Intelligent Systems Reference Library 262

Seiki Akama

Artificial Life

How to Create a Life Computationally

Springer

Intelligent Systems Reference Library

Volume 262

Series Editors

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland

Lakhmi C. Jain, KES International, Shoreham-by-Sea, UK

The aim of this series is to publish a Reference Library, including novel advances and developments in all aspects of Intelligent Systems in an easily accessible and well structured form. The series includes reference works, handbooks, compendia, textbooks, well-structured monographs, dictionaries, and encyclopedias. It contains well integrated knowledge and current information in the field of Intelligent Systems. The series covers the theory, applications, and design methods of Intelligent Systems. Virtually all disciplines such as engineering, computer science, avionics, business, e-commerce, environment, healthcare, physics and life science are included. The list of topics spans all the areas of modern intelligent systems such as: Ambient intelligence, Computational intelligence, Social intelligence, Computational neuroscience, Artificial life, Virtual society, Cognitive systems, DNA and immunity-based systems, e-Learning and teaching, Human-centred computing and Machine ethics, Intelligent control, Intelligent data analysis, Knowledge-based paradigms, Knowledge management, Intelligent agents, Intelligent decision making, Intelligent network security, Interactive entertainment, Learning paradigms, Recommender systems, Robotics and Mechatronics including human-machine teaming, Self-organizing and adaptive systems, Soft computing including Neural systems, Fuzzy systems, Evolutionary computing and the Fusion of these paradigms, Perception and Vision, Web intelligence and Multimedia.

Indexed by SCOPUS, DBLP, zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

Seiki Akama

Artificial Life

How to Create a Life Computationally

Springer

Seiki Akama
Kawasaki, Japan

ISSN 1868-4394 ISSN 1868-4408 (electronic)
Intelligent Systems Reference Library
ISBN 978-3-031-74421-1 ISBN 978-3-031-74422-8 (eBook)
https://doi.org/10.1007/978-3-031-74422-8

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Foreword

This is an introductory book on artificial life written by Dr. Seiki Akama, a researcher in logic, logical philosophy, and artificial intelligence. In particular, he has done a lot of work on non-classical logics including modal, constructive and paraconsistent logics.

Given this background, readers may wonder why a logician would publish an introductory book on artificial life. He has written academic papers and introductory books on information logic, logic-based AI, soft computing, programming, databases, and quantum mechanics.

This extensive knowledge brings new possibilities to the introduction of artificial life. The reader cannot only gain an understanding of the origins of artificial life to its various developments, but also learn from knowledge backed by deep insight that all of the various approaches to elucidating artificial life are deeply connected to knowledge and intelligence.

In the field of artificial life, various models of life have been proposed based on the structure and behavior of biological systems, such as cellular automata, neural networks, and genetic algorithms, to elucidate the essence of life.

Models inspired and designed from a biological perspective contribute to the development of AI systems that enable learning, adaptation, creative generation, and even emergence. The ability to self-regenerate, evolve, and learn are fundamental abilities that underlie the acquisition of intelligence possessed by living things.

These can be applied not only to the basic research areas of artificial intelligence and knowledge engineering, but also to all fields related to prediction and optimization, such as engineering design and optimization, robotics, financial modelling, game strategy, and bioinformatics.

Neural networks are a subject of research in artificial life. Modelling brain cells using neural networks has overcome several obstacles and led to the development of deep learning, which forms the basis of large-scale language models that have made great progress in recent years.

Regarding the question of whether generative AI has emergent properties, the question is whether bidirectional nonlinear feedback in cellular automata of artificial life occurs in the same way as it works in advanced intelligent production systems.

It is interesting that the beginning of von Neumann and Ulam's research on self-reproducing automata almost coincided with the birth of the world's first computer, ENIAC.

This is because at the beginning of research on artificial life and artificial intelligence, there was no barrier between the two, and they are nothing but different expressions of the results of tackling fundamental problems to elucidate the principles of thinking.

Neural networks, which began as brain simulations, attempt to express logical circuits using neural networks. The development of neural networks to deep learning bridges the gap between artificial life and artificial intelligence.

The study of intelligence is the study of learning, and the relationship between learning and evolution is the basis of the formation of the instincts of living organisms.

