the experimental portion of the thesis reported in Chapters 4–6, but we will come back to it in Chapter 7.

In contrast to the usual procedure for Tierra-like systems of inoculating the environment with hand-written self-reproducing programs, some models have been modified to look at the *spontaneous emergence* of self-reproducing individuals (e.g. [Koza 94], [Pargellis 96]). Similar work has also been reported within the framework of a cellular automata model [Chou & Reggia 97]. However, the details of these models are so far removed from anything in the real world that I doubt that they can really tell us anything of scientific value about the origin of life.

Walter Fontana and colleagues have developed a fundamentally more interesting approach with their 'artificial chemistry' models (e.g. [Fontana 91], [Fontana *et al.* 94], [Fontana & Buss 96]). They argue that a formalism is needed in biology (and other areas) for constructive systems (i.e. those where the components are objects whose structure can change as the result of interactions). This should be coupled with classical dynamic systems approaches to form a *constructive dynamic systems theory.* Such a theory would help us to understand dynamic systems in which new operators can emerge as the system evolves. Some of their latest work in developing such a theory is described in [Fontana & Buss 96]. The relevance of this to the study of open-ended evolution is clear.

Rasmussen and his colleagues have described a similar approach to studying self-organisation ([Rasmussen *et al.* 90], [Rasmussen *et al.* 91]), but based upon a formalism similar to the von Neumann machine (as described in Section 3.2.1) rather than the λ-calculus and other formalisms used by Fontana and Buss.[26] They successfully obtained emergent cooperative structures, but emphasised that the details of such structures depended heavily on the details of the system. However, they do suggest that a number of more general conclusions may be drawn, which may have analogies in prebiotic chemical systems [Rasmussen *et al.* 91] (pp.245–246). These include: that "there are certain relations, which need to be fulfilled, between system size, available executions per system update, and initial conditions before the systems are able to support complex cooperative dynamics"; that "functional stability to perturbations is a *product* of evolution and not a property of the details of the underlying programmable matter"; that "cooperation emerges as a natural property of the functional dynamics in systems with constructive dynamics"; that "simplifying the instruction set below a certain level of complexity inhibits the emergence of higher-order cooperative structures";[27] that "it seems easier to create a reproductive system without genes. The emerging cooperative structures have several properties in common with autocatalytic sets found for catalyzed cleavage-condensation reactions in polymer systems"; and that "the more low-level the living process is, the more fuzzy the organism-environment distinction appears".

The approach of Rasmussen *et al.* can be related to work by Kauffman, Bagley, Farmer and colleagues on artificial chemistries (e.g. [Kauffman 86], [Bagley & Farmer 91]). However, Fontana and Buss claim that their approach (mentioned previously) is more general than these, because it is based upon a *theory of object construction* (e.g. [Fontana & Buss 96], Section 3). Other recent approaches to building constructive dynamical systems include those suggested by Wolfgang Banzhaf, Peter Dittrich and colleagues (e.g. [Banzhaf 94], [Dittrich & Banzhaf 98]), by Shinichiro Yoshii and colleagues (e.g. [Yoshii *et al.* 98a], [Yoshii *et al.* 98b]), and by Yamamoto and Kaneko [Yamamoto & Kaneko 97]. Banzhaf, Dittrich and colleagues, for example, have described experiments with a catalytic self-organising reaction system of binary strings. Their most recent work involves the decoding of these strings as programs which determine how one string reacts with another, which is a similar concept to the idea of emergent operators in Holland's α-Universes.

[26] Rasmussen *et al.* also discuss the relationship between these and other universal formalisms [Rasmussen *et al.* 91] (pp.243–244).

[27] This is similar to von Neumann's conclusion that "complication is degenerative below a certain minimum level" [von Neumann 49] (p.482).

As a general comment, most of these models do not have any spatial structure (the reactions occur in a well-stirred tank), which might have important consequences for self-organising systems. For example, with no spatial structure there can be no notion of *individuality* in the organisations which emerge. Also, the models do not necessarily have conservation of matter. However, these are not fundamental limitations, and the models could be modified fairly straightforwardly. Indeed, such modifications have been discussed by some of these authors themselves (e.g. [Fontana & Buss 96], [Dittrich & Banzhaf 97]).

Finally, a handful of computer models of autopoiesis have been described (for example, [Varela *et al.* 74], [Zeleny 77], [McMullin & Varela 97]), but these have so far met with limited success at achieving sustained autopoietic organisation.

3.3 Methodology and Design Issues for Artificial Life Platforms

In this final section of the chapter, I will attempt to pull together some of the lessons learned in the work reported in the previous sections by highlighting some of the important methodological and design issues involved in creating an artificial life platform.

3.3.1 Methodology

Some of the more important methodological issues concerning the scientific use of artificial life techniques were discussed in Section 3.1.3. The key point is that simulations should be based upon explicit theories and assumptions if they are to be of scientific value ([Pattee 88], [Noble 97], [Taylor 98]).

As as example, the emergence of parasites and similar phenomena in Tierra [Ray 91] might appear to be suggestive of a close parallel between this system and biological evolution. However, the appearance of parasites in Tierra turns out to be dependent on some fairly specific aspects of the system's design, rather than on any particular general principles.[28] Furthermore, closer analysis reveals that it only requires a *single mutation* of the original ancestor program to produce a parasite; this was reported in Ray's

[28] More precisely, there *may* be some general principles concerning parasitism that Tierra shares with Nature, but the point is that no-one has explicitly stated what these might be, and Tierra was therefore not designed to be a particularly good test of any such principles.

original paper [Ray 91] (and has recently been reiterated in [Channon & Damper 98]), but has been overlooked in most subsequent reports of parasitism in Tierra. These facts suggest, if nothing else, that claims concerning the relationship between artificial life models and the real world should be treated with caution.

I think that Tierra and many (but certainly not all) of the other recent contributions to artificial life can be regarded as constituting an 'exploratory stage' of the subject, in which the potential of the new techniques has been investigated. This is a useful stage, and indeed is a normal aspect of any experimental science [Cohen 95]. However, further progress towards scientific knowledge must be based upon the development and testing of properly formulated theories.

3.3.2 Self-Reproduction

Most of the work reported in this chapter has relied upon concepts of self-reproduction in one form or another. However, different studies have modelled self-reproduction in different ways and at different levels. Ray has insisted on the importance of self-reproduction in artificial life models of evolution (see Section 3.2.1), but von Neumann's analysis shows us that a variety of different issues are involved in this concept (e.g. the method by which reproduction is achieved, the robustness of the reproducers to mutations, the heritability of variations, etc.). There is a risk of confusion if one talks simply about self-reproduction, without referring to these more specific issues.

A great deal more work is required before we can properly understand the evolutionary consequences of design decisions relating to each of these more specific issues. Von Neumann's architecture was designed specifically so that the self-reproducing automata would have the potential of participating in an evolutionary process in which more and more complicated automata arose. I am not aware of such a detailed analysis of other sorts of reproduction, although it has been suggested that collectively-autocatalytic sets generally do *not* have great evolutionary potential (see, for example, [Maynard Smith & Szathmáry 95]). It has also been suggested that reproduction by self-inspection (discussed on p.51) has less evolutionary potential than genetic reproduction (e.g. [von Neumann 66] pp.121–123, [McMullin 92a] pp.191–193), although little has actually been proved.

For truly open-ended evolution, we might also want to consider how one sort of reproduction might *evolve* from another sort.[29] I will return to discuss these issues in much more detail in Chapter 7, with the benefit of experience gained from running an extensive series of experiments with a Tierra-like platform (reported in Chapters 4–6). The main point I wish to make here is that the evolvability of the system clearly does depend upon the type of reproduction employed, and upon other issues involved in the concept of self-reproduction, even if we do not yet fully understand these dependencies.

3.3.3 Representation

In addition to consideration of the general scheme of reproduction, we also have to tackle issues concerning the representation that should be used for the reproducers and for the environment in which they exist. Rasmussen and colleagues have presented a useful survey of different universal formalisms which may be employed, and point out that a spectrum of other formalisms exist in between them [Rasmussen *et al.* 91].

There is also a choice between procedural and functional representations. Tierra and related systems have used procedural languages with considerable success, although some have argued that functional languages have many desirable properties (see, for example, [Fontana 91]).

Another issue concerns the level at which the model is pitched. Von Neumann argued that a level must be found which is neither too high nor too low:

> "If you choose to define as elementary objects things which are analogous to whole living organisms, then you have obviously killed the problem, because you would have to attribute to these parts those functions of the living organism which you would like to describe or to understand ... One also loses the problem by defining the parts too small, for instance, by insisting that nothing larger than a single molecule, single atom, or single elementary particle will rate as a part. In this case one would probably get completely bogged down in questions which, while very important and interesting, are entirely anterior to our problem." [von Neumann 49] (p.479).

[29] Maynard Smith and Szathmáry discuss ways in which the evolutionary potential of hypercycles can be enhanced through compartmentation, and suggest that this might eventually lead to the evolution of a genetic system [Maynard Smith & Szathmáry 95].

Similarly, Yaeger points out that the computational power that researchers currently have readily available is not sufficient to enable one to start off at the level of subatomic particles and expect to observe ethological behaviours [Yaeger 94] (p.268). As an example of what is presently achievable, Bernd Mayer and Steen Rasmussen have recently described a system in which emergent third-order structures (micelles) appeared from organised second-order structures (polymers), which in turn appeared from organised first-order structures (monomers)—the monomers being the fundamental level of representation in the simulation [Mayer & Rasmussen 98]. This work (which employed a two-dimensional lattice world of dimensions 100×100 units) pushes currently-available computational power to the limit. Finally, Harvey argues that the level of representation should be as low as possible, so as not to introduce the prejudices of the designer or to restrict the evolutionary potential of the system [Harvey 93] (p.48).

The representation of the environment is another issue, and one which has received little attention so far in the artificial life literature. Organisms in Tierra, the α-Universes and many other systems described in this chapter exist in a one-dimensional world; von Neumann's self-reproducing automata, Avida and others are two-dimensional, and several people have tried modelling three dimensions. An important point to note is that there are qualitative differences in the properties of spaces of different dimensionality (e.g. concerning random walks, [Feller 67] pp.359–363). Von Neumann suspected that three dimensions, or a Riemann surface (a multiply-connected plane), might be required for his kinematic model of self-reproduction, although he did not prove it [von Neumann 49] (p.485). Whether or not this is true, though, it is clear that the kinds of phenomena that might evolve in an artificial life model will be constrained by the dimensionality of the model.

More broadly, the basic laws governing components in the system and how they interact, and the spatial structure of the system, will all contribute crucially to how the components evolve. The general idea that more complicated environments lead to more complicated organisms is fairly widely accepted, but we are really only just beginning to investigate the dependencies which exist between these fundamental aspects of an evolving system.

Chapter 4

Design Details of the Cosmos Platform

Despite the potential problems associated with Tierra and similar platforms discussed in the previous chapter, it was decided that it would be worthwhile to develop and investigate a system built according to the same principles. Reasons for doing this included the following:

- To develop a better appreciation of the issues involved in designing such a system.
- To enable an extensive analysis of the behaviour of such a system to be conducted and reported.
- To observe possible differences in behaviour between the new system and existing Tierra-like platforms, arising from differences in design.
- To get a better idea of how serious some of the potential problems mentioned in the previous chapter might be, and how they might be resolved.

Issues such as these are relevant because the approach pioneered by Tom Ray with Tierra is fairly widely used, and its validity is fairly widely accepted, within the artificial life community. A new Tierra-like platform, called Cosmos,[1] was therefore developed. The design details and relevant implementational issues are discussed in the rest of this chapter.

[1] 'Cosmos' is an acronym for COmpetitive Self-replicating Multicellular Organisms in Software.

4.1 Cosmos Design Philosophy

The basic approach employed in Cosmos to model an evolutionary process is the same as in Tierra. However, many of the design details are different, reflecting the slightly different goals motivating the two systems. One of the original goals of Cosmos was that it should be able to support self-replicating programs with some of the features possessed by simple *cellular* biological organisms, such as mechanisms for communication and response to environmental stimuli (which may potentially promote coevolution between organisms), and mechanisms for regulating the genome (which may promote the evolution of differentiated programs).

Before continuing, clarification should be given of some of the terms that will be used when describing Cosmos. Biological terms will often be used, as these tend to be somewhat more concise than the associated terms relating to computer architectures. While these biological terms suggest the analogy that was in mind when Cosmos was designed, the analogies are certainly not exact; many simplifications and modifications obviously have to be made when designing such a system. With this is mind, the meanings attached to some biological terms in the present context are listed in Table 4.1.

Term	Meaning in context of Cosmos
Genotype	The instructions that make up a program (the host code within a cell).
Genome	The structure within a program which stores the program's instructions. In the current context, the terms genome and genotype are used more or less interchangeably.
Phenotype	The action (behaviour) of a program as its instructions are being executed.
Organism	A single program, which may be unicellular or multicellular.
Cell	A single process in an organism. This term encompasses the host code and any foreign code that may be present, together with associated working memory, buffers, registers and other structures.
Unicellular	An organism containing a single cell/process (in other words, a serial program).
Multicellular	An organism containing multiple cells/processes (in other words, a parallel program).

Table 4.1: Definitions of Biologically-Related Terms Used for Describing Cosmos.

Perhaps the most significant difference between Cosmos and Tierra is that programs in Cosmos cannot directly read the code of their neighbours. Cells can only communicate with each other (within or between organisms) by message passing (described in Sections 4.3.7 and 4.6.1). Apart from this intercellular communication, each cell only has read, write and execute access within its own cell boundary.

Among the other important differences between Cosmos and Tierra are a number of features in Cosmos intended to encourage the evolution of diversity and complexity[2] in the competing programs, rather than just the optimisation of their ancestral algorithms. The most important of these are the energy token allocation system, described in Sections 4.3.5 and 4.5.2, and the regulator system of *promoters* and *repressors* which governs the execution of a program's code, described in Section 4.3.3. The regulator system is closely linked to the programming language in which the self-replicators are written, introduced in Section 4.4. Further differences between Cosmos and Tierra are discussed in Section 4.10.

4.2 Preliminary Issues

Before going into the details of program representation and behaviour, a few words should be said about some general features of Cosmos.

4.2.1 Representation of Information

The underlying representation of many of the components of the Cosmos system is the BitString. Four different types of BitString are used: BitStrings, InfoStrings, WritableInfoStrings and EnvironmentalInfoStrings. These are defined as follows:

BitString A vector of binary digits (i.e. a string of 0s and 1s).

InfoString Like a basic BitString, but also has a *type* associated with it (an integer i in the range $0 \leq i \leq 15$), and a pointer to the current read/write position along the string. A string of bits belonging to an InfoString cannot usually be altered after its initial creation—it can only be read. The only exception is that an InfoString may be *mutated*, which entails one or more of its its being flipped at random.

WritableInfoString An InfoString in which the bit string can be written to as well as read from.

[2] Primarily, hierarchical object complexity (see Section 2.3.2). In other words, many of the design features of Cosmos were intended to promote the evolution of multicellular organisms from unicellular ones.

EnvironmentalInfoString An InfoString that has an *intensity* (a non-negative real valued number) associated with it.

4.2.2 Spatial Structure

The shared space in which the organisms reside is a two-dimensional grid, divided into discrete squares.[3] Each cell in the population is associated with a particular square at any given time. This environment can be configured to wrap around, or not to wrap around, in each dimension. More information about the environment in which the cells live is given in Section 4.5.

4.2.3 Time Slicing and the Top-Level Algorithm

The Cosmos operating system simulates the parallel execution of a large number of programs. As Cosmos is actually implemented on a serial machine, a form of time slicing is required to achieve this (i.e. at each time slice, a small number of instructions are executed for each program, one at a time). The top-level algorithm that implements this procedure is described in Section 4.7. At each time slice, it must be decided how many instructions are to be executed for each program. Possibly the most obvious strategy is to execute a fixed number of instructions for each program. However, from an evolutionary point of view, this would introduce selection pressure for small programs because, all else being equal, longer programs would take a larger number of time slices to reproduce. This may or may not be desirable. The decision of how many instructions to run for each program at each time slice is therefore governed by a couple of parameters which can be tuned by the user. Specifically, a program of length L bits is allowed to execute N instructions per time slice, determined by the formula:

$$N = \texttt{et\_value\_constant} * L^{\texttt{et\_value\_power}}$$

N is rounded down to an integer value. This allows considerable flexibility: for example, if `et_value_power` is set to 0.0, then each program executes `et_value_constant` instructions per time slice, regardless of length; if `et_value_power` is set to 1.0, then the allocation is linearly proportional to program length, so evolutionary selection is size-neutral (all else being equal). Further details of time slicing are given in Section 4.3.5.

[3] The system has been designed to deal with arbitrary n-dimensional environments, but the current implementation requires some minor revisions to allow this.

4.2.4 Naming of Organisms

For the purpose of analysis of the system's behaviour, individual organisms are given names according to their genotype. The name is composed of a number followed by a string of (usually four) upper-case alphabetic characters. The number is the length of the genome (expressed as a number of bits) in the organism's initial cell. The character string is a unique identifier for that particular genome. Ancestor organisms inoculated into the system at the start of the run are named with the character string AAAA. If an offspring has an identical genotype to its parent, it will share the same name. If the offspring has a different genotype, then it is given a new name (the operating system keeps track of which names have already been issued, to avoid duplication). For example, the first organism to appear in the system that differs from the inoculated ancestors will be named with the character string AAAB. Should all character strings up to ZZZZ have been issued for organisms of a particular length, an extra A is added to the string (so the next organism of that length with a different genotype to its parent will be named with the character extension AAAAA).

4.3 The Structure of an Individual Cell

4.3.1 Overview

The basic structure of a single cell is shown diagrammatically in Figure 4.1. Each cell is a process running on the (virtual) Cosmos operating system. A cell has its own program code, working memory, stack, registers and various other structures. The major features of the cell are explained in the rest of this section.

4.3.2 The Genome

The Genome is an InfoString (i.e. a BitString with an associated type), containing encoded instructions that the cell can execute.[4] Which sections of the genome are translated into instructions and executed is determined by the action of promoters and repressors (see Section 4.3.3). After a cell has been created, its genome cannot usually be altered, except by the action of mutation (see Section 4.5.7).

[4] As I am usually referring to the contents of the Genome, rather than to the structure itself, when I use the term 'genome', I will use the standard typeface from now on.

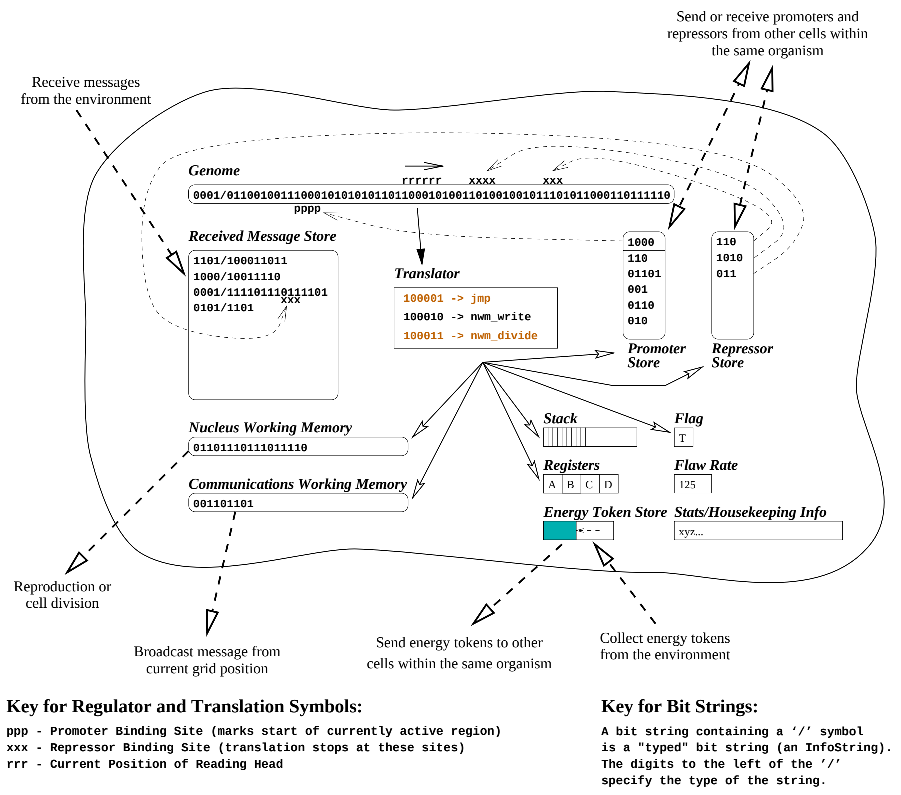

Send or receive promoters and repressors from other cells within the same organism
Receive messages from the environment
Genome
rrrrrr
xxxx
xxx
0001/0110010011100010101010110110001010011010010010111010110001101111 10
pppp
Received Message Store
1101/100011011
1000/10011110
0001/111101110111101
xxx
0101/1101
Translator
100001 -> jmp
100010 -> nwm_write
100011 -> nwm_divide
1000
110
01101
001
0110
010
Promoter Store
110
1010
011
Repressor Store
Nucleus Working Memory
01101110111011110
Communications Working Memory
001101101
Stack
Flag
T
Registers
A B C D
Flaw Rate
125
Energy Token Store
Stats/Housekeeping Info
xyz...
Reproduction or cell division
Broadcast message from current grid position
Send energy tokens to other cells within the same organism
Collect energy tokens from the environment
Key for Regulator and Translation Symbols:
ppp - Promoter Binding Site (marks start of currently active region)
xxx - Repressor Binding Site (translation stops at these sites)
rrr - Current Position of Reading Head
Key for Bit Strings:
A bit string containing a '/' symbol is a "typed" bit string (an InfoString). The digits to the left of the '/' specify the type of the string.

Figure 4.1: The Structure of a Cell in Cosmos.

4.3.3 Regulators: Promoters and Repressors

The translation of the genome is governed by regulators. These are (usually short) BitStrings, and come in two distinct types; promoters and repressors. The cell has a separate store for each of these two types of regulator, and each store can contain a number of regulators of the appropriate type. Regulators may be added to the Promoter Store and Repressor Store in two ways: either by the cell creating a new regulator (by executing an appropriate `reg_create` instruction),[5] or, in the case of a multicellular program (see Section 4.3.9), by the cell being sent a regulator from a neighbouring cell. A cell can also remove regulators from its Promoter Store and Repressor Store, by executing an appropriate `reg_destroy` instruction.

[5] See Section A.2 for an explanation of the instruction set.

Promoters and the Promoter Store

The Promoter Store is an ordered list of promoters. Only the promoter currently at the top of the list is active at any given time. The active promoter specifies the position along the genome[6] at which translation will begin. When a new promoter becomes active, a search is made along the genome for a pattern of bits that matches the promoter bit string.[7] If a matching region is found, the promoter is said to have *bound* to that region, and translation of the genome begins from the first bit to the right of the binding region. If no binding site is found for the active promoter, or when the translation of the current section of genome is terminated (e.g. when the Read position reaches the end of the genome, when it reaches a repressed region, or when a `stop` instruction is encountered), the active promoter is deactivated and placed at the bottom of the list in the Promoter Store, and the promoter which is now at the top of the list becomes active.

Repressors and the Repressor Store

The Repressor Store is a list of repressors, but, unlike in the Promoter Store, any or all of the repressors on the list may potentially be active at the same time. When a new repressor is added to the store, a search is made for a binding site on the genome,[8] in a similar way as for the active promoter. If a binding site is found, the repressor is said to be bound to the corresponding area of the genome, and that area of the genome is said to be repressed. If, during translation of the genome, the read position moves onto a repressed site, translation ceases at that point and the current promoter is deactivated.

4.3.4 The Translator

The process of translating the genome into executable instructions is illustrated in Figure 4.2. As the read head moves along the genome, it passes the string of bits that it reads to the Translator. The Translator has a table that maps bit strings to instructions in the programming language of the cells. As soon as the incoming string

[6] Or on eligible InfoStrings in the Received Message Store. See Section 4.3.7 for details.

[7] The search begins at the current Read position on the genome, and proceeds outwards in both directions simultaneously.

[8] Or on eligible InfoStrings in the Received Message Store. See Section 4.3.7 for details.

of bits matches an entry in this table, the Translator executes the associated instruction and the read head is moved along the genome to the next unread bit. In the current implementation, the map of bit strings to instructions is hard-coded into the Translator, all instructions are encoded by bit strings of equal length (six bits), and all 64 possible six-bit codes have an entry in the table (which means that in some cases, two different six-bit codes encode the same instruction). Any binary string of length six is therefore guaranteed to decode to a valid instruction. This hard-coded mapping is defined in the system input file `genetic_code.ini`, described in Section A.5.1.

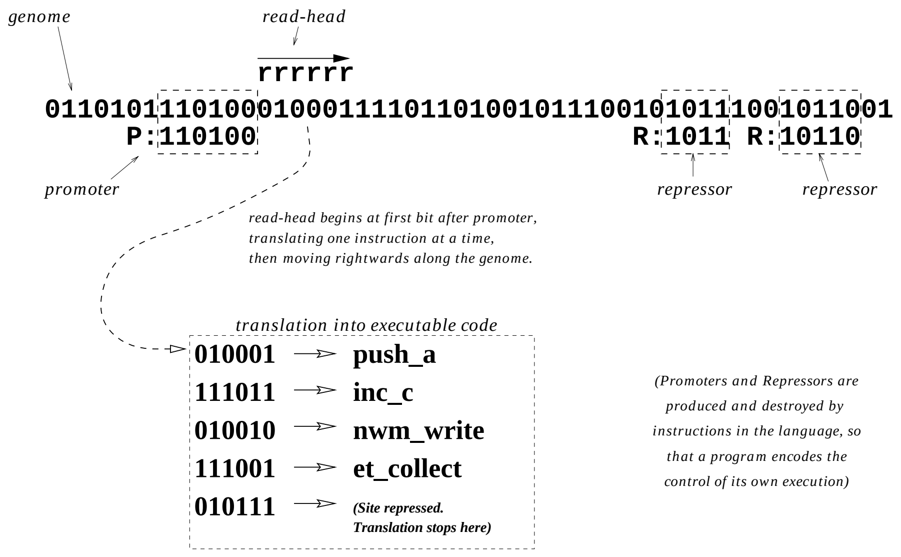

genome
read-head
rrrrrr
01101011101000100011110110100101110010101110010110 01
P:110100
R:1011
R:10110
promoter
repressor
repressor
read-head begins at first bit after promoter, translating one instruction at a time, then moving rightwards along the genome.
translation into executable code
010001 → push_a
111011 → inc_c
010010 → nwm_write
111001 → et_collect
010111 → (Site repressed. Translation stops here)
(Promoters and Repressors are produced and destroyed by instructions in the language, so that a program encodes the control of its own execution)

Figure 4.2: Translation of the Genome.

In future experiments with the system, the hard-coded mapping from bit strings to program instructions may be replaced by a mapping which can vary from one cell to the next, and which can evolve.

4.3.5 The Energy Token Store

A large number of cells may exist concurrently within Cosmos. In order to run the code of all of these cells, the processor must time slice between each cell, as described in Section 4.2.3. In that section a formula was given which shows how many instructions a cell with a genome of a given length is allowed to execute at each time slice. However, for the cell to actually execute this number of instructions, it must pay one *energy token* to the processor for each instruction it executes. A cell has a store of energy tokens (which it collects from the environment as described in Sections 4.5.2 and 4.5.3). Furthermore, a cell's Energy Token Store may be leaky, in which case a number of energy tokens are lost from the store at the end of each time slice, in addition to any that were used to pay for the execution of instructions. The leak rate of the store is determined by the parameter `ets_leak_rate_per_timeslice`, described in Section A.1.

Cell Death

If the number of tokens in this store falls below a particular threshold (defined by the global parameter `ets_lower_threshold`, described in Section 4.8), the cell dies. Additionally, when the maximum number of cells allowed in the system (as defined by the parameter `max_cells_per_process`) has been reached, the processor will kill off a number of cells which have the smallest number of stored energy tokens,[9] in order to make room for new cells. It is therefore essential that a cell maintains a reasonable level of energy tokens in its store. (There is one other way in which a cell may die—it can terminate itself by executing the `kill` instruction.)

When a cell dies, any energy tokens remaining in its Energy Token Store are distributed to the local environment. More information about energy tokens is given in Section 4.5.2.

4.3.6 Cell Division and Reproduction

It has already been mentioned that a cell only has read, write and execute permission within its own boundaries. Considering that the primary function of the cells is to make copies of themselves in other areas of the system's memory, this may seem like an odd restriction. However, the mechanism of cell division and reproduction employed in Cosmos was inspired (albeit fairly vaguely) by the process of cell division in biological organisms.

The Nucleus Working Memory. Each cell has an area called the Nucleus Working Memory, which is just a WritableInfoString. The cell can compose arbitrary bit strings in this area,[10] but in the normal operation of a self-replicating program, it would construct

[9] In this situation, the choice of which cells to kill is actually stochastic, with the level of a cell's Energy Token Store determining the probability of its being killed.

[10] The only restriction is that there is a maximum length to which these strings are allowed to grow, defined by the global parameter `info_string_size_limit`. This is to prevent the situation in which

a copy of its genome here. Thus, rather than directly writing instructions one at a time to a new area of memory (as in Tierra, for example), a Cosmos cell copies its genetic information into its own Nucleus Working Memory. When the genome has been copied in this way, the cell may issue a `nwm_divide` or a `nwm_split` instruction. These have the effect of transferring the contents of the Nucleus Working Memory into a new cell, which will be placed at a nearby grid position. The former instruction creates a cell which is completely separated from the parent cell (i.e. a new child organism), whereas the latter creates a cell which will remain a member of the same organism (i.e. an extra process in a parallel program: see Section 4.3.9).

In either case, upon division the contents of the Energy Token Store, Promoter Store and Repressor Store are divided equally between parent and child cell. The other main structures of the new child cell (i.e. the Nucleus Working Memory, the Received Message Store and the Communications Working Memory)[11] are initially empty.

4.3.7 Inter-Organism Communication Structures

Two major cell structures remain to be explained; these are the Received Message Store and the inter-organism Communications Working Memory. These two structures are both concerned with communications between organisms. The former is used to store incoming messages from other organisms, and the latter is used to compose messages to be sent out to other organisms.

The communications aspect of these structures is described in more detail in Section 4.6.1, but the part they play in the functioning of the cell is explained here.

The Communications Working Memory. The Communications Working Memory, like the Nucleus Working Memory, is a WritableInfoString (with a limited maximum length) which a cell can use to compose arbitrary sequences of bits. A cell can then issue a `cwm_send` instruction to broadcast the contents of the Communications Working Memory into the environment (explained in Section 4.5.5). The Communications Working Memory does not directly affect the functioning of the cell in any other way.

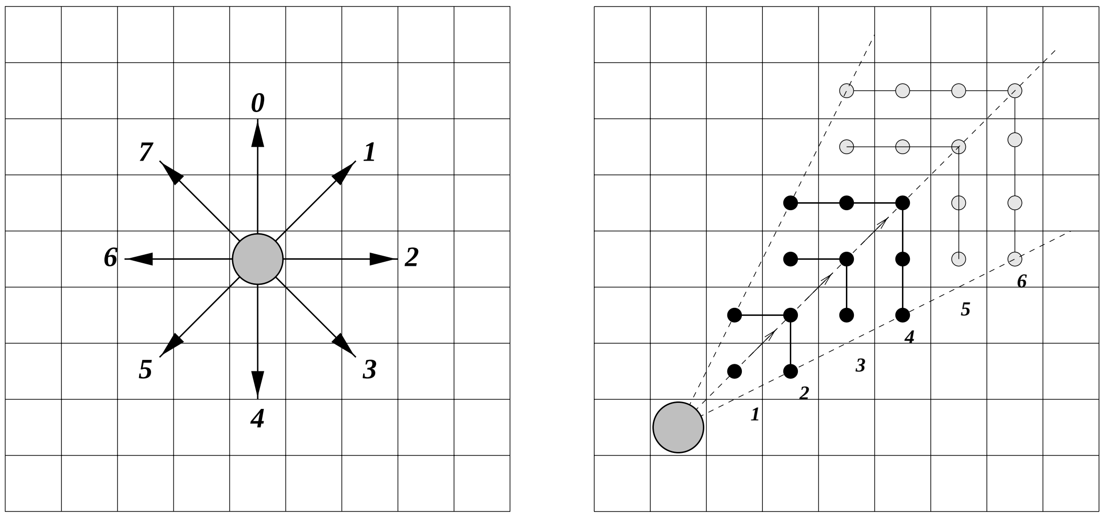
a program evolves which gets stuck in an infinite loop writing to the Nucleus Working Memory, eventually using up all of the memory in the system.

[11] The function of these latter two structures is explained in Section 4.3.7.

The Received Message Store. Inter-organism messages take the form of BitStrings. When they are being composed in the Communications Working Memory they are WritableInfoStrings, when they are broadcast in the environment they are converted to EnvironmentalInfoStrings, and when they are received by others cells into their Received Message Stores, they become plain InfoStrings.

A cell can issue a `rms_receive` instruction to receive messages which have been broadcast from nearby grid positions. These messages (which are EnvironmentalInfoStrings), like all InfoStrings, have a type (a number between 0 and 15) associated with them, and the value of a cell's **dx** register at the time that it issues a `rms_receive` specifies which type of messages are to be received. In addition, the search in the environment for EnvironmentalInfoStrings of the specified type only proceeds in a certain direction; starting from the grid position of the cell that issued the instruction, the search emanates in one of eight directions, specified by the low three bits of the **cx** register (see Figure 4.3(a)). The search proceeds one grid square at a time, covering all grid squares in the specified eighth of the area around the cell until a certain number of grid squares have been searched (defined by the global parameter `rms_receive_search_area`). For example, Figure 4.3(b) shows a cell searching in direction 1. If `rms_receive_search_area` is set to 12, say, then the grid positions marked with black dots will be searched. The search emanates from the cell along a series of wavefronts—the grid position on wavefront 1 is searched first, followed by those on wavefront 2, then 3, then 4. At this point, 12 positions have been visited, so the search stops. Any EnvironmentalInfoStrings of the specified type found in this area are copied into the cell's Received Message Store as InfoStrings. (A cell may extend the reach of a search by re-issuing an identical `rms_receive` instruction from the same grid position within a certain time limit after the first one. This time limit is specified by the global parameter `max_time_for_msg_receive_reinforcement`. If a cell does this, the search will continue outwards from the last grid position searched previously. In the example of Figure 4.3(b), the grid positions marked with gray dots, on wavefronts 5 and 6, will be the next 12 positions searched in this situation.)

The host cell may process these received messages, using the `str_switch` and `adr` instructions to set the **ax** register to an address within a message, and using the instruction `mov_ic` to sequentially read the message.



(a)
A cell can search for messages in one of 8 directions

(b)
A cell searching for messages in direction 1. See text for details.

Figure 4.3: Searching for Communications with the `rms_receive` Instruction.

Messages in the Received Message Store are normally treated as passive structures which may be inspected by the host code, but this is not always the case. As already mentioned, each message in the store has an associated type. The host code of the cell—the genome—being an InfoString, also has a type associated with it.[12] If any message in the Received Message Store happens to be of the same InfoString type as the cell's genome, then it may potentially be used as additional genetic material, and translated into executable instructions. In other words, promoters and repressors may bind to it in just the same way as they can bind to the genome. If the active promoter does indeed bind to a message in the Received Message Store, translation begins along it just as it would on the genome. A cell has several lines of defence against such parasitism, which are mentioned in Section 4.6.1.

A situation where the execution of code from messages in the Received Message Store may be particularly common is when the parameter `neighbouring_genomes_readable` is set to `yes`. In this case, whenever a new promoter becomes active in the cell (see Section 4.3.3), rather than trying to first find a binding site on the cell's genome, or even on eligible messages already resident in the Received Message Store, the cell first

[12] The type of the cell's genome cannot be directly altered, and is passed on to children when the cell splits or divides. However, it is subject to mutation like any other part of the cell (see Section 4.5.7). Therefore, it is possible for organisms with different genome types to emerge in the system.

imports copies of the genomes of any immediately neighbouring cells, one by one, into its Received Message Store. Note that this importation occurs automatically, without the host cell having to issue a `rms_receive` instruction, and without the neighbouring cell having to make a copy of its genome and issue a `cwm_send` instruction. Each imported message (the copy of the neighbouring cell's genome) is checked for a binding site for the new promoter. If a site is found, the message remains in the Received Message Store, and the cell starts executing instructions from it, starting in the position immediately following the binding site (see Section 4.3.3). If no binding site is found, the cell deletes the imported message from its Received Message Store and imports the genome of the next neighbouring cell, if there are any remaining. Only after all the neighbouring genomes have been checked in this way will the cell consider searching for a binding site on existing messages in the Received Message Store, and finally on the cell's genome itself. This mechanism was incorporated into the system in an effort to simulate the ability of programs in Tierra to read the code of neighbouring programs [Ray 91].

4.3.8 Other Structures

There are a number of other structures associated with a cell, which are mentioned briefly here.

Registers There are four (16 bit) registers. The registers **ax** and **bx** are used primarily for storing and manipulating addresses, whereas the registers **cx** and **dx** are used for arithmetic. The main use of the **ax** register is to store addresses returned by the `adr` instruction. This instruction looks for a specified bit string along the genome (or other eligible InfoString), and, if found, returns the address of the first bit of the matching area into the **ax** register. The address is simply the (zero-based) position of the bit from the left of the genome. The `mov_ic` instruction can be used in conjunction with `adr` to read an instruction from the genome, at the address pointed to by the **ax** register, into the **cx** register. Details of these instructions are given in Section A.2. (There is actually a slight complication involved with the use of `adr` and `mov_ic`; these instructions do not only work with the genome, but can also be used on InfoStrings in the Received Message Store, as already mentioned. Each cell actually keeps a pointer called the ADRStringPointer, which normally points to the genome. However, it can be changed to point to one of the InfoStrings in the Received Message Store by the use of the `str_switch` (or similar) instruction. The `adr` and `mov_ic` instructions always work on the InfoString currently pointed to by the ADRStringPointer.)

Flag There is one flag, used mainly to signal unusual or error conditions in the execution of some instructions.

Stack Each cell has a single stack, with a limited maximum capacity (defined by the global parameter `stack_size_limit`). Instructions are included in the language for pushing numbers onto the stack and for popping numbers from it.

Flaw Rate Each cell has a parameter which defines the frequency with which flaws occur in the execution of instructions (see Section 4.5.7). This flaw rate is subject to mutations (Section 4.5.7), so it may evolve over time.

Statistics and Housekeeping Information There are various other minor structures associated with a cell, mostly concerned with keeping statistics of the cell's lineage and activity (for future analysis) and with keeping track of various activities within the cell. These structures are not explained in detail here, but some are mentioned in passing throughout the rest of this chapter where appropriate.

4.3.9 Parallel Programs (Multicellular Organisms)

It has already been mentioned that the design of Cosmos was guided by an analogy to cellular biological organisms (Section 4.1). In order to model not just unicellular organisms, but also multicellular ones, Cosmos has been designed to support parallel programs—an analogy to multicellularity. Furthermore, it allows programs to dynamically create new parallel processes as they are running, as an analogy to the growth of a multicellular organism from a single celled origin.

All programs in Cosmos are instances of the Organism[13] class. An Organism may contain one or more Cells (each Cell being essentially an individual process). There is therefore no fundamental difference in the representation of serial and parallel programs; a serial

[13] A capital 'O' is used here to emphasise that we are talking about the specific implementation details. However, as the Organism class encapsulates the functionality of an organism, the two terms can be used interchangeably. Therefore, in the rest of the document I shall just use the term organism (with a small 'o'). The same applies for cells and the Cell class.

program is just an Organism which has only one Cell, while a parallel program is an Organism with more than one Cell.

Topology of a Multicellular Organism

In a parallel program, each cell has a specific position in the environment (just like any other cell). The only restrictions on the placement of cells within a parallel program (beyond those defined for all cells by the global parameters) are that every cell within the organism must be adjacent to (i.e. occupy one of the eight neighbouring grid positions) at least one other cell owned by the organism, and that two cells within the same organism cannot share the same grid position. The topology of an organism is important in terms of its intercellular communications, as any given cell can only exchange regulators and energy tokens with immediately adjacent cells within the organism. By means of this transfer between cells in a multicellular organism, the behaviour of any cell is affected by the behaviour of its neighbours. See Section 4.6.1 for more details.

As a parallel program develops, an individual cell can actually change its position relative to its neighbours, using the `migrate` instruction. This gives a cell the opportunity of interacting with different neighbouring cells throughout the life of the program.

Energy Transport

As mentioned above, a cell in a multicellular organism can pass energy tokens from its store to its neighbouring cells, using the `et_transport` instruction. In this way, it is possible for a multicellular organism to develop specialised cells that collect energy tokens from the environment and distribute them throughout the rest of the organism, leaving other cells free to specialise in other tasks if necessary.

Fission

It has already been said that all of the cells comprising a multicellular organism are restricted to being located in such a position that they are in contact with (i.e. in an adjacent grid position to) at least one other cell in the organism. However, as individual cells within a multicellular organism can die at different times (in the ways described in Section 4.3.5), it is possible to get a situation where a collection of cells that was once connected as a multicellular organism breaks into two or more unconnected groups of cells because of the death of one of more cells in the middle of the structure (see Figure 4.4). If such a situation arises, the separate sub-groups of cells each now become separate organisms in their own right. Cell division and organism fission are therefore two distinct ways in which a new organism may be created.

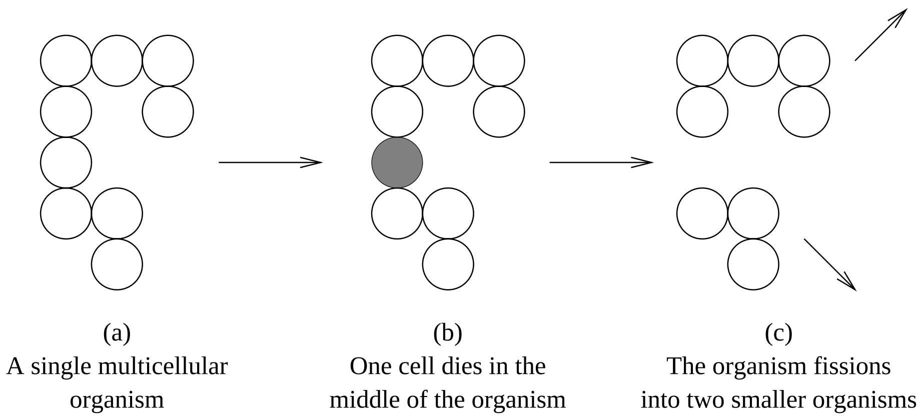

(a)
A single multicellular organism
(b)
One cell dies in the middle of the organism
(c)
The organism fissions into two smaller organisms

Figure 4.4: An Example of Organism Fission.

The Cost of Multicellularity

The cost of being part of a multicellular organism is governed by the global parameter `multicellularity_penalty_factor`. That is, for each cell in a multicellular organism, this parameter represents the number of energy tokens that are deducted from that cell's Energy Token Store at each time slice for each additional cell with which it is in contact. For example, if a cell is adjacent to two other cells belonging to the same organism (i.e. there are two cells with which it can exchange regulators and energy tokens), then at each time slice, twice the amount of energy tokens as specified by `multicellularity_penalty_factor` are deducted from that cell's store. This parameter therefore defines how expensive it is for a cell to maintain a connection with one other cell in a multicellular organism.

Organism Death

An Organism is composed of one or more cells. In Cosmos there is no specific idea of an organism, as a whole, dying—rather, an organism dies when the last of its constituent cells dies.

4.4 The Programming Language and Representation

REPLiCa,[14] the programming language in which the self-replicating programs are written, is based upon the Tierran language [Ray 91], with some changes and additions to support the extra functionality of Cosmos. Like Tierran, REPLiCa has been designed to be robust, in the sense that there is little syntactical structure to a program, so that any random collection of REPLiCa instructions will form a valid program that will do *something* (maybe not anything sensible, but it will not cause the system to crash). The REPLiCa instruction set is listed, with annotations, in Section A.2.

One big difference between Tierran and REPLiCa is in the mechanism for control flow branching and jumping. Tierran uses a system of template-driven jumping (see [Ray 91] for details). REPLiCa does not have jumps of this kind; rather, jumps may be accomplished in two different ways. The first, primarily for single jumps rather than loops, is just by the creation of an appropriate promoter to bind to the desired jump destination, either followed by the deletion from the Promoter Store of the currently active promoter (using the `reg_destroy` instruction), or by the issuing of a `stop` instruction—both of which have the effect of stopping the execution of the current section of code and activating the new promoter.[15] The second way by which (local) jumps may be performed is by the use of the `set_jmp` and `jmp` instructions. Each cell contains a pointer called the LocalJumpPointer which, if set, points to a position on the genome (or currently active InfoString in the Received Message Store). When a `set_jmp` instruction is executed, this pointer is set to the address of the next instruction. When a `jmp` instruction is executed, control passes to the instruction pointed to by the LocalJumpPointer (if it is set, otherwise no jump is performed). The LocalJumpPointer can be cleared with the `clr_jmp` instruction.

The translation of the bit-string representation of a program on the genome, and the control of execution of the program by promoters and repressors, illustrated in Figure 4.2, has already been explained in Sections 4.3.2–4.3.4.

[14] 'REPLiCa' is an acronym for Robust Evolvable Programming Language for Cosmos.

[15] Of course, when programs are evolving, especially when we are considering parallel programs, there may be more than one promoter in the Promoter Store at one time. However, here we are describing how a human might design a program that performs a jump—evolution would probably go about designing a program in a very different way.

4.5 The Environment

4.5.1 The Grid

As mentioned in Section 4.2.2, cells in Cosmos live in a discrete two-dimensional spatial environment (the 'grid'). At the start of each time slice, a number of energy tokens are deposited to each position on the grid (see Section 4.5.2). Cells can collect these energy tokens by using the `et_collect` instruction (see Section 4.5.3). If energy tokens are scarce at a cell's current location (or indeed for any other reason), the cell (to be precise, the whole organism) may move around the grid (see Section 4.5.4). For multicellular organisms, each cell must occupy a different grid position, i.e. all organisms are 'flat' (cells cannot pile on top of each other in the same grid position). However, cells from different organisms *can* occupy the same grid position. What this means is that all organisms are flat, but they can 'slide over' each other, and in this sense the environment is two-and-a-half dimensional.

4.5.2 Distribution of Energy Tokens

At the start of each time slice sweep across all of the cells in the population (in the routine `DistributeEnergyTokens`, described in Section 4.7), the Cosmos operating system releases a certain number of energy tokens into the environment. These tokens are then available to be collected by cells, by the use of the `et_collect` instruction. At the end of each time slice sweep (in the routine `AttenuateEnvironmentalEnergy`, also described in Section 4.7), the operating system takes a number of energy tokens away from each grid position. In the current implementation, different grid positions may receive different numbers of energy tokens at the beginning of each time slice sweep (determined by the various distribution schemes described below), but all positions have the same number of energy tokens removed at the end of each time slice sweep (specified by the parameter `number_of_energy_tokens_per_grid_pos_per_sweep`, if they have that number available). If the number of energy tokens received by a grid position in a time slice sweep exceeds the number removed from it, and they are not collected by cells during that sweep, the excess tokens remain there for future collection. A grid position may therefore sometimes accumulate a relatively large number of energy tokens (up to a maximum limit defined by the global parameter `max_energy_tokens_per_grid_pos`) if there is not much demand for them by cells in the locality.

The distribution of energy tokens across the grid may follow a number of different patterns, defined by the global parameter `energy_distribution_scheme`. At present, four such patterns are defined: `land`, `sea`, `mixed` and `random`. Note that the total number of energy tokens distributed to the environment at each time slice sweep is always specified by the product of the parameter `number_of_energy_tokens_per_grid_pos_per_sweep` with the number of squares in the grid. The different distribution schemes determine how many of these tokens are distributed to individual squares. The different schemes work as follows:

Land Each grid position receives a constant number of energy tokens from one time slice to the next. In the current implementation, there is one extra parameter, `x_delta`, associated with this sort of energy distribution, which defines the gradient of the distribution from the left-hand side of the grid to the right-hand side. See Figures 4.5(a) and 4.5(b) for examples of this type of distribution.

Sea In contrast to `land` distribution, for `sea` distribution each grid position receives a varying number of energy tokens from one time slice to the next. During each time slice, energy tokens are distributed to grid positions which are located under a 'wave'—a vertical band which moves one position to the right after each time slice: see Figure 4.5(c). Grid positions which are not located under a wave in the current time slice receive no energy tokens for that time slice. In the present implementation there are two parameters associated with this method; `wave_width` and `number_of_waves`. The former specifies the width, in grid positions, of a single wave, and the latter specifies how many waves are to be fitted in to the grid from left to right (the waves are evenly spaced across the grid).

Mixed This is a mixture of `land` and `sea` distributions, with the top portion of the grid receiving energy according to the `land` distribution, and the bottom portion according to the `sea` distribution. The relative sizes of these top and bottom portions of the grid are determined by the global parameter `land_fraction`. An example is shown in Figure 4.5(d).

Random Energy tokens are distributed in packets with size determined by the global parameter `energy_distribution_random_chunk_size` to randomly chosen grid positions, until the correct total number of energy tokens have been distributed. An example is shown in Figure 4.5(e).

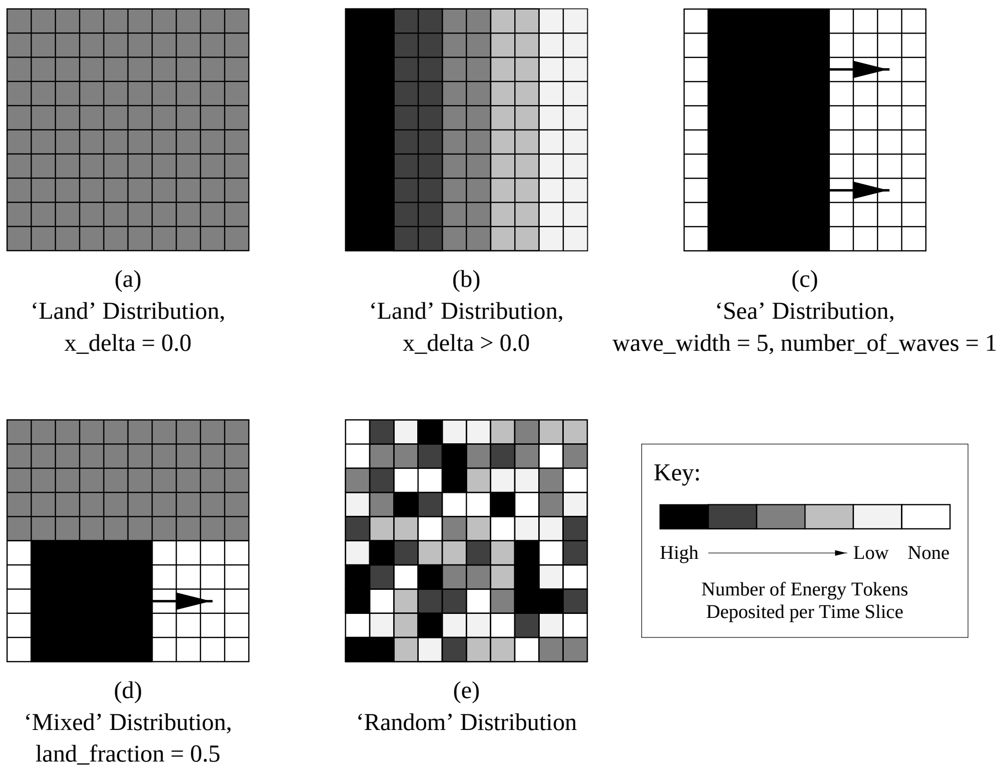


Figure 4.5: Different Patterns of Energy Token Distribution.

A multicellular organism may also pass energy tokens between its cells (using the `et_transfer` instruction), leading to the possibility of some of the cells specialising in energy token collection and distribution of these tokens to the other cells in the organism.

With such a system of CPU-time allocation, programs may potentially evolve which operate on a wide variety of time-scales. For example, very short programs may exist which quickly grab just enough energy tokens to make a copy of themselves, while much more complicated programs may coexist which gather large numbers of tokens over long periods of time, and reproduce at a much slower rate.

When a cell dies, any unused energy tokens are passed back to the local environment (where they may be collected by other organisms). This mechanism provides potential selection pressure for the evolution of organisms that kill other organisms in order to collect the energy tokens thus released into the environment. This could happen if, for example, an organism transmitted EnvironmentalInfoStrings containing the `kill` instruction, which another organism subsequently received and executed (see Sections 4.3.7 and 4.6.1 for further details of how this would work).

4.5.3 Collection of Energy Tokens

In the present implementation a choice of two energy collection schemes, `shared` and `private`, is provided. The global parameter `energy_collection_scheme` determines which scheme will be used.

Shared Energy. Under this scheme, when a cell issues an `et_collect` instruction to collect energy tokens from the environment, it first tries to collect spare tokens from its current grid position. However, if the grid position does not contain sufficient energy tokens, the cell then looks for other cells at the same grid position or in one of the eight neighbouring grid positions. If other cells exist in one of these nine locations, energy tokens will be extracted from the Energy Token Store of one (or more) of these (at random) until the cell has obtained the normal quota of energy tokens for one execution of `et_collect` (as defined by the global parameter `number_of_energy_tokens_per_collect`).

Private Energy. With this scheme, if the local grid position does not contain enough energy tokens for an `et_collect`, the cell just takes what is there, but does not attempt to gain additional energy tokens from neighbouring cells.

4.5.4 Moving around the Grid

A cell does not have to remain in its original grid position, but can move around by using the `move` instruction. The contents of the **cx** register at the time the instruction is issued determines in which direction the cell will try to move (the low 3 bits specify a direction from 0 to 7, as indicated in Figure 4.3(a)).

However, movement is complicated by the fact that a cell may be part of a multicellular organism (in which other cells are also trying to move, possibly in different directions). The organism must move as a whole, so what actually happens is that the issuing of a `move` instruction by a cell is actually a *vote* to move in a particular direction rather than an instruction that has immediate effect. At each time slice, an organism counts up all of the movement votes from its constituent cells, and decides how to move as follows:

A normalised total movement vector **A** is calculated by summing all the individual votes of cells within the organism:

$$\mathbf{A} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{a}_i \tag{4.1}$$

where n is the total number of cells in the organism, and $\mathbf{a}_i$ is the unit vector of movement (in one of eight possible directions) specified by cell i (or $\mathbf{0}$ if the cell did not issue a `move` instruction during the current time slice).

A 'multiple movement factor', M, is then calculated. This factor determines the extent to which two or more cells moving in tandem within an organism are more efficient than would be expected by simply summing their individual movements. M is defined as:

$$M = (m-1)L + 1 \tag{4.2}$$

where m is the number of cells within the organism that individually issued a `move` instruction; and L is the 'constant of leverage' when two or more cells move at the same time ($L \geq 0$). L is defined by the global parameter `movement_leverage_factor`.

Movement is further complicated in the situation where the organism is overlapping (or partially overlapping) another organism on the grid. In this case, there is a 'friction' term F which slows the organism down as it attempts to move over other cells. This term is defined as follows:

$$F = \frac{o}{n} \tag{4.3}$$

where o is the number of cells in the organism which share a grid position with cell(s) from other organisms. The friction factor can actually be turned on or off with the global parameter `apply_friction_factor`. If it is turned off, F is effectively set to zero.

The total movement that the organism attempts to make, **X**, is therefore specified by

$$\mathbf{X} = M(1-F)\mathbf{A} \tag{4.4}$$

The organism moves from its current position by the distance and direction given by **X**, unless it reaches the edge of the grid, in which case it stops at that point (if the grid boundary does not wrap).

4.5.5 Inter-Organism Communications

If a cell broadcasts an inter-organism communication using the `cwm_send` instruction (as mentioned in Section 4.3.7), the contents of its Communications Working Memory is packaged into an EnvironmentalInfoString structure (with an initial intensity specified by the global parameter `envinfostring_initial_intensity`, and a type specified by the low four bits of the **dx** register). This EnvironmentalInfoString is deposited in the environment in the same grid position as the cell, where it can be detected by other cells (by using the `rms_receive` instruction, described in Sections 4.3.7 and 4.6.1).

Each grid position in the environment can hold one EnvironmentalInfoString of each of the 16 possible types. If a string of the same type already exists in the grid position when a `cwm_send` message is issued, the existing string is deleted and replaced by the new one.

At each time slice sweep (in the `AttenuateMessageIntensities` routine, described in Section 4.7), the intensity of each EnvironmentalInfoString is attenuated according to the following equation:

$$I_{n+1} = k(I_n)^p \tag{4.5}$$

where I_n is the intensity at time n, and k and p are constants defined by the global parameters `envinfostring_decay_constant` and `envinfostring_decay_power` respectively. When the intensity of any string falls below a certain threshold (defined by the global parameter `envinfostring_lower_threshold`), the string is deleted.

There is one additional feature associated with these EnvironmentalInfoStrings, whereby a cell can reinforce the intensity of a message that it has already sent. If the cell reissues the `cwm_send` instruction within a given number of time slices (determined by the parameter `max_time_for_msg_send_reinforcement`), while still in the same grid position, and it has not written anything else into its Communications Working Memory in the meantime, then the intensity of the existing EnvironmentalInfoString is incremented by a small amount.[16]

4.5.6 Environmental Information

As well as carrying specific inter-organism communications (mentioned in Section 4.3.7 and explained in more detail in Sections 4.5.5 and 4.6.1), the environment also carries summary information about itself. These messages are transmitted (in the form of EnvironmentalInfoStrings) by the environment itself, one at each grid position, and may be intercepted by cells in exactly the same way as they intercept other inter-organism communications. The messages contain the following information (represented in a binary encoding):

- The number of cells at that grid position
- The total number of free energy tokens at that grid position

All of these messages behave just like any other EnvironmentalInfoStrings in the environment; the only distinguishing feature is that they are all given an InfoString type of 15. (There are no restrictions about organisms using the same type number for their own communications.) They may be picked up by any cell using the `rms_receive` instruction.

