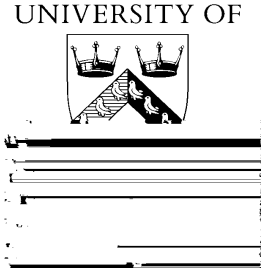
THE UNIVERSITY OF SUSSEX

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 2 , 1995



CON EN S

LIS OF FIGURES

LIS OF ABLES

DECLARA ION

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:

ACKNOWLEDGEMEN S

This thesis became possible thanks to grants (No. 200695/88.6 and No. 451458/94-0)

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

PREFACE

Drivers and Balloonists

Chapter 1

IN RODUC ION: GUIDELINE

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. Contributions of the Thesis

1. .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. .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

Chapter 2

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

([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

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. 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

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.

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

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*

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

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.

Authors	Year	Number of	Size of Initial	Universal	

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. t2.5.mpuf/BPCr.veleme,4(for)-1rmeddirectly

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

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

in fact two very similar groups of four parameters.

CHAPTER 3

ENAC : AR IFICIAL LIFE IN CELLULAR AU OMA A[1]

.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

tl	**t** : *top*	**tr**
l : *left*	**c** : *centre*	**r** : *right*
bl	**b** : *bottom*	**br**

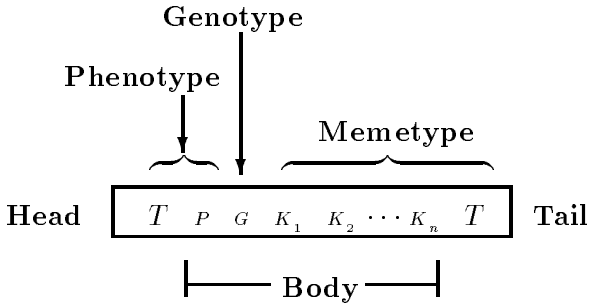


Figure 3.2: Structure and morphology of an agent in Enact. Phenotype and memetype may change through environmental interactions, but while the initial adult state of the former directly depends on the genotype, the latter is initially determined through direct parental inheritance. For practical purposes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

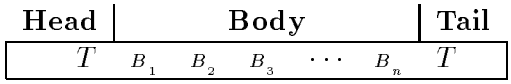


Figure 3.3: Alternative, simplified representation of an agent. The B-states represent the states of the body cells, with no reference to its internal structure.

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

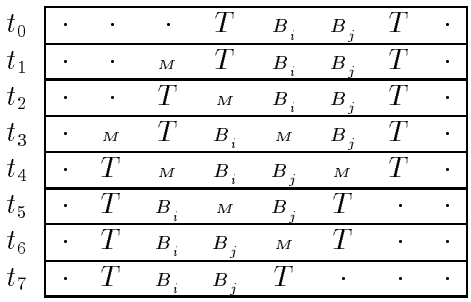


Figure 3.4: Succession of snapshots of the same set of cells as a 4-cell-long agent moves 2 cells leftwards in successive iterations. The dots represent the background state.

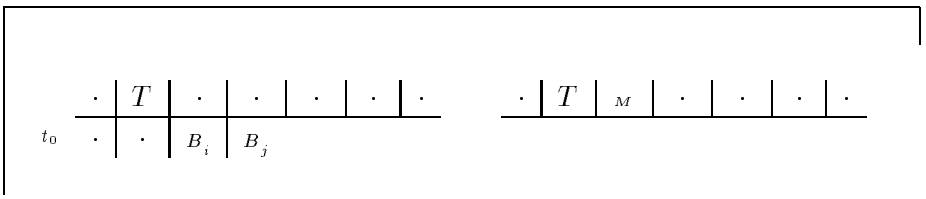
Because of the toroidal geometry of the cellular space, leftward and diagonal movements are sufficient to ensure that the agents have the ability to cover the entire world. In this way the agents are able to approach any other in the world and, when two of them reach a predefined mating configuration, 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 becauseemenbemen

emendl3reand fr


t_0
T
B_i
B_j
T
M

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.

