THE UNIVERSITY OF SUSSEX

An Empirical Exploration of Computations with a Cellular-Automata-Based Artificial Life World

Pedro Paulo Balbi de Oliveira

Submitted for the degree of D. Phil.

February 23, 1995

CONTENTS

LIST OF FIGURES

LIST OF TABLES

DECLARATION

I hereby declare that this thesis has not been submitted, either in the same or different form, to this or any other university for a degree.

Signature:

ACKNOWLEDGEMENTS

This thesis became possible thanks to grants (No. 200695/88.6 and No. 451458/94-0) from CNPq, the Brazilian Council for Scientific and Technological Development; and to the leave of absence conceded by my employer, INPE, the National Institute for Space Research. I thank them and express my deepest recognition for their investment.

I'm very grateful to Phil Husbands, who became my supervisor at the end of the first year, for his unconditional support at the various stages of this thesis, and in particular for his midwifery work as I tried to get the final version of this thesis together.

My most special gratitude to Inman Harvey, for all that I've learned from him, for all the everyday discussions, and for all the great stories from his past.

Coincidently, around the time I arrived at Cogs various people with similar interests also arrived; I regret that Peter Hyett, probably Cogs' first *alifer*, did arrive too early. Our *alergic* meetings on alife were certainly the best reflection of the extremely stimulating environment that was created. Thanks a lot to the alergic members, the first generation, who one way or another contributed to this work: Robert Davidge, Mukesh Pattel, Dave Cliff, Andy Wuensche and Serge Thill.

Thanks to Mike Scaife, my supervisor during the first year, for having accepted me at Cogs, and whose friendliness and personal support were essential in my first months.

I'm also very grateful to generous travel grants from Cogs' Postgraduate Fund; AISB, the UK Society for Artificial Intelligence and Simulation of Behaviour; Université de Compiègne; and Santa Fe Institute.

Thanks to Orlando and Sandra, Teresa, Miguel, Rui and Sandra, and Antonio for always being close; Eevi, Alistair, Arantza, Chris Taylor and Paulo for the friendship; Lionel Moser for discussions; the designers of Cellsim for keeping it in the public domain; Richard Dallaway for his helpfulness; Valeria for the ceiling; and... Anyway, thanks to all of you folks with whom I shared good moments, on campus, in Brighton, in D421 (yes, I'm the last one!) and in 4C15.

And certainly, my tenderest gratitude to Carmen and Luara, for their love, patience, and companionship during this long journey.

Para
Dona Lindinha
e
Seu Altamiro

THE UNIVERSITY OF SUSSEX

An Empirical Exploration of Computations with a Cellular-Automata-Based Artificial Life World

Submitted for the degree of D. Phil.
February 23, 1995

Pedro Paulo Balbi de Oliveira

ABSTRACT

Aligned with the recent tendency towards the conception of computational systems gleaned from evolutionary biology, a cellular-automata-based artificial life world is presented.

The aim of the thesis is the definition of the architecture of the system, named *Enact*, its implementation, and its exploration as a computing machinery. More precisely, the following connected issues provide the motivation for the research: how can an artificial-life world at an organismic level be defined in cellular automata, such that it can be viewed as a computational machine, when viewed from an external standpoint; and what kinds of models of computation are suggested by it.

Enact is a family of two-dimensional, non-deterministic cellular automata, whose temporal evolution can be described in terms of the metaphor of an artificial-life world where a population of agents undergo a coevolutionary process. During their lifetime, the agents move about, sexually reproducing, interacting with the environment, and being subjected to a developmental process.

An agent is formed by a sequence of contiguous cells, so that the cells at each end can be thought of as its head and tail, whereas the cells in between constitute its body. A single cell of the body forms the agent's genotype. Another cell, whose initial state depends on the agent's gene, represents the phenotype. The remaining cells of the body are fully determined through direct parental inheritance, constituting what we call the agent's memetype.

Enact can be regarded as a programmable, virtual machine defined by the artificial-life processes it supports, and relying upon six categories of states which represent environment, the agent's terminals, genotype, phenotype, memetype, and an additional category to allow the agents to move. It is using these high-level concepts that computations are addressed herein.

Fundamental to the system is that all events defined in it are coupled to each other through the movement of the agents.

Here we fully present Enact's architecture, and describe its possibilities; explain how Enact can be programmed; discuss conceptual issues underlying its use and design; identify and implement a model of computation in the system that relies, at least in principle, on all artificial-life processes embedded in it; show that this model of computation is in fact a simple case of a model of coupled computations, also embedded in the system; generalise the former model of coupled computations, and briefly mention a particular world set-up in which a number of computable functions could be observed as coming out of the coupling process; and finally, point at practical problems of going about the latter with the current system as it stands, setting out the requirements that its successor should have, in order to be an effective generalisation of Enact from the architectural point of view, while still preserving the conceptual underpinnings that led to its inception.

PREFACE

Drivers and Balloonists

A position one could find oneself in when starting a thesis, is to have a clear, well-defined problem *in front* of them that would provide the direction of the journey. This situation is more like riding a vehicle on a road, as unknown and unexplored as it may be. This journey is usually crammed with roundabouts, no-way-throughs, shortcuts, crossroads, whatever, and no matter the problems one may have, there will often be a safe parking area that can be quickly reached. In the worst cases, it is always possible to stop wherever one is.

An image that is probably adequate to convey the feeling I have had during these years is that of a balloonist. Unlike the driver, the constraints are less well-defined and the journey can flow in a freer way. But it is riskier too. If attention is driven away from the internal control or from the wind, returning may become more and more difficult. Driving is controlling. But when ballooning, this action has to be taken very seriously, as there is no other way to carry on: no resting areas or sudden stops are generally possible, and the wind, after all, may always blow.

I wish I could have been a driver in this thesis. But I was more like a balloonist.

Instead of a well-defined problem I had to face from the beginning, that would strongly constrain my journey, what I had was a strong motivation impelling me forwards. My struggle, even anguish in various moments, was to keep control of it so that I could reach a safe place at the end. Rather than driving into a region that I would have to find my way through, I felt more as if I was exploring an area in which I had found myself. Rather than answering questions, I felt myself asking them. This is not what I would have wanted, but it is how it came to be.

This thesis is not about a problem. It is about a motivation. And how it materialised in the form of a small piece of scientific research.

Chapter 1

INTRODUCTION: GUIDELINE FOR THE THESIS

This chapter is intended to be a general guideline for the thesis.

The attempt was to provide a clear picture of the aim of the thesis; the steps that were taken towards reaching it; the results that were obtained; the claims to be made; the content of the subsequent chapters; and the publications that were derived from the research reported herein.

1.1 What the Thesis is About

This thesis is about Enact, a family of cellular automata whose dynamics can be described in terms of the metaphor of an artificial-life world where a population of agents undergo a coevolutionary process. During their lifetime, the agents move about, sexually reproduce, interact with their environment, and are subjected to a developmental process.

The aim of the thesis is the definition of the architecture of the system and its implementation, as well as its exploration as a computing machine.

More precisely, the following connected issues provide the motivation for the research:

- how can an artificial-life world at an organismic level be defined in cellular automata terms, such that it can be viewed, from an external standpoint, as a computational machine; and
- what kinds of models of computation are suggested by it.

In the next chapter we will discuss how this thesis fits into the larger picture of artificial life and cellular automata research.

1.2 Steps to Be Taken

In the order they appear, the steps we are going to take throughout the thesis are the following:

1. To present Enact's architecture, its implementation, and to describe its possibilities.
2. To show that the system can be viewed as a programmable virtual machine, and explain how to go about using it.
3. To identify and implement a model of computation in the system that relies, at least in principle, on all artificial-life processes embedded in it.
4. To show that this model of computation is in fact a simple case of a model of coupled computations, also embedded in the system.

5. To generalise the former model of coupled computations, and to hint at a particular world set-up in which a number of computable functions could be observed coming out of the coupling process.

6. To point at practical problems of going about the latter with the current system as it stands, and to set out the requirements that its successor should have, in order to be an effective generalisation of Enact from the architectural point of view, while still preserving its conceptual underpinnings.

1.2.1 Outline of the Chapters

The content of the following chapters are as follows:

- Chapter 2 provides background knowledge for the thesis identifying the territory in which this journey will take take place, as well as the borders which will constrain it. It is aimed at rendering the thesis self-contained, and also to point out the relationships between the research carried out in the thesis with other pieces of work in the field. The main topics presented include a general identification of the field of artificial life; a characterisation of evolutionary and emergent computation; the definition of cellular automata, and aspects of their relation with computations.

- Chapter 3 is a full description of Enact. Enact is a family of two-dimensional, non-deterministic cellular automata, whose temporal evolution can be described in terms of the metaphor of an artificial-life world where a population of agents undergo a coevolutionary process. During their lifetime, the agents roam about, sexually reproducing, interacting with the environment, and being subjected to a developmental process which includes ageing and death. The overall qualitative dynamics of the system as well as the aspects involved in its underlying evolutionary activity are also discussed.

- Having described Enact in detail, Chapter 4 goes on to exemplify how the system can be used as a programming environment. Since the basic dynamics of the system is guaranteed by the underlying rule of the cellular automata, as long as it is not disrupted, all the artificial-life processes will be preserved. The idea of programming the system, therefore, means the establishment of a particular world set-up for the underlying artificial life activity. The example discussed is the implementation of a Turing machine, where the tape is implemented as a sequence of contiguous environmental cells, and the machine head as an agent. In doing so, the issue of embedding computations in Enact is introduced.

- Expanding on the Turing machine implemented in Chapter 4, Chapter 5 rebuilds it, but now relying on the entire population of agents. The intent is to show how the entire artificial life activity of the system could be considered as a computational machine, when viewed from an external standpoint. In doing so, the models of computation implicit in Enact are identified, the main one being fully described in the chapter. It is then suggested that models of coupled computations can be developed out of them.

- With that suggestion in mind, Chapter 6 describes what these models of coupled computation would be. Basically they are defined in terms of a set of Turing machines that share with each other one of their components (the tape symbols; their internal states; or their state transition table). Then, the chapter discusses in more

detail the model in which the state transition table is the shared component among the various machines. It is argued that this model, if conveniently constrained, provides a way to address the issue of coupled computations in the context of Enact's coevolutionary activity, and also that it opens the possibility of addressing the issue of criticality phenomena in constrained spaces of computable functions. In this respect Chapter 6 then briefly sketches a particular Enact set-up within which those possibilities might be realised, which has a simple definition but is sufficiently rich in terms of the space of computable functions that it entails; this set-up also serves to introduce a generalisation of Enact's main model of coupled computation. All the issues related to the mentioned set-up should be seen as preliminary ideas related to future work to be done, but even their partial presentation is useful to clarify various aspects of the issue of coupled computations in the context of Enact.

- Finally, Chapter 7 is an evaluation of what has been done throughout the thesis, as well as prospective in terms of what its achievements are pointing at. In particular it highlights what has been achieved both in practical and conceptual terms; points at practical problems with Enact as it stands; and provides a generalised definition of the system that is currently being undertaken. It then concludes the thesis with a personal statement on the historical pathways that led me to the research reported herein.

1.3 Contributions of the Thesis

1.3.1 Results

The way we explored the research theme of this thesis was by adopting an engineering standpoint. That is, although Enact is inspired by them, it is not a model of the biological notions it relies on.

In keeping with that, the following results have been established:

- The architecture of Enact itself, insofar as it is a rather complete artificial life world at the organismic level, fully couched in cellular automata terms.
- The programmability of the system, and the way to go about it.
- The identification of a model of computation that is couched in terms of the high-level artificial-life processes embedded in Enact.
- The exploration of the role of an explicit notion of space in the provision of coupling between computations.

1.3.2 Claim

Further to those concrete results, I will argue that

- by conveniently constraining the process of coupled computations, Enact may prove to be a useful tool to address the issue of coupled computations in the context of its coevolutionary activity; and also that it opens the possibility of addressing issues such as criticality phenomena in these constrained spaces of computable functions.

1.4 Publications and General Dissemination of the Work

Most of the core material in this thesis has been published in one form or another. This section provides further details, also mentioning other forms of exposition of the work such as talks that have been given.

Based on my research proposal outline I gave a talk in the Students Session of the *International Conference on Evolving Knowledge in Natural Science and Artificial Intelligence*, organised by the British Society for the Philosophy of Science, in Reading, UK, 1989. None of the student papers appeared in the proceedings volume.

In a slightly modified version, my contribution to the previous conference, [de Oliveira 1989], was presented as a poster in the second *Artificial Life Workshop*, held in Santa Fe, NM, USA, 1990; but since the work was still in a premature stage for publication, the paper was not submitted for the workshop proceedings. I then wrote a report on the workshop which was published as [de Oliveira 1990b] in *AISB Quarterly: the Newsletter of the Society for Artificial Intelligence and Simulation of Behaviour*.

I had completed a precursor of Enact when I was accepted for the *International Summer School in Complex Systems* organised by and held in the Santa Fe Institute, NM, USA. As a result of it, a paper was written and published in a book that came out of the event. The paper – [de Oliveira 1992a] – discussed the first ideas of the system, still unamed at that time, its main aspects having been included in Chapter 3 of this thesis.

The second version of the system was published as the technical report [de Oliveira 1992c], together with what became the content of Chapter 4. This work was first presented in the beginning of 1992 in a talk at the *British Computing Society Workshop on Cellular Automata*, held in London; this event did not have a full proceedings volume. Later on, it was accepted for *Complex Systems 92*, held in Australia, and published in the book that came out of it; that paper – [de Oliveira 1993] – is an abridged version of the original, and contains the essence of Chapter 4.

The third version of the system, then named Enact, was first published in the technical report [de Oliveira 1992b], and later on accepted for a poster presentation and demonstration at the third *Artificial Life Workshop*, Santa Fe, USA, but not included in the proceedings. This paper was rather long, and recently it was rewritten and split into two disjoint papers.

The first part, the description of, and discussion on Enact was accepted for the conference *Cellular Automata in Research and Industry*, held in Italy; the paper that appeared in the proceedings – [de Oliveira 1994a] – draws from Chapters 3 and 7. The second part appeared as [de Oliveira 1994c] in the proceedings of the *International Conference On Evolutionary Computation: Parallel Problem Solving from Nature, 3*, held in Israel.

Yet another paper, [de Oliveira 1994b], which is essentially Chapter 6, was published in the proceedings of the third *Workshop on Physics and Computation*, held in Dallas, USA.

And finally, a version of Chapter 5 was accepted for publication as [de Oliveira 1995] in the periodical *BioSystems: Journal of Biological and Information Processing Sciences*.

CHAPTER 2

MAPPING OUT THE TERRRITORY

As my interests became more and more focussed it became clear they were leading me to the emerging new discipline of *artificial life*, *Alife* for short.

What follows is a description of this territory, as well as of its borders that our journey will be constrained with. In addition to providing background material in order to render the thesis self-contained, the aim of this chapter is, wherever possible, to provide links from specific points of the thesis to related pieces of research found in the literature, and also to subsequent parts of the thesis where the topic at issue will be addressed in more detail. It also aims at fitting the thesis into the larger picture provided by artificial life and cellular automata research.

2.1 The Territory: Artificial Life

Ever since Langton [1986] used the term artificial life for the first time in the conference *Evolution, Games and Learning: Models for Adaptation in Machines and Nature*, [Farmer *et al.* 1986], the momentum associated with the discipline has steadily grown. As this happened two other events also had a seminal role, [Langton 1989] and [Forrest 1990]. But it was only Alife-II, the second *Artificial Life Workshop* at the beginning of 1990, that really marked the consolidation of the field; see [de Oliveira 1990b] for a report on this most exciting event.

A widely cited expression by Langton (as in [Langton 1992a]) states that the aim of the enterprise is to push current knowledge from *life-as-we-know-it* to *life-as-it-could-be*. And for this matter, artificial life is in the confluence of various disciplines such as theory of computation, artificial intelligence, physics and mathematics of nonlinear systems, and theoretical biology. As far as AI and cognitive science are concerned Alife brings with it a fierce criticism of classical cognitivism, by getting closer to biological aspects that have systematically been left aside or relegated to the background. In this respect it is worth remarking that, to some extent, Alife represents a rescue of old insights from the days of cybernetics, but now with more adequate tools for the enterprise.

Artificial life is a direct consequence of the great theoretical developments that research in *complex systems* underwent during the eighties, as well as the availability of computational power to support their simulations. These are systems that are characterised fundamentally by an architecture of many components with however local interactions between them. Even though this local activity can be very simple, the overall behaviour of the system can be very complex. This is the case, for instance, of computational entities like neural nets, cellular automata, and classifier systems, and also of systems like spin-glasses, societies, and economies. [Serra and Zanarini 1990] and [Weisbuch 1991] provide two excellent accounts of complex systems from a more computational orientation.

There have been various international meetings on artificial-life-related topics. Most notably, the *Workshops on Artificial Life*, of which Alife-II was the second edition, has taken place in USA every other year since 1987 ([Langton 1989]). It alternates with the *European Conference on Artificial Life*, since its inception in 1991 ([Varela and Bourgine 1992]). Another important regular conference is *Simulation of Adaptive Behaviour*, that has taken place biennially since 1990 ([Meyer and Wilson 1991]).

More specific workshops have also appeared such as the *Workshop on Physics and Computation*, which in its current – third – edition has become regular, but after the first one took place in 1981 ([PhysComp-81 1982]); and the *Workshop on Perception and Action*, which has just happened and very likely will have follow-ups. Various events involving cellular automata (such as the workshop [CSC 1991]) have also found a new thrust. The fact is that, in many countries more and more events have been organised around Alife-related topics, from summer schools to special sessions and tracks in the major international conferences (in control, for instance).

Various journals have also been created in the post Alife-II period in order to be partly or fully devoted to artificial-life-related issues. These are [Meyer 1993], [Langton 1994], [Jong 1994], and [Morowitz 1994].

Also, traditional journals have opened space for Alife, in particular the ones that focus on AI and cognitive science. In this respect, it is worth mentioning [Cliff 1994], a special-theme issue of *AISBQ*, the newsletter of the Society for Artificial Intelligence and the Simulation of Behaviour; [Agre and Rosenchein 1993], a special issue of the *Artificial Intelligence* journal; and [Huberman 1994], a forthcoming special issue of the latter journal on nothing less than phase transitions, an issue that has very often appeared within Alife (as in [Langton 1990]).

General presentations on artificial life abound. Langton's various discussions, such as [Langton 1992a], are mandatory; [Belew 1991] emphasises its relations with artificial intelligence; [Mikhailov 1992] follows an engineering-oriented perspective; and [Levy 1992] provides a popular presentation, with an insider's view not only of research but also of the researchers themselves, mainly the ones from the *mecca* of the field, the Santa Fe Institute, Santa Fe, USA.

Various pieces of work emerged in the context of artificial life that bear relevance to artificial intelligence and cognitive science. In addition to the fully embodied approaches to cognition based on autonomous robots (as in [Brooks 1991b] and [Brooks 1991a]), it is also worth mentioning the emphasis on embodied approaches even if with simple animals and in simulated settings, as discussed in [Cliff 1991]; and the views of cognition linked to dynamical processes, as developed in [Beer 1992] and [van Gelder 1992].

Having established artificial life as the main territory of this thesis, what follows is a set of boundaries within which our journey will be constrained with.

2.2 Evolutionary Computation

Among the artificial life topics, *evolutionary computation* is certainly the most well-known; in fact, conferences on the issue have been around for nearly ten years already. The *International Conference on Genetic Algorithms* is the oldest of them, having been taken place every two years since 1985 ([Grefenstette 1985]), always in the USA. The *Parallel Problem Solving from Nature* conference has always been more general than the former; for instance, it was in its first edition, [Schwefel and Männer 1991], that the mostly American – at least up to that moment – genetic algorithms community, first met the evolutionary strategies Germany-based community. Since then, the two have taken place in alternate years. Completing the cycle on evolutionary computation there has also been since 1992

([Fogel and Atmar 1992]) the annual, so far USA-based, *International Conference on Evolutionary Programming*.

The three techniques mentioned above – genetic algorithms (GA), evolution strategies (ES), and evolutionary programming (EP) – are the main ones currently in use, although variations do exist. The technique of genetic programming ([Koza 1990]) is also worthy of mention. [Goldberg 1989] is still the most accessible entry point to the field of genetic algorithms. The research pespectives in genetic algorithm as perceived in [De Jong 1985] are still very up-to-date, mainly if compared with an assessment of the field written nowadays, as in [De Jong and Spears 1993]. Of particular relevance to the artificial life community is the review presented in [Mitchell and Forrest 1993], and also the work on variable-length genotypes presented in [Harvey 1994]. [Fogel 1992] traces back the history of evolutionary computation, specifically from the perspective of evolutionary programming. [Hoffmeister and Bäck 1991] is a convenient introduction to evolutionary strategies, insofar as it is made by comparing it with genetic algorithms. [De Jong and Spears 1993] is also adequate to provide a unifying view on the different techniques of evolutionary computation.

Basically they are search techniques in problem spaces gleaned from (an abstraction of) evolutionary genetics. In all of them the search starts with a population of candidate solutions that are generated by a random process. This population is then evaluated in regard to their proximity to the expected solution of the problem at issue. Based on this *evaluation*, a *selection* process is then carried out that picks out a subset of the population, so as to form the basis upon which a new population will be created. The latter is achieved by applying *genetic operators* to the pool of selected individuals, one of them being sexual *reproduction* between pairs of individuals. The new population – which is expected to be formed by a better set of candidate solutions than the former – then replaces the original, and the process iterates.

The distinction between the three approaches is mostly due to the different emphasis on the role and usage of the genetic operators. So, while in GA the most important operator is *crossover* – that creates two individuals out of two others, by exchanging segments between the latter – in EP and ES *mutations* in the individuals have the primary role; in fact, crossovers are hardly used at all. Also, while in ES the mutation rate is adaptive, this is typically not the case in GA and EP. However, as [De Jong and Spears 1993] has recently discussed, these differences are mostly historical; as a coherent theory of the field progresses, these differences have become fuzzier.

It is worth distinguishing two usages of evolutionary computation techniques: as an optimisation technique, which is the most widespread form; and as a computational model for evolutionary studies in natural and artificial systems. For the purposes of this thesis it is worth bearing in mind that it is the latter usage that will be of interest. In fact, the evolutionary facet associated with the system to be presented herein will explore an aspect of evolution more related to viability than to optimisation; this will be discussed in Section 3.10.

Given the importance of coevolution in nature and the potential of coevolutionary models in practical applications, there are still relatively few references in this topic. Key pieces of work include [Hillis 1992], where two processes coevolve, one being a sorting algorithm, and the other, a parasite-like process that creates test-cases for the latter, at the end of which a new sorting algorithm is discovered; and [Husbands 1993] where a real-world application is developed for the problem of job-shop scheduling.

Another alley also yet to be explored in greater detail is the use of these techniques in conjunction with cellular automata, as is the case of [Lipsitch 1991].

One particular class of applications of evolutionary computation that has been a matter

of huge attention in the last few years is their use to evolve neural networks. Various surveys exist in this area, [Yao 1992] being a recent one.

2.3 Emergent Computation

A topic that will be particularly relevant in this thesis is what has been denoted *emergent computation*, after a workshop on the topic, which was essential to gather momentum for artificial life; its proceedings were published as [Forrest 1990]. The point here is the characterisation of the global behaviour of a complex system in terms of information processing. Many complex systems can be described in such a way, for instance, a neural network, which, after having gone through a learning period, may become capable of performing a computation; putting it in another way, its behaviour may be described in terms of a computation that is being performed.

[Forrest 1990] presents a number of contexts in which the issue of emergent computations is approached, from number-theoretic accounts, to the behaviour of networks of logical gates.

The form of emergent computation that will be of interest in this thesis is the one associated with cellular automata. That is, the characterisation of their global behaviour in terms of computations. As will be discussed in Chapter 6, the aspect of emergent computations we will be addressing is the one that can be viewed as a process of *coupled computations*.

2.3.1 Coupled Computations

By coupled computations I mean a number of computational processes occurring in a way that the steps of computation of each one interfere with the steps of the others. The interest in this process is in its facet of self-organisation and emergence. Unlike a standard process of parallel computation, where the focus of the process is the achievement of a predefined computation, in coupled computations the interest is the process itself. Naturally, the question of how to constrain the process such that a predefined computation can be performed is certainly an issue. But the general concern extends beyond that, so as to also include such question as: how is it possible to drive the process to converge to one computation or another, with the same basic architecture; which kinds of coupling processes admit reversible emerged computations; issues on the robustness of the coupling schemes, and on the differences between the coupling schemes themselves; etc.

In order to appreciate better our approach to coupled computations it is worth reviewing related approaches to the issue, which will be referred to again in later chapters.

2.3.2 From the Turing Gas to Turing Machines

One type of coupling scheme is obtained when the unit of coupling is a computable function defined an abstract space. An example of this type is the so-called Turing gas, as defined in [Fontana 1992]. In this system a population of particles are subjected to pairwise collisions with the possibility of formation of new ones which, in turn, enter the chain of already existing reactions. The Turing gas is a system of coupled computations because the particles are functions coded in a variant of pure-Lisp called *AlChemy* (a shorthand for "Algorithmic Chemistry"), the collisions between the particles being the evaluation of one of the functions having the second as the argument. The Turing gas has been used as a model of systems that have an inherent "constructive dynamics", that is, whose components act on each other constructing new ones which themselves have the

ability to take part in the constructive process; a paradigmatic example of this kind of system are chains of molecular reactions.

Another kind of system of coupled computations is the one based on coupled executions of an assembly-like language that runs in a (typically) virtual machine. [Ray 1992] and [Rasmussen *et al.* 1990] are landmark examples of this kind; [Rasmussen *et al.* 1992] is also important, although this work goes beyond the particular coupling scheme currently at focus.[1] It is worth noting the fact that they (indeed, like [Fontana 1992]), tackle the issue of coupled computations without having, however, the need to explicitly recognise it.

[Ray 1992] features *Tierra*, an artificial life world where a population of programs compete for memory space and CPU time of their host machine. The programs are subjected to an evolutionary process so that the ones that manage to replicate more, get more of these resources, thus guaranteeing their survival. In particular, there are situations in which individuals manage to run instructions that belong to another individual to their own benefit or harm. Therefore, coupled computations in Tierra occur when an area of memory contains instructions that belong to an organism, but are shared by other individuals due to their own individual nature.

The systems *Venus I* and *II*, and *Luna*, discussed in [Rasmussen *et al.* 1990] and [Rasmussen *et al.* 1992], all implement variants of the idea of a "soup" of instructions spread over an area of memory, where a population of program counters coexist executing the shared code. In terms of coupled computations these systems therefore follow the same approach as Tierra, which is the sharing of instructions among the different computing processes.

A lower-level approach to coupled computations is the one where the unit of coupling are Turing machines. A reference along this line is [McCaskil 1989] (presumably an early inspiration for the Turing gas). This work uses interacting TMs for studying functional self-organisation, and is based on a binary string encoding of their transition tables.

Recently Rucker (1993) released a software package and accompanying book which, in spite of simply aiming at being an entertainment artificial-life system, provides an interesting example of coupled computations. This alife world is inhabited by a population of two-dimensional Turing machines whose individual transition tables are coded in the organisms' genotypes. The organisms' environment is seen as a two-dimensional tape that is shared among all individuals. As they move about they leave trails that can be followed by the others. The symbols that make up the trails are, therefore, the symbols that are written on the tape which, when read by other organisms, provide the effective coupling among the computations individually defined in each Turing machine.

As far as Enact's approach to coupled computations is concerned, the best way to describe it is by thinking of the state configuration of the agents as representing the tape and the internal states of a Turing machine, and their environment as providing the locus in space where one step of a computation will take place. From this perspective, the state transition rule of the machine becomes part of the "physics" of the world, such that the computation is materialised at the points in space where an agent interacts with its environment. As will be made clear later, it is the movement of the agents that determines the outcome of an individual computation; but since movement directly depends on the availability of space, it is clear that space is the ultimate determinant of the coupling process.

This approach will be discussed in detail in Chapter 6, in particular pointing out

[1] Its theme is the more encompassing concept of dynamics of "self-programmable matter"; although not explicitly recognised in the paper, it essentially corresponds to the notion of constructive dynamics defined in [Fontana 1992].

advantages it suggests in respect to the systems described above. It is worth advancing, however, that the advantages of the current approach will be argued but, for practical reasons, will not be explored in a running set-up within the thesis; this will be explained in Chapter 7. Finally, a natural generalisation of the Turing-machine-based approach will then be made, that equates computations to the developmental processes undergone by the agents.

Hence, in respect to coupled computations the contribution of this thesis will be the definition of a modality of coupled computations that is entrenched in an artificial life activity, and where the coupling medium of the computations is the space provided by the cellular array of Enact's underlying cellular automata.

2.4 Enaction

Enaction is a research programme in cognitive science identified in [Varela 1989], and further extended in [Varela *et al.* 1991]; see also [Dennett 1992] for a review of the latter monograph.

The characterisation of enaction is built upon the notion of *embodied cognition*, whose major consequence is the decrease of emphasis on the role of representations in the understanding of cognitive phenomena. Enaction's alternative to representation is "embodiment", that is, the history of dynamically coupled interactions of active agents in their worlds. According to the enactive view, cognitive structures emerge, through lifetimes and lineages, bringing forth a significance that only makes sense through the history of interactions, and ensuring the continued, ongoing activity of the cognitive system in its world.

To help understand the technical sense of enaction it is useful to look up the term "enact" in dictionaries of the English language; in doing so it is clear that there are two typical – rather distinct from each other – meanings of the verb *to enact*. For instance, in [Hornby 1989] the first connotation is identified as to

> "...perform (a part, play, etc.) on, or as if on, the stage of a theatre ...",

and the other as to

> "...make or pass (a decree) ..."

What the proponent of enaction had in mind, therefore, was a term with which it would be possible to characterise a view of the notion of representations in cognitive science that would rely on both connotations of the term at the same time. Namely, while the first connotation stresses the view that representations do not really exist, the second emphasises that they come into existence by a deliberate act (of interpretation).

When the notion was first put forward in [Varela 1989], apparently Varela was unaware of the work that has been carried out by Brooks (cited earlier) in the construction of autonomous robots. At the time [Varela *et al.* 1991] was published, he was aware of that research line to the extent that it was treated in the book as a paradigmatic realisation of the enactive stance. But it is worth mentioning that various other approaches to building autonomous mobile robot, such as the evolutionary approach described in [Cliff *et al.* 1993], may well be regarded as another form of realisation.

Not many pieces of research have been published addressing the topic, at least from my bias of more computational accounts; I should add, though, that I have not followed the more philosophical stream of publications. [Rutkowska 1990] is one of the few examples, where the importance of the enactive perspective for developmental psychology is

recognised. Following a workshop on autopoiesis held in Dublin at the end of 1992 there was some activity on the Internet, but not much recently. It is also worth mentioning that there is a substantial overlap between the communities interested in enaction with the one interested in autopoiesis – a concept that aims at characterising an organisational principle of the living entities (see [Maturana and Varela 1987] for instance) – although the latter has been more active. Anyway, complementing the practical aspects of enaction that Brook's work epitomises (at least from Varela's viewpoint), for an in-depth account of the philosophical issues that come out of enaction, and their analysis within the context of a particular theme in cognitive science, namely, colour vision in different animals, see [Thompson *et al.* 1991].

The implicit reference to enaction that Enact carries in its name reflects an acknowledgement to enaction as an "umbrella" that encompasses various concepts which Enact also attempts to emphasize, in particular the role of self-organisation. It also represents a personal recognition to the fact that the first time I became aware of those concepts, in a coherent way, was in the context of enaction.

In particular, it is a recognition to the biological roots of enaction, epitomized by the concept of *evolution as natural drift*, the view of biological evolution that was developed in an interwined fashion with enaction, as put forward in [Maturana and Varela 1987]. From a very general perspective, evolution as natural drift could well be summed up as the view of evolution that also recognises self-organisation as another major component in biological evolution, thus stressing the necessity of going beyond the predominant view that natural selection provides. At least from this macro perspective this view is very similar to the one Kauffman has put forward in [Kauffman 1991], which was epitomized by his monograph [Kauffman 1993].

In the context of the latter topic, it is relevant to mention the biological notion of *exaptation*, a term proposed in [Gould and Vrba 1982] to refer to structures whose function did not arise as a consequence of progressive adaptations via natural selection.[2] Even though the link between the two concepts – enaction and exaptation – had apparently not yet been recognised, they seem to me very much related in their stress on viability (and self-organisation) rather than optimisation as a primary aspect of biological evolution.

Punctual references to enaction occur throughout the thesis, but the reader should be warned not to be misled by considering Enact as a system that explores the deep cognitive issues implied by enaction in any wide sense. In fact, this thesis can, and in fact, should be read from a purely cellular automata standpoint.

2.5 Cellular Automata

2.5.1 General

Cellular automata (CA) are arrays of finite-state machines that can also be described as discrete dynamical systems ([Toffoli and Margolus 1987]). In the terminology we will use here, they are made up of a set of *cells*, which are organized in a regular n-dimensional lattice, the *cellular space*, or simply, *the* lattice or array. At any time, each cell can take

[2] Incidentally, a debate involving exaptationist versus adaptationist explanations has started and is still very active; for example, [Pinker and Bloom 1990] features a recent debate in the context of the evolution of human language, with both sides fiercely represented. The predominance of the adaptationist programme is precisely the target of the criticisms expressed not only in [Gould and Vrba 1982], but also in [Gould and Lewontin 1984], [Lewontin 1984] and [Piatelli-Palmarini 1989]. As remarked in [Lewontin 1989, page 107], it is not that alternatives are not mentioned, but that they are all considered "diversions from the big event, the ascent of Mount Fitness". Quite significantly, in the list of 172 references shown in the remarkable review of the field of simulation of adaptive behaviour presented in [Meyer and Guillot 1991], none of the previous references related to the adaptationist debate can be found.

on one among a set of discrete values, which are the cell *states*. The states of all the cells in the lattice are updated (typically) synchronously, the new state of each cell being dependent upon the state of its *local neighbourhood*, i.e., its current state together with the states of a group of neighbouring cells. The updating of each cell state is achieved by applying to the cell neighbourhood a set of deterministic or non-deterministic *state transitions* which together, define the *rule* of the automaton.

The activity of cellular automata (CA) often takes place over an "inert" background, sometimes called *quiescent*; in this case the state that characterizes the background becomes the quiescent state. Very often the cellular array is wrapped around on itself which, in the case of two-dimensional CA yields a torus type of array; this kind of boundary condition is referred to as periodic background. Otherwise it is referred to as blank background. Typical neighbourhoods for two-dimensional CA involve a cell and its eight surrounding neighbours (the Moore neighbourhood), or the four surrounding neighbours in a diagonal cross (the von Neumann neighbourhood). Enact uses the former. Variations of cellular automata do exist in a number of themes like asynchronous updates of the cell states; inhomogeneous rules (that is, different state transitions for different groups of cells); variable neighbourhoods; etc. Because they are not relevant for present purposes they will not be reviewed here, but the reader is referred to [Toffoli and Margolus 1987] for an excellent introduction to these and many other issues related to the phenomenology of cellular automata.

McIntosh [1990a] partitions the history of cellular automata in two eras. The first is the *von Neumann era*, which started when Burks, after von Neumann's death, edited and published in [von Neumann 1966] the details of his famous cellular automaton that was capable of self-replication. This era continued up to 1970, when the most famous cellular automaton, the *game-of-life*, was published in Martin Gardner's column in *Scientific American*, thus initiating the *Gardner era*.[3] To these we could well add in sequence the *Wolfram era*, after Wolfram's seminal 1983 paper (republished as [Wolfram 1986a]), where he undertook a computer-based search through the properties of the elementary automata, guided by some concepts from the realm of nonlinear dynamics and statistical mechanics.

[Gutowitz 1991] is still the most comprehensive account of cellular automata, both for theoretical aspects and applications in various domains. And [Gutowitz 1994] is the most up-to-date version of the "frequently-asked-questions" file of the CA Internet-based mailing list, an extremely valuable compilation of pointers to all aspects of CA.

Because cellular automata are dynamical systems and computing devices, the interplay between their dynamical and computational capabilities makes them a particularly appealing conceptual framework for artificial life realisations. For instance, it is known that there are great difficulties in analysing the dynamical behaviour of conceptual frameworks related to artificial-life (see [Forrest and Miller 1990] in respect to classifier systems and [Vose 1991] in respect to genetic algorithms). But because there is a relatively longer tradition in that issue in the context of cellular automata, in general they may be more tractable in this respect. Also, their established formal status rules out the need for *ad hoc* dynamics that, in additional to being arbitrarily defined, might render its analysis too difficult.

Having made general comments on CA, in the next subsections I will address specific topics that bear a more direct relevance for this thesis.

[3] The game-of-life is a binary cellular automaton – let us call them *alive* and *dead* – with Moore neighbourhood. Its definition is the following. If a cell is dead but is surrounded by 2 or 3 cells which are alive, then it becomes alive in the next iteration. If the cell is alive and has exactly 3 surrounding cells which are alive, it remains alive in the next iteration. Otherwise, it becomes dead.

2.5.2 Sexual and Self-Reproduction in Cellular Automata

The important role of sexual reproduction in the provision of variability in nature, and the fact that the most important evolutionary computation techniques rely upon sexual reproduction, motivated the introduction of such a feature also in the context of Enact.

Although a number of cellular automata exhibiting the ability of self-reproduction have been discovered – for instance, [von Neumann 1966], [Codd 1968], [Banks 1971], [Langton 1984] and [Byl 1989] – no cellular automaton embedding a form of sexual reproduction has apparently been reported, other than the one in this thesis. The closest reference in the literature seems to be [Vitanyi 1973] where an abstract discussion is carried out on how to extend the cellular automaton described in [von Neumann 1966] so as to allow sexual reproduction. The complexity of the automaton, however, renders it completely impractical for present purposes, and, in fact, neither of them have ever been implemented.

On the other hand, if one wishes to create self-reproducing automata it is also important to prevent them from exhibiting a *trivial* self-reproduction. In this respect the point made in [Langton 1984] is relevant. A two-state cellular automaton – that performs addition modulo 2 – described therein exhibits self-reproduction, but is overly trivial since it can be fully described at the level of the automaton's underlying physics. That is, its self-reproduction cannot be described in terms of any higher level process that would be identifiable in its dynamics.

The first five rows of Table 2.1 present a collection of the most seminal cellular automata exhibiting the ability of self-reproduction. All of them are two-dimensional and use von Neumann neighbourhood, as defined earlier. The partial or uncertain pieces of information contained in the table are due to the respective references not providing them in a precise and unambiguous way. The highlighted row at the bottom of the table contrasts the sexual reproduction process embedded in Enact with the equivalent features of the self-reproducing automata. The universal constructability that is mentioned therein refers to the feature of a cellular automaton being able to construct in its array any state configuration that is given to it; for automata that have this feature, self-reproduction can be achieved in the special situation in which the input description for the cellular automaton is its own initial configuration of states. The last two columns of Table 2.1 refer to the imposition that the automaton at issue should have the feature stated in the column.

It is clear that their complexity – as expressed by the number of cells in the initial configuration, and by the number of possible states per cell – vary significantly and depend on the design constraints imposed on the automaton.[4] So, the imposition of both universal computability and constructability imply extremely complex automata. By relaxing these constraints much simpler cellular automata can be created, as the table shows in the fourth and fifth rows.

Another aspect worthy of comparison is the number of initial configurations which still supports sexual or self-reproduction. In [Byl 1989], for instance, various initial configurations are possible, while in [Langton 1984] only one can self-reproduce. In Enact, however, the number of possible initial configurations is only bounded by the finiteness of the cellular space; the only requirement is that the agents involved be well-formed. Similarly, so it seems that a large number also applies to von Neumann's automaton, with the difference that, in this case, each individual initial configuration has to be carefully crafted.

The initial size configuration shown in the table refers to the number of cells of the

[4] The neighbourhood size could also be a measure of complexity, but is not relevant for present purposes since they are the same for all cellular automata shown in the first five rows of the table.

Authors	Year	Number of States	Size of Initial Configuration	Universal Computability	Universal Constructability
von Neumann	≈1952	29	$\approx 10^4$	yes	yes
Codd	1968	8	$\approx 10^4$	yes	yes
Banks	1971	4	?	yes	no
Langton	1984	8	86	no	no
Byl	1989	6	11	no	no
Enact	1992	4	6	yes	no

Table 2.1: Cellular automata with ability for self-reproduction, in a comparison with the sexual reproduction process embedded in Enact.

cellular space that require initially special values. In the case of Enact, the value expressed refers to two 3-cell-long agents, since this is the requirement for a pair of minimally well-formed agents.[5]

The standpoint adopted in Enact is one of achieving sexual-reproduction, with a significant degree of complexity, that would allow the satisfaction of constraints such as the necessity of a mating configuration for the parental agents; the requirement that agents of any size can reproduce; necessity of coping with the movement of the parents and of the offspring as reproduction takes place; among others. More details will follow in the next chapter.