4.5.7 Mutations and Flaws

Little has so far been said about the role of mutation in Cosmos. 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. Mutations occur naturally throughout the system at a low rate, and may affect most of the structures within the cell (i.e. the Genome, the Received Message Store, the Nucleus Working Memory, the Communications Working Memory, the Promoter Store, the Repressor Store, the flaw rate, the stack, the registers and the flag). For structures which are based upon BitStrings, mutations are governed by the global parameter `mutation_period`, which specifies the probability of an individual bit within the structure being flipped. For structures based upon integer

[16] To be precise, the magnitude of the increment is kI^p, where I is the current intensity, and k and p are constants defined by the global parameters `envinfostring_decay_constant` and `envinfostring_decay_power` respectively.

numbers (the flaw rate, stack and registers), mutations occur at the same rate as for BitStrings, but the details are slightly different. For the flaw rate, a mutation causes a random increment or decrement in the current value within predefined limits.[17] For the stack, a mutation will, with equal likelihood, either cause a random number to be pushed onto the stack, or the top number to be popped off it. For registers, a mutation will cause the register's current value to be replaced by a random value. Mutations also affect the cell's flag at the same rate, causing the flag's state to be inverted.

In addition, variety may also be introduced into an organism by the flawed execution of instructions in its genome.[18] When a flaw occurs (which happens at a rate defined by an individual cell's flaw rate, as described in Section 4.3.8), the instruction which is about to be executed, rather than just being executed once, will either be executed twice (successively) or not at all. (The choice is random, with both events occurring with equal likelihood.) The effect of a flaw is therefore that instructions may occasionally produce abnormal results, such as an `inc_a` instruction adding 2 to the value of the **ax** register instead of 1.

Despite this distinction between mutations and flaws, the net results are the same. If the error affects what gets written to the Nucleus Working Memory of a cell just before it issues a `nwm_divide` instruction, then it will be passed on to the child organism and become a permanent addition to the gene pool. On the other hand, if the error does not affect the contents of the Nucleus Working Memory (even indirectly), and it does not affect the regulators that get passed on to any offspring, then it will only affect the current organism and will not be inherited by child organisms. From an evolutionary point of view, only the former scenario is important.

4.6 Actions and Interactions

The methods available to cells and organisms for interacting with the 'physical' environment and with other cells and organisms have already been discussed: issues such as

[17] To be precise, the flaw rate can change by plus or minus n parts per thousand, where n is determined by the parameter `flaw_period_max_change_per_thou`.

[18] Tierra features both mutations and flaws (although the mechanisms for flaws is somewhat different) but in subsequent work by Chris Adami and Titus Brown with their Avida system the authors suggested that flaws played only a minor role in evolution compared to mutations [Adami & Brown 94]. Informal observations from preliminary runs of Cosmos suggested that flaws in the execution of instructions significantly increase the rate at which useful mutants are produced.

the collection of energy tokens from the environment, and moving around the grid, were explained in Section 4.5; intercellular communications (i.e. the transfer of energy tokens and regulators) have been mentioned in Sections 4.3.9 and 4.5.2; and inter-organism communications have been mentioned (from the point of view of the mechanisms involved) in Sections 4.3.7 and 4.5.5. In the present section, more will be said about some higher-level effects and implications of both types of communication.

4.6.1 Implications of Intercellular and Inter-Organism Communications

The general philosophy governing the design of the communication facilities in Cosmos was to provide the organisms with as rich an environment as possible. In particular, the inter-organism communications instructions allow organisms to exchange arbitrary messages. The idea is that, as in nature, many possibilities for communication are provided by the 'physics' of the system. The question of whether these possibilities are realised or not is left to the evolutionary process.

Intercellular Communications

As mentioned in Section 4.3.9, a cell which is a member of a multicellular organism can communicate with other cells in the organism by sending regulators from its Promoter Store and Repressor Store (using the `reg_transport` instruction). In this way, the execution of code in a particular cell may be influenced by many other cells in the organism, because regulators which are sent from one cell to another will influence which sections of code get executed in both cells. Therefore, although each cell in a multicellular organism has the same genome (assuming there are no somatic mutations), each cell may be executing different parts of this genome at any given time.

As a cell within a multicellular organism can only exchange regulators and energy tokens with its immediate neighbours, organisms adopting different shapes will have different capacities for internal communication and regulation. Within an organism, cells can also actively switch neighbourhoods by migrating to a different position (using the `migrate` instruction). If multicellular organisms do evolve in any runs of Cosmos it will be of interest to see what sorts of shapes they adopt, and how much variety in shape exists across the population.

Inter-Organism Communications

The mechanisms for inter-organism communications were introduced in Sections 4.3.7 and 4.5.5. A cell can broadcast an arbitrary message using the `cwm_send` instruction, and receive other messages from the environment—sent from cells in other organisms, cells within the same organism, or from the environment itself (Section 4.5.6)—using the `rms_receive` instruction. Allowing organisms to exchange arbitrary bit strings has little direct biological analogy. Rather, it is an attempt to equip the organisms with some communication channels in much the way that biological organisms can communicate using channels such as light, sound etc.

Once messages (InfoStrings) have arrived in a cell's Received Message Store, they may be read by the host code (using `str_switch`, `adr`, `mov_ic` and related instructions), and messages of the same type as the genome of the host cell may even be treated as executable code, as described in Section 4.3.7. This allows for genetic information to be exchanged between organisms in a manner analogous to the direct exchange mechanisms employed by lower biological organisms such as viruses and bacteria.

If the foreign code is detrimental to the performance of the host cell, the host may be expected to evolve measures to prevent the foreign code from being executed. This can be achieved in a number of different ways, such as by using a different type number for its own genome (which may come about by mutation), by removing the foreign code from the Received Message Store (using the `str_remove` instruction), or by not receiving the foreign code in the first place. If, however, the foreign code is beneficial to the host, then it may be expected that the host will evolve to copy this code into its Nucleus Working Memory so that it will become incorporated into the host genome in future generations. The system is even flexible enough to allow for the possibility of the evolution of sexual reproduction (see Section 6.7 for details of a hand-written sexual organism).

4.7 The Top-Level Algorithm

A pseudo-code listing of the top-level algorithm is shown in Figure 4.6. Most of it should be self-explanatory. The `Inoculate` routine constructs a number of self-replicating programs and places them at specified positions on the grid (governed by the para-

```
Inoculate
currentTimeSliceSweep = 1
while (stopping criteria not met)
{
      DistributeEnergyTokens
      AttenuateMessageIntensities
      ExecuteCellTimeSlices
      PerformOrganismLevelOperations
      if ((currentTimeSliceSweep MOD mutation_application_period) = 0)
           ApplyMutations
      if ((currentTimeSliceSweep MOD overcrowding_check_period) = 0)
           CheckOvercrowding
      if ((currentTimeSliceSweep MOD env_info_broadcast_period) = 0)
           BroadcastEnvironmentalInfo
      ExportData
      AttenuateEnvironmentalEnergy
      currentTimeSliceSweep = currentTimeSliceSweep + 1
}
```

Figure 4.6: The Top-Level Algorithm.

meters `ancestor`, `number` and `placement`). The stopping criteria for the main loop may be to run for a given number of time slices (if the parameter `limited_run` is set to `yes`) or to run indefinitely (only stopping if and when all programs on the grid have died out). `DistributeEnergyTokens` places a number of energy tokens in each grid position, as described in Section 4.5.2. `AttenuateMessageIntensities` refers to the intensities of any EnvironmentalInfoStrings that currently exist in the environment. `PerformOrganismLevelOperations` checks, for each organism, whether a fission has occurred by the death of one of more cells within it (see under "Fission" in Section 4.3.9), subtracts energy tokens for each cell in a multicellular organism depending on how many neighbours the cell has (see under "The Cost of Multicellularity" in Section 4.3.9), and finally calculates and performs any movement of the organism from the contributions made by individual cells (Section 4.5.4). `CheckOvercrowding` checks whether the current population of cells on the grid exceeds the limit specified by the global parameter `max_cells_per_process`. If so, a fraction of the population (specified by the parameter `population_cutback_on_overcrowding`) is killed off. The choice of which cells to kill in this situation is stochastic, but is based upon how much energy each cell has stored in its Energy Token Store. `BroadcastEnvironmentalInfo` generates an environmental message of each grid position, as described in Section 4.5.6. `AttenuateEnvironmentalEnergy` removes a number of energy tokens from each grid position, as described in Section 4.5.2.

4.8 Global Parameters

The Cosmos system as described contains a considerable number of global parameters. These are listed and described in Section A.1. The number of parameters is much larger than in most other artificial life platforms, but this is largely because other platforms often have many features which are hard-coded in a fairly arbitrary way. In contrast, Cosmos was designed to allow the user a great degree of control over the system's configuration.

4.9 Input and Output Files

The configuration of an individual run is specified in a number of files which Cosmos reads when the run commences. These files contain details of non-default parameter settings, of the mapping between instructions in the REPLiCa programming language and the binary encoding used to represent them in a cell's genome, and of user-defined ancestor programs. Full details of these input files are given in Section A.5.1.

The core Cosmos system is a stand-alone application. In order to allow the analysis of an evolutionary run, a number of log files containing information about the different organisms are written by the system during the run. The files may then be used to produce graphs and statistics about the run. These output files are described in Section A.5.2.

4.10 Major Differences between Cosmos and Tierra

In this section the main areas in which Cosmos differs from Tierra are highlighted. A fuller explanation of why some of these differences were incorporated into the system can be found in [Taylor & Hallam 97]. In the following, the extension of standard Tierra to deal with parallel processes, as described in [Thearling & Ray 94] and [Thearling 94], is referred to as 'Parallel Tierra'.

Cellular Structure. An individual program (or more precisely, an individual process, which may be serial or parallel) in Cosmos has many more structures associated with it than do programs in Tierra. Tierran programs just have the list of instructions, a program pointer, registers and a stack. In contrast, Cosmos programs also have all of the structures explained in Section 4.3. The idea was that they should incorporate some of the features (e.g. regulators, translation machinery, and areas where new strings may be constructed) observed in cellular biological organisms. The programs must rely largely on communications to interact with the outside world, and cannot directly read the code of their neighbours.

Regulator System. The regulator systems of Cosmos (promoters and repressors: see Section 4.3.3) have no equivalent in Tierra. They were designed specifically to allow cells in a multicellular organism to be able to influence which sections of code were being executed in neighbouring cells, thereby promoting cell differentiation and specialisation. The design of the regulator systems was inspired by the processes of chemical signalling between cells, and the use of promoter sequences and repressors within cells, in biological organisms.

CPU-time Allocation and Energy Tokens. In Cosmos, each cell has to pay one energy token for every instruction it executes. Cells must collect these tokens from the environment, and store them in their Energy Token Store. A cell dies when the number of tokens in its Energy Token Store falls below a threshold (defined by the parameter `ets_lower_threshold`). Furthermore, if the population size exceeds a threshold (defined by the parameter `max_cells_per_process`), cells are killed off stochastically, but those with fewer energy tokens in their Energy Token Store have a greater chance of being killed. A cell can therefore exert considerable influence over its own longevity, via its success at collecting energy tokens from the environment.

In contrast, programs in Tierra have little control over their longevity. As individual Tierran programs have no notion of energy levels, a separate 'reaper queue' mechanism is employed to govern cell death. Programs can move up the queue if they cause error conditions during execution (see Section 3.2.1), but in general the probability of death increases with age [Ray 91]. The reaper queue therefore effectively imposes an upper limit on the lifespan of programs, whereas there is no theoretical upper limit in Cosmos.

Additionally, the energy token scheme in Cosmos introduces the idea of a competition for the available energy, which is missing in Tierra. Furthermore, if the parameter `energy_collection_scheme` is set to `shared`, cells may extract energy tokens from their neighbours. In this situation, a cell is a potential energy resource for other cells, and, if environmental energy were scarce, it would become advantageous for a cell to kill its neighbours by draining their energy. If cells could defend themselves against such attacks, some sort of coevolutionary process might arise from such interactions.

Read, Write and Execute Privileges. Tierran programs only have write access within their own 'cell membrane' (apart from when they are in the process of creating a daughter cell, when they also have write access to a specific additional chunk of memory, which has been allocated by the Tierra operating system). A similar situation exists in Cosmos. However, Tierran programs have read and execute privileges for *all* areas of instruction memory, so that they can directly examine the code of other programs, and even execute this code. Cosmos cells, on the other hand, only have direct read and execute privileges within their own cell membrane, and must rely on the system's communication facilities to interact with other cells (see Section 4.6.1). This restriction in Cosmos is related to the guiding analogy of the biological cell, which cannot directly read the genetic code of a neighbouring cell.

Exchange of Messages and Genetic Information. The Cosmos mechanisms for the direct exchange of arbitrary messages (which may, for example, be copies of genetic information) have no parallel in Tierra. This difference is linked to the differences in read, write and execute privileges described in the previous point.

Division Process. This point is related to the previous two. As a Cosmos cell only has write access within its own cell membrane even when it is composing a copy of itself, this copy must first be composed within the parent cell (in the Nucleus Working Memory). The copy is then issued *en masse* to a new memory location.

In Tierra, a cell is first allocated a new block of memory, then writes a copy of itself into this memory, and finally 'divides', signalling that the block of memory is now a new organism in its own right.

There is not a great deal of difference between the two mechanisms, but an advantage of the Cosmos method is that it allows an organism to *reproduce* (i.e. to create a child organism) and to *grow* (i.e. create a new cell which remains a member of the multicellular organism) using exactly the same technique.

In contrast, Parallel Tierra includes a `split` instruction which adds an additional CPU to the processor structure of the program. This mechanism is natural for a parallel machine architecture with a shared program space, as used with Parallel Tierra. In Cosmos memory is not shared across cells, so that a multicellular program must actually copy itself from one cell to another in order to run in parallel. With this type of architecture, it seems preferable that the bulk of such copying work should be performed by the cells themselves rather than by the Cosmos operating system (but see the further discussion on this topic in Section 7.2.3).

Additionally, having very similar mechanisms for growth and reproduction of organisms is arguably more analogous to the way that multicellular biological organisms may have evolved.

Local Competition. One of the problems that has been observed with the process of evolution in Tierra is that it suffers from premature convergence due to global interactions between cells [Adami & Brown 94].

Chris Adami and Titus Brown sought to overcome this problem in their Avida system by giving each of the cells a location on a two dimensional toroidal grid. Cells can only interact with other cells occupying nearby grid positions, thereby slowing down the rate of propagation of evolutionary changes throughout the total population and promoting heterogeneity.

Cosmos addresses this problem by placing organisms on a grid (as in Avida), and by restricting cells to only be able to communicate and interact with other cells within a certain distance on the grid (but see the further discussion on this topic in Section 7.2.3).

Binary Representation. In Tierra, programs are directly represented as lists of instructions. In Cosmos, the program code is represented as a binary string (specifically, an InfoString), and a translation process is required to produce the executable code. One consequence of this design is the possibility of the evolution of ultra-compact programs which use the same section of bit string to encode multiple sequences of instructions in different reading frames (as is observed in some biological organisms; [Matthews 91] p.144). Another consequence is that it would be easy to modify the system in order to study the evolution of the genetic code itself (i.e. the mapping from bit strings to program instructions).

Size of Instruction Set. The REPLiCa instruction set is about twice as big as that of the Tierran language. Many of the instructions can certainly be removed without having a great impact on the things that programs can do (e.g. the self-replicator listed in Section A.3.1 only uses 17 different instructions). If the genetic code were allowed to evolve, then unused instructions might be expected to be removed from the code by natural selection, allowing common instructions to be represented multiple times.

Memory Model. Cosmos uses a distributed memory model of parallelism, in contrast to the shared memory model of Parallel Tierra. In other words, each cell in a multicellular organism in Cosmos has its own copy of the program code, of the other cellular structures, and of the CPU state information (registers, instruction pointer, etc.). This distributed memory model, together with the Cosmos regulator system, should promote the emergence of differentiation in parallel programs. However, little work has so far been conducted with parallel programs in Cosmos, so it is not yet known how effective this approach really is.

Memory Addressing Scheme. For reading from and writing to structures within cells, Cosmos uses a local addressing scheme for each structure (i.e. the first bit of the Genome, of the Communications Working Memory, and of the messages in the Received Message Store, are all treated as address zero within that particular structure). Cells have no knowledge of their memory location (or that of other cells) in the global addressing scheme of the system. This is in contrast to Tierra, which uses a global addressing scheme. The only ways that cells can interact with each other are therefore by communication; by physical contact, such as by extracting energy tokens from each other (which is possible when the parameter `energy_collection_scheme` is set to `shared`—see Section 4.5.3) and slowing down passing organisms (see Section 4.5.4); and, for cells within a multicellular organism, by the exchange of regulators and energy tokens.

4.11 Summary

In this chapter I have described the design details of the Cosmos platform. Cosmos is an implementation of a virtual parallel computer that can simulate the concurrent execution of several thousand programs (limited only by the amount of memory available). The general design was influenced by Ray's Tierra platform [Ray 91], but there are some fairly significant differences between the two systems (as listed in Section 4.10). The following chapter is devoted to a detailed description and analysis of the system's behaviour during a single evolutionary run, where the memory (i.e. the grid) is inoculated with a number of hand-written self-reproducing programs. The results of a wide variety of further experiments with Cosmos are described in Chapter 6.

Chapter 5

Cosmos Experiments 1: Detailed Analysis of a Standard Run

In this chapter, the results of a standard Cosmos run (i.e. using the default parameter settings) are described. First of all, the various analysis and visualisation techniques that will be used are explained (Section 5.1). After this, the results of the run are analysed in some detail (Section 5.2). In the next chapter, results of many other runs, designed to explore the parameter space of the system, will be presented.

5.1 Analysis and Visualisation Techniques

One of the reasons that is often given for pursuing research with artificial evolutionary systems is that they allow us to study evolution in carefully controlled conditions, and to record as much data as we wish during the course of an evolutionary run. However, the very fact that we can record anything we wish to can sometimes lead to problems. Without care, it is very easy to drown in megabytes of data. In this section, we consider some of the questions we might want to ask concerning the behaviour of an evolutionary system, and describe the measures that were used in the Cosmos runs to provide answers to these questions.

There is a broad distinction that can be made between measures which relate to individual organisms within the population, and measures which relate to the population as a whole. We will use both types when analysing the Cosmos runs in this chapter.

5.1.1 Individual-based Measures

In any population of self-replicating entities which are competing against each other for resources required for replication (e.g. energy and materials), there are three factors which determine the rate at which any particular type of replicator, i.e. any particular S-lineage (see Section 2.5), will spread throughout the population [Dawkins 76]. These are the life-span (*A-longevity*) of the individual A-replicators, the rate at which they replicate (their *A-fecundity*), and the error rate[1] of the replication process (their *A-fidelity*). These three factors can be thought of as defining the orthogonal axes of the fundamental space in which evolution is occurring. Adaptive (i.e. non-neutral) evolution would be expected to move the A-replicators' positions in this space either to greater A-longevity, increased A-fecundity or increased A-fidelity (or some combination of these).

In biological systems, the fundamental units of evolution are the (L-)genes, and it does not necessarily follow that the evolution of whole organisms will follow this pattern. For example, evolution of an L-gene towards increased L-fecundity across a group of organisms carrying an A-gene instance of the L-gene does not necessarily entail increased A-fecundity of *all* of the organisms (or A-genes). In Cosmos and similar systems, however, each program as a whole is an encoding of a self-replication algorithm. As such, it can only replicate *as a whole*; the subsections of a program are highly epistatic, so evolution cannot in general vary one part of the program independently of other parts while retaining the functionality of the whole. Furthermore, replication is asexual, so there is generally no exchange of genetic material between individuals when an offspring is produced. It is therefore reasonable, at least to a first approximation, to treat the whole program as a single A-gene and to expect programs as a whole to evolve along these three axes as described above.

A number of measures were therefore chosen to track changes in each of these three factors through an evolutionary run. The plotting technique used for each of these measures was as follows: for time slice windows of equal width from the start to the

[1] Precisely, what matters is the rate at which *selectively significant* errors occur, i.e. errors which exclude the offspring from that particular S-lineage.

end of the run, we plotted the value of the measure for programs that died within that time slice window. For the plots for all three of these factors the data is pruned by only plotting values for individual programs of types which achieved a concentration of at least five individuals at some time during the run (as determined by the parameter `species_count_threshold_for_recording`). The data was further pruned by only recording information for 1 in every 50 eligible programs (as determined by the parameter `morgue_record_period`). In the plots, the darkness displayed at any point reflects the number of individual programs taking that particular value at that particular time (i.e. the more programs, the darker the plot).

For *A-longevity*, we looked at the age at death of each program. An example plot can be seen in Figure 5.7.

For *A-fecundity*, we looked at two measures: the number of time slices between the first and second successful replication of each program (the *replication period*) (this could obviously only be applied to programs that successfully replicated at least twice in their lifetime), and the length of programs. The length of a program is an indirect measure of fecundity—all things being equal, a longer program will take longer to replicate than a shorter program, as the longer one has to copy more instructions. An example plot for replication period can be seen in Figure 5.4, and for length, in Figure 5.2.

For *A-fidelity*, we looked at two measures: the flaw rate,[2] and the proportion of the total number of offspring produced by an individual that were unfaithful (i.e. less than 100% accurate). Example plots of these two measures can be seen in Figures 5.6 and 5.11 respectively.

In addition to the measures of A-longevity, A-fecundity and A-fidelity, we also used a visualisation technique developed by Mark Bedau and colleagues [Bedau & Brown 97]. The fundamental idea behind this technique is:

> "to identify those genotypes[3] that make a difference in the evolutionary process. Generally, we consider a genotype to 'make a difference' if it continues to be active in the evolving system... In [Tierra-like models] the

[2] Actually, we looked at the inverse of flaw rate, the flaw period. This is the expected number of successful instruction executions in a program between successive flaws.

[3] Note we are looking at genotypes here, i.e. groups of identical programs, rather than individual programs.

relative adaptive significance of a genotype is reflected by its concentration in the population. Relatively well adapted genotypes will have a relatively high concentration in the population, and relatively poorly adapted genotypes will we correspondingly scarce. Thus, we here define the [*cumulative evolutionary activity counter*] $a_i(t)$ of the i^{th} genotype at time t as its concentration integrated over the time period from its origin up to t, provided it exists:

$$a_i(t) = \begin{cases} \int_0^t c_i(t)dt & \text{if genotype } i \text{ exists at } t \\ 0 & \text{otherwise} \end{cases}$$

where $c_i(t)$ is the concentration of the i^{th} genotype at t. A genotype's [cumulative evolutionary activity counter] reflects its adaptedness (relative to the other genotypes in the population) throughout its history in the system." [Bedau & Brown 97][4]

To summarise the evolutionary activity of all the genotypes throughout the history of evolution in the system, we proceed as follows:

"The values of the activity counters of each [genotype] in the system over all time can be collected in the *component activity distribution*, $C(t, a)$, as follows:

$$C(t, a) = \sum_i \delta(a - a_i(t)),$$

where $\delta(a - a_i(t))$ is the Dirac delta function, equal to one if $a = a_i(t)$ and zero otherwise. Thus, $C(t, a)$ indicates the number of [genotypes] with activity a at time t." [Bedau *et al.* 98]

The visualisation technique is simply to graph these component activity distribution functions. This can be simplified by just plotting a point in (t, a) space whenever $C(t, a) > 0$.

These activity distribution functions (also referred to as "activity wave diagrams") provide a concise visual record of the appearance, competition and death of genotypes throughout an evolutionary run. Bedau and Brown also discuss how they can be interpreted in more detail, to reveal features such as periods of random drift among

[4] The rewording at the indicated places in this extract is to make it consistent with later changes in the terminology that Bedau *et al.* use to describe these techniques.

selectively neutral variants [Bedau & Brown 97]. Example activity wave diagrams can be seen in Figures 5.13–5.15.

The activity wave diagrams indicate which genotypes played important roles during the evolutionary run. The final individual-based analysis technique we used was to actually look at the code of significant genotypes, and to make comparisons between them. In particular, by looking at the genotypes which were abundant at the end of a run and comparing them to the genotype of the ancestor, we can get some idea of which parts of the ancestor were vital for its reproductive success (and therefore were more or less unchanged at the end of the run), and which parts were more redundant or less efficient (and how evolution managed to improve them).

5.1.2 Visualisation of Spatial Distributions

A number of 'movies' were recorded during the runs to show how the spatial distribution of the programs changed over the run. Each movie recorded a different piece of information about each program in the population (as well as the program's position on the grid), at various times throughout the run. This information included the program's length, age, size (number of cells, to indicate whether the program was serial or parallel), whether it had executed any foreign code during its lifetime (imported into its Received Message Store), whether it had moved during the last time slice, the amount of energy in its Energy Store, and, for each grid position, the amount of free environmental energy available at that time slice. By replaying these movies, we can see how the spatial distribution of these measures changed throughout the run.

5.1.3 Population-based Measures

In addition to the individual-based measures, a number of collective measures were also used. The population size throughout the run was recorded, as was the population diversity (the number of *different types* of program—different genotypes—in the population).

Four summary measures based upon the component activity distribution function, described in the previous section, were also used: Cumulative Activity, Mean Cumulative Activity, New Activity and Mean New Activity. These measures were developed by

Mark Bedau and colleagues ([Bedau & Packard 91], [Bedau *et al.* 97], [Bedau *et al.* 98]) to highlight between them the significant adaptive events occurring during an evolutionary run.

Cumulative Activity, $A_{cum}(t)$, is a measure of the continual adaptive success of the genotypes in the system at a given time. It is defined at time t as the sum of the activity counters, $a_i(t)$, of all genotypes (i) in the population at that time. The Mean Cumulative Activity ($\bar{A}_{cum}(t)$, defined as $\frac{A_{cum}(t)}{D(t)}$, where $D(t)$ is the diversity—the number of *different* genotypes—at time t) is the cumulative activity per genotype.

New Activity, $A_{new}(t)$, is a measure of the rate at which *new* adaptively significant genotypes are appearing in the system at time t. It is defined as follows:

> "Adaptive innovations correspond to new components flowing into the system and proving their adaptive value through their persistent activity. Let a_0 and a_1 define a strip through the component activity distribution function, $C(t, a)$, such that activity values a in the range $a_0 \leq a \leq a_1$ are among the lowest activity values that can be interpreted as evidence that a component has positive adaptive significance. Then, one reflection of the rate of the evolution of adaptive innovations is the new evolutionary activity, $A_{new}(t)$, which sums the evolutionary activity per component with values between a_0 and a_1:
>
> $A_{new}(t) = \sum_{i, a_0 \leq a_i(t) \leq a_1} a_i(t)$" [Bedau *et al.* 98][5]

Finally, Mean New Activity, $\bar{A}_{new}(t)$, is the new activity per genotype, defined as $\frac{A_{new}(t)}{D(t)}$. For more discussion of these measures, and of the reasons for defining them as they are, see [Bedau *et al.* 98] and [Bedau & Packard 91].

[5] Again, this formula has been changed from the original text to make the notation more consistent. In the original paper, $A_{new}(t)$ is defined as

$$A_{new}(t) = \frac{1}{D(t)} \sum_{i, a_0 \leq a_i(t) \leq a_1} a_i(t)$$

(i.e. as above, but divided by $D(t)$). This is what will be referred to here as *Mean* New Activity, $\bar{A}_{new}(t)$.

5.1.4 A Neutral Shadow

In order to gauge the extent to which the observed dynamics of Cosmos are due to the adaptive properties of individual genotypes rather than to any other features of the system, we also ran a "neutral shadow" of the system (as suggested by Bedau et al. [Bedau *et al.* 98], [Bedau & Brown 97], [Bedau *et al.* 97]).

The neutral shadow works as follows. During a standard run of the system, various details are recorded at each time slice (if the parameter `record_neutral_model_data` is set appropriately). These details include the number of organism births during that time slice, the number of these new births which resulted in new genotypes (through the action of flaws or mutations), the number of cell divisions (to form parallel processes), the number of cell deaths, and any movements made by individual cells.

The idea of the neutral shadow is to re-run the system using this recorded data, but in such a way that that no particular organism or genotype has any adaptive advantage over any other. Nominal "programs" exist at grid locations, reproduce, and die. At each time slice, a number (specified by the data recorded from the real run) of these programs are chosen *at random* to reproduce and to mutate. (Cell divisions in the real run to form parallel programs are treated as reproductions in the neutral shadow, so that all programs are unicellular.)[6] Similarly, a number of cells are chosen at random to be killed, and to move around the grid, determined by the numbers recorded from the standard run. In this way, we get the same number of events occurring in the neutral shadow as in the standard run, but these events happen to randomly chosen individuals. By applying the same measurement techniques to the neutral shadow as we do to the standard run, we can get an idea of the "raw" dynamics of the system, independent of the adaptive success of any particular genotype.

The results of the neutral shadow are discussed alongside the results of the standard run in the following sections.

[6] Therefore, at any given time, the number of *cells* in the neutral shadow is always the same as in the standard run, although the number of *organisms* may be lower if the standard run contains some multicellular (parallel) programs.

5.2 Detailed Analysis of a Standard Run

In this section, the results of a single run of Cosmos using the default parameter settings, together with its neutral shadow, will be analysed in some detail.

As mentioned in the previous chapter, there are many parameters associated with Cosmos, some of which control fairly minor or obscure aspects of the system. Because of this, it would be impractical to conduct a thorough investigation of the entire parameter space. Instead, many of the more obscure parameters were held constant throughout all of the experiments reported in this chapter. These are listed in Section B.1, along with their associated values.

The parameters which were varied between experiments (together with some other major parameters such as `grid_size`) are listed in Table 5.1. This table also shows the default values of these parameters, as used in the run reported in this section.

As shown in the table, the run was initialised by inoculating 100 ancestor programs into the system. The size of the world was 40 x 40 squares, and the ancestors were distributed evenly throughout this area.

The ancestor used for this run (and for nearly all of the other runs) is shown schematically in Figure 5.1 (a full listing is given in Section B.2). The basic operation of the ancestor is as follows. First, it looks for a template pattern at the start of its genome to work out the memory address of its first line of code. Next, it searches for a template pattern at the end of the genome, to work out the address of its last line of code. The ancestor then enters into a loop, copying instructions one at a time, from the beginning of the genome to the end, into the Nucleus Working Memory. When this copying is complete, the cells divides (creating a new organism), and the ancestor attempts to move into a neighbouring grid position.

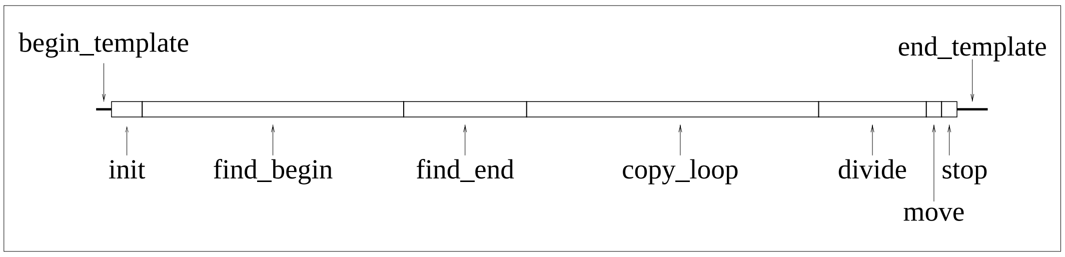


Figure 5.1: Schematic of the Standard Ancestor, 348AAAA.

Parameter	*Value*
inoculation	
ancestor	user_defined
number	100
startinfo	
rng_seed	834981793
termination	
limited_run	yes
number_of_timeslices	1000000
environment	
grid_size	40
horizontal_wrap	yes
vertical_wrap	yes
max_cells_per_process	2500
number_of_energy_tokens_per_grid_pos_per_sweep	30
max_energy_tokens_per_grid_pos	100
energy_collection_scheme	shared
energy_distribution_scheme	land
x_delta	0.0
cell	
et_value_constant	0.025
et_value_power	1.0
number_of_energy_tokens_per_collect	10
max_energy_tokens_per_cell	100
neighbouring_genomes_readable	no
mutation	
apply_mutations	yes
mutation_application_period	10
mutation_period	1000000
apply_flaws	yes
default_flaw_period	250000
io	
record_neutral_model_data	yes
visualisation_recording_on	yes
visualisation_record_energy_only	no

Table 5.1: Default Values for Major Parameters in Cosmos.

The duration of the run was one million time slices. As discussed in Section 5.1.4, data for the neutral shadow was recorded during this run, and then fed into the neutral model to give us an idea of the non-adaptive characteristics of the system.

5.2.1 Program Length

The change in program length over the run is shown in Figure 5.2. A number of interesting points are apparent from this graph. The most obvious one is that there is a trend for increasing program length over the course of the run. The length of the ancestor programs was 348 bits, but by the end of the run, virtually all of the programs in the population were of length 390 bits.[7] Recalling from the previous chapter that each instruction is represented by 6 bits, this increase in length corresponds to an extra 7 instructions being added to the ancestor.

Another point of interest in the graph is that, at any given time during the run, the population tends to be composed of organisms of only a single length.[8] The only exception to this occurs during a transition phase from programs of one length to programs of a greater length. During these phases, progams of two, or, occasionally, three different lengths coexist, but the shorter programs soon die out, to be completely replaced by the longer ones.

This result is in contrast to those of similar runs on Tierra, where shorter "parasite" programs evolved and coexisted with the longer programs (as described in Section 3.2.1). As discussed in Section 3.3.1, the non-appearance of parasites was expected, because Cosmos does not allow cells to execute the code of other cells.

The increase in program length, on the other hand, was unexpected. During each timeslice, CPU-time was allocated to each program (if it had enough energy) according to its length, so that, all else being equal, selection was size neutral (see Section 4.2.3). However, the picture is complicated by the fact that the program has to cycle once round the "copy loop" to copy each instruction. In the ancestor, the copy loop accounts for 19 of the 58 instructions, or approximately one third of the total number. The total

[7] There were in fact a handful (7) of programs of different lengths, but these are not recorded in Figure 5.2, because of the data pruning methods described in Section 5.1.1.

[8] Again, the data pruning techniques discussed in Section 5.1.1 mean that information is not displayed in these graphs for genotypes that are represented by only a very few individual programs. Such individuals typically account for about 1% of the total population.

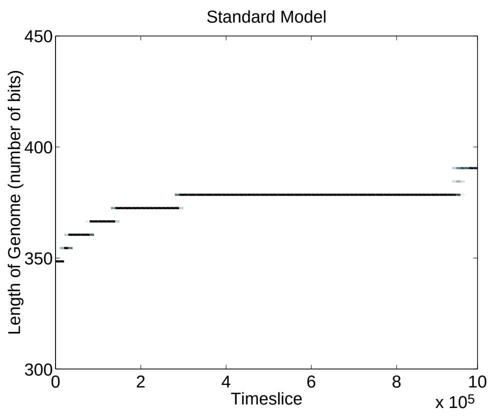


Figure 5.2: Standard Run: Length.

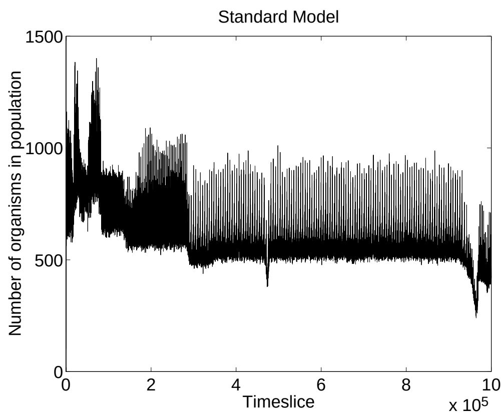


Figure 5.3: Standard Run: Population Size.

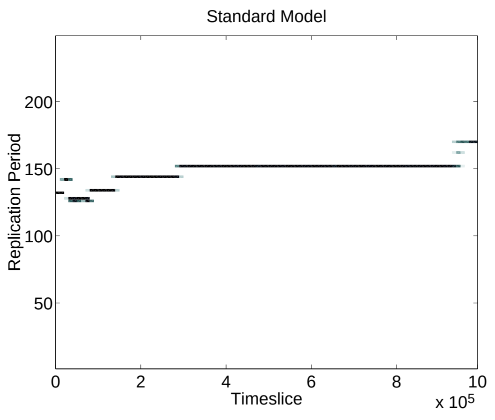


Figure 5.4: Standard Run: Replication Period.

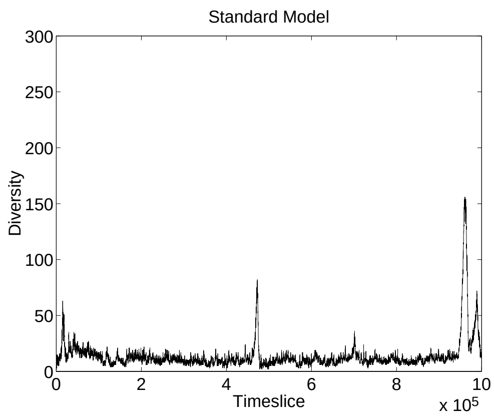


Figure 5.5: Standard Run: Diversity.

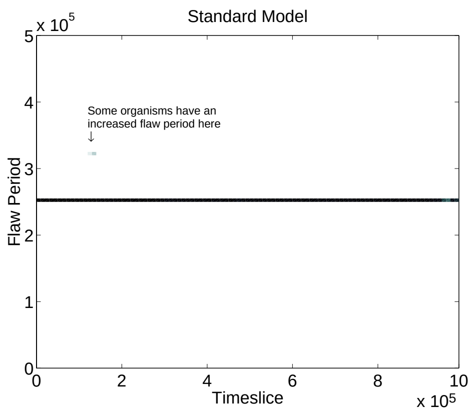


Figure 5.6: Standard Run: Flaw Period.

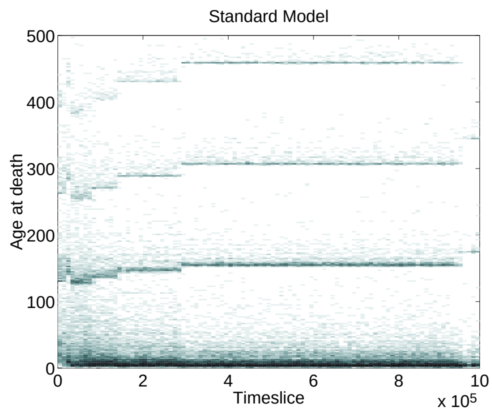


Figure 5.7: Standard Run: Age at Death.

number of instructions, I, that must be executed to replicate the program is therefore approximately:

$$I = N_{non\_loop} + N_{loop} * N_{total} \tag{5.1}$$

where N_{non_loop} is the number of instructions not involved in the copy loop ($58 - 19 = 39$), N_{loop} is the number of instructions in the copy loop ($= 19$), and N_{total} is the total number of instructions ($= 58$). In this case, I is therefore $39 + 19 * 58 = 1141$ instructions. (This is not exactly right, because there are some conditional instructions, and the templates at the beginning and end of the program do not actually get executed, but it is near enough.) Now, during this run, the parameter `et_value_constant` was set to 0.025, so at each timeslice the ancestor (of length 58 instructions, or $58 * 6 = 348$ bits) was allowed to run $0.025*348 \simeq 8$ instructions (rounded down to a whole number). The total number of timeslices required for an ancestor program to replicate itself is therefore approximately $1141 \ / \ 8 \ \simeq \ 142$ timeslices.

Similar calculations can be made for programs of different lengths. If we assume that the size of the copy loop is always one third of the size of the whole program, the results we get are as shown in the left-hand side of Figure 5.8. In this case, longer programs still take longer to replicate, so we would still expect selection for shorter programs. (The graph looks somewhat erratic because of the effects of rounding down the number of instructions per time slice to an integer value, etc.) If however we assume that the size of the copy loop is constant no matter how long the program, the results are somewhat different (see the right-hand side of Figure 5.8). In this case, the assumption of length-neutral selection is more realistic.

As an increase in program length was observed, we might therefore expect the extra instructions to have been added outside of the copy loop. However, as the allocation of CPU-time is also dependent on a program having enough stored energy, it is possible that the observed increase in program length might be associated with the evolution of programs that collect and store more energy from the environment. In this case, it might even benefit programs to add more energy collection instructions inside the copy loop, as each one will then be executed during every iteration of the loop. The true cause of the increase in program length will be investigated in Section 5.2.7.

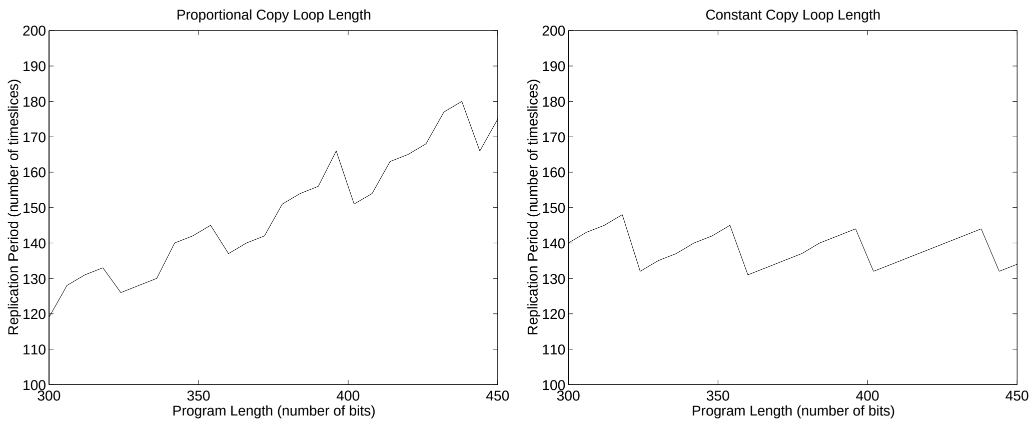


Figure 5.8: Theoretical Relationship between Program Length and Replication Period. *Left*: Size of copy loop increases proportionally with size of program. *Right*: Size of copy loop is constant.

5.2.2 Replication Period

The replication period (defined for programs that replicated faithfully at least twice during their lifetime as the number of time slices between the first and second faithful replication) is a more direct measure of a program's fecundity than is program length. From Figure 5.8 we can see that, if the length of the copy loop is increasing in proportion to the length of the program as a whole, we would expect trends in replication period during a run to roughly track trends in program length (because, generally, longer programs take a longer time to replicate in this case). However, if the length of the copy loop remains fairly constant as the program length as a whole increases, we would not expect such an association.

The observed change in replication period over the run is shown in Figure 5.4. As can be seen, the general pattern is very similar to that of change in program length (Figure 5.2), which suggests that the length of the copy loop is increasing as the programs as a whole get longer. This, together with the observation that the replication period increases during the run, suggests that there is some other factor playing an important role in determining the fitness of the programs. As suggested earlier, this factor could be a program's ability to collect energy. We will look into this in Section 5.2.7.

5.2.3 Age at Death

The graph of the age at death of programs in the population should show us whether they are evolving along the longevity axis. Figure 5.7 shows the graph for this run. There are a number of interesting points to note about this graph. First, there does appear to be some evolution towards increased longevity. On top of this, there is considerable structure in the distribution of ages at which organisms die. This is interpreted as indicating that the cycle of births and deaths in the population is highly synchronised throughout the run. The figure shows that the majority of programs live for some multiple of roughly 130 time slices at the beginning of the run, with fewer programs surviving for each successive multiple. At the end of the run, we still see the periodic structure, but the period has increased to about 160 time slices. These period lengths correspond very well with the replication period of the programs (see Figure 5.4).

A likely explanation is that the observed periodic structure arises as a consequence of the population experiencing a ceiling effect. Each time the population size reaches the ceiling, a number of programs die, creating space for the remaining programs to reproduce. Once this reproduction stage occurs, the population size is soon at the ceiling again, so the cycle repeats. The extinctions triggered by the population size hitting the ceiling are therefore periodic, resulting in the observed distribution of ages, with most organisms surviving for an integral multiple of the period of this cycle. The maximum number of programs (more accurately, cells) allowed to coexist in this run was 2500 (specified by the parameter `max_cells_per_process`). The actual population size never approached this limit (see Figure 5.3, discussed in the next section), so this was not the source of the ceiling effect. The other possible limiting factor is the total amount of energy distributed in the environment at each time slice. As discussed in the next section, it seems likely that this factor did indeed create a ceiling effect on population size, and therefore explains the observed distribution in Figure 5.7.

5.2.4 Population Size and Diversity

The graphs of population size and of diversity are shown in Figures 5.3 and 5.5 respectively. No multicellular (parallel) programs evolved during the run, so the number of cells in the population was always equal to the number of organisms, as shown in Figure 5.3.

One of the points of interest in the population size graph, as mentioned in the previous section, is the fact that the limit on the number of cells in the population imposed by the parameter `max_cells_per_process` was never reached. This suggests that the population size was instead being limited by the amount of energy available in the environment. Figure 5.3 also shows a very high frequency oscillation in the population size, convolved with the lower frequency trends. This is consistent with the explanation offered in the previous section, that the birth and death cycle in the population was highly synchronised, producing a pattern of growth and decline determined by a ceiling population size related to the total amount of energy available in the environment.

To check whether the energy available in the environment really did impose a population ceiling lower than that specified by the parameter `max_cells_per_process`, a number of test runs were conducted. The only difference between the runs was the number used to seed the random number generator at the start of each one. Nine runs were conducted. In these runs, mutations and flaws were switched off, so that no evolution occurred. This allows us to see how the population size varies in a population of the ancestor programs, divorced from any effects due to competition from newly evolved programs. The graph of population size for one of these runs (which all lasted for 100,000 time slices), is shown in Figure 5.9. The figure clearly shows that the maximum population size achievable in practice is indeed well below the limit of 2500 set by `max_cells_per_process`; in fact, it is less than half of this value. Even with mutations and flaws switched off, slight differences arose between the different runs because, at each time slice, the programs in the population are processed in random order. However, the corresponding graphs for the other runs were all very similar; the maximum population size reached at any point in any of the runs was 1280 individuals.

Looking at the larger trends in the population size, we can see that they seem to be negatively correlated to the trends in replication period (Figure 5.4). This observation is consistent with the earlier suggestion that the programs are getting longer because they are collecting more energy. The more energy each program stores on average, the less is available in the environment for use by other programs, so the lower the maximum population size achievable. We will look into the question of whether the

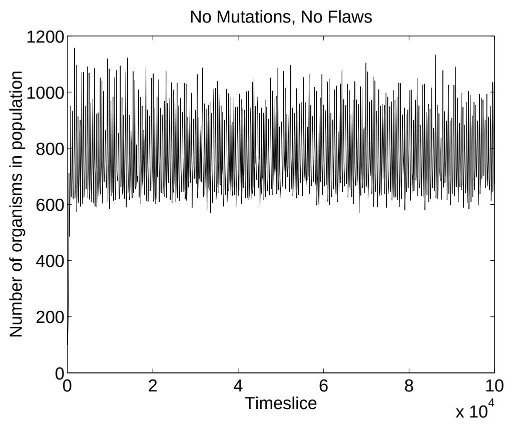


Figure 5.9: Population Size Limit Imposed on Ancestor Programs by the Default Rate of Introduction of Energy into Environment.

programs were evolving to collect more energy in Sections 5.2.7 and 5.2.8.

Looking at the diversity of the population throughout the run (the number of different types of program at any given point), we see that it is generally very low (Figure 5.5). Examination of the activity wave diagram (Figure 5.13, discussed in Section 5.2.6) reveals that there there is usually just one dominant genotype in the population at any given time (except at the very beginning of the run), and the concentration of other varieties is low.

There are interesting spikes in the diversity graph, one at roughly time 473,000, and another at 960,000. These spikes immediately precede dips in the population size (Figure 5.3). Inspection of the raw data files produced during the run reveals that these spikes correspond to the appearance of a number of longer genotypes, of lengths mostly in the region of 500–600 bits. None of these was ever represented by more than a single program in the population, suggesting that these were not viable self-replicators. As the numbers of these programs never exceeded the threshold defined by the system parameter `species_count_threshold_for_recording`, they do not show up in Figure 5.2 (as discussed in Section 5.1.1). As no further data was recorded for these programs, not much more can be said about them. However, it is reasonable to guess that they stored large amounts of energy, leading to the observed drop in population size.

To get an idea of the extent to which the diversity of the population is due specifically to the adaptive success of the individual genotypes rather than to other non-adaptive

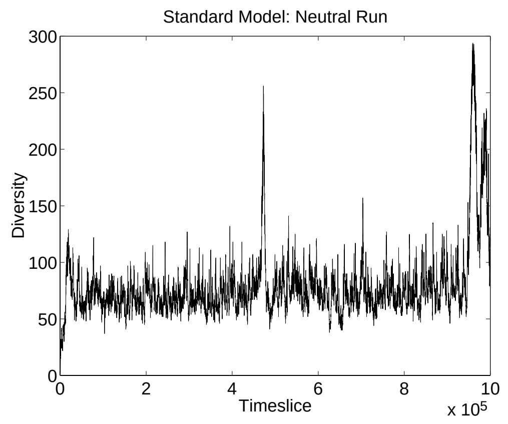


Figure 5.10: Standard Run, Neutral Shadow: Diversity.

dynamics in the system, we can look at the diversity graph of the neutral shadow. (Recall that the neutral shadow replays the exact sequence of births and deaths recorded in the real run, but acts upon random individuals. The population size in the neutral shadow is therefore always identical to the real run.) The neutral diversity graph is shown in Figure 5.10. This graph shows that the diversity in the neutral shadow was considerably higher than in the standard run, which suggests that the low diversity observed in the standard run was indeed due to the adaptive success of one or a small number of genotypes at outcompeting other varieties.

5.2.5 Flaw Period and Proportion of Unfaithful Replications

The two measures used to track evolution of the programs along the A-fidelity axis were: the flaw period of individual programs, and the proportion of the total number of offspring produced by an individual that were unfaithful (less than 100% accurate). The corresponding graphs are shown in Figures 5.6 and 5.11 respectively. In the graph of unfaithful replications (Figure 5.11), the vast majority of the points lie at zero, indicating that nearly all programs which reproduced did so faithfully. In order to see whether there was any trend for programs which did *not* always reproduce faithfully, the data from Figure 5.11 is reproduced in Figure 5.12 with the exception that points lying at zero infidelity have been omitted. The bar on the right of these two figures shows the scale (i.e. the mapping between the darkness of the plot at any point in the graph and the number of programs which had that particular infidelity at that particular time). Note that the scale in Figure 5.12 is two orders of magnitude smaller

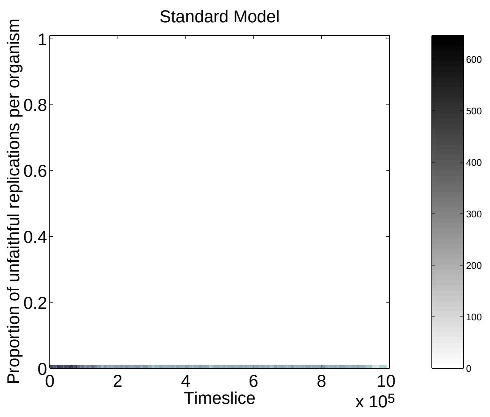


Figure 5.11: Standard Run: Unfaithful Replications.

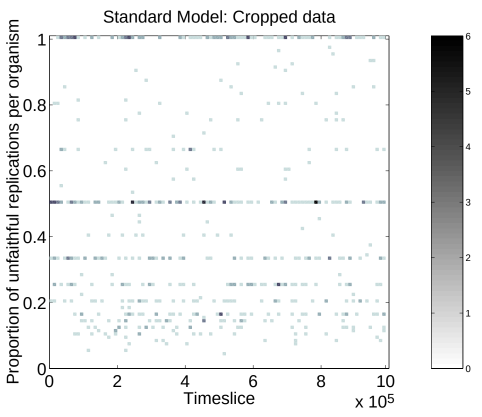


Figure 5.12: Standard Run: Unfaithful Replications (omitting points with zero infidelity).

than in Figure 5.11.

It is clear from these graphs that there was little change in either of these measures throughout the run. In other words, the programs are not evolving along the A-fidelity axis. From the analysis in the previous sections, it would appear that the parameter choices for this run create a selection pressure that predominantly favours evolution along the longevity and fecundity axes.

5.2.6 Activity Measures

The activity wave plots for the standard run and its neutral shadow are shown in Figures 5.13 and 5.14 respectively. These two figures are plotted to the same scale. A magnified version of the neutral shadow plot in shown in Figure 5.15.

The first point to note is that in the neutral shadow, compared to the standard run, there are *no* significant waves at all. In other words, no individual genotypes achieved significant continued adaptive success in this run. Even if we look at the magnified plot of the neutral shadow (Figure 5.15), we see that the pattern of activity is very different to the standard run (Figure 5.13). In the neutral shadow, there are many more activity waves than in the standard run, but they generally survive for a much shorter duration. These observations were expected (because operations are executed on randomly-chosen individuals in the neutral shadow), and they suggest that the significant activity observed in the standard run (Figure 5.13) *is* due to the adaptive success of the individual genotypes concerned.

Looking at the activity waves of the standard run, we see that the ancestor genotype (348AAAA) is fairly quickly replaced by other genotypes. The first variant to enjoy notable continued adaptive success is genotype 366AADW, which first appears at time 76,100. At time 130,100 a new genotype, 372AAFT, appears, and gradually outcompetes 366AADW. By time 146,400 only a single program of genotype 366AADW remains, although this program manages, somewhat impressively, to survive for a further 91,700 time slices, living right up to time 238,100. At time 274,700, genotype 378AALB appears on the scene. This genotype quickly displaces 372AAFT, and remains the dominant genotype in the population right up until time 955,100, under 5000 time slices short of the end of the run. During its long reign, 378AALB is challenged a small number of times, most notably by the variants 378ANGE and 378ASNT. These variants are both of length equal to the dominant genotype, and both eventually lose their challenge and are driven to extinction. Having seen off these challenges, genotype 378AALB remained the dominant type of program for a short while longer. However, at time 925,700 a new genotype, 390AAGX, is born. Within 25,000 time slices, it drives 378AALB to extinction, and at the end of the run 390AAGX is the final dominant genotype.

The graphs of Cumulative Activity, Mean Cumulative Activity, New Activity and Mean New Activity, for the standard run and its neutral shadow, are shown in Figures 5.16–5.23.

Looking at the graph of Cumulative Activity in the standard run (Figure 5.16), we can see that it reflects the shape of the activity wave plot (Figure 5.13). This is because the value plotted in Figure 5.16 is the sum of the activities of all the individual genotypes at a given point in the run (i.e. the sum of all the waves present at any given moment in Figure 5.13). As a single wave tends to dominate the activity wave plot at most points during the run, the activity value of that genotype largely determines the shape of Figure 5.16. The diversity of genotypes in the standard run is usually very low (Figure 5.5), so the graph of Mean Cumulative Activity (defined at any given time as Cumulative Activity divided by Diversity), also has a similar shape (Figure 5.18).

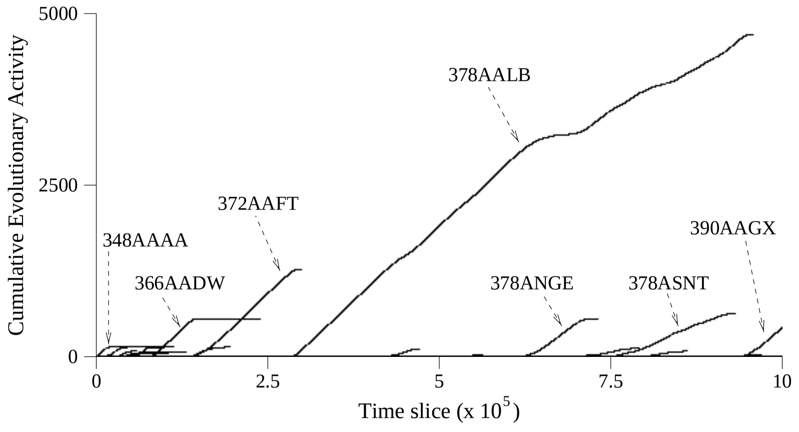


Figure 5.13: Standard Run: Activity Waves Plot.

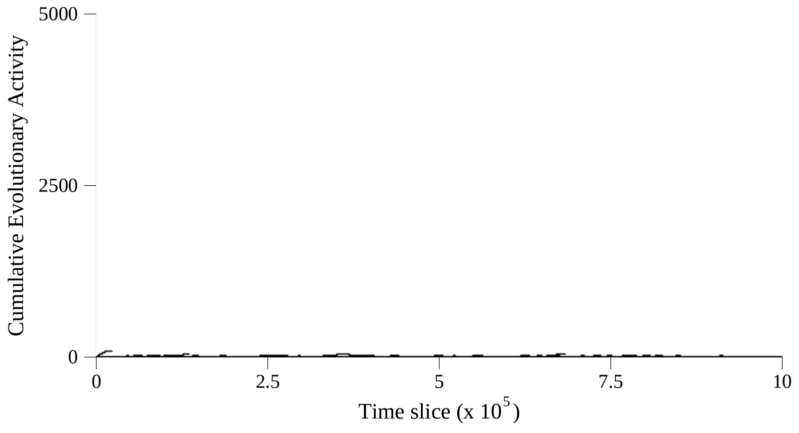


Figure 5.14: Standard Run, Neutral Shadow: Activity Waves Plot.

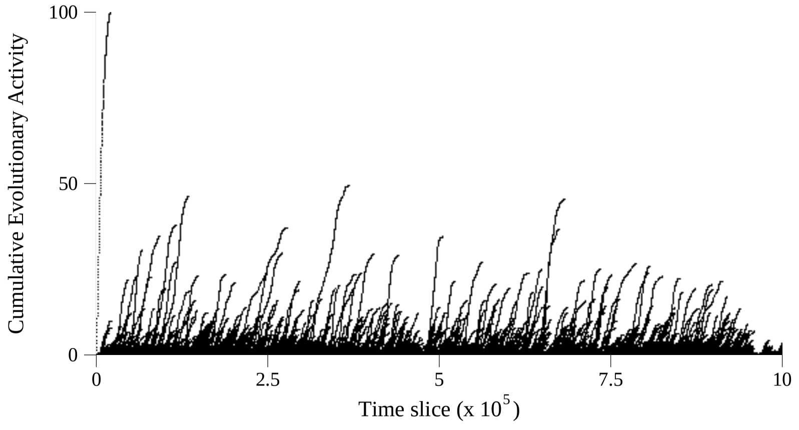


Figure 5.15: Standard Run, Neutral Shadow: Activity Waves Plot (magnified).