.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

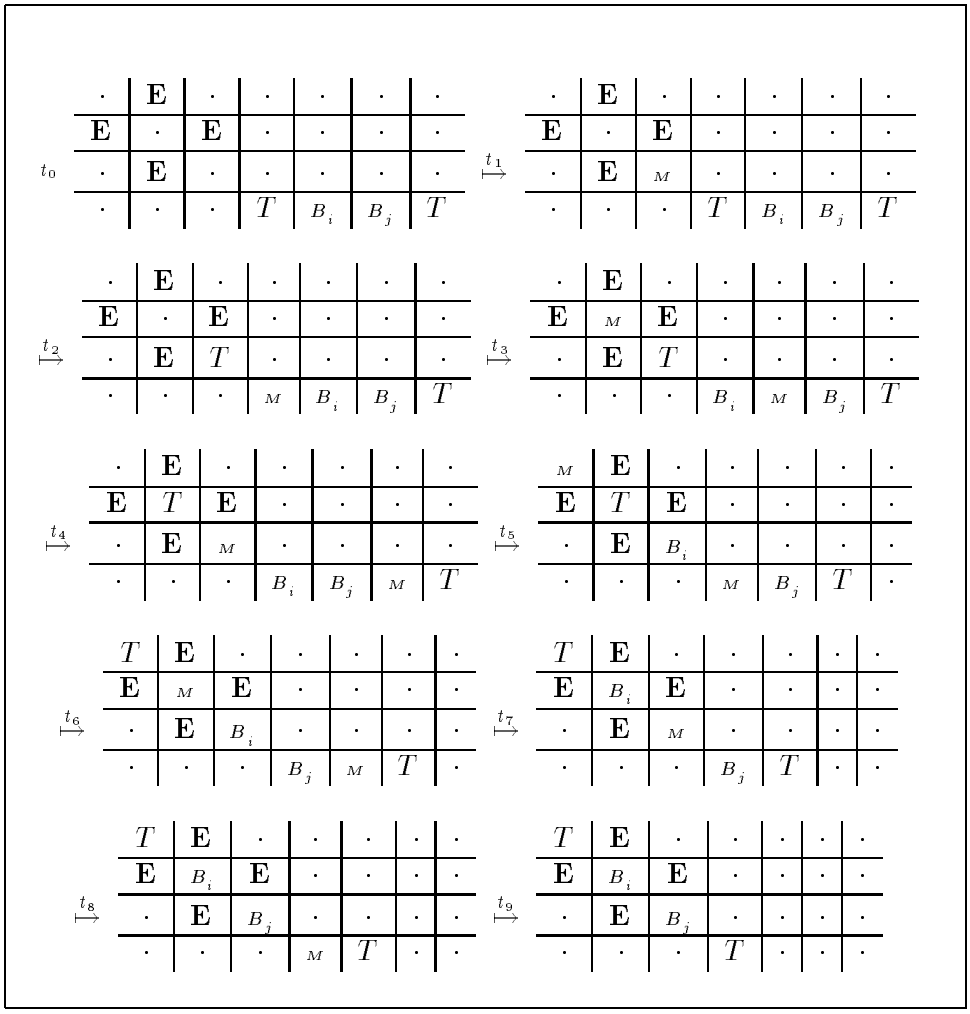


Figure 3.7: Illustration of the notion of interaction site, represented here by the set of four E-states in a cross-like fashion. The dots represent the background state, a special form of environmental state that does not obstruct an agent in its way.

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.

.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

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

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 0Td(in)Tj13.68010Td(the)Tj197445999.7(c[((viablen710Td