The fundamental principle of the behavior of living things is that life contains the property of logical structure. It manifests itself as an emergent phenomenon by logically harmonizing rules at the cellular level.

Furthermore, this has the same structure as the logic of intelligence, and its emergent nature forms the basis of the logic of advanced cognitive activities, and by coordinating and influencing each other, it is mapped to different information representations.

There is. Intelligence expressed through emergence is the basis of advanced human cognitive abilities, and this leads to current large-scale language models based on deep learning.

The problem of defining artificial life cannot avoid the fundamental question of what life is. The final chapter of this book deals with this unresolved problem of artificial life. Evolution is an inseparable characteristic of life, and the more complex life becomes, the more difficult it will be to survive and continue to develop without evolving.

It is not hard to imagine that complex life forms acquire intelligence and continue to evolve to make survival and reproduction advantageous. However, it is interesting to consider how humans acquire intelligence.

Suppose the essence of learning ability is closely related to the evolution of living things. In that case, research on artificial life is a fundamental research field of intelligence that is continuous with research on intelligence.

New Trend also covers the relationship between avatars, virtual worlds, and artificial life. This field is currently undergoing significant development, and it is possible to have an experience in the virtual world that is different from the real world. This suggests the need to consider how learning and evolution cause interactions in the digital world from the standpoint of possible world semantics.

With the recent evolution of AI, we should once again recognize the importance of artificial life as life is the root of intelligence. However, the content evolved from the ideas required in early artificial life research and the exploration of emergent intelligence is considered to be a more fundamental problem for evolution and prosperity.

We believe that this book will provide useful suggestions for students, researchers, experts, and engineers who are interested in life and intelligence, regardless of the field, such as AI, robotics, engineering, or philosophy.

Tokyo, Japan
March 2024

Yotaro Nakayama

Preface

Artificial Life (ALife or A-Life) is the research area which simulates a life computationally. It involves several approaches to living systems in artificial manners. It was recognized as a scientific field in the 1980's. Artificial Life is related to many scientific fields including biology and computer science, and its history is old, In fact, the earlier works have been done by people like Turing and von Neumann.

Fields like that Neural Networks, Evolutionary Computation, and Cellular Automaton constitute the foundations for Artificial Life. The so-called Wetware is an approach in more biology-oriented, is known that there are some Artificial Life Systems at present. It is expected to see further progresses of this exciting field.

The purpose of this book is to provide some resources of Artificial Life. The materials include: history, philosophy, related areas, recent developments, etc. Starting from the overview, we give an exposition of basic subjects like Cellular Automaton, Neural Networks, Evolutionary Computation, and WetWare. We also introduce some examples of Artificial Life Systems like Boids and Tierra.

The structure of the book is as follows. Chapter 1 introduces Artificial Life informally. First, we identify its definition by various viewpoints. Second, we overview the history and discuss related areas including Artificial Intelligence, biology and philosophy. This chapter can serve as an introduction to the present book.

Chapter 2 describes cellular automata, which are used to model the self-reproducing function of organisms. Here we introduce types of cellular automata, the Wolfram class, the Game of Life, and Langton's loops.

Chapter 3 describes neural networks, which provide the theoretical basis for machine learning. Neural networks are computational systems based on the biological neural networks. After explaining their basic idea, we introduce perceptrons, back propagation networks, Hopfield networks, Boltzmann machines, and Parallel Distributed Processing Models.

Chapter 4 presents evolutionary computation. It is a field studied in the areas including AI and soft computing based on the ideas of biological evolution. It is mainly applied to solve complex optimization problems. After reviewing the basic ideas of evolutionary computation, we introduce Genetic Algorithm, Genetic Programming, Evolutionary Programming, and Evolutionary Strategy.

Chapter 5 is an exposition of Wetware. By Wetware, we mean Artificial Life by using biochemical methods. Thus, it is considered to be any hardware or software systems with a biological component or biological systems which act like software and hardware. After providing general discussion on Wetware, we say more about DNA computers, artificial bacteria, clones, and artificial cells.

Chapter 6 presents some examples of AL systems. As typical examples, we overview Boids, Tierra, and L-systems.

Chapter 7 shows prospects of AL systems. We discuss several problems with AL. We also consider new trends of AL in connection with cybernetics, complex systems, agents and avatar. Finally, we provide remarks on some open problems.