2.5.3 Universal Computability in Cellular Automata

Here we review the general issue of universal computability in cellular automata. Table 2.2 presents a collection of cellular automata capable of universal computation. Some of the cellular automata mentioned therein have already been shown in Table 2.1, since they also have the ability of self-reproduction. The table is not meant to be exhaustive, but to provide some historical landmarks on the topic.

The complexity of the automaton naturally depends on the constraints imposed on its definition. Compare, for instance, the two cellular automata discovered by Banks [1971]. Also, bearing in mind the simple measure of complexity defined by the multiplication of the three parameters that appear in the table, it is clear that, similarly to what we have seen in the case of self-reproduction ability, simpler cellular automata have been discovered over time. Noteworthy in this respect is [Lindgren and Nordahl 1990] which, implementing Minsky's [1967] Turing machine – with 4 tape symbols and 7 internal states – describes the simplest cellular automaton currently known that is capable of universal computation. It is also worth mentioning that due to the complexity of the automaton and the size of the required initial configuration, the actual implementation of its underlying universal computing device may become practically infeasible. In fact, as far as I am aware this is the case both for von Neumann's automaton and for the game-of-life, which

[5] At least when considering that there is no differentiation between the body-cells of the agents, as will be discussed in the next chapter.

Author	Year	Number of States	Neighbourhood Size	Dimension	Note
von Neumann	≈1952	29	5	2	
Codd	1968	8	5	2	
Smith III	1971	18	3	1	
Banks	1971	3	5	2	blank background
Banks	1971	2	5	2	periodic background
Berlekamp et al.	1982	2	9	2	game-of-life
Albert and Culik II	1987	14	3	1	
Lindgren and Nordhal	1990	7	3	1	simplest one currently known
Enact	1992	20	9	2	

Table 2.2: Some cellular automata capable of universal computation.

have been proved to be capable of universal computation but, due to practical difficulties, have never been actually implemented.

The data presented for Enact in the table assumed the implementation of the aforementioned Minsky's universal Turing machine, and draws from the machine we will implement in Chapter 4. It is worth advancing, however, that the relevance of this implementation is that it is couched in an artificial-life framework, which is novel; furthermore, it will provide us with the necessary ground for more sophisticated forms of computation that we will be dealing with in the subsequent chapters.

2.5.4 Forms of Computation in Cellular Automata

Computations appear in the cellular automata literature usually in two forms. The first form, which has been called *intrinsic* computation, is the one performed directly from the rule of the cellular automaton, the initial configuration of the cellular array being the input for the computation. As an example, imagine a one dimensional cellular automata with 2 states (say, 0 and 1) and 5 neighbours – i.e., the cell at issue and its 4 closer neighbours, 2 from each side, left and right – such that, for any initial configuration the following is observed: if the number of 1's is larger than the number of 0's, after a certain number of iterations all cells usually go to 1; otherwise, they usually go to 0. This is a rule that has been a matter of great attention (mainly after its use in [Packard 1988]) and has become known as Gacs-Kurdyumov-Levin rule (after their discoverers), or simply GKL's rule, for short. This is a clear case in which a non-trivial computation – whose usual characterisation needs global evaluation of the initial conditions – is performed by purely local means. Other CA that present some intrinsic computation capabilities have also been discovered for adding binary numbers, for multiplying them, and even, for enumerating the prime numbers expressed in binary form.

The other context in which computations appear in CA research is precisely the one we will be dealing with here. These, which we can refer to as *non-intrinsic* computations, occur at a higher-level than in the latter form. The role of the rule of the automaton

in non-intrinsic computation is the provision of a sustained dynamics *over* which the computation will be performed. For instance, in the case of Enact, this dynamics is precisely the artificial-life activity of the world, where we can identify high-level concepts such as agent, environment, and so on. The input of the computation in this form of computation is not the full initial configuration of the cellular array, as in GKL's rule, but only a part of it, like the state configuration of one particular agent (again, in the case of Enact). The way the game-of-life has been proved to be computation universal (in [Berlekamp *et al.* 1982]) follows exactly this idea, as does von Neumann's self-reproducing automaton, also mentioned. As far as I know, the work presented here was the first attempt to explore non-intrinsic CA-based computations in a systematic way.

2.5.5 Computations and Complex Dynamics

From Alife-II the notion of *edge-of-chaos* dynamics started to become a widely disseminated concept, mainly due to Langton's paper, [Langton 1992b] (even though he had made the same claims earlier in [Langton 1990]). This notion was first put forward in [Wolfram 1986a] in order to identify a special dynamical regime squashed between chaotic regimes and the ones characterized by fixed points and cycle limits. The interest in these *complex* dynamical regimes, as they are also denoted, lies in the fact that, as argued by Langton and Wolfram, they possess the necessary conditions for the emergence of information processing ability; that is, the possibility of transmitting information allowed by the fluidity of chaos, and the possibility of storing it due to the stability provided by ordered regimes.

Edge-of-chaos dynamics has been the focus of intense research in various aspects, as discussed in [Adami 1994], [Crutchfield 1991], [Crutchfield 1992], [Hanson and Crutchfield 1991], [Kanebo and Suzuki 1993], [Li and Nordahl 1992], and [McIntosh 1990b]. In particular, there is an intense activity going on at this moment on the relation between computation in cellular automata and complex dynamics; in particular a major reappraisal of results presented in [Packard 1988] involving GKL's rule are casting new light on the issue, as can be seen in [Crutchfield and Mitchell 1994], [Das *et al.* 1994], [Mitchell *et al.* 1993b], [Mitchell *et al.* 1993a], and [Mitchell *et al.* 1994].

Couching this notion in terms of cellular automata, complex dynamical regimes have been associated with their ability to perform non trivial computations, and being characterized by, among other properties, the existence of propagating structures[6]. When the design of Enact was still in its early stages I thought its dynamical behaviour would have to be "tuned" in the complex regime. As will be discussed in Chapter 3, this feature turned out to be a natural consequence in the system, the tuning process having become in fact the adjusting of a particular world set-up so as to prevent extinctions and deadlocks in the cellular space due to its over-population.

2.5.6 Parameterisations of Cellular Automata Rule Spaces

This subsection has a different connotation from the others. My intention here is to report on a piece of research that I started, very much linked to the topic that has just been presented, but which will not be included in the thesis itself. Due to the suplementary role it plays in the provision of background material on some of the relations between cellular automata and computations, it is undoubtedly pertinent to this chapter; however, its presence can only be fully justified for what it represents as a register of the research that was started, at least in the form of a short preliminary report on its status.

[6] Such as the so called *glider*, that appears in the game-of-life.

The relationships between dynamical behaviour of cellular automata and computations led me to question how it would be possible to have an estimate of the dynamical behaviour of a cellular automaton directly from its state transitions, without having to run it. In particular, how well the estimation of the complex dynamics would fit in the scheme.

It had been proved that the general answer to that question is undecidable. However, it would still be possible to come up with an estimate that could be helpful in some cellular automata rule spaces. In fact, the smallest non trivial rule space, the so-called "elementary space" defined by the one-dimensional CA with binary states and three neighbours has $2^{2^3} = 256$ rules, while the one immediately larger (with five neighbours) has $2^{2^5} > 4 \times 10^9$ rules; hence, any estimate on large spaces would be really helpful.

In most studies carried out in CA rule spaces parameters have been devised, either empirically or from a more formal point of view, that might help in establishing correlations between the definition of a particular automaton and its dynamical behaviour. This is the case of [Li 1989], [Li and Packard 1989], [Li 1991], and [Binder 1993], where the elementary space was extensively studied. From a different perspective [Wuensche and Lesser 1992] provided extensive data on basins of attraction in the elementary space, although the enumerative algorithm used therein can also be applied in higher spaces; additionally, parametric analyses were also performed in that piece of work.

Analyses of a non-local version of the elementary space have also been a matter of attention ([Li 1992] and [Wuensche 1994]).

Other studies have been made with the concern of scanning CA rule spaces in order to probe their properties, such as the existence of phase transitions, which has been a matter of great attention because of its supposed relation with computability of CA. For discussion on the latter, see, for instance, [Li *et al.* 1990]; for an interesting mechanism to scan CA rule spaces by imposing only small variations in the global behaviour of the CA (in a close to continuous way), see [Pedersen 1990].

The approach I pursued to the problem was to search the space of parameters whose definition would reflect some sort of local activity with the individual state transitions. The reference was the elementary space, where the classification of the rules I used split it in six classes of dynamical behaviour: null (rules that lead to a fully homogeneous end configuration), fixed-point, periodic with cycle 2, periodic with higher cycles, complex, and chaotic. The search was undertaken as an extensive empirical process in which I could define a parameter, and obtain an analysis of the degree of discrimination it would induce over the classes of dynamical behaviour of the space. For instance, one of the parameters I found was an excellent discriminator between null and chaotic rules; two others provided good discrimination between fixed-point rules and rules with cycles of length 2. The idea was then to search for a group of parameters that jointly could provide a good discrimination between the various dynamical behaviours, but bearing in mind that the identification CA with complex behaviour was the main target.

The parameter space I searched turned out to have some important members. Langton's λ parameter ([Langton 1990]), for instance, belonged to the space. And so did the Z-parameter of [Wuensche and Lesser 1992]; in fact, in personal contact with the first author, it came out that the parameter had been defined and started to be used independently by us at about the same time (although we had different interpretations for it).

As I realised that this "parameter-hunt" enterprise was pushing me much farther away from my main research than I thought I should go, I decided to stop it at the first halting point I could see ahead. That should be when I had found the group of parameters with a good joint discrimination power, as mentioned above; this was exactly what I did. I ended up with three parameters plus another that had two slight variations, thus yielding

in fact two very similar groups of four parameters. In either case, for any quadruple of their values, there was only one rule; in other words, in this 4-D space there was only one elementary rule in each valid coordinate.

The next steps in this research direction that would be worth taking would be the following. First, to find a way to visualise this 4D space, in order to figure out how do the various dynamical behaviours cluster, in particular the complex rules; 3D slicings with adequate colouring seem to be the natural way. Second, to select a set of rules from the space of one-dimensional CA with 2 states and 5 neighbours, work out the value of the group of parameters for them, and check whether their discrimination power is preserved in this larger space. In this respect, two sets of functions are promptly available: the totalist rules of the space – i.e., the ones whose state transitions depend only on the number of cells of the neighbourhood that are in state 1 –, originally studied by Wolfram [1986b]; and a set of complex rules empirically discovered by Wuensche [1993].

2.6 The Journey

With the latter paragraph we concluded this chapter. Specific points of the thesis are now linked to related pieces of research found in the literature, and also to subsequent chapters where the topic at issue will be addressed in more detail. What follows, therefore, is the core of the thesis, the detailed account we provided of non-intrinsic computations within Enact, a cellular-automata-based artificial-life world.

CHAPTER 3

ENACT: ARTIFICIAL LIFE IN CELLULAR AUTOMATA[1]

3.1 Introduction

This chapter describes *Enact*, a cellular-automata-based architecture of autonomous agents which this thesis rests upon. The level of approach we are interested in is the *organismic* level based on a population of agents that undergoes a coevolutionary process. By autonomous agents I do not mean autonomy in its technical sense as discussed for instance in [Bourgine and Varela 1992]. Autonomy will be used in the context of Enact in its informal connotation, so as to imply the lack of centralised control in the population of agents, and the fact that each agent is an individual with its own characteristics, in particular, with its individual role in the global dynamics of the system.

In the following sections we go through all aspects of the definition of Enact. From a high-level perspective, by relying on metaphorical concepts suggested by the artificial-life type processes it supports – namely, movement, selection, environmental interaction, mating, reproduction and development – down to the details of the processes and mechanisms involved. Then, the overall qualitative dynamics of the system is considered, and the aspects underlying its coevolutionary activity are discussed. Next, we trace a brief history of Enact's unnamed predecessors, showing how they paved the way to the inception of Enact. Appendices show the complete set of state transitions that define the system, and also give the commented C-code that was used to implement it. This chapter has a descriptive emphasis; more elaborate considerations about the system – including how particular world set-ups can be created within it, or how its artificial-life activity can be viewed as a computing machine – will be done in the following chapters.

3.2 Overview

Enact is a family of non-deterministic cellular automata whose basic dynamics can be described in terms of the metaphor of an artificial-life world where a population of worm-like agents of arbitrary length undergo a coevolutionary process. During their lifetime, the agents roam around, sexually reproducing, interacting with the environment, and being subjected to a developmental process which includes ageing and death.

The artificial-life type activity takes place over an inactive background defined by environmental regions which are "free", so to speak, for occupation by any agent; we represent the background by the *0*-state.

The cellular space that defines Enact is two-dimensional, its edges being wrapped around in a toroidal geometry. The neighbourhood used for the state transitions is the

[1] This chapter draws mostly from [de Oliveira 1992a] and [de Oliveira 1994a].

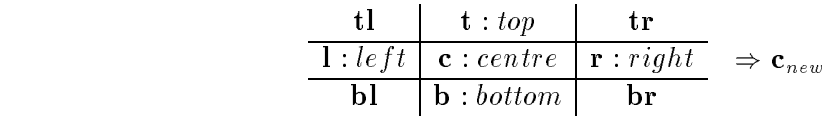


Figure 3.1: Moore neighbourhood and the notation used according to the geographic position of the cells.

Moore neighbourhood, defined by the cell itself and the 8 adjacent cells that surround it in a square lattice, as Figure 3.1 shows. Throughout the thesis we use the term *cell* with two different meanings: the defining unit of the cellular space, and the structural unit of the agent. While the former refers to the level of the definition of the automata, the latter refers to the high-level description of their dynamics.

3.3 Structure and Morphology of the Agents

Each agent can have arbitrary length and is defined by a sequence of contiguous cells, as depicted in Figure 3.2. The two cells at each end of an agent can be thought of as its head and tail, whereas the cells in between constitute its body. Head and tail are defined by one category of states, the terminal T-state, which encapsulates the agent's body. For the time being it is sufficient to think of a category of states simply in terms of a range of state values, different categories implying distinct, non-overlapping ranges; more details will follow in the next chapter.

One single cell of the agent constitutes its genotype; accordingly, its state does not change during the agent's lifetime and is defined by the state category referred to as G.

All the other cells have a phenotype-like nature, in the sense that they are subjected to change through environmental interaction, during the agent's lifetime. But while the *initial* adult state of the cells on the left-hand side of the agent's gene are, by design, directly dependent on this gene, the cells on the right are not, being initially determined through direct parental inheritance (Section 3.7 will make these points clearer). For this reason, we identify the latter cells as memetype-like, in an explicit reference to the notion of *memes*, the units of evolution that – following [Dawkins 1976] – are acquired by the offspring directly from the parents, such as knowledge and culture. Although still largely unexplored in computational systems, the importance of memetic evolution has been put forward not only in artificial life ([Farmer and d'A Belin 1992]) but also in cognitive science ([Dennett 1991]).

The morphological distinction between head, body and tail is a design artifact in order to make Enact's implementation and use easier. The tail provides a clear identification for the end of an agent, making some implementational details much easier. The head, even more significantly than its enabling the identification of the beginning of the agent, is ultimately responsible for the agent's ability to move. The body is to be the repository of the evolution-related components.

Therefore, considering that the body is effectively where Enact's evolutionary aspects are to be probed, for all practical purposes, it is convenient to think of the *effective* phenotype and *effective* memetype of an agent as only the cells of each type that are confined to the agent's body, and whose corresponding states are represented here, respectively, by the state categories P and K. Hence, throughout this thesis whenever phenotype or memetype are mentioned, the reader should bear in mind they really stand for the effective type confined to an agent's body.

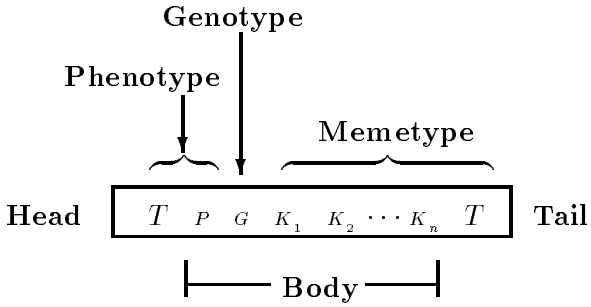


Figure 3.2: Structure and morphology of an agent in Enact. Phenotype and memetype may change through environmental interactions, but while the initial adult state of the former directly depends on the genotype, the latter is initially determined through direct parental inheritance. For practical purposes however, only the P-state is considered the agent's phenotype, and only the K-states its memetype (see Section 3.3).

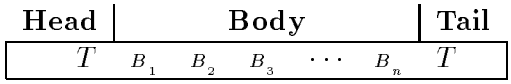


Figure 3.3: Alternative, simplified representation of an agent. The B-states represent the states of the body cells, with no reference to its internal structure.

What justifies the way I am denoting the internal components of an agent – in terms of genotype, phenotype and memetype – is the way each one of them is created in a newborn, as well as the way they are allowed to be modified during the agent's lifetime. These are details about the reproduction and development of the agents that will be discussed later on in this chapter.

It is worth advancing, however, that the (effective) phenotype will always be related to the way an agent moves. Beyond that, later on we will be referring to computations being performed out of the artificial life activity; in these contexts not only the phenotype, but also the memetype will represent elements of the computation, such as input or output. In the right context these details will be appropriately spelled out.

3.4 Movement

Enact's agents are inveterate wanderers. Whenever possible they move, one cell at a time; the head first, followed by each of their body cells, and finally by the tail. Two modes of movement are possible, leftwards or diagonally, an agent taking one of the directions according to its head starting to move to the left or to the top-left cells of its neighbourhood. Typically, the start of a movement is non-deterministic.

The basic fact about movement is that either leftward or diagonal movement can only start towards a region of the cell space that is mostly vacant; also, once started, the movement will always be completed even if the moving agent has started a reproduction process. In addition, a movement will never proceed if another agent enters the neighbourhood of its left-hand side terminal (its head); this prevents agents from "bumping" into each other.[2]

[2]The closer they get to bumping is a particular situation in which two agents are allowed to touch each

t_0	·	·	·	T	B_i	B_j	T	·
t_1	·	·	M	T	B_i	B_j	T	·
t_2	·	·	T	M	B_i	B_j	T	·
t_3	·	M	T	B_i	M	B_j	T	·
t_4	·	T	M	B_i	B_j	M	T	·
t_5	·	T	B_i	M	B_j	T	·	·
t_6	·	T	B_i	B_j	M	T	·	·
t_7	·	T	B_i	B_j	T	·	·	·

Figure 3.4: Succession of snapshots of the same set of cells as a 4-cell-long agent moves 2 cells leftwards in successive iterations. The dots represent the background state.

Because of the toroidal geometry of the cellular space, leftward and diagonal movements are sufficient to ensure that the agents have the ability to cover the entire world. In this way the agents are able to approach any other in the world and, when two of them reach a predefined mating configuration, they mate and reproduce; after each mating, they begin wandering again and so does their offspring.

As the cells of the agent move, a new state comes into play so as to occupy the empty place of the cell that has just vacated, thus preserving the spatial continuity of the agent; this state defines in fact, another category, and is represented here by the M-state. Figure 3.4 and Figure 3.5 show agents moving respectively to the left and diagonally, illustrating the action of the M-state. Note that a new M-state is created whenever the head of the agent moves one step in any direction. It then propagates along the agent, "pulling" the cells of the agent in the direction of its movement, one at a time; as the tail is finally pulled, the M-state disappears. Thus, the M-state exists within an agent only while the movement is taking place, disappearing as soon as the agent stops.

In order to start a movement, the head of the agent first "senses" its neighbourhood, in order to 'check' whether the way ahead is 'free'. If this is the case, then it 'casts' a movement state along the available direction, 'trying' to start the movement. This situation can be seen in Figure 3.4 during the transitions from time t_0 to t_1 and from t_2 to t_3, and also in Figure 3.5 during the transitions from time t_2 to t_3 and from t_4 to t_5. If only one direction is available, only one movement state is cast out, which automatically starts the movement according to the mechanism described above. However, if both directions are available two M-states are cast out, an impasse is established which is solved by a random choice not only among the competing directions, but also including the possibility that the agent discontinue its movement by "withdrawing" both M-states that have been cast out. More details about movement can be seen in Appendix A.2, where the corresponding state transitions are listed.

If an agent could not carry on its movement because of some obstacle in its way, soon after all its M-states disappeared its body would remain in a position determined by the path it went through. To compensate for that, we allow an additional kind of movement which is an *upward movement of the body*, whose effect is, whenever possible, to set the body in the horizontal position; Figure 3.6 shows one such situation. As will be made clear in Section 3.8, the body's upward movement has an effect on the reproduction process,

other in order to avoid a "head collision", that is, the situation in which there are two leftward-moving agents, one just above the other, and, at a certain moment, the one on the bottom tries to move diagonally; in this case, preference is given to the latter. The agent on the top then stops, and, for some iterations, its head is allowed to touch the body of the agent that moved. The details are not important for present purposes.

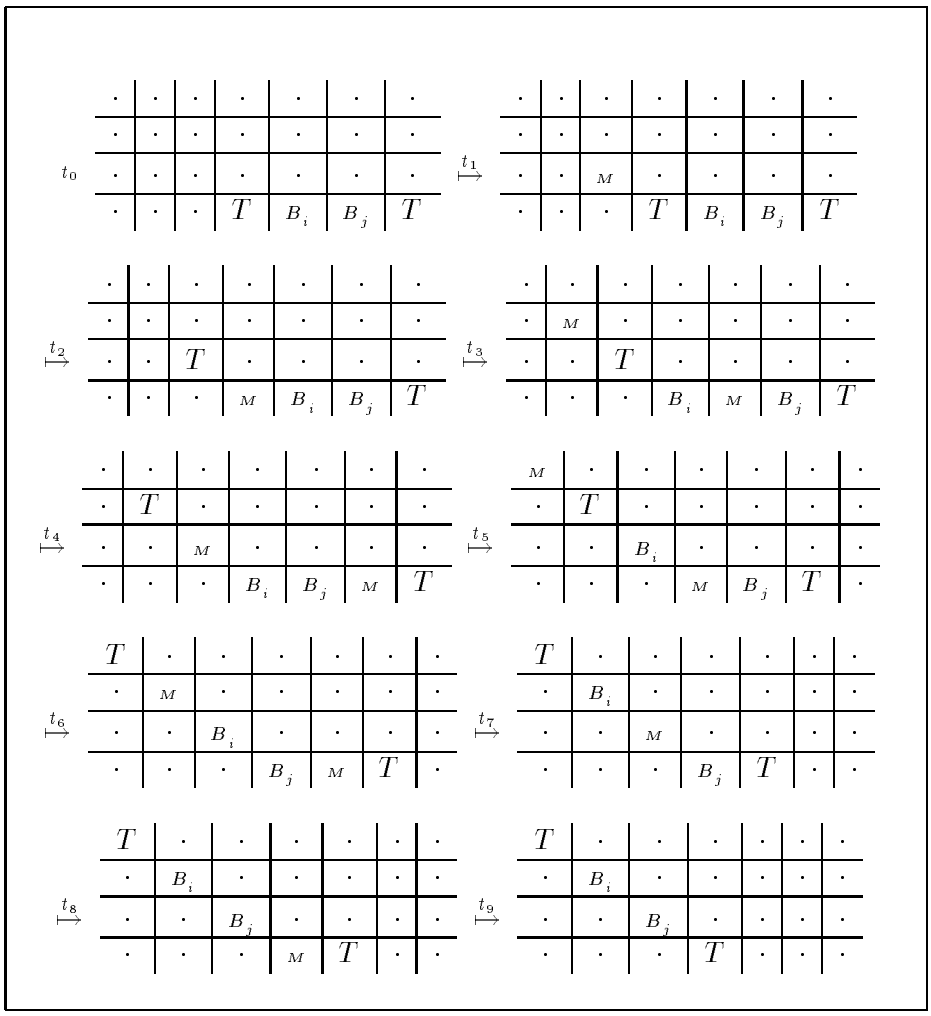


Figure 3.5: Successive snapshots from the same region of the cellular space as a 4-cell-long agent moves 3 cells diagonally, after being in a horizontal position. The dots represent the background state, and the B-states are a generic representation of any kind of body cell.

since it allows the increase of the rate of preservation of (non-deleterious) parental cell configuration in the offspring; in other words, its effect is to decrease the randomness associated with the process. The upward movement is also interesting in itself due to the extra appeal it adds to the activity of the agents without the need of an extra state. Perhaps most of all, since movement is the most fundamental aspect of Enact's dynamics, and since the system's dynamics is programmable, the upward movement provides a further way to control the dynamics of the world set-ups. We will return to Enact's programmability in Chapter 4, and to the importance of movement in Subsection 3.10.2.

Although the figures of this section show situations characterised by each individual form of movement, in typical situations the agents move in a composition of all three forms, with different cells of an agent moving in one of the three directions at the same time. The overall spatial disposition of the body cells on the space of the automaton is then always monotonically descending from the left, in a *worm-like*, wiggly fashion. The complete list of the state transitions for movement can be found in Appendix A.2.

The definition of the way in which the heads move depends on parameters that con-

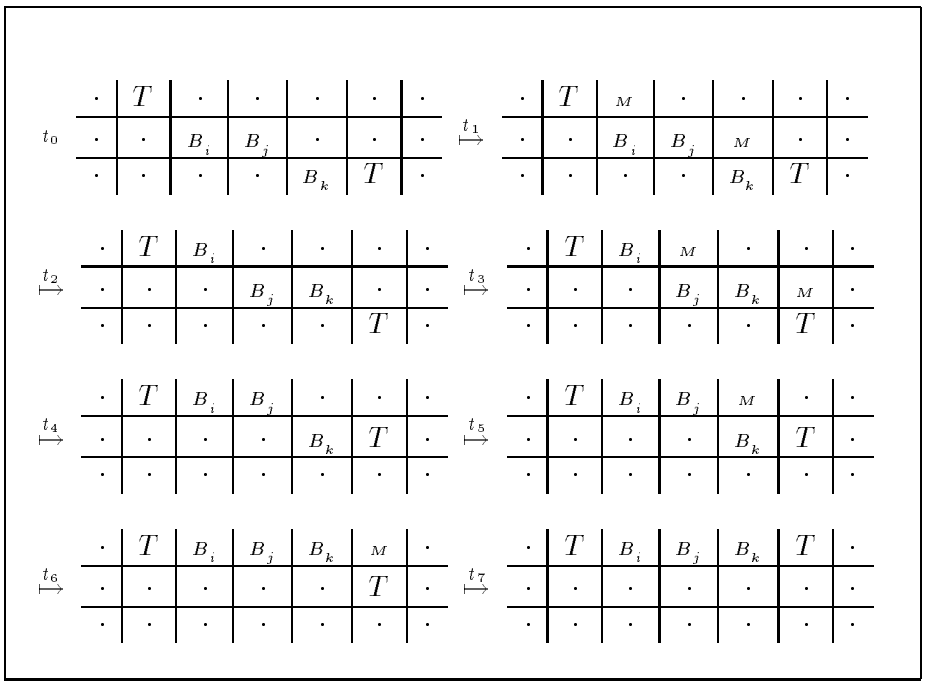


Figure 3.6: Subsequent snapshots of the same set of cells showing the body adjustment of a 5-cell-long agent, from an arbitrary initial position. The dots represent the background state.

trol how often an agent will attempt to make a single move leftwards (or diagonally) in opposition to not moving, and how a conflict of movement is to be solved. These parameters allow for the behaviour of each head to be defined, both in standard and conflict situations. Furthermore, they make it possible to define agents endowed with any kind of movement behaviour, and with any degree of determinism. All parameters work by defining the probabilities of each state involved in a non-deterministic state transition.

3.5 Environmental Interactions

In addition to the agents, it is possible to design environmental configurations which the agents are able to interact with. These configurations define another state category which we call E-states.

What supports this idea is that environmental configurations can be approached and eventually "touched" by the agents, unlike what happens among the agents themselves, which do not touch each other.[3] By designing configurations of E-states, which we refer to as *interaction sites*, and specific state transitions to be active in these sites, it is possible to have specific, controlled interactions taking place between the E-states and the agents present at the site. Note that, with this scheme, it becomes possible for any agent to interact with any other in the world, through the environment. In fact, as the next chapters will make clear, the agent-environment interaction, and consequently, agent-agent interaction can be made arbitrarily complex.

Figure 3.7 shows an example of an interaction site; the active situation that triggers the interaction is when the agent passes through the cross that defines the site. Note that

[3]Note that in this respect, the background state is special, since it is "occupied" rather than being touched.

this interaction site never blocks the movement of an agent, since the agent is always able to pass alongside it touching its top or its bottom, when it does not pass through the site. As a matter of fact, configurations of E-states arranged horizontally or diagonally can always be bypassed, while a vertical arrangement of two or more E-states is the only configuration able to block the way ahead of an agent.

The constraint imposed by interaction sites become evident by realising that Figure 3.7 is essentially Figure 3.5 with the interaction site at issue added. The trajectory described by the agent is evidently the same in both situations. But while in the former the agent is totally guided by the interaction site, in the latter it is totally dependent on the agent's ability (or chance behaviour) of making only diagonal moves at each step.

It should also be noted there is nothing that prevents the definition of a *dynamic* interaction site, i.e., a region of E-states with a dynamics of its own, independently of whether it is interacting with an agent. Again, such a dynamics can be made arbitrarily complex.

Another consequence of the introduction of the class of environmental states is that the background *0*-state – over which all activity takes place – becomes conceptually integrated into the environment as a special kind of environmental state, one that can be traversed, or occupied, by an agent.

3.6 Selection

Selection takes place in the following way: if for some reason the state of any cell of an agent changes to the background state, in a mostly vacant neighbourhood (i.e., with most of the cells being at the backgound state), the entire agent vanishes; the process occurs in a stepwise way, during the next set of iterations. This feature is equivalent to saying that agents which lose (at least) one cell, lose their contiguity, and cannot be considered to be proper, well-formed agents; therefore they must die out. Appendix A.3 presents the complete list of state transitions for selection; note there what is meant here by a 'mostly vacant' neighbourhood.

Therefore, in order to specify the selection process of a particular world set-up within Enact, it is necessary to design appropriate state transitions whose action is to switch the state of an agent's cell to the background state; as soon as the agent at issue happens to be in a mostly vacant neighbourhood it will eventually die out. For example, suppose one wishes to create selection against all agents that present some body cell in a different state from its two neighbouring cells; a state transition would then have to be added to the ones already existing, so as to "detect" the state configuration described, and "delete" – i.e., change to background state – the one associated with the centre cell of the neighbourhood. As a consequence, newborn agents with this deleterious feature would necessarily die out.

Enact incorporates a selection process that is strong enough to enable the generation of a random population of agents and environmental configurations from scratch. In other words, by initialising Enact's cellular space with a random initial configuration, the built-in selection process filters out all non-valid state configurations, leaving only environmental states and well-formed agents (see the corresponding state transitions in Appendix A.3.1). This feature is of a major practical usefulness as it rules out the previous necessity of *editing* the cellular space so as to generate a valid initial configuration of states.

Important here is that selection as described above is *not* an adaptationist process, i.e., the agents are not *selected for* by some concept of fitness. The emphasis is not on preserving the fitter agents but on killing off the ones which have deleterious state configurations in a particular situation. The emphasis thus is on concepts such as viability rather than fitness, and evolution by satisfying world constraints, rather than evolution towards

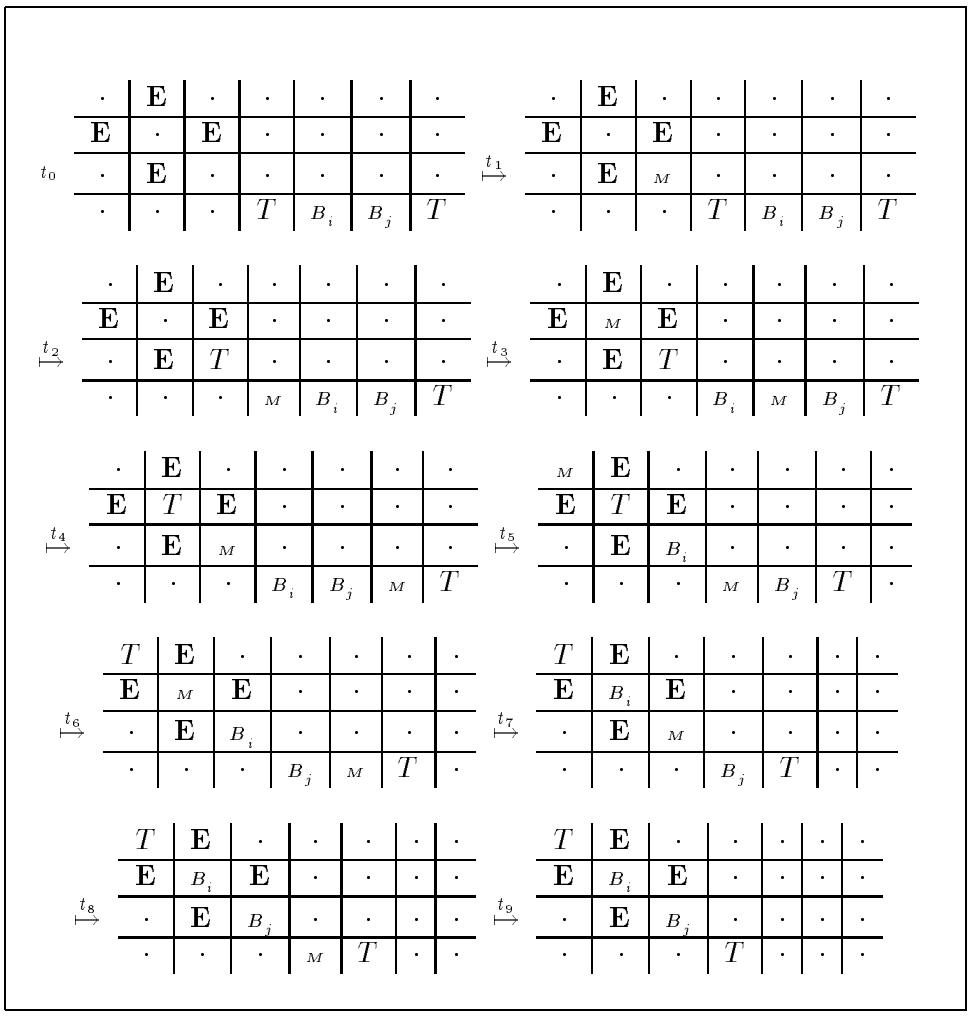


Figure 3.7: Illustration of the notion of interaction site, represented here by the set of four E-states in a cross-like fashion. The dots represent the background state, a special form of environmental state that does not obstruct an agent in its way.

solving predefined problems. We return to this topic later on, in Subsection 3.10.1.

In order to keep coherence with the view of selection implicit to Enact, we should replace the notion of an *useful* building block – the typical parlance within the context of evolutionary computation techniques – by a *non-deleterious* one. Incidentally, it is worth remarking that whenever we use the expression building block in this work all we mean is a sequence of contiguous body cells, pairs or triplets, of the memetype. In the current approach what is guaranteed is that any agent that is selected has some non-deleterious building block, even though it may be useless for any preconceived role.

3.7 Development

As will be discussed later on in this chapter, the major shortcoming of a precursor of Enact was a blur in the distinction between genotype and phenotype. Until then, both were simply valid interpretions of the agent's body cells, according to the world set-up being used. It is precisely such a problem that led to the inclusion of a developmental process

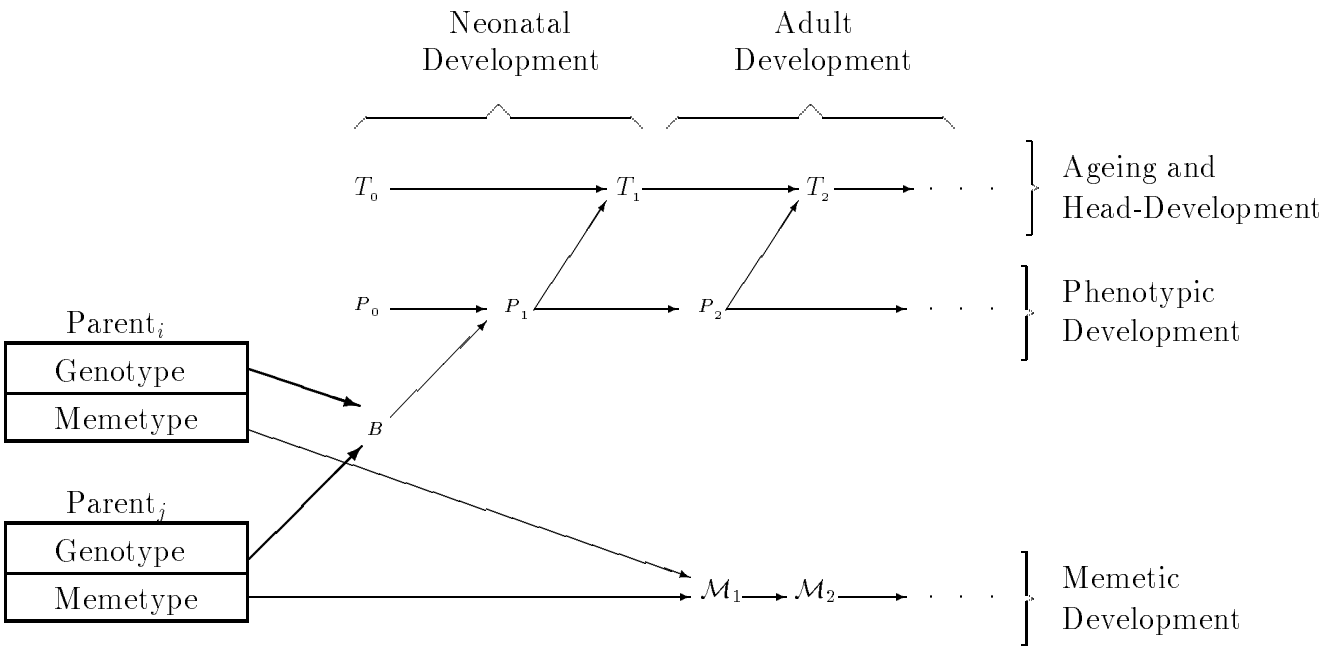


Figure 3.8: The stages of the developmental process. Only the newborn's genotype is exclusively dependent on the parents; all the other steps may have the influence of the environment. Ageing is the only developmental aspect that only depends on the "clock-tick" of the automaton. The index *1* refers to the beginning of adulthood, and the symbol $\mathcal{M}$ represents the full memetype of the developing agent.

that the agents undergo; the complete list of state transitions that allow development is in Appendix A.5.

Development in Enact has two stages, as shown in Figure 3.8: neonatal and adult. When a newborn is created it has two non-standard states T^0 and P^0, respectively, the neonatal states of its head and phenotypic cell. These are the states that characterize the neonate and triggers off its development, but have no effect on movement or reproduction, i.e., a neonate is immobile and sterile.

As state transition 1 in Appendix A.5.1 shows, the first step in the neonate's development is a state change in the phenotypic cell, replacing the neonatal state by an initial adult state; the state change depends on the gene state of the newborn and naturally can only occur after the latter has been created. The neighbourhood involved in such a step also includes parental cells which can then influence which phenotypic state will be initially created.

After the adult phenotypic cell has been formed, the second step of the neonate's development can then take place, which corresponds to a change of the head state of the newborn, according to state transition 2 in Appendix A.5.1. Again, parental features can also directly influence the formation of the new state, but in the presence of the phenotypic state that has been just created in the newborn.

As an adult, development happens in two distinct ways. On the one hand, the states of the head, the phenotype and the memetype can change in the lifetime of the agent by means of environmental interactions that are specially added into the world, according to a particular use of Enact. According to the type of cell involved in this process, we refer to head, phenotype or memetype development. Such a crafted development is supposed to be a systematic feature in Enact's use, and in fact, it will be exemplified in the experiment

to be discussed later on.

On the other hand, there is another facet of adult development which is a built-in process supported by Enact, independently of any particular way the system is being used. This facet affects the state of the head and is referred to as *ageing*. It takes place throughout the lifetime of an agent and can eventually end up with its dying out. Hence, whatever the agent is doing, it gets older, or at least, as currently implemented, there is a chance of its getting older. The way it happens is by the head going through a sequence of states, in a deterministic or non-deterministic fashion, the sequence length being defined by the world set-up in consideration. The ageing process carries on up to the point that the agent reaches the final possible state of the head, which entails its dying. It should be noted that the reference to ageing is simply a metaphor for the built-in process of continuous modification of the head state at each clock-tick of the state updating of Enact's cellular space; in fact, in the experiment described in Section 3.9, we use the metaphor of energy loss instead of ageing.