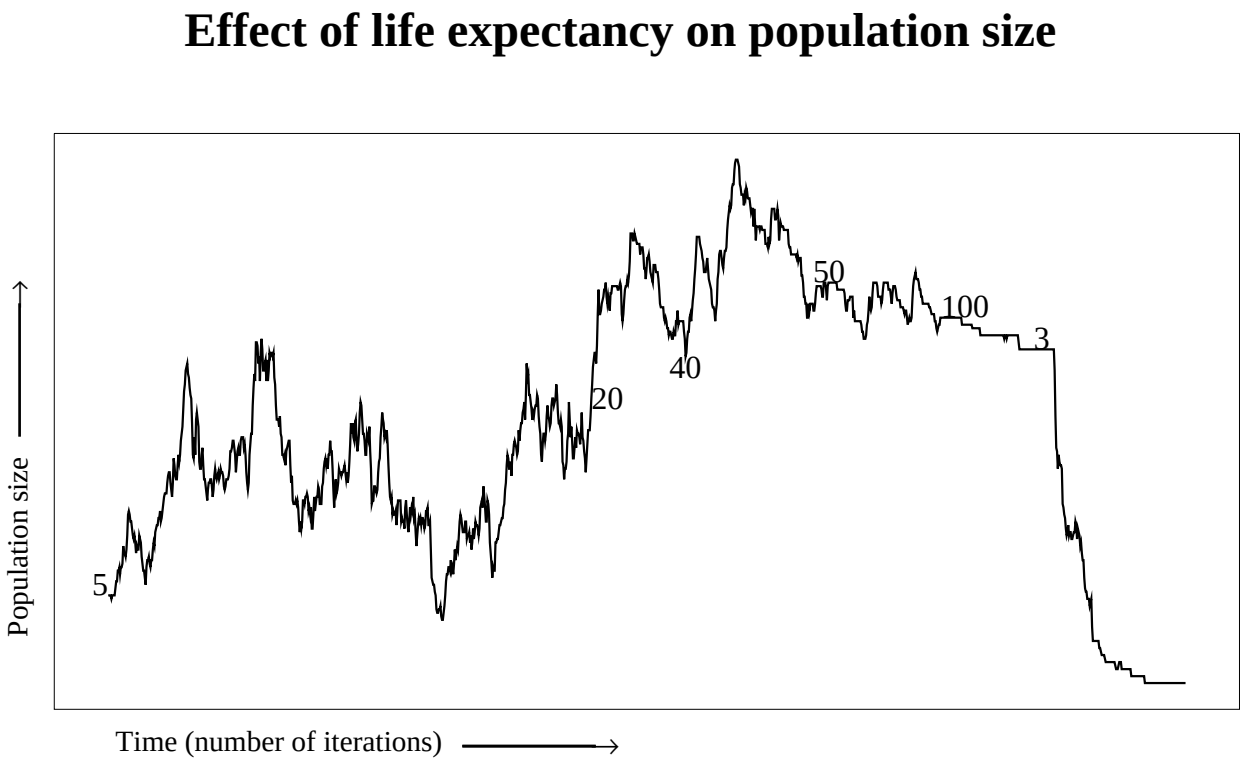


Figure 3.9: Effect of the expected

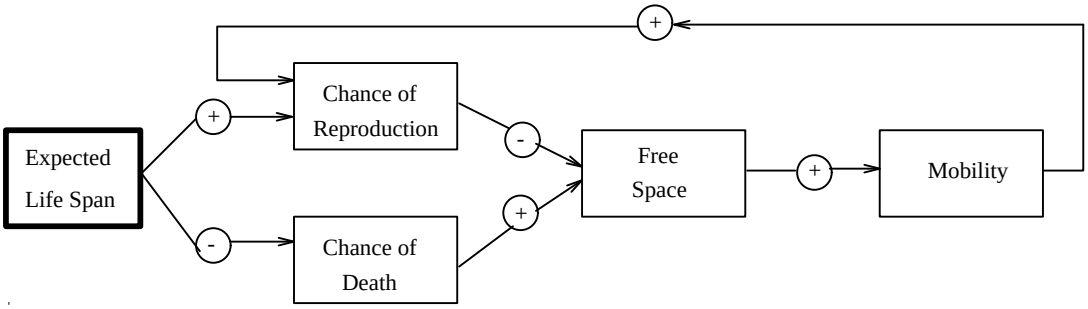


Figure 3.10: Causal links between the parameters that determine Enact's basic dynamics. Only the expected life span is an explicit parameter, effectively controlled.

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.

.9. 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

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*.

.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

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.

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

.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.

.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.

.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.