We assume that the reader has the basics of computer science. We believe that the book is suitable for those, like experts and students, who are interested in Artificial Life.

We are grateful to Prof. L. Jain, Dr. Y. Nakayama and the referees for constructive comments.

Kawasaki, Japan Seiki Akama
February 2024

Contents

1 Introducing Artificial Life 1
1.1 What Is Artificial Life? 1
1.1.1 Basics of Artificial Life 1
1.1.2 Three Positions of Artificial Life 3
1.2 History and Related Areas 5
1.2.1 1940s–1970s 5
1.2.2 1980s 7
1.2.3 1990s 9
1.2.4 2000s 9
1.2.5 Scientificc Theory 10
1.2.6 Related Areas 12
References 14

2 Cellular Automata 17
2.1 Backgrounds of Cellular Automata 17
2.1.1 What Are Cellular Automata? 17
2.1.2 History of Cellular Automata 19
2.1.3 Classification of Cellular Automata 19
2.2 Wolfram Class 20
2.2.1 Classes 1–4 20
2.2.2 Some Rules 20
2.3 Game of Life 22
2.3.1 Is Life a Game? 23
2.3.2 Computability of the Game of Life 25
2.4 Langton's Loop 25
2.4.1 Langton's Original Definition 25
2.4.2 Some Simplifications 28
References 29

3 Neural Networks 31
3.1 Ideas of Neural Networks 31
3.1.1 What Are Neural Networks? 31
3.1.2 History of Neural Networks 32
3.1.3 Advantages and Disadvantages 35
3.2 Perceptrons 35
3.2.1 Basics of Perceptron 36
3.2.2 Perceptron and Machine Learning 38
3.3 Back Propagation Networks 39
3.3.1 Basics of Back Propagation Networks 39
3.3.2 McCulloch-Pitts Neuron 39
3.4 Hopfield Networks 41
3.4.1 Basics of Hopfield Networks 41
3.4.2 Associative Memory 42
3.5 Boltzmann Machines 44
3.5.1 Basics of Boltzmann Machines 45
3.5.2 Advantages of Boltzmann Machines 47
3.6 Parallel Distributed Processing Models 47
3.6.1 Basics of Parallel Distributed Processing Models 47
3.6.2 Simple Parallel Distributed Processing Model 48
3.7 Deep Learning 49
3.7.1 What Is Deep Learning? 49
3.7.2 Advantages and Disadvantages 49
References 51

4 Evolutionary Computation 53
4.1 Ideas of Evolutionary Computation 53
4.1.1 What is Evolutionary Computation? 53
4.1.2 Approaches to Evolutionary Computation 54
4.2 Genetic Algorithm 55
4.2.1 What is Genetic Algorithm? 55
4.2.2 Basics of Genetic Algorithm 56
4.3 Genetic Programming 58
4.3.1 What is Genetic Programming? 58
4.3.2 LISP 58
4.4 Evolutionary Strategy 61
4.4.1 What is Evolutionary Strategy? 61
4.4.2 Applications of Evolutionary Strategy 62
References 63

5 Wetware 65
5.1 From Hardware and Software to Wetware 65
5.1.1 What Is Wetware? 65
5.1.2 Computer Virus 66
5.2 DNA Computers 66
5.2.1 Adleman's Approach 71

5.2.2 Lipton's Approach ... 75
5.2.3 Shapiro's Approach ... 81
5.2.4 Hagiya's Approach ... 84
5.3 Artificial Bacteria ... 90
5.4 Clone ... 92
5.5 Artificial Cells ... 95
References ... 97

6 Artificial Life Systems ... 99
6.1 Boids ... 99
6.2 Tierra ... 103
6.3 L-System ... 106
References ... 111

7 Prospects of Artificial Life ... 113
7.1 Problems with Artificial Life ... 113
7.1.1 Some Important Problems ... 113
7.1.2 Definitions of Artificial Life ... 117
7.2 New Trends ... 119
7.2.1 Cybernetics ... 120
7.2.2 Complex Systems ... 120
7.2.3 Avatars ... 127
7.2.4 Virtual Reality ... 127
7.3 Open Problems ... 128
7.4 New Ideas from Related Areas ... 133
References ... 134

Index ... 135

Chapter 1
Introducing Artificial Life

Abstract This chapter introduces Artificial Life informally. First, we identify its definition by various viewpoints. Second, we overview the history and discuss related areas including Artificial Intelligence, biology and philosophy. This chapter can serve as an introduction to the present book.