3.8 Reproduction

Any two agents mate if they vertically align their heads and phenotypic cells, leaving a layer of cells in the background state in between. In this mating configuration, one of the parental agents is on top of the layer and the other below, their heads being in the same column of the cellular space (see state transition 1 in Appendix A.4). Reproduction then goes on so that the new agent is produced in between the parents, starting from the matching heads and stretching to the right. Although there are deterministic state transitions that are triggered during reproduction, the offspring are created basically by means of non-deterministic state transitions, as Appendix A.4 (that depicts all state transitions involved in reproduction) makes it clear.

Just after reproduction has started, as soon as the parent on the top finds its way ahead "free", it may resume moving; when the way ahead is free for the newborn it too may move, even if reproduction has not yet finished. Eventually the same thing happens to the parent on the bottom.

The offspring's gene is created non-deterministically according to the parental genes. It may be the case that, at the moment the new gene is to be created, there are no parental genes in the neighbourhood; in this situation, the offspring's gene is randomly taken out of the range of possible genes available in the world set-up under consideration.

In order to support memetic evolution, reproduction of the memetic cells is based on the *preservation* of non-deleterious spatial sequences of their states. Since any agent that is able to exist in Enact is viable, in order to preserve non-deleterious sequences it is sufficient to allow the probability distribution of the non-deterministic state transitions to favour the reappearance (in the newborn) of configurations of memetic cells already present in the parental agents.

3.8.1 Sexual Reproduction

In Subsection 2.5.2 we discussed the issue of self-reproduction and sexual reproduction in cellular automata in general, and provided a comparative table between Enact and similar efforts.

Under the light of the details of the system provided so far it is worth including here a summary of the features involved in Enact in respect to reproduction. In fact, the standpoint adopted in Enact was one of achieving sexual-reproduction, with a significant

degree of complexity, that would allow the satisfaction of a number of constraints such as the

- necessity of a mating configuration for the parental agents;
- requirement that agents of any size can reproduce;
- necessity of coping with the movement of the parents and of the offspring as reproduction takes place;
- preservation of the programmability of the system;
- virtual unboundedness of the number of possible initial configurations; and
- premise of being able to describe the activity of the agents from a high-level perspective.

3.8.2 Crowding Effect

In running experiments, even though selection is killing off agents all the time, because the cellular space is finite it may get overpopulated. As a consequence, we experience a *crowding effect* which implies that, after some degree of crowding is achieved, it becomes less likely that a reproduction involving long parents will be able to produce a similarly long offspring. The point is that less and less background cells become available, and consequently, once reproduction starts, it is normally curbed by a moving agent that gets into the background layer in which the newborn is being created. But then the parental agents start moving again, and similarly the newborn; as soon as the newborn's last body cell also moves, reproduction necessarily stops, as mentioned earlier. The effect then is that, as the cellular space gets more and more crowded, an increasing bias towards shorter length agents takes place.

Note however that the real cause of the bias is the last state transition in Appendix A.4, the one that adds the tail to the newborn as soon as it moves. To some extent, therefore, there is an intrinsic selective pressure – towards smaller individuals – defined by the state transition. Nonetheless, the crowding effect itself is due to the global behaviour of the automaton, which amplifies the selective pressure already implicit in the transition. One way to minimize such an effect would be to allow a background selective process which would randomly set cells to the background state with a small probability; naturally, its value would have to be worked out empirically, according to the domain concerned, and the size of the cellular space being used.

3.8.3 Further Details of the Reproduction Process

The cells of the newborn are created one at a time, both the ones from the body and the terminal states. In addition to two special state transitions that create the neonatal states of an agent, there are four basic classes of state transitions for reproduction: deterministic rules leading to a T-state; non-deterministic rules leading only to a B-state or only to a T-state; and further non-deterministic ones leading to either of them. Reproduction starts by creating a head for the newborn whenever the situation described above takes place. Then it proceeds in a non-deterministic fashion by creating its body cells; the creation of the genotype and the neonatal phenotype are handled as special cases at this stage (the latter being created, in fact, deterministically). Finally, it creates the newborn's tail in a non-deterministic way, unless one of the following happens: first, the newborn has "moved too much" even before completely born (i.e., an M-state reached its right-hand extremity,

as the last transition in Appendix A.4 shows); or, second, there is no more possibility for the newborn to acquire a body cell from its parents (as shown in state transitions 6, 7 and 8).

What follows are details primarily involving the creation of the newborn's memetype. The fundamental point about designing state transitions for reproduction is that it must be able to provide variability without being disruptive, i.e., it should allow for the preservation of the non-deleterious (viable) configurations of body cells already existing in the neighbourhood. Since in the current approach any agent that is able to exist in the cellular space has viable building blocks, what we have to do is to allow the probability distribution of the non-deterministic rules to favour the reappearance of building blocks of the parents, which are defined in the newborn by its most recently created cell and by the cell that is about to be created; this is accomplished by equally distributing the probability of the state transitions accordingly.

If there are no building blocks to be preserved, we just randomly choose any of the parental body cells present in the neighbourhood. Because reproduction is not prevented from taking place while the parental body cells are in movement, it may be the case that no parental cell is present in a neighbourhood (see transition 3 in Appendix A.4 for clarification). In this situation, the newborn body cell to be created is randomly chosen from all the available cells of the world set-up at issue. Moreover, even when there are building blocks to be preserved in the neighbourhood, a body cell can also be created through the latter process, thus giving a minimal uniform bias towards all possible B-states of the world set-up, equivalent to the maintenance of a residual background mutation. The B^*-state which appears in Appendix A.4 refers to a B-state created in the newborn in the way we have just described. In particular, the length of the newborn's body is never more than one cell longer than the length of its longest parent.

Returning to the additional motivation for having the upward movement of the body (as hinted at in Section 3.4), let us recall that the emphasis of reproduction is on the preservation of the parental building blocks. So, if the agents did not have the upward movement the chance that a body cell in the newborn were created from a neighbourhood with few or no parental body cells would be greater. The consequence would be that the rate of preservation of viable parental body cell configurations would be smaller. Therefore, the exploration of the space of possible body-cell configurations would be more random, less oriented by the current state of the process.

3.9 Qualitative Dynamics

In this section we illustrate the qualitative dynamics of Enact by providing an overall description of the changes in population size, through the effect of varying ageing rates. Although a particular world set-up was used, the points made here are not dependent upon its particular characteristics.

3.9.1 Scanning the Dynamical Regimes by Varying Life Expectancy

The results presented here are from runs in which the initial configuration of a 32×32 cellular space contains 13 agents with random genotypes, and approximately the same length. The initial position of the agents on the cellular space is the same throughout the runs, although the agents themselves are different from each other.

Data from the evolution of the cellular automaton for about 24000 iterations is shown in Figure 3.9; the plot is compressed, each point having been plotted at each 20 iterations. The vertical axis represents the population size, the numbers attached to the graph

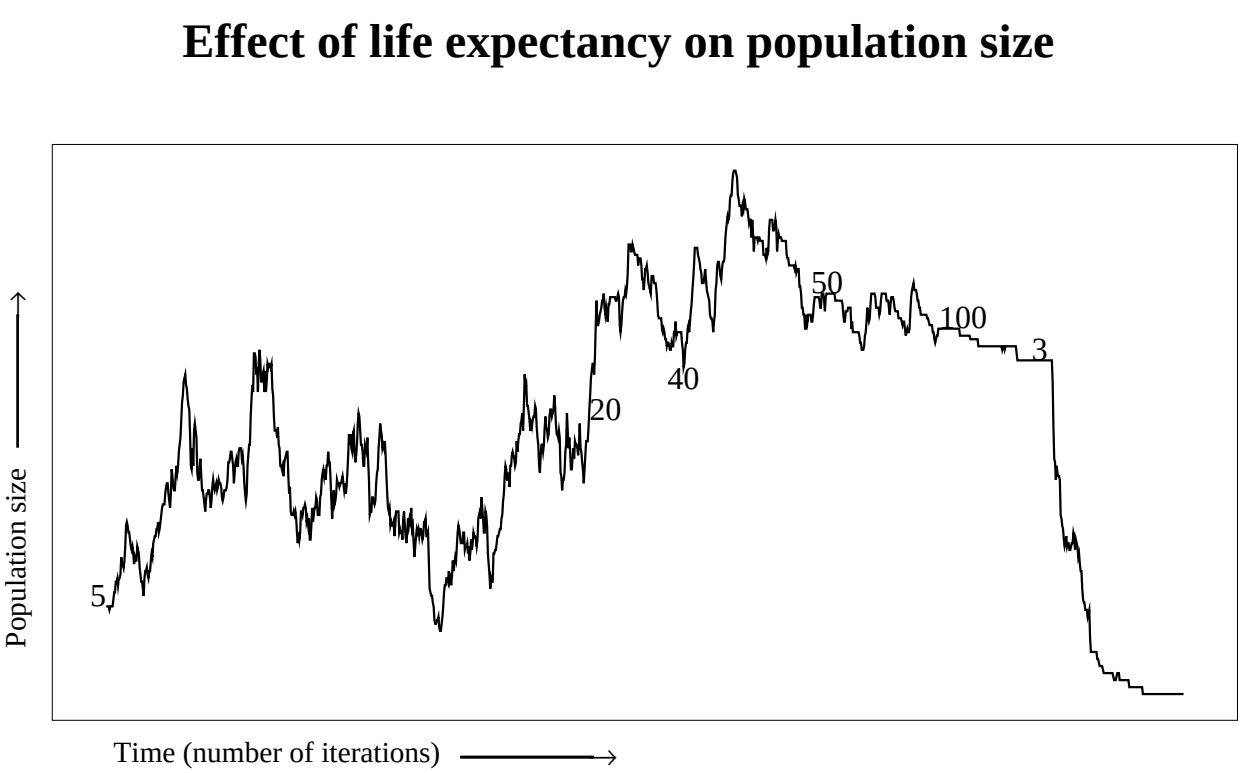


Figure 3.9: Effect of the expected life span of the individuals on the dynamics of the overall population. The origin of the graph is the bottom of the left-hand corner. The horizontal axis is time (number of iterations), and the vertical is population size. The graph spans through about 24000 iterations and is compressed, each point being plotted at each 20 iterations. The numbers attached to the graph provide a measure of the expected life span of the individuals, and were manually changed during the run; more details explained in the text.

being the normalized values of the nominal, expected life span of the individuals in the population. The normalisation factor is 1/60, and the nominal, expected life span of the individuals is directly derived from the ageing rate that was set up for the population. Therefore the graph starts with life expectancy of 5 ($5\times60=300$ iterations) and is progressively increased up to 100 ($100\times60=6000$ iterations), from which it is directly decreased down to 3.

For nearly half of the period the population expectancy is kept at value 5, the higher values occurring only in the second half. Such a division identifies two distinct dynamics. The first half is characterized by a quickly-changing population whose average size is however kept low. An indication of this highly active dynamics is the "spiky" nature of the graph during the period.

The second half is marked by an increase in the average population size, since the agents are able to live longer, thus having a higher number of offspring. Note that the increase in population is initially very sharp (at value 20), indicating that the agents are exploiting their new, longer life span in a world that, at that moment, was underpopulated; note also that the maximum population size reached in the run occurs at life expectancy 40. After the initial abrupt change, the variation in the population size becomes smoother, thus

suggesting that the dynamics has also slowed down. The reason is that the cellular space has become more crowded, which decreased the mobility of the agents, thus decreasing the chance of mating.

If the mobility of the agents continues to decrease, they can even reach mobility deadlocks, in which the movement of one is precluded by others, to the point that reproduction rate is eventually brought down to zero for long periods. Since the condition to break the deadlock is to clear the way ahead for an agent, if the agents are living too long, the periods of deadlock also increase. Therefore, as the expected life span reaches some particular high value, the deadlocks tend to prevail. This is what we see for the case shown, from life expectancy 50 onwards. This new dynamical regime is characterized by the presence of plateaux, which become longer as life span increases.

Note that a fixed population size can also be reached without mobility deadlocks. This is the transient situation in which the population size has decreased so much that the agents are moving in such a way that they do not reach the mating configuration; their mobility is indeed maximum since not even stops for reproduction would be taking place. This case can be seen in the graph in the form of the small plateaux that appear just before the population vanishes, at the near end of the graph.

If we finally decrease life expectancy to a very low value, the reproduction rate becomes lower than the death rate and the population is eventually extinguished. This is what happens when we decrease life expectancy from 100 down to 3.

3.9.2 Two Attractors: Extinctions and Deadlocks

Experiments have suggested the existence of two kind of critical population sizes, so that, beyond the highest, the dynamics becomes marked by deadlocks; below the lowest, the population quickly becomes extinct. Whenever a population has managed to avoid such an extinction, we observe self-maintaining populations with stable dynamics that are preserved for very long periods. For any life expectancy the condition for the population to be mantained is naturally that reproduction rate be balanced, on average, by death rate. But the condition that supports the balance varies for different values of life span. Hence, during the first half of the evolution shown, the degree of mobility of the agents is such that they manage to keep a high reproduction rate. For progressively higher life spans reproduction is kept at progressively lower rates; in the extreme, immortal agents would live in an eternal deadlock, but bringing the reproduction rate down to zero.

Figure 3.10 summarizes the causal links between the parameters that determine the basic dynamics in Enact. Note that, except for the expected life span, all the other parameters are only implicitly defined; note also that the chance of reproduction is a parameter that affects the offspring of an agent, while the chance of death affects the agent itself. The positive links represent a direct dependence between the parameters involved, i.e., both increase or decrease together (although at different rates); analogously, a negative link represents reverse dependence between them.

From the points above we can identify three qualitative dynamical regimes that we pass through by varying life expectancy from a very low to a very high value:

1. with high mobility and low reproduction rate (which in the graph shown is typified by life expectancy 3);
2. with high reproduction rate and high mobility (as happens for value 5); and
3. with low reproduction rate and low mobility (as for all values ≥ 20).

Experimental evidence has shown that the dynamics in the first regime has a high chance of leading to extinction of the population, while the dynamics in the third is marked by

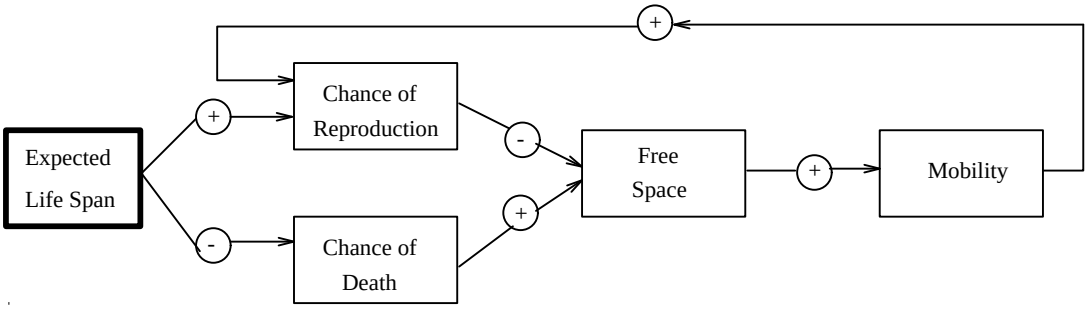


Figure 3.10: Causal links between the parameters that determine Enact's basic dynamics. Only the expected life span is an explicit parameter, effectively controlled.

frequent deadlocks. Therefore it seems fair to say that the basic dynamics of population in Enact is dominated by two attractors: extinction and deadlocks. Even though extinction is the only formal attractor. By referring to the deadlock regime as another attractor, we mean to highlight the fact that, in practical terms, once the deadlocks start, it becomes very difficult for the dynamics to be driven away from it, although such a possibility always exists.

In Enact, free cellular space equals resource: for reproduction, which can bring about novelty; and for movement, which allows environmental interactions. From the point of view of "tuning" Enact for an artificial-life use, the second dynamical regime is the natural option. Eventually, the population may fall into extinction or into frequent deadlocks, but very likely, only after a very long transient characterized by a long-lasting, dynamic population, marked by genotypic novelty and environmental interactions.

3.9.3 Enact's Regime of Operation

As discussed in Chapter 2 the studies on cellular automata dynamics presented in [Langton 1990] suggest that, as far as the emergence of life and computation in natural and artificial systems is concerned, the "interesting" dynamics lies at a phase transition between order and disorder. In the case of cellular automata rule spaces, this means the region of the space between cellular automata that typically converge to limit points and cycles, and others that lead to chaotic regimes. Although the characterisation of this region is not precise some recurring features (to be presented below) have been accepted as necessary. It happens that, by setting up Enact such that it is tuned at the dynamical regime between extinctions and deadlocks – its natural dynamical regime, as discussed above – all these features are typically observed in the system. Namely, the following parallel can be readily established:

- *Existence of very long transients.* The existence of long-lasting populations associated with Enact's natural regime of operation does not mean the population will last forever. It may eventually lead to either of the two attractors referred to above.

- *Dependence of the transients on the size of the cellular space.* As the size of the cellular space increases, the chance of reproduction decreases pulling the dynamics away from the deadlocks. On the other hand, extinction becomes likelier.

- *High, but not maximal, temporal and spatial correlation between the cell states.* This a direct consequence of the agents having the ability to move about, and also

a structure. As they move, the collection of cells that make up their bodies are displaced in a well-defined, but not predetermined range of possibilities.

- *Existence of propagating structures.* In this respect it is worth adding that the agents in Enact have a *soliton*-like behaviour: they propagate as a well-defined unity; according to well-defined laws; and can preserve their identity after their "collision" with environmental cells of an interaction site.[4]

- *Ability to support universal computation.* It has been conjectured in [Wolfram 1986b] that all cellular automata with the ability of universal computation would be characterised by the so-called "complex dynamics", essentially the same one identified in [Langton 1990]. It will be a recurring theme in forthcoming chapters that Enact has universal computation capability.

It is remarkable that all the features above turned out to hold in Enact, even though they were not preconceived, design premises.[5]

3.10 Evolutionary Activity

In this section we examine the fundamentals underlying the evolutionary activity in Enact.[6]

3.10.1 Selection: The pathways rather than the ends

In the standard use of evolutionary computation – genetic algorithms (GAs), for instance – in optimisation problems, selection is based on the definition of an explicit fitness function. In these cases, its typical role can be allegorically envisaged as the establishing of a set of beacons in the search space, that the search process would then try to reach. As there is no explicit concept of fitness function in Enact, a better image would be the exploration of a space where a set of beacons tells the exploration mechanism where are the forbidden points of the space. So, whereas in typical applications of GAs in optimisation problems selection is bound to rule out most regions of the search space, selection in Enact would not be so strict, as most of the space being explored would be in principle acceptable.

Consequently, selection in genetic algorithms is typically selection *for*: the emphasis is on the environmental interactions preserving the fitter individuals of the population, according to a fitness function predefined for the particular application.

What enables one to refer to selection *for* is either a hindsight analysis of a particular evolved trait, or the previous knowledge of the evolutionary pathway that has to be taken. But in the natural world selection operates *against* features that are incompatible with reproduction and survival; this is the only true statement that can be made in real-time, that is, during the span of time selection is still operating. Such a perspective is the one adopted in Enact, although, for short, we will simply refer to selection, omitting the word "against". In Enact, therefore, the agents that have deleterious features (state configurations) in a particular world set-up are the ones eventually killed off. The emphasis

[4] Or with a head collision with another agent, the situation mentioned in Section 3.4.

[5] Naturally, the movement of the agents was thought of in advance, but not its connection with the issue of propagating structures in cellular automata.

[6] This section revises previous material published in [de Oliveira 1994c]. The difference is that adaptations are explicitly recognised herein as a real possibility in Enact (even if in a constrained form, as will be discussed), whereas in that previous work adaptations had not been considered possible at all.

thus is on concepts such as viability rather than fitness, and evolution by satisfying world constraints, rather than evolution towards solving predefined problems.

This approach to selection then stresses the point that, in general, it is not possible to drive evolution in Enact to a predefined end-point. All that is possible is to prevent some evolutionary pathways in advance. Of course, if all evolutionary pathways to a particular end-point are known in advance, it becomes possible to precisely reach that point; however, this is not the general case. Therefore, what matters the most in the system are the pathways, not the ends.

Hence, selection in Enact is not only in tune with biological reality in terms of its focussing on the elimination of unfit agents, but also in the sense that, insofar as it does not stress the end-points of evolutionary paths, it opens up space for the exploration of its evolutionary processes beyond the scope implied by *selection for*.

3.10.2 Movement is Enact's power-house

Movement is the primary source of interaction between the agents. The movement of an agent has a local effect on the movement of its neighbours, and may propagate over the cellular space due to the whole sequence of perturbations of movement to other members of the population in its lifetime.

The amount of perturbation that a newborn is able to introduce into the organisation of the population depends on the size of the population, the current dynamics of the world, and the size of the cellular space. Essentially, the effect depends on the density of free space currently available. If the population is close to extinction, a great many free cells are available, and the newborn's influence is bound to be locally damped. The consequence is the same for the case in which the cellular space is overcrowded; in this situation, the lack of movement of the population due to lack of available free cells, leaves very little room for any perturbation to propagate through the population. However, in Enact's dynamical regime of operation, i.e., away from extinction or deadlocks, it is likely that the newborn's presence will be felt in a large extension of the cellular space.[7] Also, we can expect a certain critical population density in this regime for which the perturbation will be maximal, even reaching, in some cases, the entire population. Empirical observations have confirmed these expectations.

Except for ageing, which depends only on the clock-tick of the cellular space updating, all the other processes embedded in Enact are powered by movement. Even more importantly, through movement all processes become coupled to each other. Movement is therefore the "power-house" of Enact. Life, death and reproduction of the agents are dependent, through movement, on their being in the right place at the right time.

3.10.3 The Genotype Only Provides Initial Conditions for Development

The head, taking on the states that determine how the agent is going to move at each step of its lifetime, has the ultimate responsibility for an agent's movement. The initial adult state of the head depends on the agent's initial adult phenotypic state, which in turn, depends on the agent's genotype. As development unfolds and the initial adult phenotypic state is replaced, the genotype then becomes ineffective in regard to the individual's movement.

The role of the genotype is then evident: since an agent's movement is coupled to the movement of other members of the population, the genotype cannot fully determine the

[7] Precisely to which extension the newborn's presence will be felt is, however, a question that will not be addressed herein.

pattern of movement; it only constrains an individual's movement by providing initial conditions. Hence, it is through the genotype that the *initial pattern of movement* of an agent is established.

In analogy to searching a genotype space, similarly to the search performed in genetic algorithms, we can refer to a process of "genetic search" as also taking place in Enact. In terms of this supposed notion, the genetic search in Enact is the search for a viable genomic pool in the population that, by having contributed to the various agents moving in a certain joint pattern, has prevented the population from dying out.

Although there is a mapping between genotype and the initial state of the phenotype, there is no mapping between the former and the long-term phenotype of the agent. The reason is that the latter depends much more on the agent's history of interactions than on its genotype.

A similar process happens in relation to the memetype. While there is direct memetic transmission of parental features to the newborn, its long-term status, again, is only constrained by the original memetype; the latest states of the memetype are more dependent on the agent's history of environmental interactions and its coupled movement with the other members of the population.

Therefore, from the point of view of phenotype and memetype, the role of the genotype is the specification of an *initial trend* in phenotypic and memetic spaces.

3.10.4 Coevolution with Constrained Adaptation

The actual sequence of phenotypic and memetic transformations of the agents is thus far from being fully specified in their genotypes. Depending on the global configuration of the cellular space, and in particular, the actual physical location where the agents are born, the long-term development of two agents may significantly vary, regardless of their having the same genotype; analogously, two agents having distinct genotypes may reach the same long-term developmental state.

Although this lessened role of genotype in development (and consequently, in evolution) is certainly not the common view, it is a central tenet for a minority (though eloquent) group of researchers. Lewontin, for one, has strongly put this point forward, as in [Lewontin 1983]. In this reference he discusses the relations between organism, environment and ontogeny, and supported by a great number of illustrative examples from biology, draws conclusions (page 71) such as that

> "...there is a many-to-many relationship between gene and organism. The same genotype gives rise to many different organisms, and the same organism can correspond to many different genotypes."

With the inheritance of non-deleterious features – from parents to their offspring – playing a minor role in the formation of the agents, and consequent predominance of movement in the determination of the individual's lifetime interactions, it turns out that, for all practical purposes, as a new agent is born its parental lineages can simply be thought of as discontinued (even though, in fact, they are not). Hence, there is not much room for progressive, phylogenetic improvement of characters to occur. In other words, natural selection can only occur in a very limited form. Naturally, an additional aspect of this point is the fact that the genotype of the agents, being made up by a single gene, entails that genetic variance becomes very constrained.

This situation is somewhat analogous to a GA-like process in which reproduction occurs with very high mutation rate, or in which epistasis[8] is too high. In both cases,

[8]Epistasis is the biological phenomenon through which a phenotypic trait depends on various genes,

as far as the search for viability is concerned, each new generation corresponds to a breakdown in relation to the previous one: high mutation entailing the actual disruption of viable building blocks that may have been formed (regardless of the interdependence between their associated genes); and high epistasis provoking too much interference of the epistatically linked genes (no matter how viable they are). As discussed in [Davidor 1990], in either cases only random search of the space would be possible. In Enact, development being the result of a history of complex interactions during the agent's lifetime, to a great extent is unpredictable in the beginning. Consequently, development has a somewhat random component which entails that only random exploration of the space of possible developments becomes feasible.

A deeper way of envisaging the roots of the constrained role of adaptations in Enact is to remind of an usually non-explicit premise in the theory of natural selection; one that, for having systematically been left implicit, has very often led to a misunderstanding of the theory. The point is that for adaptations to come about, not only some form of "transgenerational stability"[9] is required for the organisms, but also for the environment.[10] We could even say that not only reproduction of organisms is needed, but also of their environment. Oyama [1985, page 22] makes this point very precisely when writing that

> "What is required for evolutionary change is not genetically encoded as opposed to acquired traits, but functioning developmental systems: ecologically embedded genomes."

Also, in [Varela *et al.* 1991, page 199] the same idea is expressed when the authors state that

> "Genes are, then, better conceived as elements that specify what in the environment must be fixed ... In every successful reproduction an organism passes on genes as well as an environment in which these genes are embedded."

In a more intuitive way, such an aspect can be readily accepted by imagining what would happen if the fecunded egg of some animal were transplanted into the womb of another animal of a very different species.

Two consequences of the preceding points can then be taken, which jointly provide the underlying aspects of Enacts' evolutionary activity:

- First, any evolution in Enact is indeed, *coevolution*, any evolutionary process then becoming the process of coevolving coupled movement among the members of the population.
- Second, all coevolutionary activity happens with a low emphasis on progressive improvement of characteristics that natural selection induces along agents' lineages in the form of adaptations.

---

which are then considered to be epistatically linked. The higher the epistatic links associated with the trait, the higher the interdependence between traits, if any, that are individually defined by each gene.

[9] This term is due to [Maturana and Varela 1987].

[10] As stated above, in general the perturbation entailed by a newborn does not reach a degree that might lead to a total reorganisation of the population. Therefore, there may be preservation of order across the generations; that is, some transgenerational stability of the population's pattern of movement. However, up to this point, it is an open question which are the consequences of such a feature for the long-term development of the individuals. It is not possible, in advance, to ensure how disruptive a new agent will be for the population. All that is guaranteed is that, provided the system is tuned in Enact's dynamics of operation, the newborn's presence will have a local effect on the population, which will, to some extent, spread out.

3.11 Implementation

Enact and its predecessors have been implemented on a Sun workstation using Cellsim 2.5, a public domain cellular automata simulator ([Langton and Hiebeler 1990].[11] The commented C code for Enact's state transitions is presented in Appendix B. In the current implementation the system has 29 state transitions for movement, 14 for reproduction, 9 for development, and about 37 for selection.

Each one of the six possible state categories is defined by a range of state values specified by the user, out of a total of 256 states. Additionally, there is a set of parameters that can be manually set up to specify details of the movement of the agents (such as the preferential direction of movement of an agent); the ageing rate of the population; the rate of background mutation; etc. Other details about the implementation are made explicit as comments in the code shown in the appendix.

3.12 An Historical Perspective of the System

This section is aimed at providing an historical perspective on the development of Enact. However, as a contrasting point it would be worthwhile to wind back in time even before the first steps towards Enact, and have a glimpse of the first attempt that was pursued.

3.12.1 Before Enact

As I started evaluating the use a cellular-automata-based architecture that could be appropriate to support the emergence of functions, the first direction that was taken was to try to insert, somehow, Lisp objects (S-expressions, i.e., lists and atoms) into cellular automata. The idea was the possibility of observing the emergence of Lisp functions.

There were two strong appeals for using this language. An empirical reason was that Lisp code had been used with great success in Koza's [1990] technique of genetic programming mentioned earlier (also in [Koza 1992], which I had read a preprint of). This technique is essentially a search method in the space of Lisp functions that finds a particular function to solve a predefined problem. Another suggestion one would get from the literature would be the use of an assembly-like language, as in [Harvey 1991], but I thought the higher-level of Lisp might be an advantage since shorter programs would be possible that would code for more complex functions.

The other reason for the choice was that there was a Lisp definition that was essentially the original proposition of a pure-Lisp, with some minor additions to make it more amenable for implementation and use. This Lisp was presented in the monograph [Chaitin 1987], where extensive formal analysis also made, from which one could work out, for instance, the number of well-formed S-expressions of a certain size. This analytical possibilities, might be an extra advantage when analysing the outcomes of the system I was trying to design.

In parallel with this enterprise, I undertook a series of experiments with Cellsim in order to learn about the behaviour of cellular automata in general; as a way to probe their applicability as computing devices; and third, to evaluate the possibilities of Cellsim as a platform for the system I wanted to implement.

Having learned that Fontana [1990] had implemented AlChemy to study the emergence of functions in the Turing gas, this piece of work acted as a reinforcement to the approach

[11] This version was implemented using the Sunview package, which is no longer available with Sun's latest environment (Solaris 2.x). An X11R5-based implementation was performed by Felicity George (fawg@epcc.ed.ac.uk) and is available from her upon request; however, it does not support colour processing.

I was pursuing. First, because of the common conceptual ground they shared to some extent. Additionally, the implementation of AlChemy was derived precisely from Chaitin's Lisp, and fundamentally for the same reasons that the latter had become a reference for me. And since Fontana had publicly offered his code – which was in C, the requirement for using it in conjunction with Cellsim – it seemed as though some of my problems with my own system would be solved.

But they were not. First, because I never managed to get hold of AlChemy's code. Second, my practical experience with Cellsim soon made it evident how computationally intensive would a cellular automaton be if its state transitions were to be based on the outcomes of a Lisp interpreter. And third, as a matter of fact, I never really came up with a Lisp-integrated architecture that I considered appropriate. The integration schemes I could think of always seemed overly *ad hoc*. Because of all that, there was no alternative than changing the approach.

3.12.2 Enact's Lineage

Enact is in fact the name of the last version in a trilogy of cellular automata embedding an architecture of autonomous agents from an artificial-life perspective. The level of approach we were interested in was the *organismic level*, based on a population of agents that should undergo a coevolutionary process. Small modifications were systematically performed throughout Enact's history, in order to account for its conceptual evolution, and to progressively improve the already existing processes at each moment. What follows is the main line of Enact's history of developments, which can be traced in three stages:

1. In [de Oliveira 1992a] the first version was introduced, presenting the basic concepts of *movement* of the agents, *selection* and *reproduction*, and showing how these processes could account for a simple form of evolutionary mechanism.

   The motivation at this stage was to embed some form of evolutionary mechanism into cellular automata, but with no optimisation concern.

   It turned out that the temporal evolution of the cellular automata that were developed had a number of interesting features from the point of view of artificial-life, to the extent that, by conveniently extending their definition and improving on the conceptual issues underlying their use, it was possible that a framework to support a class of artificial-life worlds could be developed. On pursuing this target the second stage was reached.

2. In [de Oliveira 1993] *environment* was introduced, with which we showed the implementation of a Turing machine, stressing the methodological issues involved in the use of the system as a programmable machine.

   As an artificial-life world, its major drawback was the provision of interaction between the agents, which was very poor, since the only kinds of interaction provided were reproduction, and the ones derived from movement. This problem led to the notion of the interaction sites – by means of the environmental E-states – with which the interactions among the agents could then be achieved with virtually unbounded richness. In comparison with the first stage, Enact's second stage had incorporated the following major improvements:

   - The original notion of a "genotype", represented by the state category G, was replaced by the notion of a body state B, in the sense of generic, active states of the body cells which, depending on their use, could be regarded either as a genotype or a phenotype.

- The original movement *m*-state was replaced by the state category $_M$.
- The environmental category, represented by the E-states, was created, the background *0*-state becoming conceptually encompassed by the environment, as a special kind of environmental state.

However, as hinted at above, there was yet a major problem associated with the framework, as it stood. Namely, the distinction between genotype and phenotype of the agents was blurred. On one hand, reproduction could directly act over the $_B$-states as if they were the genotypes of the agents. On the other, in some world set-ups the $_B$-states could well be interpreted as a phenotype, even featuring a transformational change in them, during an individual's lifetime, that could be interpreted as its development. With this problem in the foreground the third, current stage was then reached.

3. The current version, which is referred to as *Enact*, came from the re-evaluation of its unnamed predecessors, so as to provide a clearer account of the distinction between genotype and phenotype; this led to the introduction of a *developmental process* that the agents undergo, and to a reformulation of the internal structure of the agents, which entailed splitting an agent's body into three parts, *genotype*, *phenotype* and *memetype*.

   With these three state categories, the blur between genotype and phenotype was wiped out. Consequently, related achievements were incorporated in the system such as a more powerful account of genotypic evolution, which revises the limited notion of "genetic search" explored in [de Oliveira 1992a], and that brings the concept of coevolution to the foreground of the evolutionary process. Also, the support to memetic evolution, which allows exploration of the effects of genotypic evolution on the state of the agent's memetype.

   More precisely, I used the term genetic search in [de Oliveira 1992a] to refer to the exploration of viable configurations of body cells. But since the concept of environmental states was absent at that point, the configurations were directly due to the reproduction process, without the possibility of also being the outcome of environmental interactions, as is now the case. Furthermore, since in that paper there was no distinction between the states of the body cells, any parental body cell could be directly passed on to the offspring; as a result, the genetic search meant there was, in fact, *memetic* (without the environment playing any role, though). Therefore the notion of genetic search in Enact completely supersedes its equivalent in that previous work.

   It is also worth mentioning that in Enact's predecessors the built-in selection process was simpler than it currently is such that the artificial life activity could only be observed for apropriately crafted initial configurations.

   Finally, in the former versions of the system the *agents* were referred to as *organisms*. But as the elements of the system became more sophisticated and their usage more abstract, the original term lost its appeal, the more abstract notion of agent appearing more appropriate. Also, my own realisation of the central role of movement to the outcomes of a system run, made the reference to agents seem more appropriate to emphasize the active role of the individuals as they move in the cellular space. All in all, movement is Enact's most fundamental process. In order for selection, reproduction and development to occur, it is necessary for the agents to move; but movement itself does not require them. And while movement can be influenced by

the environment, it does not need the latter either. The ability to move on their own is the primary attribute of the agents' autonomy.

3.13 Summary

In this chapter we described the artificial-life processes and overall dynamics involved in *Enact*, a cellular-automata based architecture of autonomous agents that forms the basis of this thesis. Enact is a family of two-dimensional, non-deterministic cellular automata, whose temporal evolution on a periodic background can be described in terms of the metaphor of an artificial-life world where a population of worm-like agents undergo a coevolutionary process. During their lifetime, the agents roam around, sexually reproducing, interacting with the environment, and being subjected to a developmental process which includes ageing and death.

An agent is formed by a sequence of contiguous cells, so that the cells at each end can be thought of as its head and tail, whereas the cells in between constitute its body. A single cell of the body forms the agent's genotype. Another cell, whose initial state just after neonatal development depends on the agent's gene, represents the phenotype. The remaining cells of the body are fully determined through direct parental inheritance, constituting what we call the agent's memetype.

As a consequence, the coevolutionary process supported in Enact is in general both genetic and memetic. Since the phenotype is what determines the local direction of movement of an agent at each time, and since its initial state depends on the agents' genotype, the genetic coevolutionary process meant above refers, in fact, to the evolution of a coordinated movement of the population. On the other hand, the memetic coevolutionary process is the exploration of the effects of genotypic coevolution, as reflected in the changes that the agent's memetype undergoes.

The mechanics of the processes underlying the system was described in detail, and qualitative issues related to its dynamics were discussed. In particular, it was shown that the overall qualitative dynamics depends primarily on the ageing rate of the individuals, this being very straightforward to tune so as to prevent extinction of the agents or deadlocks due to over-population, and guaranteeing the existence of very long transients.

Enact's rule – its complete set of state transitions – is fairly complex if compared to standard cellular automata in the literature. It should be clear however, that our interest here is not on the emergence of the artificial life activity it supports, but on what can follow assuming its existence as a primitive we can rely on, and to a certain extent, manipulate. In fact, as we will see in the next chapter, Enact can be regarded as a programmable, virtual machine defined by its artificial-life processes, and relying upon its six categories of states.

CHAPTER 4

ENACT AS A VIRTUAL PROGRAMMABLE MACHINE[1]

4.1 Introduction

As mentioned at the end of the last chapter, the second version of Enact was motivated by the development of a framework to support a class of artificial-life worlds. In that context, *framework* was meant to be the complete description of the cellular automata, together with the methodology stating how to use it, and a suggestion of the kind of questions that it might be used to address. In the context of this thesis, however, the latter will only be sketched (in Chapter 6), since our emphasis is on the exploration of Enact as a computing machine.

The description of the system was given in sufficient detail in the last chapter. In the current chapter we present the methodology underlying Enact, that is, how to use it in order to set up particular worlds. As will be clear later on, ultimately this corresponds to regarding the system as a virtual machine that can be programmed.

In the next section we provide an in-depth example of its use, namely, the implementation of a Turing machine as a result of agent-environment interaction. With the example in mind, it is then possible to analyse the idea of Enact as a programmable virtual machine, which the Turing machine is implemented on. In this respect, details are given of how to go about using the system, that is, how to program it. A general discussion on the possibilities of the system is then presented. Finally, some issues relating Turing machines to the context of the present thesis are sketched out, paving the ground for the subsequent chapters.

4.2 Using Enact: Implementation of a Turing Machine

4.2.1 Introduction

In order to illustrate how to go about using Enact, in this section we analyse a particular application, namely, the implementation of a *Turing machine*. This will serve to pinpoint not only practical aspects about how to use the system, but also conceptual issues concerning which kind of applications seems to be appropriate to it.

Basically a Turing machine is constituted by a mobile reading-and-writing head that can travel along an arbitrarily long tape, according to a predefined set of instructions. As the machine obeys one instruction, the head reads the tape symbol it is pointing at and, according to the machine's current state, writes another symbol, possibly the same one just read; then it performs a single movement to the left or to the right. The importance of Turing machines lies in the fact that, despite their simplicity, they define the widest

[1] Most of this chapter was published as [de Oliveira 1993].

class of computations that can be performed by a computing device. Following [Hopcroft and Ullman 1979, page 148] we define a Turing machine as the tuple

$$(Q, \Gamma, B, \Sigma, q_0, F, \delta)$$

where

Q is the finite set of states of the head;
Γ is the finite set of tape symbols used in the computation;
B is the blank symbol ($B \in \Gamma$);
Σ is the set of input symbols ($\Sigma \subseteq \{\Gamma - B\}$);
$q_0 \in Q$ is the start state;
$F \subseteq Q$ is the set of final states; and
δ is the function specifying the behaviour of the head ($\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$).

Note that the function δ is nothing more than the "program" the machine is executing, each application of the function being equivalent to one *step of computation*. After each step of computation a symbol has been written on the tape, the head has entered a new state, and has moved one tape-position to the left (L) or to the right (R).

4.2.2 The Implementation

First of all let us make it clear that in this chapter we will be dealing with only four state categories, using the simplified structure of the agents, as depicted in Figure 3.3.

The idea behind the implementation of the Turing machine is to represent the tape by a sequence of environmental E-states, and the state of the head by an agent (or more precisely, by the first B-state of an agent). We then define a particular agent-environment configuration such that, when it is reached, then one step of computation will be performed. Using the terminology of the preceding section, the computation is a consequence of the state transitions that occur at the (carefully designed) interaction site defined by the agent and the tape. It should be clear that the implementation refers to a single agent interacting with a single tape; therefore, reproduction does not play any role in this context (we return to this point in Subsection 4.3.4).

In order to facilitate the explanation of the implementation it is useful to think in terms of the "hardware" and the "software" of the Turing machine, the former referring to the mechanism that allows the head to move right or left, and the latter referring to the behaviour of the head according to the specification in the function δ.

When thinking about a way to implement the mechanism so as to move the head, one might be led to associate the (moving) agent with the head. However, this is not possible due to the design constraint on Enact that an agent cannot move to the right. To circumvent this problem the implementation uses the idea of placing a marker on the tape representing the head of the machine; incidentally, the same kind of technique is used in [Minsky 1967]. In this way, when an agent is in the appropriate position relative to the marker, a step of computation is performed.