.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

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

.1 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

ENAC AS A VIR UAL 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

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

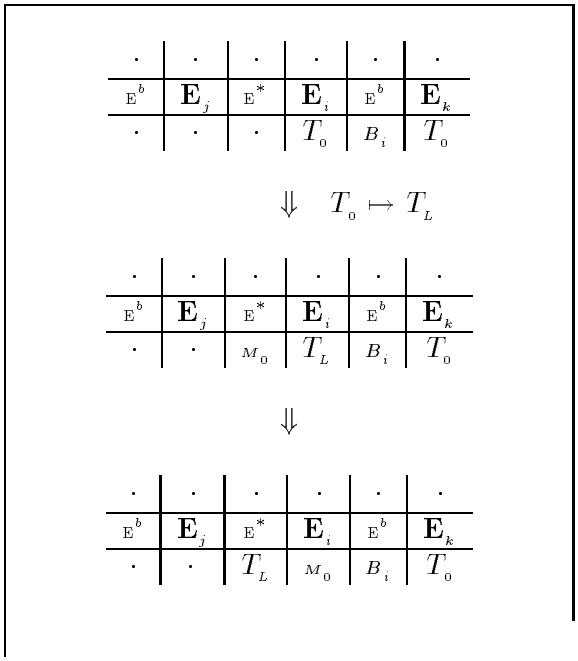
Table 4.1: Correspondence between the constituent elements of a Turing Machine and the states currently being used to implement it.

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 moves snapshots


E^b $\mathbf{E}_j$ E^* $\mathbf{E}_i$ E^b $\mathbf{E}_k$
T_0 B_i T_0
$T_0 \mapsto T_L$
M_0 T_L
T_L M_0

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

4. 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. .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

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

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. .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. How(are)Tj-38atter

- 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

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.

General Transitions	Example	Instantiated Role
$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{E}_i & \# \\ \hline 0 & T_0 & B_i \\ \hline \# & \# & \# \end{array} \Rightarrow T_L$	$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{0}_E & \# \\ \hline 0 & T_0 & B_2 \\ \hline \# & \# & \# \end{array} \Rightarrow T_L$	$\begin{array}{c\|c\|c} E & \# & \# \\ \hline E & T_c^* & M/B \\ \hline \# & \# & \# \end{array} \stackrel{\overline{d}}{\Rightarrow} T_c/(T_c^+/0)$
$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{E}_i & \# \\ \hline T_L & B_i & M_0 \\ \hline \# & \# & \# \end{array} \Rightarrow B_{ii}$	$\begin{array}{c\|c\|c} {}^{E^*} & \mathbf{0}_E & \# \\ \hline T_L & B_2 & M_0 \\ \hline \# & \# & \# \end{array} \Rightarrow B_2$	Adult Body Development
$\begin{array}{c\|c\|c} \# & \# & \# \\ \hline {}^{E^*} & \mathbf{E}_i & \# \\ \hline \multicolumn{3}{l}{T_L} \end{array}$		

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. .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 output of the K^f is tape if the pathe state