1.1 What Is Artificial Life?

1.1.1 Basics of Artificial Life

When you hear the word *Artificial Life*, it would be very interesting for you. This is because it was often to be a fantastic story. However, Artificial Life is now a field of scientific research.

Historically, we were faced with several questions about artificial creatures. They include the following: What are the conditions of life? Is it possible to make living creatures? Such questions were asked in many countries.

For example, artificial creatures can be found in methodologies in some countries. In fact, we know some examples of artificial creatures like unicorn, dragon, fairy, etc.

Questions related to life has been discussed in philosophy in connection with God. For instance, Hobbes had the idea of creating an artificial animal; see Hobbes [12]. Descartes also considered the living to be mechanical; see Descartes [7]. However, he did not consider that the soul is not mechanical. These two philosophers revealed the philosophical concept similar to Artificial Life.

Artificial Life is also recognized as a technology, and is heading for practical use. We can see that Artificial Life has in fact some theoretical basis. It is noticed also that several Artificial Life systems are developed with some applications.

Artificial Life, also abbreviated as AL or ALife, aims to artificially construct a "living system". (Henceforth, Artificial Life may be abbreviated as AL.) It is known that the term "Artificial Life" was coined by Langton; see Langton [18].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Akama, *Artificial Life*, Intelligent Systems Reference Library 262,
https://doi.org/10.1007/978-3-031-74422-8_1

However, the question of what are "artificial" and "life" remain. Before discussing AL, we need to identify these notions used for AL. It is considered to be important but difficult to interpret "artificial". It is also true that a precise definition of "life" is difficult.

What is life? There are many approaches to define the concept of life. One of the interesting approaches is due to Schrödinger [25] in the 1940's[1]. His approach is based on the idea that a living organism can be accounted for by physics and chemistry.

In his book, he gave a long argument in which he sets himself the task of nothing less than the search for unified, all-embracing knowledge to describe the natural phenomenon we refer to as life. This means that his argument is compatible to Artificial Life.

In "Artificial Life", "life" is also interpreted in the sense of "biological phenomenon". Therefore, we can define "artificial life" as a research field of studying the methods that synthesize "living phenomena" from computers, machines, molecules, etc.

Considering the concept of life, the research of AL is scientific, engineering, philosophical, and is thus interdisciplinary. The fact is not surprising for us. We can also understand that, in general, AL is based on many observations in computer science.

In addition, in AL, the concept of synthesis plays an important role. Synthesis means to combine several elements to form a whole. In computer science, such an element is usually called a *component*, or an *agent*.

AL as defined above is analogous to life in nature, since life in nature is ultimately a collection of cells or DNA and the functions of individual cells determine the functions of the entire living organism.

Therefore, a life can be interpreted as a system. Namely, a life is a system whose components are cells. And the differences of humans and other animals depend on those in cell composition.

Langton says about the goal of AL as follows; see Langton [18].

> The ultimate goal of the study of artificial life would be to create 'life' in some other medium, ideally a virtual medium where the essence of life has been abstracted from the details of its implementation in any particular hardware. (p. 147)

For the remarks, we believe that a medium means a hardware, and that the essence of life means processes, respectively. An abstracted virtual medium refers to a software.

Therefore, Artificial Life can be understood as being synthesized by means of hardware or/and software. The important point here is that a "computer" plays an essential role for investigating Artificial Life.

[1] He is known as one of the founders of quantum mechanics by proposing wave mechanics.

1.1.2 Three Positions of Artificial Life

By the way, from the standpoint of the study of Artificial Life, there are four positions, i.e. software, hardware, biochemistry, and art. Artificial Life based on each position is often called the following three:

(1) *Soft ALife*
(2) *Hard ALife*
(3) *Wet ALife*
(4) *ALife in Art*

Soft ALife is an Artificial Life realized as a computer software. Examples in the category include cellular automata, neural networks, and Evolutionary Computation. Some of the theories in soft Alife have also been used for the research of Artificial Intelligence (AI).