If the computation leads to a rightward movement of the head, the only way the computation can proceed is by the agent returning to a position that it has already passed. Naturally, this can be achieved through the toroidal geometry of the cellular space, i.e., the periodic background it supports.

It is worth observing at this point that the realisation of a Turing machine requires that the tape can be indefinetely extensible, that is, different computations may require arbitrarily long tapes. The way this requirement is dealt with herein is by allowing the size of the cellular array to increase as required by the specific computation at issue.

Turing Machine	Cellular Automaton
Tape	Sequence of E-states
Position of the head	E^*
Blank symbol (B)	E^b
Additional tape symbols ($\Gamma - B$)	$\mathbf{E}_s$ (or $\mathbf{E}_{ss}$)
States of the TM (Q)	B_s (or B_{ss})
Mechanisms to move the head and to perform the computation (δ)	Interaction agent-environment + Periodic background

Table 4.1: Correspondence between the constituent elements of a Turing Machine and the states currently being used to implement it.

Accordingly, the sequence of environmental cells that constitute the tape also acquire the necessary length for the computation to be performed. Although an infinitely long cellular array is not realisable, for all practical purposes it can be as large as required by every particular computation.

There is, yet, the additional problem of dealing with the marker: how do we place it on the tape? This is solved by imposing the requirement that the symbols to be used in the computation be separated on the tape, by single blank symbols; these blanks then provide a 'free' space in the tape which can be occupied by the marker. The situation described is illustrated in Figure 4.1; the correspondences between the constituent elements of a Turing machine (according to the preceding subsection) and their implementation in the current case are shown in Table 4.1. Refer also to the Table for explanation about the notation used in the figure.

Figure 4.1(a) shows snapshots of the same set of the cells as a 3-cell-long agent interacts with the constituent E-states of the tape, performing one step of computation that results in the head moving left; Figure 4.1(b) refers to a step of computation resulting the head moving right. Note, in each figure, the modification of the position of the symbol E^* which is the head-marker. Other features to be noted include:

- It is assumed that the marker E^* is placed on the left-hand side of the next tape symbol to be read by the head.
- The various snapshots represent the stages that are needed for the various operations associated with a step of computation. It is clear that it takes longer to complete the step leading to a leftward move than the step leading to a rightward move.
- The configuration agent/tape that initiates one step of computation is the same in both cases.
- In the sequence of configurations shown, the agent starts moving forward whenever the way ahead is free. But since in general, such a movement is non-deterministic, the configurations are not meant to be immediately consecutive to each other as the figures might suggest.
- Also due to the non-deterministic start of the agent's movement the configuration that triggers a computation step may not be achieved in one passage of the agent

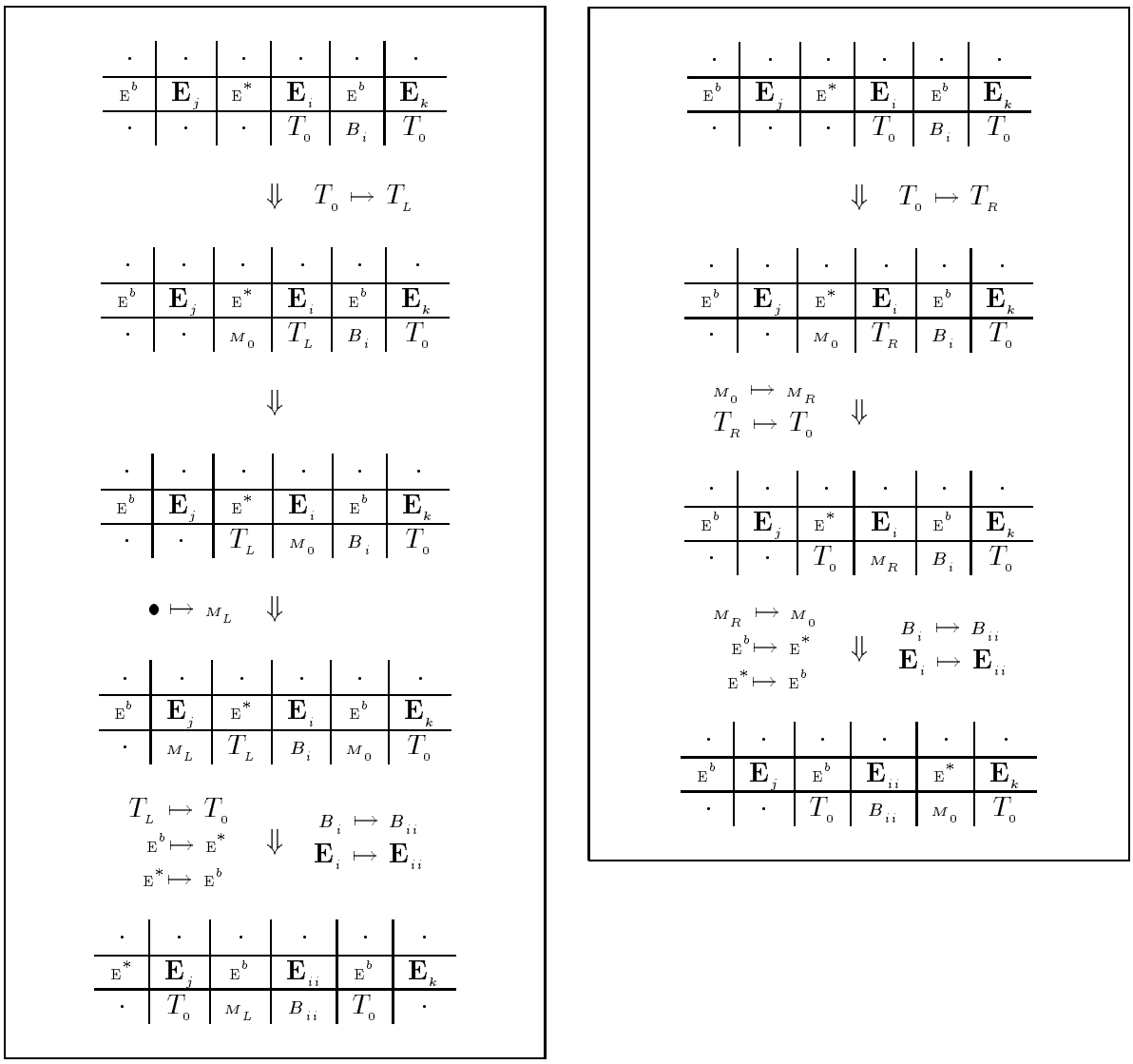


Figure 4.1: Representation of the stages involved in one step of computation of the Turing machine. Each step is defined by the symbol $\mathbf{E}_{ii}$ being written on the tape, the machine entering the new state B_{ii}, and the head then moving to the left (a) or to the right (b). The dots represent the background state; refer to Table 4.1 for additional information on the notation being used.

along the tape. In this case the agent has to return to this triggering configuration by traversing the toroidal cellular space.

- Whenever a computation step has finished, leading the head-marker to move right, the agent has to return to the configuration that triggers a new step. On the other hand, if the computation step has moved the marker to the left, the agent still has the chance to perform another computation step in the same passage. Note this possibility in the last snapshot of Figure 4.1(a).

- All state transitions to take place in the cellular automata at each stage are shown beside the downward arrows that link successive stages.

- The state transitions on the left-hand side of the downward arrows refer to the transitions which are necessary to implement the hardware of the Turing machine. The transitions represented on the right-hand side of the arrows refer to the program being executed.

The list of state transitions of the cellular automata required to implement the hardware of the Turing machine can be seen in the first column of Table 4.3 (shown at the end of this chapter). This column is subdivided into two others, the left-hand subcolumn showing the 4 transitions that are necessary to support the computation step that leads the head to the left, and another subcolumn showing the 5 necessary transitions so that the head can move rightward.

On the other hand, the state transitions which are necessary to characterize one step of computation are shown in Table 4.4 (also at the end of the chapter); note that the table is arranged in three sets of rows. As mentioned above, the 3 transitions involved within each set specify the direction the head will move (L or R), the symbol to be written on the tape ($\mathbf{E}_{ii}$), and the next state of the machine (B_{ii}). The triplet that makes up the first set of rows supports the computation step that moves the head to the left, whereas the triplet in the second set of rows supports the move to the right.

If for a particular input the computation reaches any of the final states, the agent would carry on moving in the cellular space indefinitely. In order to prevent this behaviour, we simply introduce a set of state transitions for selection (according to Section 3.6) which kills off the agent as it leaves the contact with the tape. The last row of Table 4.4 deals with this "halting condition" by defining the state transition as shown. The complete list of state transitions coding for the software of the TM is given in the Appendix C.

Two final observations. First, although an agent with a single B-state was used in the simulation of the Turing machine, it can be seen that the state transitions used for both hardware and software contain no restriction on the number of B-states an agent may have. Second, the design constraint for the actual 'algorithm' used to implement the machine was that it should use the four categories of states, and that it should entail a minimum of new state transitions to be added. However, we are neither making the point that this is the only possible implementation with all the four categories, nor that it would not be possible to implement it with a subset of the four.

4.2.3 The Turing machine in Action: Recognition of a Language

In order to test the implementation, we programmed the Turing machine to recognise the language $\Lambda = \{0^n 1^n \mid n \geq 1\}$; [2] this subsection is based on [Hopcroft and Ullman 1979]: the recognition problem was introduced in page 148, and Table 4.2 and Figure 4.2 were adapted from page 150.

The Turing machine is defined by:

$$
\begin{aligned}
&Q = \{B_0, B_1, B_2, B_3, B_4\},\\
&q_0 = B_0,\\
&F = \{B_4\},\\
&B = \mathrm{E}^b,\\
&\Sigma = \{\mathbf{0}_E, \mathbf{1}_E\},\\
&\Gamma = \{\mathbf{0}_E, \mathbf{1}_E, \mathbf{X}_E, \mathbf{Y}_E, \mathrm{E}^b\},\\
&\delta\text{: defined by Table 4.2.}
\end{aligned}
$$

[2] It can be seen that this is a context-free language. As a consequence, its recognition does not require all the computational power of a Turing machine, a deterministic push-down automaton being sufficient. We use this example just as a matter of convenience.

Current State	Symbol Read				
	$\mathbf{0}_E$	$\mathbf{1}_E$	$\mathbf{X}_E$	$\mathbf{Y}_E$	E^b
B_0	$(B_1, \mathbf{X}_E, R)$	–	–	$(B_3, \mathbf{Y}_E, R)$	–
B_1	$(B_1, \mathbf{0}_E, R)$	$(B_2, \mathbf{Y}_E, L)$	–	$(B_1, \mathbf{Y}_E, R)$	–
B_2	$(B_2, \mathbf{0}_E, L)$	–	$(B_0, \mathbf{X}_E, R)$	$(B_2, \mathbf{Y}_E, L)$	–
B_3	–	–	–	$(B_3, \mathbf{Y}_E, R)$	(B_4, E^b, R)
B_4	–	–	–	–	–

Table 4.2: Transition function (δ) of the Turing machine that recognizes the language $\Lambda = \{0^n 1^n \mid n \geq 1\}$.

Table 4.2 specifies the ten different possible steps of computation. For example, if the machine is in state B_0 and the head reads the symbol $\mathbf{0}_E$ on the tape, it writes the symbol $\mathbf{X}_E$ on the tape, the machine goes to state B_1 and the head moves right; on the other hand, if the current state is B_2 and the symbol being read is $\mathbf{0}_E$, the machine remains in the same state, the same symbol $\mathbf{0}_E$ is written on the tape, and the head of the machine moves one tape-position to the left. Whenever the input string present in the tape belongs to the language Λ, the sequence of steps computed by the machine will necessarily lead it to its final state.

Because 10 distinct steps of computation are necessary for the recognition of the language, we then need $3 \times 10 = 30$ state transitions in order to program this task in the cellular automata. For example, the set of 6 state transitions necessary to support the 2 'instructions' described in the preceding paragraph are shown in the middle column of Table 4.4. Also in the middle column of the same Table, the last element shows the state transition required for killing off ('selecting against', we might say) the agent after it has reached the final state B_4.

The result of running the Turing machine with the input string $\mathbf{0}_E\mathbf{0}_E\mathbf{1}_E\mathbf{1}_E$ is shown in Figure 4.2. After performing the sequence of operations shown there, the machine reaches its final state while the final configuration of the tape is $\mathbf{X}_E\mathbf{X}_E\mathbf{Y}_E\mathbf{Y}_E$; at this point, the agent vanishes due to the selection process, indicating that the original input string has been recognised as belonging to the language Λ. Due to the non-deterministic fashion in which the agent's movement starts, there may be a number of time steps before an agent is able to start its movement ahead; therefore, the sequence of configurations shown in Figure 4.2 should not be considered as immediately consecutive in time.

$$
\begin{array}{lllllllll}
B_0\mathbf{0}_E\mathbf{0}_E\mathbf{1}_E\mathbf{1}_E\mathrm{E}^b & \mapsto & \mathbf{X}_E B_1\mathbf{0}_E\mathbf{1}_E\mathbf{1}_E\mathrm{E}^b & \mapsto & \mathbf{X}_E\mathbf{0}_E B_1\mathbf{1}_E\mathbf{1}_E\mathrm{E}^b & \mapsto & \mathbf{X}_E B_2\mathbf{0}_E\mathbf{Y}_E\mathbf{1}_E\mathrm{E}^b & \mapsto \\
B_2\mathbf{X}_E\mathbf{0}_E\mathbf{Y}_E\mathbf{1}_E\mathrm{E}^b & \mapsto & \mathbf{X}_E B_0\mathbf{0}_E\mathbf{Y}_E\mathbf{1}_E\mathrm{E}^b & \mapsto & \mathbf{X}_E\mathbf{X}_E B_1\mathbf{Y}_E\mathbf{1}_E\mathrm{E}^b & \mapsto & \mathbf{X}_E\mathbf{X}_E\mathbf{Y}_E B_1\mathbf{1}_E\mathrm{E}^b & \mapsto \\
\mathbf{X}_E\mathbf{X}_E B_2\mathbf{Y}_E\mathbf{Y}_E\mathrm{E}^b & \mapsto & \mathbf{X}_E B_2\mathbf{X}_E\mathbf{Y}_E\mathbf{Y}_E\mathrm{E}^b & \mapsto & \mathbf{X}_E\mathbf{X}_E B_0\mathbf{Y}_E\mathbf{Y}_E\mathrm{E}^b & \mapsto & \mathbf{X}_E\mathbf{X}_E\mathbf{Y}_E B_3\mathbf{Y}_E\mathrm{E}^b & \mapsto \\
\mathbf{X}_E\mathbf{X}_E\mathbf{Y}_E\mathbf{Y}_E B_3\mathrm{E}^b & \mapsto & \mathbf{X}_E\mathbf{X}_E\mathbf{Y}_E\mathbf{Y}_E\mathrm{E}^b B_4 & & & & & &
\end{array}
$$

Figure 4.2: Computation involved in the recognition of the string 0011 which is represented here by $\mathbf{0}_E\mathbf{0}_E\mathbf{1}_E\mathbf{1}_E$. The sequence shows, at each step, the symbol configuration of the tape and the state of the Turing machine. The position of the machine state relative to tape symbols represents the position of the head at the corresponding computation step. It is assumed that the head is able to read the tape symbol which is on its right-hand side.

4.3 Methodological Issues

In this section methodological issues are considered associated with the use of Enact. We stress the general aspects of how to use the system in order to set up particular worlds, which leads to the view of Enact as a "programing environment".

4.3.1 Programming Issues

As we have already discussed Enact is family of cellular automata whose common thread is the same overall dynamics that characterizes the artificial-life world. The programming issue in the context of the system is then really one of adding a new behaviour to the family without disrupting the basic artificial-life dynamics already present.

In general, any attempt to add, in a controlled fashion, a new behaviour to cellular automata by adding new state transitions can be a very frustrating task, since the side-effects of a modification of this sort can be very difficult to predict. In the current case, in which we want to preserve the original dynamics, these side-effects may be even more critical. In order to give the reader a hint of what is at issue, I should state that my own experience in programming cellular automata calls to mind the mixed experience of writing programs in a low-level, assembly-like language, with the difficulties of foreseeing all possible outcomes usually associated with a large computational system that has a lot of interdependence between its modules. Naturally, in the context of cellular automata the latter problem arises because the space of possible neighbourhood configurations is so vast, that it becomes extremely difficult to foresee the joint effect of state transitions that have only been individually considered. The consequence is that, very often, special treatment has to be provided to neighbourhood configurations that have not been previously expected.

As a consequence, the only guaranteed, controllable way to add a new behaviour with no undesired side-effects is the process of *instantiation* of the general state categories that appear in an existing state transition, by means of specific states in the category, but that bear relevance to the desired behaviour.

The addition of these transitions, which we refer to as *instantiated transitions*, is the essence of how to use Enact. All state transitions that support the hardware and the software of the Turing machine are examples of instantiated transitions. In terms of their integration with the existing state transitions, they can work either by substitution of their equivalent, more general transitions, or by precedence over the latter, in which case they are placed before the general ones in the actual code of the cellular automata.

Note that the dynamics supported by the set of existing transitions of the system is equivalent to the concept of a virtual machine, like a programming language together with its compiler. Similarly, adding an instantiated transition is analogous to a program that is written on the virtual machine, which has to satisfy the constraints imposed by the constructs of the programming language. In particular, the introduction of an instantiated transition requires knowledge about which state transitions are available; this is like the necessity of having to know the syntax of a programming language before one goes about writing a program in this language.

4.3.2 Implicit and Explicit Instantiated State Transitions

Let us say that a state transition that does not alter the state category of the cell at issue is a category-preserving transition; otherwise, we would have a category-changing transition. Enact can be thought of as a collection of two kinds of state transitions: an *explicit* set, composed of both category-preserving and category-changing transitions, and

an *implicit*, or default set, composed of only state-preserving transitions. Accordingly, in order to add a new behaviour to Enact it is necessary to know the details of what is encoded in the category-preserving and the category-changing state transitions of the system. As Appendix A shows, most state transitions in Enact are category-changing; only the developmental process presents some category-preserving transitions.

The process of adding a new state transition to Enact only has to abide by two criteria:

1. If the state transition is identified as belonging to the explicit set of transitions, it can only be included in the system if it is crafted in such a way that it becomes a perfect instantiation of the more general transition already present in the system.

2. Dealing with the implicit state transitions is easier. All that is necessary is to be sure that a supposedly implicit state transition that is about to be added in, does not really belong to Enact's explicit set. For instance, in Enact any state transition having in the centre cell of its associated neighbourhood an E-state distinct from the background state, is by design implicit.

Let us now turn to the information presented in the rightmost column of Tables 4.3 and 4.4, under the heading of *Instantiated role*. This column simply provides a characterization of "what" is being instantiated by each instantiated transition that is added to the world set-up used to implement the Turing machine. When the new state transition is an instance of an explicit transition, the general transition already present in Enact is expressed in the table. Otherwise (i.e., if the new state transition refers to the implicit set), what is expressed is the typical role that the new state transition plays in a world set-up using Enact.

For instance, the state transitions that have a non-background E-state in the centre cell, are labelled "environmental dynamics" since they are used whenever one wishes to add some state change to these environmental states. Note, in particular, the instantiated role "adult body development" that appears in the second row of Table 4.4. This is an interesting case insofar as it suggests an extension to Enact that could well be implemented. Namely, an extension of the current built-in developmental process – currently involving only the head of an adult agent – so as to also include the development of the adult body. This is a suggestive addition for a future version of Enact.

It is worth observing some aspects about the process of crafting an instantiated transition. For instance, the transition that appears, say, in the first row of Table 4.4 is a deterministic instantiation of a non-deterministic one. Also, no matter how contradictory it may look like at first glance, note that the instantiated state transition at the fifth row of the table is apparently not an instance of the one in the rightmost column. But in fact it is. The point is that the double occurrence of the $_M$-state that the right-hand side transition is explicitly avoiding, never occurs in the situation required for the Turing machine. Hence, for the sake of conciseness of the implementation, the neighbourhood cell involved in the case – the left cell – was allowed to take on the *don't care* symbol.

4.3.3 Revisiting a Previous Work

In [de Oliveira 1992b] we stated that some issues addressed in [de Oliveira 1993] were being revisited. Subsection 4.3.2 was the result of such a revision. What follows is the set of points that were revisited:

- In that paper we referred to the explicit set as *non-quiescent* state transitions, and to the implicit set as formed by *quiescent* state transitions; that is, we defined the

concept of quiescence as related to the preservation (or not) of the state category of the centre cell of the neighbourhood in a state transition. Such an association is not preserved in this thesis and should, therefore, be considered a revision of the notion of quiescence used in that paper; as a matter of fact, a return to the way we first used the term, in [de Oliveira 1992a].

- The notion of "instantiated role" here replaces the notion of "typical role" there. With this substitution, together with the reformulation of the item above, a clearer account of the instantiation process of a state transition was achieved.

- At the time the paper was written, Enact did not yet have a developmental process. Hence, the third column of Table 4.4 had to be revised in order to accommodate the new fact.

4.3.4 On the Possibilities of Enact

Enact has been designed to be fairly general in terms of its being used to set up artificial life worlds; there are, however, several design constraints which would certainly be a burden. For example, only one species is supported by the basic artificial-life world; also, the movement of the agents is very limited. On the other hand, the flexibility provided by the state categories can be explored so as to enrich the basic dynamics in a number of ways by the addition of instantiated transitions; by keeping the latter available for use, they could be seen as libraries of functions, similar to the ones usually available in standard programming languages.

There are, also, two practical problems associated with Enact. First, it is intrinsically expensive in computational terms. Second, as hinted at earlier, programming it may require a great deal of effort, particularly in the sense that it requires the user to be aware of the all neighbourhoods the world set-up at issue will yield. But note that this is not different from a standard programming language, in which very rarely does a program run as expected, without any "bug".

In setting up worlds with the system I have experimented with several options, such as the ones mentioned below:

- A number of direct variations, including agents with distinct states for head and tail, agents whose movement starts deterministically or whose upward body movement is deterministic.

- Introduction of different kinds of heads, with distinctive properties, such as different rates at which they start their movement, or specialisation towards the directions the movement can start.

- Selective mating, for instance from parents whose head movement-properties are somehow related (e.g., being the same). The point here is that, in the basic artificial-life dynamics, mating is in principle just a matter of chance, since it occurs whenever any two agents reach the mating configuration. However, it is possible to create selective mating among the agents; all that is needed to write an instantiated state transition with the parental features that should allow reproduction to start.

- Inheritance manipulation, such as the imposition that the offspring from parents that possess the same feature – e.g., the same movement specialisation – would necessarily inherit this parental feature. Note that in the basic dynamics, reproduction would necessarily imply this kind of inheritance.

- Alternative kinds of interaction sites, such as the one depicted in Figure 3.7. In that case, depending on the "speed" an agent traverses the interaction site, a wealth of outcomes are possible, as will be mentioned in Subsection 6.5.1.

- Various forms of ageing, either unconstrained (i.e., at each iteration of the cellular automaton), or constrained by the occurrence of a predefined situation, such as whenever the agent moves ahead, whenever it is unable to move ahead, or only when it reproduces. In all these cases, death is a natural consequence as the agent reaches an old age.

It is important to note that two agents, with the same initial state configuration of the body cells, may reach completely different configurations after a certain time, because they may have had distinct histories of interaction with the environment and the other agents. And while the agents themselves are very simple, their history of interactions, as we have shown with the Turing machine, can be arbitrarily complex. This feature has an interesting consequence: by characterising the history of agent-environment interactions in terms of computable functions, and constraining the setting of the artificial-life world so that the histories can be mapped to a tractable region of the space of computable functions, it becomes possible for the agents, through their body cells, to act as probes into the emergence of new functions. Such an aspect of emergent computation associated with Enact will be addressed in Chapter 6.

4.4 Turing Machines and Enact

Since the theme of Turing machines in the context of cellular automata will reappear in the following chapters, it is useful to look already at some issues raised by the implementation described above.

But beforehand, it is worth recalling that earlier in Subsection 2.5.3 we made a general discussion on the issue of computability in cellular automata. Also, at that point of the thesis we provided in Table 2.2 the number of states in Enact that would be necessary to implement Minsky's [1967] universal Turing machine – with 4 tape symbols and 7 internal states. Bearing in mind the implementation we have just discussed, in order to implement the former universal Turing machine we can now make it clear that 20 states are then needed, as follows: $4 + 1 + 1 + 1$ E-states; 7 B-states; $1 + 1 + 1$ M-states; and $1 + 1 + 1$ T-states.

From a theoretical point of view, the implementation of the Turing machine in itself is of little relevance, since, as Table 2.2 shows, the literature abounds with examples of cellular automata capable of universal computation. But if we consider the current TM from the perspective – mentioned at the end of the previous section – of using Enact to address the issue of the emergence of computable functions, some aspects of the simulation become theoretically relevant. Firstly, as we hinted at in Chapter 2, because it is couched in an artificial-life system, which is novel, and not in an abstract setting whose relevance would be constrained by the formal aspects it is related to.

Secondly, the simulation of the Turing machine represented the tape as a sequence of E-states, and the agent's first B-state as the state of the machine. One might think however, about the alternative implementation in which the tape would be in the agent as a sequence of B-states, while the state of the machine would be part of the environment. Although this latter scheme was not tried, the experience we acquired in the simulation described strongly suggests that it is very much feasible. We will return to this aspect in Chapter 6.

Thirdly, note that the simulation requires the use of only one agent, the one that will represent the state of the Turing machine. In fact, we could have used a population of agents, but their first B-state would all have to be different from the ones representing the set of states of the machine; in this sense the population would be completely ineffective in terms of the computation being performed. As a matter of fact, not only the concept of (an effective) population is absent from the simulation, but also the built-in concept of reproduction; and, to some extent, the selection process was only partially used. Note also that Turing machines are models of serial computation, whereas cellular automata are essentially parallel devices. The situation of simulating serial computation with a parallel one seems somewhat contradictory, as if resources were being badly used. The point these considerations are driving at is the suggestion that, if it is at all possible to embed some kind of parallel model of computation within Enact, the system itself seems to be providing the clues towards this achievement, namely, the integration of the concepts of population and reproduction with that of agent-environment interaction. This point will be addressed in detail in Chapter 5.

4.5 Conceptual Issues

Artificial-life worlds have been created in various kinds of cellular worlds, as can be seen in [Langton *et al.* 1992]. However, cellular automata have a very distinctive characteristic: the clear notion of *space* provided by the cellular space. Naturally, it is always possible to provide a clearer notion of space in other cellular schemes, but while it would have to be implemented in these schemes, it is intrinsic to the definition of cellular automata. Now, from our point of view, the major achievement of the system is its provision for a unification of all these distinct processes which are necessary for the artificial-life activity. In other words, all processes mentioned previously, namely,

- reproduction;
- selection;
- agent-environment interaction;
- agent-agent interaction;
- dynamical environment, and
- development;

are all integrated through the same underlying dynamics. And the primary reason for that is that all processes are mediated by the *same space*, the cellular space. It would have been much more difficult to achieve such a feature in another conceptual framework, where the notion of space would not be as distinctively clear as in cellular automata. A promising possibility of this unification is that the mathematical analyses of the processes going on in a world set-up may become significantly more tractable.

As advanced in Chapter 2 Enact has been inspired by the work of Varela, such as in [Varela 1989], in terms of some high-level notions implied by the latter, which Enact, as an artificial world attempted to reflect. These include: the dynamic nature of the processes involved in natural phenomena; order arising from the self-organising properties of the dynamics, rather than from the concept of a predefined problem whose solution is evaluated by some, similarly predefined, fitness measure; and the non-separability of agents and their environment, due to their joint historical coupling. Implicitly these themes will reappear in the forthcoming chapters. But only in the background.

4.6 Summary

In this chapter we showed how Enact can be conceived of as a programmable virtual machine, and how to go about programming it.

By providing an in-depth discussion on the implementation of a Turing machine as a result of the interaction agent-environment, all necessary issues involved in the use of the system for setting-up artificial worlds have been addressed. In particular, we have discussed the central concept underlying its use, namely, the addition of instantiated state transitions. As far as I am aware, Enact is the first cellular-automata-based system to support the aspect of programmability in such an explicit way.

As pointed out earlier, in order to use the system knowledge of the high-level issues discussed is not sufficient; an understanding of the role of the state transitions in the global dynamics is also important. In this respect, it should be clear that we did not intend to provide an "user's manual" about the set of state transitions of the system.

Based on the implementation of the Turing machine we then started the discussions involving Enact and computations, paving the way for the next chapters.

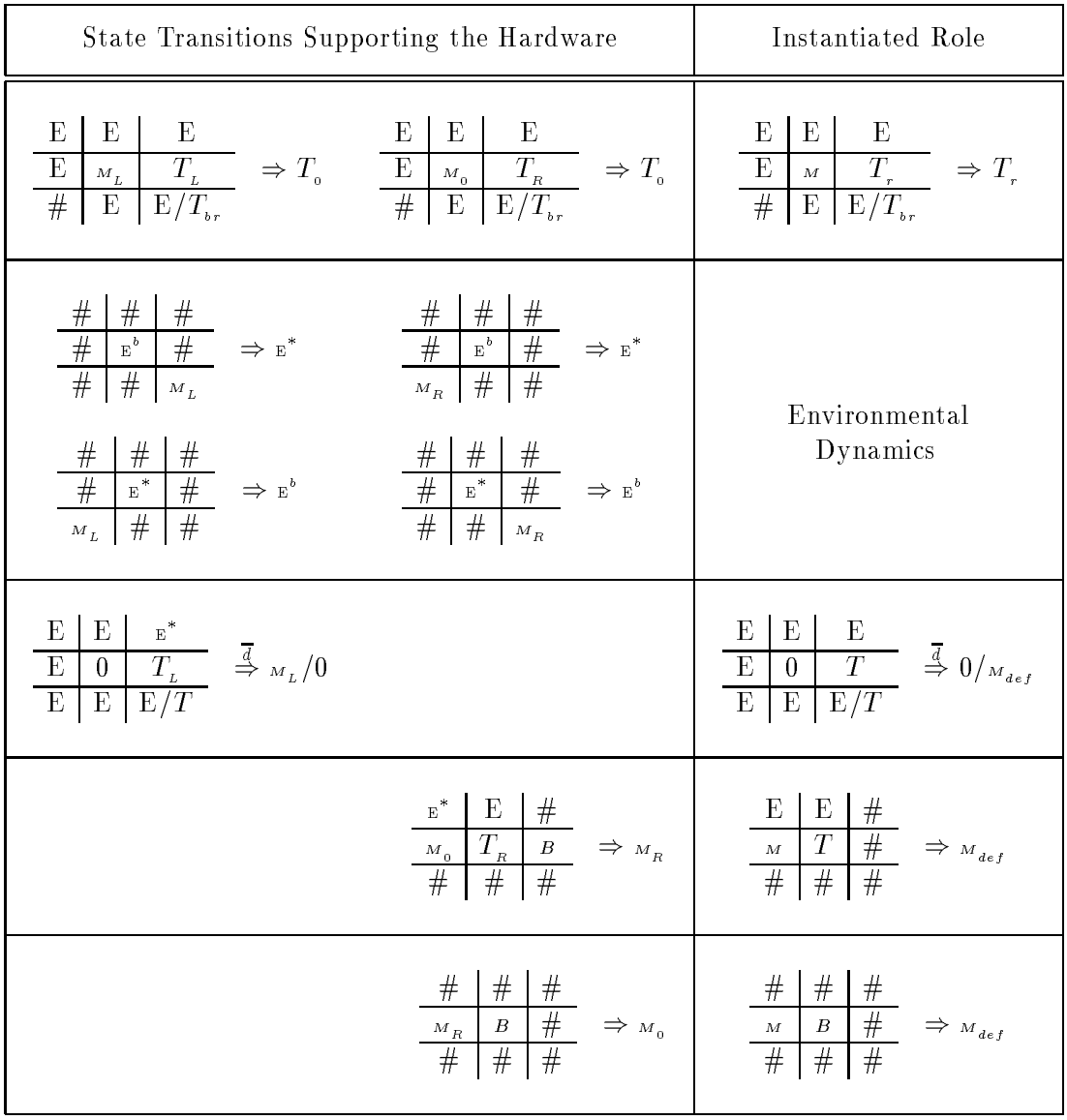


Table 4.3: State transitions supporting the hardware of the Turing Machine. The transitions in the first subcolumn of the first column support the mechanism that allows the head to move leftward, while the transitions in the other subcolumn allows the rightward move of the head. The cells marked with the symbol # mean that their state is irrelevant in these neighbourhoods. The subscript *def* refers to the default value used in the cellular automata. The subscripts *r* and *br* refer to the geographic position of the cell in its neighbourhood. The background state is represented by *0*.

General Transitions	Example	Instantiated Role
$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{E}_i & \# \\ \hline 0 & T_0 & B_i \\ \hline \# & \# & \# \end{array} \Rightarrow T_L$	$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{0}_E & \# \\ \hline 0 & T_0 & B_2 \\ \hline \# & \# & \# \end{array} \Rightarrow T_L$	$\begin{array}{c\|c\|c} E & \# & \# \\ \hline E & T_c^* & M/B \\ \hline \# & \# & \# \end{array} \overset{\bar{d}}{\Rightarrow} T_c/(T_c^+/0)$
$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{E}_i & \# \\ \hline T_L & B_i & M_0 \\ \hline \# & \# & \# \end{array} \Rightarrow B_{ii}$	$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{0}_E & \# \\ \hline T_L & B_2 & M_0 \\ \hline \# & \# & \# \end{array} \Rightarrow B_2$	Adult Body Development
$\begin{array}{c\|c\|c} \# & \# & \# \\ \hline {}^{E^*} & \mathbf{E}_i & \# \\ \hline T_L & B_i & M_0 \end{array} \Rightarrow \mathbf{E}_{ii}$	$\begin{array}{c\|c\|c} \# & \# & \# \\ \hline {}^{E^*} & \mathbf{0}_E & \# \\ \hline T_L & B_2 & M_0 \end{array} \Rightarrow \mathbf{0}_E$	Environmental Dynamics
$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{E}_i & \# \\ \hline 0 & T_0 & B_i \\ \hline \# & \# & \# \end{array} \Rightarrow T_R$	$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{0}_E & \# \\ \hline 0 & T_0 & B_0 \\ \hline \# & \# & \# \end{array} \Rightarrow T_R$	$\begin{array}{c\|c\|c} E & \# & \# \\ \hline E & T_c^* & M/B \\ \hline \# & \# & \# \end{array} \overset{\bar{d}}{\Rightarrow} T_c/(T_c^+/0)$
$\begin{array}{c\|c\|c} \# & \mathbf{E}_i & \# \\ \hline \# & M_R & B_i \\ \hline \# & \# & \# \end{array} \Rightarrow B_{ii}$	$\begin{array}{c\|c\|c} \# & \mathbf{0}_E & \# \\ \hline \# & M_R & B_0 \\ \hline \# & \# & \# \end{array} \Rightarrow B_1$	$\begin{array}{c\|c\|c} \# & \# & \# \\ \hline \neq M & M & B_r \\ \hline \# & \# & \# \end{array} \Rightarrow B_r$
$\begin{array}{c\|c\|c} \# & \# & \# \\ \hline \# & \mathbf{E}_i & \# \\ \hline \# & M_R & B_i \end{array} \Rightarrow \mathbf{E}_{ii}$	$\begin{array}{c\|c\|c} \# & \# & \# \\ \hline \# & \mathbf{0}_E & \# \\ \hline \# & M_R & B_0 \end{array} \Rightarrow \mathbf{X}_E$	Environmental Dynamics
$\begin{array}{c\|c\|c} 0 & 0 & 0 \\ \hline T_0 & B_F & T_0 \\ \hline 0 & 0 & 0 \end{array} \Rightarrow 0$	$\begin{array}{c\|c\|c} 0 & 0 & 0 \\ \hline T_0 & B_4 & T_0 \\ \hline 0 & 0 & 0 \end{array} \Rightarrow 0$	Selection

Table 4.4: General representation of the state transitions of the Turing machine. The rows are shown in three sets; the first set refers to the head moving left, while the next refers to the rightward movement. The isolated transition on the bottom shows the halting condition, which should apply to each and every final state B_F. The cells marked with the symbol # mean that their state is irrelevant in these neighbourhoods. The subscript r refers to the geographic position of the cell in its neighbourhood. The background state is represented by *0*.

CHAPTER 5

COLLAPSING A COEVOLUTIONARY PROCESS INTO A COMPUTABLE FUNCTION[1]

5.1 Introduction

Aligned with the recent tendency towards the conception of computational systems gleaned from biology, and the use of computational explanations in biological reality, a model of computation was developed that is deeply entrenched in the concepts typically associated with evolutionary computation. This chapter will present this result.

Expanding on the Turing machine implemented in Chapter 4, this chapter rebuilds it, but now relying on the entire population of agents. The intent is to show how the entire artificial life activity of the system could be considered as a computational machine.

Although the model will be presented in the context of Enact, its conceptual basis is not restricted to the current implementation. In addition, without loss of generality the presentation will be centred on the computation of a *function*, that is, a computation that requires an input and such that, each valid input always yields a uniquely determined output. In this chapter, any computation will be considered misperformed if, for a *valid* input (a "question" that has a "correct" answer), the corresponding output is not returned (the "correct" answer is not entailed). Finally, since the model of computation will be based on a modified Turing machine, it should be clear that any computation will be performed by means of a sequence of *computational steps*, represented by each triplet of the state transition table of the corresponding machine.

The chapter is organised as follows. The next section and the subsequent present the model; firstly, as an overview, and secondly, in detail. The latter is the implementation of the model in Enact, realised by a mechanism based on the components of the same Turing machine from Chapter 4. Finally, we sum up the discussions by emphasising the most significant features of the model, in particular how it can give rise to a model of coupled computations.

5.2 The Model of Computation at a Glance

The model of computation will be presented by implementing a particular function that will be defined in terms of a modified Turing machine. The process can be roughly described as follows. Initially the cellular space contains a population of identical agents and a set of interaction sites. Each agent's memetype represents the input of the function being computed; the phenotype represents the state of the Turing machine, and is also related to the way the agent moves. Each interaction site provides the integration between

[1] A version of this chapter was accepted as [de Oliveira 1995].

Current State $(\mathbf{E}_{ii})$	Symbol Read $(K_i^*) \mapsto (\mathbf{E}_{ii}, K_{ii}, LR)$					
	$\mathbf{0}_K$	$\mathbf{1}_K$	$\mathbf{X}_K$	$\mathbf{Y}_K$	K^b	K^f
E_0	$(\mathrm{E}_1, \mathbf{X}_K, R)$	$(-,0,-)$	$(-,0,-)$	$(\mathrm{E}_3, \mathbf{Y}_K, R)$	$(-,0,-)$	$(-,0,-)$
E_1	$(\mathrm{E}_1, \mathbf{0}_K, R)$	$(\mathrm{E}_2, \mathbf{Y}_K, L)$	$(-,0,-)$	$(\mathrm{E}_1, \mathbf{Y}_K, R)$	$(-,0,-)$	$(-,0,-)$
E_2	$(\mathrm{E}_2, \mathbf{0}_K, L)$	$(-,0,-)$	$(\mathrm{E}_0, \mathbf{X}_K, R)$	$(\mathrm{E}_2, \mathbf{Y}_K, L)$	$(-,0,-)$	$(-,0,-)$
E_3	$(-,0,-)$	$(-,0,-)$	$(-,0,-)$	$(\mathrm{E}_3, \mathbf{Y}_K, R)$	(E_f, K^b, R)	$(-,0,-)$
E_f	$(-,0,-)$	$(-,0,-)$	$(-,0,-)$	$(-,0,-)$	$(-,0,-)$	$(-,-,-)$

Table 5.1: Transition table governing the sequence of steps of computation that allow the associated Turing machine to recognise the language $\Lambda = \{\mathbf{0}^n\mathbf{1}^n \mid n \geq 1\}$. The 0-state corresponds to Enact's background state and should be distinguished from $\mathbf{0}_K$, the memetype state that represents the character **0** of the language. The third element of the triplets stands for the head moving to the left or to the right in the corresponding step of computation.

the TM state and the tape configuration, thus enabling the computation to be performed. At each interaction site one step of computation can be performed. In parallel to this process the way the agents move may be altered at each site.

Depending on the world set-up, as defined by the experimenter, it may be that the computation is not correctly performed; by including appropriate transitions for selection, all the agents related to these incorrect computations are killed off. Analogously, if an agent is subjected to the correct computation, it becomes immortal, and therefore, insensitive to any further interaction site. Therefore, there is an unescapable interdependence on each other's pattern of movement in the sense that, whenever an agent has survived it must have gone through a coupled history of movement that allowed its being in the right interaction site at the right step of computation. Whenever an agent has survived and is no longer affected by any interaction site, the function has been computed and its result is to be found in the agent's current memetype.