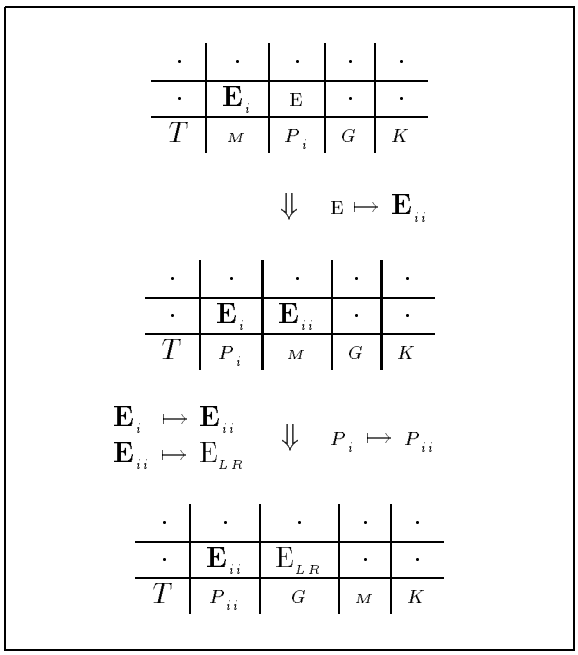


Figure 5.2.a

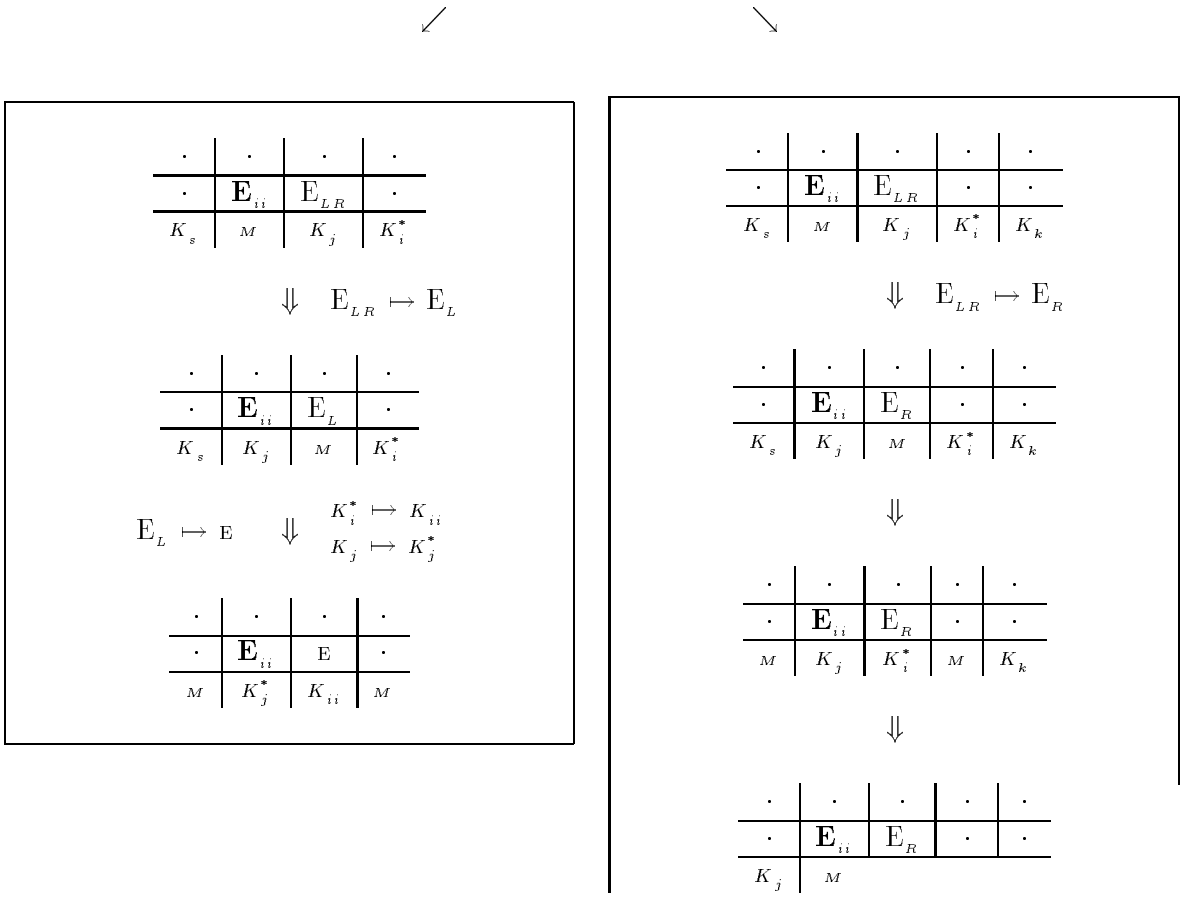
Figure 5.2.b

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]

Figure 5.1


$\mathbf{E}_{ii}$
E_{LR}
K_s
M
K_j
K_i^*
$\Downarrow \quad \mathrm{E}_{LR} \mapsto \mathrm{E}_L$
E_L
$\mathrm{E}_L \mapsto \mathrm{E}$
$K_i^* \mapsto K_{ii}$
$K_j \mapsto K_j^*$
E
K_{ii}
K_k
$\Downarrow \quad \mathrm{E}_{LR} \mapsto \mathrm{E}_R$
E_R

implemented mechanisms.