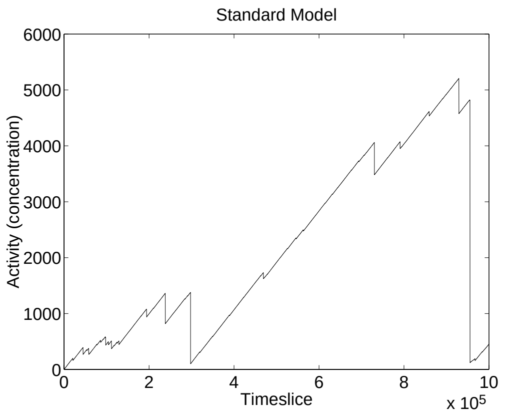


Figure 5.16: Standard Run: Cumulative Activity.

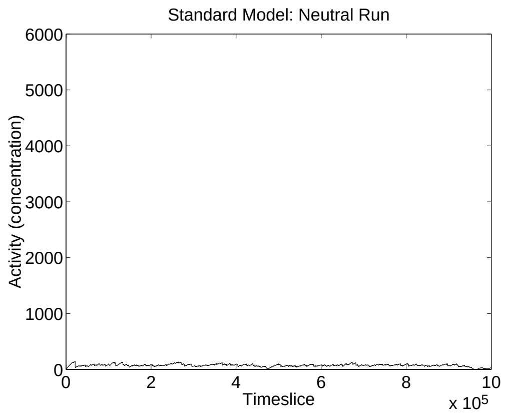


Figure 5.17: Standard Run, Neutral Shadow: Cumulative Activity.

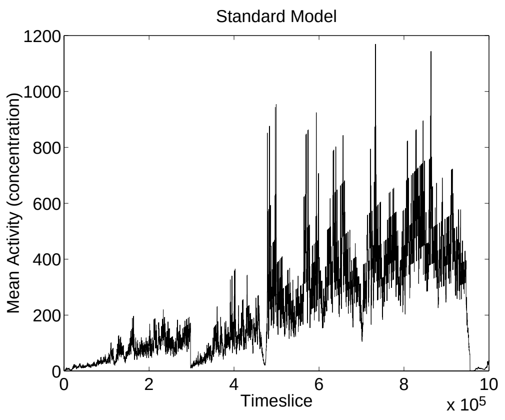


Figure 5.18: Standard Run: Mean Cumulative Activity.

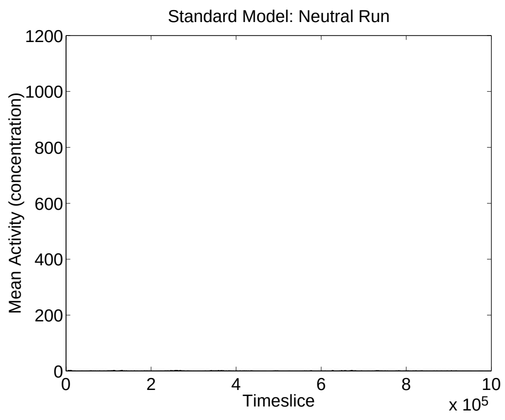


Figure 5.19: Standard Run, Neutral Shadow: Mean Cumulative Activity.

The graphs of Cumulative Activity and Mean Cumulative Activity for the neutral shadow (Figures 5.17 and 5.19) show again (as we saw in the activity wave plot of Figure 5.14) that there was no significant cumulative evolutionary activity in the neutral shadow compared to the standard run.

The graphs for *New* Activity, reflecting the introduction of new genotypes with adaptive significance, are shown in Figures 5.20–5.23. The parameter a_0, which determines the lowest level of activity that is taken to indicate adaptive significance, was set at the lowest level such that the neutral shadow produced zero new activity throughout the run. The value chosen for a_0 was 150, and for the upper activity limit, a_1, the value chosen (somewhat more arbitrarily) was 300. When applying the same thresholds to the standard run (Figures 5.20 and 5.22), we can see that this measure clearly indicates the points during the run at which significant new genotypes were introduced (c.f. Figure 5.13).

In [Bedau *et al.* 98], a scheme is presented for classifying evolutionary systems into (at least) three distinct classes, depending on whether adaptive evolutionary activity is absent (class 1), bounded (class 2), or unbounded (class 3). The classification of a system is determined by its diversity D, mean cumulative activity $\bar{A}_{cum}$, and mean new activity $\bar{A}_{new}$.[9] If the system has bounded D, zero $\bar{A}_{cum}$ and zero $\bar{A}_{new}$, it belongs to class 1 (no adaptive evolutionary activity). If it has bounded[10] D, bounded $\bar{A}_{cum}$ and positive[11] $\bar{A}_{new}$, it belongs to class 2 (bounded adaptive evolutionary activity). If it has unbounded D, bounded $\bar{A}_{cum}$ and positive $\bar{A}_{new}$, it belongs to class 3 (unbounded adaptive evolutionary activity). Bedau and colleagues claim that the biosphere (at least with consideration to the Phanerozoic fossil record), belongs to class 3, whereas all artificial evolutionary systems they have studied belong to classes 1 and 2. From this perspective, a major challenge for the field of artificial life is to produce an artificial evolutionary system which exhibits class 3 dynamics.

This classification scheme is not perfect, as we can never be sure whether the results of a limited run of a system (which might have led us to suppose, for example, that D is bounded) will hold true for a much longer run (e.g. D might actually turn out to be unbounded). However, it is at least a step in the right direction, towards a rigorous classification of evolutionary dynamics. According to the scheme, and based upon the results reported in this section, Cosmos would be classified as class 2—the same as various other artificial evolutionary systems. Despite its weaknesses, the scheme does at least indicate that the type of evolution evidenced by the fossil record may be

[9] As explained earlier, Bedau calls this measure simply the new activity, A_{new}, rather than mean new activity. The name has been changed here to make it consistent with the naming of other measures.

[10] The definition of (un)boundedness given in [Bedau *et al.* 98] is: The function $f(t)$ is unbounded *iff*

$$\lim_{t\to\infty}\left(\frac{sup(f(t))}{t}\right) > 0$$

where $sup(.)$ is the supremum function.

[11] Similarly, the definition of a positive function is given in [Bedau *et al.* 98] as follows: The function $f(t)$ is positive *iff*

$$\lim_{t\to\infty}\left(\frac{\int_0^t f(t)dt}{t}\right) > 0$$

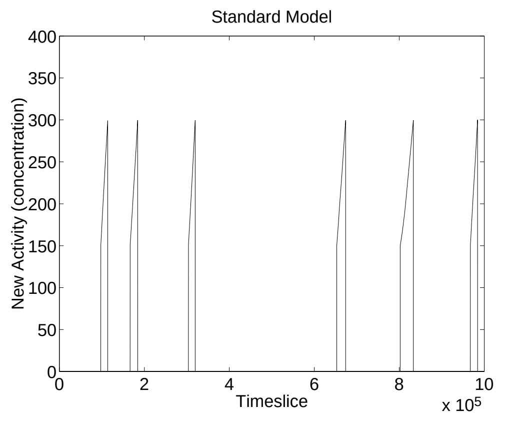


Figure 5.20: Standard Run: New Activity.

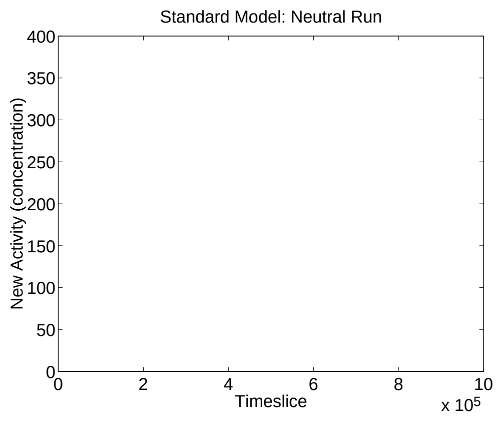


Figure 5.21: Standard Run, Neutral Shadow: New Activity.

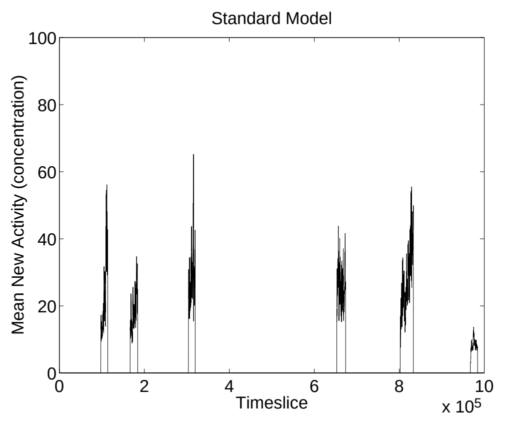


Figure 5.22: Standard Run: Mean New Activity.

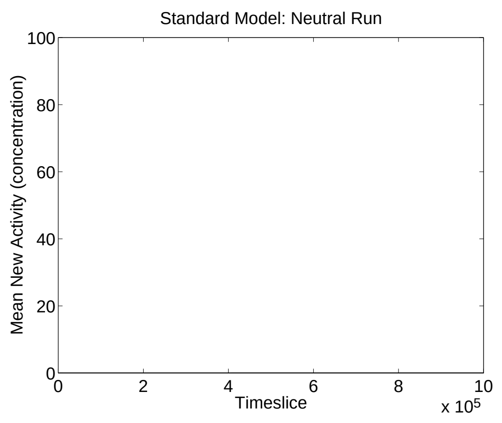


Figure 5.23: Standard Run, Neutral Shadow: Mean New Activity.

qualitatively different to that observed in our artificial systems.

5.2.7 Analysis of Significant Genotypes

By analysing the output data from the run, it is possible to reconstruct the phylogenetic tree (the evolutionary history) of the significant genotypes during the run. The reconstruction for the standard run is shown in Figure 5.24.

An interesting point to note about the phylogeny is that some of the ancestors of the genotypes that were significant throughout the course of the run did not themselves enjoy much success. An extreme example is genotype 360ABFL, from which all of the subsequent adaptively significant genotypes are descended. Despite its significance as

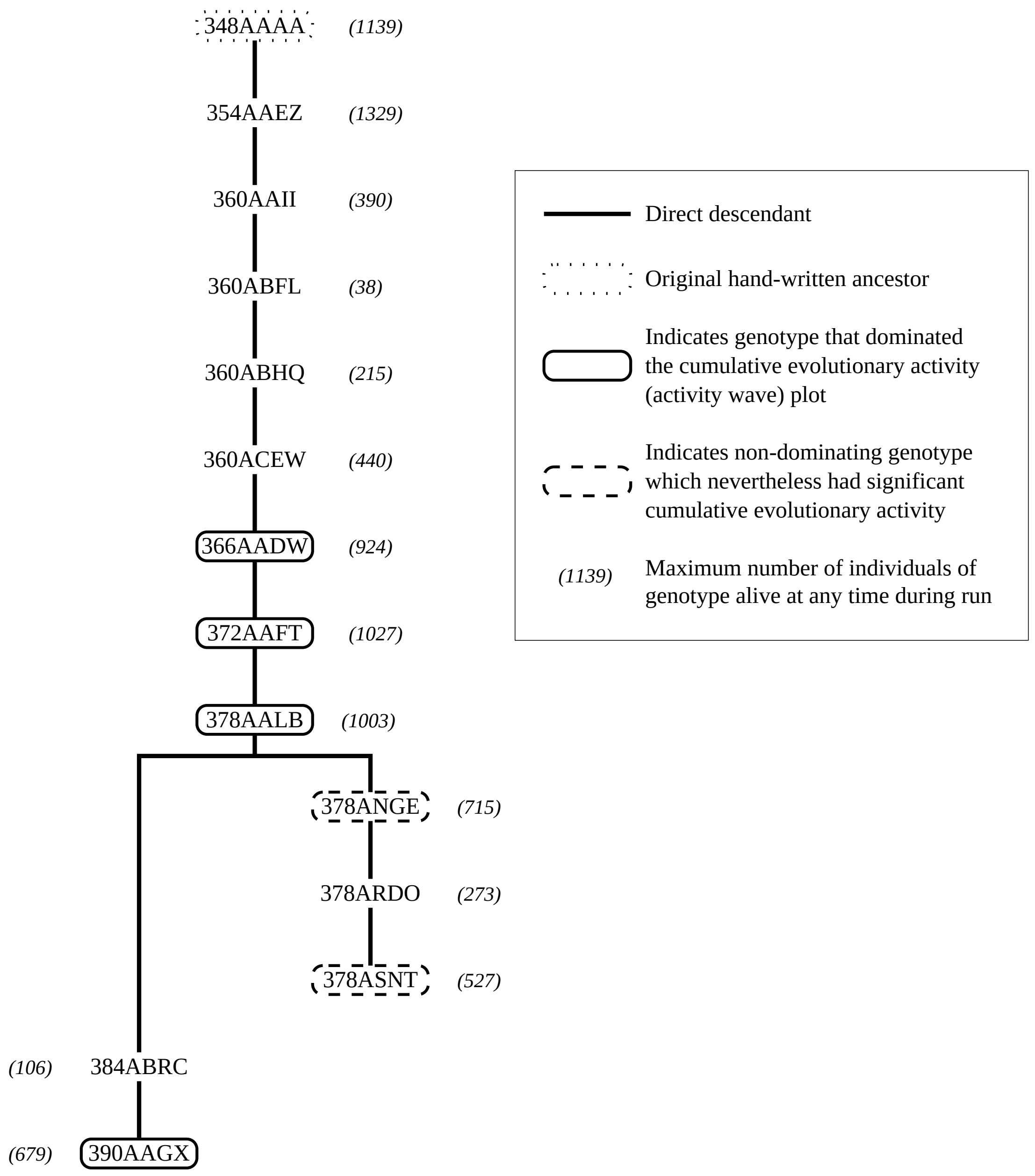

348AAAA (1139)
354AAEZ (1329)
360AAII (390)
360ABFL (38)
360ABHQ (215)
360ACEW (440)
366AADW (924)
372AAFT (1027)
378AALB (1003)
378ANGE (715)
378ARDO (273)
378ASNT (527)
(106) 384ABRC
(679) 390AAGX
Direct descendant
Original hand-written ancestor
Indicates genotype that dominated the cumulative evolutionary activity (activity wave) plot
Indicates non-dominating genotype which nevertheless had significant cumulative evolutionary activity
(1139) Maximum number of individuals of genotype alive at any time during run

Figure 5.24: Standard Run: Phylogeny of Significant Genotypes.

genotype	ancestor	begin		find_begin		find_end		copy_loop		divide		end	
		3		17		8		19		7		4	
		+	−	+	−	+	−	+	−	+	−	+	−
366AADW	348AAAA	0	0	0	0	0	0	3(3)	1	0	0	3(0)	2
372AAFT	366AADW	0	0	0	0	0	0	1(1)	0	0	0	0	0
378AALB	372AAFT	0	0	0	0	0	0	1(1)	0	0	0	0	0
378ANGE	378AALB	0	0	0	0	0	0	0	0	0	0	1(0)	1
378ASNT	378ANGE	0	0	0	0	0	0	0	0	0	0	2(0)	2
390AAGX	378AALB	0	0	0	0	0	0	2(2)	0	0	0	0	0

Each row presents data on the comparison of a genotype (first column) with its ancestor (second column). Columns 3–8 refer to subsections of the genome (c.f. Figure 5.1; the subsections labelled begin_template and init have been grouped into a section labelled begin here, and similarly the subsections move, stop and end_template have been grouped together as end). In the row below the subsection name is the length of that subsection (in number of instructions) in the original ancestor 348AAAA. The column marked + shows how many new instructions were *added* to this subsection during the evolution from the indicated ancestor. The column marked − shows how many instructions were *removed*. Numbers in braces () in the + column show how many of the added instructions were the energy collection instruction (`et_collect`).

Table 5.2: Location of Differences Between Significant Genotypes and Their Ancestors.

an ancestor, the number of programs of genotype 360ABFL existing at any one time was *never* more than 38 (in a total population of over 800), and the genotype only existed for about 4000 time slices in total. Had circumstances been slightly different, the genotype might have died off before giving rise to its descendant 360ABHQ, or indeed 360ABFL might never have arisen at all. This indicates that the role of chance might play a large part in determining the outcome of a single evolutionary run. We will look into this issue in detail in Section 6.1.

A summary of the sources of innovation from ancestor to descendent for the most significant genotypes is presented in Table 5.2. The table reveals that the only sections of the genome which vary between the genotypes are the copy_loop and end sections.

Looking at the changes in the copy_loop section, we see that *all* of the additions were extra `et_collect` instructions. Comparing the final dominant genotype 390AAGX with the original hand-written ancestor 348AAAA, *seven* extra `et_collect` instructions have been added within the copy_loop section. If we compare the other subsection that experienced changes, the end section, we see that it only actually grew by one instruction from 348AAAA to 390AAGX (there were three additions and two removals). These findings therefore confirm the prediction made in Sections 5.2.2 and 5.2.4 that the programs are evolving to collect more energy, and that the extra energy collection instructions are being placed within the copy loop, so that they are executed at each iteration of the loop.

The changes made in the `end` section included the deletion of the `move` instruction in genotype 366AADW (the genome of the ancestor 348AAAA is shown in Section B.2). None of the subsequent dominant genotypes re-inserted this instruction, so the ability to move around the grid apparently did not convey an adaptive advantage in this run. The `stop` instruction was also deleted (both this and the `move` instruction were replaced by instructions which had no effect in the program), with subsequent genotypes taking advantage of the fact that when program execution reached the final instruction in the genome, the `stop` instruction was implicitly executed anyway. The extra instruction added at the end of the `end` section (in the transition from genotype 348AAAA to 366AADW) was an extra copy of the final instruction in the program (which actually formed part of the end template). This extra instruction appeared as a consequence of the `if_not_fl` instruction in the copy loop being replaced by an `et_collect`. The removal of the `if_not_fl` meant that the program did not check whether it had already reached the position marked by the end template before copying the next instruction to its offspring, so the final section of the end template was copied twice before the copy loop was exited. Apparently, the inclusion of an extra `et_collect` in the copy loop outweighed the penalty of having an extra, redundant instruction at the end of the program.

5.2.8 Spatial Distribution

By replaying the 'movies' recorded during the run (Section 5.1.2) we can see how the spatial distribution of the programs, and of various specific aspects of the programs and the environment, changed throughout the run.

As the run settled down, the spatial distribution of the programs became more regular, with many programs placed such that all of their immediately neighbouring grid positions were empty (compare the positions of programs in Figure 5.25, taken at time slice 5,000, with those in Figure 5.26, taken at 910,000).

The fact that the programs were evolving to collect more energy from the environment is demonstrated by the much higher average energy levels of programs in Figure 5.26 compared to those in Figure 5.25.

As the population tended to be dominated by a single genotype throughout the run,

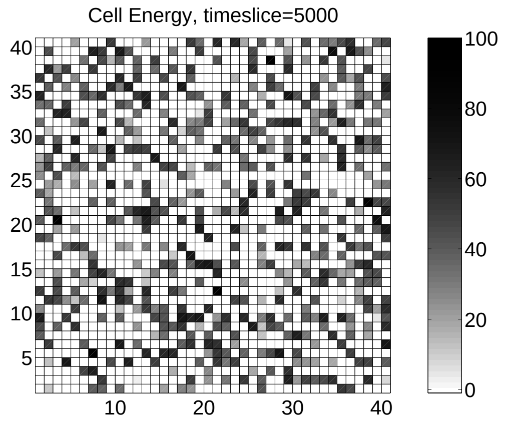


Figure 5.25: Standard Run: Cell Energy Distribution, Time Slice 5,000. (White squares are empty.)

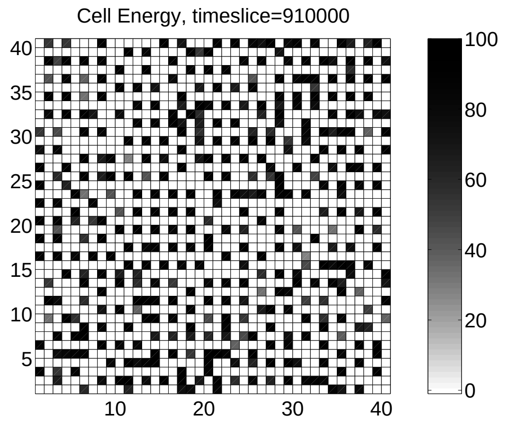


Figure 5.26: Standard Run: Cell Energy Distribution, Time Slice 910,000. (White squares are empty.)

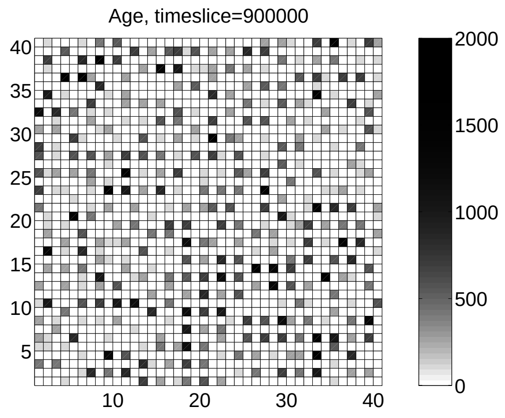


Figure 5.27: Standard Run: Cell Age Distribution, Time Slice 900,000. (White squares are empty.)

the movie of spatial distribution of program length is uninteresting. The distribution of ages of programs across the grid was fairly even, as shown in Figure 5.27.

No multicellular (parallel) programs emerged during the run. Also, because the system parameter `neighbouring_genomes_readable` was set to `no`, neither did any programs evolve which executed foreign code. After the `move` instruction included in the original genotype 348AAAA had been lost in genotype 366AADW, only an occasional variant rediscovered it subsequently, but this was never again taken up by most of the population.

5.2.9 Summary of Results

The summarise the results of this run, the programs are evolving towards increased A-longevity (Figure 5.7), but in doing so their A-fecundity decreases (i.e. their replication period increases: Figure 5.4). The programs do not evolve along the A-fidelity axis (Figures 5.6, 5.11, 5.12). The predominant evolutionary innovation is the accumulation of extra energy collection instructions within the programs' copy loops, which increases a program's chances of survival relative to its competitors (Section 5.2.7). No multicellular (parallel) programs evolved, nor did any parasites or any other sort of program that used code from neighbouring programs (Section 5.2.8). In other words, the evolution observed in this run was steady microevolution, with no spectacular macroevolutionary innovation. Analysis of the activity measures reveals that this run exhibits Class 2 dynamics, according to Bedau *et al.*'s classification scheme, which groups it together with various other artificial life platforms, and distinct from the evolutionary dynamics of the biosphere (Section 5.2.6). Reconstruction of the phylogenetic tree of the significant genotypes reveals that chance events may play a major role in determining the outcome of such a run (Section 5.2.7). We will look into this issue in detail in the next chapter.

Chapter 6

Cosmos Experiments 2: Exploring the Parameter Space

The previous chapter described the results of a single Cosmos run. In the present chapter we will first investigate the extent to which the differences in behaviour between a number of runs are due purely to the stochastic nature of the system (Section 6.1). This gives us an idea of how many times we should run each experiment, keeping all parameter values constant, to see a range of different behaviours arising solely through this stochasticity. With this in mind, the results of a collection of further experiments are then described, which were designed to explore the parameter space of the system somewhat. For all of the experiments reported, any non-default parameter settings are listed; most are specified in the relevant sections of this chapter, but the values of the parameter `rng_seed`, used to seed the random number generator for each of the runs, are listed separately in Section B.5.

6.1 The Role of Chance[1]

Some of the results reported in the previous chapter suggest that chance events may have played a significant role in the outcome of the run (see Section 5.2.7). As discussed in Section 2.3.5, it is generally accepted that contingency ("historical accident") has played a large part in determining the course of evolution on Earth, although there is still much debate as to the relative importance of chance events versus other factors

[1] This section is based upon a previously published paper, [Taylor & Hallam 98].

(e.g. the inherent self-organisational properties of the biochemical world, discussed in Section 2.1.1). With artificial evolutionary systems we have the advantage of being able to "replay" evolution under experimental control. When considering the performance of an evolutionary system, we generally wish to disentangle the relative influence of three factors: (1) contingency, (2) performance due to particular details of the system's design, and (3) performance which may be general to a wide class of evolutionary systems [Taylor & Hallam 97]. In this section, an experiment is reported in which Cosmos was run a number of times, varying just the random number seed between runs. The results will give us a better idea of the role of contingency in the system.

6.1.1 Method

Nineteen runs of Cosmos were initialised, each with exactly the same ancestor programs, and exactly the same parameter values except for the seed for the random number generator (RNG). Most parameters took on their default values; those that did not are listed in Table 6.1.[2] For each completed run, the following measures were investigated:

1. Program age at death
2. Replication period (time between 1st and 2nd faithful replication)
3. Program length
4. Flaw period
5. Number of faithful replications per program
6. Number of unfaithful replications per program
7. Population size
8. Population diversity
9. Cumulative activity
10. Mean cumulative activity
11. Activity wave plots

[2] These experiments were run on slower machines than most of the others, which is why the size of the runs (i.e. the maximum number of cells allowed in the population and the run duration) is smaller than most of the others. The other differences in parameter values came about because these runs were actually conducted before most of the others, and the default values of a few of the parameters were changed in the intervening time. The differences are only minor, and it is not expected that they affect the applicability of the results to the other experiments reported in this chapter.

Parameter	*Value*
inoculation	
`number`	64
startinfo	
`rng_seed`	[*variable*]
termination	
`number_of_timeslices`	300000
environment	
`max_cells_per_process`	800
`number_of_energy_tokens_per_grid_pos_per_sweep`	10
`max_energy_tokens_per_grid_pos`	25
`energy_collection_scheme`	private
`x_delta`	0.025
cell	
`max_energy_tokens_per_cell`	50
`neighbouring_genomes_readable`	yes
mutation	
`mutation_application_period`	1
`default_flaw_period`	1000000

Table 6.1: Non-default Parameter Values for Contingency Experiments.

6.1.2 Results

For each measure, the results from each of the 19 runs were compared.[3] Some of the measures (Population Size, Age at Death, Flaw Period, Number of Faithful Replications and Number of Unfaithful Replications) generally showed no trends, and their absolute values were very similar across different runs. In contrast, trends *were* observed in the five other measures (i.e. Cumulative Activity, Mean Cumulative Activity, Diversity, Program Length and Program Replication Period), with noticeable differences between some of the runs. Plots for some of these measures are presented for two example runs (10 and 17) in Figure 6.1.

Ideally, we would like to know whether the differences in these measures between any of the runs are statistically significant. Such differences would indicate that evolution might genuinely be treading a different path, for no other reason than the different seed used for the random number generator when the runs commenced. The choice of a statistical test for this task was not immediately obvious. We wished to avoid parametric tests, as we did not want to make assumptions about the population parameters

[3] In the following, the pairs of run results displayed in Figure 6.1 and in Figure 6.8 were generally chosen because they illustrate noticeably different results.

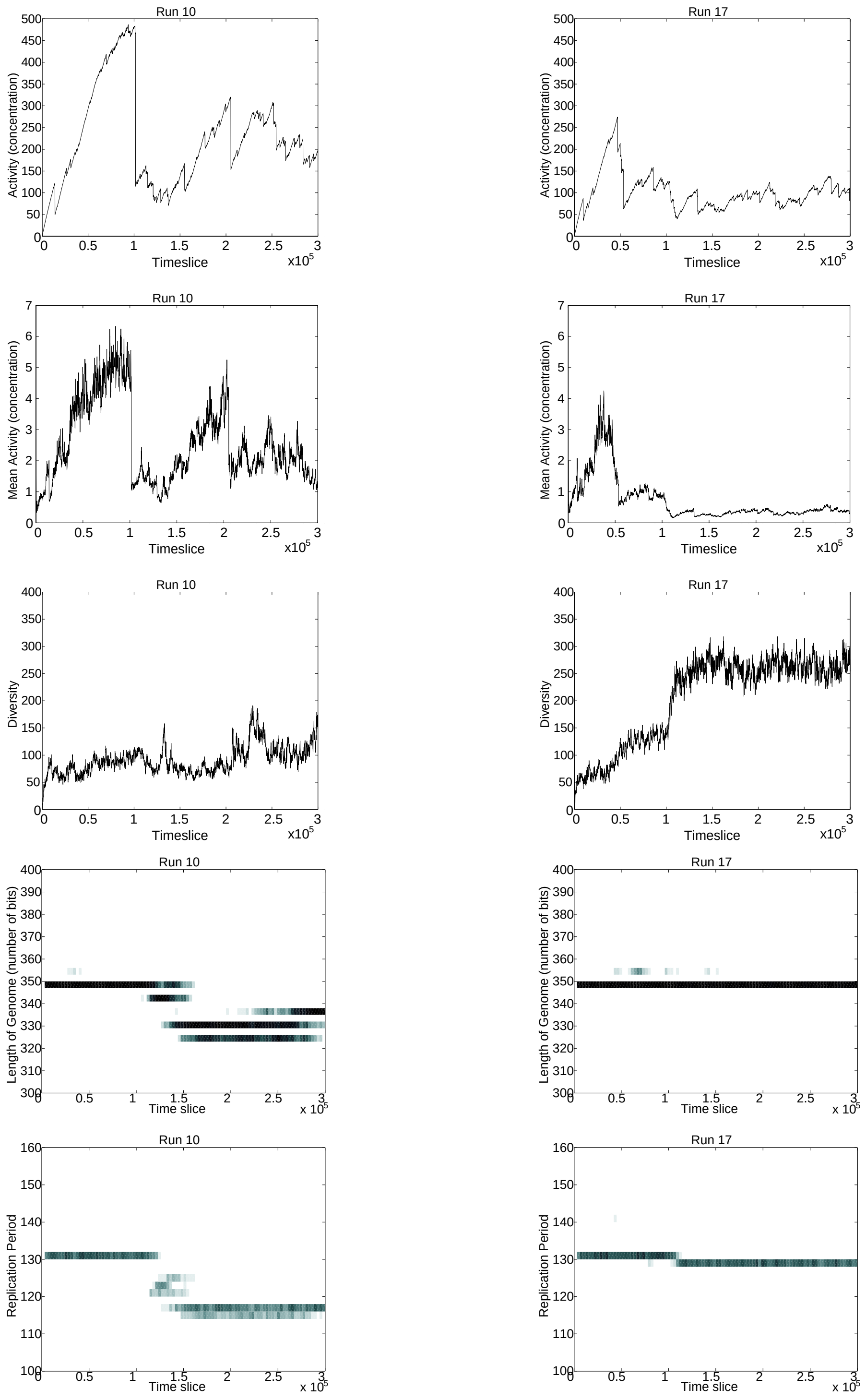


Figure 6.1: Contingency Experiments. Graphs of Cumulative Activity, Mean Cumulative Activity, Diversity, Program Length and Replication Period for Runs 10 (*left*) and 17 (*right*).

(for example, there is no reason to suspect that any of the measures we are looking at are normally distributed across all possible evolutionary runs).

We therefore chose a non-parametric method—a randomisation version of the paired sample t test (described in [Cohen 95]). For each measure of interest, this test will tell us, for each run, which other runs produced significantly different results. The test can indicate whether two samples are related without any reference to population parameters. The procedure used was as follows:

Procedure: Randomisation Version of the Paired-Sample t Test

For each run, 10 sample data points were extracted, each one representing the value of the measure in question at one of 10 equally spaced times throughout the run. The basic idea of the paired sample t test in this situation is to consider the 10 sample points for pairs of runs in turn. By doing pairwise tests at 10 sample points we are comparing the measures at a number of points through the run, with no point having more significance than any other. For each pair of runs, the difference between corresponding samples is calculated, together with the mean value for the 10 differences. We then ask what the likelihood is of achieving this mean difference under the null hypothesis that the two runs are statistically equivalent. The method by which this is done will be explained shortly.

Obtaining Raw Sample Points. In the case of measures which are already statistics of the whole population at any given time (i.e. Cumulative Activity, Mean Cumulative Activity, and Diversity), these 10 sample points could be taken directly from the value of the measure at the appropriate time. However, to prevent high-frequency changes in these measures from producing aberrant results, the measures were smoothed before the samples were taken (using median-smoothing with a window of 10,000 time slices).

In the case of the measures where the existing data consisted of multiple values at each time slice, each representing individual programs (i.e. the Program Length and Replication Period measures), each of the 10 sample points was produced by taking the median value of all values lying within a window of 1000 time slices around the time slice being sampled.

Obtaining Differenced Sample Points. Because of the cumulative nature of evolution, it is possible that a small difference in the sampled value of a measure early on in a pair of runs will be magnified into a large difference later on, even if the two runs are actually proceeding in a fairly similar fashion. In order to gauge the magnitude of this effect, a duplicate set of tests was run, which used the *difference* in value between adjacent sample points as the figure to compare between runs, rather than the *absolute* value of the sample points. Using differenced data should reduce the influence of any cumulative disparity between runs.

Testing for Significance. We are considering the difference in values between corresponding sample points in a pair of runs. Under the null hypothesis that the two runs are equal, however, it is equally likely that these values would be reversed (i.e. for sample point n for runs A and B, the null hypothesis is that the values A_n from run A and B_n from run B are just as likely to have come from the other run—A_n from run B and B_n from run A). If this were the case, the difference between the values would be the same as before, but with the sign reversed. We can test for the significance of the observed mean difference by constructing the distribution of all mean differences obtained from looking at each possible combination of each of the paired samples into one or other of the runs. As there are 10 paired samples, there are 2^{10} (1024) such combinations.

The exact procedure, which is adapted from [Cohen 95], is shown in Figure 6.2. The number that the procedure produces, p, is the (one-tailed) probability of achieving a result greater than or equal to $\bar{x}_D$ (or less than or equal to $\bar{x}_D$ if $\bar{x}_D < 0$) by chance under the null hypothesis. That is, p is the probability of incorrectly rejecting the null hypothesis that systems I and J have equal population mean scores for the measure in question.

For each of the five measures being considered (Cumulative Activity, Mean Cumulative Activity, Diversity, Program Length and Replication Period), this procedure was followed for each of the $19(19-1)/2 = 171$ pairwise comparisons between runs, for both the raw sample data and the differenced sample data.

The p values for each pairwise comparison are shown graphically in Figures 6.3–6.7. These figures show one histogram for p values obtained using raw sample data, and

```
1. For run I and J, if $S_I$ and $S_J$ are lists of the 10 sample data points for each run,
   construct a list D of the differences between these values, $D = S_I - S_J$. Denote
   the mean of these differences $\bar{x}_D$.
2. if $\bar{x}_D = 0$
       $p = 0.5$
   else
    (a) Set a counter C to zero.
    (b) for $i = 0..1023$
          • Construct a list $D^*$ such that $D^*_j = D_j$ if $b_{ij} = 0$, or $D^*_j = -D_j$ if
            $b_{ij} = 1$, for $j = 1..10$, where $b_{ij}$ is the $j^{th}$ digit of $i$ in base 2.
          • denote the mean of the new list $\bar{x}_{D^*}$
          • if $\bar{x}_D > 0$
                if $\bar{x}_{D^*} \geq \bar{x}_D$, then increment C by one
            else if $\bar{x}_D < 0$
                if $\bar{x}_{D^*} \leq \bar{x}_D$, then increment C by one
            endif
    (c) $p = (C/1024)$
```

Figure 6.2: Procedure for Significance Testing in Contingency Experiments.

another for p values obtained using differenced sample data. In all of the histograms, any p value less than 0.05 is plotted as zero. Bars of non-zero height on the histograms therefore represent pairs of runs which are not significantly different from each other for the measure in question at the $p = 0.05$ level.

(Note that, in order to emphasise the formation of various clusters of runs in these histograms, the runs in each histogram are arranged along the x and y axes in increasing order according to the mean of their 10 sample values. While this emphasises clusters in any one histogram, it means that clusters occurring in similar positions in the histograms of different measures do not necessarily represent the same runs.)

The randomisation version of the paired-sampled t test has some advantages over other methods of investigating pairwise comparisons (e.g. it is non-parametric), but it has the disadvantage that it is “virtually certain to produce some spurious pairwise comparisons” [Cohen 95] (p.203). Cohen suggests one way, not to get around this problem, but at least to have some idea of the reliability of a particular set of pairwise comparisons [Cohen 95] (p.204). The idea is to first calculate, at the 0.05 level, how many runs, on average, each run differed from (call this $\bar{n}_{0.05}$). Then calculate a similar figure at a much more stringent level. As we have 1024 numbers in our distribution of mean differences, the 0.001 level is appropriate. Finally, calculate the *criterion differential*, $C.D. = \bar{n}_{0.05} - \bar{n}_{0.001}$. If $C.D.$ is large, this indicates that many significant differences at the 0.05 level did not hold up at the 0.001 level. A small $C.D.$ value indicates that the experiment differentiates runs unequivocally, therefore lending more weight to the validity of the results at the 0.05 level. Table 6.2 shows $\bar{n}_{0.05}$, $\bar{n}_{0.001}$ and $C.D.$ for each measure, and for both raw and differenced sample data.

Table 6.2 reveals a number of interesting results. The most striking is the difference in the results of using raw sample points compared with differenced sample points.

Using raw data, the average number of runs that any particular run was significantly different to at the 0.05 level ranged from 8.42 for Cumulative Activity to 13.26 for Diversity. However, the criterion differential for all of these measures is high (ranging from 6.21 for Mean Cumulative Activity to 12.32 for Program Length). This suggests that the validity of the figures at the 0.05 level are questionable, and the true figures are probably somewhat lower than those calculated. Having said this, the average number of runs that any particular run was significantly different to, even at the 0.001 level, was non-zero for three of the measures (Cumulative Activity, 2.11; Mean Cumulative Activity, 4.11; Diversity, 6.32).

Using differenced data, the results have a very different look. In only two measures were any runs significantly different from any others even at the 0.05 level (0.11 for Cumulative Activity and 0.42 for Diversity), and both of these vanished at the 0.001 level. In other words, these figures suggest that, for *all* of these measures, starting off at any point during any of the runs, the amount the measure *changed* over a given period was not significantly different compared to any of the other runs.

6.1.3 Activity Wave Diagrams

Whereas the Activity and Mean Activity measures produce a summary figure for a whole population of genotypes at time t, activity wave diagrams plot the success of every genotype in the population at every stage of the run [Bedau & Brown 97]. They are therefore a useful visualisation technique for competition between genotypes, and the shape of an individual wave can also suggest the level of adaptive value of the

Measure	Data Type	$\bar{n}_{0.05}$	$\bar{n}_{0.001}$	*C.D.*
Cumulative Activity (concentration)	raw	8.42	2.11	6.32
	differenced	0.11	0.00	0.11
Mean Cumulative Activity (concentration)	raw	10.32	4.11	6.21
	differenced	0.00	0.00	0.00
Diversity	raw	13.26	6.32	6.95
	differenced	0.42	0.00	0.42
Program Length	raw	12.32	0.00	12.32
	differenced	0.00	0.00	0.00
Replication Period	raw	10.21	0.00	10.21
	differenced	0.00	0.00	0.00

Table 6.2: Mean Number of Runs that Each Run is Significantly Different from at the 0.05 Level ($\bar{n}_{0.05}$) and 0.001 Level ($\bar{n}_{0.001}$), and the Criterion Differential (*C.D.*). See text for details.

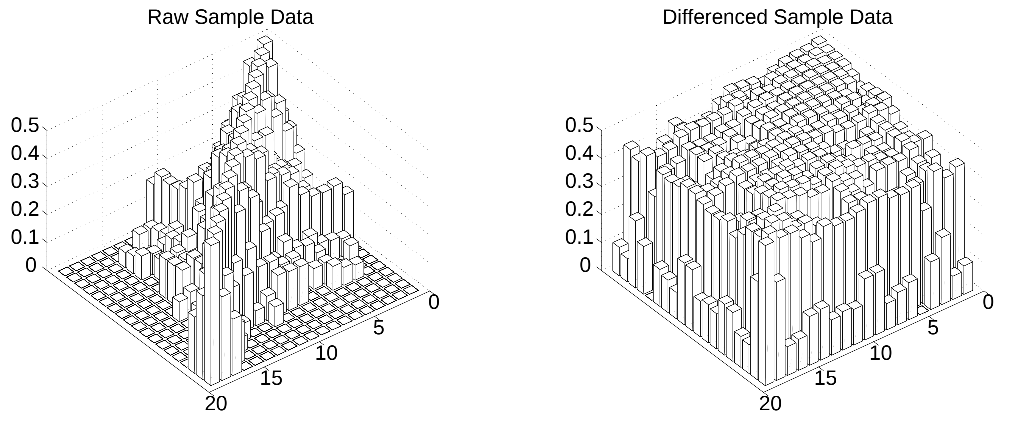


Figure 6.3: **Cumulative Activity (concentration)**: Pairwise comparisons (p values) between runs. Raw Sample Data (*left*). Differenced Sample Data (*right*). p values below 0.05 are plotted as zero, so bars of non-zero height indicate pairs of runs that are not significantly different at the 0.05 level. See text for details.

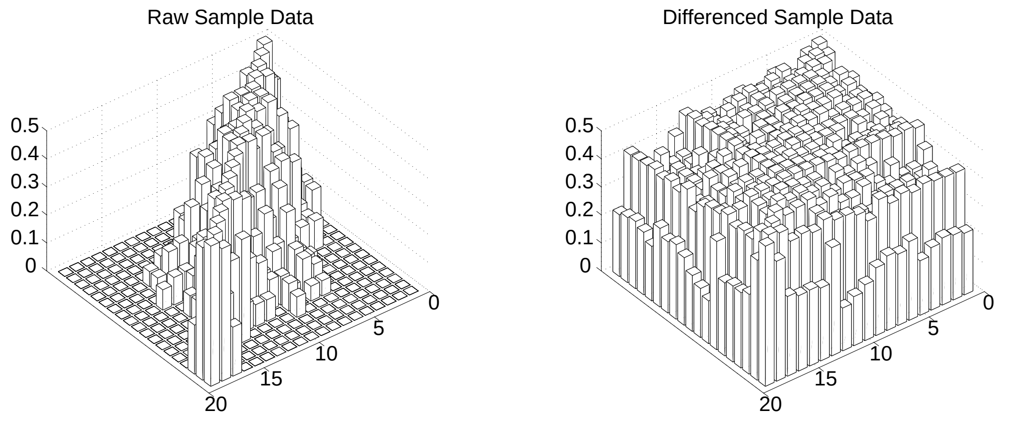


Figure 6.4: **Mean Cumulative Activity (concentration)**: Pairwise comparisons between runs. See text and caption of Figure 6.3 for details.

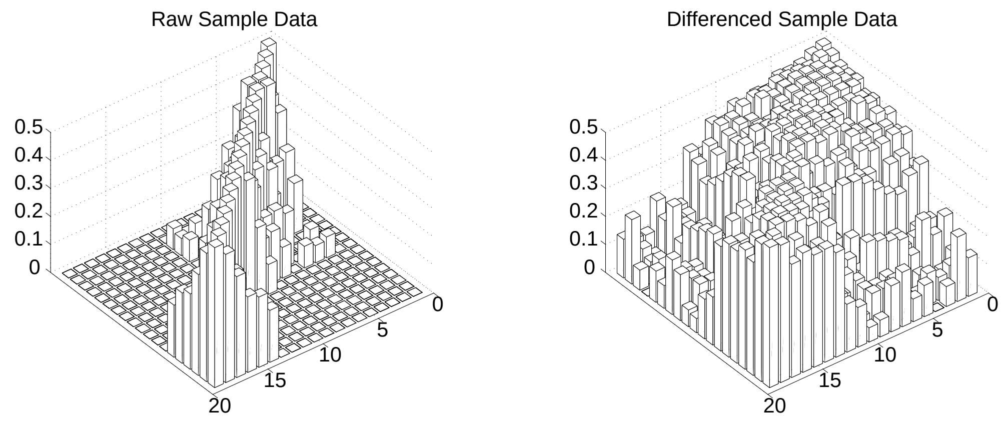


Figure 6.5: **Diversity**: Pairwise comparisons between runs. See text and caption of Figure 6.3 for details.

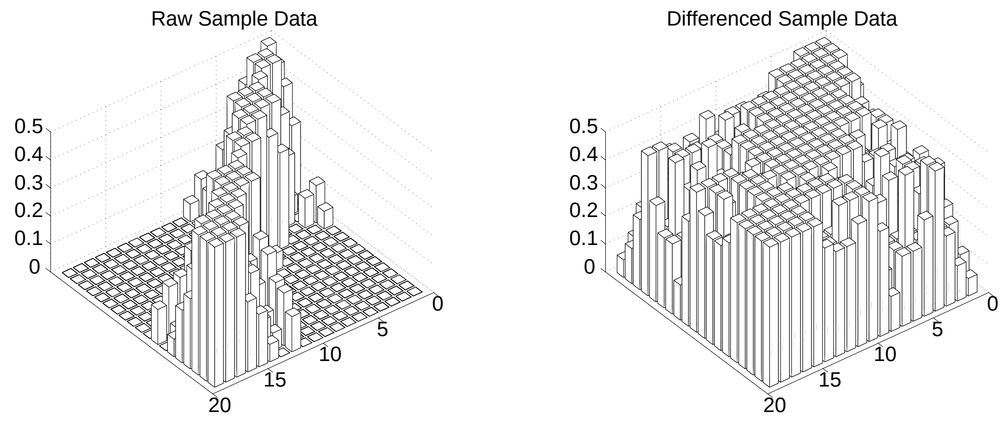


Figure 6.6: **Program Length**: Pairwise comparisons between runs. See text and caption of Figure 6.3 for details.

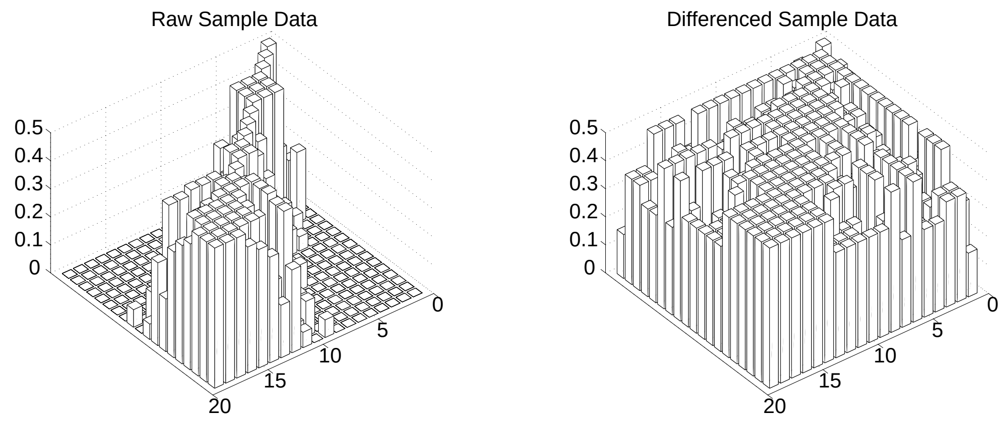


Figure 6.7: **Replication Period**: Pairwise comparisons between runs. See text and caption of Figure 6.3 for details.

corresponding genotype relative to its competitors (see Section 5.1.1).

The activity wave diagrams for most of the runs looked surprisingly different, although it is hard to quantify these differences (the Activity and Mean Activity measures do quantify some aspects of them, but no single measure captures all of the important information that the diagrams can tell us). Example activity wave diagrams (for runs 10 and 17) are presented in Figure 6.8.

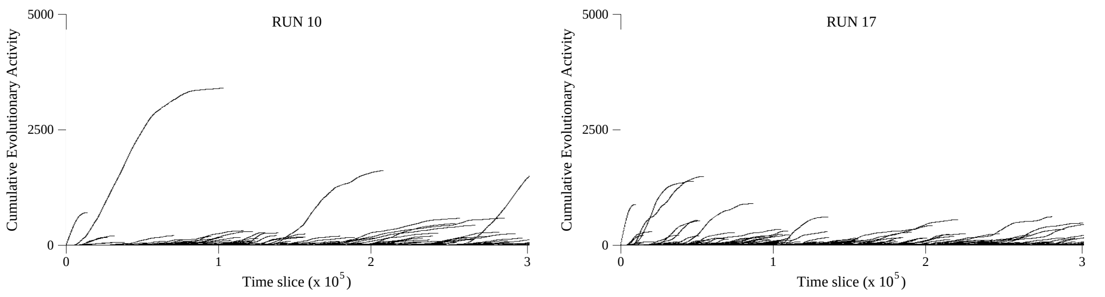


Figure 6.8: Activity Wave Diagram, Runs 10 (*left*) and 17 (*right*).

One way in which the activity wave diagrams can be very useful is in evaluating the effectiveness of different measures of evolution at highlighting the important adaptive events during a run. Cumulative Activity usually seems to give a better reflection of the wave diagram than does Mean Cumulative Activity. This is possibly because the latter measure is defined as Activity divided by Diversity, but diversity, by its very nature, does not take account of the *concentrations* of different genotypes, but merely their presence.

6.1.4 Discussion

Significant differences were found between runs for five of the measures investigated, at least when using the raw data. These measures were: Program Length, Replication Period, Cumulative Activity, Mean Cumulative Activity, and Diversity.

For Program Length and Replication Period, significant differences (at the 0.05 level) were observed in the raw data values between some runs. For these measures, the mean number of runs that each run is significantly different from at this level was calculated as 12.3 for Program Length and 10.2 for Replication Period, but the high criterion differential on these scores suggests that the true value should be somewhat lower (looking at Figures 6.6 and 6.7, probably somewhere in the range of 6 to 10).

Looking at the derived measures suggested by Bedau and colleagues (i.e. Cumulative Activity, Mean Cumulative Activity and Diversity), significant differences were found between runs which did hold up even at the 0.001 level. Again, the true value of each of these differences probably lay in the range of roughly 6 to 10.

These results indicate that each run, on average, performed significantly differently to between a third and a half of the other runs. One of the main reasons for doing these experiments was to understand how we should deal with contingency when conducting further experiments with Cosmos. If we assume that at least the finding that each run is statistically different to more than a third of the others is a general result, then we can use the following rule of thumb: For each re-run of a trial with a different seed for the RNG, the probability of its outcome being statistically equivalent (at the $p = 0.05$ level) to the original one is, at most, about $\frac{2}{3}$. Therefore, the number of *re*-runs that should be conducted to be confident (at the 95% level) of at least seeing one statistically different type of behaviour is n, where $(\frac{2}{3})^n \leq 0.05$, i.e. $n \geq 7.388$, or, in round figures, $n \geq 8$. This is the number of *re*-runs *after* the original, so, finally, we can say that *any trial should be conducted nine times with different seeds for the RNG.*

Having said that each run performed significantly differently to at least a third of the other runs, precisely *which* runs were significantly different depended upon the particular measure being looked at. This emphasises the fact that one should be clear about exactly what measure is being used when talking about comparisons between evolutionary runs.

The fact that *no* significant differences were found between any of the runs for any of the measures when looking at *differenced* sample data suggests that the significant differences observed in *raw* sample data may be caused (at least in part) by the cumulative magnification of initially small differences as a run proceeds. If this effect is controlled for (which was the purpose of using differenced data), the behaviour of the runs in terms of the *change* in values of the measures over a given time period would seem to be very similar in all of the runs. However, because of the cumulative magnification of small differences, the *absolute* outcomes of the runs *do* differ significantly in some cases, so contingency *does* play a big role.

As an aside, we can ask to what extent these results can be generalised to other evolutionary systems. Considering biological evolution first, it is clear that even just in terms of population size and the length of runs, the system is completely trivial. Also, the role of contingency may be different in systems which have rich ecological interactions (of which Cosmos programs have very little). It would therefore be unwise to claim that these results can tell us much about the role of contingency in biological evolution, but they may be relevant in specific cases. As for other artificial evolutionary systems, Cosmos is of comparable design, so the results, and the rule of thumb about the number of trials that should be run, should be broadly applicable to these platforms as well. The extent to which ecological interactions affect the results may be investigated by running similar trials on systems that display stronger interactions of this kind (such as Tierra).

In the remainder of this chapter, we will look at the effects of changing various parameter values on the behaviour of the system. In light of the results reported in this section, each of the following experiments is conducted nine times, using different seeds for the RNG each time.

6.2 Re-running the Standard Model

As the experiments reported in the previous section were done using slightly different parameter settings to those used for the standard run reported in Chapter 5, it was decided to re-run the standard run nine more times in order to see whether the results reported in that chapter were representative for the standard parameter settings. For each of the nine runs, all of the parameter settings were the same as those used in the previous chapter, except for the number used to seed the RNG at the start of the run.

6.2.1 Results

Some of the runs behaved very similarly to the one reported in Chapter 5, but others did not. By studying the results, it turns out that those runs which did behave similarly to the standard run were all examples of Bedau's Class 2 systems (see p.126), and those which behaved differently were all examples of Class 1 systems, as explained below.

Class 2 Systems

The results of four of the nine new runs (Runs 6–9) were very similar to those reported for the standard run. They had bounded Mean Cumulative Activity ($\bar{A}_{cum}$), positive Mean New Activity ($\bar{A}_{new}$), and bounded Diversity (D). At the end of each run, the diversity was very low, with a single dominant genotype and a dozen or fewer mutant genotypes present in much smaller numbers.

The results of Run 7 were slightly different because a mutant of length 372 bits arose which produced an offspring of length 378, which in turn produced an offspring of length 372, and so on in a cycle. However, the general pattern was the same as in Runs 6, 8 and 9.

Class 1 Systems

The results of the other five runs (Runs 1–5) were different. Each of these runs started off in a similar manner to those already described, but a some point during the run, their behaviour changed. The behaviour of the runs after this turning point was reached corresponds to Bedau's Class 1, with bounded D, zero $\bar{A}_{cum}$, and zero $\bar{A}_{new}$. The time at which this change occurred was different in each of the five runs, ranging from time slice 45,000 (approximately) in Run 2 to time slice 770,000 (approximately) in Run 4.

From the measures we looked at, reliable indicators of Class 1 dynamics were a combination of zero Mean Cumulative Activity ($\bar{A}_{cum}$), zero Mean New Activity ($\bar{A}_{new}$), stable population size, high diversity (of the same order as the population size), and high program length. Graphs of these measures for a typical run showing Class 1 behaviour are shown in Figure 6.9, alongside the corresponding graphs for a run showing Class 2 behaviour.

Analysis of these runs shows that in each one, at the time of the transition from Class 2 to Class 1 dynamics, a mutation appeared which caused a program not to stop copying instructions to its offspring after it had copied its final instruction, but rather to jump back and copy a section of itself a second time around. As a result, the offspring is approximately twice as long as its parent, but the first half of it is still a fully functional self-replication program. As long as the mutant retains the `stop` instruction at the end of the functional code, then the extra instructions do not get executed; they are the

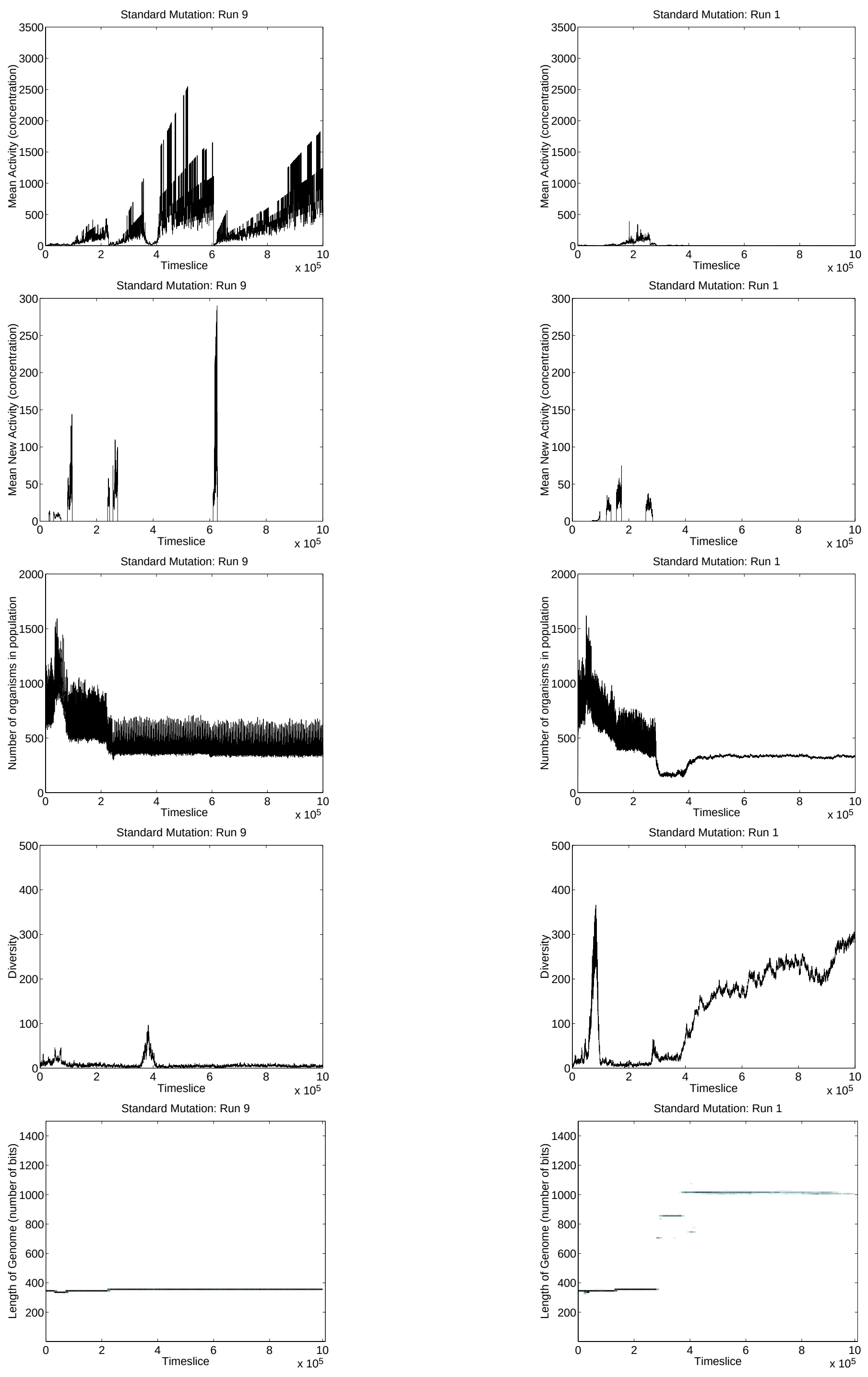


Figure 6.9: Class 2 versus Class 1 Dynamics. *Left*: Mean Cumulative Activity, Mean New Activity, Population Size, Diversity and Program Length for a Class 2 run (Run 9). *Right*: Corresponding graphs for a Class 1 run (Run 1, transition to Class 1 occurs at time slice $\sim 3 \times 10^5$).

equivalent of ‘selfish DNA’. The Class 1 programs had also lost the ability to move around the environment.

