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 theor-

etical 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 condi-

tion 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-

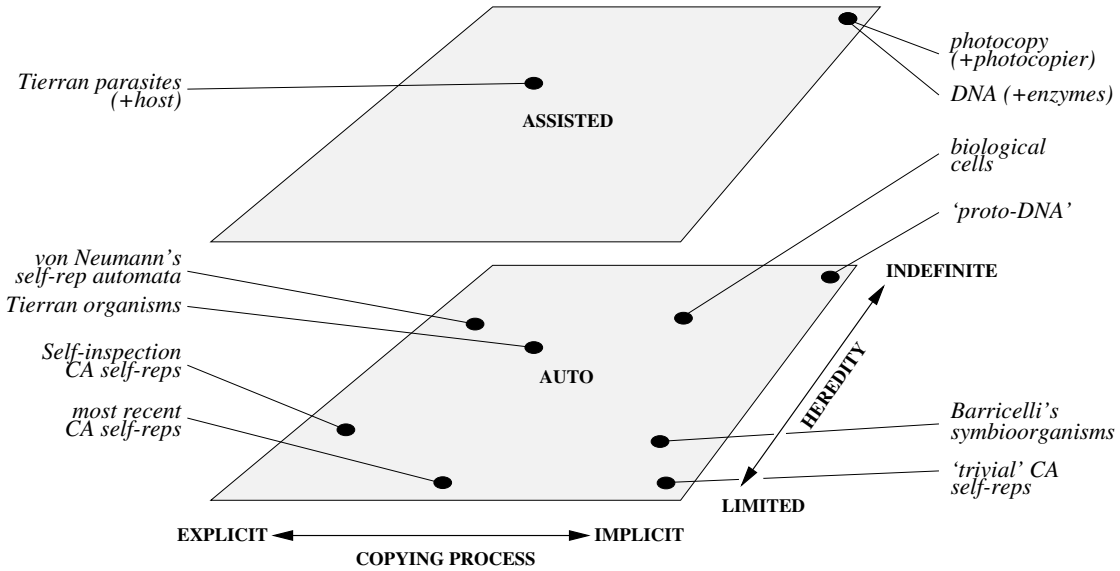

Figure 7.1: Categorisation of Reproducers.

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. **A**, **B**, **C** and **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, **B** and **D** are explicitly encoded on the tape $\phi(\mathbf{B}+\mathbf{D})$, but **A** and **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 **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 **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.

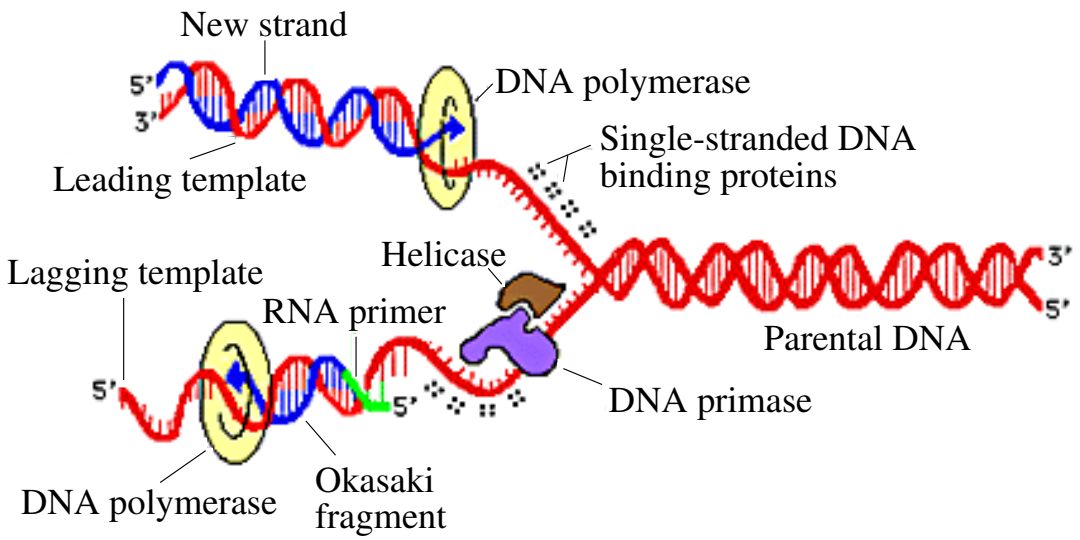

Figure 7.2: DNA Replication.[15]

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].