$\mathbf{E}_i$	$P_i \mapsto P_{ii}$				
	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			

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_{11} is represented as depending on $\mathbf{E}_1$ and P_1, following the details of Table 5.4.

5. .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.

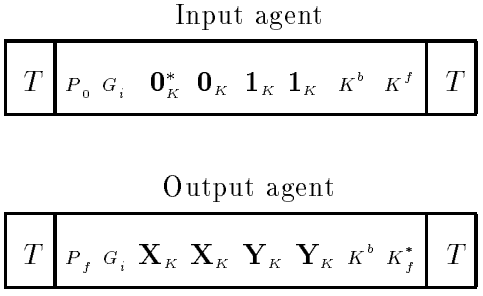


Figure 5.5: State configuration of agents that represent a correct input and the corresponding output of the computation. The input agent represents a string that is recognised as belonging to the language Λ.

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.

in order

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.

#	K/M	$\neq$ E
G	0	E
$\neq$ E	K/M	#

$\Rightarrow \mathbf{0}^*_K$

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

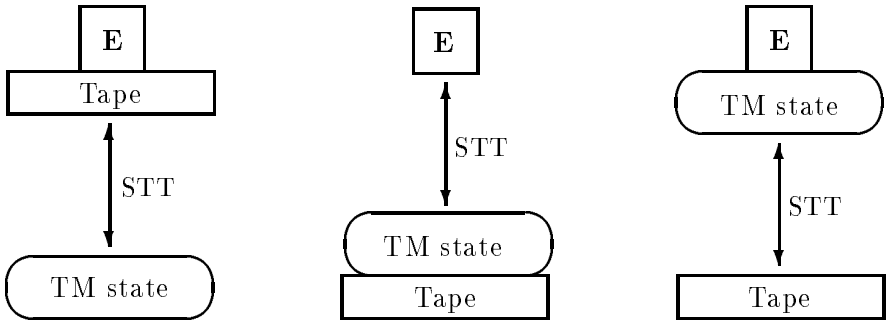
It was shown how the entire dynamics of a class of evolutionary systems can be used to perform a computation. The argument was constructive by presenting a Turing-machine-based set-up implemented in Enact. As a byproduct, the chapter also served to characterise the main model of computation underlying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

are shared among the processing agents.

6. Coupling Turing Machines through



$\mathcal{SoA}$ 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

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).

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

Instantiated State Transitions

[1]

$K_i \mid_{K_i > K_j}$	$\mathbf{E}^+$	#
$\mathbf{E}$	K_j	#

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

> "...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

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

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

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

he Complete list of State ransitions in Enact

A.1

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. 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] $\begin{array}{|c|c|c|}\hline \mathrm{E} & B & \# \\ \hline \mathrm{E} & B & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$

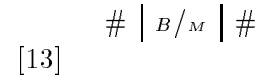
[16] $\begin{array}{|c|c|c|}\hline M & T & \# \\ \hline \mathrm{E} & B/M & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [17] $\begin{array}{|c|c|c|}\hline \mathrm{E} & M & B/T \\ \hline \mathrm{E} & B & \# \\ \hline \# & \# & \# \\ \hline \end{array} \Rightarrow 0$ [18] $\begin{array}{|c|c|c|}\hline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

[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] $\begin{array}{|c|c|c|}\hline M & \mathrm{E} & \# \\ \hline \mathrm{E} & M & \# \\ \hline \# & & \\ \hline \end{array}$



[13]

Appendix B

he 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(C)Tj17.03980Td1IOrin
```

```
#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(
```

```
    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(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,
```

```
}

/*************************************************************************/
/* 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 td990T4J01dsnte=0:
```

```
             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;
```

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

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

$\Rightarrow T_R$

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

Appendix D

Details of the Implementation of the uring 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.

[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 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

[Forrest 1990] S. Forrest, editor. *Emergent Computation:*

[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

[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,

[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.

[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.