Its subfields and systems include artificial chemistry, complex adaptive systems, artificial ecology, digittal evolution, stringmol, etc. For example, *complex adaptive systems* (CAS) is a framework for studying, explaining, and understanding systems of agents that collectively combine to form emergent, global level properties.

Hard ALife is an Artificial Life realized as a computer hardware (or a machine). Examples in the category include self-replicating machine and robots.

Its subfields and systems include evolutionary robotics, cognitive robotics, FloraRobotics, etc. For example, *florarobotica* is an innovative, smart approach to combining technology with living organisms.

Flora Robotica is the idea of creating a bio-hybrid system of tightly coupled natural plants and distributed robots to grow architectural artifacts and spaces. Their motivation with this ground research project is to lay a principled foundation towards the design and implementation of living architectural systems that provide functionalities beyond those of orthodox building practice, such as self-repair, material accumulation and self-organization. The reader is referred to Hamann et al. [11] for details.

Wet ALife is an Artificial Life biochemically realized. Examples in the category include artificial bacteria, artificial cells, and synthetic biology.

Its subfields and systems include Droplets, Xenobots, etc. For instance, *Droplets* are created by putting a drop of one liquid (such as oil) into a different liquid that it cannot mix with (such as water). ALife in Art uses Artificial Life systems to create art and using art to examine the properties of and assumptions about life-like systems.

ALife in Art is manifested in a variety of ways, including computer-based, evolutionary life-systems (virtual ecologies); animated creatures with life-like behaviors (virtual creatures); living organisms and semi-living objects.

In these forms, life phenomena are artificially realized, but their forms are quite different. However, there is a sense in which the final form should be called “life”. We can

also say that theories and methods of computer science and biochemistry undoubtedly play important roles of the study of the above three categories in Artificial Life.

Artificial life is artificially synthesized "life", but it must satisfy the properties of life in the natural world. Of course, as the basic nature of life, there are several possibilities.

It seems that the following properties are essential for a life.

---

(1) self-regenerable

(2) evolvable

(3) learnable

---

(1) means that a life is inherently regenerative. A life is to be able to create offspring and repair its own functional defects. For example, humans can recover from minor injuries and give birth to children.

Self-regenerability may include self-replication. We can find it the work of cellular automata by von Neumann, Langton, etc.

(2) means that a life is interrelated with other lives and can choose the best change. In other words, evolvability is the ability of a biological system to produce phenotypic variation that is both heritable and adaptive. For example, according to Darwin's theory of evolution, humans evolved from a family of monkeys.

Evolution plays a crucial role in the so-called evolutionary computation. The main purpose of evolutionary computation is to search suitable solutions in problem spaces that are difficult to explore with more traditional heuristic methods.

(3) means that a life can learn various things from external information. For example, humans can acquire knowledge by studying. Of course, other animals are also thought to be learning by various methods.

Learning is a basic aspect of adaptive behavior for living organisms. Computer scientists attempted to model learning by a computer, This is the origin of machine learning, which has been also developed by the methods studied in neural networks.

The above three properties are the innate properties of living things in nature. In other words, any living thing is born with these properties in different forms. Therefore, Artificial Life must also artificially realize such basic properties.

Of course, there may be other properties that can be considered to be essential for Artificial Life. For example, humans have hearts and can cooperate. And so do other animals may have such properties. Observe that behavior, society, development, etc. are considered as intriguing properties of Artificial Life.

Thinking in this way, when researching Artificial Life, it is necessary to ask a philosophical problem, i.e., what is life? The problem cannot be avoided in our study. In other words, an Artificial Life system differs from existing systems. Therefore, we can say that it has a peculiar point dealing with the concept of life.

The above three properties are believed to have been the starting point of research on Artificial Life. And it is certain that such properties are reasonable. But, it would be impossible to say that these are all the properties of life.

In other words, a detailed philosophical discussion on life is considered to be necessary. In particular, the above problem is also of importance to the research on Artificial Life.

1.2 History and Related Areas

Here is a brief introduction to the history of Artificial Life. We believe that its history (including prehistory) began in 1940s. Note that the emergence of the field of AL, only about 30 years have passed. And in recent years, foundations for Artificial Life have presumably established to some extent. Therefore, the overview of the history may be very interesting for the readers.