Now, in Section 5.2.1 we looked at how the replication period of a program would be expected to vary as the program length increased under two different scenarios. In the first scenario, the length of the program’s ‘copy loop’ increased in proportion to the total program length, and in the second scenario, it remained constant (Figure 5.8). In the present situation, the second scenario is the case. Under these conditions, we found that the replication period of the program would actually remain fairly constant no matter how long the program was. The longer programs in these runs are therefore not at any disadvantage to shorter ones in this respect, which explains why they were able to survive. However, to explain why these longer programs not only survived, but actually flourished and rapidly displaced the shorter programs, we need to look for specific selective *advantages* they might have had over the shorter programs. A likely explanation is that, because they are executing more instructions per time slice (due to their greater length), they collect more energy from the environment (by executing more `et_collect` instructions) during each time slice. In these runs, programs can extract energy from neighbouring programs as they try to collect energy from the environment (the parameter `energy_collection_scheme` is set to `shared`), so a program that extracts a large amount of energy from the environment per time slice might kill off some of its neighbours by using all of their energy. Those programs which have least stored energy, i.e. probably the shorter programs which collect less per time slice, will run a greater risk of being killed in this way, hence the longer programs will tend to out-compete them.

If this explanation is correct, then we would expect not to see this Class 1 behaviour if longer programs were not able to execute more instructions per time slice than shorter programs. We will look into this in Section 6.4.

Evidence to support this explanation comes from a number of different sources. First, if the long programs are extracting energy from their immediately neighbouring programs, then we might expect the system to move towards an equilibrium position where the eight locations neighbouring each viable program are empty (because any programs which happen to find themselves in these positions will quickly be killed by energy extraction). The largest number of programs that could be stably supported in this

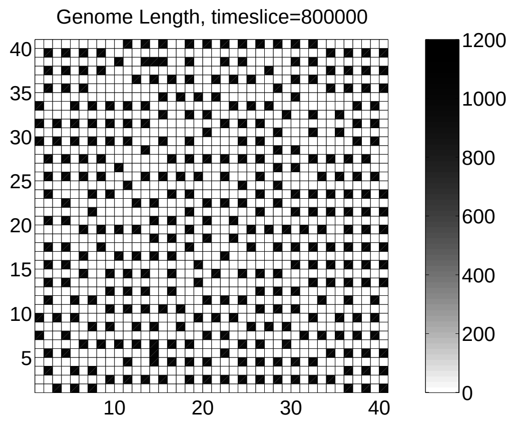


Figure 6.10: Standard Model Re-Runs: Spatial distribution of programs in Run 1 during Class 1 phase (time slice 800,000). Grey level represents program length. (White squares are empty.)

configuration would be, for the 40 x 40 grid used in these runs, 20 x 20 = 400 (i.e. alternate positions would be occupied and empty, in both directions). Looking at the population size graphs for each of the Class 1 runs, the population size does indeed move to around 400 at the time of the transition, and stay at that level thereafter. An example plot, for Run 1, is shown in the right-hand side of Figure 6.9. More direct support for the explanation comes from the recorded 'movies' of spatial distribution. A frame from the movie of program length for Run 1, at a time after the transition to Class 1 dynamics, is shown in Figure 6.10. This figure demonstrates that the spatial distribution is almost exactly as predicted (the observed pattern will never be exactly as predicted, because of the dynamic and stochastic nature of the system).

A consequence of the population size being limited to 400 is that it is no longer affected by the ceiling effect discussed in Section 5.2.3. The population size therefore becomes more stable, and is not subject to the oscillations observed in the Class 2 dynamics (compare the population size graphs for Class 1 versus Class 2 dynamics in Figure 6.9).

Another consequence of the spatial distribution is that the 400 or so programs which are favourably placed will tend to survive for a long time. This is because any new program that moves into a position in between two or more favourably placed programs will probably be killed off before it gets a chance to collect enough energy to fend off such attacks. The chances are weighted against the new program for two reasons: (i) being a new program, it probably has little stored energy to start with, and; (ii) because, even

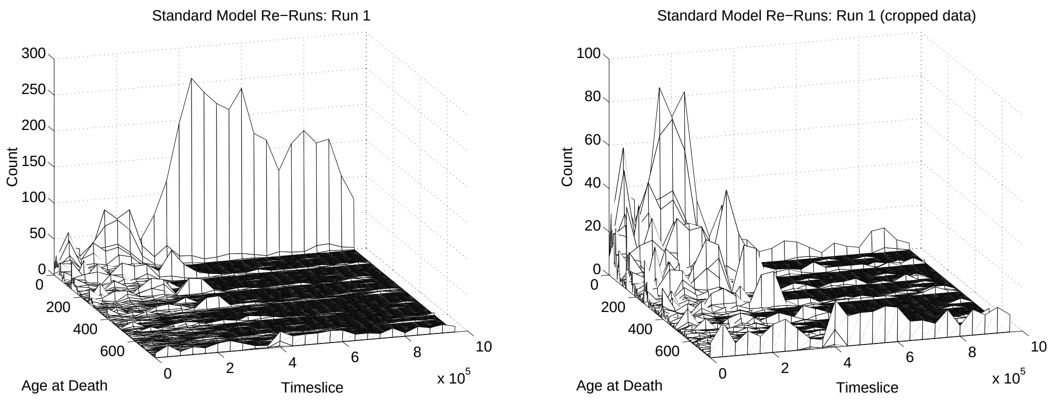


Figure 6.11: Standard Model Re-Runs: Age at Death of programs in Run 1. *Left*: All data. *Right*: Omitting data from programs that died within first 15 time slices after birth (i.e. first row of age axis). In these plots, the vertical axis ('count') shows the number of recorded organisms which died during that particular timeslice window that were of that particular age. The final row on the age axis show the number of programs that lived for *longer than* 750 timeslices.

though at each time slice the order in which programs are given a chance to execute instructions is determined at random, there will probably be two or four favourably placed programs surrounding the new, unfavourably placed one, so it is likely that at least one of these will be executed before the new program is. Evidence for this comes from the graphs of age-at-death. These are shown, for Run 1, in Figure 6.11 (plotted in 3D for clarity). The left-hand side of this figure includes all of the recorded data, and shows that after the transition to Class 1 dynamics at around 400,000 time slices, the number of programs that are dying within their first 15 time slices increases dramatically (these are the programs placed in unfavourable positions). On the right-hand side of the figure, the same data is shown with the first row (corresponding to programs which died in their first 15 time slices) omitted, to emphasise the rest of the data (note the difference in vertical scales). This plot shows that after the transition, programs tended to live for a very short time, or a very long time, with relatively few living for intermediate durations.

Finally, another consequence of the transition is that those programs which are favourably placed will tend to produce more offspring (because they live longer) than did most programs before the transition. This is demonstrated, for Run 1, in Figure 6.12. Note, however, that even though these programs are producing more offspring, most of these will be killed off immediately, as already explained. The fact that few programs will produce offspring which reach the reproductive stage themselves means that lineages

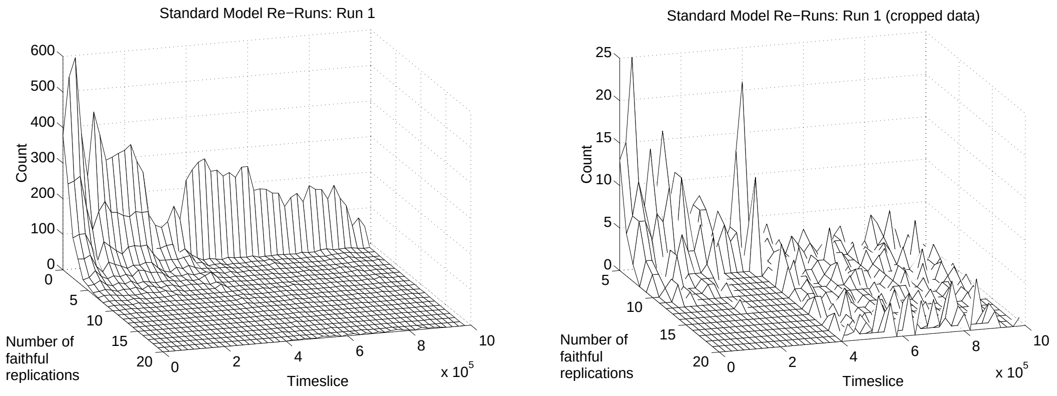


Figure 6.12: Standard Model Re-Runs: Number of faithful offspring of programs in Run 1. *Left*: All data. *Right*: Omitting data from programs that faithfully reproduced fewer than 6 times. In these plots, the vertical axis ('count') shows the number of recorded organisms which died during that particular timeslice window that faithfully reproduced exactly that number of times. The final row on the 'number of faithful replications' axis show the number of programs that faithfully replicated *more than* 20 times.

will not spread through the system, and explains why the diversity of the programs is very high during Class 1 dynamics. (The fact that there was presumably no selection pressure to prevent mutations in the non-functional (selfish) section of the programs must have also contributed to the high diversity.) Class 1 is therefore a degenerative state in which adaptations cannot spread throughout the population, and evolution effectively ceases.

6.3 Mutations and Flaws

In the standard run, the `mutation_period` parameter was set at 1,000,000, and the `default_flaw_period` parameter was set at 250,000. The behaviour of the system was investigated when these parameters were set an order of magnitude higher, and an order of magnitude lower.

6.3.1 High Mutation and Flaw Rates

The non-default parameter settings used for these experiments are shown in Table 6.3.

Parameter	*Value*
`mutation_period`	`100000`
`default_flaw_period`	`25000`

Table 6.3: Non-default Parameter Values for High Mutation Experiments.

Results

With high mutation and flaw rates, eight of the nine runs rapidly fell into the degenerative Class 1 dynamics. The transition occurred within the first 300,000 time slices in all of these cases. One of the runs, number 2, showed slightly different behaviour. The population size settled down to around 400 programs as is typical for Class 1 systems, but the diversity was somewhat lower than expected for Class 1 dynamics, it had non-zero mean cumulative activity and mean new activity, and the length of the programs was much shorter than is typical (around 400 bits, compared to the more usual 1000–1200 bits for Class 1 systems).

Analysis of the programs that lived in Run 2 after the initial transition from Class 2 dynamics revealed that they were almost identical to the ancestral program 348AAAA. The only major differences were that the programs had lost the ability to move (as is the case for normal Class 1 programs), but they also had in the order of *ten* extra `et_collect` instructions in their copy loops, in addition to the two which were present in the ancestor. Now, the addition of these energy collection instructions to the copy loop is the standard method by which the programs improve their ability to survive, as we saw in the analysis of the standard run in Section 5.2.7. However, in that analysis, we only saw programs which had gained an additional *seven* `et_collect` instructions in their copy loops.

It could be that a threshold exists at a certain number of extra `et_collect`s, after which point the programs are extracting too much energy from their neighbours to allow them all to coexist, thereby bringing on the Class 1 dynamics. This scenario would just be a slightly different method of bringing about the same result: in the Class 1 systems we have seen before, the programs execute more instructions per time slice because of their length, and therefore can run through several iterations of the copy loop and consequently execute many `et_collect` instructions per time slice; in the present case, the programs execute fewer instructions per time slice (because they are shorter), but still execute many `et_collect` instructions in a single iteration of the copy loop. To test this explanation, a number of runs were conducted using different ancestor programs for inoculation. Between runs, the ancestors differed in the number of `et_collect` instructions contained in their copy loops. Apart from this, they were the same as the standard ancestor 348AAAA, except that they all also had the `move` instruction removed so that they were sessile. All of the parameters took on their default values, except that mutations and flaws were switched off to enable us to study the neutral (non-adaptive) dynamics of the model. Each run lasted for 25,000 time slices. The population size graphs for four of these experiments are shown in Figure 6.13, for ancestors with 2, 5, 9 and 13 `et_collect` instructions inside the copy loop. These graphs clearly demonstrate that the dynamics do indeed start to change as the number of `et_collect`s rises, as predicted. It therefore seems reasonable to classify Run 2 as a Class 1 run, along with the other eight runs. In other words, all nine runs can therefore be classified as Class 1.

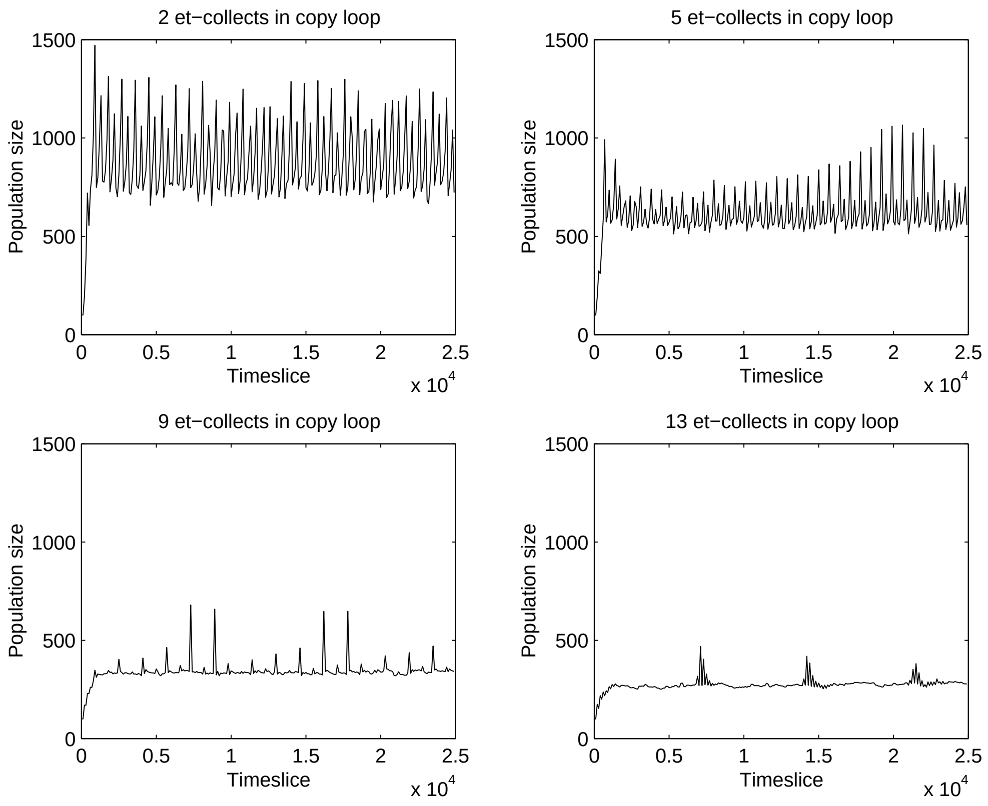


Figure 6.13: Population size graphs for ancestors with different numbers of `et_collect` instructions within copy loop. See text for details.

6.3.2 Low Mutation and Flaw Rates

Experiments were also conducted in which the mutation and flaw rates were much lower than in the standard run. The non-default parameter settings used for these experiments are shown in Table 6.4.

Parameter	*Value*
`mutation_period`	10000000
`default_flaw_period`	2500000

Table 6.4: Non-default Parameter Values for Low Mutation Experiments.

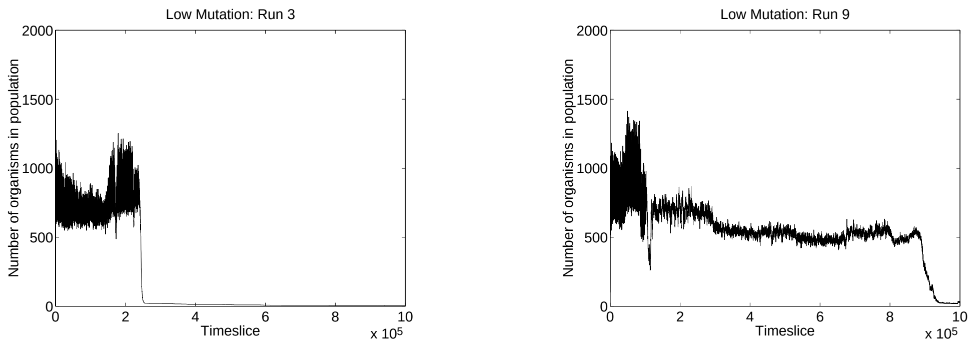


Figure 6.14: Low Mutation: Population size graphs for Run 3 (*left*) and Run 9 (*right*). Both runs experience population extinctions.

Results

Two of the runs (numbers 2 and 8) displayed typical Class 2 dynamics for their whole duration, and four others (numbers 1, 5, 6 and 7) began as Class 2 then switched to Class 1 behaviour at some point during the run. As expected with the lower mutation and flaw rates, the timing of these transitions was generally later than in the runs reported previously, ranging from time slice 200,000 (approximately) in Run 1 to 750,000 (approximately) in Run 6.

The other three runs (numbers 3, 4 and 9) displayed behaviour which had not been previously encountered. Each of these three runs initially displayed Class 2 behaviour as normal. Run 9 subsequently switched to Class 1 behaviour. However, at some later point during each run, the entire population was suddenly wiped out. (In Runs 3 and 9 a handful of programs remained, but these had lost the ability to reproduce and therefore spent the rest of the run collecting energy from the environment with no competition.) The population size graphs for runs 3 and 9 are shown in Figure 6.14.

Analysis of Run 3 revealed that the population was being invaded by a sort of 'cancer' mutant. The effect of this mutant was to add an extra (non-functional) instruction to the end of the program each time it reproduced. The length of the program therefore increased by one instruction (6 bits) each generation. The original mutant (342ABQQ) had actually lost an instruction (`if_not_fl`) from within the copy loop, which meant that it could reproduce quicker than the standard programs. This gave it an immediate benefit and enabled it to quickly spread and displace the existing programs. However, once established in the system, the mutant's offspring gradually grew longer and longer. The additional code at the end of the programs was executed, but had no significant action, and in particular, it did not include any additional `et_collect` instructions. The only effect of this growth was therefore to drain the program's energy store. Once the growth passed a certain size, it could *completely* drain the program's energy, thereby killing it. The takeover of the population by this mutant can be seen for Run 3 in the sequence of plots shown in Figure 6.15. This is a good example of the 'short-sightedness' of evolution, where a variant with a short-term advantage is selected for, even though it eventually leads to the total collapse of the entire population.

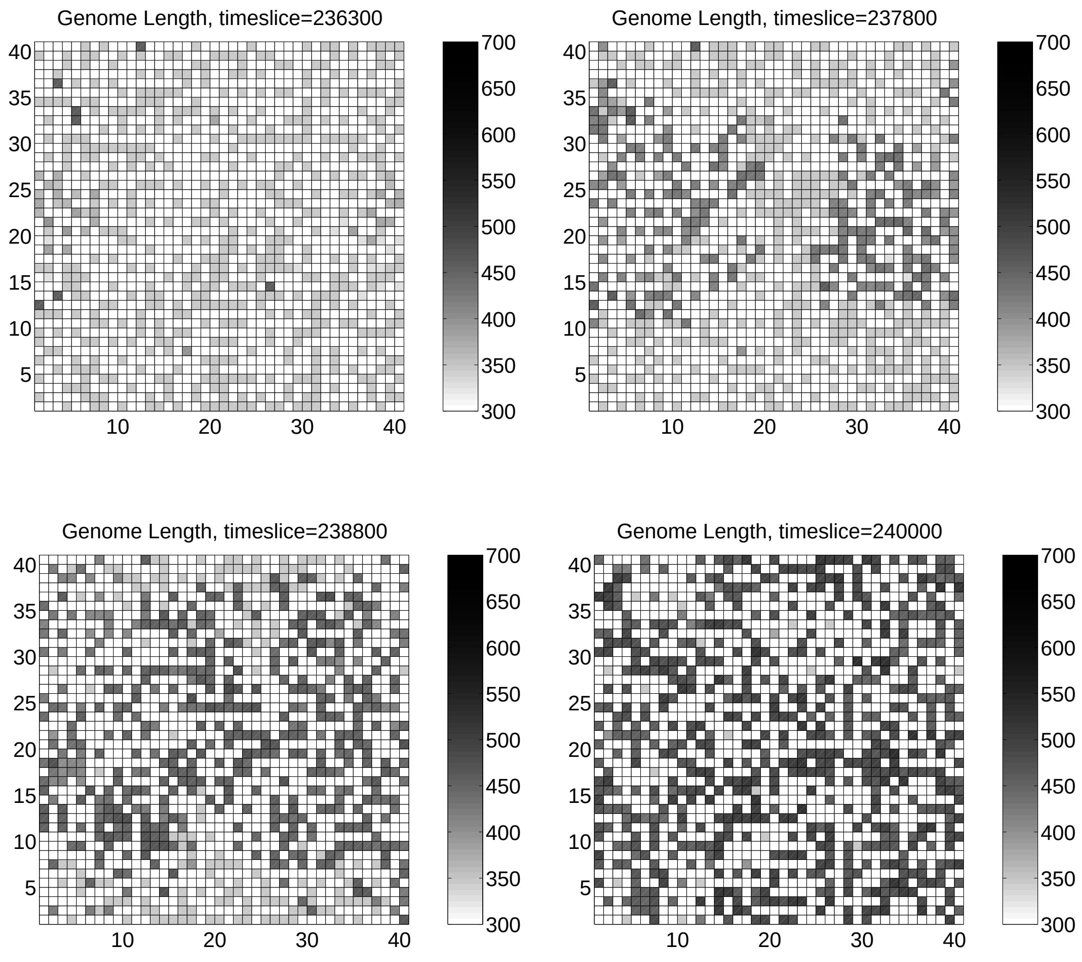


Figure 6.15: Low Mutation: Takeover of population by 'cancerous' mutant (darker squares) in Run 3. The sequence runs from top left to top right, bottom left then bottom right. The cancerous mutants can first be seen clearly as two clusters to the left and right of the central area of the top right figure. (White squares are empty.)

It is interesting that this mutant was only observed in these runs with low mutation and flaw rates. It is possible that such mutants did in fact arise in other runs, but with higher mutation and flaw rates, it is more likely that, once the detrimental effects of the mutant began to manifest themselves, a new variant would arise which corrects the defect. Such a variant would quickly displace the 'cancerous' programs which have low energy reserves, and thereby save the population from extinction.

6.4 The CPU-time Distribution Scheme

In the standard run reported in the previous chapter, CPU-time was distributed to the programs according to their length; longer programs were allowed to execute more instructions per time slice. Recall from Section 4.2.3 that the number of instructions, N, that a program of length L can execute at each time slice (provided it has enough stored energy tokens) is governed by the two parameters `et_value_constant` and `et_value_power` as follows:

$$N = \texttt{et\_value\_constant} * L^{\texttt{et\_value\_power}}$$

In the standard run, the values of these parameters were 0.025 and 1.0 respectively. In that run, the program length actually increased during the course of evolution, as described in Sections 5.2.1 and 5.2.7. To see whether this pattern could be repeated in the face of greater selective pressure for *shorter* programs, the run was repeated using the parameter values shown in Table 6.5. The values chosen were such that every program was allowed to execute 10 instructions per time slice, regardless of its length.

Parameter	*Value*
`et_value_constant`	`10.0`
`et_value_power`	`0.0`

Table 6.5: Non-default Parameter Values for CPU-time Distribution Experiments.

6.4.1 Results

All nine runs showed Class 2 dynamics for their whole duration. This therefore confirms the prediction made in Section 6.2.1 that Class 1 dynamics would not arise if longer programs had no advantage in terms of the number of instructions they were allowed to execute per time slice.

In each of the nine runs, the program length (and the replication period) of the programs still showed a net increase from the beginning to the end of the run, but the increments were fairly gradual. The smallest end-of-run dominant program length seen in any of the runs was 354 (Run 8), the largest was 402 (Run 1), and the average was 375. Plots of program length from four of the runs are shown in Figure 6.16. Analysis of individual programs revealed that the increase in size was due largely to the accumulation of energy collection instructions within the programs' copy loops.

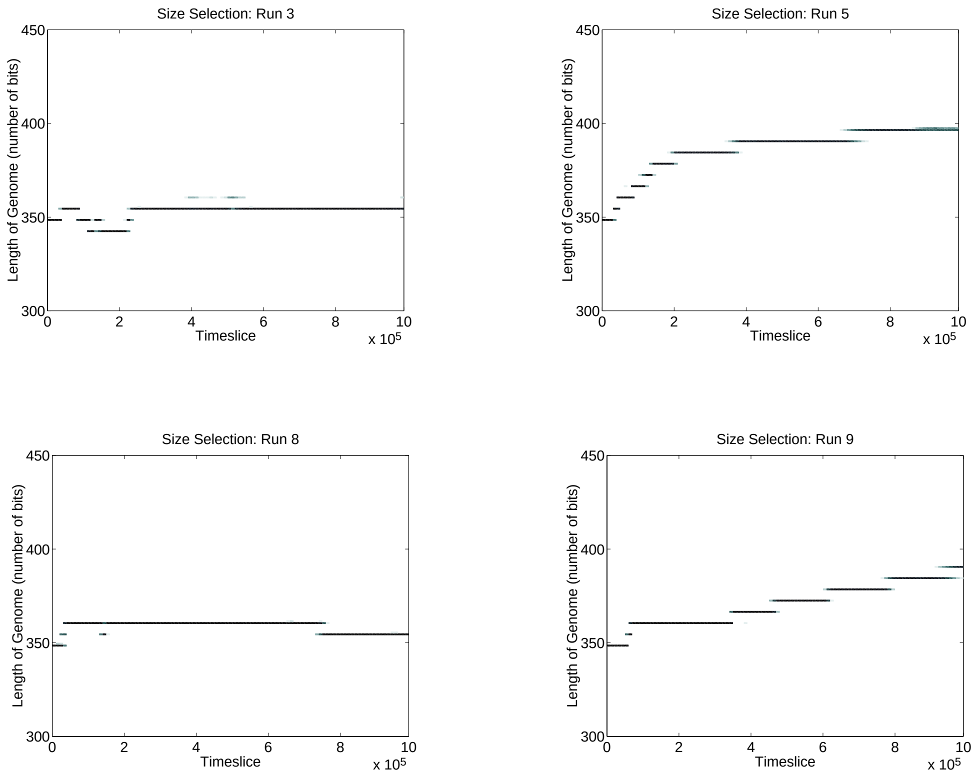


Figure 6.16: CPU-time Distribution: Graphs of Program Length for 4 of the 9 Runs.

6.5 Energy

From the experiments already discussed, it is clear that the distribution of energy in the environment, and its collection and storage by individual programs, are important factors in determining the behaviour of the system. In this section, a number of sets of experiments are reported in which various system parameters concerning energy distribution and collection are changed.

6.5.1 High Energy Levels

In the first set of energy experiments, the amount of energy introduced into the environment at each time slice was increased, as was the maximum amount of energy that any particular grid position could hold. The non-default parameter settings are shown in Table 6.6.

Parameter	*Value*
`number_of_energy_tokens_per_grid_pos_per_sweep`	100
`max_energy_tokens_per_grid_pos`	500

Table 6.6: Non-default Parameter Values for High Energy Experiments.

Results

The only major difference observed with these parameter settings was that the population size during Class 2 dynamics oscillated about the maximum of 2500, defined by the parameter `max_cells_per_process`, rather than in the lower region of 600–1000 individuals as seen in most other experiments. This was expected, as the higher energy levels allowed more programs to coexist. Runs 3, 4, 7 and 8 displayed Class 2 dynamics for their whole durations, whereas the other 5 runs switched to Class 1 dynamics at some point. The typical length of programs in these 5 runs during Class 1 dynamics was longer than observed in other experiments (ranging from approximately 1850 bits in Run 7 to 3800 bits in Run 2), which was again due to the increased energy levels.

6.5.2 Private Energy Collection

The inability of the population to continue evolving effectively in the Class 1 runs that we have encountered up to now seems to be due to the fact that a program's offspring cannot spread through the population. This is because the offspring are usually drained of energy by existing programs before they are able to reproduce, as discussed at the end of Section 6.2.1.

If we were to prevent programs from using energy stored in neighbouring programs, then we would expect that evolution might be able to continue even if the system were to enter a state resembling Class 1 dynamics. A collection of runs was conducted to test this, which used the non-default parameter value shown in Table 6.7.

Parameter	*Value*
`energy_collection_scheme`	`private`

Table 6.7: Non-default Parameter Value for Private Energy Experiments.

Results

Three of the runs (numbers 2, 3 and 7) displayed Class 2 dynamics for their whole duration. However, because programs in overcrowded areas were no longer being killed off by having their energy reserves drained, far more programs were able to coexist within the population at any given time. In fact, the population size oscillated around the limit of 2500 individuals imposed by the parameter `max_cells_per_process`, with a proportion (10%) of the population being culled each time this limit was reached (as determined by the parameter `population_cutback_on_overcrowding`).

Five of the other runs (numbers 1, 4, 5, 6 and 8) showed behaviour which looked similar to Class 1 dynamics. Graphs of some of the measures of interest are shown for Runs 4 and 5 in Figure 6.17.

Looking at these graphs, it is clear that the behaviour of these runs is not exactly the same as the Class 1 runs we have encountered before. Mean Cumulative Activity, $\bar{A}_{cum}$ (not shown), and Mean New Activity, $\bar{A}_{new}$, are both non-zero after the transition, the population size settles down in the region of 1500–2000 individuals, the diversity is lower than seen in previous Class 1 systems (but still higher than in Class 2 systems), the program length initially shows a marked increase, but in some runs subsequently became somewhat shorter again during the rest of the run, and the replication period shows a downward trend.

The fact that $\bar{A}_{cum}$ and $\bar{A}_{new}$ are non-zero suggests that particular individual genotypes *were* spreading throughout the population even after the transition, as expected (because their offspring were not being killed off by energy drainage before they had a chance to reproduce). This is also demonstrated by the appearance of significant waves in the activity wave diagrams (not shown) for these runs after the time of the transition. In other words, effective evolution is still possible, even after the transition, although it proceeds at a slower pace (the activity waves are smaller, and $\bar{A}_{cum}$ and $\bar{A}_{new}$ are lower after the transition). The slower pace is due to the increased competition for energy (because the longer programs are collecting more energy from the environment at each

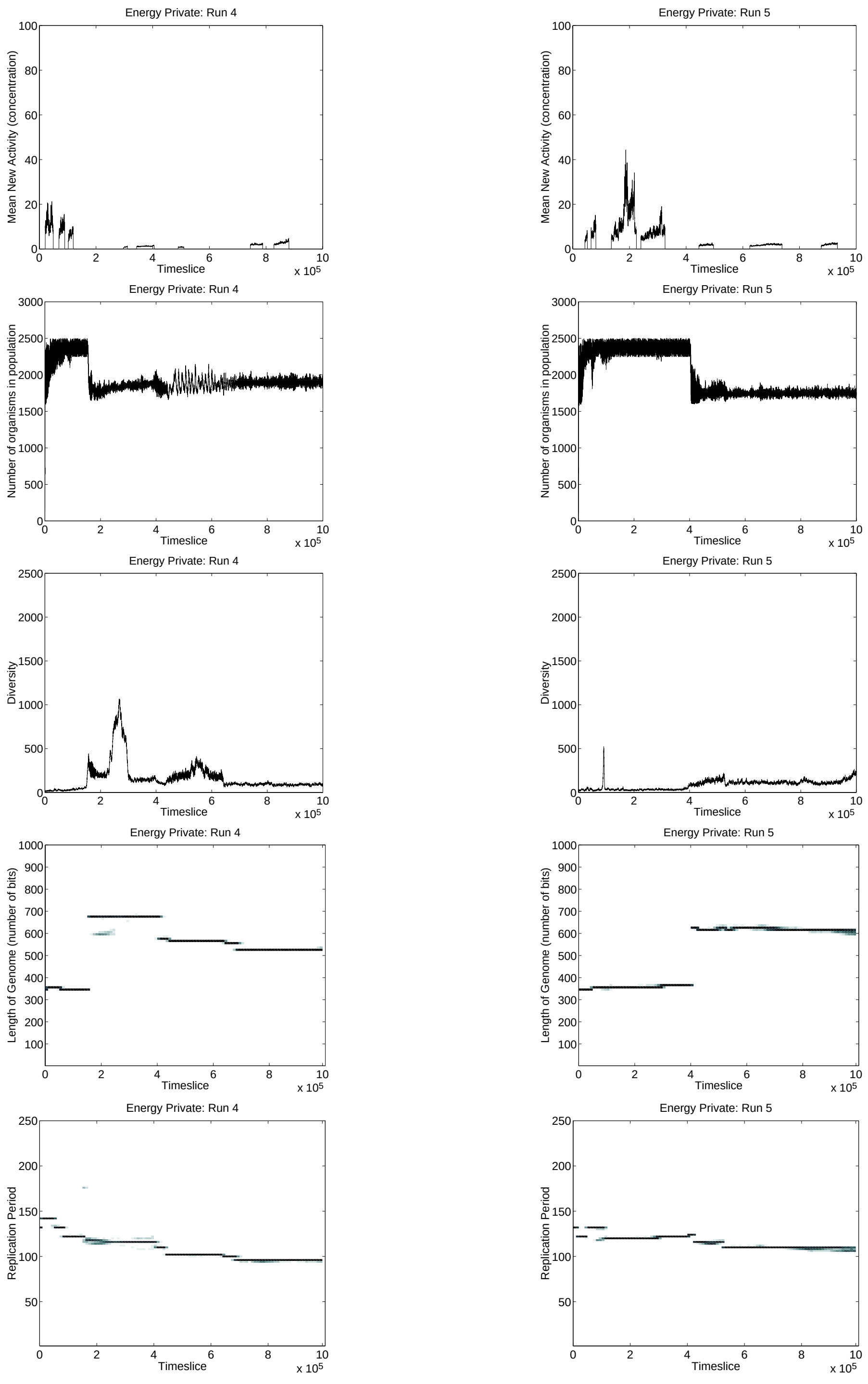


Figure 6.17: Private Energy Collection: Class 1-like behaviour. *Left*: Run 4, transition occurs at time slice $\sim 1.8\times10^5$. *Right*: Run 5, transition occurs at time slice $\sim 4.0\times10^5$.

time slice), which generally leads to shorter lifetimes for the programs. The somewhat smaller population size may also have contributed to the slower pace (but note that this itself is related to the greater competition for energy).

The fact that well-adapted genotypes can still spread throughout the population also explains why the diversity is lower than was observed in other Class 1 systems.

The decrease in program length in the period after the initial increase which caused the transition is explained by the fact that the system is still able to evolve. Analysis of individual genotypes in Run 4 showed that the shorter programs which emerged towards the end of the run had lost redundant instructions from within their copy loops, and were therefore able to reproduce faster (i.e. the replication period decreased). This process can be seen as a partial recovery of the system from the initial transition to Class 1 dynamics, bringing it back towards Class 2 dynamics.

The final run, number 9, showed results unlike any of the others. The most obvious differences were evident in the graphs of population size and diversity, shown in Figure 6.18.

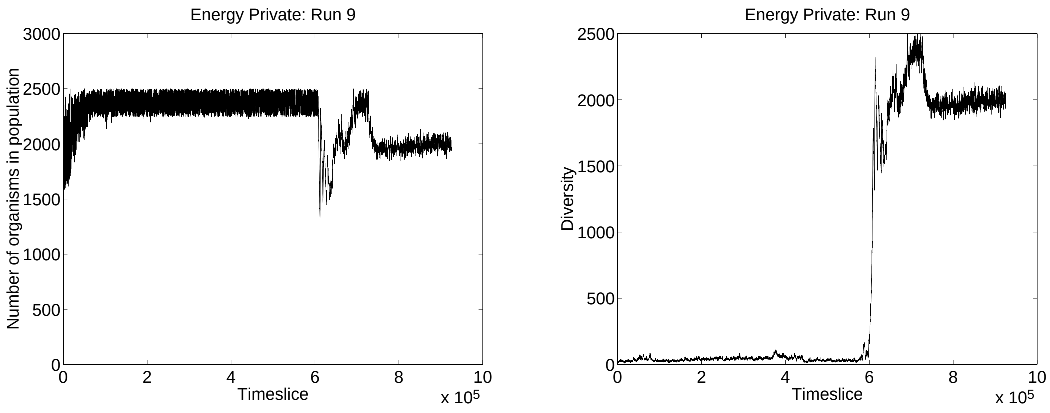


Figure 6.18: Private Energy Collection: Population Size *left* and Diversity *right* for Run 9.

Investigation of individual genotypes in this run revealed that the population had succumbed to a type a 'cancerous' mutation, as encountered in a previous experiment (see Section 6.3.2). As before, the effect of this mutation was that a program's offspring had an extra, redundant instruction tagged onto its end, so that the length of programs in the population grew by one instruction each generation. When we encountered this before, the effect was that the entire population eventually died out, because the programs were being drained of energy by executing all of the redundant instructions.

However, in the present case, it happened that the extra instruction that was being added to the end of the programs at each generation was an energy collection instruction (`et_collect`). As a consequence, rather than dying of energy shortage, the programs were collecting more and more energy as the 'cancer' grew. The programs therefore continued to live, although, as each one was so energy-hungry, the environment could only sustain a population size at the level of about one program per grid position (i.e. approximately 1600 individuals). Also, as each offspring was different from its parent (because it had an extra instruction at the end), the diversity of the population was very high, as shown in Figure 6.18. This mutation could therefore be described as a 'non-fatal cancer'.

6.5.3 Energy Gradient

In all of the experiments reported previously, energy has been supplied to the environment uniformly across the whole grid. In this section and the next, experiments are reported in which two alternative energy distribution schemes were used. In this section, the amount of energy given to each grid position varied in one direction along a linear gradient. This was achieved by using a non-zero value for the parameter `x_delta`, as described in Section 4.5.2. Two different gradients were investigated, as described in the following two subsections.

Small Gradient

The value used in these experiments is shown in Table 6.8. The effect of this value was that grid positions on the extreme left of the grid received 26 energy tokens at each pass through the main Cosmos control loop, those on the extreme right received 34, and those in between received intermediate numbers according to their position. Positions on the extreme right of the grid therefore received about 30% more energy than those on the extreme left. The mean number of energy tokens per grid position was 30, as in the standard run. It was hoped that a non-uniform environment such as this might lead to speciation, with different species specialising in living in areas with different availabilities of energy.

Parameter	*Value*
`x_delta`	`0.2`

Table 6.8: Non-default Parameter Value for Small Energy Gradient Experiments.

Results. No speciation was observed in these experiments. Three of the runs (numbers 3, 5 and 9) exhibited Class 2 dynamics for their whole duration, and five of the others (numbers 2, 4, 6, 7 and 8) switched to Class 1 dynamics during the run.

The remaining run (Run 1) probably fits into Class 1 as well, although the results were slightly unusual. Systems showing Class 1 dynamics typically contain properly functioning self-replicating programs, but as these are unable to spread though the population, numbers of each type of program are low, and the population diversity is high. However, in Run 1 of this experiment, *all* programs capable of faithful replication were lost after only 20,000 time slices. The population survived, but was composed of programs which either reproduced unfaithfully, or not at all. The evolutionary potential of this run had therefore been destroyed.

Analysis of the three runs which remained in Class 2 dynamics for their whole duration revealed that the ancestor program 348AAAA was able to survive even in the low-energy region, and that evolution proceeded in the same way as usual, with a single dominant genotype at any given time, towards greater numbers of energy collection instructions within the copy loop.

Large Gradient

In the light of the results obtained with a small energy gradient, the experiments were re-run using a larger gradient. The chosen `x_delta` value is shown in Table 6.9. Under these conditions, positions on the leftmost column of the grid received one energy token at each pass through the control loop, and those in the rightmost column received 59, with a linear gradient in between. The mean number of energy tokens per grid position across the whole grid was again 30.

Parameter	*Value*
`x_delta`	`1.5`

Table 6.9: Non-default Parameter Value for Large Energy Gradient Experiments.

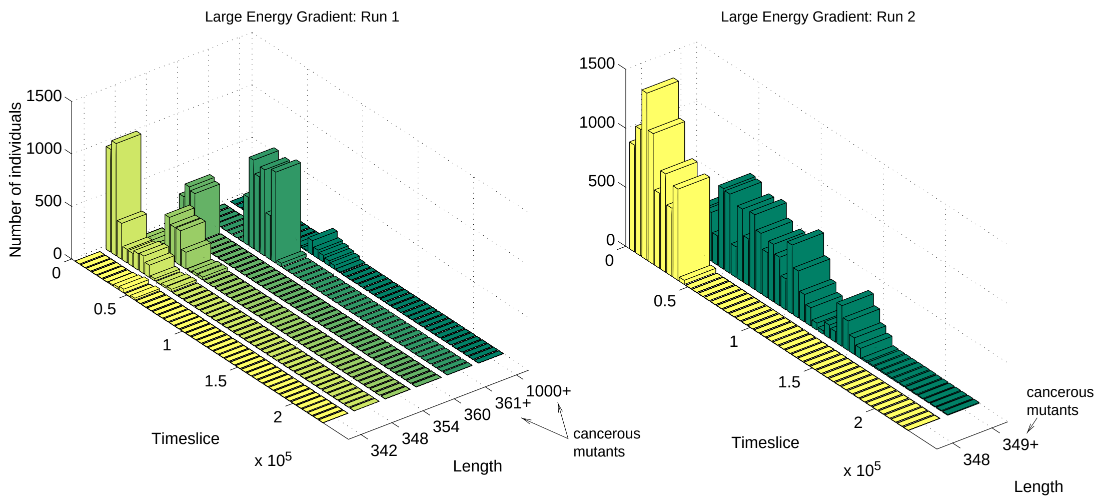


Figure 6.19: Large Energy Gradient, Runs 1 and 2: Change in constitution of population over time. In each graph, the number of individuals shown for each particular length is the sum of all programs in the population at that time which are instances of any genotype of that length. The bin labelled 1000+ in the left graph, and the bin labelled 349+ in the right graph, show data for all programs of the specified length *or longer*. The bin labelled 361+ in the left graph shows data for all programs with length in the range 361–999 bits.

Results. With the larger gradient, the population in two of the runs (numbers 1 and 2) was driven to extinction within the first 200,000 time slices. Of the other seven runs, graphs of the standard measures revealed that four runs displayed fairly standard Class 2 dynamics for their whole duration (Runs 3, 5, 7 and 8), while the other three experienced a transition to a state similar to Class 1 dynamics (Runs 4, 6 and 9).

A closer analysis of the two runs (numbers 1 and 2) where the population suffered extinction revealed that both had succumbed to a cancerous mutation early on in the run. The rise and fall of genotypes of different lengths in these runs in shown in Figure 6.19. In both runs, virtually all non-cancerous programs had disappeared within the first 50,000 time slices, and the whole population had become extinct by time slice 100,000 in Run 1, and time slice 200,000 in Run 2.

Investigation of the Class 2 runs revealed some interesting results. In two of them (Runs 3 and 8), the population was dominated by programs of a single length at the end of the run. However, in the other two (Runs 5 and 7), there were significant numbers of programs of more than one size: the end-of-run population in Run 5 contained programs of lengths 372, 378 and 384 bits; while the end-of-run population in Run 7 contained programs of lengths 366 and 372 bits. The change in the constitution of the populations of all four of these runs is shown graphically in Figure 6.20.

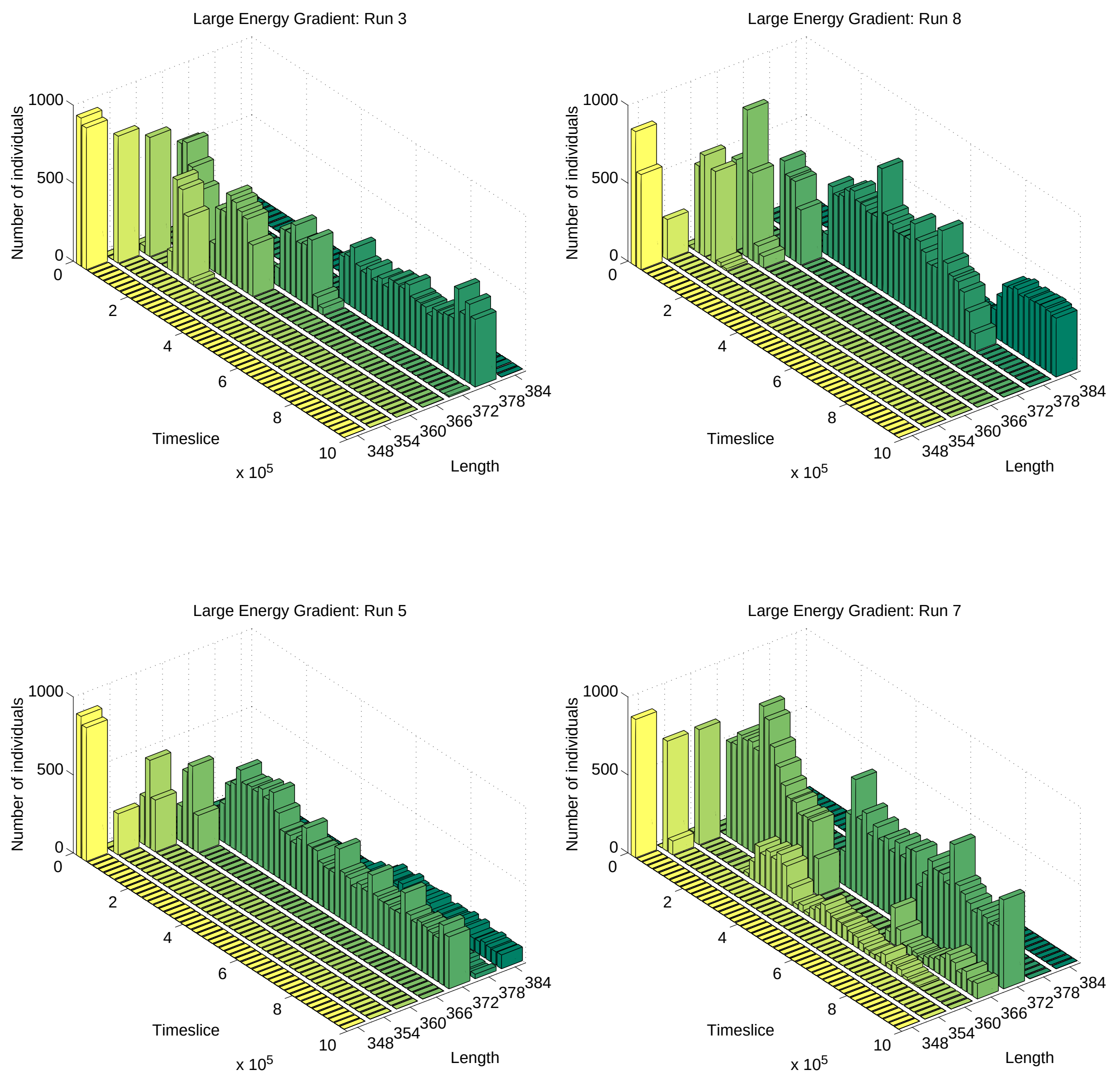


Figure 6.20: Large Energy Gradient, Runs 3, 5, 7 and 8: Change in constitution of population over time. In each graph, the number of individuals shown for each particular length is the sum of all programs in the population at that time which are instances of any genotype of that length.

These graphs further reveal that the population in Runs 3 and 8 tended to be composed of programs of a single dominant length at any point during the run. In contrast, the population in Runs 5 and 7 comprised programs of two or more different lengths for most of the duration of the run. This is the first experiment in which programs of different lengths have been observed to coexist for prolonged periods, at least while the system is in a Class 2 state.

In order to see whether these programs of different lengths were inhabiting different areas of the environment, graphs of spatial distributions were plotted for these four Class 2 runs.

For Runs 3 and 8, which contained programs of a single length only, the distributions are shown in Figures 6.21 and 6.22. This figures show that programs are occupying positions in the rightmost three-quarters of the grid. The reason why the leftmost quarter appears to be inhospitable can be understood by looking at the 'equilibrium' environmental energy distribution, plotted in Figure 6.23. This graph is plotted from data extracted from an output file, and shows how much energy is normally available in grid positions along the x-axis (i.e. along the energy gradient). For each position, this is the amount of energy which it will have in store if it has not been drained by programs in the recent past. The graph shows that positions in the leftmost quarter of the grid typically have fewer than 10 energy tokens available at any given time. Now, the ancestor program was designed on the assumption that each `et_collect` instruction would collect 10 energy tokens; if it regularly collects fewer than 10, it will eventually starve. The leftmost quarter of the grid in these experiments therefore contains insufficient energy for the ancestor programs to survive. The fact that no programs occupied these positions even towards the end of Runs 3 and 8 (and, as we will see, this is also true for all of the other runs) suggests that no programs evolved the ability to collect more energy from the environment and thus be able to survive in this region.

The shape of the graph in Figure 6.23 needs a little more explanation. The energy gradient on the left half of the grid is exactly as expected, and reflects the amount of energy distributed to those positions at each pass through the Cosmos control loop. The levels on the right half of the grid are somewhat unexpected. In these experiments, the parameter `max_energy_tokens_per_grid_pos` was set to its default value of

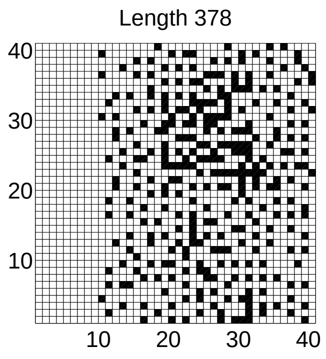


Figure 6.21: Large Energy Gradient, Run 3: Spatial Distribution of Programs at Time Slice 900,000. At this time, virtually all programs in the population were of length 378 bits.

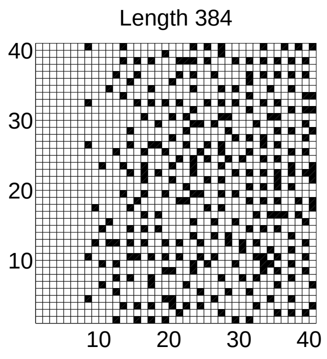


Figure 6.22: Large Energy Gradient, Run 8: Spatial Distribution of Programs at Time Slice 900,000. At this time, virtually all programs in the population were of length 384 bits.

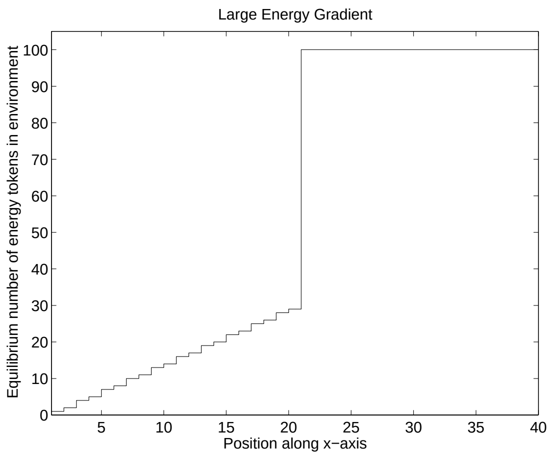


Figure 6.23: Large Energy Gradient: Equilibrium distribution of environmental energy.

100, and all positions on the right half of the grid store this maximum amount of energy (in the absence of programs to drain it). This turns out to be due to a feature of the `AttenuateEnvironmentalEnergy` routine, which is executed at the end of each pass through the main control loop (see Section 4.7). After all programs have had a chance to execute some instructions, this routine removes a certain amount of energy from each grid position in order to prevent it from building up excessively. The current implementation of this routine works by subtracting an amount determined by the parameter `number_of_energy_tokens_per_grid_pos_per_sweep` from each position. (The default value of this parameter, as used in the current experiments, is 30.) When energy is distributed evenly across the environment, this means that each position loses the same amount at the end of each iteration of the control loop as it is given at the beginning of the loop. However, in the current case, positions in the right half of the grid are given more than 30 energy tokens per iteration. They are therefore able to accumulate these tokens over time, and soon reach the maximum level allowed. This feature of the energy distribution turns out to have important consequences for the results of the other runs, as will now be explained.

The spatial distributions of programs in the two Class 2 runs which showed some differentiation in program lengths (Runs 5 and 7) are shown in Figures 6.24 and 6.25. These figures clearly show that programs of different lengths are generally located in different regions of the environment. The segregation is not straightforwardly related to program length itself, as in Run 5 the smaller programs are further to the right of the grid, whereas in Run 7 the opposite is true. Similarly, a comparison of the individual genotypes reveals that the segregation is also not straightforwardly related to the number of `et_collect` instructions that a program contains. In fact, comparison of the genotypes reveals that the only consistent difference between programs living in different areas is that those dwelling exclusively in the right half of the grid (i.e. programs of length 372 in both runs) retain the ability to *move*, where those living further towards the left of the grid are sessile.

Considering the fact that a quarter of the grid does not contain sufficient energy for programs to survive for long, being able to move (and potentially wander into this area) might not seem like a particularly advantageous capacity for the programs to retain. However, analysis of the programs' movements reveals that they have a very strong preference for moving upwards (Direction 4 in Figure 4.3(a)). In comparison, the number of programs that move in any other direction is negligible. Remembering that the environment wraps around in both directions in these experiments (i.e. it is really toroidal in shape), these programs can therefore move around in this manner, while always remaining in the right half of the grid. Figure 6.20 shows that the number of programs of length 372 is fairly high towards the end of Runs 5 and 7, so the right half of the grid is fairly crowded. (This is not so clear in Figures 6.24 and 6.25, because these do not distinguish grid positions occupied by more than one program from those occupied by a single program.) In this situation, it is conceivable that a program might do better on average by moving around, rather than risk being stuck at a location where it is competing for energy with one or more other programs.

In contrast, programs that live in the left half of the grid in these runs were sessile.

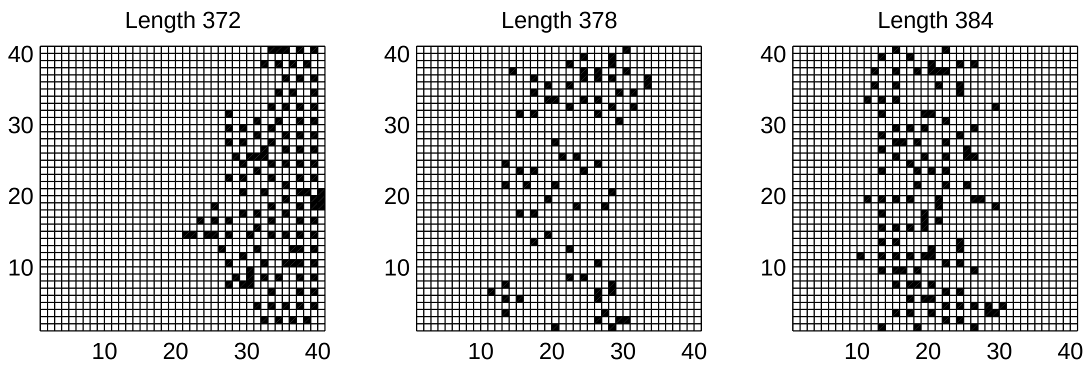


Figure 6.24: Large Energy Gradient, Run 5: Spatial Distribution of Programs of Different Sizes at Time Slice 900,000.

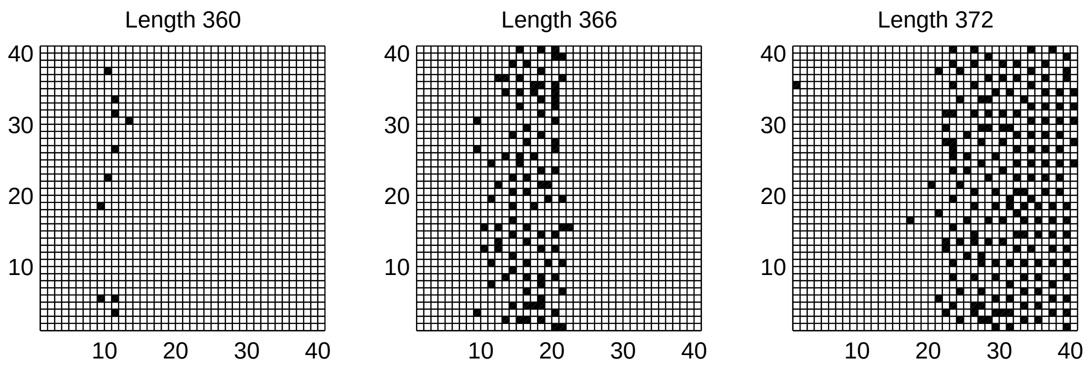


Figure 6.25: Large Energy Gradient, Run 7: Spatial Distribution of Programs of Different Sizes at Time Slice 900,000.

Grid positions in this half typically have much less available energy than those in the other half (Figure 6.23). The number of programs in this area is also much lower than in the other half (compare the number of programs of different lengths towards the end of Runs 5 and 7 in Figure 6.20 with the position of these programs in Figures 6.24 and 6.25). Under such conditions, these programs have adopted the strategy of staking a claim on a particular grid position. The concentration of programs in this area is low enough that they generally do not have directly neighbouring programs with which to compete for energy.

As a final point about these programs, notice that in Run 5, some of the sessile programs of lengths 378 and 384 are located in the right half of the grid, in competition with the motile programs of length 372. This suggests that the adaptive significance of movement in comparison to non-movement is not particularly great even in this half of the grid. The segregation of the different populations of programs has presumably been sustained by the existence of the energy gradient, but the initial appearance of the differences may have been due to rather contingent factors ('historical accidents'). This is supported by analysis of the programs in Runs 3 and 8, where no differentiation was seen. In Run 3, all of the programs were motile, and in Run 8 they were all sessile, despite the fact that in both runs the populations covered the full three-quarters of the environment (Figures 6.21 and 6.22).

Finally, the three runs which display Class 1-like dynamics (numbers 4, 6 and 9) were analysed. All three of these showed length differentiation by the end of the run, as shown in Figure 6.26. As is typical for Class 1 runs, the lengths of programs were much longer than in Class 2 runs. The difference in length between the longest and shortest species in these runs was also much larger than in the differentiated Class 2 runs we have seen already (Runs 5 and 7). Note that the overall population sizes towards the end of these runs are also considerably smaller than in the Class 2 runs, especially in Runs 4 and 9 (Figure 6.26).