If the initial population eventually becomes extinct, a new trial has to be started, with a different initial condition, i.e., different number of interaction sites or initial agents; different spatial disposition of sites or initial population; or a different set of interaction sites. It should be remarked that the computation is only performed *in the limit*, that is, as a population vanishes another has to be tried, and so on, without any a priori guarantee of when (at least) one immortal agent (as defined above) will be found. But if the computation is correctly performed, the immortal agent will be found.

5.3 The Model of Computation in Detail

In this section we provide a detailed description of the model of computation underlying Enact. We go about it by implementing a mechanism that computes a particular function in terms of a Turing machine computation; the generalisation to an arbitrary function is straightforward. Initially we describe the mechanism that enables a state change of the TM; and then, the mechanism that supports the manipulation of the TM tape and head. Afterwards, these mechanisms are applied to the implementation of the function that recognises the language $\Lambda = \{0^n1^n \mid n \geq 1\}$. It is worth observing that this function is the same one used in Chapter 4, where it was used in the context of exemplifying the

Turing Machine	Implementation in Enact
Tape	Memetype
States of the TM	Phenotype
Position of the head	K^*
Blank symbol of the tape	K^b
Mechanisms to move the head and to perform the computation	Agent-environment interaction in periodic background

Table 5.2: Correspondence between the constituent elements of a Turing Machine and the states currently being used to implement it.

usage of Enact.[2]

5.3.1 Transforming the State Transition Table

In order to implement a TM according to the requirements of the current implementation, the original state transition table of the function being implemented (as presented in [Hopcroft and Ullman 1979]) has to undergo a transformation. Table 5.1 shows the outcome of the transformation. The main aspects to be noted are as follows.

First, all state transitions that do not belong to the state transition diagram of the original function are represented by $(-, 0, -)$. They represent the steps of computations that do not belong to the pathways that lead to the correct computation of the function. When an agent performs one of these steps of computation, the agent has gone through a developmental pathway that will not lead to the end of the computation; therefore, the agent should be killed off. For this reason the written state on the tape is "0", the background state, which, by means of the built-in selection, automatically exterminates the agent.

Second, there is an extra column in the table when compared with the original state transition table (from [Hopcroft and Ullman 1979]). It corresponds to the additional tape symbol K^f, that is linked to E_f. E_f is the unique final state of the TM. K^f is a tape symbol that should be already in the tape, which will be reached when the final state E_f is achieved. When the computation reaches this combination of symbols the computation has finished, and therefore neither the TM state nor the content of the tape should be allowed to be modified further. This situation is represented in Table 5.1 by the occurrence of $(-, -, -)$ at the intersection of K^f and E_f.

5.3.2 General Aspects

The implementation presupposes that the state of the Turing machine is represented by the agent's phenotype, whereas the agent's memetype represents the tape; this and other correspondences between a TM and its current implementation in Enact is shown in Table 5.2.

[2]Which was done by implementing a Turing machine where a single agent, representing the state of the machine, interacted with a tape which was built out of a sequence of environmental states. In addition, the agents were much simpler, since their bodies were made-up of an undifferentiated type of cell.

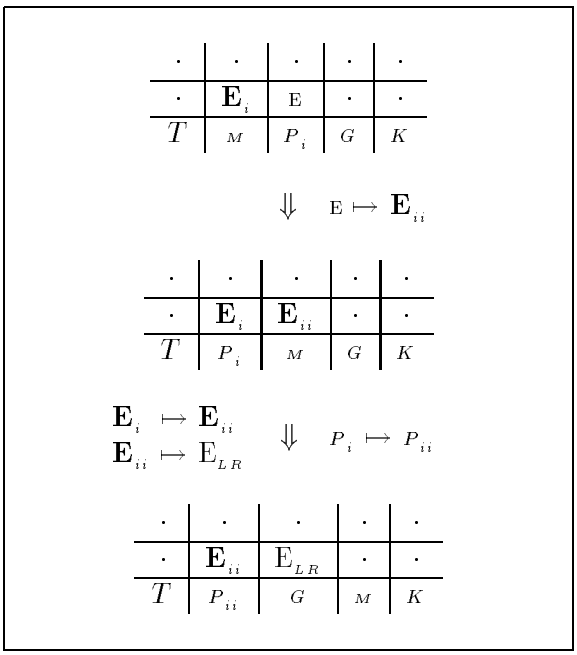


Figure 5.2.a

Figure 5.2.b

Figure 5.1: Representation of the first phase of a step of computation. The phase is characterised by a state change in the associated Turing machine of the computation. The dots represent the background state.

The requirement for indefinitely extensible tape – as first discussed in Subsection 4.2.2 – also has to be raised herein. In the current context this requirement is achieved by using an arbitrarily long agent (with consequent arbitrarily long memetype), according to the specific computation at issue. Naturally, there is no problem in achieving that, since the dynamics of Enact can handle agents with arbitrary size.

As the computation is performed it is necessary to mark the position of the head in an agent's memetype. Unlike the way we proceeded in Chapter 4 (where a special symbol moved along the agent's body so that its position at any time indicated the next tape symbol to be read), in the current case we point at the symbol to be read by means of the association of each κ_i-state of the memetype, to a corresponding κ_i^*-state that marks the head position on that memetype cell.[3]

In order to make the explanation clearer we divide the operation of the machine into two phases: one, where the state of the associated Turing machine is changed; and the other, where a symbol is written on the tape and the position of the head is modified. The former is illustrated in Figure 5.1 and the latter in Figure 5.2. For present purposes, not each and every detail of the implementation is relevant; the reader is recommended to find in Chapter 4 all the information needed to fully understand the details of the

[3]Such a change of approach yields a neater implementation because it illustrates another style of programmimg, and because it is clearer than the one the former approach would have led to; however, it is less economical due to the increased number of states.

Figure 5.1

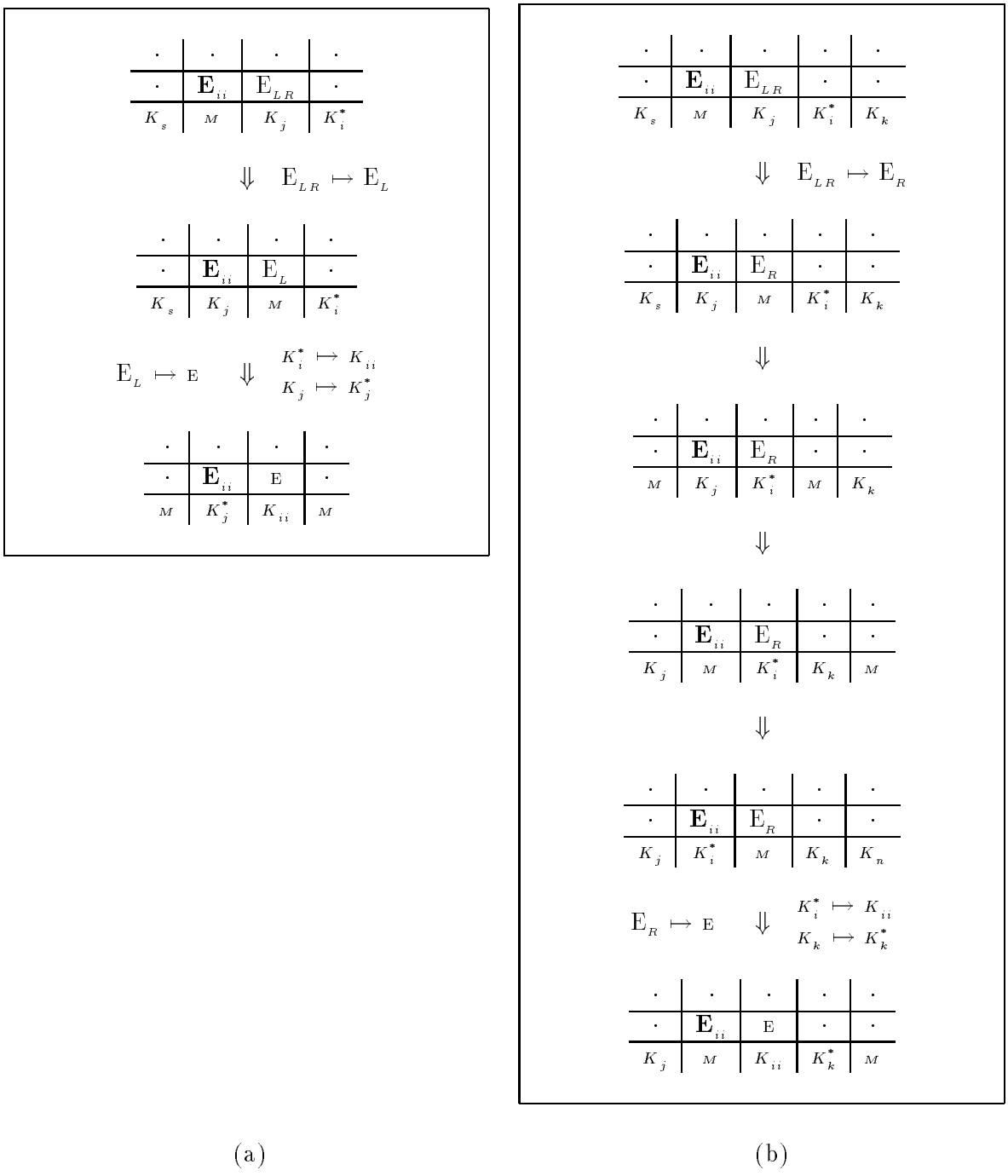


Figure 5.2: The second phase of a step of computation, where a symbol is written on the tape of the Turing machine and the head of the machine moves a step on the tape. The dots represent the background state.

implemented mechanisms. Therefore, it is sufficient to note the following:

- Both figures feature snapshots of the same set of cells as an agent interacts with a set of E-states. The various snapshots represent the stages that are needed for the various operations associated with a step of computation.
- All alterations that occur in the agents take place at interaction sites composed of a horizontal pair of contiguous environmental cells. The state (E) of the right-hand cell is the same for all sites and, although it is modified as one step of computation is performed, the cell returns to the original state after the end of this step. The state of the left-hand cell may be modified as a result of the step of computation. The right-hand cell of the interaction site will be referred to here as the *template* cell.
- All state transitions to take place at each stage are shown beside the downward arrows that link successive stages. The state transitions on the left-hand side of the downward arrows refer to the ones which are necessary to implement the "hardware" of the Turing machine. The transitions represented on the right-hand side refer to the "program" being executed.
- All bold-faced environmental states refer to state transitions defined by the experimenter. All the others are auxiliary states used as part of the machinery that allows a step of computation to be performed.
- Whenever a memetype state becomes the next tape symbol to be read, it becomes "starred", e.g., K_i becomes K_i^*.
- Whenever a a doubled letter is used as the subscript of a state (as in P_{ii}), this means that the original state, before the state transition, was characterised by the equivalent state with a single letter subscript (i.e., P_i).

5.3.3 State Change of the Turing Machine

Figure 5.1 shows the state change of the TM. Whenever an interaction site relates to an agent in a state configuration as shown at the first step of the figure, the state change process of the TM is triggered.

According to the agent's phenotype, E is changed into $\mathbf{E}_{ii}$ and eventually substitutes the original $\mathbf{E}_i$ of the interaction site. The former state change should take place according to a predefined mapping which should preserve the diversity of $\mathbf{E}_i$ states originally present in the space; the actual mapping used in the current implementation is shown in Table 5.3. At the same moment, the template cell changes into the state E_{LR}, setting the conditions for a move of the TM head later on. The latter will happen, or not, depending on the step of computation in consideration.

P_i	$\mathrm{E} \mapsto \mathbf{E}_{ii}$
P_0	E_0
P_1	E_1
P_2	E_2
P_3	E_3
P_f	E_f

Table 5.3: Environmental dynamics from E to $\mathbf{E}_{ii}$. $\mathbf{E}_{ii}$ is a mirror of P_{ii}.

$\mathbf{E}_i$	$P_i \mapsto P_{ii}$				
	P_0	P_1	P_2	P_3	P_f
E_0	P_1	P_2	P_2	P_f	P_f
E_1	P_3	P_1	P_0	P_3	P_f
E_2	P_1	P_2	P_2	P_f	P_f
E_3	P_3	P_1	P_0	P_3	P_f
E_f	P_0	P_1	P_2	P_3	P_f

Table 5.4: Adult phenotypic development of the agents $P_i \mapsto P_{ii}$. The new phenotypic state P_{ii} is given for each value of $\mathbf{E}_i$ and P_i. Note that E_0 and E_1 are respectively equivalent to E_2 and E_3; also, that P_f is always preserved, i.e., not affected by $\mathbf{E}_i$.

The state change of the TM also occurs during the last stage shown in the figure. This is the state change of the agent's phenotype, by virtue of its current state and the original $\mathbf{E}_i$ of the interaction site. Table 5.4 shows the details of such a state change as used in the implementation. In order to prevent too strong a selective pressure over the agents, the actual state transitions were chosen so that all of them are circumscribed to the state transition diagram of the TM; therefore, the actual state transitions of the P-states, as expressed by the columns of the table, conform to the state transition diagram of the TM, as shown in Figure 5.3.

Since it is under the presence of $\mathbf{E}_{ii}$ that all actions on the tape are performed (including the displacement of the head), $\mathbf{E}_{ii}$ has to unequivocally mirror P_i; otherwise, those actions would not be performed as they should, according to the step of computation in consideration. For this reason the mapping in Table 5.3 is one-to-one.

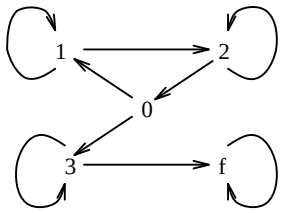


Figure 5.3: Representation of all the possible state transitions.

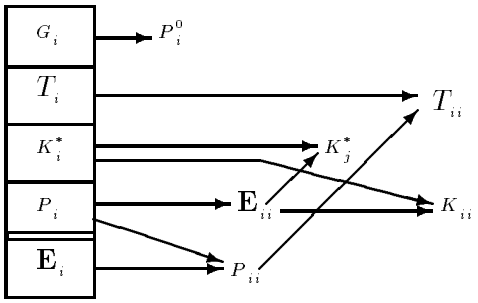


Figure 5.4: Dependence between the states involved in two successive steps of computation.

5.3.4 Tape and Head Manipulation

Figure 5.2 shows the details of the manipulation of the TM tape and head. Figure 5.2.a refers to a leftward movement of the head, while Figure 5.2.b refers to a rightward movement. The former case is characterised by the symbol E_L which E_{LR} is changed into, while in the latter case E_{LR} gives rise to E_R.

We see that both situations depicted in the figure present a high degree of analogy with each other. Both processes start with the same kind of state transition, the one that triggers the head movement in the appropriate direction. Also, after a step of computation has finished, E is restored in both cases, as mentioned earlier in this section. Finally, only

at the last stage of the process is the new position of the TM head defined, and the new symbol written on the tape. It takes longer to accomplish the right-hand side process because the agent has to be in an adequate position with respect to the template cell, so as to create the conditions for the desired actions on the tape.

The state transitions that implement the mechanisms described above are summed up in Tables D.1 and D.2 (both shown in Appendix D.1), which respectively account for the hardwired mechanisms and for the state changes that are specifically determined by the state transition table of the function being computed. The state transitions which are centred in the first column of those tables apply to both the rightward and leftward movement; the ones the appear only on one side (right-hand or left-hand) uniquely apply to the movement corresponding to this side. The occurrence of the symbol $\mathrm{E}_{\overline{LR}}$ which has been omitted here so far, should be noticed in the tables. It is a state of the template cell that results from E_{LR}, when the head of the TM should not move in either direction (left or right), corresponding to the cases $(-,0,-)$ and $(-,-,-)$ as shown in Table 5.1.

The dependences among the state changes associated with an agent and with the interaction sites, as discussed in the current and the previous subsections, are shown in a compact form in Figure 5.4. The figure displays those states according to the others they depend upon. For instance, P_{ii} is represented as depending on $\mathbf{E}_i$ and P_i, following the details of Table 5.4.

5.3.5 Halting Condition

At each environmental interaction the action on the tape will always be performed according to the current state of the agent, as defined by its phenotype. But the state change is determined by the $\mathbf{E}_i$-state of the interaction site. If the latter state is one that leads to a phenotype state that does not match the corresponding actions of the tape and head, then this step of computation will have been misperformed. The next environmental interaction of the resulting agent will therefore correspond to the $(-,0,-)$-triplet of the state transition table; consequently, selection will wipe out the agent as a result of that interaction.

The only way an agent can survive an interaction with an arbitrary site is the situation where its phenotype state is one of the final states of the computations, and the position of the head is in the memetype final state. In Table 5.1 those states correspond, respectively, to E_f and K^f, and the situation itself corresponds to the $(-,-,-)$-triplet.

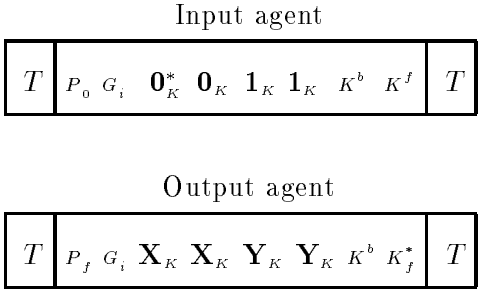


Figure 5.5: State configuration of agents that represent a correct input and the corresponding output of the computation. The input agent represents a string that is recognised as belonging to the language Λ.

From this it becomes clear that, whenever the computation has finished, some agent has become "immortal". But it may be the case that one or more agents reaches a pattern of movement that keeps them away from visiting any interaction site. Even if these agents are not related to the end of a computation, in these situations they become immortal;

but it should be remarked that this in fact a condition of pseudo-immortality, insofar as the agent would not survive another interaction. The final test for checking the end of the computation is, therefore, to subject the "suspected" agent to any interaction site.

Figure 5.5 shows an agent that represents an input string that belongs to the language Λ, and the same agent after it has been subjected to a successful computation, i.e., the string was accepted as belonging to Λ. We denote that input a *correct* input; and the corresponding agent, the input agent. Whenever an output agent, like the one shown, appears, its initial configuration has necessarily been that of the input agent; however, the opposite does not hold: the existence of an agent with the correct input does not imply that it will always "develop" into the output agent shown. Naturally, the latter will depend on its lifetime history of interactions. Hence, the only way to perform a computation is by continuously re-initialising the cellular space, so that, at a certain time, an immortal agent is given rise to. At each re-initialisation the possibilities are the modification of the initial number of agents and interaction sites, their spatial distribution, or the sites themselves. Evidently, at each re-initialisation the initial agents are preserved, since they constitute the input data for the computation.

The previous issue leads to the matter of making explicit in which situations should the experimenter re-initialise the cellular space. There are two of them:[4]

- *Environmental Damage.* There may be irreversible damage of the environment, a suggestive metaphor for the situation that would block the agents' development in such a way that none of them would ever be able to develop into the immortal individual that characterises the end of the computation. In the context of the implementation this may occur because, as a result of the state transitions, the initial set of E_i states may change in such a way that can preclude a certain $P_{i,i}$, required for a particular step of computation, from appearing in an agent.

- *Population extinction.* An entire population may disappear due to the individuals following computational pathways that do not lead to the end of the computation. This is due to the difficulty of an agent visiting the right sequence of environmental sites in the right order.

Summing up, the computation is achieved if and only if, for some initial configuration of the cellular space, at least one immortal agent emerges, one that is able to survive to a sequence of interaction sites of an arbitrary number. As long as that individual is not found the cellular space has to be re-initialised.[5] There is no way to know beforehand, for an arbitrary input agent, when the immortal individual will be found, that is, when the computation will finish. In other words, the computation is only achieved in the limit of successive re-initialisations of the cellular space.

5.4 The Character of Reproduction

5.4.1 Impossibility of Automatic Generation of Inputs

Re-initialisations are necessary only to provide input agents that represent a correct input. In these occasions one could be led to think that reproduction could be adequately set up

[4] The condition of pseudo-immortalily mentioned earlier requires the system to be re-initialised. However, this situation can be avoided by properly designing the cellular space, being sufficient to place the interaction sites in such way that each line and each row of the cellular space have at least one interaction site. Because this situation can be totally avoided it has no major implications.

[5] It is not really necessary to wait for the entire population to vanish before adding new input agents to the process; an alternative strategy is to keep feeding them into the cellular space, analogously to a steady-state genetic algorithm.

in order to allow any newborn to be created with the required input configuration; this would provide an automatic means for generating input agents, freeing the experimenter from the task. The process would carry on until an immortal agent came about.[6]

Reproduction would act by alleviating the problem of population extinction mentioned above. The more effective its action would become, the less frequent the population would get extinct prior to the emergence of an immortal individual.

With the ability to continuously re-initialise itself, the autonomy facet of the computational process would significantly increase. In Enact, however, such an idealised situation is not generally possible. This is a consequence of the way reproduction takes place: for example, the locality constraint imposed by the neighbourhood being used avoids more than three adjacent body cells of an agent appearing in any state transition. If, on the contrary, the neighbourhood of the cellular automaton did not have such a kind of restriction, reproduction could be arbitrarily directed towards the creation of any predefined newborn.

In a generalisation of the model, so as to remove this particularity (among others) of the implementation, the role of reproduction could be extended as above. But despite the gain in autonomy that would accrue, the manual action of the experimenter would still be needed. After all, the problem of environmental damage, being fully due to the history of interactions between the agents with themselves and with the environmental sites, is beyond the experimenter's control.

5.4.2 Automatic Generation of Non-Intermediate Steps of Computation

Notwithstanding the impossibility aforementioned, it is possible, however, to ensure the automatic generation of newborn agents that will have, among them, others that represent the correct input. Even stricter than that, it is possible to guarantee that, additionally, no newborn will be generated that represent an intermediate step of computation. The latter condition is already partially guaranteed by selection insofar as the $(-, 0, -)$-triplets ensure the pruning of all computational pathways that do not belong to the computation. What is still lacking is a way to also prune all pathways that belong to the computation but that did not start at its initial configuration.

Consequently, all newborn agents whose state configuration does not conform to the initial state-tape configuration of the corresponding TM should not be able to survive. In terms of Table 5.1, the specific conditions that would have to be guaranteed are that it must be possible to create any newborn with state configuration corresponding to the topmost, leftmost element; and that all the others must not survive a visit to any interaction site.

In order to ensure that the newborn's phenotype has the initial TM state, it is sufficient to instantiate the reproductive state transition that creates the phenotype, so as to impose that only the initial TM state be created. In order to ensure that the tape is allowed to have the initial (correct) configuration, it suffices to guarantee that the initial tape symbol that the TM head should read at the start of the computation, be created in a uniquely determined way. For example, let us consider the recognition problem discussed so far. The latter condition can be translated into the state transitions shown in Figure 5.6. The first is the instantiated form of another, original from Enact, that creates the first memetype cell; the state transition means the corresponding cell can only take on the state $\mathbf{0}^*_K$, which represents the head position at the start of the computation. All the other memetype cells can assume any state, except the "starred" ones, since they represent the initial head position being on another state different from $\mathbf{0}^*_K$. Note that the state

[6] The immortal agent should be made sterile to prevent reproduction from being pushed any further.

transitions shown do not guarantee that all newborn agents will represent the correct input. But the ones that do not, will certainly be killed off as they visit an interaction site.

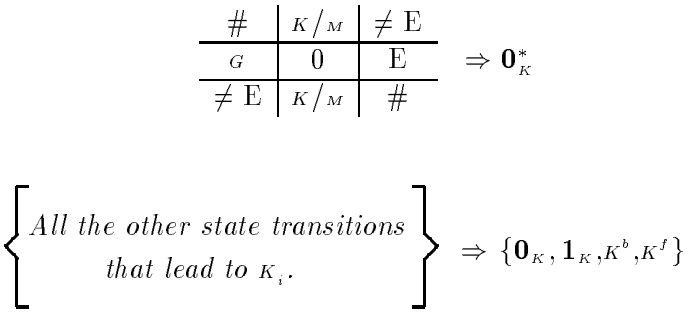


Figure 5.6: Specification of a reproductive process that would yield newborn agents that represent the initial state of the computation. All the ones representing intermediate steps are prohibitted. The # represents any state category.

This aspect of reproduction can be generalised for an arbitrary computation. The rationale is the same as for the example above. The remaining point is how to ensure that the initial tape symbol the TM head should read, be created in a uniquely determined way. The first state transition of Figure 5.6 ensures this condition for the example above because the presence of the genotype leaves no room for any ambiguity to occur. A fact that can be verified for this example is that the leftmost position of the TM head is precisely the position of the first symbol being read at the beginning of the computation; this is the feature that is really being explored here. Although this feature does not hold true for an arbitrary computation, it is always possible to modify the state transition table of the original computation, so as to reflect that; it is this modification that allows the generalisation meant above. The fact that the new, modified table will have additional states or tape symbols is just of a practical consequence, in terms of the computational process becoming more inefficient. But the ability to perform the computation is not altered.

5.5 Implication

5.5.1 From a Model of Computation to a Model of Coupled Computations

We have discussed the implementation of a population of movable agents that undergoes an individual dynamical process of development, whose main feature is the dependence upon their lifetime history of interactions with each other and with the interaction sites present in their world. Some developments are prohibited by means of a predefined, built-in selection process; any agent that follows these developmental pathways is killed off. The ones that manage to avoid them, become immortal. Whenever a population vanishes, another has to be created, and the process iterated until an immortal agent emerges. When this happens we can say that the surviving agent was subjected to a dynamical process that can be interpreted as a computation. If the computation is a function, then its result is to be found in the agent's memetype. This function is nothing more than a collapsed representation for the entire dynamical process defined by the artificial-life processes and the action of the experimenter.

But there is another possibility. Instead of fixing the state transition table and the inputs in order to obtain appropriate outputs, one could take the alternative perspective of trying to infer the function that can be associated with a set of outputs, given a prior knowledge of the inputs. Putting it in other terms, by collecting a number of outputs

– agents that have been detected as having completed a computation – and by having an unique description of the possible inputs, one may be able to infer a function that consistently links output to input.[7]

But a further step can be given. Suppose now that the state transition table is modified by adding some extra triplets, that is, by changing some of the $(-, 0, -)$-triplets to others that represent valid steps of a computation. Additionally, assume that each step of computation is made dependent also on the actual interaction sites, much like the situation described earlier in the chapter. With this procedure, pathways that would have been pruned in the original situation, will now be preserved; hence, alternative pathways towards the final states become available. And what determines which pathway will be laid down by an agent is its lifetime history of interactions.

The bad consequence is that the original model of computation that this chapter described will have been lost. The good news is that a model of *coupled* computations will have been established. One that is based on a clear notion of space, and integrated with the processes of an artificial-life world. It is worth remarking, however, that it is not being implied that any arbitrary Enact world can demonstrate computations. In fact, only in restricted, well-defined worlds can we refer to computations in some useful way. This point will be made clearer in Chapter 6.

5.5.2 Coupled Computations

The important aspect of space is that it can be used to integrate the coupled computations, with a substantial gain in the autonomy of the process. The point is that the need for any sort of centralised process – that would be in charge of determining which components of the system would have to be used at a certain moment – becomes unnecessary. In fact, since it is the movement of the agents that determines the outcome of the computational coupling, and movement directly depends on the availability of space, it is clear that space is the ultimate determinant of the emergence process. So, the coupling is achieved without any preconceived target in mind; it is the dynamics of the system itself that determines what comes up, in a totally autonomous way.

Since the system is an artificial-life world, the process of emergence is couched in terms of the artificial-life processes it supports, like development, selection, and so on. The advantages of this approach are: an increased autonomy (due to space providing the support for coupling); robustness (due to using Turing machines); the possibility of incorporating evolutionary themes into the coupling scheme; the ability to explore functional self-organisation in limited regions of the space of computable functions (by carefully crafting the changes in the state transition tables); the support for linking functional self-organisation to the issue of phase-transitions (like the study of critical phenomena involved in the pathways that lead to one computation or another); among others. These themes are analysed in greater depth in the next chapter.

5.6 Conclusion

We have presented a model of computation that is couched in terms of artificial-life processes defined in a space. The computations are defined through the developmental process of an agent's lifetime history of interactions with its environment, and with the population of agents in its world. The issue of using well-defined notions of space as an

[7] The use or not of reproduction is only related to the degree of autonomy that we associate to a run; as we have seen in the previous section, the unique description of the inputs can be achieved in both cases.

intrinsic part of a process of evolutionary computation has been used before, although from other perspectives. For instance, [Sannier II and Goodman 1987] demonstrates a technique for doing artificial evolution using computations such that their outputs are expressed as patterns of movement of agents on a two-dimensional space; and in [Whitley 1993] an architecture was proposed to integrate the notion of space in cellular automata with standard genetic-algorithms, but which is still under development.

The implementation that provided the basis for the explanation of the model has cast the computational process in terms of its performance by the elements of a Turing machine. Such an approach served to characterise the main model of computation in Enact; yet, while the discussions were made in this particular context, all conclusions and conceptualisations should be regarded as general. We have shown how to code the input data in the agents; how to transform the state transition table of the original computation into an appropriate format that makes it amenable to the model's features; and how to identify that the computation has finished. For this implementation, it was described how to go about allowing reproduction to autonomously create a population of newborn agents so that, whenever an immortal agent has emerged, it has necessarily been the result of a developmental process over a newborn that represented a correct input (and never an intermediate triplet of the computational pathway).

Because the model of computation relies upon a population, it is essentially parallel; but we have not explored this alley in the chapter. Other features worth remarking include its emphasis on the notion of computation as a dynamical process (linking phenotype and memetype); and its stress on computation as taking place precisely at the interaction between agents and environment. Most significant of all, the model has the appealing feature of being cast in terms of a biological metaphor that integrates such concepts as population, genotype, phenotype, memetype, development, selection and environmental interactions.

This integration is possible because all the processes share the same notion of space, the cellular space, which is explored via the coupled movement of the population. Based upon such a feature we showed that the model of computation discussed can serve the basis for a model of *coupled* computations, an issue that will be addressed in detail in the next chapter.

5.7 Summary

It was shown how the entire dynamics of a class of evolutionary systems can be used to perform a computation. The argument was constructive by presenting a Turing-machine-based set-up implemented in Enact. As a byproduct, the chapter also served to characterise the main model of computation underlying Enact.

This model is essentially parallel, and relies upon the machinery defined by the artificial-life processes. According to the model, a particular computation is considered to have been performed, if and only if, for some initial population and environmental configuration, at least one agent has developed into a state configuration that is insensitive to any further environmental interactions; in this situation, if the computation involved is a function, this individual has the result. If the population ever vanishes, or if the environment becomes short of the resources needed for development, the cellular space has to be re-initialised, and the process iterated.

The presentation relied on the implementation of a function that recognises a particular context-free language. Implications of the model of computation were then discussed, in particular the model of coupled computations suggested by it.

CHAPTER 6

COUPLING COMPUTATIONS THROUGH SPACE[1]

6.0.1 Introduction

In the previous chapter the main model of computation in Enact has been identified. It was then suggested that models of coupled computations could be developed out of it. This chapter describes what these models of coupled computation could be.

Basically they are defined in terms of a set of Turing machines that share with each other one of their components (the tape symbols; their internal states; or their state transition table). The chapter discusses in more detail the model in which the state transition table is the shared component among the various machines. This model, if conveniently constrained, provides a way to address the issue of coupled computations in the context of Enact's coevolutionary activity; also, it opens the possibility of addressing the issue of criticality phenomena in constrained spaces of computable functions.

6.1 Coupled Computations

Complex dynamical behaviours can arise out of the coupling of even a small number of systems of the same kind, in particular when they are iterating. A variety of such systems have been reported in the literature, such as coupled maps of various kinds, cellular automata, etc. An interesting class of these systems is the one obtained as a result of coupled computations.

Parallel computations normally refer to the cooperation of computational processes towards the accomplishment of a well-known, predefined task. In coupled computations however, the interest is on the emergent computation that comes out of it.

As we discussed in Subsection 2.3.2, depending on the model of computation being used different coupling schemes can be obtained. In particular, depending on the way the components of a model of computation are partitioned, different modes of coupling can be established. For instance, by sharing memory between various von-Neumann machines, a coupling scheme is defined that is distinct from another built up by sharing, say, one of their internal registers. In the case of Turing machines, the natural partition would be the following set of components: internal states, tape symbols, and state transition table.

In this chapter a conceptual framework for models of coupled computations is developed based upon the assumption that the computations are performed by a population of processing agents whose structure is derived from Turing machines (TM). As a fundamental premise, the agents are embedded in a well-defined space which ultimately provides constraints on the individual movements, thus enabling their autonomous behaviour. The framework takes the form of a taxonomy, according to which Turing-machine components

[1] A version of this chapter was published as [de Oliveira 1994b].

are shared among the processing agents. Following the presentation of the taxonomy, the $\mathcal{STA}$ coupling model – the one that keeps the internal states and the tape as parts of the processing agents – is picked out and further developed, its features and advantages being stressed in relation to the other models. A key aspect of this model is that its notion of computation can only be made sense of at the interaction between the agents and their environment. Weak and strong versions of the $\mathcal{STA}$ model are then identified. Subsequently, a particular implementation of the latter is briefly discussed so as to clarify issues that appear in the context of the model; the actual implementation that is mentioned should be regarded as future work to be done. The resulting model – that also has embedded in it the features of development and coevolution inherent to Enact – is then discussed, in particular by contrasting its features with those possessed by some well-known systems. The aim of the chapter is to discuss the issues that its theme gives rise to; therefore, no actual computer run derived from a particular set-up of the resulting model is shown.

6.2 The Role of Space

The implications of different coupling schemes can be seen from various perspectives, such as, the degree of perturbation one machine has onto another (for instance, sharing a register versus sharing the entire memory space); or the adequacy of the scheme, when it is considered as a model of some phenomenon (the model of computation used in the Turing gas fits nicely into the analogy of the Lisp-functions being the particles or objects of the world, and the capability of evaluating the functions being the underlying physics). Another perspective is the role of the space in which the coupling takes place.

Explicitly or not, some notion of space is embedded in any coupled computation scheme. For instance, the notion of space used in systems that rely on coupled executions of assembly-like languages, like Tierra ([Ray 1992]) and the Venus family ([Rasmussen *et al.* 1990] and [Rasmussen *et al.* 1992]), is defined by the memory space of the computer involved. The Terrestris system presented in [Davidge 1994] uses a population of register machines that inhabit points on a two-dimensional memory, where they also exchange data with each other. The space inhabited by the two-dimensional Turing machines described in [Rucker 1993] is a lattice on the surface of a torus. Even in the context of other applications rather than coupled computations, a number of systems have used well-defined notions of space, such as in artificial life ([Werner and Dyer 1992]), neural netwoks ([Roska and Vandewall 1993]), genetic algorithms ([Sannier II and Goodman 1987] and [Whitley 1993]) and robotics. In contrast, although the activity in the Turing gas ([Fontana 1992]) has been metaphorically described as taking place in a "volume", this is in fact an abstract, rather ill-defined space.

The crucial point about space is that it can be used as a very natural way to integrate the coupled computations, the upshot of it being a substantial gain in autonomy for the process. This is the alley to be explored herein. By autonomy I mean the lack of centralised control, an intrinsic parallellism, and the fact that the agents involved have an individual ability to act. It should be clear that this notion of autonomy I am subscribing to is a simple, "intuitive", and non-technical notion; [Bourgine and Varela 1992] is a good pointer to deeper, more technical aspects on the topic.

6.3 Coupling Turing Machines through Space

6.3.1 Assumptions and Definitions

In this section and the next, models for coupling Turing-machine-based computations will be discussed. All these models are based on a population of TMs such that:

- They are embedded in a space, thus making it possible to distinguish a TM from its *environment*, that is, the rest of the space, apart from them.
- They have the autonomy to move in the space.
- Most of the environment is free for the TMs to move through.
- There is a special part of the environment, denoted by *interaction site*, that can be reached by the TMs but not traversed by them.
- The only direct interaction between the TMs is one obstructing the way of another as they move.

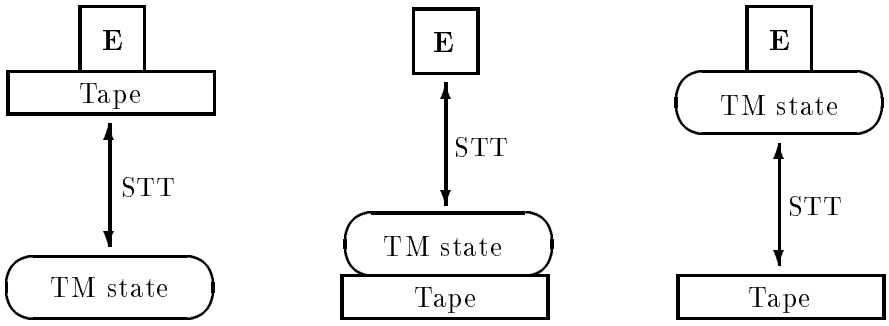
Let us consider the situations in which any of the main components of a TM (tape, state transition table and internal state) is taken out of each member of the population, and ascribed to the interaction site. This situation yields a coupling scheme between all TMs in the population, insofar as any computation that is performed at them depends on the content of the interaction site, which is shared among all the machines. Let us denote the shared component as the *coupling unit*.

As a consequence of "disabling" a TM as above, the entities that are formed by the remaining components can no longer be characterised as Turing machines; let us term these moving entities – that can take part in a computation defined in terms of Turing machines – *agents*.

6.3.2 Models of Coupling

Although the action of a state transition table only makes sense, by definition, at the interaction between tape and head states, in a coupling scheme the table could well be confined to the environment, to the agent, or to both at the same time (in which case it can be thought of as being a part of the entire space in which environment and agents are defined). In [Rucker 1993], for instance, the state transition table of the Turing machines are internally defined in the organisms as a code in their genotypes (while the two-dimensional tape is the entire environment inhabited by the organisms).

Depending on the choice of which one of the three components of a Turing machine is shared among the others, nine distinct coupling models can be obtained. Figure 6.1 features three of them; the others are irrelevant for present purposes but can be envisaged by considering the state transition table either in the environment or in the agent. In the models shown the state transition table plays the role of the "physics" of the world that creates the condition for a step of computation to be performed involving an agent. In other words, the state transitions can be thought of as belonging to both the agent and the environment. The unit of coupling resides in the interaction sites. One possibility is the interaction site representing either the tape or the state of the TM; another is to keep the tape and state as part of the same agent, while allowing the interaction site to act as the "physical support" for the computations to be performed. These three models of coupling are named, respectively, $\mathcal{T}o\mathcal{A}$ (tape-only in the agent), $\mathcal{S}o\mathcal{A}$ (state-only in the agent) and $\mathcal{ST}\mathcal{A}$ (state and tape in the agent).



$\mathcal{SoA}$ Model ($\mathcal{S}$tate-$o$nly in the $\mathcal{A}$gent)

$\mathcal{STA}$ Model ($\mathcal{S}$tate and $\mathcal{T}$ape in the $\mathcal{A}$gent)

$\mathcal{ToA}$ Model ($\mathcal{T}$ape-$o$nly in the $\mathcal{A}$gent)

Figure 6.1: The three models of coupled Turing machines, in which the state transition table (STT) is part of both the environment and the agent; that is, it is part of the space they are defined in. The agents are represented at the bottom and the interaction sites at the top.

The essence of all models of interest here is that the coupling will happen according to the coordinated movement of the population of agents in the space they are defined in. As they move about, they interfere with each other's trajectories, leading different agents to different interaction sites at different times, in a totally autonomous and decentralised fashion.

Because the $\mathcal{ToA}$ and $\mathcal{SoA}$ models fully share so fundamental parts of the structure of a TM – its internal state and the tape configuration – two major problems arise:

- Both models become too brittle in terms of their ability to support coupled computations in any practical way. The outcome of the couplings would too often lead to meaningless computations, much like the effect of arbitrarily putting together pieces of code from a standard programming language.
- The process of identification of the end of a computation becomes irremediably impaired. The end of a computation requires that not only a final state is reached, but also that the TM head points at a predefined symbol at the tape. Because the internal state and the tape are always disconnected in both $\mathcal{ToA}$ and $\mathcal{SoA}$, there is no straightforward way to identify when a computation has been completed. The only possibility is by fully inspecting the state of the world at each iteration; but this is a trivially uninteresting situation.

So far we have considered the interaction site as a singleton. This has been done because it is easier to convey the idea of coupling with a unique coupling unit. However, multiple interaction sites could alternatively be used. The major consequence is that the resulting coupling scheme would be even tighter, insofar as the interference possibilities between agents and sites would certainly increase. Naturally, the problems mentioned above for the $\mathcal{ToA}$ and $\mathcal{SoA}$ models would become even more critical.

6.4 The $\mathcal{STA}$ Model

Because of the problems aforementioned a case will be made in this section for the advantages of using the $\mathcal{STA}$ model. First of all, let us assume that multiple interaction sites

are in use.