1.2.1 1940s–1970s

From 1940s to 1970s, many theories that are now the foundations of Artificial Life were proposed. However, not all of these are aimed at (artificial) life. In the late 1940s, von Neumann and Ulam proposed a discrete computational model which is self-replicable, i.e., now called *cellular automata*: see von Neumann [26].

In 1945, von Neumann gave a lecture for designing a machine whose complexity can grow automatically similar to biological organisms by using natural selection. For this purpose, he could reach the idea that one must construct machines which can evolve.

Undoubtedly, cellular automaton is one of the theories that gave the greatest impact on Artificial Life. The main features of cellular automata are self-regenerable and evolvable by means of simple rules. Cellular automata can also be applied to games.

In 1948, Wiener proposed *Cybernetics*; see Wiener [27, 28]. He defined cybernetics as the science of control and communications in the animal and machine thus giving foundations of any "system". This means that Wiener's idea also gave some impacts on Artificial Life although it may be neglected by researchers.

He created the term "Cybernetics" from the Greek word for "steersman" to describe the principle governing or directing a technology or system. Wiener claimed to have been the first to unify control theory with communications theory. He explored its implications for philosophy, psychology, and mathematics. We discuss Cybernetics in connection with Artificial Life in Chap. 7.

Therefore, Cybernetics described phenomena in terms of their function rather than their substrate, so similar principles were applied to animals and machines alike. Some of the ideas in Cybernetics seemed to provide a road to Artificial Life, but he did not explicitly discuss Artificial Life.

In 1958, Rosenblatt developed a *perceptron*; see Rosenblatt [23]. The perceptron was the beginning of the research on the so-called *Neural Networks*. His work is

obviously influential for Artificial Life. He addressed the point that basic laws of organization are common to all information handling systems including machines and men.

Rosenblatt's perceptron can be regarded as a classical model of neural networks. The interesting features are that it can process non-boolean inputs and that it can assign different weights to each input automatically and the threshold is computed automatically.

Perceptron is one of the simplest Artificial neural networks. It is the simplest type of feedforward neural network, consisting of a single layer of input nodes that are fully connected to a layer of output nodes. It can thus learn the linearly separable patterns.

In 1968, Lindenmayer proposed *L-system* that mathematically models the growth process of living things; see Lindenmayer [19]. It is based on formal grammar theory and is suitable to the mathematical formulation of the growth process of living things.

Lindenmayer first used L-systems in order to describe the growth patterns of algae, and applied to various living things. L-systems can viewed as formal grammars in the sense of Chomsky, but are more powerful. In L-systems, structures develop through a process of string rewriting.

A string of letters is transformed into a new string of letters by simple rules called productions. The process is repeated indefinitely, each time using the string that was just produced as the source for the next string.

Since the mid of the 1970s, there have been many studies that foreshadow the establishment of the field of Artificial Life (also Artificial Intelligence). They also play a crucial role for developing some Artificial Life systems.

In 1970, a cellular automaton called the *Game of Life* due to Conway was featured by Gardner; see Gardner [9]. It is a game that models the processes of life's birth, evolution, and selection by means of simple rules.

In the Game of Life, we start with a simple configuration of counters (organisms), one to a cell, then observe how it changes as you apply Conway's "genetic laws" for births, deaths, and survivals. The Game of Life provided one of the games, and it can be interpreted as a kind of cellular automata.

In 1975, Holland proposed the *Genetic Algorithm* (GA) in connection with AI; see Holland [13]. Now, the method based on the notion of evolution is generally said to be *Evolutionary Computation*.

GAs are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. GAs are based on the ideas of natural selection and genetics. In other words, GAs rely on Darwin's theory of evolution.

These are intelligent random searches provided with historical data to direct the search into the region of better performance in solution space. They are commonly applied to generate high-quality solutions for optimization problems and search problems.

1.2.2 1980s

In the 1980s, research that could offer the foundation of Artificial Life began. Note that Langton is actually credited with being the founder of the field of Artificial Life. In 1987, Langton organized the first Workshop on the Synthesis and Simulation of Living Systems in Santa Fe, New Mexico, where the term "artificial life" was coined in its current usage.