The spatial distribution of programs of the dominant lengths at time slice 900,000 is shown for each of the three runs in Figures 6.27–6.29. Again, programs of different length are clearly generally located in different areas of the environment, for all three runs. However, analysis of the individual genotypes revealed that this time the segregation was not related to the ability or inability of programs to move; in Run 6, *all*

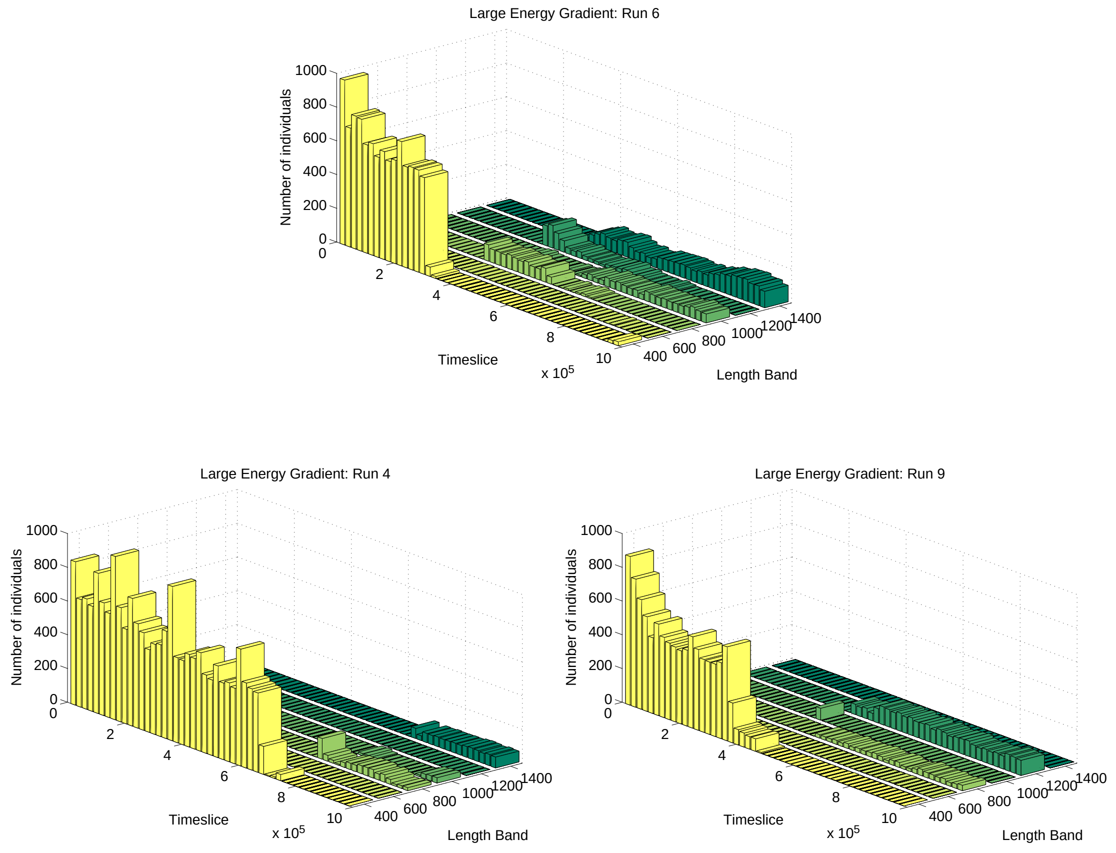


Figure 6.26: Large Energy Gradient, Runs 4, 6 and 9: Change in constitution of population over time. Each bin of the Length Band axis shows data for all programs in the population with length in a range of 200 bits, centred on the value marked. For example, the first bin, marked 400, shows the collected data of all programs of length 300–499 bits.

programs were sessile, and in Runs 4 and 9, *all* programs were motile, with a strong preference for moving upwards (Direction 4), or occasionally downwards (Direction 0). It appears that in these runs, the segregation was mainly due to the lengths of the programs. Grid positions in the area immediately to the left of the centre of the environment receive approximately 30 energy tokens at each pass through the control loop (Figure 6.23). This is the same amount that all positions normally receive under the standard parameter settings, and under those conditions (i.e. in most of the other experiments reported in this chapter) we have seen that the lengths of Class 1 programs are generally in the range 600–1200 bits. In the present runs, we see that programs in this central region also have lengths in that range (Figures 6.27–6.29). In contrast, programs to the right of the environment are of greater length (typically 1250–1500 bits). Remembering that longer programs are allowed to execute more instructions per time slice, and therefore are potentially able to collect more energy per time slice, the location of these programs can be explained by the larger amounts of energy typically available in the right half of the environment.

6.5.4 Random Distribution

Another way in which the distribution of environmental energy can be made non-uniform in Cosmos is to have it distributed randomly. Under this scheme, chunks of energy tokens are distributed to grid positions at random, until the total amount has been allocated.[4] If the mean number of energy tokens distributed to each grid position was the same as in the standard run (i.e. 30 per time slice), then the populations soon died out. The mean number was therefore increased by threefold to 90 energy tokens per time slice, and the maximum number of energy tokens that any grid position could store was also increased somewhat. Under these conditions, the populations could often survive for reasonable durations. The non-default parameter settings for these experiments are shown in Table 6.10.

[4] Recall from Chapter 4 that under this scheme, the parameter `number_of_energy_tokens_per_grid_pos_per_sweep` determines the *mean* number of energy tokens that should be distributed to each grid position.

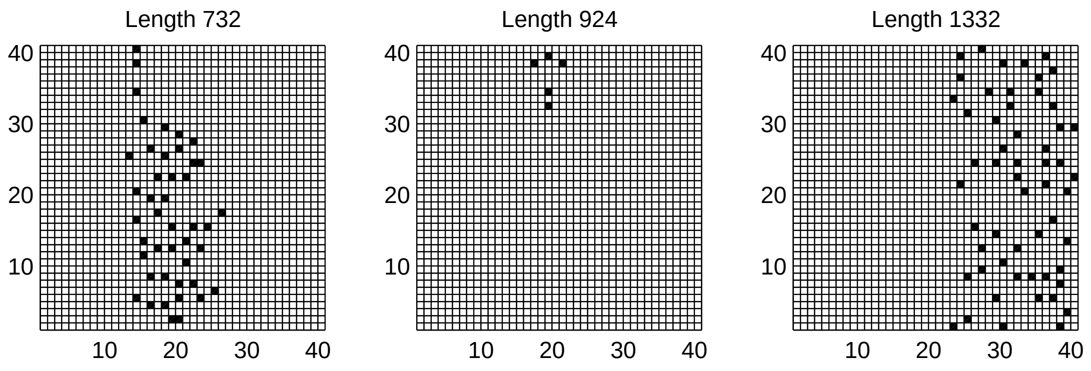


Figure 6.27: Large Energy Gradient, Run 4: Spatial Distribution of Programs of Different Sizes at Time Slice 900,000.

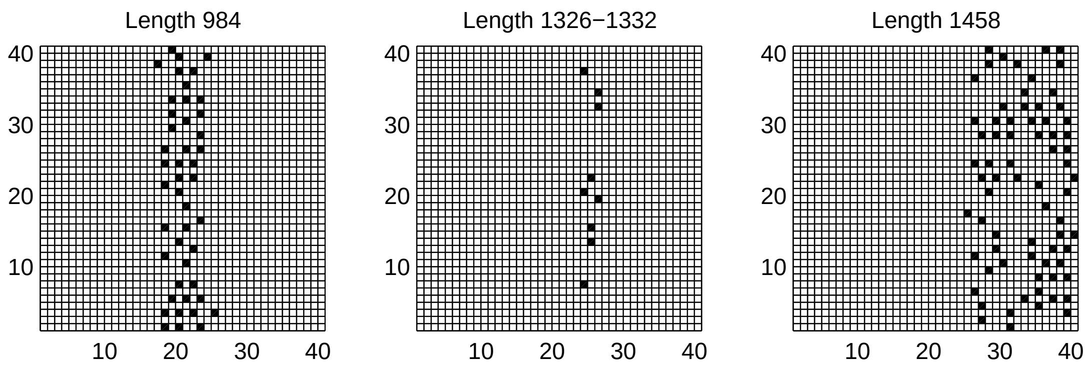


Figure 6.28: Large Energy Gradient, Run 6: Spatial Distribution of Programs of Different Sizes at Time Slice 900,000.

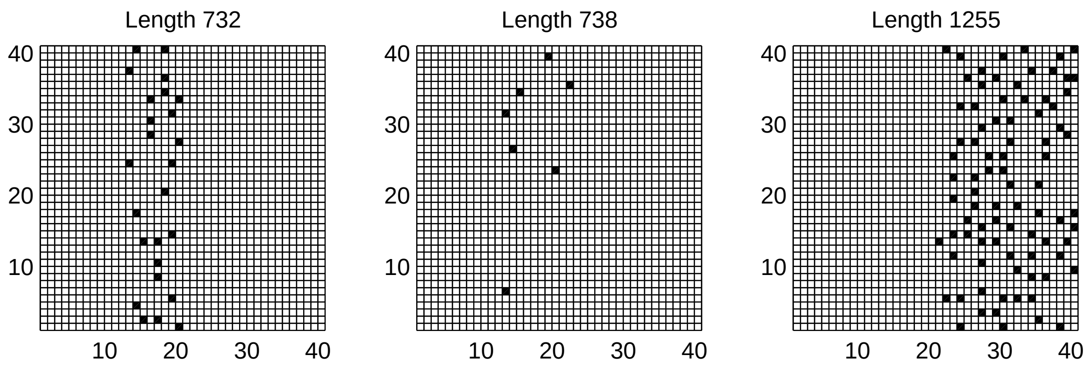


Figure 6.29: Large Energy Gradient, Run 9: Spatial Distribution of Programs of Different Sizes at Time Slice 900,000.

Parameter	*Value*
number_of_energy_tokens_per_grid_pos_per_sweep	90
max_energy_tokens_per_grid_pos	150
energy_distribution_scheme	random
energy_distribution_random_chunk_size	30

Table 6.10: Non-default Parameter Values for Random Energy Experiments.

Results

Under these conditions, the population survived for the duration of the run in six out of the nine experiments. Of the three in which the entire population died out (Runs 2, 3 and 9), the extinction occurred fairly early in the run in two cases (by time slice 78,600 for Run 2, and time slice 141,300 for Run 3), but at the very end of the run in the other case (Run 9, time slice 971,247).

Of the six which persisted for the whole run, three maintained Class 2 dynamics for the duration (Runs 1, 4 and 6), and three switched to Class 1 dynamics before completion (Runs 5, 7 and 8). The results were generally very similar to the standard runs, but there are a few points to note.

One difference was that programs retained the ancestral ability to move around the environment—an ability which is generally lost in the other sets of experiments reported. Indeed, analysis of the dominant end-of-run programs in the Class 2 runs (1, 4 and 6) revealed that they had all actually gained at least one more `move` instruction[5] somewhere in their code (but, interestingly, not within the copy loop). The explanation for retaining this capacity for movement is presumably because there is at least a chance that a nearby position will be rich in energy, while a position at which the program has been situated for some time will generally have fairly low energy reserves.

Another difference was that the population sizes in the runs were generally somewhat larger than usual, at least while they were in the Class 2 state. This is attributed to the greater mean energy levels in the environment.

Finally, in two of the three runs which switched to Class 1 dynamics (Runs 7 and 8), the population size in the Class 1 state was lower than normal (fewer than 150 organisms), and the diversity was also lower (approximately 20 different types of program). Analysis

[5] Or sometimes a `migrate` instruction, which is equivalent to `move` for a single-celled program.

of the programs in the population at the end of these runs shows that they had retained the ability to move, as in the Class 2 runs. This is in contrast to Run 5, which displayed more typical Class 1 dynamics, and in which the programs had lost the ability to move. Remember from Section 6.2.1 that Class 1 dynamics come about when programs grow so long that they drain energy from their neighbouring programs to the extent that the neighbours die. The stable configuration shown in Figure 6.10 allows for the maximum number of programs under such conditions. As the programs are motile in Runs 7 and 8, this stable configuration is not possible, so the population size is inevitably lower. However, another consequence of the programs' movement is that their offspring are distributed all around the environment rather than all being born in the same locality. This may provide a program with a slightly better chance than it would have were it sessile of propagating throughout the population (because at least *some* of its offspring might be born in favourable positions). In other words, the evolutionary potential of these runs has not been completely lost, which explains the lower diversity than is normal for Class 1 systems.

6.6 Reading Neighbouring Code

Programs in Cosmos cannot generally read the code of neighbouring programs. In contrast, programs in Ray's Tierra system do have this ability, and this is why parasites and other sorts of ecological interactions emerge fairly easily in it. A mechanism was built into Cosmos to simulate the reading of neighbouring code using the program's general communications input and output apparatus, as described in Chapter 4 (Sections 4.3.7 and 4.6.1). However, the mechanism is still more complicated than in Tierra, and would require somewhat more evolutionary innovation in order to be utilised effectively. A set of experiments was conducted in order to test the mechanism and see whether parasites could evolve in Cosmos. The non-default parameter setting is shown in Table 6.11.

Parameter	*Value*
`neighbouring_genomes_readable`	`yes`

Table 6.11: Non-default Parameter Value for Readable Neighbouring Code Experiments.

Results

No parasites or similar phenomena were observed in any of the nine runs. It appears that the mechanism used to simulate the reading of neighbouring programs' code was too complicated to be exploited by evolution. In particular, unlike in Tierra (see Section 3.3.1), there is no single mutation of the ancestor which would produce a functional parasite.

An analysis of the activity of the most numerous genotypes in each of the runs shows that they generally produced a much larger proportion of unfaithful offspring than did genotypes in the standard run. For example, the data recorded in the output file `morgue.dat` shows that the mean number of unfaithful reproductions for programs of the ancestor genotype 348AAAA was 0.277 (with standard deviation 0.099) across all nine runs. This compares to a mean of 0.010 unfaithful reproductions (with standard deviation 0.003) in the nine re-runs of the standard model, in which neighbouring code was not readable.[6]

This decrease in the copy fidelity of the programs was evidently caused by the mechanism for reading neighbouring code. Recall that this mechanism will attempt to bind a promoter onto the code of every available neighbour before it tries the host code. In eight of the nine runs, a state resembling Class 1 dynamics quickly arose, with the appearance of programs with length of order 1000 bits, and population sizes of order 400 individuals. The high proportional of unfaithful offspring produced by programs meant that the evolutionary potential of the runs was limited right from the start, and no trends for adaptive improvements of any sort were observed. In the other run (Run 7), the population was reduced to a single program by time slice 87,600, which survived until the end of the run.

In summary, the mechanism for reading the code of neighbouring programs failed to reproduce the results observed in Tierra. In particular, no parasites emerged during these runs.

[6] In contrast, the mean number of *faithful* reproductions of programs of the ancestor genotype 348AAAA were similar in the two sets of experiments. In the present experiment, the figure was 1.077 (standard deviation 0.045), and in the standard model re-runs, it was 0.993 (standard deviation 0.037).

6.7 Inoculation with Sexual Ancestors

All of the experiments reported so far have used the same hand-written self-replicating program, 348AAAA, as the inoculated ancestor. This program is fairly simple, and reproduces asexually, as described in Section 5.2.

In this final results section of the chapter, a set of experiments is described in which a more complicated ancestor program, 1314AAAA, was used. The most significant difference between this program and 348AAAA is that 1314AAAA reproduces *sexually.* The program is shown schematically in Figure 6.30, and a full listing is given in Section B.3.

The basic operation of the program is as follows. Most importantly, the program is divided into two sections (male and female). Under normal operation, any particular individual will only ever execute one of these two sections. The choice of which section it executes (i.e. the sex of the individual) is determined by a promoter; two different promoters are used, one specifying a male, and the other a female.

The female section works in a fairly similar fashion to the usual ancestor 348AAAA. The difference is that rather than copying the genome to the Nucleus Working Memory, it copies it to the Communications Working Memory. When the whole genome has been copied, the female then emits the copy into the environment as a message. At this point, the female's work is done, and it attempts to repeat this procedure indefinitely.

The male section begins with a section of code which causes the organism to wander around the environment for a predetermined duration, collecting energy.[7] The male then enters a behavioural loop, in which it seeks messages in the environment. During this phase, the male is sessile, but it looks for messages in all directions. It also checks its internal energy levels, and, if these are low, attempts to collect more energy from the environment. This phase continues until a message has successfully been recovered from the environment. When this occurs, the message is transferred to the male's Received Message Store. The start and end addresses of the message are calculated, and it is copied, one instruction at a time, into the male's Nucleus Working Memory. When the copying is complete, a calculation is performed to determine the sex of the

[7] This section was included for the practical reason that when a run is first inoculated, several hundred time slices pass before the females emit their first messages. During this time, collecting energy is the only useful task that the males can do.

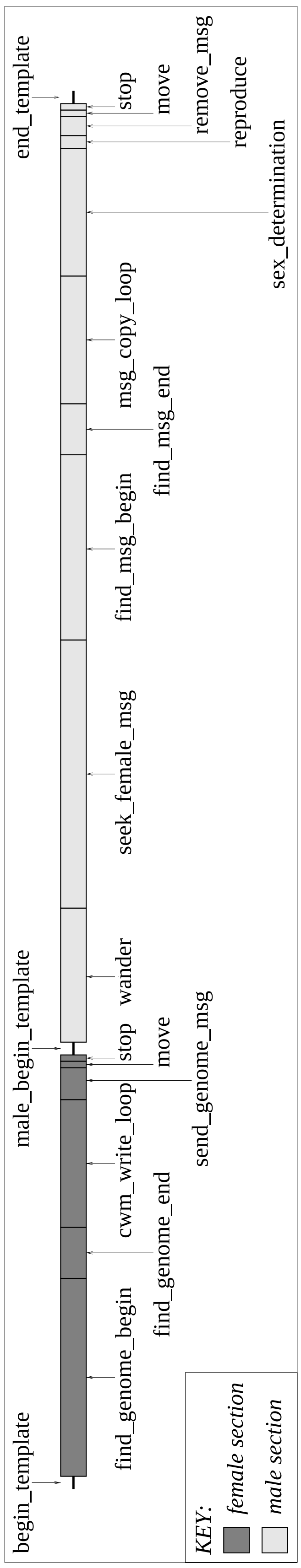


Figure 6.30: Schematic of the Sexual Ancestor, 1314AAAA.

offspring which is about to be produced. This is determined according to whether the male's internal energy store contains an odd or even number of energy tokens (i.e. sex determination is pseudo-random). Depending on this choice, an appropriate promoter is then manufactured for the offspring. At this stage, the offspring is finally produced. As always, the contents of the parent's Nucleus Working Memory becomes the genome of the offspring. The male parent finally removes the message from its Received Message Store, then repeats the whole process again.

To summarise, the whole genome is initially copied by the female, who transmits the copy as a message into the environment. The males attempt to pick up these messages. When they have done so, they produce an offspring whose genome is a copy of the received message. In this way, both males and females are required to propagate the species.

In the following experiments, half of the inoculated programs were given a male promoter, and the other half a female promoter. Males and females were placed in alternate positions in both directions, so they were regularly interspersed. The other non-default parameter settings used for these experiments are shown in Table 6.12. These values were arrived at after a series of trial-and-error attempts to reach a state where the population of programs in most runs survived for a reasonable length of time.

Parameter	*Value*
`number`	400
`rms_receive_search_area`	24
`energy_collection_scheme`	`private`
`et_value_constant`	10
`et_value_power`	0
`number_of_energy_tokens_per_collect`	15

Table 6.12: Non-default Parameter Values for Sexual Reproduction Experiments.

Results

Analysis of the output data revealed one basic result; the capacity for sexual reproduction was lost at a very early stage in all nine runs. By studying some of the individual genotypes from the runs, it was apparent that a minor mutation in the male section of the code responsible for collecting female 'spores' from the environment (the section labelled seek_female_msg in Figure 6.30) was sufficient to turn the male into a fully functional, independent, asexual organism. Mutations of this sort led to the sexual organisms being completely displaced by asexual organisms within the first 90,000 time slices in all runs, and indeed much earlier in most of them.

After the transition to asexuality had been made, subsequent improvements in the organisms were possible by the loss of the now redundant female section of the genome, and also by the loss of redundant sections of the male section (mostly within the areas labelled wander and seek_female_msg in Figure 6.30). The state of the system at the end of each of the runs can be classified as follows: in Runs 5 and 8, no progress had been made in removing redundant code from the asexual programs—they were still of the same length as the original sexual ancestor; in Runs 1, 2, 6, 7 and 9, most of the female section of the code had been removed from the programs, leading to programs with lengths in the range of 936–1098 bits (although in Run 7 this improvement only occurred within the final 50,000 time slices); and in Runs 3 and 4, parts of the redundant areas of the male section had been removed in addition to most of the female section, leading to programs with lengths in the region of 580–670 bits.

A final point to note is that in the programs alive at the end of Run 4, there was also evidence that a cancer had invaded the population. This was of the type where the extra instructions accumulating at the ends of the programs were energy collection instructions, as observed in a previous run discussed in Section 6.5.2. It is interesting that in both of the experiments in which this phenomena was observed, the parameter `energy_collection_scheme` was set to `private`. Recall that this means that programs can only extract energy from the (abiotic) environment, and not from their neighbouring programs. If such a cancer were to arise in a run in which programs could extract energy from their neighbours, the system would soon fall into Class 1 dynamics, as the amount of energy extracted would soon rise to a level where all such neighbours were killed. It is only when programs are prevented from extracting energy from their neighbours that this cancer can develop to the extent that it did in Run 4 of the current set of runs, and in Run 9 of the Private Energy Collection runs reported in Section 6.5.2.

6.8 Summary and Discussion

To end this rather long chapter, it might be useful to recap the more important results, and to discuss their significance. As well as reporting the experiments which have been performed, it is also important to consider what types of experiment might also be possible but are yet to be attempted. A number of possible lines of future experimentation are suggested at the end of the section.

Perhaps the most interesting result to come out of this research has been the analysis of the role of contingency in determining the course of evolution. In Section 6.1 we discovered that in a set of 19 runs of Cosmos which differed only in the number used to seed the random number generator, each one performed significantly differently, in a number of measures, from at least one third of the other runs. Using these results, it was decided that each Cosmos experiment in future should be run at least nine times in order to be confident of seeing a variety of results.

Despite the significant quantitative differences that arise, by chance, between many of the runs, we have seen that the system's behaviour can generally be categorised into one of two classes proposed by Bedau and colleagues [Bedau *et al.* 98]. This finding is reminiscent of the evolutionary dictum of "laws in the background and contingency in the details" discussed in Chapter 2 (Section 2.3.5), although in the present case even the background laws would appear to be rather dependent upon the details of the system's design.

During Class 2 dynamics, the population of programs is able to evolve fairly successfully. The chief methods by which adaptive improvements are made are by the accumulation of `et_collect` instructions within the programs (enabling them to collect more energy from the environment), and by the removal of redundant instructions. Note, however, that there is a limit to the extent that each of these methods can be applied; in Section 6.3.1 we saw that as the number of `et_collect` instructions in a program's copy loop was increased, the system moved towards Class 1 dynamics, where evolution effectively ceases; also, there is obviously a limit to the number of redundant instructions that can be removed from the programs. Once these limits have been reached, another kind of adaptive innovation would have to be discovered to allow the system to continue evolving, but we have so far seen little evidence for other innovations of this nature.

The other category of behaviour that was often observed was Class 1 dynamics. Analysis of runs which had switched to this state revealed that it is degenerative from an evolutionary point of view, because adaptive innovations are no longer able to spread through the population. The transition from Class 2 to Class 1 dynamics is usually brought about by the appearance of programs much longer than the ancestor (typically in the range of 600–1200 bits). Under the standard parameter settings whereby CPU-time is allocated to programs according to their length, longer programs such as these are able to collect more energy from the environment per time slice, giving them an advantage over their shorter ancestors. The onset of Class 1 dynamics can be avoided by using a length-neutral CPU-time allocation scheme (Section 6.4). The evolutionary activity of the system in the Class 1 state can also be improved by using a private energy collection scheme (Section 6.5.2).

Another interesting result was the speciation[8] of programs of different lengths, observed in the experiments in which a large gradient existed in the distribution of environmental energy across the grid (Section 6.5.3). This result suggests that a heterogeneous environment is an important factor in the emergence of speciation, and also demonstrates that it is not necessary to have a physical boundary for the population to differentiate into a number of species. In biological terms, the origin of species which live in the same place is called sympatric speciation. A number of processes have been proposed to explain this phenomenon, but the mechanisms involved are usually somewhat more complicated than the simple gradient in resource availability involved in these Cosmos runs [Maynard Smith 89]. Although the experiments reported here cannot really tell us much about this issue,[9] they do at least demonstrate the potential of spatially explicit, individual-based models to investigate such questions.

The experiments with sexually reproducing ancestors reported in Section 6.7 demonstrated that this capacity is easily lost. In all of the runs, the sexual ancestors were rapidly displaced by asexual programs. Further analysis revealed that this may have

[8] I am using the term 'speciation' very loosely here. For a start, as the programs are reproducing asexually, it is debatable whether the concept of species is applicable at all. Even if we do allow the term to be used, it might be argued that the programs in the different length groups are fundamentally very similar, and are more appropriately viewed as different varieties of the same species. However, remember that they did not only differ in their lengths, but also sometimes in their ability to move, and by the amount of energy they collected.

[9] They were not designed with this in mind. A proper test would need to be based upon an explicit set of assumptions, and would almost certainly require sexually reproducing programs.

been partially due to the ease at which the sexual ancestor could be mutated into a functional asexual program. However, the results still suggest that it might be hard to devise an environment in which sexual programs have a selective advantage.

Finally, another phenomenon which arose on a number of occasions was the so-called 'cancer' mutation (Sections 6.3.2, 6.5.2, 6.5.3 and 6.7). This mutation resulted in an extra instruction being tacked onto the end of a program's offspring at each reproduction. Over the generations the size of this tumour therefore grew larger and larger, resulting eventually in the death of the programs and the extinction of the population (unless the tumour happened to be composed of energy collection instructions). Although this result bears little relation to cancer in biological organisms, it does at least demonstrate how mutations which are harmful to the population in the long-run can initially be selected for and thereby invade the population.

Two things which we have *not* observed in evolutionary runs with Cosmos are the appearance of parasites or similar symbiotic phenomena, and the appearance of multicellular (parallel) programs.

The fact that parasites did not evolve in Cosmos, unlike in Tierra, was anticipated. Their emergence in Tierra depends upon some fairly specific aspects of the system's design and of the ancestor program used, as discussed in Chapter 3 (Sections 3.2.1 and 3.3.1). As the design of Cosmos differed from Tierra in these respects, it is much harder for evolution to 'discover' parasites in Cosmos. This highlights the importance that seemingly innocuous design features might have in determining the behaviour of artificial life platforms such as these, and suggests that we should take great care when claiming anything about the generality of results obtained from any particular one.

Recall from Chapter 4 that one of the original reasons for building Cosmos was to create a platform in which the emergence of multicellular programs might occur fairly easily. However, the experiments reported in this chapter have been concerned with exploring rather more basic aspects of the system's design, and we have not yet had the opportunity to look into the issue of multicellularity in a systematic way. As a passing note, developmental biologist Lewis Wolpert has suggested that multicellular organisms might originally have emerged in conditions where food was sparsely distributed in the environment.[10] When no food was available, a multicellular organism would be able to begin eating its own cells to survive until environmental food was available again. The experiments with environmental energy gradients and with random energy distribution reported in this chapter could be seen as a recreation of such a scenario, but no multicellularity was observed in them. However, a much more thorough investigation into this topic is required before anything can be said on the matter.

Although this chapter has been rather long, we have really only just scratched the surface of exploring the parameter space. We cannot claim to have thoroughly investigated the parameter space for those parameters discussed in this chapter, and indeed we have not even begun to investigate the effect of varying the parameters listed in Section B.1. Many of these concern the mechanisms for dealing with environmentally-conveyed messages, and various factors involved with multicellular (parallel) programs. The sexual ancestor reported in Section 6.7 did use environmentally-conveyed messages, but this ability was rapidly lost during the course of evolution.

All of the experiments except those reported in Section 6.7 used the same ancestor program, 348AAAA. There is an infinite variety of other ancestors which could have been used, some of which would have undoubtedly led to very different results. For example, one of the theories which has been put forward to explain the sudden origin and rapid explosion in diversity of multicellular biological organisms during the late Precambrian sees the evolution of heterotrophs (i.e. organisms which eat other organisms in order to obtain the complex organic compounds they require for metabolism) as the prime cause [Stanley 73]. Such a scenario could be recreated in Cosmos by introducing heterotrophic organisms (modelled as programs which kill other programs to steal their energy reserves),[11] in an effort to bring about a similar explosion in diversity. Indeed, initial experiments in this direction have been attempted, but it has so far proved impossible to engineer a viable community that includes heterotrophic programs.

A final feature of the model which has received little attention is the mapping between a program's genome (represented as a string of binary digits), and the instructions in

[10] This theory, which he named 'cannibalistic altruism', was discussed during a talk by Wolpert at the Royal Museum of Scotland, Edinburgh, on 20 February 1997.

[11] But note that in this case the prey organisms would only be a resource of energy, not of matter. This distinction is discussed in more detail in the next chapter (e.g. Section 7.1.4), along with possible evolutionary consequences arising from it.

the language provided. At present, there are 62 instructions in the Cosmos instruction set, and these are encoded using six bits (giving a total of 64 different possibilities). All of the runs reported in this thesis have used the same mapping, in which each of the 62 instructions is represented once, except the `move` instruction which appears three times (see Section B.4). There is therefore virtually no redundancy in the encoding, in contrast to the biological genetic code which encodes 20 amino acids with 64 possible codons. An alternative design has been developed, which uses a reduced instruction set. This consists of just 21 primary units which can be encoded on the genome. The full functionality of the existing system is maintained by allowing the primary units to form compound instructions. This is somewhat analogous to the way in which biological genomes encode just 20 amino acids, which, when decoded, are then assembled into a vast array of useful proteins. It may turn out that such redundancy would have a considerable effect upon the evolvability of the system. For example, important or frequently-used instructions in the language could be represented by multiple codons, thereby biassing the system to use these rather than some of the more obscure instructions. However, this design has so far not been implemented.

Chapter 7

Reappraisal of the Approach

"There is nothing wrong with a good illusion as long as one does not claim it is reality."

Howard Pattee ([Pattee 88] p.74)

In the previous two chapters, the behaviour of Cosmos has been analysed in considerable detail. However, as suggested in Section 6.8, it is still unclear exactly what the scientific value of these results is, in terms of what they can tell us about biological evolution, or about a more universal view of biology. It has been hard to escape the nagging feeling that many of the more interesting results we have seen have been dependent on rather specific features of the system's design. On top of this, even though considerable time and effort has been spent in analysing the system, we have only really scratched the surface of exploring the parameter space.

Over the course of this work, I have come to see a number of problems with the general approach that has been taken, if it is to be used as a scientific framework. In this chapter, I will discuss these problems and related issues, and suggest, in some fairly general ways, how the approach might usefully be modified and extended.

7.1 Problems with Tierra-like Models

In this section I discuss some of the major problems (or at least sources of concern) that I perceive with Cosmos and other Tierra-like models. These are summarised in the following list:

- Lack of an explicit theoretical grounding.
- Predefined organism structure.
- Restricted ecological interactions.
- No competition for matter or energy.
- Evolving a self-reproduction algorithm.

7.1.1 Lack of an Explicit Theoretical Grounding

Although Cosmos was designed to study evolution, and in particular (originally, at least) the evolution of multicellular organisms from unicellular ones, it was not built around any particular theory of what the important features of this transition might have been. Features such as the two-dimensional environment, energy tokens and so on were included for the reasons discussed in Chapter 4, but there were no coherent theoretical reasons for deciding which features should be modelled, and which should be left out. This weakness is not specific to Cosmos, but is shared by all of the other Tierra-like systems I have come across.

In describing the philosophy behind the Tierra system, Tom Ray explains that "rather than attempting to create prebiotic conditions from which life may emerge, this approach involves engineering over the early history of life to design complex evolvable organisms, and then attempting to create conditions that will set off a spontaneous evolutionary process of increasing diversity and complexity of organisms" [Ray 91] (p.373). However, in order to 'engineer over' several billion years of evolution, we would need to have a very good idea of the design and behaviour of the resulting organisms, and an understanding of why they had evolved in such a way (in order to know which aspects of their design and behaviour were the most important for us to model).[1] Unfortunately we do not possess such details of the organisms which existed immediately prior to the Cambrian explosion.

I am certainly not the first person to criticise artificial life models on these grounds (see the references in Section 3.1.3). For example, Howard Pattee warns that "simula-

[1] Ray himself recognises these difficulties, but is more optimistic that they can be overcome [Ray 91] (p.399).

tions that are dependent on ad hoc and special-purpose rules and constraints for their mimicry cannot be used to support theories of life" [Pattee 88] (p.68).

To be fair, Ray does offer a definition of life in his work with Tierra. He says "I would consider a system to be living if it is self-replicating, and capable of open-ended evolution" [Ray 91] (p.372). From the discussion of the concept of life in Chapter 2 it will be clear that this definition is controversial. Now, this lack of agreement is not in itself a particular problem when formulating a scientific model, as long as a precise definition has at least been proposed (and is falsifiable). However, in Section 7.2 I will argue that the concept of self-replication, when simply stated, is *not* a precise concept, and is potentially the source of much confusion. Additionally, determining necessary and sufficient conditions for a system to be capable of open-ended evolution is half of the problem, and Ray's definition tells us nothing about how we should go about building such a system. This being the case, the definition does not provide an adequate theoretical grounding for Tierra and similar models.

A feature of Ray's definition of life is that it does not define what sorts of environments might support life, or the sorts of ecological interactions which should be available. In Section 2.3.1 it was suggested that ecological processes may play a primary role in promoting evolutionary progress and the evolutionary increase of complexity of some organisms. Furthermore, in Section 3.2.2 we saw that some of the most spectacular examples of artificial evolution rely upon coevolutionary interactions between organisms. This suggests that we should think more carefully about such issues, rather than treating them in the rather ad hoc way that has often been used in the past. This point has been made by Pattee, who says:

> "... life must have arisen and evolved in a nonliving milieu. In real life we call this the real physical world. If artificial life exists in a computer, the computer milieu must define an artificial physics ... What is an artificial physics or physics-as-it-could-be? Without principled restrictions this question will not inform philosophy or physics, and will only lead to disputes over nothing more than matters of taste in computational architectures and science fiction." [Pattee 95a] (p.29).

The ad hoc feel of Tierra-like systems is a direct consequence of this lack of theoretical grounding. The unmanageable parameter space of many of them can also be attributed to this lack of direction. As a result of these weaknesses, even if interesting behaviours are observed in these systems, we are unlikely to be able to explain why (as was demonstrated in Sections 3.3 and 6.8). It may be that a model of self-replication and open-ended evolution is necessarily somewhat complex, but, even if this is so, the theoretical framework upon which it is built should prescribe the implementational details as much as is practically possible.

7.1.2 Predefined Organism Structure

Organisms in Tierra, Cosmos and similar systems have a predefined, and generally fixed, structure. In Tierra, this structure includes the program's code, various registers, an instruction pointer and a stack; in Cosmos, it is somewhat more complicated (see Section 4.3). It has already been suggested that an attempt to engineer over the early history of life would be problematic. In particular, it is difficult to say how appropriate these structures are as models of any given stage of biological or chemical evolution, or indeed how appropriate they are for modelling living organisms at all.

This is of course related to the lack of theory underlying the model. By imposing a fixed, predefined structure upon the organisms, designers of such systems risk introducing their own prejudices, and restricting the evolutionary potential of the system (as mentioned in Section 3.3.3). In particular, the organism structures used in most of these systems would appear to restrict their potential for modelling some of the important evolutionary phenomena discussed in Chapter 2, such as symbiogenesis and hierarchical evolution. The lack of a theory to govern the design of the environment and the kinds of ways in which organisms may interact does not help in this respect either.

In addition to the predefined organism structure, the decoding of instructions in Tierra is also 'hard-wired' into the system's operating system. Now, from an epistemological point of view, Howard Pattee points out that symbolic information (such as that contained in an organism's genes) has "no intrinsic meaning outside the context of an entire symbol system as well as the material organization that constructs (writes) and interprets (reads) the symbol for a specific function, such a classification, control, construction, communication ..." [Pattee 95b]. He argues that a necessary condition for an organism to be capable of open-ended evolution is that it encapsulates this entire self-referent organisation (Pattee refers to this condition as *semantic closure*). From this it follows that organisms should be constructed "with the parts and the laws of an artificial physical world" [Pattee 95a] (p.36).[2] In other words, the interpretation (phenotype) of the symbolic information (genotype) of an artificial organism should be constructed and act within the artificial physical environment of the system. Additionally, if the system is to model the *origin* of genetic information, then the genotype itself must also be embedded within the environment; that is, the complete semantically-closed organisation—the *entire organism*—must be constructed and act within the physical environment. This issue of the 'embeddedness' of organisms in their environment will be analysed in more detail in Section 7.2.3.

7.1.3 Restricted Ecological Interactions

On top of the problems associated with imposing a predefined organism structure, a related problem concerns the extent to which organisms are free to interact with their environment (including other organisms).[3]

In Cosmos, a program does not generally have read and write privileges to memory external to its structure. The only exceptions are when it reproduces (i.e. a new program is written to the environment), and when it sends and receives messages from the environment. In Tierra, the restrictions on read privileges are more relaxed (programs can read the code of neighbouring programs), but the write privileges are similar to those in Cosmos. Moreover, in Cosmos (as in Avida [Adami & Brown 94], Computer Zoo [Skipper 92], and similar models), the two-dimensional environment in which the programs live is unrelated to the address space of the programs; the programs live simultaneously in two different worlds. The situation is not as extreme in Tierra, but even there some aspects of the programs (e.g. registers and instruction pointers) are not physically represented in the shared environment.

Compare this situation to von Neumann's work with self-reproduction using cellular

[2] Although he also stresses that "some epistemic principles must restrict physics-as-it-could-be if it is to be any more than computer games" [Pattee 95a].

[3] By 'environment' I mean the shared space in which (at least some aspects of) all of the organisms exist, which I will refer to as the 'arena of competition' (see Section 7.2.3), together with the universal 'laws of physics' of this space which determine how entities within it act and interact.

automata (discussed in Section 3.2), where automata are entirely represented (constructed) in a single, shared space, and interact entirely within this space as well. The interactions are defined by the CA's low-level transition rules, and do not, for example, respect boundaries between the higher-level self-reproducing automata. This is of course similar to the situation of biological organisms, which have the freedom to interact with their environment in a variety of ways only limited by the laws of physics (although organisms themselves generally evolve mechanisms to restrict such free interaction).

7.1.4 No Competition for Matter or Energy

The arguments in the previous sections are bringing us to the fundamental question of how matter is represented in these models. If there is a representational distinction between organisms and the environment in which they exist (which comes about by having a hard-wired organism structure and by restricting ecological interactions), some of the fundamental concepts associated with living beings, such as competition for resources, self-maintenance and so on, can become ill-defined.

One of the tenets of Darwinism is that organisms are engaged in a struggle for existence. However, it is difficult to identify the precise nature of this struggle, as Darwin himself observed. In *The Origin of Species*, he wrote "What checks the natural tendency of each species to increase in number is most obscure ... The amount of food for each species of course gives the extreme limit to which each can increase; but very frequently it is not the obtaining food, but the serving as prey to other animals, which determines the average numbers of a species" [Darwin 59] (pp.119–120). Thus, an important aspect of the struggle for existence is the obtaining of food not from passive, abiotic sources, but through predator-prey relationships. In the biological realm, the struggle for existence involves organisms *killing* other organisms, because the very matter from which they are constructed is a valuable resource of matter and energy. This competition is therefore very much a matter of life or death.

It may be difficult to identify the precise nature of the struggle for existence, but it seems likely that the numerous forms of competition can be categorised in terms of a small number of fundamental resources. In the biosphere, a (speculative) list might be:

matter, energy, space and information.[4]

Tierra-like systems generally do not have any notion of competition for matter. Indeed, they cannot really be said to have a notion of matter at all, in terms of fundamental units from which all structures are built, and which are conserved during reactions. Instead, when a program is writing a copy of itself, it can produce the copied instructions spontaneously rather than first having to collect a copy of the individual instruction from somewhere else in memory. In other words, the individual instructions are represented as states of specific memory locations, rather than as units of matter.[5] The only fundamental competition that exists in Tierra is for space (memory) into which to divide. This is allocated at a global level by the Tierran operating system's memory allocation services. The programs are not even really competing for energy (CPU-time), because any number of programs are allowed to execute instructions at each time slice; the limiting factor is how many programs can fit into the available memory.

Programs in Tierra *can* act as resources for other programs in another way, by acting as 'library code' which can be read by their neighbours (as happens in the evolution of parasites). In other words, they can act as information resources. However, this is not as strong an ecological interaction as when one organism acts as a resource of matter or energy, in the sense that acting as an information resource is not a direct matter of life or death for the host.

Cosmos introduces competition for energy through the 'energy token' mechanism. These tokens are distributed across the environment, and programs must compete for them locally in order to be given the chance to run further instructions. However, as programs are read-protected as well as write-protected, they cannot act as resources of library code for their neighbours. In these respects, Cosmos therefore has some advantages over Tierra, but also some disadvantages.

Ray has suggested that introducing the notion of "conservation of instructions" would be an interesting extension to Tierra [Ray 91] (pp.399–400). Morris has also suggested

[4] For example, a virus requires information contained in its host's genome in order to reproduce. This information is more than the matter from which the host's DNA is constructed; it involves a particular ordering of matter.

[5] This is also the case in von Neumann's cellular automata models. In contrast, structures in his kinematic model, as in models proposed by various others (such as Myhill, and Holland's α-Universes), are constructed from atomic units of matter (see Chapter 3).

a similar extension to Hofstadter's Typogenetics [Morris 88] (p.387). However, I am not aware that these suggestions have been implemented.

The issue of how energy is represented in these systems is perhaps more controversial. Some would claim that it is essential to model certain fundamental energetic considerations (e.g. [Morán *et al.* 97], [Ruiz-Mirazo *et al.* 98]). An important point to note is that *all* artificial life platforms have to model energy at the basic level of determining when a component can perform an action (e.g. when a program can execute an instruction, as determined by the system's CPU-time allocation scheme). Without a theoretical grounding, any scheme is just as arbitrary as any other (e.g. the schemes in Tierra and Cosmos). Ideally, the system's design should be based upon explicit considerations of how energy should be modelled.

Only when one organism can act as a resource of energy and matter for other organisms do ecological concepts such as food webs and trophic levels become relevant. Furthermore, without competition for matter and energy, or other interactions whereby one organism can benefit by physically damaging another, it is doubtful whether any selection pressure exists for the evolution of self-maintaining (and eventually fully autopoietic) organisms. As mentioned in Section 6.8, it has even been suggested that the emergence of heterotrophs (organisms which eat other organisms) might have been the prime cause of the Cambrian explosion [Stanley 73]. If this is so, more careful consideration of these matters in artificial life systems is surely required.

These considerations of the specific resources for which individuals are competing may not be necessary in the context of open-ended evolution in general, but they probably are relevant for modelling other processes commonly associated with biological life. Of course, the extent to which these deficiencies are considered important will depend upon one's conception of life, as discussed in Chapter 2. The extent to which the various phenomena associated with biological life can be recreated in artificial life systems with or without features such as competition for matter and energy is a matter to be resolved empirically. In doing so, we can develop a better understanding of the fundamental nature of life. We will return to these topics in Sections 7.3.2 and 7.3.3.

7.1.5 Evolving a Self-Reproduction Algorithm

Another concern with the Tierra approach is the fact that the individual organisms are encodings of self-reproduction *algorithms.* This is related to the concern over imposing a predefined structure on the organisms, but is a more fundamental worry. The fact that the copying process is explicitly encoded in the individual organisms in Tierra-like systems, rather than being implicit in the physical rules of the system, might have serious consequences for the evolvability of the organisms. Additionally, Tierran programs do not appear, at least at first sight, to have a clear genotype-phenotype distinction, and this may also hinder their evolvability.

These issues are a source of concern, but I have not yet offered any detailed analysis to suggest how great a problem they actually represent, if indeed they represent a problem at all. As these issues relating to self-reproduction are fundamental to the goal of creating an open-ended evolutionary system, they deserve a more thorough analysis. Such an analysis is provided in the following section.

7.2 Self-Reproduction and Evolution Revisited

In this section, I will first consider some general issues involved in the concept of reproduction. From there, the sort of self-reproduction schemes used by Tierra-like systems, and the self-reproduction architecture proposed by von Neumann, will be located within this more general picture. This will lead on to a discussion of some more specific issues relating to reproduction architectures that are capable of supporting open-ended evolutionary processes.

7.2.1 Some Definitions

To begin, I present definitions of some of the terms that will be used in the following discussions. These definitions were proposed by Lars Löfgren, and emphasise the relationship between a reproducing entity and the environment in which it is embedded [Löfgren 72].

Definition 1 (Productive) An object $\mathcal{A}$ *is productive in a surrounding* $\mathcal{S}$ if $\mathcal{A}$ causes $\mathcal{S}$ to produce another entity $\mathcal{B}$, symbolised by $\mathcal{A} \to d \overset{\mathcal{S}}{\to} \mathcal{B}$ and read as follows: The configuration (output state) d of $\mathcal{A}$ forces $\mathcal{S}$ to produce $\mathcal{B}$. Here d can be considered a description of $\mathcal{B}$ relative to $\mathcal{S}$.

The reference to the configuration d in this definition is to that part of the object $\mathcal{A}$ which explicitly directs $\mathcal{S}$ to produce $\mathcal{B}$. To anticipate the discussion of explicit versus implicit encoding of (re-)production processes in the following sections, d is that attribute of $\mathcal{A}$ which is an explicit encoding of part of the production process. As the above definition implies, some of the production process may also be implicit in $\mathcal{S}$.

Löfgren points out that the product $\mathcal{B}$ of a productive object $\mathcal{A}$ in surrounding $\mathcal{S}$ may itself be productive, or it may not be. If $\mathcal{B}$ *is* itself productive, he calls $\mathcal{A}$ *re*productive. The definition of reproduction is as follows:

Definition 2 (Reproductive) An object $\mathcal{A}$ *is reproductive in a surrounding* $\mathcal{S}$ if there are objects $\mathcal{A}_i$ with descriptions d_i such that

$$\mathcal{A} \to d_1 \overset{\mathcal{S}}{\to} \mathcal{A}_1 \quad \text{and} \quad \mathcal{A}_i \to d_{i+1} \overset{\mathcal{S}}{\to} \mathcal{A}_{i+1} \quad \text{for } i = 1, 2, 3, \ldots$$

Note that these definitions are intentionally phrased in a way that emphasises that an object $\mathcal{A}$ which is productive in a surrounding $\mathcal{S}$ may rely upon some properties of $\mathcal{S}$ to achieve the production.

Following on from the previous two definitions, we can now provide a definition of self-reproduction:

Definition 3 (Self-Reproductive) An object $\mathcal{A}$ *is self-reproductive in a surrounding* $\mathcal{S}$ if $\mathcal{A}$ produces a copy of itself in $\mathcal{S}$.

Löfgren points out that the notation 'self-reproductive' is actually logically redundant; the term 'self-productive' has the same meaning according to our definitions. However, I will stick with the term 'self-reproductive' in the following, as this is the terminology commonly used in the literature.

Within this framework, Löfgren examines the concept of self-reproduction from a logical point of view. He points out that "there are two distinct but natural interpretations.

One is to say that $\mathcal{A}$ is self-reproductive if, first of all, $\mathcal{A}$ is reproductive and, second, $\mathcal{A}$ reproduces a copy of itself. The other interpretation is that $\mathcal{A}$ is self-reproductive if $\mathcal{A}$ is reproductive by itself, that is, $\mathcal{A}$ is reproductive in a surrounding $\mathcal{S}$ with no properties" [Löfgren 72] (p.360).

It is worth discussing this distinction a bit further, as it is the source of some confusion. Löfgren refers to the former case as 'second-level reproduction', and to the latter as 'complete self-reproduction'. It has been argued from a logical-mathematical point of view that complete self-reproduction is paradoxical; for example, Löfgren cites Wittgenstein's argument that no function can be its own argument [Wittgenstein 21] (esp. aphorism no. 3.333), and work by Rosen (e.g. [Rosen 59]).[6] Second-level reproduction, however, implies no such paradox. Löfgren observes:

> "It would seem that many misunderstandings concerning the concept of self-reproduction are due to the different meanings which are commonly attached with it. Von Neumann [von Neumann 66] and Penrose [Penrose 58], for example, use the word self-reproduction for a second-level reproduction, whereas [others, such as] Rosen [Rosen 59], use self-reproduction in a complete sense.
>
> In ordinary biological language the name self-reproduction is mostly used for second-level reproduction, for example, when the mechanisms of cell-division are used to explain the 'self-reproducing' properties of the cell. That no logical difficulty arises in connection with this type of 'self-reproduction' is well known." [Löfgren 68] (pp.423–424).

In the following discussion, I will only be concerned with second-level reproduction, and will use the general term self-reproduction to implicitly refer to this meaning rather than to complete self-reproduction.

A final point to note before delving further into some of the issues involved with systems capable of supporting productive, reproductive and self-reproductive objects, is that some of the examples I will use are logical systems (e.g. cellular automata and

[6] Although Löfgren has shown that complete self-reproductive functions can exist through axiomatisation [Löfgren 68].

Tierra), whereas others are material systems (e.g. biological cell division).[7] The distinction is unimportant for most of the discussion. However, it is relevant when talking about objects which can direct their own production (auto-reproduction) compared to those which rely on the existence of auxiliary objects in the environment (assisted-reproduction), as I will discuss in Section 7.2.2.

7.2.2 General Issues of Reproduction

The description of self-reproduction in the definitions of the previous section is stated in terms of the configuration d of object $\mathcal{A}$ *forcing* surrounding $\mathcal{S}$ to produce a copy of $\mathcal{A}$. It says nothing of how exactly d can (or should) achieve this feat, although this is a rather fundamental question when comparing different sorts of reproduction. Indeed, when looking at any sort of reproduction, I think it is useful to look at the process by which reproduction is accomplished in (at least) three different ways:

1. The degree to which the algorithm for reproduction (the way in which the process is specified and controlled) is *explicitly encoded* on the configuration being reproduced (cf. d in Definition 1), rather than being *implicit in the physical laws* of the world (cf. $\mathcal{S}$ in Definition 1).

2. Whether reproduction happens purely by the action of the physical laws of the world on the configuration to be reproduced (*auto*-reproduction), or whether it also requires auxiliary physical (or logical) machinery (*assisted*-reproduction). I use the term auto-reproduction rather than self-reproduction here because the latter is often used less specifically. I wish to emphasise that this auto-assisted distinction is only one of a variety of issues involved in the general concept of reproduction.

3. The number of different configurations that exist, connected by mutational pathways, that are capable of reproducing their specific form (i.e. the distinction between *limited hereditary* reproducers and *indefinite hereditary* reproducers). From the point of view of an individual reproducer, this can be expressed in

[7] The important distinction between these two types of system in the present context is that logical systems manipulate states, whereas material systems manipulate matter. Reproduction in material systems, in contrast to logical systems, therefore requires an object to collect the 'raw materials' to build a copy of itself.

terms of the proportion of all possible mutations it may experience that will result in the production of distinct, yet viable, reproducers.

There are a number of points to note about these distinctions. First it should be said that (3), in contrast to (1) and (2), does not properly relate to individual reproducers *per se*, but rather to lineages of reproducers. It is therefore not relevant when considering self-reproduction in and of itself, but is an important factor when considering the evolutionary potential of a class of reproducers.

Secondly, with regard to distinction (2) in the context of material as opposed to logical systems, I do not consider the fact that objects in material systems need to collect raw materials to be relevant. As long as the surrounding $\mathcal{S}$ ordinarily contains sufficient raw materials for a reproducing object $\mathcal{A}$ to build a copy of itself, and that the configuration d of $\mathcal{A}$, and the surrounding $\mathcal{S}$ between them effect the collection of these materials to build the copy without further assistance, then our definition of self-reproduction given in Section 7.2.1 is still satisfied.

The distinction between auto- and assisted-reproduction is a dichotomy, but the other two distinctions each define a spectrum of possibilities. The distinctions are generally independent of each other, although the more explicitly encoded the reproduction algorithm is, the less likely, in general, it is to be an indefinite hereditary reproducer (because of the increased chance of mutations disrupting the copying process; see Section 7.2.3).

Figure 7.1 shows how some of the reproducers that have been discussed so far can be categorised according to each of these three distinctions. The diagram is not supposed to be quantitatively accurate (not least because the limited-indefinite heredity axis is in fact infinitely long, and also because I have not offered any way of quantifying these factors), but I have tried at least to highlight the general relationships between different types of reproducers according to each of the three distinctions.

There are a number of points about this diagram that require further explanation:

- Tierran organisms and von Neumann's self-reproducing automata are placed midway along the limited-indefinite hereditary scale because, although both representations are *capable* of supporting universal computation in principle, only muta-

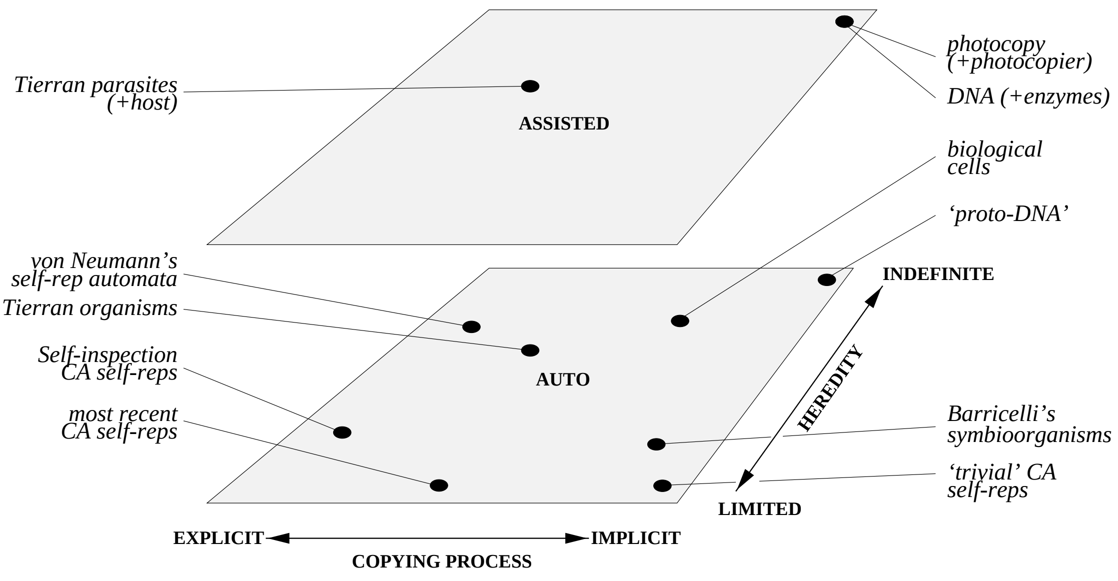


Figure 7.1: Categorisation of Reproducers.

tions which retain the ability to reproduce will be *viable*. In particular, in von Neumann's architecture (described in Section 3.2.1), a mutation which affects a section of the tape which encodes the constructing automaton **A**, the copying automaton **B**, or the control automaton **C**, will generally disrupt the ability of the combined automaton to produce viable offspring. Likewise, in Tierra, a mutation which affects a section of the program which encodes the self-reproduction algorithm will generally disrupt the ability of the program to reproduce. We will return to these issues in Section 7.2.3.

- Trivial cellular automata self-reproducers (e.g. where the state of a single cell is reproduced in neighbouring cells purely due to the CA's transition rules) are, in general, limited hereditary reproducers, because even though a single state may be able to reproduce, a compound set of states will usually not be able to reproduce as a whole. Notice that in much of the recent artificial life work with self-reproduction (e.g. [Langton 84] and other studies mention in Section 3.2.1), the distinction between trivial and non-trivial self-reproduction is perceived to be a distinction on the implicit-explicit axis. However, from an evolutionary point of view, the limited-indefinite heredity axis is clearly the most relevant. Indeed, this is exactly what von Neumann himself says:

"One of the difficulties in defining what one means by self-reprod-

> uction is that certain organizations, such as growing crystals, are self-reproductive by any naive definition of self-reproduction, yet nobody is willing to award them the distinction of being self-reproductive. A way around this difficulty is to say that self-reproduction includes the ability to undergo inheritable mutations as well as the ability to make another organism like the original" [von Neumann 49] (p.489).

Barry McMullin has presented an enlightening discussion on the history of the confusion over von Neumann's work, which he refers to as the 'von Neumann Myth' (see, for example, Section 4.2.7 of [McMullin 92a]). One result of this confusion has been that the majority of subsequent research concerning this issue of trivial self-reproduction has concentrated on the implicit-explicit distinction, rather than the limited-indefinite heredity distinction.[8]

- DNA reproduction is assisted, because it can only do so with the aid of a host of enzymes to control the unwinding of the double helical structure, the polymerisation of the individual bases of the new molecule, etc. (see Section 7.2.3). At the same time, DNA is, in the presence of suitable enzymes, capable of indefinite heredity assisted-reproduction; the enzymes are able to copy any DNA double helix, no matter what sequence of bases it comprises. Within this context, the copying process is implicitly encoded in the DNA's environment (in the enzymes which support the reproduction process, and in the physical laws governing the inherent bonding affinities of the bases) rather than being explicitly encoded upon individual strands on DNA. In contrast, a cell as a whole can be considered an auto-reproducer, as it can completely direct its own reproduction (in the presence of sufficient energy and raw materials from the environment), but its hereditary potential is slightly more restricted than DNA assisted-reproduction,

[8] Any attempt to classify a reproducer as trivial or non-trivial according to the explicit-implicit distinction is bound to be somewhat arbitrary, because we are generally considering examples of second-level reproduction rather than complete self-reproduction (Section 7.2.1). Therefore, the surroundings will always play some role in bringing about the reproduction of the object in question. There is considerable irony in much of the recent work relating to this issue. For example, Langton not only ignores von Neumann's actual solution to the 'problem' of trivial reproduction, but furthermore says that "von Neumann's work suggests an appropriate criterion ... : the configuration must treat its stored information in ... two different manners ... : *interpreted*, as instructions to be executed ..., and *uninterpreted*, as data to be copied" [Langton 84] (p.137). As far as von Neumann's actual analysis of the subject is concerned, this distinction between interpreted and uninterpreted is only important insofar as it gives the reproducing automaton the capacity to support inheritable mutations, potentially leading to the evolution of more complicated and more efficient machines.

because some mutations may disrupt the ability of the cell to reproduce. Similarly, each of Barricelli's symbioorganisms (described in Section 3.2.2) can be considered, collectively, as an auto-reproducer, although individual digits within the symbioorganism are assisted-reproducers. The same analysis can be applied to collectively-autocatalytic reaction networks (e.g. [Kauffman 93]).