The role of the interaction sites is really twofold: they are an essential part of the computation (for instance, by being the repository of the tape in the $\mathcal{ToA}$ model), and they provide a spatial reference for when the coupling should effectively take place.

While in $\mathcal{ToA}$ and $\mathcal{SoA}$ the interaction sites already possess one of the defining components of the TMs, in $\mathcal{STA}$ they do not, since tape and TM state are confined to the agents. From this observation it is clear that the coupling in $\mathcal{STA}$ is not as as tight as it is in the other two. Brittleness therefore has decreased, thus leaving room for multiple interaction sites to be used. It is worth noting that this scheme is indeed more appealing than a single interaction site, insofar as it explores the parallel nature of the various coupled computations.

Another consequence is that the problem mentioned above of the identification of the end of a computation is now solved. Whenever the individual has reached a final state and the head of the corresponding TM is pointing at the appropriate last symbol, the computation has finished and the corresponding tape represents its outcome.

Finally, the $\mathcal{STA}$ model *necessarily* leads to a modified Turing machine where, in addition to the standard TM components, the new dependence on the *state of the interaction sites* would have to be made explicit.[2] In fact, two (equivalent) possibilities for creating the new state transition table would be: the addition of the site state as a new entry variable, thus yielding a three-dimensional table; or the modification of the original table, by replacing the TM state entry by a new entry formed by pairs of the two kinds of states. The model thus has a "hint" of the $\mathcal{ToA}$ model, in the sense that the coupling scheme works as though the interaction site would be the repository of a new state which the steps of computation become dependent upon.

Depending on the extent of these table modifications, which are related to the degree of coupling that is allowed for the interaction site, two distinct possibilities for the $\mathcal{STA}$ model can be distinguished: the weak and the strong versions.

6.4.1 The Weak $\mathcal{STA}$ Model: only the form of the table is modified

In the weak $\mathcal{STA}$ model the role of the interaction site states is to enable or not a step of computation. That is, some sites would allow the step of computation to normally occur, as defined in the state transition table, and some would prevent it, leaving the state-tape configuration unchanged at that point. At the same time, the state of the interaction site might be modified, according to the agents' configuration. But no new entries in the table would be created, keeping its original content the same. They would only have to be modified to reflect the new table format.

Hence, coupling in the weak $\mathcal{STA}$ refers only to the fact of whether it is possible for a step of computation to be performed as the result of an interaction. What is at stake is the speed at which the (entire) computation will be performed, or whether it will be completed at all (not getting stuck at some entry of the transition table). For different runs, any computation that starts with the same initial state-tape configuration will lead to a final configuration that will be the same in all runs. In other words, all computational pathways (sequences of steps of computation as defined by the entries in the state transition table) are unique, regardless of their being related to a correct computation or not. Naturally, this behaviour is nothing more than the one normally expected from functioning programs written in standard programming languages.

By contrasting this weak version of the $\mathcal{STA}$ model with $\mathcal{SoA}$ and $\mathcal{ToA}$ models, it is

[2] Naturally, it is possible to think of table modifications also in $\mathcal{SoA}$ and $\mathcal{ToA}$; but while this would be a deliberate action in them, it is a necessity in $\mathcal{STA}$.

clear that while the coupling provided by the latter two is too tight (too much coupling), in the weak $\mathcal{STA}$ scheme it is too loose (too little coupling). So, while the interaction sites in $\mathcal{ToA}$ and $\mathcal{SoA}$ are *primary agents* of the coupling, insofar as they incorporate fundamental components of the computations, the role of the interaction site in the weak $\mathcal{STA}$ is simply one of *enabler* of a computation step. The desired degree of coupling should be somewhere between the two extremes; one that would allow the computation steps to become dependent, in a stronger way, on the state of the interaction sites.

6.4.2 The Strong $\mathcal{STA}$ Model: the table content is modified

This stronger dependence means that the result of an interaction should be expressed not only in terms of the possibility or otherwise of the corresponding step of computation performed, but also in terms of which one it will actually be. That is, new entries should be created in the state transition table with new actions corresponding to the new possible interactions.

With such a kind of modification of the transition table, a coupling scheme is achieved that, depending on the sequence of interaction sites the agent comes across, the agent may be led into a distinct sequence of computational steps, that is, into distinct computational pathways. The major consequence is that, for different runs, the same initial state-tape configuration may lead to distinct final configurations. And finally, no computational pathway is uniquely determined by its corresponding initial configuration.

Assuming the table has been modified, there is no unique way to traverse it. Naturally, the model presupposes that it will take place through the coordinated movement of the population of agents.

So, even though an experiment can be run with a single state transition table various distinct functions can be identified. How the transition table should be modified is a matter of implementation, the possibilities being the composition of state transition tables from distinct, well-formed functions; the addition of arbitrary state transitions to the state transition table of a well-formed function; or the mixing-up of state transitions from whatever origin. The new table formed as above then has the potential to yield not only the original functions that might have been used, but also others that are the result of "interferences" between the individual contents of each one of the primitive tables, or the individually added state transitions.

Summing up, what we have gained with the strong $\mathcal{STA}$ model is a tighter model of coupling than the weak version, one that opens up the possibility of distinct functions to emerge. But at the same time, the coupling is loose enough to provide us with a way to identify the end of a computation.

6.5 The $\mathcal{STA}$ Model Embedded in Enact

Enact is a system that embeds the models of computation discussed here, in particular the $\mathcal{STA}$ model, in either of its versions. In fact, the Turing machine implementation described in Chapter 4 essentially uses the idea of the $\mathcal{SoA}$ model as it keeps the state of the head in agent and the tape in the environment; also, in Chapter 5 we showed the implementation of a population of Turing machines that would be able to act along the lines of the strong version of the $\mathcal{STA}$ model.

The main feature of embedding the $\mathcal{STA}$ model in Enact is that it casts the issue of coupled computations in terms of an artificial-life world. In this context, it is the lifetime history of coupled movement of the population that determines what an individual agent

will develop into. And it is an agent's development that is interpreted as a function, the final state of the agent being the outcome of the function.

In this section we discuss what is gained by implementing the strong version of the $\mathcal{STA}$ model within Enact. No actual computer run derived from a particular set-up of the resulting model will be shown; only the conceptual issues that come out of it will be of interest for present purposes. It should be said that, in fact, we will rely on a generalised version of the $\mathcal{STA}$ model, which will be characterised below.

6.5.1 Towards Probing a Region of the Space of Computable Functions

It should also be remarked that, as I made clear at the end of Subsection 5.5, that only conveniently constrained world set-ups, provide a useful way to address the issue of coupled computations in the context of Enact's coevolutionary activity. This subsection hints at one such Enact world that is currently under analysis, but whose details are beyond the scope of this thesis. In addition to clarifying points that come out of embedding the $\mathcal{STA}$ model within Enact, this section should also be regarded as a pointer to a world set-up that is yet to be effectively probed in the follow-up to the work reported in this thesis.

6.5.1.1 Beyond the $\mathcal{STA}$ Model

Before explaining the world set-up, let us first generalise the $\mathcal{STA}$ model by going beyond its definition in terms of Turing-machine-based computations, which is done by using another model of computation. The point here is: let us assume that the initial state configuration of an agent's memetype is input data for a computation; let us also assume that we know how to detect that a computation has finished. When an output is observed that is indicative of the end of a computation, the computed function – that we interpret the agent having been subjected to – can then be inferred. This process can be regarded as a generalisation of the $\mathcal{STA}$ model insofar as the following features are preserved:

- the input and output of the computations are confined to the agents;
- the interaction sites only define a locus for the steps of a computation to take place;
- the physics of the world contains the process that is in the root of the computational process (the "algorithm", so to speak); and
- the robustness of the scheme, with several functions being potentially able to emerge.

6.5.1.2 Rationale of the World Set-Up

The set-up relies on six state transitions and two interaction sites, as shown in Table 6.1. Note that at each kind of interaction site only three of the state transitions can be triggered, and also that, basically, the state transitions simply entail local operations over pairs of states of the agent's memetype. While some of these operations are due to individual state transitions, others require that a pair of state transitions be triggered together.

Each time an agent traverses an interaction site it is subjected to a succession of those local operations. As the agent successively traverses the various interaction sites of the cellular array these compositions "accumulate" and eventually may converge and become stable. Along this line it can be seen, for instance, that for any combination of state transitions of only the $\mathbf{E}^+$-type, only monotonically increasing sequences converge and become stable.

Instantiated State Transitions												
[1]	$K_i \mid_{K_i > K_j}$	$\mathbf{E}^+$	#		[2]	$K_i \mid_{K_i < K_j}$	$\mathbf{E}^-$	#	
	$\mathbf{E}$	K_j	#	$\Rightarrow K_i$		$\mathbf{E}$	K_j	#	$\Rightarrow K_i$
	#	#	#			#	#	#	
[3]	K/G	$\mathbf{E}$	0		[4]	K/G	$\mathbf{E}$	0	
	$\mathbf{E}$	K_i	$\mathbf{E}^+$	$\Rightarrow K_j$		$\mathbf{E}$	K_i	$\mathbf{E}^-$	$\Rightarrow K_j$
	0	$\mathbf{E}$	$K_j \mid_{K_j < K_i}$			0	$\mathbf{E}$	$K_j \mid_{K_j > K_i}$	
[5]	M	$\mathbf{E}$	0		[6]	M	$\mathbf{E}$	0	
	$\mathbf{E}$	K_i	$\mathbf{E}^+$	$\Rightarrow K_j$		$\mathbf{E}$	K_i	$\mathbf{E}^-$	$\Rightarrow K_j$
	0	$\mathbf{E}$	$K_j \mid_{K_j < K_i}$			0	$\mathbf{E}$	$K_j \mid_{K_j > K_i}$	

Table 6.1: The complete set of instantiated state transitions for the world set-up mentioned in the text. There are two types of interaction sites, characterised by a cross-like shape, its rightmost cell being the one that differentiates the two types, according to its state being $\mathbf{E}^+$ or $\mathbf{E}^-$. As the interaction sites are traversed by the agents, the latter become subjected to the instantiated state transitions.

By inspection of the cellular space, it becomes possible to determine when a systematic pattern of activity has been established, in which case a function can be identified.

The state transitions that use the $\mathbf{E}^+$-state provide a local bias in the dynamics of the system towards monotonically increasing sequences appearing in the sequence of state-values of an agent's memetype cells; analogously, the ones with the $\mathbf{E}^-$-state locally bias the development of the agent towards monotonically decreasing ones. Naturally, at a certain point in an agent's development, sorted sequences should come about in its memetype, from which it would then become possible to refer to a *sorting* function that was "applied" to the original memetype. This is indeed the case and, in fact, various other functions can also be identified, which are "close" – in the function space – to the sorting function.

It should be observed, however, that all that there is in the system is a dynamical process whose effect – the development of the agents – can, with hindsight, be equated to functions being performed.

In general, at each iteration of the automaton two possibilities may happen in respect to each interaction site: either there is an M-state at the position in which it appears in state transition [5], or that position of the neighbourhood has not an M-state. Whichever the case, naturally only one pairwise operation is performed.

But the presence of the (critical) M-state at each neighbourhood configuration depends on the mutual coupling in trajectory and speed that exists between the agents in their lifetime. In other words, it is the result of the coupled history of interactions of the agent

with the rest of its world. Hence, although the operations over pairs of memetype cells are local and predefined, their overall consequence in the full memetype depends, in a long span of time, on the whole history of events of the world.

Now, different subsets of the state transitions define distinct – possibly overlapping – subspaces of the space of possible emergent functions. But because of the coupled history of the agents' developments, which functions effectivelly emerge out of a run depend on the actual initial condition of the set-up.

6.5.1.3 Possible Consequence

I believe that the set-up hinted at above constitutes an appealing case in which the issue of coupled computations – in the context of Enact's coevolutionary activity – may yield fruitful consequences. In general, it is expected that it may prove to be useful in addressing some issues that link dynamics and computations in the context of cellular automata (along the lines mentioned in Chapter 2), in particular in the context of criticality phenomena.[3] More precisely, the following questions could be addressed in this context:

1. For which initial and boundary conditions could only monotonic functions be identified?

2. For which conditions would it be possible to prevent the existence of critical interactions that would rule out a particular kind of development of the agents (which would prevent a particular function from emerging)?

3. By introducing a time-dependence on the movement of the agents, how would that affect the overall dynamics of the system, for instance in terms of the two questions above?

The main aspect of this world set-up is that it is characterised by a simple definition that renders it tractable to formal analysis, and at the same time, is sufficiently rich in terms of the space of computable functions that it entails. But despite all these features, to address those questions is beyond the scope of this thesis. My intent here was only their identification in the context of the set-up we mentioned, taking some first steps towards work to be done following the thesis.

6.5.2 Enact's Approach to Coupled Computations in Perspective

The issue of different levels of descriptions of cellular automata have been recognised in the literature. For instance, although recognising cellular automata as providing

> "...a powerful approach to the study of the emergence of loops between objects and functions ...",

Fontana ([Fontana 1992, page 198]) then remarks that it

[3] Recently, there has been great interest in critical phenomena, as they bear relevance to the understanding of various natural phenomena. See [Schroeder 1991] for examples of critical phenomena in various physical and mathematical systems; [Bak *et al.* 1988] or [Bak *et al.* 1991] for a particular model, applicable in avalanche-type phenomena; [Kauffman and Johnson 1992] for the use of criticality issues in a coevolutionary model in theoretical biology; [Martin 1990] for an abstract account in the context of a class of one-dimensional systems; [Adami 1993] for an artificial-life-based approach as a model to living systems; and [Langton 1990] or [Langton 1992b] for their being invoked in the context of the emergence of life and computation in natural systems.

> "...becomes, however, difficult to study the consequences of such a loop at the same level of description that has been used to study its emergence."

From another perspective, in the context of a discussion on cellular automata as "self-programmable" systems, it is in [Rasmussen *et al.* 1992, page 219] that the

> "...main difficulty with the CA approach seems to be associated with ... the extreme low-level representation of interactions."

It is worth remarking that Enact has two levels of description. Accordingly, the use of a population of autonomous agents – the processing units involved in the computations – are realised at a higher level than the one Enact itself is implemented at. In other words, while Enact is defined from basic state transitions, the population of processing agents is mainly defined through the high-level concepts of the system (such as agents, phenotypes, memetypes and so on.)

Several issues can be explored in a comparison between the $\mathcal{STA}$ model of coupled computations in Enact and other approaches. What follows is an attempt to compare some of the aspects, mainly with respect to the Turing gas and Tierra. The model just discussed has the following features:

- *Evolutionary capabilities.* Just like Tierra, the model can be used within an evolutionary context, even though the actual evolutionary possibilities is not the same for each of them. The Turing gas however lacks this feature, which is even explicitly recognised in [Fontana 1992].

- *Focussed emergent computations.* The model can be used to tackle the problem of emergent computations (or, in particular, of emergent functions) even in small regions of the function space. The point is that the function space which is implicitly defined by the state transition table – that characterises the interactions between the agents and the environment – can be controlled in an independent fashion. This is possibly the most fundamental aspect of the system. The tractability that is gained implies that it becomes possible to approach the issue of functional emergence by looking at the *actual* functions that emerge. In fact, this feature can be inferred from Subsection 6.5.1 for that particular set-up that enables the emergence of functions over a sequence of integers.

- *Ability to link functional emergence to the (apparently disconnected) concept of "phase" transition.* By enabling the process of functional emergence to be focussed in a region of the function space, it becomes possible to create a link between the issues of functional self-organisation and phase transitions in some dynamical spaces (see [Langton 1990]).[4] For instance, it becomes possible to refer to criticality phenomena by means of the situations in an agent's lifetime which are determinant of its long term development. That is, the critical points that determine which computable function the development of an agent will end up being characterised by. Aspects of the reversibility of computations also come up in this context. As suggested in Subsection 6.5.1 the world set-up mentioned therein also seems appropriate to address these aspects.

[4] The primary concern on phase transitions presented in [Langton 1990] was its characterisation in the rule space of cellular automata. The fact that Enact is implemented as cellular automata is even more appealing with this respect.

- *Copying or reproductive function is not essential.* A copy or reproduction function does not play any major role in Enact as they do in various of the experiments discussed for the Turing gas, or in virtually all reported experiments performed with Tierra. Indeed, all interesting reported outcomes from the latter system depend on the existence of the so-called "Ancestor", a self-reproductive agent that is innoculated in the Tierran soup at the start of a run. A step towards an exception was reported in [Tackett 1992]; with the addition of a new register to Tierra, selective pressure was allowed according to the processing of the content of the register. By acting as a connection of Tierra with the outside world, the added feature provides an additional way to drive Tierra's evolution beyond mere reproduction.

- *Functions with any number of parameters.* For the purposes of making AlChemy's implementation easier, it can only handle functions of a single parameter. Naturally, this is a strong constraint that restricts the sorts of emergence that can be observed; Fontana (1992) himself recognises the problem. But again, it should be clear that the problem is not a consequence of the model, but only of its implementation. As this chapter made it clear, the approach that was discussed does not require a limitation on the number of parameters of the functions involved.

- *Robustness.* The major problem when computer programs become the subject of evolutionary and of self-organisation processes is how to achieve robustness, i.e., how to escape from the brittleness of their semantics when arbitrarily putting chunks of code together. One way or the other this problem has been solved in AlChemy, Tierra and various other systems. Turing machines have a very robust semantics because they simply handle states, which are indifferentiable from each other. It is such a robustness that enables the approach supported by the $\mathcal{STA}$ model. The set-up mentioned in Subsection 6.5.1 does not use Turing machines, but since the functions it deals with only requires pure integer values, the argument of robustness still holds in that case.

- *Autonomy.* It is the movement of the agents that determine the outcome of the functional emergence; but movement is part of the nature of the agents, constrained by the availability of free space. Hence, there is no need for any sort of centralised process that would be in charge of determining which components of the system would have to be used at a certain moment. The latter is exactly what happens in the Turing gas in regard to the necessity of arbitrating the pairs of Lisp-particles that will collide at a given instant. In Enact, this "decision" is not only decentralised, but also is just a natural consequence of the dynamics of the system.

6.6 Final Remark

The computer metaphor has been widely used to describe natural phenomena; its success, however, is questionable. For instance, as [Varela *et al.* 1991] reminds us, two very common misconceptions that come out of it can be perceived: in cognitive science, when considering environment as data that is given to a program in the cognizer; and in biology, when assuming the genome of an organism as a program that is run by the biomolecular machinery. I believe that the main cause of these flaws is not in the approach itself. Instead, it comes from the model of computation that is used to ground the metaphor. It is expected that the issues raised in the context of the $\mathcal{STA}$ model of computation may shed light on the track that leads to descriptions of natural phenomena in harmony with the use of the computer metaphor.

Accordingly, the fact that the state transition table of the models discussed herein were allowed to be shared by agents *and* interaction sites is meant to provide a notion of computation that has to be regarded as taking place only at the interaction between agents and environment.

In the strong $\mathcal{STA}$ model the role of the environment has been lessened, if compared to the $\mathcal{ToA}$ and $\mathcal{SoA}$ models, and has been strengthened in relation to the weak $\mathcal{STA}$. It still has an active role in the computation, but became a *mediator* rather than an enabler that it is in the latter model, or a primary agent of the computation that it is in the former two. The environment's role has become the provision of *active physical support* for the computation to occur.

In addition, the fact that the actual functions that are computed by the agents critically depend on their lifetime history of interactions, strengthens the role of the interactions, and consequently, the role of the agents, since the (local) movement of the latter crucially depend on themselves. But since the agent itself provides the place where the outcome of the computation is visualized, it is tempting to say that in the strong $\mathcal{STA}$ model the agent has become the *subject* and the *object* of the computation, a notion that is based on an idea originally developed in [Lewontin 1983] in the context of biological evolution.

Other aspects of the $\mathcal{STA}$ model of coupled computation which are worth bearing in mind include its intrinsic parallellism, and the stress on the notion of computation as a dynamical process. Even more fundamentally, the definition itself of the coupling scheme, as well as its reliance on the notion of autonomy of the processing agents could only be achieved by explicitly resorting to a well-defined notion of *space* that permeates all activity.

6.7 Summary

A conceptual framework for models of coupled computations was developed based upon the assumption that the computations are performed by a population of processing agents whose structure is derived from Turing machines.

As a fundamental premise, the agents are embedded in a well-defined space which ultimately provide constraints on the individual movements, thus enabling their autonomous behaviour. The framework takes the form of a taxonomy, according to which Turing-machine components – the tape, the state transition table, or the set of internal states – are allowed to be shared among the processing agents.

Following the presentation of the taxonomy, the $\mathcal{STA}$ coupling model – the one that keeps the internal states and the tape as parts of the processing agents – was picked out and further developed, its features and advantages having been stressed in relation to the other models. A key aspect of this model is that its notion of computation can only be made sense of precisely at the interaction between the agents and their environment; in other words, this means that the computing procedure (the state transition table, in the case of Turing machines) becomes part of the "physics" of the world. Weak and strong versions of the $\mathcal{STA}$ model were then identified, and an implementation of the latter was briefly discussed in the context of Enact, thus allowing to explore the clear-cut notion of space provided by cellular automata.

The resulting model – that also embeds into it the features of development and coevolution inherent to Enact – was then discussed, in particular by contrasting its features with those possessed by some well-known systems. It was finally argued that, in order for the resulting model to be used in a useful way, particular world set-ups would have to be used; one such world was then hinted at so as to help to make it clear the general advantages of the resulting model of coupled computations supported within Enact.

Chapter 7

CONCLUSION

This chapter is an evaluation of what has been done throughout the thesis, as well as prospective in terms of what the achievements are pointing at. Having dealt with these topics, I will then conclude with a personal statement on the historical pathways that led me to the research reported in this thesis.

7.1 The Thesis in Retrospect

We have explored in this thesis a cellular-automata-based system which supports an artificial-life activity that can be viewed as a computing machine.

This exploration was made possible because the development of an architecture for this machine has been achieved, as well as a way to program it. A key issue that enabled the creation of the machine was the way we explored the clear notion of space that is intrinsic to cellular automata, coupled to the design imposition that the movement of the agents should be constrained by the availability of free space. A related key issue in this respect was the introduction of the concept of interaction site as a way to impose the *loci* in space, where specific interactions would take place. These and other issues were presented in Chapter 3. As for the programmability of this machine, the key issue was the concept of instantiated state transitions that has been introduced in the thesis. We discussed this topic in Chapter 4.

The way we progressed towards the exploration of that machine went through the following steps. Firstly, in Chapter 4 we showed that the machine is capable of universal computation. This step can be regarded as a mere existence proof of such ability, insofar as it did not rely on all the processes that underlie the artificial-life activity.

The second step, given in Chapter 5, was to show that the universal computability could also be defined in a more fundamental way that relies on all the artificial-life processes together.

In the spirit of the latter, we then showed, as the third step, that together, all those processes could also be regarded as a set of coupled computations. It turns out that the issue of coupled computations within Enact has various interesting architectural features, in particular when the coupling process is defined within a well-defined space of computable functions. All these features we discussed in Chapter 6, which we can summarise as follows: robustness; independence of the coupling process from the necessity of introducing copying functions into the system; autonomy; and ability of the coupling process to handle functions with any number of parameters. Further to these architectural features is what they jointly entail, which is the possibility of linking the evolutionary activity within Enact with the issue of coupled computations; also, they open the possibility of addressing issues such as criticality phenomena in the space of computable functions.

Naturally, its description in terms of coupled computations would not be beneficial from any arbitrarily defined world set-up as far as the latter two possibilities are concerned. With this consideration in mind, we then complemented the third step by sketching out one particular set-up within which those possibilities might be realised. This was the set-up suggested – in a very preliminary way, I should say – in Subsection 6.5.1; it is characterised by a simple definition, although being sufficiently rich in terms of the space of computable functions that is entailed by it. It should be clear, however, that all the issues related to this set-up were not developed in the thesis, and therefore remain as future work to be done.

7.1.1 Open Issues

Due to their emergence-based nature, the process of interpreting the outcomes of artificial life worlds usually have to face a problem of finding a reference (or domain) upon which the interpretations can be grounded. One of the facets of the problem is that a predetermined domain – say, standard biological concepts – might not be able to account for the wealth of possible dynamical behaviours that can come out of a run. Naturally, the usage of one form of interpretation or another depend on the aim of the particular artificial life world at issue. However, there would be advantages in the use of less informal approaches; one of them could be the possibility of using knowledge about the dynamics of one particular system onto another. One approach towards realising the former that can hold promise is the characterisation of the outcome of these systems in terms of computable functions, thus casting whatever has been learned about the dynamics of a particular system into the solid grounds of computational theory.

The former problem was not at all a concern in this thesis; neither has the world set-up hinted at in Subsection 6.5.1 been mentioned in the context of the previous purpose. However, since that set-up is couched in a coevolutionary system, and an aspect of its dynamics is well-defined in a space of computable functions, an extra open question associated with it is left as to whether it could be an instance of a world set-up that could fulfil the concern we have just expressed.

Another issue that has not been addressed in the thesis but which is worth recognising is the exploration of Enact as a parallel machine, that is, the use of the coupled computation scheme in order to perform computations in a collaborative way. Having experienced some rather complex implementations within Enact, my perception is that an account of the system from this perspective is possible, although I have not taken any step in this direction.

I have stated at the outset that the research described herein is inspired by, but not a model of biological reality. In fact, that was the perspective we adopted throughout this work. However, at some points of the thesis the reader might be tempted to consider the possibility that the system, or another derived from it, might be useful for the acquisition of some level of insight into the natural system it is inspired by. It should be clear, however, that this is a completely open issue in the current context.

There is a caveat to be made in regard to the second step of the main thread of the thesis, as mentioned above. Although the argument presented effectively leads to the identification of a model of computation that relies on the entire artificial life activity, it should be observed that this process of computation was not fully autonomous. That is, the actual implementation could not avoid the necessity of having an agent, external to Enact, that would be in charge of re-initialising the system with a new population when the individuals of the existing population have all died. This is due to the issue discussed in Subsection 1.3.1 involving the reproduction process. Namely, since reproduction oc-

curs in a way that is limited by the locality constraint imposed by the neighbourhood, it is not possible for reproduction to generate an arbitrarily specified state configuration in the agent (in its memetype, to be precise). As a consequence, the automatic process of continuous recreation of new inputs for the computation becomes impaired, thus necessitating the external introduction of new individuals that represent the input of the computation.

7.2 The Balloonist Becomes a Driver: A Generalisation of Enact

A generalisation of Enact is currently under way, its rationale being the following: instead of agents with a spatially distributed internal structure, they have become particle-like agents in the new system, in a similar fashion to the one described in [Packard 1989]. The upshot of this new design is that any neighbourhood involving an agent contains all the information about its structure.

As for the internal structure of the agents, they still feature genotype, phenotype and memetype, but head and tail are no longer necessary, since they were simply artifacts for rendering Enact's implementation simpler.

It is worth spelling out some other features of the new architecture:

- Genotype, phenotype and memetype now have arbitrary length.
- The movement of the agents became isotropic, that is, the agents now have the ability to move in any of the eight possible directions. As a consequence, agents can touch each other arbitrarily, but subject to the condition that only one agent can occupy one cell of the cellular array.
- The genotype specifies the initial configuration of the phenotype. The genotype is created during sexual reproduction, its configuration being given by the parental genotypes, according to an arbitrary procedure, such as a standard genetic algorithm.
- The phenotype specifies the way an agent should move. The initial phenotype of an agent depends only on its genotype, but may change arbitrarily during the agent's lifetime.
- The initial configuration of the memetype is fully inherited from the parents with no locality constraint, and may change arbitrarily during the agent's lifetime.

Generally speaking, the major motivation behind the new system is that it provides a much "cleaner" implementation of the artificial life world, insofar as all processes can occur in a fully symmetrical and isotropic way. Among the immediate benefits we could cite:

- A much smaller and simpler set of state transitions can be obtained, which at least in principle may render much faster runs.
- A solution is given to the problem mentioned above regarding the role of reproduction in the model of computation.
- A much simpler way of going about setting up worlds becomes possible.
- The crowding effect mentioned in Chapter 3 can be eliminated.

Any generalisation of Enact should preserve the two following design principles, that are implicit to Enact's architecture:

- *Full Coupling.* All the processes in the artificial life world should be fully coupled.
- *Internally-driven Dynamics.* The coupling process should be a consequence only of its internal activity.

In the case of coupled functions, these principles entail that the outcome of the events that lead to the emergence of a function becomes dependent on the full history of past events in the world. To clarify this point it is worth remembering what this situation was for the world set-up described in Subsection 6.5.1. In that context the speed of an agent, as it traverses an interaction site, determines which local operation is applied over a pair of memetype states; but its speed depends on the full history of past events in the world. Naturally, it is through space-constrained movement that the two principles above can be met.

The implementation of the system is intended to be with the SWARM package ([Langton 1993]), whose first release is expected in the current year (1994). This is a general package for simulations of architectures of autonomous agents, with a number of built-in tools, in an object-oriented environment. While this package is not available, a provisional implementation is being done with Cellang ([Eckart 1994]), a cellular automata simulator that has a very simple facility for dealing with agents based on [Stephenson 1992].

The first intended implementation is the world set-up hinted at in Subsection 6.5.1. It will be used to address the issue of criticality phenomena in the space of computable functions implicit in the set-up. Initial questions to drive this enterprise will be the ones presented in Subsection 6.5.1.3. The practical difficulties that would have to be faced in order to perform such a study within Enact as it stands would be to so serious that the most sensible alternative is to building a new tool. Among these difficulties one seems unsurmountable: the package in which Enact has been developed is based on a windowing system that is no longer available in its host machine.

In conclusion, and using the allegory I introduced in the preface of the thesis, it seems clear that in the next leg of the journey that will follow the completion of this thesis, the balloonist that started this journey will resume it as a driver.

7.3 Personal Statement

This section will finish at the point this thesis started. Here I will allow myself to take a rather personal stance in order to trace the origins of this thesis, from its deepest motivations to the paths I followed while trying to bring it into reality. It is a personal testimony of my own history in this research programme.

Behind the steps I have taken, the quest for understanding the *emergence of meaning and function* in natural and artificial systems has been the primary motivation. Due to a mix of historical contingencies and personal choice, I ended up approaching that issue in the context of intelligence, which, most researchers would agree, is the phenomenon that enables the creation and manipulation of meaning in an apparently open way. My particular trajectory started with artificial intelligence.

Soon after my first degree in Electronics Engineering I joined the artificial intelligence group (of the Brazilian Institute for Space Research) I am still with, and for about six years I worked with machine learning and knowledge acquisition, from a cognitivist perspective.

By the end of my Master degree I found myself deeply dissatisfied with the cognitivist over-emphasis on representations (i.e., symbols with predefined meanings) that my work

had been related to, and in searching for alternatives, started getting acquainted with computational evolutionary biology. At this period I had my first contact with genetic algorithms and classifier systems in [Holland 1986], where a learning model was couched in evolutionary terms; the essence of classifier systems, really. Particularly important for me was a very interesting paper by Lenat [1983] where he speculated on the possible advantages that natural evolution might be taking from having learnt how to search the space of species.

My search continued until 1987, when I attended a talk by F.Varela, during a Brazilian scientific meeting. In this talk he presented a comparative analysis of approaches to cognition, which was published as [Varela 1989]. His own view of cognition, named *enaction* had a strong biological slant, and, although I could not understand exactly what he was hinting at, I felt allured by it. What attracted me was not so much the point he made about cognitive processes themselves, but the associated world view that the enactive perspective was apparently suggesting; one in which the world would not be a pregiven, independent, and predefined entity. One, as a consequence, from within which the possibility would be open for the emergence of meaning to be observed in a genuine way; that is, without the constraints and determinants that standard knowledge-based approaches to learning featured at that time.

Varela's talk put everything I had read about cognitive science – learning in particular – into context and I could, for the first time, see the whole and have a glimpse of a direction I was willing to go in. But I could only see very dim lights flickering in that direction. Enaction still seemed to me an overly philosophic standpoint that I did not quite understand. I needed more ground, a way to link those concepts to computational systems as a whole.

A few months later, during the first half of 1988, I then came across two special issues of *Physica D*, proceedings of two conferences held in the USA: [Farmer *et al.* 1984], from *Cellular Automata: an Interdisciplinary Workshop*; and [Farmer *et al.* 1986], from the conference *Evolution, Games and Learning: Models for Adaptation in Machines and Nature* mentioned earlier in the thesis. Particularly the latter, with its breadth of scope and the explicit reference to learning provided the light that would guide me in the subsequent couple of years. In those volumes I found a wealth of computational approaches that looked very solid. My concern from that point onwards became the clarification of the shapes and forms that were being suggested to me, as well as the straightening of the light beam that had been switched on.

With that frame of mind, I eventually arrived at Sussex at the end of 1988 as a research student for the DPhil in Cognitive Studies. The stated multidisciplinarity of the programme; permeability to new ideas; and explicit interest – on the part of the faculty member who accepted me as his research student – in the "conceptual relations between cognitive science, biology and artificial intelligence", were the key factors underlying my choice.

While still in Brazil I had read a report on the first Workshop on Artificial Life, also held at Los Alamos, in 1987. As I started to look for bibliography for my research proposal outline, with the Internet at my fingers (a facility I did not have formerly), it was a natural step to get hold of the proceedings of that event, [Langton 1989]. It became clear that this emerging new discipline would provide my systematic focus from that point onwards, giving deep roots to the issues presented in [Farmer *et al.* 1986]. And it really did.

At the beginning of 1989 I attended a symposium in France, where one of the invited speakers was Varela. His talk was basically a deepened version of the one I had heard earlier. But this time I had more conceptual tools to help me follow his arguments. In particular, [Varela 1989], his "petit bouquin" (his words) had just been published and

reading it was fundamental to clarify a bit further my understanding of his thought. Also very helpful in this regard was [Winograd and Flores 1986], who widened various obscure philosophical points somewhat further. It is worth remarking that, having come from a purely symbolicist tradition in artificial intelligence – and not even fully aware of it! – all those philosophical discussions were very much a novelty for me. The chapter on "Cognition as a biological phenomenon" was particularly relevant as it drew from previous work of Varela, mainly his joint work with Maturana on autopoiesis, already referred to in the thesis. The latter eventually led me to read [Maturana and Varela 1987], but my concern remained in enaction itself.

Around the middle of 1989, I submitted my research proposal outline, which had made its motivation clear in its title, *Probing the emergence of a new function: A computational account based on evolutionary genetics*. I presented the theme in terms of the metaphor of "crossing the barrier of meaning" developed in a little paper by Rota [1986], at the opening of the aforementioned [Farmer *et al.* 1986]. The conceptual orientation was also fairly coherent. The ideas concerning the likely computational model, however, turned out to be premature. In fact, I still thought of it in terms of production systems and genetic algorithms. Cellular automata were not even considered. And the stance of looking at evolutionary processes in an intertwined fashion with learning (as in [Harley 1981], [Draper 1987], [Hinton and Nowlan 1987], and [Smith 1987]) was still very much present. Also, the way I had approached the issue put too much emphasis on biological concepts, as if I was going to model some aspect of biological reality. Finally, the research proposal also had elements of a view of evolution from a developmental psychology point of view, as in [Bateson 1985] and [Scaife 1989].

As a whole, the research outline made clear the motivation underlying the thesis. Namely, looking at the emergence of functions with an enactive orientation, where the emphasis on the issue of self-organisation would be fundamental.

However, I soon came to realise that, on the one hand, I did not have the right tools, or at least, I did not know enough about the new ones I had come across, such as cellular automata. On the other, it also became clear that I was overly committed to a biological account just because of the original basis of genetic algorithms, and also that I was being unecessarily influenced by high-level notions derived from my former background; both biases, by the way, were echoed in the background of my then supervisor, a developmental psychologist who had been formerly a biologist. And finally, the notion of function I was explicitly subscribing to (the utilitarian sense), was blurred with the one I had implicitly in mind (the formal sense of computable functions).

Conceptually, those perceptions were pointing towards more abstraction; and in implementation terms, towards the use of a more fundamental framework. And cellular automata seemed to be at the convergence between the two. It was then a matter of evaluating their possibility. With this impulse, the journey started. And what happened afterwards has already been told...

Appendix A

The Complete list of State Transitions in Enact

A.1 Introduction

This appendix presents the complete list of non-quiescent state transitions for Enact. For all sections the following conventions hold:

- The B-states refer to the cells in the agent's body, i.e., either P-, G- or K-state.
- The E-state refers to any environmental state, including the background state. When it is necessary to refer uniquely to the latter, the *0*-state representation is used.
- The symbol # is a *don't care* referring to either of the six possible state categories: T, M, P, G, K, and E.
- When the symbol "/" appears in a neighbourhood separating two state categories (as in B/T), the corresponding cell can take on either of the state categories involved.
- When the states of two or more cells of the same neighbourhood refer to the same state category, in general no distinction is made between them. The only exception is when the transition also leads to a state of the same category. This situation of ambiguity is resolved by subscripting the state categories involved, with the geographic location of their corresponding cells, according to the notation of Figure 3.1.
- The same rationale applies for the neighbourhoods which have more than one cell in a state represented by #.
- The transitions characterized by the symbol $\overset{\overline{d}}{\Rightarrow}$ are non-deterministic; the ones with $\Rightarrow$ are deterministic.
- The index *def* used in some cases is an implementation detail defining the "default" state-value to be used.
- The special states T^0 and P^0 are the initial states of neonatal development. For the sake of conciseness we denote by T^* a T-state that is different from T^0.

An additional remark: the way we present the state transitions directly reflects the way they are currently implemented, i.e., as a set of successive *if's*, each one corresponding to the state transition shown. Evidently, this is not the most efficient way to implement them but we have kept it because its modularity provides clarity, as well as making debugging much easier. In a similar vein, we also have not been concerned with the

efficiency of the implementation in terms of avoiding redundancies and inconsistencies between different transitions. Noteworthy in this respect is the redundancy expressed by transitions A.3.18 and A.3.19, and the intrinsic inconsistencies due to transition A.4.14 in relation to A.2.11 or A.2.27. In practice such an inconsistency is solved by letting transition A.4.14 occur after the ones it is conflicting with, which means that precedence is given for the transitions occurring during movement. What these points suggest is that it is possible to express the set of transitions in a significantly more optimized way not only in terms of their not presenting internal conflicts or redundancies, but also in terms of a more concise representation which would significantly speed up its computation at each iteration.