However, his proposed concept of Artificial Life may be slightly different from that widely recognized in the present time. His first definition of Artificial Life is "life made by man rather than by nature", Later, he thought Artificial Life as being more wider sense, in that it involves biology in addition to computer science. We will discuss his several definitions of Artificial Life in more details in Chap. 7.

Langton has been researching cellular automaton since the late 1970s; see Langton [17, 18]. And it seems that he was already aware of the potential for Artificial Life in the mid-1980s.

In 1982, Wolfram classified the classes of the complexity of one-dimensional cellular automata, now known as the *Wolfram class*. Furthermore, Wolfram investigated the connections of one-dimensional cellular automata and life; see Wolfram [30–32].

Wolfram proposed a classification of one-dimensional cellular automaton rules into four types. It is base on the results of evolving the system from a disordered initial state.

We summarize as follows: (class 1) Evolution leads to a homogeneous state. (class 2) Evolution leads to a set of separated simple stable or periodic structures. (class 3) Evolution leads to a chaotic pattern. (class 4) Evolution leads to complex localized structures, sometimes long-lived.

In 1982, Bennett proposed a computer based on thermodynamics; see Benett [4]. This research suggested a DNA computer which is one of the bases of Artificial Life. Benett described the Brownian computer based on the principle of Brownian motion, showing a possibility of computing based on molecules.

In 1982, Hopfield proposed a new type of neural network called *Hopfield network*; see Hopfield [14]. His theory can serve as a model for human memory, thus giving some impacts on AI and cognitive science.

The Hopfield Network is a form of recurrent artificial neural network (ANN). It is a straightforward associative memory that has the ability to retain and recall patterns. It can be applied to pattern recognition, content-addressable memory, and optimization problems.

In 1984, Langton used a cellular automaton to compute successfully generated self-replicating tissue loops; see Langton [17]. Now, the loop is called *Langton's loop*. His work is understood as the substantial beginning of the research of Artificial Life.

In 1985, Conrad showed the basic principle of the so-called *molecular computer*; see Conrad [6]. Later he proposed molecular computing in which molecular and bioelectronics, biocomputing, biomolecular information processing, and biochip are

some of the terms that have been used to describe this new direction. Another term is molecular functional systems.

In 1985, Ackley, Hinton, and Sejnowski proposed the *Boltzmann machine* (BM) which is a type of recurrent neural network whose nodes make binary decisions with some bias; see Ackley, Hinton, Sejnowski [1]. BM can learn the probability density from the input data to generate new samples from the same distribution.

A BM is a network of symmetrically connected, neuronlike units that make stochastic decisions about whether to be on or off. BMs have a simple learning algorithm allowing them to discover interesting features in datasets composed of binary vectors.

The learning algorithm is very slow in networks with many layers of feature detectors. However, it can be made much faster by learning one layer of feature detectors at a time.

In 1986, *Parallel Distributed Processing model* (PDP model) was proposed by Rumelhart and McClelland; see Rumelhart and McClelland [22]. In the model, memories are stored and retrieved in a system with simple computational elements, all working at the same time and all contributing to the outcome.

In PDP models, memories are stored and retrieved in a system consisting of a large number of simple computational elements, all working at the same time and all contributing to the outcome.

They are sometimes also called *connectionist models*. This is because the knowledge that governs retrieval is stored in the strengths of the connections among the elements.

The advent of Hopfield network, parallel distributed processing model, and Boltzmann machine, the neural networks have regained attention in AI. However, their connections with AL have not been fully discussed.

In 1987, the First International Conference on Artificial Life was held in Los Alamos, USA. At this international conference, Langton proposed a definition of Artificial Life which can be seen as a starting point of Artificial Life research.

Since then, the International Conference on Artificial Life has been held every year, and the study became active. Note also that some conferences adopt contributions on Artificial Life.

In 1987, Reynolds constructed a system called the *boids*. Here, "Boids" stands for "Bird-oid". The system simulates the movement of a flock of birds; see Reynolds [21]. Boids is based on a model of polarized, noncolliding aggregate motion, such as that of flocks, herds, and schools.

Boids are a type of flocking algorithm, and stand for bird-oid object, signifying how it has been used to realistically represent flocks of birds or schools of fish without any special intelligence and easily translatable, standardized behaviours.