- Most importantly, I have placed the hypothetical 'proto-DNA' (i.e. a desirable seed for open-ended evolution) in the auto-implicit-indefinite hereditary corner of the space. The seed should be auto-reproducing (i.e. not rely upon auxiliary machinery) if it is to have a reasonable chance of spontaneously emerging, and it should be an indefinite hereditary reproducer to support an on-going, open-ended evolutionary process. The requirement that it be an indefinite hereditary reproducer is most easily fulfilled if it reproduces implicitly. I will talk about this in more detail in the next section.

7.2.3 Self-Reproduction and Open-Ended Evolution

In the previous section, discussion was given to issues relating to reproduction in general. I now wish to focus upon issues of reproduction in the specific context of evolution. In this section I will concentrate on a number of these issues in turn.

Trivial versus Non-Trivial Reproduction

As already noted, in terms of the evolutionary potential of the organisms, the most relevant distinctions relate to the heredity axis, and not the implicit-explicit axis.

Von Neumann's work on self-reproduction concerned the question of how machines might be able to evolve increased complication in order to perform increasingly complex tasks. This is why his design for a self-reproducing machine had to be capable of universal construction, and why it was designed in such a way that it could withstand some kinds of mutations.

It has already been stated that much confusion exists over von Neumann's goals, and it is often assumed that his work was a general treatment concerning the topic of self-reproduction as a whole (i.e. concerning the entire space of possibilities depicted in Figure 7.1). By reading his lecture notes and papers (e.g. [von Neumann 66], [Aspray & Burks 87]), however, it is quite clear that he was primarily interested in self-reproduction that could lead to open-ended evolution (i.e. the rear portion of the lower plane in Figure 7.1).

This confusion has led to considerable debate over the 'problem' of trivial reproduction. The problems arise when the issue of trivial versus non-trivial reproduction is taken as relating to the implicit-explicit axis. If we adopt von Neumann's own suggestion that it is the limited-indefinite heredity axis which defines the most interesting distinction, most of the arguments over trivial reproduction in the literature can be seen as irrelevant, or at best subsidiary, to the real issues. Of course, new issues arise when we adopt this new viewpoint. I will discuss some of them in the remainder of this section.

Genetic Reproduction versus Self-Inspection

Von Neumann's architecture was designed specifically to allow for a possible increase in complexity and efficiency of machines by evolution. However, even if we accept that his design *is* a solution to this problem, it is by no means the only conceivable solution, as von Neumann himself was well aware. In particular, he also discussed the possibility of a machine which built a copy of itself by actively inspecting its parts, without the need for this design information to be duplicated on a tape (i.e. without a 'genetic' description). Indeed, systems which reproduce by self-inspection have been designed by Laing, e.g. [Laing 77], and by Ibáñez and colleagues [Ibáñez *et al.* 95].

Although he certainly did not *prove* that reproduction by self-inspection could not support open-ended evolution, von Neumann did suggest a number of reasons why his genetic architecture would be a more powerful and more general design for this purpose. First of all, as mentioned in Section 3.2.1, he noted that the essential feature which allowed his automata to overcome the otherwise seemingly valid rule that machines are necessarily superior (in size and in organisation) to their output, was that they contained a general copying automaton **B**, which was capable of copying any linear tape [von Neumann 66] (p.121). Although **B** is of fixed, finite size, it is able to copy a tape of any size. Now, this action of copying a tape is essentially reproduction by self-inspection, but this is generally a straightforward task for a linear tape. The major problems arise when trying to copy a two- or three-dimensional structure by the same method, for example in specifying the precise spatial relationships between parts.

Von Neumann also pointed out that self-inspection requires that we have a representation which is 'quasi-quiescent' in the sense that it can be read (for the purposes of copying and possibly for interpretation) without being essentially disturbed. With a separate genetic description, we only require that this description is quasi-quiescent, but copying by self-inspection would require that the whole structure to be copied would have this quasi-quiescent property. In general, however, most machines would not have this property, nor would we want to restrict ourselves to only considering those machines which did. In conclusion, von Neumann says:

> "To sum up, the reason to operate with 'descriptions' ... instead of the 'originals' ... is that the former are quasi-quiescent (i.e. unchanging, not in an absolute sense, but for the purposes of the exploration that has to be undertaken), while the latter are live and reactive. In the situation in which we are finding ourselves here, the importance of descriptions is that they replace the varying and reactive originals by quiescent and (temporarily) unchanging semantic equivalents and thus permit copying. Copying, as we have seen above, is the decisive step which renders self-reproduction (or, more generally, reproduction without degeneration in size or level of organization) possible." [von Neumann 66] (pp.122–123).

From a biological perspective, Waddington has made the same point. While discussing possible reasons for the universal adoption of genetic architectures for self-reproduction by biological life, he suggested that the issue "is presumably related to the problem [of] how to combine a store which is unreactive enough to be reliable, with something which interacts with the environment sufficiently actively to be 'interesting'" [Waddington 69] (p.118).

As to the nature of the information encoded on the tape, von Neumann suggested that "it is better not to use a description of the pieces and how they fit together, but rather a description of the consecutive steps to be used in building the automaton" [von Neumann 49] (p.486). In other words, the information should be in the form of a developmental 'recipe' rather than a 'blueprint'. The advantages of this kind of genetic description have also been discussed by many biologists (e.g. [Dawkins 82] pp.177–8 & pp.250–264, [Maynard Smith 86] pp.21–23). From an evolutionary point of view, one of the most important features of the developmental approach is that it allows mutations on the genotype to have a wide range of magnitudes of phenotypic effect. For example, mutations affecting the early developmental process can potentially lead to gross macroscopic changes in phenotype, whereas those affecting later stages can have the effect of 'fine-tuning' particular structures. Furthermore, if there is a degree of modularity in the developmental process, then these mutations have some chance of 'making sense' from the point of view of the overall design of the organism (see the quotation from Waddington on p.12). For example, a mutation affecting the development of the spine in a vertebrate could conceivably result in the formation of an extra vertebra.

McMullin has pointed out that von Neumann's genetic architecture also effectively decouples the geometry of the variational space of the reproducers (i.e. the space of the genetic tapes) from the peculiarities of the environment in which they exist (i.e. the space of the phenotype), and in this way allows the possibility of a mutational pathway linking large numbers of viable reproducers [McMullin 92a] (pp.191–193). In addition, recall from Section 3.2.1 that the architecture will accept any tape of the general form $\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D})$, where $\mathbf{A}$ is a general constructing automaton, $\mathbf{B}$ is a general copying automaton, $\mathbf{C}$ is a control automaton, and $\mathbf{D}$ is any other automaton with arbitrary function. Assuming the the description of $\mathbf{D}$ on the tape can be separated from the description of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$,[9] this design guarantees that mutations which affect the part of the tape describing automaton $\mathbf{D}$ will not interfere with the reproductive capacity of the machine. Machines which reproduce by self-inspection would generally not be certain to have this localisation property. This being the case, we would not always be able to say that there was a particular section of the machine which could be disrupted by mutation without interfering with the machine's ability to reproduce.

Now, in the context of the experimental portion of this thesis, and of Tierra-like platforms in general, it is interesting to ask whether the programs in these systems reproduce according to von Neumann's genetic architecture or rather by self-inspection. As the arguments of the previous paragraphs suggest that a marked difference exists in the evolutionary potential of these two methods, it is an important question, but it has not received much discussion in the literature. McMullin argued that these programs are

[9] This is not an inherent property of the architecture *per se*, but von Neumann's analysis of evolvability did assume a 'compositional' structure in the language of the tape descriptions (see Section 3.2.1).

reproducing by self-inspection [McMullin 92a] (p.200). Ibáñez and colleagues appear to agree [Ibáñez *et al.* 95] (p.574).[10]

In contrast, I would like to suggest that they can sensibly be analysed in terms of von Neumann's genetic architecture. Before getting into this topic, it should be noted that the terminology commonly used to describe reproducers in Tierra-like systems is somewhat different to that used for von Neumann's work. Because of the similarity between Tierra-like operating systems and those of standard digital computers, the actions of Tierran reproducers are often referred to as computations rather than constructions, even when a reproducer is in the process of building a new copy of itself. Additionally, the terms 'program' or 'algorithm' are generally used instead of 'machine' or 'automaton'. These may seem trivial issues, but the different terminology can lead to confusion when comparing the two architectures. In the following, also remember that von Neumann's general constructing automaton **A** is the machinery which *interprets* the tape to produce a new machine (phenotype), and the general copying automaton **B** copies the tape uninterpreted.

At first sight it might seem that there is no separate genetic description of the program in a Tierra-like system. The picture is complicated by the fact that the machinery which interprets the program (i.e. automaton **A**) does not reside in the same part of the computer in which the program itself is stored. The state information for this machinery—a program's 'virtual CPU' (i.e. the instruction pointer, stacks, registers, etc.)—is generally represented in an independent area of memory to the program's instructions. Furthermore, the actual 'interpreting machinery' of the virtual CPU is encoded in the global operating system provided by the platform, and is in this sense implicit in the program's environment. Additionally, the control automaton **C**, which controls when the instructions in the program are executed, is also implicit in the part of the operating system which governs mechanisms such as how a program's instruction pointer is updated after the execution of each instruction. All that is left to be explicitly encoded by the program, therefore, is the copying automaton **B**, and potentially any other arbitrary automaton **D**.

[10] The passage to which I am referring is ambiguous. Ibáñez and colleagues talk about "the possibility of using our self-inspection based reproductive scheme as a basis for artificial evolution. This attempt is not new; it has quite successfully been applied in other environments, and the most paradigmatic of them is probably Tierra" [Ibáñez *et al.* 95] (p.574). I take this to imply that they also regard Tierra as an example of a self-inspection based scheme.

Now, the instructions which make up the program exist in an unreactive state in the system's random-access memory. It is only when the control automaton **C** transfers instructions to the interpreting automaton **A** that they become 'active'. Looked at in this way, we can see that it is the *behaviour* of the program (including looping, jumping around the code, etc.) that is the result of automaton **A** interpreting the unreactive genetic description. This behaviour, or computation, is therefore the equivalent to the constructed machine, or phenotype, in von Neumann's design.[11] The string of instructions residing in the random-access memory (which is normally referred to as the program) can now been seen as nothing more than the tape or genetic description of this phenotype. It is perhaps easier to see the distinction if one considers a parallel program, with multiple processes (with different state information) using the same program listing.

I therefore suggest that a self-reproducing program in a Tierra-like system is consistent with von Neumann's architecture. However, as automata **A** and **C** are largely implicit in the environment in which the programs reside (the only explicit representation being the state information in a program's virtual CPU), and are certainly not encoded by the individual programs, we can see that the 'program', in the sense of a string of instructions in the system's random-access memory, corresponds to the tape $\phi(\mathbf{B}+\mathbf{D})$ in von Neumann's scheme.

The situation is complicated not only because the interpretation machinery resides partly implicitly in the environment, and partly in a different area of memory, but also for (at least) two further reasons. First, I am claiming that the string of instructions comprising the 'program' in random-access memory should be viewed as the genetic tape in a von Neumann style self-reproduction architecture. Now, von Neumann pointed out that the process of copying the tape in his automaton was essentially itself a process of self-inspection. In this sense, Tierran programs do reproduce by self-inspection. However, the overall mechanism for reproduction, including the implicit encodings of the interpretation and control automata, fits in with von Neumann's architecture, in which the copying of the tape by self-inspection is an integral feature. The major consequence of this is that programs in Tierra-like systems should, all else being equal, have similar evolutionary potential to von Neumann's self-reproducing

[11] Indeed, for organisms in *any* kind of evolving system, the notion of a phenotype fundamentally involves behaviour, in the form of interaction with the (biotic and abiotic) environment.

automata, because extra instructions can be added to the end of the 'tape' and subject to mutations. As long as the mutations do not affect that part of the tape which encodes the self-reproduction algorithm, they will be inherited without disrupting the capacity of the program to reproduce.

A second source of complication concerns the way in which information is encoded on the tape of a reproducing program in a Tierra-like system. The statement in the previous paragraph about the evolvability of such programs was qualified by the phrase "all else being equal". Now, as already mentioned, von Neumann claimed that it would be better to store the information of how to build the machine as a recipe rather than a blueprint. The description of the behaviour of a computer program by that program's listing (i.e. its tape) falls somewhere in between these two alternatives. It is not a straightforward blueprint in the sense of a one-to-one mapping to the program's behaviour (phenotype), because it involves things such a looping, branching and conditional execution. However, these features are fairly transparent in the program's tape (at least if we restrict ourselves to a serial rather than a parallel program). This being the case, single mutations will usually have a rather small effect on the program's behaviour. Only if we start considering the development of a multi-process parallel program from a single-process 'egg' would mutations have the potential for changing both coarse-level and fine-level details of the programs, thereby allowing for the sorts of benefits, from an evolutionary point of view, discussed earlier (p.204).

Implicit versus Explicit Encoding

The preceding arguments have led us to consider the question of implicit versus explicit encoding of automata. However, rather than the general question that has been the subject of much debate relating to trivial versus non-trivial reproduction, here we are interested in rather more specific questions relating to von Neumann's architecture. Now, as we are interested in the evolution of these self-reproducing machines, and as the inheritable information of each machine (i.e. the part which gets passed on from parent to offspring) is contained on the tape ϕ, I will assume that the tape must be explicitly represented in some fashion, otherwise there would be nothing which could evolve. We can now ask *which parts* of the $[\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D}]$ architecture are explicitly encoded on the tape ϕ, and which are implicit in the environment. Of course, even the behaviour of those parts which are represented on the tape will still to some extent be encoded in the 'laws of physics' of the environment (recall from Section 7.2.1 that we are considering second-level reproduction rather than complete reproduction), but I think the analysis is nevertheless worthwhile.

In the case of von Neumann's design for a self-reproducing automaton, it is clear that all four subcomponents (i.e. $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$) are very explicitly encoded on the tape $\phi(\mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D})$; the environment in which the automaton exists implicitly encodes only very low-level actions in the form of the local transition rules of individual cells.

The analysis of self-reproducing programs in Tierra-like systems above suggests that in these systems, $\mathbf{B}$ and $\mathbf{D}$ are explicitly encoded on the tape $\phi(\mathbf{B}+\mathbf{D})$, but $\mathbf{A}$ and $\mathbf{C}$ are implicitly encoded in the environment (the operating system). Notice that with this design the 'genetic code' which maps the genotype $\phi(\mathbf{B}+\mathbf{D})$ to the phenotype $[\mathbf{B}+\mathbf{D}]$ cannot itself evolve, because the interpretation automaton $\mathbf{A}$ is not encoded on the tape.

It is interesting to ask how the process of biological cell division fits into this picture. To a first approximation, the information contained in the genome can be thought of as the tape which encodes all of the inheritable information of the cell.[12] The interpretation of the DNA involves its transcription to messenger RNA, and the translation by ribosomes of this mRNA into proteins. All of the molecular components involved in this process are ultimately derived from information contained in the genome.[13] The genome therefore contains an explicit encoding of the interpretation machinery $\mathbf{A}$.[14] The process by which the tape (the genome) is copied is much more complicated than in the artificial systems we have considered up to now. An illustration of the important

[12] For eukaryotic cells we would have to include the DNA contained in mitochondria and, in the case of plants, chloroplasts, as well as that contained in the nucleus. There is some evidence that particular features of cells can occasionally be inherited without apparently requiring a change in DNA sequence (e.g. abnormal patterns of cilia on the surface of particular protozoans), but such exceptions are very infrequent [Maynard Smith 86] (pp.24–25).

[13] To be precise, the initial interpretation machinery is derived from information contained in the *parent* cell's genome, in the same way as in von Neumann's self-reproducing automata.

[14] Note however that the laws of physics and chemistry still play a vital role in the construction of this machinery. For example, the genome only encodes information about the linear sequence of amino acids in any protein that it can construct. As the ribosomes build the protein based upon this information, the growing protein folds into a three-dimensional shape due to attractive and repulsive forces between its subsections. Thus the primary (linear) structure of the protein is encoded in the genome, but its secondary and higher structure is determined by the laws of physics (and by the manner in which it is constructed).

processes involved in the replication of a DNA double-helix is shown in Figure 7.2.

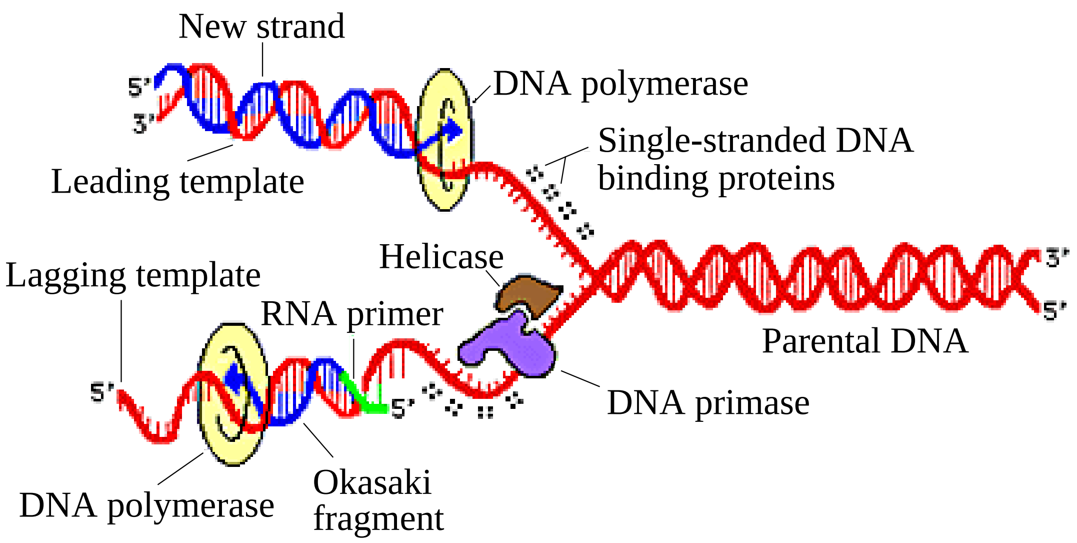


Figure 7.2: DNA Replication.[15]

Many enzymes are involved in the replication process. Replication begins when a topoisomerase enzyme initiates the unwinding of the normally supercoiled DNA. Once this is accomplished, another enzyme, helicase, unwinds the double strands of the helix. DNA polymerase then travels down the single strands of DNA, recruiting free 'raw materials' (deoxy-nucleotide-triphosphates, or dNTP's) to hydrogen bond with their appropriate complementary bases on the single strand, and forming a covalent bond with the previous nucleotide on the newly emerging second strand. The formation of these covalent bonds on the new strand is catalysed by the DNA polymerase, and also by another enzyme, ligase. The DNA polymerase can only build up the new second strand once this strand has been initiated by an RNA primer synthesised by the enzyme DNA primase. DNA polymerase also plays a proofreading function, replacing any incorrectly inserted bases as the new strand grows. A number of other enzymes are also involved in the process, and in maintaining the stability of the original DNA as replication proceeds.

Now, despite the immense complexity of this process, the fundamental principle by which the DNA strands are copied is the complementary base-pairing of the dNTP's. Many of the enzymes are required to get the original double-stranded DNA into a state

[15] Illustration © Genentech, Inc. (`http://www.gene.com/ae/AB/GG/`). Reprinted with permission.

in which it can be replicated, and can therefore be viewed as part of the process that *controls* when replication takes place (i.e. automaton **C** in von Neumann's architecture). The proofreading aspects of DNA polymerase improve the fidelity of the copy, but are not fundamental to the replication procedure *per se* (although these aspects do have important evolutionary implications). Of course, some of the enzymatic functions must be properly viewed as essential to the copying process itself, particularly those enzymes which join the individual segments of the new strand together to form a single continuous molecule. Therefore, some of the copying process proper is governed by enzymes and therefore ultimately encoded upon the DNA itself, but a major element of the process (complementary base-pairing) is determined by the inherent bonding affinities of the molecules, and is therefore implicit in the laws of physics and chemistry.

We might therefore say that the genome of a biological cell corresponds to a tape $\phi(\mathbf{A} + \mathbf{b} + \mathbf{C} + \mathbf{D})$ in von Neumann's architecture, where the lower-case **b** denotes the fact that the fundamental copying process is implicit in the environment, even though extra machinery is involved and explicitly encoded in the genome.

It is interesting to speculate on what information we might desire to be explicitly encoded on a molecule or other structure which would be suitable for acting as a robust initial seed for an open-ended evolutionary process. I will refer to such a structure as 'proto-DNA'. Now, we would like our proto-DNA to be an indefinite hereditary replicator if it is to be such a seed. In other words, it should be able to exist in an unlimited number of configurations which retain the ability to reproduce. If the copying process is encoded on the tape itself, then mutations have the potential to disrupt its ability to be reproduced. It would therefore seem desirable that the copying automaton **B** of our proto-DNA be largely implicitly encoded in the environment. Note that this would not necessarily prevent a more complicated, and possibly more reliable, explicit copying process $\mathbf{B}'$ later evolving from (but still based upon) the simpler implicit process, as indeed seems to have happened during biological evolution.

If the copying procedure for our proto-DNA is implicitly encoded in the environment, however, any configuration of proto-DNA would, all else being equal, be able to reproduce as well as any other. In other words, there would be no basis for preferentially selecting some configurations over others, and therefore no basis for an evolutionary process. Specific configurations of proto-DNA must therefore have some specific properties that are selectively significant. Models of the origin of life commonly presume that these simple phenotypic properties were things such as increased stability of the molecule, simple control of the local environment, catalytic activity, etc. (e.g. [Eigen & Schuster 77], [Cairns-Smith 85], [Szathmáry & Demeter 87]).

At the initial stages of an evolutionary process, however, we would not expect there to be mechanisms for explicitly decoding the proto-DNA; in other words, the interpretation machinery **A** is implicit. This means that particular configurations of proto-DNA should have some specific phenotypic properties (such as the ability to act as catalysts) which can be determined directly from their structure rather than having to be explicitly decoded from the genotype. We could therefore regard the proto-DNA as merely $\phi(\mathbf{D})$, meaning that particular configurations have particular phenotypes associated with them, which are (a) not related to the process of self-reproduction *per se*, and (b) do not require to be decoded by an explicit interpretation automaton **A**. Regarding the kinds of simple phenotypes that we might wish to be available to our proto-DNA, some possibilities are suggested by the origin-of-life models mentioned previously, but in general the options seem endless. Graham Cairns-Smith observes:

> "It is almost too easy to imagine possible uses for phenotype structures—because the specification for an effective phenotype is so sloppy. A phenotype has to make life easier or less dangerous for the genes that (in part) brought it into existence. There are no rules laid down as to how this should be done." [Cairns-Smith 85] (p.106).

If more complicated phenotypes are to arise later on in the evolutionary process, however, we require that the proto-DNA at least has the potential for explicit interpretation machinery $\mathbf{A}'$ and control machinery $\mathbf{C}'$ to become associated with it. This would involve some form of specific reaction to subsections of information in the proto-DNA, but more work is needed to fully identify how this potential for explicit interpretation might be assured.

Ability to perform other tasks

In the previous paragraph it was suggested that proto-DNA in its primitive form should not involve much interpretation or control machinery. However, it is important that some specific phenotypic properties are implicitly associated with specific structures (i.e. these properties are apparent without the need for explicit interpretation machinery). Furthermore, for semantic closure and open-ended evolution, the proto-DNA should also have the potential to be explicitly interpreted (see Section 7.1.2). Without the ability of individual replicators to have other properties as well as self-reproduction, the evolving system will not be very interesting. Indeed Muller, who, in the early part of this century was the first person to explicitly propose an exclusively evolutionary definition of life, emphasised the importance of this material "affecting other materials and, therewith, its own success in genetic survival" [Muller 66] (p.512).

To digress a little, with regard to the issue of how symbolic information arises in evolution (discussed, for example, in [Pattee 95b]), this requirement ensures that the matter-symbol relationship is inherent in the system from the beginning. The material is selected for its phenotypic properties, but it is its genetic information which is passed on to its offspring. In this situation, it is necessary to assume that by inheriting this genotype, the offspring will also share the phenotypic properties. For example, in a simple RNA-world scenario (see Section 2.2), we could imagine that molecules which inherit a particular sequence of bases would adopt a particular three-dimensional structure, which might, say, confer specific catalytic properties (as demonstrated in [Zaug & Cech 86]). We could therefore regard the genetic information (the sequence of bases on the RNA molecule) as a symbolic representation of its phenotypic properties (its catalytic action in this example). However, the question of how explicit interpretation machinery evolves is more complicated (as mentioned in the previous section).

Returning to the main topic of discussion, Barricelli was well aware of the need for reproducers to perform other tasks when he designed his artificial life platform in the early 1950s. He says "It may appear that the properties one would have to assign to a population of self-reproducing elements in order to obtain Darwinian evolution are of a spectacular simplicity. The elements would only have to: (1) Be self-reproducing and (2) Undergo hereditary changes (mutations) in order to permit evolution by a process based on the survival of the fittest" [Barricelli 62] (pp.70–71). He goes on to describe a simple discrete one-dimensional model where each cell is either empty or contains an integer number. The numbers reproduce according to the implicit rules of the system ('trivial reproduction' in the common use of the phrase), and mutations arise under certain circumstances. This simple model therefore fulfils the fundamental requirements for an evolutionary process. However, as Barricelli notes, this model of evolution "clearly shows that something more is needed to understand the formation of organs and properties with a complexity comparable to those of living organisms. No matter how many mutations occur, the numbers ... will never become anything more complex than plain numbers" (*ibid.* p.73). Barricelli therefore concentrated on looking for the 'missing ingredient'.[16] It should be noted that von Neumann, also, was not so much interested in machines which could only self-reproduce, but rather in machines which could perform other tasks as well ([von Neumann 66] p.92; see also [McMullin 92a] pp.174–175).

The preceding arguments are leading us in the direction of requiring a form of proto-DNA which reproduces due to the implicit laws of the environment in which it exists, but which also explicitly specifies some properties which can be selected for or against in an evolutionary process. At this point we might note that artificial evolutionary systems which have just these properties already exist, and indeed their use is widespread—I am of course referring to genetic algorithms (e.g. [Holland 75], [Goldberg 89]), genetic programming (e.g. [Koza 92]) and similar techniques. The difference is that we require a system with the potential for a large degree of *intrinsic* adaptation for modelling biological evolution, rather than a system where the selection of individuals is determined by an externally-defined fitness function (see Chapter 3, Sections 3.1.1, 3.2.1 and 3.2.2). Intrinsic adaptation is introduced when the *domain of interaction* of the individual replicators is within the evolving system itself. This is in contrast to systems with an explicitly defined fitness function, where the replicators do not directly interact with other replicators. Ray recognised this point himself when discussing the design of artificial life platforms:

> "What all of this discussion points to is the importance of imbedding evolving synthetic organisms into a context in which they may interact with other evolving organisms. A counter example is the standard implementations of genetic algorithms in which the evolving entities interact only

[16] His solution was to require that elements could only reproduce in symbiotic association with other elements (see Section 3.2.2). While this may indeed be an important aspect of the 'missing ingredient', it is extremely doubtful that it is the *only* important aspect.

with the fitness function, and never 'see' the other entities in the population. Many interesting behavioral, ecological and evolutionary phenomena can only emerge from interactions among the evolving entities." [Ray 94b] (Section 11.1).

A small but nevertheless very important point should be emphasised here. In considering particular types of interactions and sources of selection, we are no longer considering the nature of generic evolutionary processes *per se*. Rather, we are now starting to think specifically about the particular kinds of evolutionary processes that might be capable of supporting phenomena associated with biological organisms (e.g. specific ecological interactions). This involves consideration not only of the nature of the individual replicators, but also of how they interact with each other, and of the general properties of the environment in which they exist. Such issues are fundamental to the design of artificial life platforms, but have so far received little attention from the Artificial Life community. We will return to this topic in Sections 7.3.2 and 7.3.3.

Similar arguments for proto-DNA with the properties of implicit reproduction and the potential for explicitly-encoded attributes with selective significance have been put forward by McMullin [McMullin 92a] (p.267), who points out the connection with Cairns-Smith's general model for the original of terrestrial life based upon inorganic information carriers (i.e. clay minerals, as mentioned in Section 2.2).

Embeddedness in the Arena of Competition and Richness of Interactions

An essential requirement for an evolutionary process is that some form of selection mechanism exists, so that some variations of the reproducing entities are favoured over others. The selection mechanism therefore introduces a form of competition between the individual reproducers; they become engaged in a struggle for existence. The presence of such a mechanism implies that, in some form, the individuals coexist in an arena of limited capacity, and that they are competing with their neighbours (either globally or locally) for the right to be there.

An evolutionary system must therefore have an arena of competition of some description, although there are few restrictions on the particular form it should take. All that is required is that it introduces the concept of (one or more) *resource(s)*, each of which is: (a) a vital commodity to individuals in the population; (b) of limited availability; and (c) that individuals can compete for (at either a global or local level). At the lowest level, each of these resources can usually be interpreted as energy, space, matter or information (see Section 7.1.4).

An issue that arises when considering different evolutionary systems is the extent to which individuals are embedded in this arena of competition. In von Neumann's cellular automata design, individuals are fully embedded—there is no 'hidden' state information (i.e. information which is not embedded in the cellular space itself). If one believes in materialism, the same can be said of the biosphere. At the other extreme, individuals in a genetic algorithm (GA) have minimal embeddedness—the arena of competition merely contains place holders for the chromosomes, and the restriction is generally on the number of individuals, regardless of their size (although most GAs have constant-size chromosomes anyway). These two extremes, together with intermediate situations arising in Cosmos and Tierra, are depicted in Figure 7.3. Note that individuals in Cosmos are not really embedded in the arena of competition at all; the two-dimensional environment only holds pointers to the cells, in much the same way as in a GA.[17] In Tierra, a program's instructions are embedded in the arena, although each program still has some additional state information.

It should be emphasised that this notion of embeddedness is unrelated to the distinction between implicit and explicit encoding, which concerns the degree to which a process is governed by the environment ($\mathcal{S}$ in Definition 1) as opposed to a specific object situated within that environment ($\mathcal{A}$ in Definition 1). The notion of embeddedness exclusively concerns $\mathcal{A}$ rather than $\mathcal{S}$, and, in particular, how much of $\mathcal{A}$ can be manipulated by other objects in the environment, and in what ways.

Related to the issue of physical embeddedness is that of how restricted is the range of interactions allowed between objects within the arena. In a standard GA, no direct interactions are allowed between chromosomes at all; the continued existence of an individual is decided by the externally-defined selection mechanism. In Cosmos, programs cannot directly interact with their neighbours, but they can exchange messages and energy tokens via the local environment. Although programs in Tierra are embedded

[17] The same applies to similar artificial life platforms with two-dimensional environments, such as Avida [Adami & Brown 94].

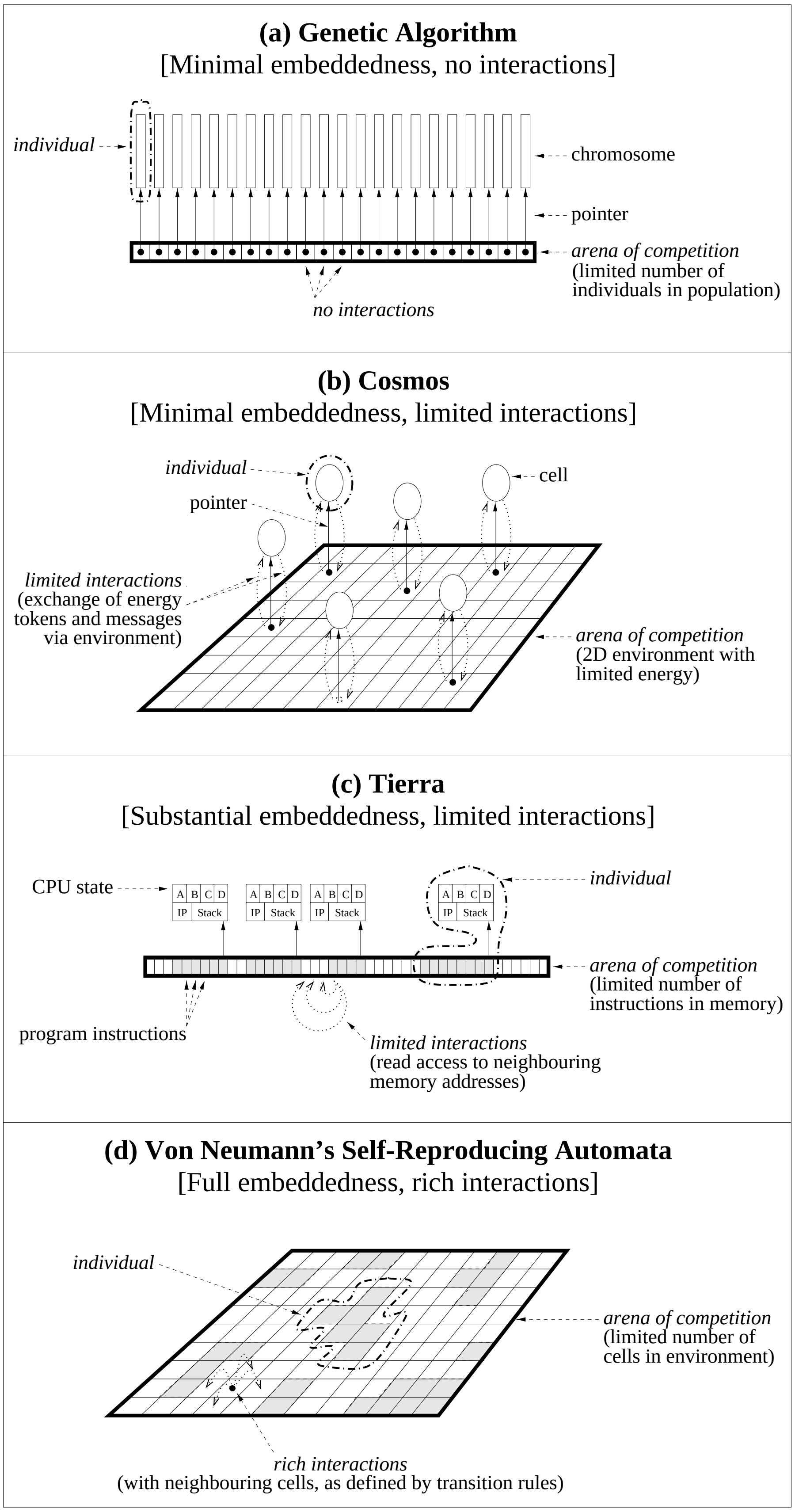

(a) Genetic Algorithm
[Minimal embeddedness, no interactions]
individual
chromosome
pointer
arena of competition
(limited number of individuals in population)
no interactions
(b) Cosmos
[Minimal embeddedness, limited interactions]
individual
cell
pointer
limited interactions
(exchange of energy tokens and messages via environment)
arena of competition
(2D environment with limited energy)
(c) Tierra
[Substantial embeddedness, limited interactions]
CPU state
A B C D
IP Stack
individual
arena of competition
(limited number of instructions in memory)
program instructions
limited interactions
(read access to neighbouring memory addresses)
(d) Von Neumann's Self-Reproducing Automata
[Full embeddedness, rich interactions]
individual
arena of competition
(limited number of cells in environment)
rich interactions
(with neighbouring cells, as defined by transition rules)

Figure 7.3: Embeddedness of Individuals and Richness of Interactions in Various Artificial Evolutionary Platforms.

in the arena of competition to a much greater extent than they are in Cosmos, the range of interactions allowed with neighbouring programs is still fairly restricted; programs can read the code of their neighbours, but they cannot directly write to neighbouring memory addresses.[18] In contrast, von Neumann's cellular automata implementation is far less restrictive; the transition rules of the cellular automata define neighbourhood interactions which occur at the level of individual cells and which therefore do not respect boundaries between individual organisms.

From the point of view of the evolvability of individuals, the more embedded they are, and the less restricted the interactions are, then the more potential there is for the very *structure* of the individual to be modified. Sections of the individual which are not embedded in the arena of competition are 'hard-wired' and likely to remain unchanged unless specific mechanisms are included to allow them to change (and the very fact that specific mechanisms are required suggests that they would still only be able to change in certain restricted ways). Additionally, recall from Section 7.1.2 that Pattee has argued that open-ended evolution fundamentally requires the evolution of new meaning in the system, and this can only be achieved in the context of a semantically closed organisation which is completely embedded within the physical world.

To end this section, I would like to briefly return to Holland's recent work with the Echo model ([Holland 95]: see Section 3.2.2). Echo possesses many of the features that I have just argued are desirable for a model of open-ended evolution. For example: selection in determined intrinsically by interactions between Echo organisms (or to use Holland's terminology, agents), rather than by an externally-defined fitness function; the process by which agents reproduce is implicitly defined in the Echo operating system rather than being explicitly encoded by individual agents; the agents are able to perform a variety of phenotypic behaviours; also, Echo is a material model, in which agents are composed of atomic units of matter and must collect raw materials from the environment before they are able to reproduce (I will say more about this topic in Sections 7.3.2 and 7.3.3). Echo is also designed upon more explicit design considerations than were most earlier artificial life models; the considerations for Echo are based upon a core set of principles which Holland believes are common to all complex adaptive systems. For all these reasons, I believe Echo represents a significant advance. However, the structure

[18] Recall that the allocation of memory for reproduction is performed by the external operating system.

of the individual agents—the notion of what it is to be an agent—is still predefined, and the representation of agents is not fully embedded in the arena of competition. Additionally, the interpretation of agent's chromosomes is handled implicitly by the operating system. Now, the system was designed in this way because it is primarily intended as a general model of complex adaptive systems, rather than a specific model of biological evolution. Indeed, the various successful applications of Echo (mentioned in Section 3.2.2) testify to the value of the particular way in which the organism and environment structure have been predefined; if no higher-level structure were imposed, it would be difficult to model most complex adaptive systems of interest (e.g. ecologies, economies, etc.).

In the context of open-ended evolution, however, the design still has some shortcomings. The fact that the Echo operating system implicitly interprets the agents' chromosomes means that they can never come to encode anything more than the fixed range of actions (e.g. offence, defence, conditional exchange of resources) predefined by the designer. In *Hidden Order*, Holland discusses how new meaning can arise in a system, but acknowledges that Echo is deficient in this respect [Holland 95] (p.138). As Pattee has suggested ([Pattee 95b]: see Section 7.1.2), it is only when an organism's genotype, phenotype, and the interpretation machinery that produces the latter from the former, are all embedded in the arena of competition that fundamentally new symbolic information can arise in the genome (thereby permitting truly open-ended evolution). In the discussion of the desirable properties of proto-DNA in Section 7.2.3, it was suggested that this too would initially be interpreted implicitly. It was, however, stressed that the potential should exist for explicit interpretation machinery to evolve (although how this potential might be assured is an open question).

7.3 Improving the Approach

At the start of this thesis I stated that my grand research interest was in the synthesis of artificial life. However, the lack of a precise definition of life soon forced me to focus on narrower, more well-defined goals. In particular, most of the work reported has been concerned with self-reproduction and evolution. In this section I will summarise what I regard as the most important lessons learned from this work, and talk about how they may help in improving the methodology of the approach. I will then suggest some ways in which the study of open-ended evolution by computer simulation may be beneficially extended. Finally, I will reconsider the original goal of creating artificial life, and suggest how the study of more specific issues (such as open-ended evolution) may help us to achieve a better understanding of the fundamental nature of life.

7.3.1 Beyond Digital Naturalism: The Need for Clear Goals[19]

As I suggested in Section 3.3.1, Tierra and similar systems can be viewed as an exploratory investigation into the potential of synthetic evolutionary techniques. Such exploratory studies are a useful and normal aspect of any experimental science. However, as the reappraisal presented in Section 7.1.1 of this chapter suggests, a more rigorous methodology must be adopted if the approach is to be of any further scientific value. In particular, artificial life systems should be designed to address specific issues; these issues may be big or small, but they must be specific. Associated with this is the need for a coherent theoretical framework upon which the system should be grounded. This framework serves as a proposal for an explanation (at some level) of the phenomenon being addressed,[20] and explicitly states which aspects of the system's design are claimed to be relevant to the phenomenon. The system should, to the greatest extent practicable, model these aspects and nothing else; in other words, it should ideally be a minimal model. In practice there will always be choices to be made when deciding how to model a given object or process (cf. David Marr's work, mentioned in Section 3.1.3), but the theory should claim that such choices will be irrelevant with respect to the particular phenomena being investigated.

With this need for clear goals in mind, and considering the lack of a precise definition of life, in the following section I will concentrate on a particular, and fairly well defined, goal: to create an artificial evolutionary system which exhibits open-ended evolution (as defined in Section 2.5).

[19] The phrase 'Beyond Digital Naturalism' is borrowed from [Fontana *et al.* 94].

[20] Even in the situation where some high-level 'emergent' phenomenon is under investigation, and the model is expressed in terms of low-level entities and interactions, it may still be viewed as a potential explanation in the sense that it could demonstrate that no additional entities or interactions are required to produce the phenomenon. David Chalmers refers to such accounts as 'mystery removing' explanations [Chalmers 96].

7.3.2 A Full Specification for Open-Ended Evolution

Perhaps the most important point to be raised in the discussion of self-reproduction and evolution (Section 7.2) was that these processes operate *within an environment* rather than in isolation. The properties of this environment, and the ways in which evolving entities may interact with it (and with each other), fundamentally influence the evolutionary process.

Reflecting upon the significance of his work on evolution, and in particular on his demonstration of the possibility of machines which could build modified copies of themselves, von Neumann said "It is clear that this is a step in the right direction, but it is also clear that it requires considerable additional analyses and elaborations to become really relevant" [von Neumann 66] (p.131).

It has long been recognised that chief among these additional analyses and elaborations is the incorporation of the evolutionary process into a broader framework that also considers the properties of the environment. Holland has emphasised that the study of adaptation "involves the study of both the adaptive systems and its environment. In general terms, it is a study of how systems can generate procedures enabling them to adjust efficiently to their environments" [Holland 62] (p.299). Moreover, Conrad stresses that "the characterization of the substrate is of such immense importance for the effectiveness of evolution" [Conrad 88] (p.304).

Studies of evolution in vitro, such as Orgel's experiments with evolving RNA sequences using the viral enzyme Qβ replicase [Orgel 79], have also demonstrated the need for a better theoretical understanding of these issues. Maynard Smith explains:

> "More or less independently of the starting point ... the end point is a rather small molecule, some 200 bases long, with a particular sequence and structure that enable it to be replicated particularly rapidly. In this simple and well-defined system, natural selection does not lead to continuing change, still less to anything that could be recognized as an increase in complexity: it leads to a stable and rather simple end point. This raises the following simple, and I think unanswered, question: What features must be present in a system if it is to lead to indefinitely continuing evolutionary change?" [Maynard Smith 88] (p.221).

The question raised by Maynard Smith is exactly the one of interest in this section: What sort of system (in terms of individuals, interactions and environments) will give rise to an open-ended evolutionary process?

Waddington's Paradigm for an Evolutionary Process

A characterisation of a process which might be capable of supporting open-ended evolution was proposed by C.H. Waddington 30 years ago [Waddington 69]. Waddington goes as far as to call this characterisation a new paradigm under which biological evolution should be studied. This paradigm is of particular interest because it provides a general characterisation of the individuals involved, of how they interact, and of the kind of environment in which they reside. To my knowledge, little work has been devoted to exploring Waddington's proposal, probably because of the difficulties in capturing it fully with an analytical model (the traditional approach of theoretical biology). However, it is formulated in a way which makes it particularly amenable to synthetic (artificial life) modelling, and is therefore an ideal starting place for developing a better theoretical understanding of open-ended evolution within an artificial life framework.

Waddington describes a replicator as "a material structure $\mathcal{P}$ with a characteristic $\mathcal{Q}$ such that the presence of $\mathcal{P}$ with $\mathcal{Q}$ produces $\mathcal{Q}$ in a range of materials $\mathcal{P}_i$ under circumstances $\mathcal{E}_j$" (*ibid.* p.115).[21] The overall scenario is summarised as follows:

> "The complete paradigm must therefore include the following items: A genetic system whose items ($\mathcal{Q}$s) are not mere information, but are algorithms or programs which produce phenotypes ($\mathcal{Q}^*$s). There must be a mechanism for producing an indefinite variety of new $\mathcal{Q}'^*$s, some of which must act in a radical way which can be described as 'rewriting the program'. There must also be an indefinite number of environments, and this is assured by the fact that the evolving phenotypes are components of environments for their own or other species. Further, some at least of the species in the evolving biosystem must have means of dispersal, passive

[21] Note that this description closely resembles Löfgren's definition of self-reproduction given in Section 7.2.1, if we substitute $\mathcal{A}$ in those definitions for $\mathcal{P}$ above, d for $\mathcal{Q}$, and $\mathcal{S}$ for $\mathcal{E}_j$. Sticking to Waddington's labels, the difference is that Löfgren's definition results in $\mathcal{P}$ being reproduced, whereas Waddington's results in $\mathcal{Q}$ being reproduced. If we assume that $\mathcal{Q}$ is the only aspect of $\mathcal{P}$ that affects the reproduction process, then the end result is the same.

> or active, which will bring them into contact with the new environments (under these circumstances, other species may have the new environments brought to them). These environments will not only exert selective pressure on the phenotypes, but will also act as items in programs, modifying the epigenetic processes with which the $\mathcal{Q}$s become worked out into [$\mathcal{Q}^*$s]." [Waddington 69] (p.120).[22]

This general characterisation raises a number of important issues. First of all, the requirement that $\mathcal{Q}$s act not only as information but also as algorithms—that they must act as operators as well as operands—locates the relationship between genotype and phenotype at the very heart of the paradigm. (The same requirement was suggested for proto-DNA, in Section 7.2.3.)

This insistence that the replicator be treated as an operator as well as an operand is reminiscent of Langton's suggestion that this is the crucial distinction between trivial and non-trivial reproduction (Section 7.2.2). However, the difference is that Waddington's insistence arises through consideration of how to achieve an open-ended evolutionary process. Moreover, Waddington does not claim that this is the only important factor in this respect. In particular, he points out that the open-ended nature of his model relies on the fulfillment of two conditions: (1) that $\mathcal{E}_j$ is an infinite-numbered set; and (2) that there are sufficient $\mathcal{Q}$s to provide $\mathcal{Q}^*$s suitable for an infinite sub-set of $\mathcal{E}_j$s.

The first condition is satisfied by the fact that $\mathcal{Q}^*$s are components of $\mathcal{E}_j$s. A vital direction for future research is the investigation of the different sorts of ways in which $\mathcal{Q}^*$s could be components of $\mathcal{E}_j$s, and the evolutionary consequences of such choices.

Of the other condition, Waddington says that "the second requirement, that the available genotypes must be capable of producing phenotypes which can exploit the new environments, requires some special provision of a means of creating genetic variation ... It is important to emphasize that the new genetic variation must not only be novel, but must include variations which make possible the exploration of environments which the population previously did not utilize ... It is not sufficient to produce new mutations which merely insert new parameters into existing [programs]; they must actually

[22] In the original paper, the final word of this paragraph appears as $\mathcal{Q}'$s rather than $\mathcal{Q}^*$s. This is fairly clearly a typographical error.

be able to rewrite the [program]" (*ibid.* pp.116–118).

Another important direction for future research is to explore how this second condition can be satisfied. Providing the $\mathcal{Q}$s with access to sufficient processes to ensure (something close to) universal construction will undoubtedly be part of the solution. This does not necessarily mean that each $\mathcal{Q}^*$ has to be a universal constructor, but they should at least have access to a basic set of operations to give the set of all $\mathcal{Q}$s the ability to construct a sufficient set of $\mathcal{Q}^*$s. This task may be related to the ability to perform universal computation, which depends on the combination and conditional iteration of a simple set of operations (e.g. [Gandy 88]), although the spatial aspect of construction is an extra complication.

It is worth mentioning at this point that some of the artificial evolutionary systems described in Chapter 3, such as Barricelli's later studies with evolving game strategies (e.g. [Barricelli 63]), Conrad and Pattee's model [Conrad & Pattee 70], and Holland's α-Universes [Holland 76], do have the notion of emergent operators (phenotypes). However, these phenotypes generally have a limited range of action, thereby preventing the systems from engaging in truly open-ended evolutionary processes.

Returning to Waddington's paradigm, notice that his second condition for open-ended evolution is more subtle than that of universal construction alone. A full analysis of this condition would also involve the question of how phenotypes which are, in some sense, fundamentally new may be introduced into the population to take advantage of new environments. This question is related to Pattee's, of how fundamentally new measuring devices may evolve ([Pattee 88]: see Section 3.1.2).

Now, the requirement in systems capable of open-ended evolution that individual reproducers have selectively significant phenotypic properties, on top of the ability to reproduce, has already been discussed (see Section 7.2.3). However, it may turn out that the fulfillment of Waddington's second condition would require reproducing structures to possess not just one, but *multiple* phenotypic properties, possibly of different functional modalities (e.g. catalysis, light sensitivity, motility, etc.). Maynard Smith has observed that "it seems to be a general feature of evolution that new functions are performed by organs which arise, not *de novo*, but as modifications of pre-existing organs" ([Maynard Smith 86], p.46: see Section 2.3.5). This principle could potentially solve the problem raised by Waddington and Pattee, of how new measuring devices (or fundamentally new phenotypes) arise during evolution: a structure with multiple properties might originally be selected for one of these properties, but it might later turn out (quite accidentally) that some of its other properties also confer (unrelated) adaptive advantages upon the bearer of that structure. In such a scenario, an organism which duplicated this structure might have an adaptive advantage over those possessing a single copy, because each structure could be optimised for a single property. In this way, the organism can acquire fundamentally new phenotypic properties. This perspective may bring some light to bear upon the evolution of fundamental innovations, but it also opens up a whole range of new problems relating to the modelling of multiple, and mostly (initially at least) irrelevant, properties of objects. Such questions require much more investigation, but existing work reported in the biological literature on multifunctional enzymes may be helpful (e.g. [Kacser & Beeby 84]).

I end this section with the observation that Waddington's paradigm for an evolutionary process is very similar to what Bedau has recently referred to as 'supple adaptation' ([Bedau 98b]: see Section 2.1.1). Bedau says that "natural selection produces supple adaptation only when it is continually creative. Adaptation cannot be continually creative without ongoing environmental change. One way to bring about ongoing change is for the evolving system's own evolution to continually reshape the selection criteria ... [which could perhaps be achieved if] each organism's environment consists to a large degree of its interactions with other organisms" (*ibid.* p.127).

Individuals, Interactions and Environments: Observations, Speculations and Open Questions

Waddington's work provides us with a promising framework within which to study open-ended evolution, but, as we have seen, it also raises many questions. In this section I will briefly pull together a number of observations and further open questions relating to the practical implementation of an system capable of supporting open-ended evolution. I have attempted to categorise the following points into those relating to individuals, to interactions and to environments, although this is necessarily imperfect as the categories are all interrelated at some level. As I have mainly concentrated on analysing individual reproducers up to this point, the issues raised in the latter two categories will be of a somewhat more speculative nature.

Individuals. As suggested by the analysis of self-reproduction in Section 7.2.3, individual replicators should have the capacity for indefinite heredity if they are to participate in a process of open-ended evolution. Additionally, it is desirable that the copying process (von Neumann's automaton **B**) is implicit in the environment, at least at the start of the evolutionary process. Moreover, these individuals should have other selectable properties as well. Initially, these properties might be directly associated with the individual's structure (e.g. catalytic capabilities), although the potential should also exist for the evolution of a more indirect translation from genotype to phenotype.

Formulated in this way, the focus shifts from self-reproduction to questions concerning what other kinds of processes could or should be associated with the individuals. In other words, what is their 'phenotype space'? It has often been noted that open-ended evolution requires that the domain of interaction of the replicators is within the evolving system itself. Furthermore, in Section 7.2.3 it was also argued that individuals should be fully embedded within the arena of competition.

Interactions. Evolution depends upon selection pressure. For the kind of system we are considering, with intrinsic rather than extrinsic adaptation, this selection pressure depends upon the types of *interaction* between components in the population.

In Section 7.2.3 it was argued that the less restricted these interactions are, the more potential the system has for open-ended evolution. In biological evolution, for example, a fundamental aspect of the struggle for existence is that organisms act as potential resources of both matter and energy for other organisms (see Section 7.1.4).

Regarding the particular sorts of interactions required for open-ended evolution, it is possible that these might fall into two broad categories: *synergisms* and *conflicts* (e.g. [Buss 87], [Maynard Smith & Szathmáry 95], [Stewart 97]: see Section 2.3.4). In biological systems, some important general principles governing interactions at the molecular level include: specificity of reaction, control of reaction, and switching mechanisms (e.g. [Maynard Smith 86] p.70). Furthermore, Morán and colleagues have remarked that origin-of-life models typically rely upon four types of biochemical interaction: reaction (the construction of new compounds, or the decomposition of existing ones), diffusion, catalysis, and template replication [Morán *et al.* 97]. Morán *et al.* suggest that the latter two interactions can be considered as special cases of reaction, so reaction and diffusion may be considered as the fundamental interactions. Another basic interaction principle, concerning the use of information in biological system (e.g. in the genetic code, allosteric enzymes, hormones and nervous conduction), is the arbitrary nature of the messenger. Maynard Smith, following Monod, refers to this as the 'gratuity' of the messenger [Maynard Smith 86]. It is an open question whether these principles are truly universal, and therefore also necessary for artificial systems.

Environments. The environment plays a crucial role in evolution, as I have already emphasised. It has often been remarked that the evolution of complex organisms requires a complex, heterogeneous and changing environment (e.g. [Waddington 69], [Pattee 95a], [Bedau 98b]). One way to ensure such an environment is to make the evolving organisms part of the environment experienced by other organisms, as Waddington and others have suggested.

The notion of a (spatially) heterogeneous environment, and indeed the very notion of individuality, requires that the environment has some spatial structure. Spatial structure not only introduces the notion of individuality (in the sense that a particular component in the system can be distinguished at any point in time by virtue of its particular relationship with other components), but also makes possible concepts such as compartmentation and the control of the local environment. Such concepts have important consequences for the evolution of cooperative organisations (e.g. [Maynard Smith & Szathmáry 95]), as indeed has already been demonstrated in a number of artificial evolutionary systems (for example, [Boerlijst & Hogeweg 91], [van Baalen & Rand 98]). These studies are a good start, but a great deal of further research is required to improve our knowledge of how the spatial structure of the environment affects the evolutionary behaviour of the system.

With spatial structure comes the requirement for components to move (either actively or passively), so that they may experience different environments. Active motility opens up a wide range of phenotypic and ecological possibilities, but it may be that passive diffusion is sufficient for open-ended evolution in general.

Another fundamental aspect of the environment is related to the choice between a purely logical model (where reproducing entities are configurations of states) or a material model (where the entities are related to material structures, composed of atomic units of matter, and with energetic considerations of one form or another). These choices are probably best viewed as opposite ends of a spectrum, with a host of other possibilities in between. Some of the issues concerning the consequences of this choice on the evolution of the system were discussed in Section 7.1.4. Whether a purely logical model is sufficient to capture an open-ended evolutionary process remains to be seen. However this issue is resolved, it is likely that the development of a theory of constructive dynamic systems (i.e. dynamic systems where new operators may appear intrinsically over time) will be a central requirement. The work described in Section 3.2.3 represents a useful step in this direction.

Regardless of the degree to which states are grounded in a material environment, the discussion of the advantages of genetic reproduction over self-inspection (Section 7.2.3) suggests that a fairly fundamental distinction may exist between reactive states and (quasi-)quiescent states. Whether this distinction is absolutely necessary for open-ended evolution is unknown, but, as Waddington remarked "in practice—and perhaps because of a profound law of action-reaction—it is difficult (impossible?) to find a [molecule] which is stable enough to be an efficient store and at the same time reactive enough to be an efficient operator" [Waddington 69] (p.115).

With respect to the types of bonding between atoms in the more materialistic models, it has generally been found that at least two types of bond are required: one strong and fairly permanent, and another weak and temporary. Examples include Penrose's analysis of self-replication ([Penrose 62]: see Section 3.2.1), Myhill's model ([Myhill 64]: see Section 3.2.1), and an unpublished model of my own, named Nidus. Holland's α-Universes had only one type of bond [Holland 76],[23] but McMullin's implementation of this model revealed that its evolutionary behaviour was much more restricted than Holland had anticipated ([McMullin 92a], [McMullin 92b]). It is possible that this is related to the apparent requirement for two kinds of matter; in particular, a mechanism for temporary association might be necessary to allow the two kinds to interact. At this stage, however, this is just a speculation.

[23] Holland refers to strong and weak bonds in his model, but the weak bond effectively denotes the absence of a bond in the normal sense of the word.

Finally, even though it has been argued that the genetic material should be quasi-quiescent, the fact remains that the DNA of biological organisms is actually rather active; mobile segments of DNA—transposons—play an important role in gene regulation. Furthermore, it has been suggested that features such as splicing and mobile genetic elements were present even at the prebiotic stage of evolution (e.g. [Reanney 79], [Buss 87] p.194). Taking a broader view of mobile genetic elements, the process of symbiogenesis, described in Section 2.1.1, can also be included. Barricelli's remarkable results can therefore be seen as an example of the potential of such genetic mobility. It is also relevant that Ray's latest work with Tierra has included additional system-defined operations such as insertion, deletion and crossover ([Ray & Hart 98]: see Section 3.2.1). We can therefore say that in nature, individual DNA segments (genes) seem to have retained at least some kind of individuality even when collected together on a chromosome, and that in artificial systems, mobile genetic elements sometimes seem to enhance the system's evolvability. It is easy to think of reasons why this may be the case. Whether any of this has any bearing on the necessary features of a genetic system able to support open-ended evolution is an open question.

7.3.3 Evolution and Life Revisited

To end this chapter I would like to suggest ways in which the analysis of open-ended evolution, as discussed here, can usefully contribute to the broader investigation of the nature of life.

Evolution can explain how self-reproducers come to be adapted to their environment, but they may not necessarily ever be capable of evolving into anything we might regard as living (e.g. autopoietic organisations) because of the way in which their environment is constructed. We therefore need a theory of the necessary and sufficient requirements for the environment to support living organisations, as well as to support evolution.