A.2 State Transitions for Movement

[1] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \mathrm{E} & 0 & T^{*} \\ \hline \mathrm{E} & \mathrm{E} & \mathrm{E}/T \\ \hline \end{array} \overset{\overline{d}}{\Rightarrow} 0/_{M_{def}}$ [2] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \mathrm{E} & 0 & \mathrm{E} \\ \hline \mathrm{E} & \mathrm{E} & T^{*} \\ \hline \end{array} \overset{\overline{d}}{\Rightarrow} 0/_{M_{def}}$ [3] $\begin{array}{|c|c|c|}\hline \mathrm{E} & M & \mathrm{E} \\ \hline \mathrm{E} & M & T \\ \hline \neq \mathrm{E} & \mathrm{E} & \mathrm{E}/T \\ \hline \end{array} \Rightarrow 0$

[4] $\begin{array}{|c|c|c|}\hline \mathrm{E} & M & \mathrm{E} \\ \hline \mathrm{E} & M & T \\ \hline \mathrm{E} & \mathrm{E} & \mathrm{E}/T \\ \hline \end{array} \overset{\overline{d}}{\Rightarrow} 0/_{M_{def}}$ [5] $\begin{array}{|c|c|c|}\hline \neq \mathrm{E} & \mathrm{E} & \# \\ \hline \mathrm{E} & M & \mathrm{E} \\ \hline \mathrm{E} & M & T \\ \hline \end{array} \Rightarrow 0$ [6] $\begin{array}{|c|c|c|}\hline \# & \mathrm{E} & T/_{B} \\ \hline \mathrm{E} & M & \mathrm{E} \\ \hline \mathrm{E} & M & T \\ \hline \end{array} \Rightarrow 0$

[7] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \mathrm{E} & M & \mathrm{E} \\ \hline \mathrm{E} & M & T \\ \hline \end{array} \overset{\overline{d}}{\Rightarrow} 0/_{M_{def}}$ [8] $\begin{array}{|c|c|c|}\hline B & M & \mathrm{E} \\ \hline \mathrm{E} & T & \mathrm{E} \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [9] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & T \\ \hline \mathrm{E} & M & T \\ \hline \# & \mathrm{E} & \mathrm{E}/T \\ \hline \end{array} \Rightarrow 0$

[10] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \neq M & M & M \\ \hline \# & \# & \#_{br} \\ \hline \end{array} \Rightarrow \#_{br}$ [11] $\begin{array}{|c|c|c|}\hline \# & \mathrm{E}/_{M} & \# \\ \hline \neq M & M & \mathrm{E} \\ \hline \# & \mathrm{E} & B_{br} \\ \hline \end{array} \Rightarrow B_{br}$ [12] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \neq M & M & B_{r} \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow B_{r}$

[13] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline B & M & T_{r} \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow T_{r}$ [14] $\begin{array}{|c|c|c|}\hline \neq \mathrm{E} & \# & \# \\ \hline \mathrm{E} & M & T_{r} \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow T_{r}$ [15] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline M & B & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow M_{def}$

[16] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline B & M & \mathrm{E} \\ \hline \mathrm{E} & T_{b} & \mathrm{E} \\ \hline \end{array} \Rightarrow T_{b}$ [17] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline M & M & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [18] $\begin{array}{|c|c|c|}\hline M & M & \mathrm{E} \\ \hline \mathrm{E} & B & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow M_{def}$

[19] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline M & T & \mathrm{E} \\ \hline \# & \# & \mathrm{E}/T \\ \hline \end{array} \Rightarrow 0$ [20] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline M & T & \mathrm{E} \\ \hline \# & \neq \mathrm{E} & B/_{M} \\ \hline \end{array} \Rightarrow 0$ [21] $\begin{array}{|c|c|c|}\hline M & \# & \# \\ \hline \mathrm{E} & T & \mathrm{E} \\ \hline \# & \# & \mathrm{E}/T \\ \hline \end{array} \Rightarrow 0$

[22] $\begin{array}{|c|c|c|}\hline M & \neq T & \# \\ \hline \mathrm{E} & T & \mathrm{E} \\ \hline \# & \neq \mathrm{E} & B/_{M} \\ \hline \end{array} \Rightarrow 0$ [23] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \# \\ \hline M & T^{*} & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow M_{def}$ [24] $\begin{array}{|c|c|c|}\hline M & \mathrm{E} & \# \\ \hline \mathrm{E} & T^{*}/_{B} & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow M_{def}$

[25] $\begin{array}{|c|c|c|}\hline \neq \mathrm{E} & \neq \mathrm{E} & \# \\ \hline M & T^{*} & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow {}_{M_{def}}$ [26] $\begin{array}{|c|c|c|}\hline \# & \mathrm{E} & \mathrm{E} \\ \hline B/T & 0 & \mathrm{E} \\ \hline \mathrm{E} & B & B/T \\ \hline \end{array} \overset{\overline{d}}{\Rightarrow} 0/_{M_{def}}$ $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline B/T & M & \mathrm{E} \\ \hline \mathrm{E} & B_b & B/T \\ \hline \end{array} \Rightarrow B_b$

[28] $\begin{array}{|c|c|c|}\hline B/T & M & \mathrm{E} \\ \hline \mathrm{E} & B & B/T \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [29] $\begin{array}{|c|c|c|}\hline \# & \mathrm{E} & \mathrm{E} \\ \hline B & 0 & \mathrm{E} \\ \hline \mathrm{E} & T & \mathrm{E} \\ \hline \end{array} \overset{\overline{d}}{\Rightarrow} 0/_{M_{def}}$

A.3 State Transitions for Selection

[1] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \neq \mathrm{E} & B & \mathrm{E} \\ \hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$ [2] $\begin{array}{|c|c|c|}\hline \neq \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \mathrm{E} & B & \mathrm{E} \\ \hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$ [3] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \# \\ \hline \mathrm{E} & T & \mathrm{E} \\ \hline \# & \mathrm{E}/T & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$

[4] $\begin{array}{|c|c|c|}\hline T & \# & \# \\ \hline \mathrm{E} & T & \mathrm{E} \\ \hline \# & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$ [5] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \# \\ \hline \mathrm{E} & T & \mathrm{E} \\ \hline \# & \# & T \\ \hline \end{array} \Rightarrow 0$ [6] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \# \\ \hline \mathrm{E} & B & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$

[7] $\begin{array}{|c|c|c|}\hline \mathrm{E} & M & \mathrm{E} \\ \hline \mathrm{E} & \neq \mathrm{E} & \mathrm{E} \\ \hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$ [8] $\begin{array}{|c|c|c|}\hline \# & \mathrm{E} & \# \\ \hline \mathrm{E} & M & \mathrm{E} \\ \hline \# & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$ [9] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \# \\ \hline \# & M & \mathrm{E} \\ \hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$

[10] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \# \\ \hline \mathrm{E} & M & \mathrm{E} \\ \hline \# & \# & \neq T \\ \hline \end{array} \Rightarrow 0$ [11] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \# \\ \hline \mathrm{E} & M & \neq T \\ \hline \# & \# & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$ [12] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \mathrm{E} & M & \mathrm{E} \\ \hline \mathrm{E} & B/T & \# \\ \hline \end{array} \Rightarrow 0$

[13] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline B & M & \mathrm{E} \\ \hline \mathrm{E} & B & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$ [14] $\begin{array}{|c|c|c|}\hline \# & \neq B & \# \\ \hline M & B & \mathrm{E} \\ \hline \# & T/\mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$ [15] $\begin{array}{|c|c|c|}\hline \mathrm{E} & B & \# \\ \hline \mathrm{E} & B & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$

[16] $\begin{array}{|c|c|c|}\hline M & T & \# \\ \hline \mathrm{E} & B/M & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [17] $\begin{array}{|c|c|c|}\hline \mathrm{E} & M & B/T \\ \hline \mathrm{E} & B & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [18] $\begin{array}{|c|c|c|}\hline T/\mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline T/\mathrm{E} & B & T/\mathrm{E} \\ \hline \mathrm{E} & \mathrm{E} & T/\mathrm{E} \\ \hline \end{array} \Rightarrow 0$

[19] $\begin{array}{|c|c|c|}\hline T/\mathrm{E} & \mathrm{E} & \# \\ \hline T/\mathrm{E} & G/K & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$

A.3.1 Selection from Random Initial Configuration

[20] $\begin{array}{|c|c|c|}\hline \mathrm{E} & \mathrm{E} & \# \\ \hline \mathrm{E} & T & \mathrm{E} \\ \hline \# & T/B & \# \\ \hline \end{array} \Rightarrow 0$ [21] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \# & T & T \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [22] $\begin{array}{|c|c|c|}\hline \# & T & \# \\ \hline \mathrm{E} & T & \mathrm{E} \\ \hline \# & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$

[23] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline B & T & B \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [24] $\begin{array}{|c|c|c|}\hline \# & \mathrm{E} & \# \\ \hline \mathrm{E} & M & \mathrm{E} \\ \hline \# & \mathrm{E} & M \\ \hline \end{array} \Rightarrow 0$ [25] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \mathrm{E} & M & \mathrm{E} \\ \hline \mathrm{E} & \mathrm{E}/M & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$

[26] $\begin{array}{|c|c|c|}\hline M & \mathrm{E} & \# \\ \hline \mathrm{E} & M & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [27] $\begin{array}{|c|c|c|}\hline \mathrm{E} & T/B & \# \\ \hline \mathrm{E} & T/B & \mathrm{E} \\ \hline \mathrm{E} & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow 0$ [28] $\begin{array}{|c|c|c|}\hline \mathrm{E} & T & \# \\ \hline \mathrm{E} & B/M & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$

[29] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \# & \mathrm{E} \neq 0 & \# \\ \hline \# & \mathrm{E} \neq 0 & \# \\ \hline \end{array} \Rightarrow 0$ [30] $\begin{array}{|c|c|c|}\hline \# & T/B & \# \\ \hline \mathrm{E} & B & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [31] $\begin{array}{|c|c|c|}\hline \# & \mathrm{E} & \# \\ \hline \# & T^0 & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$

[32] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \# & G & P/G \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [33] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \# & K & G/P \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [34] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \# & P & K/P \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$

[35] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \# & G & \mathrm{E} \\ \hline \# & \mathrm{E} & P/G \\ \hline \end{array} \Rightarrow 0$ [36] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \# & K & \mathrm{E} \\ \hline \# & \mathrm{E} & G/P \\ \hline \end{array} \Rightarrow 0$ [37] $\begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \# & P & \mathrm{E} \\ \hline \# & \mathrm{E} & K/P \\ \hline \end{array} \Rightarrow 0$

A.4 State Transitions for Reproduction

- The state B^* is a very concise representation for a B-state that is non-deterministically generated in the offspring in order to: recreate a configuration of two consecutive B-states that is already present in its parents; if this is not possible, recreate any individual B-state of the parents; otherwise, create any B-state. Naturally, any of these conditional actions must conform to the definition of a well-formed agent, according to Figure 3.2.

[1] $\begin{array}{|c|c|c|}\hline \mathrm{E} & T_t^* & P \\ \hline \mathrm{E} & 0 & 0 \\ \hline \mathrm{E} & T_b^* & P \\ \hline \end{array} \Rightarrow T^0$ [2] $\begin{array}{|c|c|c|}\hline T & P/M & \neq \mathrm{E} \\ \hline T^0 & 0 & 0 \\ \hline T & P/M & \# \\ \hline \end{array} \Rightarrow P^0$ [3] $\begin{array}{|c|c|c|}\hline \# & B/M & \neq \mathrm{E} \\ \hline B & 0 & \mathrm{E} \\ \hline \neq \mathrm{E} & B/M & \# \\ \hline \end{array} \overset{d}{\Rightarrow} B^*$

[4] $\begin{array}{|c|c|c|}\hline \# & T_t & \mathrm{E} \\ \hline B & 0 & \mathrm{E} \\ \hline \neq \mathrm{E} & B/M & \# \\ \hline \end{array} \overset{d}{\Rightarrow} B^*/T_t$ [5] $\begin{array}{|c|c|c|}\hline \# & B/M & \neq \mathrm{E} \\ \hline B & 0 & \mathrm{E} \\ \hline M/B & T_b & \mathrm{E} \\ \hline \end{array} \overset{d}{\Rightarrow} B^*/T_b$ $\begin{array}{|c|c|c|}\hline \# & T_t & \mathrm{E} \\ \hline B & 0 & \mathrm{E} \\ \hline M/B & T_b & \mathrm{E} \\ \hline \end{array} \overset{d}{\Rightarrow} T_t/T_b$

[7] $\begin{array}{|c|c|c|}\hline \# & T_t & \mathrm{E} \\ \hline B & 0 & \mathrm{E} \\ \hline \# & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \Rightarrow T_t$ [8] $\begin{array}{|c|c|c|}\hline \mathrm{E}/T & \mathrm{E} & \mathrm{E} \\ \hline B & 0 & \mathrm{E} \\ \hline M/B & T_b & \mathrm{E} \\ \hline \end{array} \Rightarrow T_b$ $\begin{array}{|c|c|c|}\hline \# & B/M & \neq \mathrm{E} \\ \hline B & 0 & \mathrm{E} \\ \hline \# & \mathrm{E} & \mathrm{E} \\ \hline \end{array} \overset{d}{\Rightarrow} B^*/T_{def}$

[10] $\begin{array}{|c|c|c|}\hline \mathrm{E}/T & \mathrm{E} & \mathrm{E} \\ \hline B & 0 & \mathrm{E} \\ \hline \neq \mathrm{E} & B/M & \# \\ \hline \end{array} \overset{d}{\Rightarrow} B^*/T_{def}$ [11] $\begin{array}{|c|c|c|}\hline \# & B/M & \neq \mathrm{E} \\ \hline \# & B & \mathrm{E} \\ \hline \# & \mathrm{E} & M \\ \hline \end{array} \Rightarrow 0$ [12] $\begin{array}{|c|c|c|}\hline \# & \mathrm{E} & M \\ \hline \# & B & \mathrm{E} \\ \hline \# & B/M & \# \\ \hline \end{array} \Rightarrow 0$

[13]

#	$B/_M$	#
E	T	E
E	T	B

$\Rightarrow 0$

[14]

#	#	#
B	M	E
#	#	#

$\Rightarrow T_{def}$

A.5 State Transitions for Development

- The superscript $+$, as in T_c^+, means the *aged* T-state.
- The death of an agent is implicit to ageing. It takes place through the built-in selection process when the state of the agent's head becomes the background state. Therefore, the outcome of ageing can be another T-state or the *0*-state; hence the notation "$T^+/0$" being used.

A.5.1 Neonatal Development

[1]

#	#	#
T^0	P^0	G
#	#	#

$\Rightarrow P_{def}$

[2]

#	#	#
#	T^0	$P \neq P^0$
#	#	#

$\Rightarrow T_{def}$

A.5.2 Adult Development: Ageing and Death

[3]

E	#	#
E	T_c^*	$M/(P \neq P^0)$
#	#	#

$\overline{\Rightarrow}^d T_c/(T_c^+/0)$

[4]

E	E	#
E	T_c^*	E
#	E	$M/(P \neq P^0)$

$\overline{\Rightarrow}^d T_c/(T_c^+/0)$

[5]

M	B/T	#
E	T_c^*	$P \neq P^0$
#	#	#

$\overline{\Rightarrow}^d T_c/(T_c^+/0)$

[6]

M	E	#
M	T_c^*	$P \neq P^0$
#	#	#

$\overline{\Rightarrow}^d T_c/(T_c^+/0)$

[7]

M	E	#
M	T_c^*	E
#	E	$P \neq P^0$

$\overline{\Rightarrow}^d T_c/(T_c^+/0)$

[8]

E	E	E
E	M	T_r^*
#	E	E/T_{br}

$\overline{\Rightarrow}^d T_r/(T_r^+/0)$

[9]

#	$E/_M$	#
$\neq M$	M	E
#	E	T_{br}^*

$\overline{\Rightarrow}^d T_{br}/(T_{br}^+/0)$

Appendix B

The C code that implements Enact in Cellsim 2.5

```
#include "nborhood.h"
#include <stdio.h>
#include <values.h>
#define ON  1
#define OFF 0

#define Tmin  1              /* minimum value for a Terminal-state */
#define Tmax  7              /* maximum value for a Terminal-state */
#define Dmin  8              /* minimum value for a boDy_state */
#define Dmax  20             /* maximum value for a boDy-state */
#define E00   0              /* Background environment-state */
#define Emin  21             /* minimum value for an Environment-state */
#define Emax  24             /* maximum value for an Environment-state */
#define Mmin  25             /* minimum value for a Movement-state */
#define Mmax  255            /* maximum value for a Movement-state */

#define PGK          ON
#define MUT_R        2       /* rate of D_star in relation to D_mutant during repr. */
#define DST_R        MAXINT  /* rate of D_star in relation to T during reproduct. */
#define BTAIL_UP_R   MAXINT  /* rate of body movement (in relation to no movement) */

#define Mdef  Mmin
#define Ddef  Dstar()
#define Tdef  Tmin+2
#define Pdef  Pmin+1

#define T00    Tmin
#define TLmin  3              /* TL represents the agents
#define TLmax  4
#define TDmin  5
#define TDmax  6

#define P00   Pmin
#define Pmin  8
#define Pmax  12

#define Gmin  13
#define Gmax  16

#define Kmin  17
#define Kmax  20
```

```
#define DEATH  ON

/*
The parameter "parm1" sets the probability that, at each iteration, the
agents will NOT age (thus having the chance of living longer). The
probability of an agent getting older at each iteration is given by:

                              1                   1
              P(ageing) = ---------  * 100% * -------
                          parm1 + 1            parm2

"parm2" is just a scaling parameter to allow a wider range of
expected life spans.
The "expected life span" is: (TLmax - TLmin) * (parm1 + 1) * parm2
                         or: (TDmax - TDmin) * (parm1 + 1) * parm2
*/

byte alife_dynamics(), Rand2(), Dstar(), Older(), Dev_T(),
     get_bblock(), get_any_in_Neighb(), get_any_at_all(), get_mutant(),
     D(), T(), M(), E(), P(), G(), K(), TD(), TL();
int L_MV(), D_MV(), L_MV_CONF(), D_MV_CONF();
static int i;
static byte s;

void init_function()
{
        update_function = alife_dynamics;
        parm1 = 29;
        parm2 = 5;
}

byte alife_dynamics(nbors)
moore_nbors *nbors;
{
Get_moore_nbors;


/*****************************************/
/************* SELECTION *****************/
/*****************************************/

/* sel_0: Just to speed up computation */
    if( !tl && !t && !tr &&
        !l  && !c && !r  &&
        !bl && !b && !br )                          return (byte)0;

/* sel_1 */
    if( E(tl) && E(t) && E(tr) &&
        !E(l) && D(c) && E(r)  &&
        E(bl) && E(b) && E(br) )                    return (byte)0;
    if( !E(tl) && E(t) && E(tr) &&
        E(l)   && D(c) && E(r)  &&
        E(bl)  && E(b) && E(br) )                   return (byte)0;
    if( E(tl) && E(t)  &&
        E(l)  && T(c)  && E(r) &&
        (E(b) || T(b)) && E(br) )                   return (byte)0;
/* sel_4 */
```

```
    if( T(tl) &&
        E(l)  && T(c) && E(r) &&
        E(b)  && E(br) )                                return (byte)0;
    if( E(tl) && E(t) &&
        E(l)  && T(c) && E(r) &&
        T(br) )                                         return (byte)0;
    if( E(tl) && E(t) &&
        E(l)  && D(c) )                                 return (byte)0;
/* sel_7 */
    if( E(tl) && M(t)  && E(tr) &&
        E(l)  && !E(c) && E(r)  &&
        E(bl) && E(b)  && E(br) )                       return (byte)0;
    if( E(t) &&
        E(l) && M(c) && E(r) &&
        E(b) && E(br) )                                 return (byte)0;
    if( E(tl) && E(t) && E(tr) &&
        M(c)  && E(r) &&
        E(bl) && E(b) && E(br) )                        return (byte)0;
/* sel_10 */
    if( E(tl) && E(t) &&
        E(l)  && M(c) && E(r) &&
        !T(br) )                                        return (byte)0;
    if( E(tl) && E(t) &&
        E(l)  && M(c) && !T(r) &&
        E(br) )                                         return (byte)0;
    if( E(tl) && E(t) && E(tr) &&
        E(l)  && M(c) && E(r)  &&
        E(bl) && (D(b) || T(b)) )                       return (byte)0;
/* sel_13 */
    if( E(tl) && E(t) && E(tr) &&
        D(l)  && M(c) && E(r)  &&
        E(bl) && D(b) && E(br) )                        return (byte)0;
    if( !D(t) &&
        M(l)  && D(c) && E(r)  &&
        (T(b) || E(b)) && E(br) )                       return (byte)0;
    if( E(tl) && D(t) &&
        E(l)  && D(c) )                                 return (byte)0;
/* sel_16 */
    if( M(tl) && T(t) &&
        E(l)  && (D(c) || M(c)) )                       return (byte)0;
    if( E(tl) && M(t) && (D(tr) || T(tr)) &&
        E(l)  && D(c) )                                 return (byte)0;

/************ WITH "PGK" AGENTS ******************/
/* sel_18: */
    if( (T(tl) || E(tl)) && E(t) && E(tr) &&
        (T(l) || E(l))   && D(c) && (T(r) || E(r)) &&
        E(bl) && E(b)    && (T(br) || E(br)) )          return (byte)0;
    if( (T(tl) || E(tl)) && E(t) &&
        (T(l) || E(l))   && (G(c) || K(c)) )            return (byte)0;

/************* FROM RANDOM CONFIGURATIONS ******************/
/* sel_20 */
    if( E(tl) && E(t) &&
        E(l)  && T(c) && E(r) &&
        (T(b) || D(b)) )                                return (byte)0;
    if( T(c) && T(r) )                                  return (byte)0;
```

```
    if( T(t) &&
        E(l) && T(c) && E(r) &&
        E(b) && E(br) )                                 return (byte)0;
/* sel_23: */
    if( D(l) && T(c) && D(r) )                          return (byte)0; }
    if( E(t) &&
        E(l) && M(c) && E(r) &&
        E(b) && M(br) )                                 return (byte)0;
    if( E(l)  && M(c) && E(r) &&
        E(bl) && (E(b) || M(b)) && E(br) )              return (byte)0;
/* sel_26 */
    if( M(tl) && E(t) &&
        E(l)  && M(c) )                                 return (byte)0;
    if( E(tl) && (T(t) || D(t)) &&
        E(l)  && (T(c) || D(c)) && E(r) &&
        E(bl) && E(b) && E(br) )                        return (byte)0;
    if( E(tl) && T(t) &&
        E(l)  && (D(c) || M(c)) )                       return (byte)0;
/* sel_29: prevents agents from being blocked by a "stack" of E-states (E>0). */
    if( E(c) && c && E(b) && b )                        return (byte)0;
    if( (D(t) || T(t)) &&
        E(l) && D(c) )                                  return (byte)0;
    if( E(t) && T(c)==T00 )                             return (byte)0;


/********** WITH "PGK" AGENTS, AND FROM RANDOM CONFIGURATIONS *********/
/* sel_32 */
    if( G(c) && (P(r) || G(r)) )                        return (byte)0;
    if( K(c) && (G(r) || P(r)) )                        return (byte)0;
    if( P(c) && (K(r) || P(r)) )                        return (byte)0;
/* sel_35 */
    if( G(c) && E(r) &&
        E(b) && (P(br) || G(br)) )                      return (byte)0;
    if( K(c) && E(r) &&
        E(b) && (G(br) || P(br)) )                      return (byte)0;
    if( P(c) && E(r) &&
        E(b) && (K(br) || P(br)) )                      return (byte)0;
/* sel_38 */
/*    if( T(l)==T00 && M(c) )                           return (byte)0; */

/*****************************************/
/************* MOVEMENT ******************/
/*****************************************/
/* mov_1 */
    if( E(tl) && E(t) && E(tr) &&
        E(l)  && !c && T(r) &&
        E(bl) && E(b) && (E(br) || T(br)) )   return Rand2(Mdef, L_MV(r), 0);
    if( E(tl) && E(t) && E(tr) &&
        E(l)  && !c && E(r)  &&
        E(bl) && E(b) && T(br) )              return Rand2(Mdef, D_MV(br), 0);
    if( E(tl)  && M(t) && E(tr) &&
        E(l)   && M(c) && T(r)  &&
        !E(bl) && E(b) && (E(br) || T(br)) )  return (byte)0;
/* mov_4 */
    if( E(tl) && M(t) && E(tr) &&
        E(l)  && M(c) && T(r)  &&
        E(bl) && E(b) && (E(br) || T(br)) )  return Rand2(Mdef, L_MV_CONF(r), 0);
```

```
    if( !E(tl) && E(t) &&
        E(l)   && M(c) && E(r) &&
        E(bl)  && M(b) && T(br) )           return (byte)0;
  if( PGK )
  {
    if( E(t)  && (T(tr) || D(tr)) &&
        E(l)  && M(c)   && E(r) &&
        E(bl) && M(b)   && T(br) )          return (byte)0;
  }
  else
  {
    if( E(t)  && !E(tr) &&
        E(l)  && M(c)   && E(r) &&
        E(bl) && M(b)   && T(br) )          return (byte)0;
  }
/* mov_7 */
    if( E(tl) && E(t) && E(tr) &&
        E(l)  && M(c) && E(r)  &&
        E(bl) && M(b) && T(br) )            return Rand2(Mdef, D_MV_CONF(br), 0);
    if( D(tl) && M(t) && E(tr) &&
        E(l)  && T(c) && E(r) )             return (byte)0;
    if( E(tl) && E(t) && T(tr) &&
        E(l)  && M(c) && T(r)  &&
        E(b)  && (E(br) || T(br)) )         return (byte)0;
/* mov_10 */
    if( (!M(l)) && M(c) && M(r) )           return br;
    if( (E(t) || M(t))  &&
        (!M(l)) && M(c) && E(r) &&
        E(b)    && D(br) )                  return br;
    if( (!M(l)) && M(c) && D(r) )           return r;
/* mov_13 */
    if( D(l) && M(c) && T(r) )              return r;
    if( !E(tl) &&
        E(l)   && M(c) && T(r) )            return r;
    if( M(l) && D(c) )                      return (byte)Mdef;

/* mov_16 */
    if( D(l)  && M(c) && E(r) &&
        E(bl) && T(b) && E(br) )            return b;
    if( M(l) && M(c) )                      return (byte)0;
    if( M(tl) && M(t) && E(tr) &&
        E(l)  && D(c) )                     return (byte)Mdef;
/* mov_19 */
    if( M(l) && T(c) && E(r) &&
        (E(br) || T(br)) )                  return (byte)0;
    if( M(l)  && T(c) && E(r) &&
        !E(b) && (D(br) || M(br)) )         return (byte)0;
    if( M(tl) &&
        E(l)  && T(c) && E(r) &&
        (E(br) || T(br)) )                  return (byte)0;
/* mov_22 */
    if( M(tl) && (!T(t)) &&
        E(l)  && T(c) && E(r) &&
        !E(b) && (D(br) || M(br)) )               return (byte)0;
    if( E(tl) && E(t) &&
        M(l)  && T(c) && T(c)!=T00 )              return (byte)Mdef;
    if( M(tl) && E(t) &&
```

```
        E(l)  && ((T(c) && T(c)!=T00) || D(c)) )   return (byte)Mdef;
/* mov_25  */
    if( !E(tl) && !E(t) &&
        M(l)   && T(c) && T(c)!=T00 )          return (byte)Mdef;
    if( E(t)  && E(tr) &&
        (T(l) || D(l)) && !c && E(r) &&
        E(bl) && D(b)  && (T(br) || D(br)) )   return Rand2(Mdef, BTAIL_UP_R, 0);
    if( (T(l) || D(l)) && M(c) && E(r) &&
        E(bl) && D(b)  && (T(br) || D(br)) )   return b;
/* mov_28 */
    if( (T(tl) || D(tl)) && M(t) && E(tr) &&
        E(l) && D(c) && (T(r) || D(r)) )       return (byte)0;
    if( E(t)  && E(tr) &&
        D(l)  && !c  && E(r) &&
        E(bl) && T(b)  && E(br) )              return Rand2(Mdef, BTAIL_UP_R, 0);


/*****************************************/
/************* REPRODUCTION ***************/
/*****************************************/
/* rep_1 */
  if( PGK )
  {
    if( E(tl) && T(t) && T(t)!=T00 && P(tr)  &&
        E(l)  && !c && !r  &&
        E(bl) && T(b) && T(b)!=T00 && P(br) )  return T00;
    if( T(tl) && (P(t) || M(t)) && !E(tr) &&
        T(l)==T00 && !c && !r &&
        T(bl) && (P(b) || M(b)) )              return P00;
  }
  else
  {
    if( E(tl) && T(t) && D(tr)  &&
        E(l)  && !c && !r  &&
        E(bl) && T(b) && D(br) )               return Rand2(t, 1, b);
    if( T(tl) && (D(t) || M(t)) && !E(tr) &&
        T(l) && !c && !r &&
        T(bl) && (D(b) || M(b)) )              return Rand2(Ddef, DST_R, Ddef);
  }
    if( (D(t) || M(t)) && !E(tr) &&
        D(l) && !c && E(r) &&
        !E(bl) && (D(b) || M(b)) )             return Rand2(Ddef, DST_R, Ddef);
/* rep_4 */
    if( T(t) && E(tr)  &&
        D(l) && !c && E(r) &&
        !E(bl) && (D(b) || M(b)) )             return Rand2(Ddef, DST_R, t);
    if( (D(t) || M(t)) && !E(tr) &&
        D(l) && !c && E(r) &&
        (M(bl) || D(bl)) && T(b) && E(br) )    return Rand2(Ddef, DST_R, b);
    if( T(t) && E(tr) &&
        D(l) && !c  && E(r) &&
        (M(bl) || D(bl)) && T(b) && E(br) )    return Rand2(t, 1, b);
/* rep_7 */
    if( T(t) && E(tr)  &&
        D(l) && !c && E(r) &&
        E(b) && E(br) )                        return t;
    if( (T(tl) || !tl) && E(t) && E(tr) &&
```

```
        D(l) && !c && E(r) &&
        (M(bl) || D(bl)) && T(b) && E(br) )   return b;
    if( (D(t) || M(t)) && !E(tr) &&
        D(l) && !c && E(r) &&
        E(b) && E(br) )                       return Rand2(Ddef, DST_R, Tdef);
/* rep_10 */
    if( (T(tl) || !tl) && E(t) && E(tr) &&
        D(l) && !c && E(r) &&
        !E(bl) && (D(b) || M(b)) )   return Rand2(Ddef, DST_R, Tdef);
    if( (M(t) || D(t)) && !E(tr) &&
        D(c) && E(r) &&
        E(b) && M(br) )              return (byte)0;
    if( E(t) && M(tr) &&
        D(c) && E(r) &&
        (M(b) || D(b)) )             return (byte)0;
/* rep_13 */
    if( (D(t) || M(t)) &&
        E(l) && T(c) && E(r) &&
        E(bl) && T(b) && D(br) )     return (byte)0;
    if( D(l) && M(c) && E(r) )       return (byte)Tdef;


/*****************************************/
/************* DEVELOPMENT ***************/
/*****************************************/
/* NEONATE DEVELOPMENT */
/* dev_1 */
    if( T(l)==T00 && P(c)==P00 && G(r) )                  return (byte)Pdef;
    if( T(c)==T00 && P(r) && P(r)!=P00 )                  return (byte)Tdef;

/* ADULT DEVELOPMENT: AGEING AND DEATH */
/* dev_3: While not moving */
    if( E(tl) &&
        E(l)  && T(c) && T(c)!=T00 &&
        ((P(r) && P(r)!=P00) || M(r)) )
                                        return Rand2(c, parm1*parm2, Older(c));
    if( E(tl) && E(t) &&
        E(l)  && T(c) && T(c)!=T00 && E(l) &&
        E(b)  && ((P(br) && P(br)!=P00) || M(br)) )
                                        return Rand2(c, parm1*parm2, Older(c));
    if( M(tl) && (D(t) || T(t))   &&
        E(l)  && T(c) && T(c)!=T00 &&
        (P(r) && P(r)!=P00) )
                                        return Rand2(c, parm1*parm2, Older(c));

/* dev_6: While trying to move in BOTH directions...*/
    if( M(tl) && E(t) &&
        M(l)  && T(c) && T(c)!=T00 && P(r) && P(r)!=P00 )
                                        return Rand2(c, parm1*parm2, Older(c));
    if( M(tl) && E(t) &&
        M(l)  && T(c) && T(c)!=T00  && E(r) &&
        E(b)  && P(br) && P(br)!=P00 )
                                        return Rand2(c, parm1*parm2, Older(c));

/* dev_8: ...or just in EITHER of them. */
    if( E(tl) && E(t) && E(tr) &&
        E(l)  && M(c) && T(r) && T(r)!=T00 &&
```

```
        E(b)  && (E(br) || T(br)) )
                                        return Rand2(r, parm1*parm2, Older(r));
    if( (E(t) || M(t)) &&
        !M(l) && M(c)  && E(r) &&
        E(b)  && T(br) && T(br)!=T00 )
                                        return Rand2(br, parm1*parm2, Older(br));


/****************************************/
/*********** OTHERWISE... ***************/
/****************************************/
    else return c;
}

/*****************************************************************************/
/* Predicate returning state "s" if it is a D-state, and 0 if not.  */

byte D(s)
byte s;

{
if (s>=Dmin && s<=Dmax) return s;
else                    return (byte)0;
}

/*****************************************************************************/
/* Predicate returning state "s" if it is a T-state, and 0 if not.  */

byte T(s)
byte s;

{
if (s>=Tmin && s<=Tmax) return s;
else                    return (byte)0;
}

/*****************************************************************************/
/* Predicate returning state "s" if it is an M-state, and 0 if not.  */

byte M(s)
byte s;

{
if (s>=Mmin && s<=Mmax) return s;
else                    return (byte)0;
}

/*****************************************************************************/
/* Predicate returning state "s" if it is an E-state, and 0 if not.  */

byte E(s)
byte s;

{
if   (s>=Emin && s<=Emax) return s;
else if(s==E00)           return 1;  /* any number>0 actually */
     else                 return (byte)0;
```

```
}

/*****************************************************************************/
/* Predicate returning state "s" if it is a P-state, and 0 if not.  */

byte P(s)
byte s;

{
if (s>=Pmin && s<=Pmax) return s;
else                    return (byte)0;

}

/*****************************************************************************/
/* Predicate returning state "s" if it is a G-state, and 0 if not.  */

byte G(s)
byte s;

{
if (s>=Gmin && s<=Gmax) return s;
else                    return (byte)0;

}

/*****************************************************************************/
/* Predicate returning state "s" if it is a K-state, and 0 if not.  */

byte K(s)
byte s;

{
if (s>=Kmin && s<=Kmax) return s;
else                    return (byte)0;

}

/*****************************************************************************/
/*  If rate>0: returns state s1  "rate"  times as often as s2.
    If rate<0: returns state s2 "|rate|" times as often as s1.
    If rate=0: returns s2.
     ND: In terms of probability, if rate>0, it returns state s1 with
                      rate                                 1
         probability  ------ x 100%  and s2 with prob. ------ x 100%
                     rate+1                            rate+1
*/

byte Rand2(s1, rate, s2)
int s1, rate, s2;

{
if (rate==0)                              return (byte)s2;
if (rate>0) { if (random() % (rate+1))    return (byte)s1;
              else                        return (byte)s2; }
if (rate<0) { if (random() % ((-rate)+1)) return (byte)s2;
```

```
             else                          return (byte)s1; }
}

/*************************************************************************/
/* Returns a D-star state. If PGK=1, the body of the agents has to be
   considered as formed by P_G_K states. If PGK=0, the body cells are made
   of the ordinary, general D-states. */

byte Dstar()

{
byte D_star, D_mutant, Rand2(), Xmin, Xmax,
     get_bblock(), get_any_in_Neighb(), get_any_at_all(), get_mutant();

if( PGK )
  {
   if( P(1) ) { Xmin=(byte)Gmin; Xmax=(byte)Gmax;
                D_star=get_any_in_Neighb(G); }
   else if( G(1) )
              { Xmin=(byte)Kmin; Xmax=(byte)Kmax;
                D_star=get_any_in_Neighb(K); }
        else if( K(1) )
              { Xmin=(byte)Kmin; Xmax=(byte)Kmax;
                D_star=get_bblock(K);
                if(D_star==0) D_star=get_any_in_Neighb(K); }
  }
else  { Xmin=(byte)Dmin; Xmax=(byte)Dmax;
        D_star=get_bblock(D);
        if(D_star==0) D_star=get_any_in_Neighb(D); }

if(D_star==0) D_star=get_any_at_all(Xmin, Xmax);
D_mutant=get_mutant(Xmin, Xmax);
return Rand2(D_star, MUT_R, D_mutant);
}

/*************************************************************************/
/* Randomly choose a D-state of the parents, from the ones that
   recreates in the offspring an existing, non-deleterious building blocks
   in them. */

byte get_bblock(X)
byte (*X)();

{
byte X_star, bblock[4];
int i;

for(i=0; i<4; i++) bblock[i]=0;
X_star=0;
if (l==tl && (*X)(t) ) bblock[0]=t;
if (l==t  && (*X)(tr)) bblock[1]=tr;
if (l==bl && (*X)(b) ) bblock[2]=b;
if (l==b  && (*X)(br)) bblock[3]=br;
if( bblock[0] + bblock[1] + bblock[2] + bblock[3] > 0 )
    while((X_star=bblock[random() % 4])==0);
return X_star;
}
```

```
/*****************************************************************************/
/* If no building blocks can be created again, randomly choose any of the
   D-state of the parents. */

byte get_any_in_Neighb(X)
byte (*X)();

{
byte X_star, neigh_Xs[6];
int i;

for(i=0; i<6; i++) neigh_Xs[i]=0;
X_star=0;
if( (*X)(tl) ) neigh_Xs[0]=tl;
if( (*X)(t)  ) neigh_Xs[1]=t;
if( (*X)(tr) ) neigh_Xs[2]=tr;
if( (*X)(bl) ) neigh_Xs[3]=bl;
if( (*X)(b)  ) neigh_Xs[4]=b;
if( (*X)(br) ) neigh_Xs[5]=br;
if((neigh_Xs[0]+neigh_Xs[1]+neigh_Xs[2]+
    neigh_Xs[3]+neigh_Xs[4]+neigh_Xs[5])!=0)
            while((X_star=neigh_Xs[random() % 6])==0);
return X_star;
}

/*****************************************************************************/
/* If there are no D-states in the parents, just get any of the
   possible D-states. */

byte get_any_at_all(Xmin, Xmax)
byte Xmin, Xmax;

{
byte X_star, Rand2();

if(Xmax-Xmin==0)      X_star=Xmin;
else if(Xmax-Xmin==1) X_star=Rand2(Xmin, 1, Xmax);
     else             X_star=(random() % (Xmax-Xmin+1)) + Xmin;
return X_star;
}

/*****************************************************************************/
/* During reproduction there's always chance of a mutation to occur. */

byte get_mutant(Xmin, Xmax)
byte Xmin, Xmax;

{
byte X_mutant, Rand2();

if(Xmax-Xmin==0)      X_mutant=Xmin;
else if(Xmax-Xmin==1) X_mutant=Rand2(Xmin, 1, Xmax);
     else             X_mutant=(random() % (Xmax-Xmin+1)) + Xmin;
return X_mutant;
}
```

```
/*************************************************************************/
/* Defines the amount of leftward movement for each kind of head,
   AFTER the agent HAS TRIED to move in BOTH directions.
   It returns the rate at which an agent will make an attempt to move
   leftwards, independently of the diagonal movement. */

int L_MV_CONF(s)
byte s;

{
byte TL(), TD();

if      (TL(s)) return  MAXINT;
else if (TD(s)) return  0;
     else       return  1;
}

/*************************************************************************/
/* Defines the amount of diagonal movement for each kind of head,
   AFTER the agent HAS TRIED to move in BOTH directions.
   It returns the rate at which an agent will make an attempt to move
   diagonally, independently of the leftward movement. */

int D_MV_CONF(s)
byte s;

{
byte TL(), TD();

if      (TL(s)) return  0;
else if (TD(s)) return  MAXINT;
     else       return  1;
}

/*************************************************************************/
/* Defines the amount of leftward movement for each kind of head, by
   returning the rate at which an agent will make an attempt to move
   leftwards, independently of the diagonal movement.
   With all cases returning 1, it means that an attempt to move in either way
   is equally likely to not making an attempt at all (i.e., M or 0 have the same
   probability). The existence of these new routines is handy
   because they allow to control the movement rates in each direction
   independently; it is even possible to approach the deterministic case
   in which, for example, TD would always move diagonally except when the
   only possibility is the leftward movement. */

int L_MV(s)
byte s;

{
byte TL(), TD();

if      (TL(s)) return  MAXINT;
else if (TD(s)) return  MAXINT;
     else if(s==T00) return  0;   /* T00 is immobile */
          else       return  1;
}
```

```
/*************************************************************************/
/* Defines the amount of diagonal movement for each kind of head, by
   by returning the rate at which an agent will make an attempt to move
   diagonally, independently of the leftward movement. */

int D_MV(s)
byte s;

{
byte TL(), TD();

if      (TL(s)) return  MAXINT;
else if (TD(s)) return  MAXINT;
     else if(s==T00) return  0;   /* T00 is immobile */
          else       return  1;
}

/*************************************************************************/
/* Predicate returning T-state "s" if it moves leftwards, and 0 if not.  */

byte TL(s)
byte s;

{
if( s>=TLmin && s<=TLmax ) return s;
else                       return (byte)0;
}

/*************************************************************************/
/* Predicate returning T-state "s" if it moves  diagonally, and 0 if not.  */

byte TD(s)
byte s;

{
if( s>=TDmin && s<=TDmax ) return s;
else                       return (byte)0;
}

/*************************************************************************/
/* An agent gets older according to the following "ageing" sequence:
   TLmin (=Tmin) --> TL(1) ---> ... ---> TL(n) --> TLmax --> 0
   TDmin (=TL+1) --> TD(1) ---> ... ---> TD(n) --> TDmax --> 0
*/

byte Older(s)
byte s;

{
if( (s>=TLmin && s<TLmax) ||
    (s>=TDmin && s<TDmax) )                return (byte)(s+1);
else if( (s==TLmax || s==TDmax) && DEATH)  return (byte)0;
     else                                  return s;
}

/*************************************************************************/
```

```
/*
*/

byte Dev_T(s)
byte s;

{
byte TL();

switch(s)
 {
 case  9: return TL(c) ? (byte)TLmin : (byte)TDmin;
 case 10: return TL(c) ? (byte)TLmin : (byte)TLmin;
 case 11: return TL(c) ? (byte)TDmin : (byte)TDmin;
 case 12: return TL(c) ? (byte)TDmin : (byte)TLmin;
 }
}

/*****************************************************************************/
```

Codification of the State Transition Table of the Turing Machine Implemented in Chapter 4

What follows is the list of state transitions in Enact that are necessary to code for Table 4.2, the state transition table of the Turing machine implemented in Chapter 4. The notation being used is explained in Appendix A.