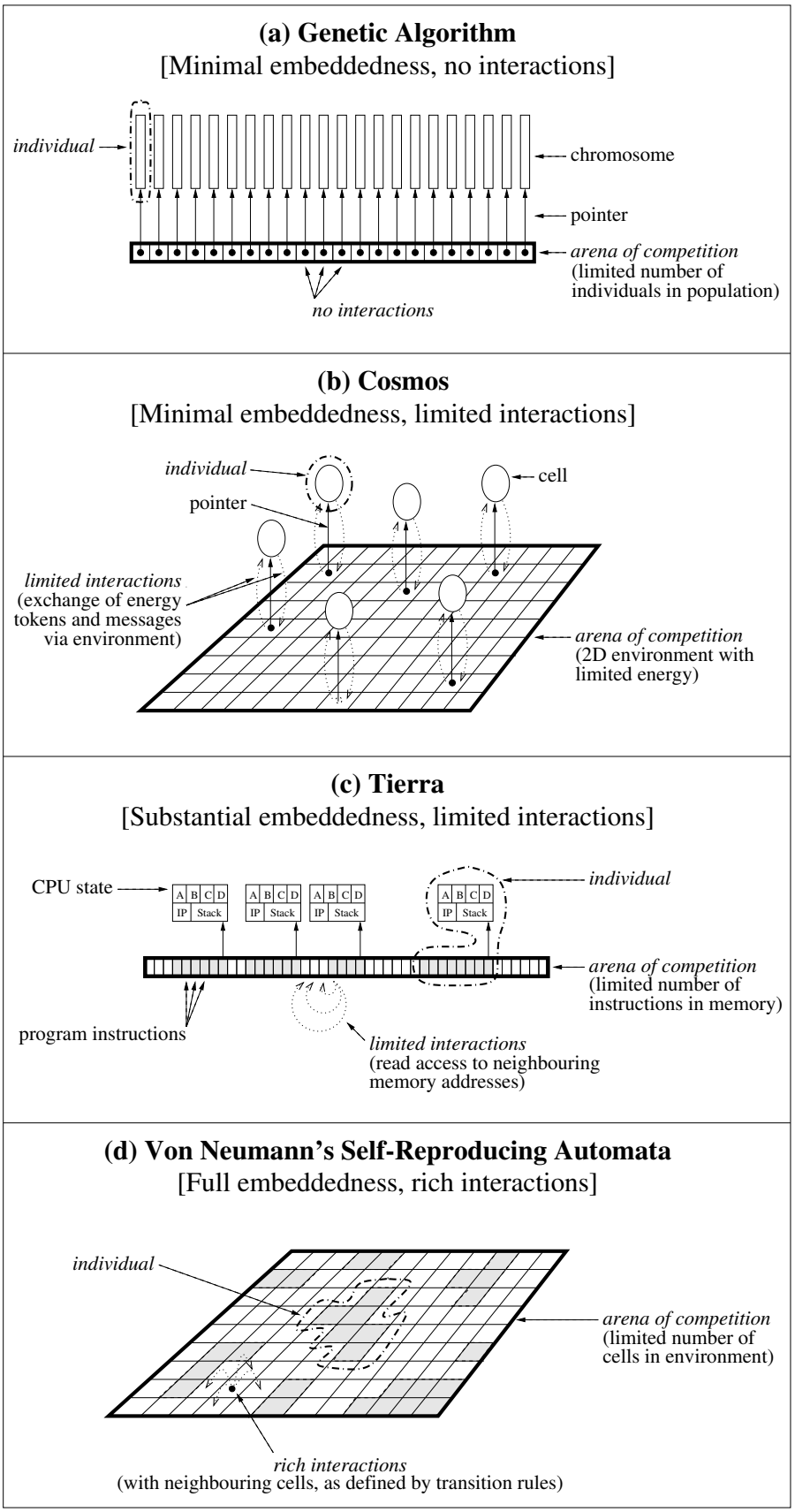

Figure 7.3: Embeddedness of Individuals and Richness of Interactions in Various Artificial Evolutionary Platforms.

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 dis-

tinction 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 ex-

plicit 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 predefined 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:

```
 1  101100
 2  et_collect
 3  nwm_clear
 4  adrb
 5  nop_10
 6  nop_11
 7  nop_00
 8  push_a
 9  pop_c
10  et_collect
11  dec_c
12  dec_c
13  dec_c
14  dec_c
15  dec_c
16  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.