Recall from Chapter 3 that Bedau has suggested that life should actually be defined in relation to a system that exhibits open-ended evolution ('supple adaptation' in his words) [Bedau 98b]. In this view, *the evolving system as a whole* (including individuals, interactions and environments) is the primary form of life, and particular components within the system (e.g. individual organisms) qualify as (secondary forms of) life by virtue of their specific relationship with the system. This is a heretical view even among those who define life in evolutionary terms (see Section 2.1.1), as even they usually offer definitions in terms of individual organism, cells, or genes, and not in terms of (ultimately) the biosphere as a whole.

However, Bedau's equation of life with open-ended evolution raises some intriguing issues. Chief among these is the question: To what extent does open-ended evolution imply life (in the common sense of the word, embracing phenomena such as metabolism, autopoiesis, food webs, evolutionary arms races, hierarchical evolution, etc.)? In other words, would it be possible to create a system with the capacity for open-ended evolution which did *not* exhibit these phenomena? If so, what would open-ended evolution look like without them?

As a specific example, it seems likely that competition for matter and energy would be essential to provide selection pressure for self-maintenance and autopoiesis. However, a purely logical implementation of a suitable evolutionary process *might* still be sufficient to bring about open-ended evolution. Comparison of the behaviour of logical versus material models might therefore lead to a better understanding of the relationship between the evolutionary and ecological aspects of life.

Regarding the fundamental nature of life, and specifically regarding its associated ecological (as opposed to evolutionary) phenomena, it is common to hear questions such as: Is metabolism a necessary component for a definition of life? (e.g. [Boden 99]). Using this question of metabolism as a specific example, I think it is more useful, given that life is not a well-defined concept, but that metabolism seems to be a ubiquitous feature of terrestrial life, and that there is widespread agreement that complex metabolisms have arisen from simpler origins by a process of evolution, to ask questions such as: How specific a form of evolution (in terms of individuals, interactions and environments) is required such that metabolism is able to (or necessarily) arise(s)? Conversely, is a form of metabolism required if a system is to have the capacity for open-ended evolution? Such questions can obviously also be applied to other features of living systems, such as autopoiesis, food webs, etc. The advantage of questions such as these is that they are not expressed in terms of the imprecise concept of life.

By modelling processes such as evolution and metabolism in artificial life systems and observing the resulting behaviour, we can see how closely this behaviour corresponds to our common conceptions concerning living systems. In this way, artificial life models can give us a better understanding of the relevance and interdependencies of such processes, and, in so doing, lead us towards a better understanding of the essential properties of life.

Chapter 8

Summary

"Fly me to the moon
and let me play among the stars
Let me see what spring is like
on Jupiter and Mars ..."

Bart Howard, 'Fly Me to the Moon'

I began this thesis by talking about life, and the possibility of recreating it on a computer. However, the lack of a precise and satisfactory definition of life led me to concentrate on more specific issues. In particular, most of this thesis has been concerned with open-ended evolution.

The approach to modelling open-ended evolution pioneered by Tom Ray with the Tierra platform has been fairly widely used, and its validity fairly widely accepted, within the artificial life community. However, a number of concerns have been voiced about Tierra; for example, the influence that specific yet fairly arbitrary design features exert upon the system's behaviour was mentioned in Sections 3.2.1 and 3.3.1. I suggested that by experimenting with a similar—but not identical—system, some light might be shed on such issues. I therefore decided to design, implement and experiment with such a system, called Cosmos (described in Chapter 4).

A wide range of experiments with Cosmos were reported and analysed in Chapters 5 and 6. The behaviour of the system was different to Tierra in various ways. For example, no parasitism or similar ecological phenomena were observed. This result was, in fact, expected, and is due to differences in the kinds of inter-organism interactions allowed in the two platforms.

The role of contingency (chance events), as opposed to general evolutionary principles or to specific design details, in determining the outcome of Cosmos experiments was investigated (Section 6.1). It was found that contingency did play an important role; in a series of 19 experiments run under identical conditions apart from the number used to seed the random number generator, each one performed significantly differently, on a number of measures, to at least a third of the other runs. I suggested that these results should be broadly applicable to similar platforms, although the increased ecological interactions in Tierra compared to Cosmos might change the situation to some extent.

Other results were reported, such as the emergence of 'speciation' in runs where energy was distributed heterogeneously to the environment (Section 6.5.3). However, it was hard to escape the feeling that many results were due to fairly specific features of the system's design. On top of this, the parameter space was far too large to allow a full and systematic study of the platform's capabilities.

With the benefit of the experience gained with Cosmos, I took a step back in Chapter 7 to discuss some problems that I now perceive with the Tierra approach to modelling open-ended evolution. Specific problems include the facts that organisms have a 'hard-wired' structure, and that their interactions with other organisms are very restricted. I also suggested that the fact that organisms are represented by self-reproduction algorithms might limit the system's evolvability, and that the lack of competition for matter and energy might restrict the potential for various ecological phenomena (such as self-maintaining organisations, food webs, etc.) to emerge in the system. Note that we do not know *a priori* whether this last shortcoming deprives the system of the capacity for open-ended evolution (although it may turn out that it does), but it does deprive it of the capacity for modelling many other processes associated with life. All of these shortcomings can ultimately be traced back to the lack of an adequate theoretical grounding to guide the design of such systems.

I then went on to analyse the process of self-reproduction in Tierra, and in a number of other systems, in terms of von Neumann's genetic self-reproduction architecture, $\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D} + \phi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D})$ (Section 7.2). I suggested that the general distinction between implicit and explicit encoding of the self-reproduction process, which has been a preoccupation (in the context of attempts to avoid 'trivial' self-reproduction) of many researchers working with models of self-reproduction recently, is not very enlightening when considering the capacity of the self-reproducers to partake in an open-ended evolutionary process (Section 7.2.3). However, I argued that it *is* enlightening to consider this distinction in the rather more specific context of the components of von Neumann's self-reproduction architecture.

In Section 7.2.3 I demonstrated that programs in Tierra-like platforms can be seen to conform to von Neumann's architecture, despite suggestions to the contrary in the literature. However, the distinction between genetic self-reproduction and reproduction by self-inspection is blurred in these systems, because the interpretation machinery (automaton **A** of von Neumann's architecture) is hard-wired into the operating system rather than being explicitly encoded on the individual self-reproducers. Furthermore, the process by which the program's instructions (the equivalent to the genetic tape in von Neumann's architecture) are copied is itself an example of self-inspection; indeed, von Neumann remarked upon this feature of his architecture himself. I argued that a program's instructions in a Tierra-like platform correspond to the tape $\phi(\mathbf{B}+\mathbf{D})$ in terms of von Neumann's analysis.

After considering the process of DNA replication in von Neumann's terms, I next discussed the desirable properties of 'proto-DNA', i.e. a class of object capable of acting as a seed for an open-ended evolutionary process. I concluded that such an object would correspond to the tape $\phi(\mathbf{D})$ only (Section 7.2.3). Unlike programs in Tierra-like platforms, I argued that the copying process **B** of the proto-DNA should be implicitly encoded in the environment (e.g. the operating system or the laws of physics), at least initially. The fact that **B** is explicitly encoded in Tierran programs means that it is susceptible to disruption by mutations and perturbations from the environment. This is why the interactions between programs in platforms such as this have to be restricted—in Tierra, for example, direct interaction between programs is restricted to the reading by one program of the instructions of its neighbours. If the copying process is implicit in the environment, far fewer restrictions need to be placed upon interactions between components in the system to ensure that the population survives and evolves. With fewer restrictions upon interactions, the system is able to evolve in more diverse ways.

By suggesting that proto-DNA should be of the form $\phi(\mathbf{D})$, I am claiming that a suitable scenario for the start of an open-ended evolutionary process entails a population of strings of arbitrary information, which can all potentially be reproduced via implicit properties of the environment in which they exist. However, in order for there to be any selection between different strings in such a scenario (and therefore any evolution), individual strings must have specific phenotypic properties associated with them. In this way, my arguments suggest a shift of focus away from the process of self-reproduction *per se*, and towards a more careful consideration of phenotypic action. At the beginning of an evolutionary process we would not expect any complicated interpretation machinery to be available to decode the proto-DNA, so the environment should be able to implicitly determine the phenotypic properties (e.g. catalytic activity) of specific strings—in other words, $\mathbf{A}$ is implicit in the environment.

An inspection of the behaviour of existing artificial evolutionary models leaves one with the impression that they are only capable of evolving variations on a limited number of themes—evolutionary innovations usually have a "more of the same" quality rather than being fundamentally novel. Howard Pattee's analysis of the question of how *fundamentally new* symbolic information can arise in a physical system (see Section 7.1.3) suggests that truly open-ended evolution requires that the genotype, phenotypic and interpretation machinery should all be explicitly represented within the (artificial) physical world—this is his condition of *semantic closure* [Pattee 95b]. Therefore, even though I have argued that proto-DNA should initially be interpreted implicitly, it is vital that it also has the potential to evolve explicit interpretation machinery, $\mathbf{A}'$ (together with the necessary control machinery $\mathbf{C}'$). This would require at least that certain processes within the system could become associated with specific sections of information on the proto-DNA strings (representing the birth of the explicit genotype-phenotype distinction). More work is required to further investigate how this potential for the proto-DNA may be assured. (A more explicitly-encoded copying process $\mathbf{B}'$ may also evolve once the proto-DNA is being explicitly interpreted, as indeed seems to have happened during the evolution of life on Earth.) In Section 7.2.3 I also argued that the representation of the organism should be fully embedded in the evolutionary 'arena of competition' in order not to further restrict the evolutionary potential of the system.

Another important issue regarding the evolution of fundamental novelty concerns how organisms can evolve measuring instruments to probe new aspects of their environment (Section 3.1.2). In Section 7.3.2 I suggested that the common evolutionary principle that new functions are generally formed by organs with arise as modifications to pre-existing organs (e.g. [Maynard Smith 86] p.46), together with a move towards modelling components with multiple phenotypic properties (in different modalities), may help here. Much more work is required on this topic, however.

Finally, I returned to the issue of modelling life in general, as opposed to specifically modelling open-ended evolution. In Sections 7.1.4 and 7.3.2 I observed that many of the more interesting ecological and evolutionary phenomena in the biosphere arise because organisms are able to interact in much richer ways than are allowed in most artificial life models. My discussion of proto-DNA included consideration of how these restrictions may be relaxed whilst maintaining the robustness of the self-reproduction process. However, a further issue in the context of modelling life concerns the distinction between logical and material models. Biological organisms are embedded in a material world, and therefore represent useful resources of matter and energy for potential use by other organisms. Without a material grounding (i.e. a system where organisms are composed of structural units which are, at their lowest level, conserved, and which are in limited supply), it is doubtful whether any selection pressure can exist for organisms to evolve properties such as self-maintenance. Also, it is only with such a material grounding that ecological phenomena such as food webs and trophic levels can be realised. If we wish to allow artificial life models the capacity to evolve in these ways, it is therefore likely that we would have to use a material model of some sort.

In Section 7.3.3 I noted that Bedau's picture of life as supple adaptation raises some intriguing issues. For example, it suggests the important ecological processes we commonly associate with life (i.e. precisely the processes that necessitate the use of a material model) might actually be necessary features of any system that is capable of open-ended evolution. Further work on this hypothesis could therefore provide us with a clearer idea of the relationship between the evolutionary and ecological aspects of life discussed in Chapter 2.

In short, I have argued that future progress in the synthetic modelling of open-ended evolution in general, and of the evolution of living systems in particular, requires explicit theoretical consideration not only of individual self-reproducers, but also of the ways in which they interact, and of the environments in which they exist (including consideration of spatial structure, of the way in which organisms form part of the environment experienced by other organisms, and of the degree of implicit or explicit encoding of processes). In the long term, the discipline requires a unifying paradigm—a general picture of individuals, interactions and environments—in terms of which more specific questions can be framed. I have suggested that the paradigm proposed by Waddington in [Waddington 69] represents a useful starting point, and would also provide a valuable connection between artificial life and more traditional theoretical biology. Further development of the theoretical issues identified in Chapter 7 should lead to useful contributions to our understanding of biological life, and to the development of computer-based systems with greatly improved evolutionary potential.

Appendix A

Cosmos System Details

A.1 Global Parameters

An annotated list of all of the parameters available in Cosmos is presented in this section. These are grouped into a number of different categories according to their function. The user may specify non-default values for these parameters in the Cosmos input file `params.ini`, described in Section A.5.

A.1.1 Inoculation

`ancestor` (*type*: enumerated, *range*: {`a1`,`a2`,`user_defined`})
: Specifies the ancestor(s) programs to be used for inoculation. There are two pre-defined ancestors (`a1` and `a2`, listed in Section A.3). If `user_defined` is specified, the ancestor(s) are read from the file `ancestor.ini`. The format of this file is described in Section A.5.

`number` (*type*: non-negative integer, *range*: 1–10000)
: The number of individual programs to inoculate the system with at the start of the run. If more than one type of ancestor is specified in the `ancestor.ini` file, these are introduced alternately until a total of `number` individuals is reached. If the parameter `placement` is set to `even`, then the actual number of inoculated individuals may be slightly smaller than that specified by `number` (see description of `placement` for details).

`placement` (*type*: enumerated, *range*: {`even`,`random`})
: Determines the placement of the inoculated ancestors. For `even` placement, the ancestors are placed evenly on the grid in a square pattern, where the sides of the square are as close as possible to the square root of the number specified by the parameter `number`. If `number` is not a square number, the actual number of individuals will therefore be slightly less than specified. For `random` placement, individuals are placed completely randomly, and no check is made to see whether the chosen position is already occupied.

A.1.2 Start of Run

`rng_seed` (*type*: integer, *range*: any)
: Used to seed the pseudo-random number generator at the start of the run. If `rng_seed` is negative, then an arbitrary seed is chosen (based upon the current clock time).

`comment` (*type*: character string, *range*: any)
: An optional description of the run, which will appear in the `run.log` output file.

`restart` (*type*: boolean, *range*: {`yes`,`no`})
: A value of `yes` will cause an interrupted run recorded in the file specified by the parameter `restart_file` to be restarted.

`restart_file` (*type*: character string, *range*: any)
: The name of the file to be used to restart an interrupted run (see `restart`).

`run_neutral_model` (*type*: boolean, *range*: {`yes`,`no`})
: If set to `yes`, a neutral model is run based upon data recorded in the input file `neutral.dat`. This file is generated during a previous run in which the parameter `record_neutral_model_data` is set to `yes`. For an explanation of neutral models, see Section 5.1.4.

A.1.3 Termination

`limited_run` (*type*: boolean, *range*: {`yes`,`no`})
: If `yes`, run will stop after the number of time slices specified by the parameter `number_of_timeslices`. Otherwise, the run will continue indefinitely.

`number_of_timeslices` (*type*: non-negative integer, *range*: any)
: See `limited_run`.

A.1.4 Environment

`grid_size` (*type*: positive integer, *range*: any)
: Specifies the number of squares along each direction of the grid.

`horizontal_wrap` (*type*: boolean, *range*: {`yes`,`no`})
: Specifies whether the grid wraps around in the horizontal direction.

`vertical_wrap` (*type*: boolean, *range*: {`yes`,`no`})
: Specifies whether the grid wraps around in the vertical direction.

`max_cells_per_process` (*type*: non-negative integer, *range*: any)
: Specifies an absolute population ceiling for the number of cells in the environment.

`population_cutback_on_overcrowding` (*type*: real number, *range*: any)
: If the number of cells in the environment exceeds `max_cells_per_process`, then a proportion of the population, specified by `population_cutback_on_overcrowding`, is killed off. Cells to be killed are chosen stochastically, but based upon the number of energy tokens they have stored.

`overcrowding_check_period` (*type*: positive integer, *range*: any)
Specifies the period (expressed as a number of time slice sweeps) between successive checks for population overcrowding.

`number_of_energy_tokens_per_grid_pos_per_sweep`
(*type*: non-negative integer, *range*: any)
The average number of energy tokens distributed to each grid position at the beginning of each time slice sweep. The number of tokens distributed to individual squares may vary, as determined by the parameter `energy_distribution_scheme`. This parameter also determines the number of energy tokens taken away from each grid position at the end of each time slice sweep. See Section 4.7.

`max_energy_tokens_per_grid_pos` (*type*: non-negative integer, *range*: any)
The maximum number of free energy tokens that any square in the environment can store. If additional tokens are deposited on a square which already contains the maximum number allowed, the extra tokens are lost.

`env_info_broadcast_period` (*type*: non-negative integer, *range*: any)
Specifies the period (expressed as a number of time slice sweeps) between broadcasts of environmental information. See Section 4.5.6.

`envinfostring_decay_constant` (*type*: real number, *range*: any)
Governs the decay rate of messages in the environment. See Section 4.5.5.

`envinfostring_decay_power` (*type*: real number, *range*: any)
Governs the decay rate of messages in the environment. See Section 4.5.5.

`envinfostring_lower_threshold` (*type*: real number, *range*: any)
Specifies a threshold intensity for messages in the environment, below which they are deleted. See Section 4.5.5.

`envinfostring_initial_intensity` (*type*: real number, *range*: any)
Specifies the intensity assigned to newly created environmental messages. See Section 4.5.5.

`max_time_for_msg_send_reinforcement` (*type*: non-negative integer, *range*: any)
Specifies the maximum time interval (in number of time slices) in which a cell can reinforce the intensity of a message it has previously sent using the `cwm_send` instruction. See Section 4.5.5.

`max_time_for_msg_receive_reinforcement` (*type*: non-negative integer, *range*: any)
Specifies the maximum time interval (in number of time slices) in which a cell can extend the search area of a previously issued `rms_receive` instruction. See Section 4.3.7.

`rms_receive_search_area` (*type*: non-negative integer, *range*: any)
Specifies the number of squares searched for environmental messages upon each execution of the `rms_receive` instruction. See Section 4.3.7.

`energy_collection_scheme` (*type*: enumerated, *range*: {`private`,`shared`})
Specifies the rules governing the collection of energy tokens by a cell from the environment and from neighbouring cells. See Section 4.5.3.

`energy_distribution_scheme` (*type*: enumerated, *range*: {`land,sea,mixed,random`})
Specifies how energy tokens are distributed across the environment by the Cosmos operating system at the beginning of each time slice sweep. See Section 4.5.2.

`energy_distribution_random_chunk_size` (*type*: non-negative integer, *range*: any)
Specifies how many energy tokens are distributed to each randomly chosen square when `energy_distribution_scheme` is set to `random`. See Section 4.5.2.

`x_delta` (*type*: real number, *range*: any)
Specifies the energy gradient when `energy_distribution_scheme` is set to `land` (or `mixed`). See Section 4.5.2.

`wave_width` (*type*: positive integer, *range*: any)
Specifies the width of energy wave columns (expressed in number of squares) when `energy_distribution_scheme` is set to `sea` (or `mixed`). See Section 4.5.2.

`number_of_waves` (*type*: positive integer, *range*: any)
Specifies the number of energy waves, each of width `wave_width`, are fitted across the grid when `energy_distribution_scheme` is set to `sea` (or `mixed`). See Section 4.5.2.

`land_fraction` (*type*: real number, *range*: 0.0–1.0)
Determines the proportion of the environment to be treated as `land` when the parameter `energy_distribution_scheme` is set to `mixed`. An integer number of rows to be treated as `land` is calculated by rounding down the product of `land_fraction` and `grid_size`. These `land` rows are always at the top of the grid, and the `sea` rows at the bottom.

A.1.5 Organism

`max_cells_per_organism` (*type*: non-negative integer, *range*: any)
Specifies the maximum number of cells in a multicellular organism.

`movement_leverage_factor` (*type*: non-negative real number, *range*: any)
Partially specifies how a multicellular organism moves as a result of its constituent cells trying to move. See Section 4.5.4.

`apply_friction_factor` (*type*: boolean, *range*: {`yes,no`})
Determines how organisms move when two or more cells occupy the same square in the environment. See Section 4.5.4.

`multicellularity_penalty_factor` (*type*: real number, *range*: any)
Specifies a cost for multicellularity, in the form of a number of energy tokens removed from each cell in a multicellular organism at each time slice, depending on how many other cells it neighbours within the organism. See Section 4.3.9.

A.1.6 Cell

`ets_lower_threshold` (*type*: non-negative integer, *range*: any)
Specifies a threshold number of energy tokens in a cell's Energy Token Store, below which the cell dies.

`ets_leak_rate_per_timeslice` (*type*: non-negative integer, *range*: any)
: Specifies the number of energy tokens removed from each cell's Energy Token Store at each time slice, on top of those removed for executing instructions. See Section 4.3.5.

`et_value_constant` (*type*: real number, *range*: any)
: Partially determines the number of instructions a given cell is allowed to execute at each time slice. See Section 4.2.3.

`et_value_power` (*type*: real number, *range*: any)
: Partially determines the number of instructions a given cell is allowed to execute at each time slice. See Section 4.2.3.

`default_ets_level_of_ancestor` (*type*: non-negative integer, *range*: any)
: Specifies the default number of energy tokens given to each inoculated ancestor program at the start of the run. This default can be overridden if a different number is specified in the `ancestor.ini` file for a user-defined ancestor.

`number_of_energy_tokens_per_collect` (*type*: non-negative integer, *range*: any)
: Specifies the number of energy tokens that a cell will attempt to collect from the environment for each execution of the `et_collect` instruction. The actual number of energy tokens collected depends upon availability. See Section 4.5.3.

`max_energy_tokens_per_cell` (*type*: non-negative integer, *range*: any)
: Specifies the maximum number of energy tokens that a cell can store in its Energy Token Store.

`info_string_size_limit` (*type*: positive integer, *range*: any)
: Specifies the maximum length of any InfoString object in the system. This imposes an upper limit on the size of genomes, environmental messages, etc.

`stack_size_limit` (*type*: non-negative integer, *range*: any)
: Specifies the capacity (maximum number of items) of the cells' stacks.

`rms_size_limit` (*type*: non-negative integer, *range*: any)
: Specifies the capacity (maximum number of messages) of the cells' Received Message Stores.

`neighbouring_genomes_readable` (*type*: boolean, *range*: {`yes,no`})
: Specifies whether the genomes of neighbouring cells are imported as messages into the Received Message Store and checked for binding sites when a newly active promoter is searching for a binding site. See Section 4.3.7.

A.1.7 Mutations and Flaws

`apply_mutations` (*type*: boolean, *range*: {`yes,no`})
: Specifies whether mutations are to be operative during the run. If set to `no`, then the associated parameters `mutation_period` and `mutation_application_period` have no effect.

`mutation_period` (*type*: non-negative integer, *range*: any)
: Specifies the expected number of bits within the cells of all the organisms in the

population that will be unaffected by mutations between successive bits which *are* affected, at each application of the mutation procedure.

`mutation_application_period` (*type*: non-negative integer, *range*: any)
Specifies the period (expressed as a number of time slice sweeps) between successive applications of the mutation procedure.

`apply_flaws` (*type*: boolean, *range*: {`yes`,`no`})
Specifies whether the flawed execution of instructions is to be operative during the run. If set to `no`, then the associated parameters `default_flaw_period` and `flaw_period_max_change_per_thou` have no effect.

`default_flaw_period` (*type*: non-negative integer, *range*: any)
Specifies the default flaw period initially associated with inoculated ancestor programs. This is the expected number of successful executions of instructions by the Cosmos operating system between successive flawed executions. This default can be overridden if a different number is specified in the `ancestor.ini` file for a user-defined ancestor.

`flaw_period_max_change_per_thou` (*type*: non-negative integer, *range*: any)
Specifies the degree to which a cell's flaw period may be changed by a single mutation. Expressed in parts per thousand. The flaw rate may be mutated to any number in the range of its current value plus or minus the specified fraction of that value.

A.1.8 Input and Output

`species_count_export_period` (*type*: non-negative integer, *range*: any)
Specifies the period (expressed as a number of time slice sweeps) between successive outputs to the file `concentrations.dat`.

`species_count_threshold_for_recording` (*type*: non-negative integer, *range*: any)
Specifies the minimum number of individuals of a given species (genotype) that must coexist in the population before information about that species is written to the file `species_current.dat`.

`max_output_file_size` (*type*: non-negative integer, *range*: any)
Specifies the maximum size (in number of bytes) of output files. When an output file exceeds this threshold it is closed and compressed, and a new file (with a different extension) is opened for writing. See Section A.5.

`morgue_record_period` (*type*: non-negative integer, *range*: any)
Specifies the expected number of deaths of eligible organisms between successive recordings of information about the death of an eligible organism (i.e. an organism of a genotype that has already been recorded in the file `species_current.dat`) into the file `morgue.dat`.

`backup_period` (*type*: non-negative integer, *range*: any)
Specifies the period (expressed as a number of time slice sweeps) between successive backups of the run being written to the file `autosave.ser`.

`record_neutral_model_data` (*type*: boolean, *range*: {`yes`,`no`})
: Specifies whether data for the run is to be written to the file `neutral.dat` for subsequent playback as a neutral model. See also `run_neutral_model`.

`neutral_model_data_export_period` (*type*: non-negative integer, *range*: any)
: Specifies the period (expressed as a number of time slice sweeps) between successive recordings of information to the file `neutral.dat`. Only relevant if the parameter `record_neutral_model_data` is set to `yes`.

`group_zero_length_genotypes` (*type*: boolean, *range*: {`yes`,`no`})
: Specifies whether all organisms of zero length are to be regarded as belonging to the same genotype (i.e. 0AAAA) for the purposes of data collection and analysis.

`visualisation_recording_on` (*type*: boolean, *range*: {`yes`,`no`})
: Specifies whether visualisation output files ('movie' files) are to be recorded for the run.

`visualisation_record_energy_only` (*type*: boolean, *range*: {`yes`,`no`})
: Specifies whether only the energy-related visualisation files will be recorded, or whether they all will be. This is only relevant if `visualisation_recording_on` is set to `yes`.

`visualisation_intersample_period` (*type*: non-negative integer, *range*: any)
: Specifies the period (expressed as a number of time slice sweeps) between the beginning of recording of successive samples of the visualisation data. This is only relevant if the parameter `visualisation_recording_on` is set to `yes`.

`visualisation_intrasample_period` (*type*: non-negative integer, *range*: any)
: Specifies the period (expressed as a number of time slice sweeps) between successive recording of visualisation data within a single sample period. This is only relevant if the parameter `visualisation_recording_on` is set to `yes`.

`visualisation_sample_size` (*type*: non-negative integer, *range*: any)
: Specifies the number of data points (i.e. the number of recorded time slices) for each sample in the visualisation data. This is only relevant if the parameter `visualisation_recording_on` is set to `yes`.

A.2 The REPLiCa Instruction Set

The instruction set of the REPLiCa programming language contains 62 instructions in total, as listed below. The user is able to include and exclude any of these instructions from the available instruction set for any particular run of the system, according to the specification of the genetic code in the input file `genetic_code.ini` (described in Section A.5).

In the following description of the instructions, RMS stands for Received Message Store, CWM for Communications Working Memory, and NWM for Nucleus Working Memory. ADRString refers to the string pointed to by ADRStringPointer (mentioned in Section 4.3.8); this is the InfoString upon which the adr and mov_ic instructions will act. This may be the cell's own genome, or it may be a message in the cell's Received Message Store. ADRStringPointer can be changed to point to a different InfoString with the `str_switch` and similar instructions. A register enclosed in square brackets (e.g. `[ax]`) indicates the contents of the memory location specified by the value of that register.

Some of the instructions relating to regulators (e.g. `reg_create`), and to searching for binding sites (e.g. `adr`), must be immediately followed by a valid binding site specification if they are to operate correctly. A valid specification is a consecutive string of `nop` instructions (i.e. taken from the set {`nop_00`, `nop_01`, `nop_10`, `nop_11`}).

A few instructions involve actions which occur in a particular direction relative to the cell executing them (e.g. `move`, `et_transport`). In these cases, the direction is specified by the low 3 bits of the **cx** register. This gives a number between 0 and 7, which corresponds to the directions shown in Figure 4.3(a).

- Register Manipulation Operations

```
push_a          ; push ax onto stack
push_c          ; push cx onto stack
pop_a           ; pop stack into ax
pop_c           ; pop stack into cx

swap_ab         ; ax=bx, bx=ax
swap_cd         ; cx=dx, dx=cx

mov_ic          ; copy one instruction from ADRString, starting
                ; from address [ax], into cx. The length of the
                ; instruction copied is written to dx. ax += dx.
                ; If ax>length of ADRString, flag=true.

clr_f           ; flag=false

inc_a           ; increment ax (if overflow, flag=true)
inc_c           ; increment cx (if overflow, flag=true)
dec_c           ; decrement cx (if underflow, flag=true)

add_cd          ; cx=cx+dx (if overflow, flag=true)
sub_cd          ; cx=cx-dx (if underflow, flag=true)
sub_ab          ; cx=ax-bx (if underflow, flag=true)

zero_c          ; cx=0
not_c           ; cx=NOT cx (bitwise)
and_cd          ; cx=cx AND dx (bitwise)
or_cd           ; cx=cx OR dx (bitwise)
shl_c           ; shift bits in cx left
                ;  (lo bit <- flag, hi bit -> flag)
shr_c           ; shift bits in cx right
                ;  (hi bit <- flag, lo bit -> flag)
not_lo_c        ; flip low bit of cx
```

- Flow of Control Operations

```
if_fl           ; if (flag=false) increment instruction pointer
                ; otherwise do nothing

if_not_fl       ; if (flag=true) increment instruction pointer
                ; otherwise do nothing

if_z            ; if (cx!=0) increment instruction pointer
                ; otherwise do nothing

stop            ; stop execution and unbind current promoter

set_jmp         ; point the local jump marker to the next
                ; instruction

clr_jmp         ; clear the local jump marker

jmp             ; if local jump marker is set, jump to that
                ; instruction, otherwise do nothing (set flag=true)
```

- Nucleus Working Memory

```
nwm_clear       ; Erase the NWM WritableInfoString

nwm_write       ; Copy first n bits of cx to the end of the NWM,
                ; where n is given by the low 4 bits of dx.

nwm_write_bit   ; Copy the first bit of cx to the end of the NWM.

nwm_divide      ; Create a new single-celled organism by copying
                ; NWM WritableInfoString as the new genome,
                ; splitting the contents of the regulator stores
                ; and Energy Token store, and creating an initially
                ; empty RMS and CWM. The NWM of parent cell is
                ; empty after the division. Child cell is placed
                ; randomly at a free location near the parent (no
                ; preferred direction).

nwm_split       ; Transfer contents of NWM into a new cell which
                ; will become an additional process of the
                ; multicellular organism. Child cell is placed in a
                ; position relative to the parent specified by the
                ; low 3 bits of the cx register. If this location
                ; is already occupied, the nearest free neighbour
                ; is occupied. If all 8 neighbours are occupied,
                ; child cell replaces the parent (parent dies).
```

- (Inter-organism) Communications Working Memory

```
cwm_clear       ; Erase the CWM WritableInfoString

cwm_write       ; Copy first n bits of cx to the end of the CWM,
                ; where n is given by the low 4 bits of dx.

cwm_write_bit   ; Copy first bit of cx to the end of the CWM.

cwm_send        ; Transfer contents of CWM to an
                ; EnvironmentalInfoString at the current grid
                ; position of the cell, with a type given by the
                ; low 4 bits of dx. The msg is given a standard
                ; intensity. This msg replaces any existing msg at
                ; that grid posn with the same msg type. After the
                ; instruction is issued, the CWM is emptied. If
                ; another cwm_send is sent within n time slices of
                ; the last one (and msg type is the same, and CWM
                ; is now empty), the intensity of the existing msg
                ; at that grid pos with msg type=dx is increased
                ; (by a standard amount).
```

- Received Message Store

```
rms_receive     ; Receive msg(s) from environment. One execution of
                ; this instruction will search over a catchment
                ; area of 1/8th of a full circle (45 degrees), in a
                ; direction specified by the low 3 bits of the cx
                ; register. The search is for messages of String
                ; type specified by the low 4 bits of the dx
                ; register. Each search initially spreads out from
                ; the cell and covers a fixed number (n) of grid
                ; cells (specified by the parameter
                ; rms_receive_search_area) and all msgs of the
                ; right type are received and added to the end of
                ; the RMS. If another rms_receive is issued for the
                ; same String type and same direction within a
                ; fixed number of time slices (specified by the
                ; max_time_for_msg_receive_reinforcement
                ; parameter), the current search continues outwards
                ; (covering n more grid positions).
```

- Energy token collection / transfer

```
et_collect      ; if (environmental energy token available)
                ;       pick up n tokens from environment
                ;       (n specified by global parameter
                ;       number_of_energy_tokens_per_collect)
                ; else
                ;       flag=true

et_transport    ; if ((number of tokens in store >=
                ;     ets_lower_threshold)
                ;     &&
                ;     (there is a cell belonging to the same
                ;     organism in the direction indicated by the
                ;     lower 3 bits of cx))
                ;       send n tokens to neighbouring cell in
                ;       direction indicated (n specified by
                ;       the global parameter
                ;       number_of_energy_tokens_per_collect)
                ; else
                ;       flag=true

et_check        ; cx=current level of energy token store
```

- Regulators

```
[the following instructions all work for both promoters and
repressors. To work correctly, they must both be followed by two or
more nop's. The first nop specifies whether a promoter or a
repressor is being referred to (nop_00 and nop_01 indicate a
promoter, and nop_10 and nop_11 a repressor). The second and
subsequent nop's specify the binding pattern of the regulator.]

reg_destroy     ; if ((valid binding site specification follows
                ;     instruction) &&
                ;     (a matching regulator exists in the store))
                ;       remove one of the matching regulators
                ; else
                ;       flag=true

reg_transport   ; if ((valid binding site specification follows
                ;     instruction) &&
                ;     (a matching regulator exists in the store) &&
                ;     (there is a cell belonging to the same
                ;     organism in the direction indicated by the
                ;     lower 3 bits of cx))
                ;       send one matching regulator to neighbour
                ;       cell indicated (removing it from store in
                ;       first cell).
                ; else
                ;       flag=true

reg_create      ; if two or more nop's follow, create a regulator
                ; from the specified bit pattern and place it in
                ; the appropriate regulator store. If the first nop
                ; is a nop_00 or a nop_01, create a promoter with
                ; bit pattern specified by the second and
                ; subsequent nop's, and place it at the bottom of
                ; the list in the Promoter Store. Otherwise (if the
                ; first nop is a nop_10 or a nop_11), create a
                ; repressor with bit pattern specified by the
                ; second and subsequent nop's, and place it in the
                ; Repressor Store, checking for possible binding
                ; sites on the Genome (and other eligible
                ; InfoStrings in the Received Message Store).
```

- NOPs / Binding Site Specification

```
nop_00          ; symbols for specifying binding sites (used in
nop_01          ; creating promoters and repressors, and by
nop_10          ; adr instructions)
nop_11          ;
```

- Searching for Binding Sites

```
adr             ; if ((valid binding site specification follows
                ;     instruction) &&
                ;     (a matching binding site is found on the
                ;     ADRString))
                ;       ax = address of the memory location
                ;            immediately succeeding the
                ;            nearest matching template
                ; else
                ;       flag=true

adrf            ; as adr, but only searches forwards from current
                ; position of read-head on ADRString

adrb            ; as adr, but only searches backwards from current
                ; position of read-head on ADRString

str_switch      ; if (there exists an InfoString with type=low 4
                ;    bits of dx)
                ;       ADRString=first matching String found
                ; else
                ;       flag=true

str_switchf     ; as str_switch, but only searches forwards from
                ; current ADRString

str_switchb     ; as str_switch, but only searches backwards from
                ; current ADRString

str_host        ; ADRString=cell's genome String
```

```
str_latest      ; ADRString=last String in RMS list

str_next        ; ADRString=next String in RMS list. If at end,
                ; loop back to the cell's genome String. If on the
                ; genome String, move to first String in RMS.

str_previous    ; ADRString=Previous String (i.e. just the
                ; reverse of the action of str_next)

str_remove      ; if (there exists an InfoString with type=low 4
                ;     bits of dx in the RMS)
                ;       remove first matching string found
                ; else
                ;       flag=true
```

- Cell Movement Operations

```
move            ; Attempt to move cell (and whole organism) in the
                ; direction specified by the low 3 bits of the cx
                ; register. For multicellular organisms, each cell
                ; that issues a move instruction during a time
                ; slice is actually casting a vote for the desired
                ; direction of movement. The overall effect of this
                ; is given by a formula described elsewhere.

migrate         ; Attempt to move cell relative to other cells in
                ; the organism in a direction specified by the low
                ; 3 bits of the cx register. If direction if full,
                ; nearest free direction is taken. If no free
                ; direction is available, migration has no effect
                ; (flag=true). Note that migration for a single
                ; celled organism has the same effect as a move
                ; instruction.
```

- Killing the current process (cell)

```
kill            ; Kill current cell. Any energy tokens in the
                ; cell's Energy Token Store are added to the
                ; current grid position's store. Note that if cell
                ; was part of a multicellular organism, cell death
                ; may lead to the organism physically breaking up
                ; into two or more distinct organisms.
```

A.3 Predefined Ancestor Programs

A.3.1 Ancestor A1

This self-reproducing program operates by copying itself one instruction at a time into the Nucleus Working Memory. It is assumed that the program starts at memory location zero (as is usually the case); no attempt is made to look for a binding pattern to calculate the start address. Similarly, copying continues until an execution of the instruction `mov_ic` sets the flag, which indicates that the end of the program has been reached. When copying is complete, the `nwm_divide` instruction is issued to produce the offspring. A promoter is provided that will attach itself to the beginning of the program to initiate execution. A new promoter of the same type must be produced by the program itself, to be passed on to its offspring. The program listing is as follows:

```
     1-2  101100111000     Start marked with a specific binding pattern
       3  et_collect       Collect some energy
       4  nwm_clear
       5  zero_c
       6  push_c
       7  pop_a            ax=0 (i.e. points to start of program)
       8  set_jmp
       9  et_collect     |
      10  clr_f          |
      11  mov_ic         | Main loop:
      12  if_not_fl      |   Copy instructions one at a time into
      13  nwm_write      |   the Nucleus Working Memory, and check
      14  if_fl          |   whether end of program has been reached
      15  clr_jmp        |
      16  jmp            |
      17  reg_create       Create a new regulator:
      18  nop_00             The regulator will be a promoter
      19  nop_10         |
      20  nop_11         |   These nop's specify a promoter
      21  nop_00         |   that match the binding pattern
      22  nop_11         |   at the beginning of this program
      23  nop_10         |
      24  nop_00         |
      25  nwm_divide
-------------------------------------------------------------------------
promoter  101100111000     Initial promoter
-------------------------------------------------------------------------
```

A.3.2 Ancestor A2

This self-reproducing program works in a similar fashion to ancestor A1. The difference is that it explicitly searches for its beginning and end positions by looking for appropriate binding sites, rather than assuming that copying should start from memory address zero and continue until execution of the instruction `mov_ic` sets the flag. The program listing is as follows:

```
     1-2  101100111000     Start marked with a specific binding pattern
       3  et_collect
       4  et_collect
       5  nwm_clear
```

```
 6  adrb        |
 7  nop_10      |
 8  nop_11      |  Search for binding pattern
 9  nop_00      |  at start of program
10  nop_11      |
11  nop_10      |
12  nop_00      |
13  push_a
14  pop_c
15  et_collect
16  dec_c       |
17  dec_c       |
18  dec_c       |
19  dec_c       |  We now have to subtract the
20  dec_c       |  length of the binding pattern
21  dec_c       |  (12 bits) to get the address
22  dec_c       |  of the actual start of the
23  dec_c       |  program
24  dec_c       |
25  dec_c       |
26  dec_c       |
27  dec_c       |
28  push_c
29  pop_a
30  swap_ab
31  et_collect
32  adrf        |
33  nop_01      |
34  nop_00      |  Search for binding pattern
35  nop_11      |  at end of program
36  nop_00      |
37  nop_01      |
38  nop_11      |
39  swap_ab
40  set_jmp
41  et_collect
42  et_collect
43  clr_f
44  mov_ic
45  push_c
46  swap_cd
47  push_c
48  swap_ab
49  sub_ab
50  swap_ab
51  pop_c
52  swap_cd
53  pop_c
54  if_not_fl
55  nwm_write
56  if_fl
57  clr_jmp
58  jmp
59  et_collect
60  reg_create
61  nop_00
```

62	nop_10	
63	nop_11	
64	nop_00	
65	nop_11	
66	nop_10	
67	nop_00	
68	nwm_divide	
69	stop	
70-71	010011000111	End marked with a specific binding pattern
promoter	101100111000	

A.4 Running Cosmos

Cosmos is started with the following command:

```
cosmos [OutputDirectory] [InputDirectory]
```

`OutputDirectory` and `InputDirectory` are optional arguments to tell the program where to place output files and where to search for input files. If only one directory is specified, it is taken to be the output directory. The system is configured via the input files `params.ini`, `genetic_code.ini` and `ancestor.ini`, described in Section A.5. If either the input or output directory is unspecified when the program is started, the current working directory will be used by default.

A.5 Format of Input and Output Files

The formats of the various input and output files are described below. With the exception of the automatically generated backup file (`autosave.ser`), all files are in ASCII format.

A.5.1 Input Files

When a Cosmos run commences, the program will search for the three files listed below. Cosmos will look for these files in the current working directory, unless a different input directory is specified as an argument when the program is started. The files `genetic_code.ini` and `params.ini` are always required. The file `ancestor.ini` is only required if the parameter `ancestor` is set to `user_defined`.

The Genetic Code (`genetic_code.ini`)

The mapping between the bit string representation of instructions in a cell's genome and the instructions listed in Section A.2 is defined in the file `genetic_code.ini`. The format of the file is shown in Figure A.1.

Note that the file *must* contain a mapping for each of the 64 six-bit codons (although they do not have to be listed in any particular order). It is permitted for multiple

```
000000 instruction_1
000001 instruction_2
000010 instruction_3
:
111111 instruction_64
```

Figure A.1: Format of the `genetic_code.ini` file.

codons to point to the same instruction; if this is the case, fewer than 64 instructions are therefore available for organisms to use.

Parameter specification (`params.ini`)

Non-default parameter values may be specified in this file. The format is shown in Figure A.2. The allowable section names are: `inoculation`, `startinfo`, `termination`, `environment`, `organism`, `cell`, `mutation` and `io`. These correspond to the groupings of parameters in Section A.1. The parameter names and allowable values are as listed in Section A.1. The file `params.ini` may also contain blank lines, and comments (lines beginning with the `%` character).

```
[section_name_1]
parameter_name_1=value_1
parameter_name_2=value_2
:
parameter_name_N1=value_N1

[section_name_2]
parameter_name_1=value_1
parameter_name_2=value_2
:
parameter_name_N2=value_N2

:
[section_name_N]
parameter_name_1=value_1
parameter_name_2=value_2
:
parameter_name_N3=value_N3
```

Figure A.2: Format of the `params.ini` file.

User-defined ancestor programs (`ancestor.ini`)

If the parameter `ancestor` is set to `user_defined`, Cosmos looks in the `ancestor.ini` file for a description of the ancestor(s) to be used to inoculate the environment at the beginning of the run. The format of this file is shown in Figure A.3.

Any number of different ancestors may be specified, each separated by the line `###`. If multiple ancestors are defined in this file, they are introduced alternately into the

```
ancestor_1_description
###
ancestor_2_description
###
ancestor_N_description
```

Figure A.3: Format of the `ancestor.ini` file.

environment during inoculation, until the specified total number of organisms has been reached (see the description of the parameters `number` and `placement` in Section A.1). Each ancestor description is a consecutive sequence of lines, each of which may be any one of the following:

1. A blank line.
2. A comment (commencing with the `%` character).
3. An explicit bit string to be directly added to the ancestor's genome. Specified by a line consisting of a string composed of the characters `0` and `1`. Useful for specifying binding patterns.
4. An instruction (as listed in Section A.2). This has the effect of writing the bit string corresponding to the instruction (as defined in the file `genetic_code.ini`) to the ancestor's genome.
5. An instruction enclosed in square brackets `[]`. This has the effect of determining a sequence of `nop` instructions (i.e. taken from the set `nop_00`, `nop_01`, `nop_10`, and `nop_11`) corresponding to the bit string representation of the specified instruction. The bit string representation of this string of `nop`s is then written to the ancestor's genome. Useful for writing code that will produce regulators which will bind to a particular sequence of instructions (without requiring the programmer to know the bit string representation of these instructions).
6. A promoter to be added to the ancestor's promoter store. Specified by a line beginning with `p:` followed by a string composed of the characters `0` and `1` representing the promoter's bit string.
7. A repressor to be added to the ancestor's repressor store. Specified in same way as promoter, but with line starting `r:`.
8. An initial energy level for the ancestor. Specified by a line beginning with `e:` followed by a number to represent the desired energy level.
9. An initial flaw period for the ancestor. Specified by a line beginning with `f:` followed by a number to represent the desired flaw period.

A valid ancestor description consists of at least one instruction and one promoter. If no initial energy level or flaw period are specified, the default values defined by the parameters `default_ets_level_of_ancestor` and `default_flaw_period`, respectively, are used. If multiple ancestors are defined, they are distributed alternately across the environment, as described in Section A.1 under the description of the `placement` parameter.

A.5.2 Output Files

As the run proceeds, Cosmos writes data to various output files. These are stored in the current working directory unless a different output directory is specified as an argument when the program is started. As Cosmos runs can last for an indefinitely long time (if no limit is set on the length of the run by the parameters `limited_run` and `number_of_timeslices`), the output files could also potentially grow indefinitely large. In order to keep the files at a manageable size, Cosmos breaks down the files described below (except for `run.log`, `species_current.dat` and `autosave.ser`) in the following way: each filename is given an additional extension, which is initially `.AA` (e.g. `concentrations.dat.AA`). When the size of the output file exceeds a threshold (set by the parameter `max_output_file_size`), Cosmos closes the file, compresses it (using gzip), and opens a new file with an incremented extension name (i.e. the second file will have the extension `.AB`). Writing continues in this new file until that too reaches the threshold size, and the compression procedure is repeated.

General Information About Run (`run.log`)

The file `run.log` contains the following information about the run:

1. Cosmos version number.
2. Run comment (specified by the parameter `comment`).
3. Time run commenced.
4. A listing of the genetic code (as specified in the file `genetic_code.ini`). Includes a list of instructions that have multiple codon mappings, and a list of instructions which have no codon mappings.
5. A listing of the ancestor(s) being used. Includes the bit string representation and corresponding instructions, together with the initial promoters, repressors, energy level and flaw period, and the ancestor's ID number.
6. A full list of system parameters together with their values.
7. The number used to seed the random number generator. If the run has been re-started, the new seed is also listed, together with the time slice at which the run re-commenced.
8. The time at which the run finished, together with the time slice at which it stopped, and a comment to indicate why the run terminated.

All except the final item on the above list are written to the `run.log` file at the beginning of the run. The final item is written when the run terminates.

Species Concentrations (`concentrations.dat`)

At regular intervals determined by the parameter `species_count_export_period`, summary information about the species currently in the population is written to the file concentrations.dat. This file has a two-line header which is required by the actiview program.[1] Subsequent lines of the file are of the format:

```
TimeSliceNumber:  Species1-ID Species1-Number;Species2-ID Species2-Number; ...
;SpeciesN-ID SpeciesN-Number;
```

where SpeciesN-Number is the number of individuals of genotype SpeciesN-ID in the population at time slice TimeSliceNumber.

Species Details (species_current.dat, species_extinct.dat)

Whenever a new organism is born, a check is made to see whether the number of individuals of that genotype currently in the population exceeds a threshold defined by the parameter species_count_threshold_for_recording. If this threshold is exceeded, and if information about the species has not previously been recorded in the file species_current.dat, then a line is appended to the file containing information about the species. The format of the line is:

```
SpeciesID ParentSpeciesID TimeOfFirstOccurrence InitialReadingFrame Genome
```

where ParentSpeciesID is the ID of the species from which the present species is descended; TimeOfFirstOccurrence is the time slice in which the first organism of the species was born; InitialReadingFrame is the frame in which the genome of the first organism of the species is being translated, expressed as a number in the range 0–5 (because codons are 6 bits long) indicating an offset from the beginning of the genome; and Genome is a full listing of the species' genome (written as a bit string).

When a species which has been recorded in species_current.dat becomes extinct, the record of that species is removed from this file, and transferred to species_extinct.dat. When this happens, two extra fields of information are appended to the line describing the species: TimeOfExtinction and FinalReadingFrame (the reading frame in which the last organism of the species was being translated).

Information on Individual Organisms (morgue.dat)

When an individual organism dies, if it is of a species which has already been recorded in the file species_current.dat, then information about the organism is considered for recording in the file morgue.dat. To restrict the size of this file, eligible organisms only have a 1 in N chance of being recorded in it, where N is determined by the parameter morgue_record_period. Each line of the file is of the format:

[1] Actiview is a program developed by Emile Snyder and Mark Bedau at Reed College in the USA, to produce various summary statistics and graphs depicting the evolutionary activity of the run. These are described in Section 5.1.

```
TimeSliceNumber SpeciesID-Numeric SpeciesID-Alpha TimeOfBirth LastReadingFrame
NumberFaithfulOffspring NumberUnfaithfulOfspring TimeOfFirstFaithfulOffspring
TimeOfSecondFaithfulOffspring FlawRateAtBirth MaximumCellsInOrganism
ForeignCodeExecution
```

where `SpeciesID-Numeric` and `SpeciesID-Alpha` are the numeric and alpha components of the `SpeciesID` of the organism (which are split to make subsequent extraction of organism genome length data easier); `NumberFaithfulOffspring` is the number of faithful offspring that the organism gave birth to (and `NumberUnfaithfulOfspring` has a similar meaning relating to unfaithful offspring); `TimeOfFirstFaithfulOffspring` is the time slice at which the organism gave birth to its first faithful offspring, or 0 if it did not achieve this (and `TimeOfSecondFaithfulOffspring` has a similar meaning relating to the second faithful offspring); `MaximumCellsInOrganism` is the maximum number of cells that the organism was composed of at any stage of its life; and `ForeignCodeExecution` is `1` if the organism ever executed any code from its Received Message Store during its lifetime, and `0` otherwise.

Phylogenetic Information (`phylogeny.dat`)

Information about the phylogeny (ancestry) of all species that arise during a run is recorded in the file `phylogeny.dat`. Each line of this file is of the format:

```
SpeciesID,ParentSpeciesID
```

where `ParentSpeciesID` (the immediate ancestor of `SpeciesID`) is set to `0` for the record of a species that was used to inoculate the system at the start of the run. The entire phylogeny of any species can therefore be reconstructed from the data in this file, right back to an ancestor used to inoculate the system at the start of the run; the Perl script `phyl` will print the full phylogenetic tree for a species passed in as an argument.

Neutral Model Data (`neutral.dat`)

If the parameter `record_neutral_model_data` is set to `yes`, Cosmos will record data about the run in the file `neutral.dat`. The period between successive updates to this file is set by the parameter `neutral_model_data_export_period`. The data in `neutral.dat` can subsequently be used to run a neutral shadow of the run (see the description of the parameter `run_neutral_model` in Section A.1). The first two lines of the file are a header: the first line records the size of the environment (as specified by the parameter `grid_size`), the number of organisms with which the system was inoculated at the start of the run (as specified by the parameter `number`), and the maximum number of cells allowed in a multicellular organism (as specified by the parameter `max_cells_per_organism`); and the second line is a separator. Each subsequent line of the file is of the format:

```
TimeSliceNumber NewSpecies OrganismBirths CellSplitInfo CellDeathInfo
OrganismFissionInfo OrganismMovementInfo
```

where `NewSpecies` is the number of new species that have appeared in the population in the period since the previous line of the file was recorded; `OrganismBirths` is the number of new organisms that were born in that period; `CellSplitInfo` shows the number of organisms which grew in size (by executing the instruction `nwm_split`), followed by a list of the size (in terms of number of cells—before the cell division) of each such organism; `CellDeathInfo` shows the number of cells that died, followed by a list of the size (before the cell death) of the organism to which each one belonged; `OrganismFissionInfo` shows the number of organisms which fissioned (due to cell death), followed, for each one, by the number of fragments (new organisms) that resulted from the fission, and a list of the size of each new organism; and the final block of information, `OrganismMovementInfo`, shows the number of organisms which moved, followed, for each one, by a triplet of numbers indicating the size of the organism, and the x and y components of its movement.

Backup File (`autosave.ser`)

Cosmos records its state at regular intervals during a run, so that in the event of a run being stopped prematurely, it can be restarted from the last saved position. The time period between saves is set by the parameter `backup_period`. The data is recorded to the file `autosave.ser`. Should a run need to be restarted from this file, it should be placed in the input directory, and the system started with the parameter `restart` set to `yes`. The format of the saved data is somewhat complicated, but the user should not need to worry about this. Occasionally, however, it may be desirable to extract data from this file, as it contains a complete snapshot of the run at the given time. If this is necessary, the format can be ascertained by studying the `Serialise` method of the class CM_Process, in the source file `Process.cc`.

Visualisation Output Files

If the parameter `visualisation_recording_on` is set to `yes`, various kinds of data are written to files for subsequent playback as 'movies' of the run. Each file contains data about the spatial distribution of a particular aspect of the system, at a number of times during the run.

There are seven different aspects of the system that can be recorded in this way. These are:

1. The ages of the cells in the population, expressed as the number of time slices that have elapsed since their birth (recorded in the file `v_age.dat`).
2. The number of energy tokens that each cell has in its Energy Token Store (recorded in the file `v_cell_energy.dat`).
3. A flag to indicate whether each cell has executed any foreign code from its Received Message Store during its lifetime (recorded in the file `v_comms.dat`).

4. The number of energy tokens stored at each square in the environment (recorded in the file `v_env_energy.dat`).

5. The SpeciesID of each cell, which also indicates the length of each cell's genome (recorded in the file `v_id.dat`).

6. The size of each organism, in terms of number of cells (recorded in the file `v_orgsize.dat`).

7. The direction of movement (if any) of each organism in the previous time slice (recorded in the file `v_move.dat`).

Normally, if `visualisation_recording_on` is set to `yes`, then all of these files get written. However, if `visualisation_record_energy_only` is additionally set to `yes` rather than `no`, then only the files `v_cell_energy.dat` and `v_env_energy.dat` are written.

The files are updated during the run at intervals determined by the system parameters `visualisation_intersample_period`, `visualisation_intrasample_period` and `visualisation_sample_size`. Specifically, data is recorded for a number of sample periods during the run. The number of time slices between successive samples is determined by `visualisation_intersample_period`. Each sample consists of data for a number of time slices, determined by `visualisation_sample_size`. The number of time slices between successive records within a sample is determined by the parameter `visualisation_intrasample_period`.

At each time slice when a record is to be made, a batch of data is written to each file. This data is written in `grid_size`+1 rows of `grid_size`+1 columns. The elements of the final row, and of the final column, are all `-1`. This extra row and column is added purely to easy the process of producing a graphical display from the data using the MATLAB visualisation software package. The remaining elements of the data correspond to individual squares of the environment. For the file `v_env_energy.dat`, each element represents the number of energy tokens available at the corresponding square. For the other files, the element represents data associated with any cell(s) that are present at the corresponding square. If no cells are present, the element is given the value `-1`. If a single cell is present, the element is given the appropriate value (according to which file is being written) for that cell. If multiple cells are present, the element contains the appropriate values for each cell, separated by colons (`:`s).

An extra file, `v_idx.dat`, is also written along with these other visualisation files. At each time slice when data is written to the other files, the corresponding time slice number is written to `v_idx.dat`.

A.6 Implementation Details

The core of the Cosmos system and REPLiCa programming language is implemented as an object-oriented system in ANSI standard C++ (with heavy use of the C++ Standard Template Library). It is compiled with the GNU C++ complier in a Unix (Solaris) environment, but should be portable to other compilers and platforms.

Cosmos uses the bsd_random() pseudo-random number generator (RNG), which uses the linear feedback shift register generation technique. bsd_random() does not suffer from some of the deficiencies of many versions of the standard `random()` RNG.

Appendix B

Details of the Reported Cosmos Runs

B.1 Default Values for Minor Parameters in Cosmos

Table B.1: Default Values for Minor Parameters in Cosmos.