C.1 State Transition Establishing the End of the Computation

0	0	0
T_0	B_4	T_0
0	0	0

$\Rightarrow 0$

C.2 State Transitions Coding for the Rightward Movement of the Head

E*	$\mathbf{0}_E$	#
0	T_0	B_0
#	#	#

$\Rightarrow T_R$

#	$\mathbf{0}_E$	#
#	M_R	B_0
#	#	#

$\Rightarrow B_1$

#	#	#
#	$\mathbf{0}_E$	#
#	M_R	B_0

$\Rightarrow \mathbf{X}_E$

E*	$\mathbf{0}_E$	#
0	T_0	B_1
#	#	#

$\Rightarrow T_R$

#	$\mathbf{0}_E$	#
#	M_R	B_1
#	#	#

$\Rightarrow B_1$

#	#	#
#	$\mathbf{0}_E$	#
#	M_R	B_1

$\Rightarrow \mathbf{0}_E$

E*	$\mathbf{X}_E$	#
0	T_0	B_2
#	#	#

$\Rightarrow T_R$

#	$\mathbf{X}_E$	#
#	M_R	B_2
#	#	#

$\Rightarrow B_0$

#	#	#
#	$\mathbf{X}_E$	#
#	M_R	B_2

$\Rightarrow \mathbf{X}_E$

E*	$\mathbf{Y}_E$	#
0	T_0	B_0
#	#	#

$\Rightarrow T_R$

#	$\mathbf{Y}_E$	#
#	M_R	B_0
#	#	#

$\Rightarrow B_3$

#	#	#
#	$\mathbf{Y}_E$	#
#	M_R	B_0

$\Rightarrow \mathbf{Y}_E$

E*	$\mathbf{Y}_E$	#
0	T_0	B_1
#	#	#

$\Rightarrow T_R$

#	$\mathbf{Y}_E$	#
#	M_R	B_1
#	#	#

$\Rightarrow B_1$

#	#	#
#	$\mathbf{Y}_E$	#
#	M_R	B_1

$\Rightarrow \mathbf{Y}_E$

E^*	$\mathbf{Y}_E$	#
0	T_0	B_3
#	#	#

$\Rightarrow T_R$

#	$\mathbf{Y}_E$	#
#	M_R	B_3
#	#	#

$\Rightarrow B_3$

#	#	#
#	$\mathbf{Y}_E$	#
#	M_R	B_3

$\Rightarrow \mathbf{Y}_E$

E^*	E^b	#
0	T_0	B_3
#	#	#

$\Rightarrow T_R$

#	E^b	#
#	M_R	B_3
#	#	#

$\Rightarrow B_4$

#	#	#
#	E^b	#
#	M_R	B_3

$\Rightarrow \mathrm{E}^b$

C.3 State Transitions Coding for the Lefward Movement of the Head

E^*	$\mathbf{0}_E$	#
0	T_0	B_2
#	#	#

$\Rightarrow T_L$

E^*	$\mathbf{0}_E$	#
T_L	B_2	M_0
#	#	#

$\Rightarrow B_2$

#	#	#
E^*	$\mathbf{0}_E$	#
T_L	B_2	M_0

$\Rightarrow \mathbf{0}_E$

E^*	$\mathbf{1}_E$	#
0	T_0	B_1
#	#	#

$\Rightarrow T_L$

E^*	$\mathbf{1}_E$	#
T_L	B_1	M_0
#	#	#

$\Rightarrow B_2$

#	#	#
E^*	$\mathbf{1}_E$	#
T_L	B_1	M_0

$\Rightarrow \mathbf{Y}_E$

E^*	$\mathbf{Y}_E$	#
0	T_0	B_2
#	#	#

$\Rightarrow T_L$

E^*	$\mathbf{Y}_E$	#
T_L	B_2	M_0
#	#	#

$\Rightarrow B_2$

#	#	#
E^*	$\mathbf{Y}_E$	#
T_L	B_2	M_0

$\Rightarrow \mathbf{Y}_E$

Appendix D

Details of the Implementation of the Turing Machine Described in Chapter 5

D.1 State Transitions Used for the TM Machinery

Tables D.1 and D.2 present the state transitions required to implement, respectively, the "hardwired" machinery of the Turing machine, and the "software" that implements a particular function being computed, as defined by its state transition table. The cells marked with the symbol # mean that their state is irrelevant in these neighbourhoods. The subscript *def* refers to the default value used in Enact. The subscripts r and br refer to the geographic position of the cell in its neighbourhood. The background state is represented by *0*.

Table D.2 is horizontally divided into two sections, the one on the top referring to the state transition of the TM, and the second accounting for the movement of the head and the manipulation of the tape symbols.

In both tables all occurrences of $E_{\overline{LR}}$ are related to the occurrence of E_{LR}. This is a consequence of the fact that, by default, the agents move leftwards, and therefore the neighbourhood required for the actions of $E_{\overline{LR}}$ corresponding to $(-,-,-)$ are the same one required for E_{LR}.

For present purposes the second collumn in both tables is simply a reference to the *kind* of state transition that appears at each row, or to the original (built-in) state transition from which the state transitions of the first collumn are an instance of. More details about this aspect can be found in Subsection 4.3.2.

D.2 Movement of the Agents

While the preceding section was meant to provide additional details of the implementation of the TM machinery, the current one gives details of the implementation of the agents, with respect to the nature of their movement in the cellular space. This is an important aspect, because a correct computation can only be performed provided the agents move in certain ways, influenced by each other. This is the only criterion to be followed.

Evidently, there are several possibilities to meet that requirement. In the present implementation we imposed that at least some of the agents should be allowed to change their direction of movement from time to time. The actual details of how they should change is partly expressed in Tables D.3, D.4 and D.5. Additionally, Figure D.1 depicts the way the heads of the agents are modified as a result of the environmental interactions. One can note in that figure that even when an agent is moving over a free background its direction of movement is allowed to change.

State Transitions Supporting the Hardware	Instantiated Role
$\begin{array}{\|c\|c\|c\|}\hline \# & \# & \# \\ \hline \# & \mathbf{E}_{i} & \mathbf{E}_{ii} \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow \mathbf{E}_{ii}$ $\begin{array}{\|c\|c\|c\|}\hline \# & \# & \# \\ \hline \mathbf{E}_{i} & \mathbf{E}_{ii} & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow \mathrm{E}_{LR}$ $\begin{array}{\|c\|c\|c\|}\hline \# & \# & \# \\ \hline \mathbf{E}_{ii} & \mathrm{E}_{L}/\mathrm{E}_{\overline{LR}} & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow \mathrm{E}$ $\begin{array}{\|c\|c\|c\|}\hline \# & \# & \# \\ \hline \mathbf{E}_{ii} & \mathrm{E}_{R} & \# \\ \hline K^{*} & \# & \# \\ \hline \end{array} \Rightarrow \mathrm{E}$ $\begin{array}{\|c\|c\|c\|}\hline \# & \# & \# \\ \hline \mathbf{E}_{ii} & \mathrm{E}_{R} & \# \\ \hline \neq M & M & T \\ \hline \end{array} \Rightarrow \mathrm{E}$	Environmental Dynamics
$\begin{array}{\|c\|c\|c\|}\hline \# & \mathbf{E}_{ii} & \mathrm{E}_{L} \\ \hline \# & G & M \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ $\begin{array}{\|c\|c\|c\|}\hline \mathbf{E}_{ii} & \mathrm{E}_{R} & \# \\ \hline \# & M & T \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$	Instantiated Selection

Table D.1: State transitions supporting the actions of the Turing machine that do not depend on the state transition table of the function being computed.

$$\begin{array}{|c|c|c|}\hline M & \mathrm{E} & \# \\ \hline M & T_{D}/T_{L} & P_{r} \neq P^{0} \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow T_{L\,min}/T_{D\,min}\,|_{\{P_{r},T_{c}\}\mapsto\{T_{L\,min}/T_{D\,min}\}}$$

$$\begin{array}{|c|c|c|}\hline M & \mathrm{E} & \# \\ \hline M & T_{D}/T_{L} & \mathrm{E} \\ \hline \# & \mathrm{E} & P_{br} \neq P^{0} \\ \hline \end{array} \Rightarrow T_{L\,min}/T_{D\,min}\,|_{\{P_{br},T_{c}\}\mapsto\{T_{L\,min}/T_{D\,min}\}}$$

Figure D.1: State transitions specifying the way the heads of the agents are modified as a result of the environmental interactions. The states the transitions lead to are made explicit in Table D.5.

State Transitions Supporting the Software	Instantiated Role
$\begin{array}{\|c\|c\|c\|}\hline \# & \# & \# \\ \hline \mathbf{E}_i & \mathrm{E} & \# \\ \hline M & P_i & G \\ \hline \end{array} \Rightarrow \mathbf{E}_{ii}$	Environmental Dynamics 1
$\begin{array}{\|c\|c\|c\|}\hline \# & \mathbf{E}_i & \mathbf{E}_{ii} \\ \hline \# & P_i & M \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow P_{ii}$	Phenotypic Development
$\begin{array}{\|c\|c\|c\|}\hline \# & \# & \# \\ \hline \mathbf{E}_{ii} & \mathrm{E}_{LR} & \# \\ \hline M & G/K & K^* \\ \hline \end{array} \Rightarrow \mathrm{E}_L/\mathrm{E}_R/\mathrm{E}_{\overline{LR}}\,\vert_{(\mathbf{E}_{ii},K^*)}$	Environmental Dynamics 2
$\begin{array}{\|c\|c\|c\|}\hline \# & \mathbf{E}_{ii} & \mathrm{E}_L \\ \hline \# & K_j & M \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow K^*_j$ $\quad$ $\begin{array}{\|c\|c\|c\|}\hline \# & \mathbf{E}_{ii} & \mathrm{E}_R \\ \hline \# & K^*_i & M \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow K_{ii}$	Memetic Development
$\begin{array}{\|c\|c\|c\|}\hline \mathbf{E}_{ii} & \mathrm{E}_L/\mathrm{E}_{\overline{LR}} & \# \\ \hline \neq M & M & K^*_i \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow K_{ii}$ $\quad$ $\begin{array}{\|c\|c\|c\|}\hline \mathbf{E}_{ii} & \mathrm{E}_R & \# \\ \hline \neq M & M & K_k \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow K^*_k$ $\quad$ $\begin{array}{\|c\|c\|c\|}\hline \# & \# & \# \\ \hline \neq K^* & M & B_r \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow B_r$	

Table D.2: State transitions supporting the actions of the Turing machine that are defined by the state transition table of the function being computed.

G_i	$P^0 \mapsto P_i$
G_0	P_0
G_1	P_1
G_2	P_2
G_3	P_3
G_4	P_f

Table D.3: Neonate development from P_0 to P_i.

P_i	$T^0 \mapsto T_i$
P_0	T_L
P_1	T_L
P_2	T_L
P_3	T_D
P_f	T_D

Table D.4: Neonate development from T_0 to T_i.

P_{ii}	$T_i \mapsto T_{ii}$	
	T_L	T_D
P_0	T_L	T_L
P_1	T_D	T_D
P_2	T_D	T_L
P_3	T_D	T_L
P_f	T_L	T_D

Table D.5: Development of the heads of the agents according to their P_{ii}- and T_i-states. The head is inactive for P_f, and active otherwise.

BIBLIOGRAPHY

[Adami 1993] C. Adami. Self-organized criticality in living systems. W. K. Kellog Laboratory, California Institute of Technology, December 1993. Preprint.

[Adami 1994] C. Adami. Learning and complexity in genetic auto-adaptive systems. W. K. Kellog Laboratory, California Institute of Technology, December 1994. Preprint. To be presented at Artificial Life IV, MIT, Boston, USA, July 1994.

[Agre and Rosenchein 1993] Phil Agre and Paul Rosenchein, editors. *Computational Theories of Interaction and Agency*, Special issue of the journal Artificial Intelligence, 1993.

[Bak *et al.* 1988] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality. *Phys. Rev.*, A 38:364–374, 1988.

[Bak *et al.* 1991] P. Bak, , and K. Chen. Self-organized criticality. *Scientific American*, pages 26–33, January 1991.

[Banks 1971] E. R. Banks. *Information processing and transmission in cellular automata.* PhD thesis, MIT, 1971.

[Bateson 1985] Patrick Bateson. Problems and possibilities of fusing developmental and evolutionary thought. In G. Butterworth, J. Rutkowska, and M. Scaife, editors, *Evolution and developmental psychology*, pages 3–21. Harvester, Brighton, E. Sussex, UK.

[Beer 1992] Randall D. Beer. A dynamical systems perspective on autonomous agents. Technical Report CES-92-11, Dept. of Computer Engineering, Case Western Reserve University, Cleveland, USA, 1992. Submitted to the Special Issue of the AI Journal on Computational Theories of Interaction and Agency.

[Belew 1991] Richard Belew. Artificial life: A constructive lower bound for artificial intelligence. *IEEE Expert*, pages 8–15, February 1991.

[Berlekamp *et al.* 1982] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for your Mathematical Plays*, volume 2. Academic Press, New York, 1982.

[Binder 1993] Philippe M. Binder. Parametric ordering of complex systems. *Physical Review E*, 1993.

[Bourgine and Varela 1992] P. Bourgine and F. J. Varela. Introduction: Towards a practice of autonomous systems. In Francisco J. Varela and Paul Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages xi–xvii, Cambridge, USA, 1992. MIT Press/Bradford Books.

[Brooks 1991a] R. A. Brooks. Intelligence without reason. MIT, Boston, USA, 1991. Preprint. To appear in the Proceedings of IJCAI-91: Joint International Conference on Artificial Intelligence.

[Brooks 1991b] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[Byl 1989] John Byl. Self-reproduction in small cellular automata. *Physica D*, 34:295–299, 1989.

[Chaitin 1987] Gregory J. Chaitin. *Algorithmic Information Theory.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1987.

[Cliff *et al.* 1993] D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behavior*, 2(1):71–104, 1993.

[Cliff 1991] David T. Cliff. Computational neuroethology: A provisional manifesto. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats: Proceedings of The First International Conference on Simulation of Adaptive Behavior*, pages 29–39, Cambridge, MA, 1991. MIT Press/Bradford Books.

[Cliff 1994] Dave Cliff, editor. *AI and Artificial Life*, Special issue of AISB Quarterly: Newsletter of the Society for the Study of Artificial Intelligence and Simulation of Behaviour, number 87, Spring 1994.

[Codd 1968] E.F. Codd. *Cellular Automata.* Academic Press, New York, 1968.

[Crutchfield and Mitchell 1994] James P. Crutchfield and Melanie Mitchell. The evolution of emergent computation. 94-03-012, Santa Fe Institute, Santa Fe, NM, USA, 1994. Submitted to Science.

[Crutchfield 1991] James P. Crutchfield. Knowledge and meaning... chaos and complexity. Working Paper 91-09-035, Santa Fe Institute, Santa Fe, NM, USA, September 1991.

[Crutchfield 1992] James Crutchfield. Semantics and thermodynamics. Physics Department, University of California, Berkeley, CA, 1992. Preprint.

[CSC 1991] CSC, editor. *Workshop on Cellular Automata Proceedings.* Centre for Scientific Computing, Espoo, Finland, 1991.

[Das *et al.* 1994] R. Das, M. Mitchell, and J. P. Crutchfield. A genetic algorithm discovers particle computation in cellular automata. In Y. Davidor; H.-P. Schwefel and R. Maenner, editors, *Parallel Problem Solving from Nature, 3.* Springer-Verlag, Oct. 1994. To appear.

[Davidge 1994] Robert Davidge. *Computer Processors which Behave like Unicellular Organisms: A Thesis in Artificial Life.* To appear as a Cognitive Science Research Report, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1994.

[Davidor 1990] Yuval Davidor. Epistasis variance: A viewpoint on representations, GA hardness, and deception. *Complex Systems*, 4(4), 1990.

[Dawkins 1976] Richard Dawkins. *The Selfish Gene.* Oxford University Press, Oxford, 1976.

[De Jong and Spears 1993] Kenneth A. De Jong and W. Spears. On the state of evolutionary computation. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 618–623, San Mateo, CA, USA, 1993. Morgan Kaufmann.

[De Jong 1985] Kenneth A. De Jong. Genetic algorithms: a 10 years perspective. In J. Grefenstett, editor, *Genetic algorithms and their application: Proccedings of an International Conference on Genetic Algorithms*, pages 169–177. Carnegie-Mellon University, 1985.

[de Oliveira 1989] Pedro P. B. de Oliveira. Towards the specification of a freely evolving system. Unpublished manuscript. Presented at the *International Conference on Evolving Knowledge in Natural Science and Artificial Intelligence*, University of Reading, England, Sept. 1989.

[de Oliveira 1990a] Pedro P. B. de Oliveira. Parallel generation of combinations. Computer Science Report 4, School of Cognitive and Computing Sciences, University of Sussex, England, July 1990.

[de Oliveira 1990b] Pedro P. B. de Oliveira. The second workshop on artificial life. *AISB Quarterly: Newsletter of the Society for the Study of Artificial Intelligence and Simulation of Behaviour*, pages 30–32, number 72, Spring 1990.

[de Oliveira 1992a] Pedro P. B. de Oliveira. A cellular automaton to embed genetic search. In L. Nadel and D. L. Stein, editors, *1991 Lectures in Complex Systems*, Santa Fe Institute Studies in the Sciences of Complexity, Lectures Vol. IV, pages 389–408. Addison-Wesley, 1992.
Also as: Cognitive Science Research Report CSRP-210, School of Cognitive and Computing Sciences, University of Sussex, England, Dec. 1991.

[de Oliveira 1992b] Pedro P. B. de Oliveira. Enact: An artificial-life world in a family of cellular automata. Cognitive Science Research Report CSRP-248, School of Cognitive and Computing Sciences, University of Sussex, England, Sept. 1992. Presented at the *3rd Workshop on Artificial Life*, Santa Fe, NM, USA, June 1992.

[de Oliveira 1992c] Pedro P. B. de Oliveira. Methodological issues within a framework to support a class of artificial-life worlds in cellular automata. Cognitive Science Research Report CSRP-237, School of Cognitive and Computing Sciences, University of Sussex, England, May 1992. Presented at the *British Computing Society Workshop on Cellular Automata*, Imperial College, London, England, Feb. 1992.

[de Oliveira 1993] Pedro P. B. de Oliveira. Methodological issues within a framework to support a class of artificial-life worlds in cellular automata. In D. G. Green and T. Bossomaier, editors, *Complex Systems: From Biology to Computation*, pages 82–96, Amsterdam, 1993. IOS Press. (Abridged version of [de Oliveira 1992c]).

[de Oliveira 1994a] Pedro P. B. de Oliveira. Cellular automata for an approach to emergent functionality. In Salvatore Di Gregorio and Giandomenico Spezzano, editors, *Proceedings of ACRI'94: Cellular Automata in Research and Industry*, pages 99–111. CRAI (Consorzio per la Ricerca e le Applicazioni di Informatica), S. Stefano di Rende, CS, Italy, Sept. 1994.

[de Oliveira 1994b] Pedro P. B. de Oliveira. Coupling computations through space. In W. Porod and G. Frazier, editors, *Proceedings of the 3rd Workshop on Physics and Computation*. IEEE Press, Los Alamitos, CA, USA, Nov. 1994.

[de Oliveira 1994c] Pedro P. B. de Oliveira. Simulation of exaptive behaviour. In Y. Davidor; H.-P. Schwefel and R. Maenner, editors, *Parallel Problem Solving from Nature, 3*, Lecture Notes in Computer Science 866, pages 354–364. Berlin, Germany, Springer-Verlag, Oct. 1994.

[de Oliveira 1995] Pedro P. B. de Oliveira. Collapsing a coevolutionary process into a computable function. *BioSystems: Journal of Biological and Information Processing Sciences*, 1995. To appear.

[Dennett 1991] Daniel C. Dennett. The evolution of consciousness. Chapter 7 from "Consciousness Explained", Allen Lane: Penguin, 1991.

[Dennett 1992] Daniel C. Dennett. Book review: "The embodied mind: Cognitive science and human experience". *New Scientist*, pages 48–49, 1992.

[Draper 1987] S. W. Draper. Machine learning and cognitive development. In J. Rutkowska and C. Crook, editors, *Computers, cognition and development*, pages 255–279. John Wiley and Sons, Chichester, W. Sussex, UK.

[Eckart 1994] J. Dana Eckart. *A Cellular Automata Simulation System.* Computer Science Department, Radford University, Radford, VA, 1994.

[Farmer and d'A Belin 1992] J. Doyne Farmer and Alleta d'A Belin. Artificial life: The coming evolution. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life: Proceedings of the second workshop on Artificial Life*, pages 815–840. Addison-Wesley, 1992.

[Farmer *et al.* 1984] D. Farmer, T. Toffoli, and S. Wolfram, editors. *Cellular Automata: Proceedings of an Interdisciplinary Workshop, Los Alamos, NM, 1983.* Special issue of Physica D 10(1-2), 1984.

[Farmer *et al.* 1986] J.D. Farmer, A. Lapedes, N. Packard, and B. Wendroff, editors. *Evolution, Games and Learning: Models for Adaptation in Machines and Nature*, Proceedings of the 5th Annual International Conference of the Center for Nonlinear Studies, Los Alamos, NM, 1985, 1986. Special issue of Physica D, 22(1-3), 1986.

[Fogel and Atmar 1992] D. B. Fogel and W. Atmar, editors. *First Annual Conference on Evolutionary Programming*, La Jolla, California, 1992. Evolutionary Programming Society.

[Fogel 1992] D. B. Fogel. A brief history of simulated evolution. In David B. Fogel and Wirt Atmar, editors, *Proceedings of the First Annual Conference on Evolutionary Programming*, pages 1–16, La Jolla, California, 1992.

[Fontana 1990] Walter Fontana. Functional self-organization in complex systems. CNLS Newsletter, Center for Nonlinear Studies, Los Alamos National Lab., (60):1–23, November 1990.

[Fontana 1992] Walter Fontana. Algorithmic chemistry. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life: Proceedings of the second workshop on Artificial Life*, pages 159–209. Addison-Wesley, 1992.

[Forrest and Miller 1990] S. Forrest and J. H. Miller. Emergent behavior in classifier systems. *Physica D*, 42:213–227, 1990.

[Forrest 1990] S. Forrest, editor. *Emergent Computation: Self-organizing, Collective and Cooperative Phenomena in Natural and Artificial Computing Networks*, Proceedings of the 9th Annual International Conference of the Center for Nonlinear Studies, Los Alamos, N.M. Special issue of Physica D 42, June 1990.

[Goldberg 1989] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA, 1989.

[Gould and Lewontin 1984] S. J. Gould and R. C. Lewontin. The spandrels of san marco and the panglossian paradigm: A critique of the adaptationist programme. In E. Sober, editor, *Conceptual Issues in Evolutionary Biology: An Anthology*, pages 252–270. MIT Press: Bradford Books, Cambridge, MA.

[Gould and Vrba 1982] S. J. Gould and E. S. Vrba. Exaptation — a missing term in the science of form. *Palaeobiology*, 8(1):4–15, 1982.

[Grefenstette 1985] J. J. Grefenstette, editor. *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Hillsdale NJ, 1985. Lawrence Erlbaum Associates.

[Gutowitz 1991] Howard Gutowitz. *Cellular Automata: Theory and Experiment.* MIT Press/Bradford Books, Cambridge Mass., 1991. ISBN 0-262-57086-6.

[Gutowitz 1994] Howard Gutowitz. Frequently asked questions about cellular automata: Contributions from the community. Available via anoynmous ftp from `think.com`, 1994.

[Hanson and Crutchfield 1991] James E. Hanson and James P. Crutchfield. The attractor-basin portrait of a cellular automaton. Working Paper 91-02-012, Santa Fe Institute, Santa FE, NM, USA, February 1991.

[Harley 1981] C. B. Harley. Learning the evolutionary stable strategy. *Journal of Theoretical Biology*, 89:611–633, 1981.

[Harvey 1991] Inman Harvey. The artificial evolution of behaviour. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats: Proceedings of The First International Conference on Simulation of Adaptive Behavior*, pages 400–408. MIT Press/Bradford Books, Cambridge, MA, 1991.

[Harvey 1994] I. Harvey. *The Artificial Evolution of Adaptive Behaviour.* To appear as a Cognitive Science Research Report, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1994.

[Hillis 1992] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization parameter. In J.D. Farmer, C.G. Langton, S. Rasmussen, and C. Taylor, editors, *Artificial Life II*, pages 311–324. Addison-Wesley, 1992.

[Hinton and Nowlan 1987] G. E. Hinton and S. J. Nowlan. How learning can guide evolution. *Complex Systems*, 1((3)):495–502, June 1987.

[Hoffmeister and Bäck 1991] F. Hoffmeister and T. Bäck. Genetic algorithms and evolution strategies: similarities and differences. In Hans-Paul Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN 1*, volume 496 of *Lecture Notes in Computer Science*, pages 455–469, Dortmund, Germany, 1-3 Oct 1991. Springer-Verlag, Berlin, Germany.

[Holland 1986] John H. Holland. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalsky, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning: an artificial intelligence approach, Volume II*, pages 593–623. Morgan Kaufmann, Palo Alto, CA.

[Hopcroft and Ullman 1979] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Menlo Park, 1979.

[Hornby 1989] A. S. Hornby. *Oxford Advanced Learner's Dictionary of Current English.* Oxford University Press, Oxford, UK, 4th. edition, 1989. Chief Editor: A. P. Cowie.

[Huberman 1994] Bernardo Huberman. Call for papers: Special issue of the journal Artificial Intelligence on "Phase Transitions in Problem Spaces". *Artificial Intelligence*, 68: 201–202, 1994.

[Husbands 1993] P. Husbands. An ecosystems model for integrated production planning. *International Journal of Computer Integrated Manufacturing*, 6(1-2):74–86, 1993.

[Jong 1994] Kenneth De Jong, editor. *Evolutionary Computation*, Cambridge, Mass, 1994. MIT Press.

[Kanebo and Suzuki 1993] Kunihiko Kanebo and Junji Suzuki. Evolution to the edge of chaos in imitation game. Department of Pure and Applied Sciences, College of Arts and Sciences, University of Tokyo, Tokyo, 1993. Preprint.

[Kauffman and Johnson 1992] Stuart A. Kauffman and Sonke Johnson. Coevolution to the edge of chaos: Coupled fitness landscapes, poised states, and coevolutionary avalanches. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life: Proceedings of the second workshop on Artificial Life*, pages 325–369. Addison-Wesley, 1992.

[Kauffman 1991] Stuart A. Kauffman. The sciences of complexity and "origins of order". Working Paper 91-04-021, Santa Fe Institute, Santa Fe, NM, USA, April 1991.

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

[Koza 1990] John R. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. STAN-CS-90-1314, Stanford University, 1990.

[Koza 1992] John R. Koza. Genetic evolution and co-evolution of computer programs. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life: Proceedings of the second workshop on Artificial Life*, pages 603–629. Addison-Wesley, 1992.

[Langton and Hiebeler 1990] Christopher G. Langton and David Hiebeler. Cellsim version 2.5, 1990. Public domain software available via anonymous ftp from think.com or santafe.edu.

[Langton *et al.* 1992] C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors. *Artificial Life II*, volume XI of *Santa Fe Institute Studies in the Sciences of Complexity.* Addison Wesley, 1992. Proceedings of the Workshop on Artificial Life, held in Santa Fe, NM, USA, Feb. 1990.

[Langton 1984] Christopher G. Langton. Self-reproduction in cellular automata. *Physica D*, 10:134–144, 1984.

[Langton 1986] Christopher G. Langton. Studying artificial life with cellular automata. *Physica D*, 22:120–149, 1986.

[Langton 1989] C. G. Langton, editor. *Artificial Life: Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, Santa Fe Institute Studies in the Sciences of Complexity, Proceedings Volume VI. Addison-Wesley, 1989. Workshop held in Los Alamos, NM, USA, Sept. 1987.

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

[Langton 1992a] Christopher G. Langton. Artificial life. In L. Nadel and D. L. Stein, editors, *1991 Lectures in Complex Systems*, Santa Fe Institute Studies in the Sciences of Complexity, Lectures Vol. IV, pages 189–241. Addison-Wesley, 1992.

[Langton 1992b] Christopher G. Langton. Life at the edge of chaos. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life: Proceedings of the second workshop on Artificial Life*, pages 41–91. Addison-Wesley, 1992.

[Langton 1993] C. G. Langton. The SWARM project. *The Bulletin of the Santa Fe Institute*, 8, Spring-Summer(1):?–?, 1993.

[Langton 1994] Christopher G. Langton, editor. *Artificial Life*, Cambridge, Mass, 1994. MIT Press.

[Lenat 1983] D. B. Lenat. The role of heuristics in learning by discovery: three case studies. In R. S. Michalsky, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning: An artificial intelligence approach*, pages 243–306. Tioga, Palo Alto, CA.

[Levy 1992] Steven Levy. *Artificial Life: The Quest for a New Creation.* Pantheon, New York, USA, 1992.

[Lewontin 1983] R. C. Lewontin. The organism as the subject and object of evolution. *Scientia*, 118:63–82, 1983.

[Lewontin 1984] R. C. Lewontin. Adaptation. In E. Sober, editor, *Conceptual Issues in Theoretical Biology: An Anthology*, pages 234–251. MIT Press: Bradford Books, Cambridge, MA, USA.

[Lewontin 1989] R. C. Lewontin. A natural selection. *Nature*, 339:107, 1989.

[Li and Nordahl 1992] Wentian Li and Mats Nordahl. Transient behavior of cellular automata rule 110. *Physics Letters A*, 166(5-6):335–339, 1992.

[Li and Packard 1989] Wentian Li and Norman Packard. The structure of the elementary cellular automata rule space. Technical Report CCSR-89-8, Center for Complex Systems Research, Department of Physics, Beckman Institute, University of Illinois at Urbana-Champaign, September 1989.

[Li *et al.* 1990] Wentian Li, Norman H. Packard, and Christopher G. Langton. Transition phenomena in cellular automata rule space. *Physica D*, 45(1-3):77–94, 1990.

[Li 1989] Wentian Li. *Problems in Complex Systems.* PhD thesis, Center for Complex Systems Research, Department of Physics, Beckman Institute, University of Illinois at Urbana-Champaign, 1989.

[Li 1991] Wentian Li. Parameterizations of cellular automata rule space. Santa Fe Institute, Santa Fe, NM, August 1991. Preprint.

[Li 1992] Wentian Li. Phenomenology of non-local cellular automata. *Journal of Statistical Physics*, 68(5-6), 1992.

[Lindgren and Nordahl 1990] C. Lindgren and M. Nordahl. Universal computation in simple one dimensional cellular automata. *Complex Systems*, 4:299–318, 1990.

[Lipsitch 1991] Marc Lipsitch. Adaptation on rugged landscapes generated by iterated local interactions of neighboring genes. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[Martin 1990] O. Martin. Critical dynamics of 1-D irreversible systems. *Physica D*, 45:345, 1990.

[Maturana and Varela 1987] Humberto R. Maturana and Francisco J. Varela. *The Tree of Knowledge: The Biological Roots of Human Understanding.* Shambhala Press, Boston, 1987.

[McCaskil 1989] John S. McCaskil. Polymer chemistry on tape: A computational model for emergent dynamics. Max-Planck Institut für Biophysikaliche Chemie, Göttingen, Germany, 1989. Preprint.

[McIntosh 1990a] Harold V. McIntosh. Linear cellular automata. Internal publication, Universidad Autonoma de Puebla, Puebla, Mexico, May 1990.

[McIntosh 1990b] Harold V. McIntosh. Wolfram's class IV automata and a good life. *Physica D*, 45:105, 1990.

[Meyer and Guillot 1991] Jean-Arcady Meyer and Agnès Guillot. Simulation of adaptive behavior in animats: Review and prospect. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats: Proceedings of The First International Conference on Simulation of Adaptive Behavior*, pages 2–14. MIT Press/Bradford Books, Cambridge, MA, 1991.

[Meyer and Wilson 1991] J-A. Meyer and S. W. Wilson, editors. *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behaviour*, Cambridge, Mass, 1991. MIT Press/Bradford Books.

[Meyer 1993] Jean-Arcady Meyer, editor. *Adaptive Behaviour*, Cambridge, Mass, 1993. MIT Press.

[Mikhailov 1992] A. S. Mikhailov. Artificial life: An engineering perspective. In R. Friedrich and A. Wunderlin, editors, *Evolution of Dynamical Structures in Complex Systems*, Springer Proceedings in Physics. Springer-Verlag. Submitted.

[Minsky 1967] Marvin Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, 1967.

[Mitchell and Forrest 1993] Melanie Mitchell and Stephanie Forrest. Genetic algorithms and artificial life. Working Paper 93-11-072, Santa Fe Institute, Santa Fe Institute, Santa Fe, NM, USA, November 1993.

[Mitchell *et al.* 1993a] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Dynamic computation, and the "edge of chaos": A re-examination. In G. Cowan, D. Pines, and D. Melzner, editors, *Integrative Themes*, Santa Fe Institute, Proceedings Volume 19, Reading, MA, 1993. Addison–Wesley.

[Mitchell *et al.* 1993b] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. Working Paper 93-03-014, Santa Fe Institute, Santa Fe, NM, USA, 1993. Submitted to Complex Systems.

[Mitchell *et al.* 1994] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. Working Paper 93-11-071, Santa Fe Institute, 1994. Submitted to Physica D.

[Morowitz 1994] Harold Morowitz, editor. *Complexity*, New York, NY, 1994. John Wiley & Sons.

[Oyama 1985] S. Oyama. *The Ontogeny of Information.* Cambridge University Press, Cambridge, 1985.

[Packard 1988] Norman H. Packard. Adaptation toward the edge of chaos. Technical Report CCSR-88-5, Center for Complex Systems Research and the Physics Department, University of Illinois at Urbana-Champaign, 1988.

[Packard 1989] Norman Packard. Intrinsic adaptation in a simple model for evolution. In C. G. Langton, editor, *Artificial Life: Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, Santa Fe Institute Studies in the Sciences of Complexity, Proceedings Volume VI, pages 141–155. Addison-Wesley, 1989.

[Pedersen 1990] John Pedersen. Continuous transitions of cellular automata. *Complex Systems*, 4:653–665, June 1990.

[PhysComp-81 1982] PhysComp-81, editor. *Proceedings of the First Conference on the Physics of Computation, MIT, USA, 1981.* International Journal for Theoretical Physics, Vol 21, April, June and December issues, 1982.

[Piatelli-Palmarini 1989] M. Piatelli-Palmarini. Evolution, selection and cognition: From 'learning' to parameter setting in biology and in the study of language. *Cognition*, 31(1), 1989.

[Pinker and Bloom 1990] S. Pinker and P. Bloom. Natural language and natural selection. *Behavioral and Brain Sciences*, 13:707–784, 1990.

[Rasmussen *et al.* 1990] 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.* 1992] Steen Rasmussen, Carsten Knudsen, and Rasmus Feldberg. Dynamics of programmable matter. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life: Proceedings of the second workshop on Artificial Life*, pages 211–254. Addison-Wesley, 1992.

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

[Roska and Vandewall 1993] T. Roska and J. Vandewall. *Cellular Neural Networks*. Wiley, 1993.

[Rota 1986] Gian-Carlo Rota. In memoriam of Stan Ulam: The barrier of meaning. In J.D. Farmer, A. Lapedes, N. Packard, and B. Wendroff, editors, *Evolution, Games and Learning: Models for Adaptation in Machines and Nature*, pages Special issue of Physica D, 22(1–3):4–12, 1986.

[Rucker 1993] Rudy Rucker. *Artificial Life Lab*. Waite Group Press, Corte Madera, CA, USA, 1993.

[Rutkowska 1990] Julie C. Rutkowska. Action, connection and enaction: A developmental perspective. *AI & Society: The Journal of Human and Machine Intelligence*, 1990.

[Sannier II and Goodman 1987] A. V. Sannier II and E. D. Goodman. Genetic learning procedures in distributed environments. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the 2^{nd} International Conference on Genetic Algorithms*, pages 162–169, Hillsdale, NJ, 1987. Lawrence Erlbaum Associates.

[Scaife 1989] M. Scaife. A framework for research in developmental cognitive science. Report prepared for the human behaviour and development group of the esrc, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1989.

[Schroeder 1991] Manfred Schroeder. *Fractals, Chaos, Power Laws*. W. H. Freeman and Company, New York, 1991.

[Schwefel and Männer 1991] H. P. Schwefel and R. Männer, editors. *Parallel Problem Solving from Nature – Proceedings 1st Workshop PPSN 1*, volume 496 of *Lecture Notes in Computer Science*, Berlin, 1991. Springer-Verlag.

[Serra and Zanarini 1990] R. Serra and G. Zanarini. *Complex Systems and Cognitive Processes*. Springer-Verlag, Heidelberg, Germany, 1990.

[Smith III 1971] Alvy Ray Smith III. Simple computation-universal cellular spaces. *Journal of the Association for Computing Machinery*, 18:339–353, 1971.

[Smith 1987] John Maynard Smith. When learning guides evolution. *Nature*, 329:761–762, 1987.

[Stephenson 1992] Ian Stephenson. Creature processing: An alternative cellular architecture. Technical Report ASEG-92.04, Department of Electronics, University of York, UK, 1992.

[Tackett 1992] Walter A. Tackett. Fitness and adaptation of digital organisms. Talk given at Artificial Life III, Santa Fe, NM, USA, 1992.

[Thompson *et al.* 1991] E. Thompson, A. Palacios, and F. Varela. Ways of coloring: Comparative color vision as a case study for cognitive science. *Brain and Behavioral Sciences*, 15:1–74, 1991.

[Toffoli and Margolus 1987] Tommaso Toffoli and Norman Margolus. *Cellular Automata Machines: A New Environment for Modeling*. MIT Press, Cambridge, Mass, 1987.

[van Gelder 1992] Timothy van Gelder. What might cognition be if not computation? Research Report 75, Cognitive Science, Indiana University, 1992.

[Varela and Bourgine 1992] Francisco J. Varela and Paul Bourgine, editors. *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. Series: Complex Adaptive Systems. MIT Press, Cambridge, MA, 1992.

[Varela *et al.* 1991] F. Varela, E. Thompson, and E. Rosch. *The Embodied Mind*. MIT Press, 1991.

[Varela 1989] F. Varela. *Connaître: les sciences cognitives, tendances et perspectives*. Seuil, Paris, 1989.

[Vitanyi 1973] Paul M. B. Vitanyi. Sexually reproducing cellular automata. *Mathematical Biosciences*, 18:23–54, 1973.

[von Neumann 1966] John von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana-Champaign, 1966. Editor: A. W. Burks.

[Vose 1991] M. D. Vose. Formalizing genetic algorithms. Technical Report CS-91-127, University of Tennessee, 1991.

[Weisbuch 1991] Gérard Weisbuch. *Complex Systems Dynamics: An Introduction to Automata Networks*. Lecture Notes Volume II, Santa Fe Institute, Studies in the Sciences of Complexity. Addison Wesley, Redwood City, CA, USA, 1991.

[Werner and Dyer 1992] Gregory M. Werner and Michael G. Dyer. Evolution of communication in artificial organisms. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life: Proceedings of the second workshop on Artificial Life*. Addison-Wesley, 1992.

[Whitley 1993] Darrell Whitley. Cellular genetic algorithms. In Stephanie Forrest, editor, *Proceedings of the 5^{th} International Conference on Genetic Algorithms*, page 658, San Mateo, CA, USA, 1993. Morgan Kaufmann.

[Winograd and Flores 1986] T. Winograd and F. Flores. *Understanding computers and cognition: a new foundation for design*. Ablex, Norwood, NJ, 1986.

[Wolfram 1986a] Stephan Wolfram. Statistical mechanics of cellular automata. In Stephan Wolfram, editor, *Theory and Applications of Cellular Automata*. World Scientific, Singapore. Reprinted from: *Reviews of Modern Physics*, 55:601–644, 1983.

[Wolfram 1986b] Stephan Wolfram. Universality and complexity in cellular automata. In Stephan Wolfram, editor, *Theory and Applications of Cellular Automata*. World Scientific, Singapore.

[Wuensche and Lesser 1992] Andrew Wuensche and Mike Lesser. *The Global Dynamics of Cellular Automata*. Santa Fe Institute Studies in the Sciences of Complexity, Reference Volume 1. Addison-Wesley, 1992.

[Wuensche 1993] Andrew Wuensche. The ghost in the machine: Basins of attraction of random boolean networks. Cognitive Science Research Paper CSRP-281, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1993. To appear in Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity.

[Wuensche 1994] Andrew Wuensche. Complexity in one-D cellular automata: Gliders, basins of attraction and the Z parameter. Cognitive Science Research Papers CSRP 321, School of Cognitive and Computing Science, University of Sussex, Brighton, UK, February 1994.

[Yao 1992] Xin Yao. A review of evolutionary artificial neural networks. Commonwealth Scientific and Industrial Research Organisation, Division of Building, Construction and Engineering, Australia, 1992. Preprint.