Parameter	*Value*
inoculation	
`placement`	`even`
startinfo	
`run_neutral_model`	`no`
`restart`	`no`
environment	
`population_cutback_on_overcrowding`	`0.10`
`overcrowding_check_period`	`1`
`env_info_broadcast_period`	`8`
`envinfostring_decay_constant`	`0.5`
`envinfostring_decay_power`	`1.0`
`envinfostring_lower_threshold`	`0.01`
`envinfostring_initial_intensity`	`5.0`
`max_time_for_msg_send_reinforcement`	`3`
`max_time_for_msg_receive_reinforcement`	`3`
`rms_receive_search_area`	`8`
`energy_distribution_random_chunk_size`	`10`
`wave_width`	`2`
`number_of_waves`	`2`
`land_fraction`	`1.0`
`reserved_cells_per_grid_pos`	`10`
organism	
`max_cells_per_organism`	`16`
`movement_leverage_factor`	`0.2`
`apply_friction_factor`	`yes`
`multicellularity_penalty_factor`	`1.0`

Table B.1: (continued)

Parameter	*Value*
cell	
`ets_lower_threshold`	1
`ets_leak_rate_per_timeslice`	1
`default_ets_level_of_ancestor`	50
`info_string_size_limit`	5000
`stack_size_limit`	1000
`rms_size_limit`	30
`reserved_bits_per_genome`	400
`reserved_bits_per_binding_pos`	25
mutation	
`flaw_period_max_change_per_thou`	500
io	
`species_count_export_period`	100
`species_count_threshold_for_recording`	5
`max_output_file_size`	1000000
`morgue_record_period`	50
`backup_period`	50000
`neutral_model_data_export_period`	1
`group_zero_length_genotypes`	yes
`visualisation_intersample_period`	5000
`visualisation_intrasample_period`	20
`visualisation_sample_size`	10

B.2 The Standard Ancestor Program, 348AAAA

The standard ancestor program, 348AAAA, used for most of the runs reported herein is very similar to the predefined ancestor A2 described in Section A.3.2. The only differences are that the promoter and associated binding region at the start of the program are 6 bits long rather than 12, and a `move` instruction has been added near the end of the program. The listing of 348AAAA is as follows:

```
101100
et_collect
nwm_clear
adrb
nop_10
nop_11
nop_00
push_a
pop_c
et_collect
dec_c
dec_c
dec_c
dec_c
dec_c
dec_c
```

```
      17   push_c
      18   pop_a
      19   swap_ab
      20   et_collect
      21   adrf
      22   nop_01
      23   nop_00
      24   nop_11
      25   nop_00
      26   nop_01
      27   nop_11
      28   swap_ab
      29   set_jmp
      30   et_collect
      31   et_collect
      32   clr_f
      33   mov_ic
      34   push_c
      35   swap_cd
      36   push_c
      37   swap_ab
      38   sub_ab
      39   swap_ab
      40   pop_c
      41   swap_cd
      42   pop_c
      43   if_not_fl
      44   nwm_write
      45   if_fl
      46   clr_jmp
      47   jmp
      48   et_collect
      49   reg_create
      50   nop_00
      51   nop_10
      52   nop_11
      53   nop_00
      54   nwm_divide
      55   move
      56   stop
   57-58   010011000111
promoter   101100
```

B.3 The Sexual Ancestor Program, 1314AAAA

The sexual ancestor program, used for the runs reported in Section 6.7, is a great deal more complicated than ancestor 348AAAA. The structure of the program is shown schematically is Figure 6.30. The program listing is as follows:

```
1-2  101100111000    Template for start of genome
  3  et_collect      Start of female section
  4  et_collect
  5  cwm_clear
  6  adrb          |
 7  nop_10        |
 8  nop_11        |
 9  nop_00        |
10  nop_11        |
11  nop_10        |
12  nop_00        |
13  et_collect    |
14  push_a        |
15  pop_c         |
16  dec_c         |
17  dec_c         |  Get address of start of genome
18  dec_c         |
19  dec_c         |
20  dec_c         |
21  dec_c         |
22  et_collect    |
23  et_collect    |
24  dec_c         |
25  dec_c         |
26  dec_c         |
27  dec_c         |
28  dec_c         |
29  dec_c         |
30  push_c        |
31  pop_a         |
32  swap_ab       |
33  et_collect
34  adrf          |
35  nop_01        |
36  nop_00        |
37  nop_11        |  Get address of end of genome
38  nop_00        |
39  nop_01        |
40  nop_11        |
41  swap_ab
42  set_jmp
43  et_collect    |
44  et_collect    |
45  et_collect    |
46  clr_f         |
47  mov_ic        |
48  push_c        |  Female copy loop
49  swap_cd       |
50  push_c        |  Copy instructions one at a time
51  swap_ab       |  into the Communications Working Memory
52  sub_ab        |
53  swap_ab       |
54  pop_c         |
55  swap_cd       |
56  pop_c         |
57  if_not_fl     |
58  cwm_write     |
59  if_fl         |
60  clr_jmp       |
61  jmp           |
62  et_collect
```

```
   63  zero_c        | Now broadcast the composed message to
   64  inc_c         | the environment (as an environmental
   65  swap_cd       | message of type number 1)
   66  cwm_send      |
   67  move
   68  stop            End of female section
69-70  111000110010  ____________________________________________
                       Template for start of male section
                     ____________________________________________
   71  et_collect      Start of male section
   72  et_collect
   73  clr_f
   74  zero_c
   75  not_lo_c
   76  shr_c
   77  shr_c
   78  shr_c
   79  shr_c
   80  clr_f
   81  set_jmp
   82  move          |
   83  move          |
   84  et_collect    |
   85  move          |
   86  move          | Move around the grid and collect
   87  et_collect    | energy tokens for a while
   88  dec_c         |
   89  if_fl         |
   90  clr_jmp       |
   91  jmp           |
   92  zero_c
   93  inc_c
   94  push_c
   95  swap_cd
   96  zero_c
   97  clr_f
   98  et_collect
   99  et_collect
  100  et_collect
  101  et_collect
  102  set_jmp
  103  inc_c         |
  104  rms_receive   |
  105  str_switch    |
  106  if_not_fl     |
  107  clr_jmp       |
  108  push_c        |
  109  swap_cd       |
  110  push_c        |
  111  et_collect    |
  112  clr_f         |
  113  zero_c        | Enter a loop to look for environmental
  114  not_lo_c      | messages of type 1.  Continue looking
  115  shr_c         | in all directions until a suitable message
  116  shr_c         | is found.  Also check level of Energy Token
  117  shr_c         | Store while searching, and collect more
  118  shr_c         | energy from the environment if necessary.
  119  shr_c         |
120  swap_cd       |
121  et_check      |
122  sub_cd        |
123  pop_c         |
124  swap_cd       |
125  pop_c         |
126  if_not_fl     |
127  jmp           |
128  et_collect    |
129  et_collect    |
130  et_collect    |
131  et_collect    |
132  et_collect    |
133  jmp
134  et_collect
135  nwm_clear
136  adr           |
137  nop_10        |
138  nop_11        |
139  nop_00        |
140  nop_11        |
141  nop_10        |
142  nop_00        |
143  et_collect    |
144  push_a        |
145  pop_c         |
146  dec_c         |   Look for start of received message
147  dec_c         |
148  dec_c         |
149  dec_c         |
150  dec_c         |
151  dec_c         |
152  dec_c         |
153  et_collect    |
154  dec_c         |
155  dec_c         |
156  dec_c         |
157  dec_c         |
158  dec_c         |
159  push_c
160  pop_a
161  swap_ab
162  et_collect
163  adrf          |
164  nop_01        |
165  nop_00        |
166  nop_11        |   Look for end of received message
167  nop_00        |
168  nop_01        |
169  nop_11        |
170  swap_ab
171  set_jmp
172  et_collect    |
173  et_collect    |
174  clr_f         |
175  mov_ic        |
```

176	push_c	
177	swap_cd	
178	push_c	Male copy loop
179	swap_ab	
180	sub_ab	Copy instructions one at a time
181	swap_ab	from received message into the
182	pop_c	Nucleus Working Memory
183	swap_cd	
184	pop_c	
185	if_not_fl	
186	nwm_write	
187	if_fl	
188	clr_jmp	
189	jmp	
190	et_collect	
191	et_check	
192	shl_c	
193	if_fl	
194	reg_create	
195	nop_00	
196	nop_10	
197	nop_11	Create a new female or male promoter
198	nop_00	
199	nop_11	The choice is based upon whether the
200	nop_10	number of energy tokens stored in the
201	nop_00	Energy Token Store is odd or even
202	if_not_fl	
203	reg_create	
204	nop_00	
205	nop_11	
206	nop_10	
207	nop_00	
208	nop_11	
209	nop_00	
210	nop_10	
211	et_collect	
212	nwm_divide	Reproduce
213	pop_c	
214	swap_cd	Remove received message from
215	str_remove	the Received Message Store
216	move	
217	stop	End of male section
218-219	010011000111	Template for end of genome
female promoter	101100111000	
male promoter	111000110010	

B.4 Genetic Code

The mapping between the binary representation of the genome and the instructions in the REPLiCa programming language is specified in the input file `genetic_code.ini` (see Section A.5.1). In the current implementation of the system, each instruction is represented by a six-bit 'codon'. The mapping used for the runs reported in Chapters 5 and 6 is shown in Table B.4. All 62 of the REPLiCa instructions appear once in this mapping except the `move` instruction, which appears three times (codons `110111`, `011111` and `111111`).

Codon	*Instruction*	*Codon*	*Instruction*	*Codon*	*Instruction*
000000	push_a	011010	if_not_fl	110101	reg_create
100000	push_c	111010	if_z	001101	nop_00
010000	pop_a	000110	stop	101101	nop_01
110000	pop_c	100110	set_jmp	011101	nop_10
001000	swap_ab	010110	clr_jmp	111101	nop_11
101000	swap_cd	110110	jmp	000011	adr
011000	mov_ic	001110	nwm_clear	100011	adrf
111000	clr_f	101110	nwm_write	010011	adrb
000100	inc_a	011110	nwm_write_bit	110011	str_switch
100100	inc_c	111110	nwm_divide	001011	str_switchf
010100	dec_c	000001	nwm_split	101011	str_switchb
110100	add_cd	100001	cwm_clear	011011	str_host
001100	sub_cd	010001	cwm_write	111011	str_latest
101100	sub_ab	110001	cwm_write_bit	000111	str_next
011100	zero_c	001001	cwm_send	100111	str_previous
111100	not_c	101001	rms_receive	010111	str_remove
000010	and_cd	011001	et_collect	110111	move
100010	or_cd	111001	et_transport	001111	migrate
010010	shl_c	000101	et_check	101111	kill
110010	shr_c	100101	reg_destroy	011111	move
001010	not_lo_c	010101	reg_transport	111111	move
101010	if_fl				

Table B.4: Genetic Code for Runs in Chapters 5 and 6.

B.5 RNG Seeds for Runs in Chapter 6

This section lists the values used for the parameter `rng_seed` in each of the individual runs reported in Chapter 6.

B.5.1 The Role of Chance

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	637485923	8	1111333	15	77777765
2	437854826	9	222333211	16	446253478
3	28346478	10	85367222	17	33447752
4	8452633	11	5372	18	7532157
5	76548009	12	37462489	19	222229985
6	13472554	13	644444312		
7	556664432	14	66443381		

B.5.2 Re-Running the Standard Model

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	482754	4	2329	7	234169
2	7384012	5	87928125	8	9987821
3	8926422	6	3312390	9	1526328

B.5.3 Mutations and Flaws

High Mutation and Flaw Rates

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	6376423	4	3454651	7	123654
2	76312311	5	7783342	8	1113
3	2263784	6	22555631	9	7368445

Low Mutation and Flaw Rates

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	761231	4	11178968	7	28972833
2	2327861	5	82543621	8	56732631
3	90071231	6	327887	9	43526713

B.5.4 The CPU-time Distribution Scheme

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	5826384	4	983849	7	99032423
2	87198471	5	3321122	8	4445522
3	12638912	6	367	9	2228367

B.5.5 Energy

High Energy Levels

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	628942	4	24351234	7	71426
2	7236781	5	3425324	8	53763251
3	61739112	6	12328	9	112963

Private Energy Collection

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	8379214	4	224719	7	1980920
2	25179563	5	129942	8	6391234
3	6654723	6	8987234	9	3676199

Small Energy Gradient

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	6357321	4	538216	7	6654710
2	235	5	7864782	8	8254301
3	829781	6	234160	9	2350129

Large Energy Gradient

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	71236	4	43261010	7	5563511
2	879812	5	2278976	8	6868127
3	120231	6	6637129	9	9989015

Random Distribution

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	32418	4	7263210	7	6651900
2	783167	5	100927	8	5562071
3	5214187	6	2733401	9	2217763

B.5.6 Reading Neighbouring Code

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	4243190	4	2289081	7	8871032
2	7621	5	1232162	8	1100234
3	6617818	6	7652190	9	529898

B.5.7 Inoculation with Sexual Ancestors

Run	*Seed*	*Run*	*Seed*	*Run*	*Seed*
1	617812	4	4562701	7	1125301
2	7182190	5	121221	8	77212
3	1115271	6	8829009	9	3342871

Glossary

Definitions of some of the biological terms used in this thesis are presented below. A number in square brackets after a definition indicates that it is based upon (but may be an abbreviated version of) a definition from one of the following sources: [1] the glossary of [Dawkins 82]; [2] the glossary of [Salthe 85]; [3] the glossary of [Margulis & Sagan 86]; [4] the Encyclopaedia Britannica (`http://www.eb.com`); [5] the Merriam-Webster Dictionary (`http://www.m-w.com`); and [6] the web pages of Tom Herbert, Professor of Biology at the University of Miami (`http://fig.cox.miami.edu/Faculty/Tom/bil160/-06_adaptive.html`). Words appearing in **bold** in the definitions are themselves defined elsewhere in the glossary.

abiotic Used to refer to that portion of the world not immediately making up part of the **biomass** of living systems. [2]

adaptive radiation Evolutionary divergence of members of a sincle phyletic line into a series of rather different **niches** or **adaptive zones**. Adaptive radiation is considered to be a rapid process, where adaptation from a recent common ancestor takes place in a short period of time. [6]

adaptive zone A set of ecological **niches** that may be occupied by a group of species that exploits the same resources in the same manner. [6]

allele An alternative form of a genetic locus. [2]

allopatric Living in another region; said of populations of species which occupy ranges in different places on the earth's surface such that gene flow between them would be restricted or absent. [2]

allosteric In enzymology, inhibition or activation of an enzyme by a small regulatory molecule that interacts at a site (allosteric site) other than the active site (at which catalytic activity occurs). [4]

biomass The actual amount of matter included in some living system. [2]

biotic Referring to living systems, as opposed to **abiotic**. [2]

Cambrian Period Earliest time division of the **Paleozoic Era**, extending from about 540 to 505 million years ago. [4]

Cambrian explosion The beginning of the **Cambrian Period**, now thought to date from 540 rather than 570 million years ago, witnessed an unparalleled explosion of life. Many of the major phyla that characterise modern animal life–various researchers recognise between 20 and 35–appear to have evolved at that time, possibly over a period of only a few million years. Many other phyla evolved during this time, the great majority of which became extinct during the following 50 to 100 million years. Ironically, many of the most successful modern phyla (including the chordates, which encompass all vertebrates) are rare elements in Cambrian assemblages; the phyla that contained the most numerically dominant forms were those that became extinct. [4]

coevolution A long-term process of coming into relation with each other of species or populations by active reciprocal modifications of all the members of the coadapting group. [2]

commensalism A relation between individuals of two species in which one species obtains food or other benefits from the other without either harming or benefiting the latter. [4]

deme A local group of organisms in a species that might mate with each other in any given generation. [2]

DNA Deoxyribonucleic acid. A linear, unbranched **nucleotide** polymer, containing deoxy-ribose sugars. In biological cells, DNA codes genetic information for the transmission of inherited traits.

dNTP Deoxynucleotide-triphosphate. The units from which **DNA** molecules are constructed, each carrying a single nitrogenous base (adenine, guanine, cytosine or thymine).

epistasis In genetics, the condition whereby the proximity of a gene to others in the **genome** affects its activity on the **phenotype**. [2]

eukaryote One of the two major groups of organisms on Earth, including all animals, plants, protozoa and fungi. Characterised by the possession of a cell nucleus, and other membrane-bound cell organelles. Contrast with **prokaryote**. [1]

gene A unit of heredity. Commonly refers to a small section of the **genome**, but a number of different precise definitions exist.

genome All the **DNA** in the cells of an organism. That portion of a cell that physically represents its **genotype**. [2]

genotype The particular combination of **alleles** present at one or more genetic loci in some organism.

gradualism The doctrine that evolutionary change is gradual and does not go in jumps. [1]

macroevolution The study of evolutionary changes that take place over a very large time-scale. Contrast to **microevolution**. Macroevolutionary change is usually recognised as change in gross morphology in a series of fossils. There is some controversy over whether macroevolutionary change is fundamentally just cumulated microevolutionary change, or whether the two are 'decoupled' and driven by fundamentally different kinds of process. [1]

metazoan Any of a group (*Metazoa*) that comprises all animals having the body composed of cells differentiated into tissues and organs and usually a digestive cavity lined with specialised cells. [5]

microevolution The study of evolutionary changes within populations. Microevolutionary change is change in gene frequencies in populations. [1]

mRNA A type of **RNA** (abbreviation of 'messenger RNA'). Carries codes from the **DNA** in the nucleus to the sites of protein synthesis in the cytoplasm. [4]

mutualism Association between organisms of two different species in which each is benefited. Mutualistic arrangements are most likely to develop between organisms with widely differing living requirements. The partnership between nitrogen-fixing bacteria and leguminous plants is an example. [4]

niche The place of an organism within an ecosystem, defined by the ranges of the resources that it utilises. [6]

nucleotide Any of a class of compounds made of nitrogenous bases and pentose sugars (ribose, deoxy-ribose) with phosphates attached. [3]

ontogeny The process of individual development. In practice development is often taken to culminate in the production of the adult, but strictly speaking it includes later stages such as senescence. [1]

Paleozoic Era Major interval of geologic time that began about 540 million years ago with an extraordinary diversification of marine animals during the **Cambrian Period** and ended about 245 million years ago with the greatest extinction event in Earth history. [4]

parasitism Relationship between two species of plants or animals in which one benefits at the expense of the other, without killing it. [4]

Phanerozoic Eon The span of geologic time extending about 540 million years from the beginning of the **Paleozoic Era** to the present. [4]

phenotype The manifested attributes of an organism, the joint product of its genes and their environment during **ontogeny**. [1]

phylogeny A branching diagram of ancestor-descendant relationships along a temporal axis. [2]

phylum A group related by a direct line of descent. In taxonomies of biological organisms, the phylum is the basic unit of differentiation within kingdoms.

prokaryote One of the two major groups of organisms on Earth (contrast **eukaryote**) including bacteria and blue-green algae. They have no nucleus and no membrane-bounded organelles such as mitochondria; indeed one theory has it that mitochondria and other such organelles in eukaryotic cells are, in origin, symbiotic prokaryotic cells. [1]

protozoan any member of the subkingdom *Protozoa*, a collection of single-celled eukaryotic (i.e., possessing a well-defined nucleus) organisms. As such, they are among the simplest of all living organisms. [4]

punctuated equilibrium The pattern of evolution (very common in invertebrates) whereby species once in existence do not change radically over long periods of time (except perhaps in size) and then are suddenly replaced by other, quite different species. [2]

RNA Ribonucleic acid. Consists of ribose **nucleotides** in strands of varying lengths. The structure varies from helical to uncoiled strands.

selfish DNA A section of **DNA** on the **genome** which is not expressed phenotypically.

symbiogenesis The evolutionary origin of new morphologies and physiologies by **symbiosis**.

symbiosis Any of several living arrangements between members of two different species, including **mutualism**, **commensalism**, and **parasitism**. Both positive (beneficial) and negative (unfavourable to harmful) associations are therefore included, and the members are called symbionts. The terms symbiosis and mutualism are sometimes equated and used interchangeably; this practice has resulted in some confusion. [4]

sympatric Living in the same region. [2]

Bibliography

[Adami & Brown 94] Chris Adami and C. Titus Brown. Evolutionary learning in the 2D artificial life system 'Avida'. In R Brooks and P Maes, editors, *Artificial Life IV*, pages 377–381. The MIT Press, 1994.

[Adami *et al.* 98] C. Adami, R. Belew, H. Kitano, and C.Taylor, editors. *Artificial Life VI. Proceedings of the Sixth International Conference on Artificial Life*, MA, 1998. MIT Press.

[Arbib 69] Michael A. Arbib. Self-reproducing automata—some implications for theoretical biology. In C.H. Waddington, editor, *Towards a Theoretical Biology*, volume 2, pages 204–226. Edinburgh University Press, 1969.

[Aspray & Burks 87] W. Aspray and A. Burks, editors. *Papers of John von Neumann on Computing and Computer Theory.* MIT Press, 1987.

[Ayala 88] Francisco J. Ayala. Can "progress" be defined as a biological concept? In M.H. Nitecki, editor, *Evolutionary Progress*, pages 75–96. University of Chicago Press, 1988.

[Bagley & Farmer 91] Richard J. Bagley and J. Doyne Farmer. Spontaneous emergence of a metabolism. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 93–140. Addison-Wesley, Redwood City, CA, 1991.

[Banzhaf 94] Wolfgang Banzhaf. Self-organisation in a system of binary strings. In R. Brooks and P. Maes, editors, *Proceedings of Artificial Life IV*, pages 109–118, Cambridge, MA, 1994. MIT Press.

[Barricelli 57] Nils Aall Barricelli. Symbiogenetic evolution processes realized by artificial methods. *Methodos*, 9(35-36), 1957.

[Barricelli 62] Nils Aall Barricelli. Numerical testing of evolution theories. Part I. Theroetical introduction and basic tests. *Acta Biotheoretica*, XVI(1/2):69–98, 1962.

[Barricelli 63] Nils Aall Barricelli. Numerical testing of evolution theories. Part II. Preliminary tests of performance. Symbiogenesis and terrestrial life. *Acta Biotheoretica*, XVI(3/4):99–126, 1963.

[Barricelli 72] Nils Aall Barricelli. Numerical testing of evolution theories. *Journal of Statistical Computation and Simulation*, 1:97–127, 1972.

[Bedau & Brown 97] Mark A. Bedau and C. Titus Brown. Visualizing evolutionary activity of genotypes. (preprint). Also published as Working Paper No.98-03-023, Santa Fe Institute, NM., 1997.

[Bedau & Packard 91] Mark A. Bedau and Norman H. Packard. Measurement of evolutionary activity, teleology and life. In Langton, Taylor, Farmer, and Rasmussen, editors, *Artificial Life II*, pages 431–461. Addison-Wesley, Redwood City, CA, 1991.

[Bedau 96] Mark A. Bedau. The nature of life. In M.A. Boden, editor, *The Philosophy of Artificial Life*, pages 332–357. Oxford University Press, 1996.

[Bedau 98a] Mark A. Bedau. Personal communication, 1998.

[Bedau 98b] Mark A. Bedau. Four puzzles about life. *Artificial Life*, 4(2):125–140, 1998.

[Bedau *et al.* 97] Mark A. Bedau, Emile Snyder, C. Titus Brown, and Norman H. Packard. A comparison of evolutionary activity in artificial evolving systems and in the biosphere. In P Husbands and I Harvey, editors, *Fourth European Conference on Artificial Life*, pages 124–134. MIT Press/Bradford Books, 1997.

[Bedau *et al.* 98] Mark A. Bedau, Emile Snyder, and Norman H. Packard. A classification of long-term evolutionary dynamics. In C. Adami, R. Belew, H. Kitano, and C.Taylor, editors, *Proceedings of Artificial Life VI*, pages 228–237, MA, 1998. MIT Press.

[Belew & Mitchell 96] Richard K. Belew and Melanie Mitchell, editors. *Adaptive Individuals in Evolving Populations: Models and Algorithms.* Addison-Wesley, Reading, MA, 1996.

[Boden 96] Margaret A. Boden, editor. *The Philosophy of Artificial Life.* Oxford University Press, 1996.

[Boden 99] Margaret A. Boden. Is metabolism necessary? *British Journal for the Philosophy of Science*, 50, 1999. Forthcoming. Also available as University of Sussex Cognitive Science Research Paper No. 482.

[Boerlijst & Hogeweg 91] M.C. Boerlijst and P. Hogeweg. Spiral wave structure in pre-biotic evolution: Hypercycles stable against parasites. *Physica D*, 48:17–28, 1991.

[Bonabeau & Theraulaz 94] Eric W. Bonabeau and G. Theraulaz. Why do we need artificial life? *Artificial Life*, 1(3):303–325, 1994.

[Bonner 88] John Tyler Bonner. *The Evolution of Complexity.* Princeton University Press, 1988.

[Bratley & Millo 72] Paul Bratley and Jean Millo. Computer recreations: Self-reproducing programs. *Software—Practice and Experience*, 2:397–400, 1972.

[Bronowski 73] J. Bronowski. *The Ascent of Man.* British Broadcasting Corporation, 1973.

[Bull & Fogarty 95] Lawrence Bull and Terence C. Fogarty. Artificial symbiogenesis. *Artificial Life*, 2(3):269–292, 1995.

[Burger *et al.* 80] John Burger, David Brill, and Filip Machi. Self-reproducing programs. *Byte*, 5(8):72–74, August 1980.

[Buss 87] Leo W. Buss. *The Evolution of Individuality.* Princeton University Press, 1987.

[Cairns-Smith 71] A.G. Cairns-Smith. *The Life Puzzle.* Oliver and Boyd, Edinburgh, 1971.

[Cairns-Smith 85] A.G. Cairns-Smith. *Seven Clues to the Origin of Life.* Cambridge University Press, 1985.

[Cavalier-Smith 85] T. Cavalier-Smith. *The Evolution of Genome Size.* Wiley, Chichester, 1985.

[Chalmers 96] David J. Chalmers. *The Conscious Mind: In Search of a Fundamental Theory.* Oxford University Press, 1996.

[Channon & Damper 98] A.D. Channon and R.I. Damper. Perpetuating evolutionary emergence. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S.W. Wilson, editors, *From Animals to Animats 5: Proceedings of the Fifth International Conference of the Society for Adaptive Behavior.* MIT Press, 1998.

[Chou & Reggia 97] Hui-Hsien Chou and James A. Reggia. Emergence of self-replicating structures in a cellular automata space. *Physica D*, 110(3–4):252–276, 1997.

[Cliff & Miller 96] Dave Cliff and Geoffrey F. Miller. Co-evolution of pursuit and evasion II: Simulation methods and results. In P. Maes, M.J. Mataric, J.-A. Meyer, J. Pollack, and S.W. Wilson, editors, *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation and Adaptive Behavior*, pages 506–515, Cambridge, MA., 1996. MIT Press.

[Cohen 95] Paul R. Cohen. *Empirical Methods for Artificial Intelligence.* MIT Press, Camridge, MA., 1995.

[Collins 92] Robert James Collins. *Studies in Artificial Evolution.* Unpublished PhD thesis, Department of Computer Science, University of California, Los Angeles, 1992.

[Conrad & Pattee 70] Michael Conrad and H.H. Pattee. Evolution experiments with an artificial ecosystem. *Journal of Theoretical Biology*, 28:393–409, 1970.

[Conrad 88] Michael Conrad. The price of programmability. In R. Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 285–307. Oxford University Press, 1988.

[Coyne *et al.* 97] Jerry A. Coyne, Nicholas H. Barton, and Michael Turelli. A critique of Sewall Wright's shifting balance theory of evolution. *Evolution*, 51(3):643–671, June 1997.

[Daida *et al.* 96] Jason M. Daida, Catherine S. Grasso, Stephen A. Stanhope, and Steven J. Ross. Symbionticism and complex adaptive systems I: Implications of having symbiosis occur in nature. In *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, Cambridge, MA, 1996. MIT Press.

[Darwin 59] Charles Darwin. *The Origin of Species.* John Murray, London, 1859. (Reprinted in Penguin Classics, 1985. Any page numbers given in the text refer to this reprint.).

[Darwin 72] Charles Darwin, December 4 1872. Letter to the American paleontologist Alpheus Hyatt.

[Davidge 92] Robert Davidge. Processors as organisms. CSRP report 250, School of Cognitive and Computing Sciences, University of Sussex, 1992.

[Dawkins 76] Richard Dawkins. *The Selfish Gene.* Oxford University Press, Oxford, 1976. (Second edition published by O.U.P., 1989).

[Dawkins 82] Richard Dawkins. *The Extended Phenotype.* W.H. Freeman, Oxford, 1982.

[Dawkins 83] Richard Dawkins. Universal Darwinism. In D.S. Bendall, editor, *Evolution from Molecules to Men*, pages 403–425. Cambridge University Press, Cambridge, 1983.

[Day 85] Trevor Day. *Biology.* Penguin Passnotes. Penguin, 1985.

[de Dinechin 97] Florent de Dinechin. Self-replication in a 2D von Neumann architecture. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artificial Life*, pages 560–565. MIT Press/Bradford Books, 1997.

[de Duve 96] Christian de Duve. The birth of complex cells. *Scientific American*, pages 38–45, April 1996.

[Dennett 94] Daniel Dennett. Artificial life as philosophy. *Artificial Life*, 1(3):291–292, 1994.

[Dewdney 84] A.K. Dewdney. Computer Recreations: In the game called Core War hostile programs engage in a battle of bits. *Scientific American*, 250(5):15–19, May 1984.

[Di Paolo 96] Ezequiel A. Di Paolo. Some false starts in the construction of a research methodology for artificial life. In J. Noble and S. Parsowith, editors, *The 9th White House Papers: Graduate Research in the Cognitive and Computing Sciences at Sussex.* School of Cognitive and Computing Sciences, University of Sussex, 1996.

[Dittrich & Banzhaf 97] Peter Dittrich and Wolfgang Banzhaf. A toplogical structure based on hashing - emergence of a 'spatial' organisation. Poster presented at the Fourth European Conference on Artificial Life (ECAL97), Brighton, U.K., 1997.

[Dittrich & Banzhaf 98] Peter Dittrich and Wolfgang Banzhaf. Self-evolution in a constructive binary string system. *Artificial Life*, 4(2), 1998. in press.

[Dyson 85] Freeman J. Dyson. *Origins of Life.* Cambridge University Press, 1985.

[Dyson 97] George B. Dyson. *Darwin Among the Machines: The Evolution of Global Intelligence.* Addison Wesley, 1997.

[Eigen & Schuster 77] Manfred Eigen and Peter Schuster. The hypercycle: A principle of natural self-organization. *Die Naturwissenschaften*, 64(11):541–565, 1977.

[Eigen 71] Manfred Eigen. Self-organization of matter and the evolution of biological macromolecules. *Die Naturwissenschaften*, 58:465–523, 1971.

[Emmeche 92] Claus Emmeche. Life as an abstract phenomenon: Is artificial life possible? In F.J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 466–474. MIT Press, 1992.

[Feldman 89] Marcus W. Feldman. Discussion: Ecology and Evolution. In J. Roughgarden, R.M. May, and S.A. Levin, editors, *Perspectives in Ecological Theory*, chapter 9, pages 135–139. Princeton University Press, 1989.

[Feller 67] William Feller. *An Introduction to Probability Theory and its Applications*, volume 1 of *Wiley Series in Probability and Mathematical Statistics.* Wiley, New York, London, 3rd edition, 1967.

[Fisher 30] R.A. Fisher. *The Genetical Theory of Natural Selection.* Clarendon Press, Oxford, 1930.

[Floreano *et al.* 98] Dario Floreano, Stefano Nolfi, and Francesco Mondada. Competitive co-evolutionary robotics: From theory to practice. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S.W. Wilson, editors, *From Animals to Animats 5: Proceedings of the Fifth International Conference of the Society for Adaptive Behavior*, Cambridge, MA., 1998. MIT Press.

[Fogel 98] David B. Fogel, editor. *Evolutionary Computation: The Fossil Record.* IEEE Press, New York, 1998.

[Fontana & Buss 96] Walter Fontana and Leo W. Buss. The barrier of objects: From dynamical systems to bounded organizations. In J. Casti and A. Karlqvist, editors, *Boundaries and Barriers*, pages 56–116. Addison-Wesley, 1996.

[Fontana 91] Walter Fontana. Algorithmic chemistry. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 211–254, Redwood City, CA, 1991. Addison-Wesley.

[Fontana *et al.* 94] Walter Fontana, Günter Wagner, and Leo W. Buss. Beyond digital naturalism. *Artificial Life*, 1:211–227, 1994.

[Gandy 88] Robin Gandy. The confluence of ideas in 1936. In R. Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 55–111. Oxford University Press, 1988.

[Goldberg 89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[Goodwin 94] Brian Goodwin. *How the Leopard Changed its Spots: The Evolution of Complexity.* Weidenfeld and Nicolson, London, 1994.

[Gould 88] Stephen Jay Gould. On replacing the idea of progress with an operational notion of directionality. In M.H. Nitecki, editor, *Evolutionary Progress*, pages 319–338. University of Chicago Press, 1988.

[Gould 89] Stephen Jay Gould. *Wonderful Life: The Burgess Shale and the Nature of History.* Penguin Books, 1989.

[Harvey 93] Inman Harvey. *The Artificial Evolution of Adaptive Behaviour.* Unpublished PhD thesis, Department of Cognitive Science, University of Sussex, 1993.

[Haskell 40] E. F. Haskell. Mathematical systematization of environment, organism and habitat. *Ecology*, 21:1–16, 1940.

[Hillis 90] W. Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[Hofstadter 79] Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid.* Basic Books, New York, 1979.

[Holland 62] John H. Holland. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, 9(3):297–314, July 1962. (Reprinted in Essays on Cellular Automata, A.W. Burks (ed.), University of Illinois Press, 1970. Any page numbers given in text refer to this reprint.).

[Holland 75] John H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[Holland 76] John H. Holland. Studies of the spontaneous emergence of self-replicating systems using cellular automata and formal grammars. In A. Lindenmayer and G. Rozenberg, editors, *Automata, Languages, Development*, pages 385–404. North-Holland, New York, 1976.

[Holland 95] John H. Holland. *Hidden Order: How Adaptation Builds Complexity.* Addison-Wesley/Helix Books, 1995.

[Hraber & Milne 97] Peter T. Hraber and Bruce T. Milne. Community assembly in a model ecosystem. *Ecological Modelling*, 103:267–285, 1997.

[Hraber *et al.* 97] Peter T. Hraber, Terry Jones, and Stephanie Forrest. The ecology of Echo. *Artificial Life*, 3(3):165–190, 1997.

[Hull 80] David L. Hull. Individuality and selection. *Annual Review of Ecology and Systematics*, 11:311–332, 1980.

[Hull 88] David L. Hull. Progress in ideas of progress. In M.H. Nitecki, editor, *Evolutionary Progress.* University of Chicago Press, 1988.

[Husbands & Harvey 97] P. Husbands and I. Harvey, editors. *Proceedings of the Fourth European Conference on Artificial Life.* MIT Press/Bradford Books, 1997.

[Ibáñez *et al.* 95] Jesús Ibáñez, Daniel Anabitarte, Iker Azpeitia, Oscar Barrera, Arkaitz Barrutieta, Haritz Blanco, and Francisco Echarte. Self-inspection based reproduction in cellular automata. In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life: Third European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 564–576, Berlin, 1995. Springer.

[Ikegami & Kaneko 90] Takashi Ikegami and Kunihiko Kaneko. Genetic fusion. *Physical Review Letters*, 65(26):3352–3355, December 1990.

[Jacobson 58] Homer Jacobson. On models of reproduction. *American Scientist*, 46:255–284, 1958.

[Jefferson *et al.* 91] D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang. The Genesys system: Evolution as a theme in artificial life. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 549–578. Addison-Wesley, Redwood City, CA, 1991.

[Kacser & Beeby 84] Henrik Kacser and Richard Beeby. Evolution of catalytic proteins. *Journal of Molecular Evolution*, 20:38–51, 1984.

[Kauffman 86] Stuart A. Kauffman. Autocatalytic sets of proteins. *Journal of Theoretical Biology*, (119):1–24, 1986.

[Kauffman 93] Stuart A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.

[Kauffman 95] Stuart A. Kauffman. *At Home in the Universe.* Viking, 1995.

[Kelly 94] Kevin Kelly. *Out of Control: The New Biology of Machines.* Fourth Estate, London, 1994.

[Khakhina 79] Liya N. Khakhina. *Concepts of Symbiogenesis.* Akademie NAUK, USSR, 1979.

[Koehl 89] M.A.R. Koehl. Discussion: From individuals to populations. In J. Roughgarden, R.M. May, and S.A. Levin, editors, *Perspectives in Ecological Theory*, chapter 3, pages 39–53. Princeton University Press, 1989.

[Koza 92] John R. Koza. *Genetic Programming: on the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[Koza 94] John R. Koza. Artificial life: Spontaneous emergence of self-replicating and evolutionary self-improving computer programs. In C Langton, editor, *Artificial Life III*, pages 225–262. Addison-Wesley, 1994.

[Laing 77] Richard Laing. Automaton models of reproduction by self-inspection. *Journal of Theoretical Biology*, 66:437–456, 1977.

[Langton 84] Christopher G. Langton. Self-reproduction in cellular automata. *Physica D*, (10):135–144, 1984.

[Langton 86] Christopher G. Langton. Studying artificial life with cellular automata. *Physica D*, (22):120–149, 1986.

[Langton 88] Christopher G. Langton. Artificial life. In C.G. Langton, editor, *Artificial Life*, volume VI of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 1–47, Reading, MA, 1988. Addison-Wesley.

[Lazcano 95] Antonio Lazcano. Prebiotic chemistry, artificial life and complexity theory: What do they tell us about the origin of biological systems? In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life: Third European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 105–115, Berlin, 1995. Springer.

[Lindgren 91] Kristian Lindgren. Evolutionary phenomena in simple dynamics. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 295–312, Redwood City, CA, 1991. Addison-Wesley.

[Löfgren 68] Lars Löfgren. An axiomatic explanation of complete self-reproduction. *Bulletin of Mathematical Biophysics*, 30(3):415–425, 1968.

[Löfgren 72] Lars Löfgren. Relative explanations of systems. In G.J. Klir, editor, *Trends in General Systems Theory.* John Wiley and Sons, 1972.

[Lovelock 79] James E. Lovelock. *Gaia: A New Look at Life on Earth.* Oxford University Press, 1979.

[Maley 93] Carlo C. Maley. A model of early evolution in two dimensions. Unpublished M.Sc. thesis, Oxford University, 1993.

[Maley 94] Carlo C. Maley. The computational completeness of Ray's Tierran assembly language. In C.G. Langton, editor, *Artificial Life III*, volume XVII of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 503–514. Addison-Wesley, 1994.

[Margulis & Sagan 86] Lynn Margulis and Dorion Sagan. *Origins of Sex: Three Billion Years of Genetic Recombination.* Yale University Press, New Haven, 1986.

[Margulis 81] Lynn Margulis. *Symbiosis in Cell Evolution: Life and its Environment on the Early Earth.* Freeman, 1981.

[Margulis 91] Lynn Margulis. Symbiogenesis and symbionticism. In Lynn Margulis and René Fester, editors, *Symbiosis as a Source of Evolutionary Innovation*, pages 1–14. MIT Press, Cambridge, MA, 1991.

[Marr 82] David Marr. *Vision.* W.H. Freeman, New York, 1982.

[Matthews 91] R.E.F. Matthews. *Plant Virology.* Academic Press, San Diego, CA, 3rd edition, 1991.

[Maturana & Varela 80] Humberto R. Maturana and Francisco J. Varela. *Autopoiesis and Cognition: The Realization of the Living.* Reidel, 1980.

[Mayer & Rasmussen 98] Bernd Mayer and Steyn Rasmussen. Self-reproduction of dynamical hierarchies in chemical systems. In C. Adami, R. Belew, H. Kitano, and C.Taylor, editors, *Proceedings of Artificial Life VI*, pages 123–129, MA, 1998. MIT Press.

[Maynard Smith & Szathmáry 95] John Maynard Smith and Eörs Szathmáry. *The Major Transitions in Evolution.* W.H. Freeman, Oxford, 1995.

[Maynard Smith 86] John Maynard Smith. *The Problems of Biology.* Oxford University Press, 1986.

[Maynard Smith 88] John Maynard Smith. Evolutionary progress and levels of selection. In M.H. Nitecki, editor, *Evolutionary Progress*, pages 219–230. University of Chicago Press, 1988.

[Maynard Smith 89] John Maynard Smith. *Evolutionary Genetics.* Oxford University Press, 1989.

[Maynard Smith 91] John Maynard Smith. A Darwinian view of symbiosis. In Lynn Margulis and René Fester, editors, *Symbiosis as a Source of Evolutionary Innovation*, pages 26–39. MIT Press, Cambridge, MA, 1991.

[Maynard Smith 96] John Maynard Smith. A developing synthesis. *Evolution*, 50(6):2562–2563, 1996. Book review of "The Shape of Life: Genes, Development, and the Evolution of Animal Form", Rudolf A. Raff. 1996. University of Chicago Press.

[McMullin & Varela 97] Barry McMullin and Francisco J. Varela. Rediscovering computational autopoiesis. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artificial Life*, pages 38–47. MIT Press/Bradford Books, 1997.

[McMullin 92a] Barry McMullin. *Artificial Knowledge: An Evolutionary Approach.* Unpublished PhD thesis, Department of Computer Science, University College Dublin, October 1992. URL: `ftp://ftp.eeng.dcu.ie/pub/alife/bmcm_phd/`.

[McMullin 92b] Barry McMullin. The Holland α-universes revisited. In F.J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 317–326. MIT Press, 1992.

[McMullin 95] Barry McMullin. Replicators don't! In F Morán, A Moreno, JJ Merelo, and P Chacón, editors, *Advances in Artificial Life: Third Euopean Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 158–169. Springer, 1995.

[McShea 91] Daniel W. McShea. Complexity and evolution: What everybody knows. *Biology and Philosophy*, 6:303–324, 1991.

[McShea 94] Daniel W. McShea. Mechanisms of large-scale evolutionary trends. *Evolution*, 48(6):1747–1763, December 1994.

[McShea 96] Daniel W. McShea. Metazoan complexity and evolution: Is there a trend? *Evolution*, 50(2):477–492, April 1996.

[McShea 98] Daniel W. McShea. Possible largest-scale trends in organismal evolution: Eight "live hypotheses". *Annula Review of Ecology and Systematics*, 29:293–318, 1998.

[Miller & Cliff 94] Geoffrey F. Miller and Dave Cliff. Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3, Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior (SAB'94)*, pages 411–420. MIT Press/Bradford Books, 1994.

[Miller 95] Geoffrey F. Miller. Artificial life as theoretical biology: How to do real science with computer simulation. Cognitive Science Research Paper 378, School of Cognitive and Computing Sciences, University of Sussex, 1995.

[Moore 62] Edward F. Moore. Machine models of self-reproduction. In *Proceedings of Symposia in Applied Mathematics*, volume 14, pages 17–33. The American Mathematical Society, 1962. (Reprinted in Essays on Cellular Automata, A.W. Burks (ed.), University of Illinois Press, 1970).

[Morán *et al.* 97] Federico Morán, Alvaro Moreno, Eric Minch, and Francisco Montero. Further steps towards the realistic description of the essence of life. In C.G. Langton and K. Shimohara, editors, *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, pages 216–224, Cambridge, MA, 1997. MIT Press.

[Moreno *et al.* 94] Alvaro Moreno, Arantza Etxeberria, and Jon Umerez. Universality without matter? In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 406–410. The MIT Press, 1994.

[Morita & Imai 97] Kenichi Morita and Katsunobu Imai. A simple self-reproducing cellular automaton with shape-encoding mechanism. In C.G. Langton and K. Shimohara, editors, *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, pages 450–457, Cambridge, MA, 1997. MIT Press.

[Morowitz 59] Harold J. Morowitz. A model of reproduction. *American Scientist*, 47:261–263, 1959.

[Morris 88] Harold C. Morris. Typogenetics: A logic for artificial life. In C.G. Langton, editor, *Artificial Life*, volume VI of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 369–395, Reading, MA, 1988. Addison-Wesley.

[Muller 66] H.J. Muller. The gene material as the initiator and the organizing basis of life. *The American Naturalist*, 100(915):493–517, 1966.

[Myhill 63] John Myhill. The converse of Moore's Garden-of-Eden theorem. In *Proceedings of the American Mathematical Society*, volume 14, pages 658–686, 1963. (Reprinted in Essays on Cellular Automata, A.W. Burks (ed.), University of Illinois Press, 1970).

[Myhill 64] John Myhill. The abstract theory of self-reproduction. In M.D. Mesarović, editor, *Views on General Systems Theory: Proceedings of the Second Systems Symposium at Case Institute of Technology*, pages 106–118, New York, 1964. John Wiley and Sons. (Reprinted in Essays on Cellular Automata, A.W. Burks (ed.), University of Illinois Press, 1970. Any page numbers given in text refer to this reprint.).

[Nitecki 88] Matthew H. Nitecki. Discerning the criteria for concepts of progress. In M.H. Nitecki, editor, *Evolutionary Progress*, pages 3–24. University of Chicago Press, 1988.

[Noble 97] Jason Noble. The scientific status of artificial life. Poster presented at the Fourth European Conference on Artificial Life (ECAL97), Brighton, U.K., 1997.

[Nolfi & Floreano 98] Stefano Nolfi and Dario Floreano. Co-evolving predator and prey robots: Do 'arms-races' arise in artificial evolution? *Artificial Life*, 4(4), 1998. (in press).

[Nuño *et al.* 95] Juan C. Nuño, Pablo Chacón, Alvaro Moreno, and Federico Morán. Compartimentation in replicator models. In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life: Third European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 116–127, Berlin, 1995. Springer.

[Numaoka 95] Chisato Numaoka. Symbiosis and co-evolution in animats. In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life: Third Euopean Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 261–272. Springer, 1995.

[Olson 97] Eric T. Olson. The ontological basis of strong artificial life. *Artificial Life*, 3:29–39, 1997.

[O'Neill *et al.* 86] R.V. O'Neill, D.L. DeAngelis, J.B. Waide, and T.F.H. Allen. *A Hierarchical Concept of Ecosystems.* Princeton University Press, Princeton, NJ., 1986.

[Orgel 79] L.E. Orgel. Selection in vitro. *Proceedings of the Royal Society, London*, B, 205:435–442, 1979.

[Overton 82] William Overton. Creationism, science, artificial intelligence, and artificial life: The memorandum opinion of Judge William Overton—Part 1. *Computers and People*, 31(11–12):16–24, 1982.

[Oyama 85] Susan Oyama. *The Ontongeny of Information: Developmental Systems and Evolution.* Cambridge University Press, 1985.

[Packard 88] Norman H. Packard. Intrinsic adaptation in a simple model for evolution. In C.G. Langton, editor, *Artificial Life*, volume VI of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 141–155, Reading, MA, 1988. Addison-Wesley.

[Pargellis 96] A.N. Pargellis. The spontaneous generation of digital 'life'. *Physica D*, 91:86–96, 1996.

[Pask 69] Gordon Pask. The computer-simulated development of populations of automata. *Mathematical Biosciences*, 4:101–127, 1969.

[Pattee 88] H.H. Pattee. Simulations, realizations, and theories of life. In C. Langton, editor, *Artificial Life*, volume VI of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 63–77, Reading, MA, 1988. Addison-Wesley. Reprinted in "The Philosophy of Artificial Life", M.A. Boden (ed.), Oxford University Press, 1996.

[Pattee 95a] H.H. Pattee. Artificial life needs a real epistemology. In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life: Third European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 23–38, Berlin, 1995. Springer.

[Pattee 95b] H.H. Pattee. Evolving self-reference: Matter, symbols, and semantic closure. *Communication and Cognition—Artificial Intelligence*, 12(1–2):9–28, 1995.

[Penrose 58] L.S. Penrose. Mechanisms of self-reproduction. *Annals of Human Genetics*, 23:59–72, 1958.

[Penrose 59] L.S. Penrose. Self-reproducing machines. *Scientific American*, 200(6):105–113, 1959.

[Penrose 62] L.S. Penrose. On living matter and self-replication. In I.J. Good, A.J. Mayne, and J. Maynard Smith, editors, *The Scientist Speculates: An Anthology of Partly-Baked Ideas.* Heinemann, London, 1962.

[Perrier *et al.* 96] Jean-Yves Perrier, Moshe Sipper, and Jacques Zahnd. Toward a viable, self-reproducing universal computer. *Physica D*, (97):335–352, 1996.

[Pesavento 95] Umberto Pesavento. An implementation of von Neumann's self-reproducing machine. *Artificial Life*, 2(4):337–354, 1995.

[Raff 97] R.A. Raff. *The Shape of Life: Genes, Development and the Evolution of Animal Form.* University of Chicago Press, 1997.

[Rasmussen 91] Steen Rasmussen. Aspects of information, life, reality and physics. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 767–773. Addison-Wesley, Redwood City, CA, 1991.

[Rasmussen *et al.* 90] Steen Rasmussen, Carsten Knudsen, Rasmus Feldberg, and Morten Hindsholm. The coreworld: Emergence and evolution of cooperative structures in a computational chemistry. *Physica D*, 42:111–134, 1990.

[Rasmussen *et al.* 91] Steen Rasmussen, Carsten Knudsen, and Rasmus Feldberg. Dynamics of programmable matter. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 211–254, Redwood City, CA, 1991. Addison-Wesley.

[Raup & Sepkoski 82] D.M. Raup and J.J. Sepkoski. Mass extinctions in the marine fossil record. *Science*, 215:1501–1503, 1982.

[Ray & Hart 98] Thomas S. Ray and Joseph Hart. Evolution of differentiated multi-threaded digital organisms. In C. Adami, R. Belew, H. Kitano, and C.Taylor, editors, *Proceedings of Artificial Life VI*, pages 295–304, MA, 1998. MIT Press.

[Ray 91] 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, Redwood City, CA, 1991.

[Ray 94a] Thomas S. Ray. Evolution, complexity, entropy and artificial reality. *Physica D*, 75:239–263, 1994.

[Ray 94b] Thomas S. Ray. An evolutionary approach to synthetic biology: Zen and the art of creating life. *Artificial Life*, 1(1/2):195–226, 1994.

[Ray 97] Thomas S. Ray. Evolving complexity. *Artificial Life and Robotics*, 1:21–26, 1997.

[Reanney 79] D. Reanney. RNA splicing and polynucleotide evolution. *Nature*, 277:598–600, February 1979.

[Rosen 59] Robert Rosen. On a logical paradox implicit in the notion of a self-reproducing automaton. *Bulletin of Mathematical Biophysics*, 21:387–394, 1959.

[Roughgarden *et al.* 96] Jonathan Roughgarden, Aviv Bergman, Sharoni Shafir, and Charles Taylor. Adaptive computation in ecology and evolution: A guide for future research. In R.K. Belew and M. Mitchell, editors, *Adaptive Individuals in Evolving Populations*, pages 25–30. Addison-Wesley, Reading, MA, 1996.

[Ruiz-Mirazo *et al.* 98] Kepa Ruiz-Mirazo, Alvaro Moreno, and Federico Morán. Merging the energetic and the relational-constructive logic of life. In C. Adami, R. Belew, H. Kitano, and C.Taylor, editors, *Proceedings of Artificial Life VI*, pages 448–451, MA, 1998. MIT Press.

[Ruse 88] Michael Ruse. Molecules to men: Evolutionary biology and thoughts of progress. In M.H. Nitecki, editor, *Evolutionary Progress*. University of Chicago Press, 1988.

[Salthe 85] Stanley N. Salthe. *Evolving Hierarchical Systems: Their Structure and Representation.* Columbia University Press, New York, 1985.

[Schmitz & Booth 96] Oswald J. Schmitz and Ginger Booth. Modeling food web complexity: the consequence of individual-based spatially explicit behavioral ecology on trophic interactions. *Evolutionary Ecology*, 11:379–398, 1996.

[Schwemmler 89] Werner Schwemmler. *Symbiogenesis: A Macro-Mechanism of Evolution.* Walter de Gruyter, Berlin, 1989.

[Sims 94a] Karl Sims. Evolving 3D morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Proceedings of Artificial Life IV*, pages 28–39, Cambridge, MA, 1994. MIT Press.

[Sims 94b] Karl Sims. Evolving virtual creatures. In *Computer Graphics, Annual Conference Series (SIGGRAPH'94)*, pages 15–22, July 1994.

[Sipper 98] Moshe Sipper. Fifty years of research on self-replication: An overview. *Artificial Life*, 4(3):237–257, 1998.

[Skipper 92] Jakob Skipper. The computer zoo—evolution in a box. In F.J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 355–364, Cambridge, MA, 1992. MIT Press.

[Smith 91] Alvy Ray Smith. Simple nontrivial self-reproducing machines. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 709–725. Addison-Wesley, Redwood City, CA, 1991.

[Sober 91] Elliott Sober. Learning from functionalism—Prospects for strong artificial life. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, 1991.

[Stanley 73] Steven M. Stanley. An ecological theory for the sudden origin of multicellular life in the late Precambrian. *Proc. Nat. Acad. Sci.*, 70:1486–1489, 1973.

[Stanley 89] Steven M. Stanley. Fossils, macroevolution, and theoretical ecology. In J. Roughgarden, R.M. May, and S.A. Levin, editors, *Perspectives in Ecological Theory*, chapter 8, pages 125–134. Princeton University Press, 1989.

[Stewart 97] John Stewart. Evolutionary transitions and artificial life. *Artificial Life*, 3(2):101–120, 1997.

[Szathmáry & Demeter 87] Eörs Szathmáry and László Demeter. Group selection of early replicators and the origin of life. *Journal of Theoretical Biology*, 128:463–486, 1987.

[Szathmáry & Maynard Smith 95] Eörs Szathmáry and John Maynard Smith. The major evolutionary transitions. *Nature*, 374:227–232, March 1995.

[Taylor & Hallam 97] Tim Taylor and John Hallam. Studying evolution with self-replicating computer programs. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artificial Life*, pages 550–559. MIT Press/Bradford Books, 1997.

[Taylor & Hallam 98] Tim Taylor and John Hallam. Replaying the tape: An investigation into the role of contingency in evolution. In C. Adami, R. Belew, H. Kitano, and C.Taylor, editors, *Proceedings of Artificial Life VI*, pages 256–265, MA, 1998. MIT Press.

[Taylor & Jefferson 94] Charles Taylor and David Jefferson. Artificial life as a tool for biological inquiry. *Artificial Life*, 1(1–2):1–13, 1994.

[Taylor 98] Tim Taylor. Using bottom-up models to investigate the evolution of life: Steps towards an improved methodology. In C.L. Nehaniv and G.P. Wagner, editors, *The Right Stuff: Appropriate Mathematics for Evolutionary and Development Biology*, number 315 in Computer Science Technical Report, pages 23–26, U.K., June 1998. University of Hertfordshire School of Information Sciences.

[Taylor *et al.* 88] C.E. Taylor, D.R. Jefferson, S.R. Turner, and S.E. Goldman. RAM: Artificial life for the exploration of complex biological systems. In C. Langton, editor, *Artificial Life*, volume VI of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 275–295, Reading, MA, 1988. Addison-Wesley.

[Tempesti 95] Gianluca Tempesti. A new self-reproducing cellular automata capable of construction and computation. In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life: Third European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, Berlin, 1995. Springer.

[Thearling & Ray 94] Kurt Thearling and Thomas S. Ray. Evolving multicellular artificial life. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 283–288. The MIT Press, 1994.

[Thearling & Ray 96] Kurt Thearling and Thomas S. Ray. Evolving parallel computation. *Complex Systems*, 10:229–237, 1996.

[Thearling 94] Kurt Thearling. Evolution, entropy and parallel computation. In W. Porod, editor, *Proceedings of the Workshop on Physics and Computation (PhysComp94)*, Los Alamitos, November 1994. IEEE Press.

[Thompson 17] D'Arcy W. Thompson. *On Growth and Form.* Cambridge University Press, 1917. (Abridged edition, edited by John Tyler Bonner, also published by C.U.P., 1992).

[Toquenaga & Wade 96] Yukihiko Toquenaga and Michael J. Wade. Sewall Wright meets artificial life: the origin and maintenance of evolutionary novelty. *Trends in Ecology and Evolution*, 11(11):478–482, November 1996.

[Travis & Mueller 89] Joseph Travis and Laurence D. Mueller. Blending ecology and genetics: Progress toward a unified population biology. In J. Roughgarden, R.M. May, and S.A. Levin, editors, *Perspectives in Ecological Theory*, chapter 7, pages 101–124. Princeton University Press, 1989.

[Turing 36] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 42:230–265, 1936.

[van Baalen & Rand 98] Minus van Baalen and David A. Rand. The unit of selection in viscous populations and the evolution of altruism. *Journal of Theoretical Biology*, 193:631–648, 1998.

[van Baalen 98] Minus van Baalen. Pair dynamics analysis of artificial ecologies on lattices with different geometries. In J.A.J. Metz, U. Dieckmann, and R. Law, editors, *The Geometry of Ecological Interactions: Simplifying Spatial Structure.* Cambridge University Press, 1998. (to appear).

[Van Valen 73] L. Van Valen. A new evolutionary law. *Evolutionary Theory*, 1:1–30, 1973.

[Varela 79] Francisco J. Varela. *Principles of Biological Autonomy.* North Holland, Amsterdam, 1979.

[Varela *et al.* 74] Francisco J. Varela, Humberto R. Maturana, and R. Uribe. Autopoiesis: The organization of living systems, its characterization and a model. *BioSystems*, 5:187–196, 1974.

[Varetto 93] Louis Varetto. Typogenetics: An artificial genetic system. *Journal of Theoretical Biology*, 160:185–205, 1993.

[von Neumann 49] John von Neumann. Re-evaluation of the problems of complicated automata—problems of hierarchy and evolution (Fifth Illinois Lecture). December 1949. Lecture notes published in "Papers of John von Neumann on Computing and Computer Theory" (pp.477-490), W. Aspray and A. Burks (eds.), MIT Press, 1987.

[von Neumann 66] John von Neumann. *The Theory of Self-Reproducing Automata.* University of Illinois Press, Urbana, Ill., 1966.

[Waddington 57] C.H. Waddington. *The Strategy of the Genes: A Discussion of Some Aspects of Theoretical Biology.* George Allen and Unwin, London, 1957.

[Waddington 69] C.H. Waddington. Paradigm for an evolutionary process. In C.H. Waddington, editor, *Towards a Theoretical Biology*, volume 2, pages 106–128. Edinburgh University Press, 1969.

[Weiner 48] Norbert Weiner. *Cybernetics.* John Wiley, New York, 1948. (Second edition 1962).

[Williams 66] G.C. Williams. *Adaptation and Natural Selection.* Princeton University Press, Princeton, NJ, 1966.

[Wittgenstein 21] Ludwig Wittgenstein. Tractatus logico-philosophicus. *Annalen def Naturphilosophie*, 1921. In German. English edition London: Routledge and Kegan Paul, 1961.

[Wolpert 93] Lewis Wolpert. *The Triumph of the Embryo.* Oxford University Press, 1993.

[Wright 31] Sewall Wright. Evolution in Mendelian populations. *Genetics*, 16:97–159, 1931.

[Wright 82] Sewall Wright. The shifting balance theory and macroevolution. *Annual Review of Genetics*, 16:1–19, 1982.

[Yaeger 94] Larry Yaeger. Computational genetics, physiology, metabolism, neural systems, learning, vision and behavior or poly-world: Life in a new context. In C.G. Langton, editor, *Artificial Life III*, volume XVII of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 263–298. Addison-Wesley, 1994.

[Yamamoto & Kaneko 97] Tomoyuki Yamamoto and Kunihiko Kaneko. Igniting the cycle of creation—an approach to create metabolism with Tile Automaton. In C.G. Langton and K. Shimohara, editors, *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, pages 399–406, Cambridge, MA, 1997. MIT Press.

[Yoshii *et al.* 98a] Shinichiro Yoshii, Satoshi Ohashi, and Yukinori Kakazu. Self-organized complexity in computer program ecosystem. In C. Adami, R. Belew, H. Kitano, and C.Taylor, editors, *Proceedings of Artificial Life VI*, pages 483–488, MA, 1998. MIT Press.

[Yoshii *et al.* 98b] Shinichiro Yoshii, Satoshi Ohasi, and Yukinori Kakazu. Modeling of emergent ecology for simulating adaptive behavior of universal computer programs. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S.W. Wilson, editors, *From Animals to Animats 5: Proceedings of the Fifth International Conference of the Society for Adaptive Behavior*, Cambridge, MA., 1998. MIT Press.

[Zaug & Cech 86] A.J. Zaug and T.R. Cech. The intervening sequence RNA of *Tetrahymena* is an enzyme. *Science*, 231:470–475, 1986.

[Zeleny 77] Milan Zeleny. Self-organization of living systems: A formal model of autopoiesis. *International Journal of General Systems*, 4:13–28, 1977.