1.2.3 1990s

In the 1990s, Artificial Life became a hot topic. Also, many Artificial Life systems were developed. In 1990, Koza proposed *Genetic Programming* as an application of the Genetic Algorithm; see Koza [15, 16]. It is one of the methods of evolutionary computation and can be applied to automatic programming.

In 1993, the academic journal *Artificial Life* was launched by MIT Press. It published both research papers and review papers. By reading the articles published in this journal, you can see the research trends of Artificial Life.

In the same year, Ray developed AL system called *Tierra*, which simulates the dynamics and evolutionary processes of an ecosystem and proliferates in the memory of a computer; see Ray [20].

In 1994, Adleman at the University of Southern California experimentally realized a *DNA compute*; see Adleman [2]. By this work, DNA computers came to be noticed.

In 1996, Boden edited several papers for the philosophy of Artificial Life; see Boden [5] for details. Artificial life research seeks to synthesize the characteristics of life by artificial means, particularly employing computer technology.

The essays in the book explore such fascinating themes as the nature of life, the relation between life and mind, and the limits of technology. We believe that the book is the early work on the philosophy of AL.

In 1996, the first clone of female sheep called Dolly, an adult mammal, was produced by Wilmut in U.K.; see Wilmut [33], We will discuss clone in Chap. 5.

1.2.4 2000s

In the 2000s, biotechnology attracted attention, and research on *wetware* began to accelerate. Also noted is that many commercial applications of Artificial Life were appeared.

In 2000, Hagiya's group in University of Tokyo proposed the DNA computer for solving the satisfiability of three variables (3-SAT); see Sakamoto et al. [24]. Their DNA computer was developed by using the method called the *hairpin*.

Hairpin formation by single-stranded DNA molecules was exploited in a DNA-based computation to explore the feasibility of autonomous molecular computing. The satisfiability problem, which is a famous hard combinatorial problem, was solved by using molecular biology techniques. The satisfiability of a given Boolean formula was examined autonomously, by hairpin formation by using the molecules that represent the formula.

In 2004, Shapiro and colleagues at the Weizmann Institute in Israel developed a DNA computer with input-output mechanism; see Benenson et al. [3].

Their DNA computer encodes both the software and the data in the four letters of the genetic code, A, C, G and T. The hardware which is the part of the computer that does not change, is an enzyme that cuts the strands of DNA in a particular way.

In 2010, Venter's group announced the synthesis of artificial DNA containing the entire genetic information of a bacterium; see Gibson et al. [10]. Venter's work can be regarded as the first step to the study of wet Alife.

This achievement is very shocking and truly an Artificial Life. This news was also reported in several countries.

In 2021, Xu et al. developed artificial cell-like structures by inorganic matter that autonomously ingest, process, and push out material—recreating an essential function of living cells; see Xu et al. [34]. They created the so-called *cell mimic*, which has potential applications ranging from drug delivery to environmental science.

In relation to AL research, the advent of *generative Artificial Intelligence* should be recognized. Generative Artificial Intelligence can generate texts, images, videos by means of generative models.

The first interesting generative AI system is clearly *ChatGPT* developed by OpenAI in 2022. Since then, similar generative AI systems have been developed by others. We believe that generative AI systems are directly or indirectly related to AL research.

However, some people pointed out the disadvantages of generative AI. For instance, it can easily create fake news, fake images, and fake video. It also seems to provide negative influences for our daily life in that it could be used as a practical tool for crimes.

We end with a brief history of Artificial Life. The field of Artificial Life is steadily progressing, and the results are socially beneficial with great impacts. Historically, there are many cases in which research in other fields began to affect Artificial Life later.

Artificial Life is based on a new way of thinking about life, It is related to various existing fields. Here, we will introduce such relations. As mentioned above, Artificial Life is generally considered a field of computer science.

1.2.5 Scientificc Theory

Now, some words about *scientific theory*. A *theory* is characterized as a postulated system, i.e., a set of premises. And it deduces empirical laws as theorems. For this purpose, it has an abstract logical form, with axioms, formation rules, and rules for deductions from the axioms as well as definitions for empirically interpreting its symbols.

Scientific theories are of special importance for the philosophy of science and technology. For scientific theories, the reader is advised to consult, e.g., Gale [8] for details.

Generally, they expand a family of empirical (experiential) laws for regularities existing in objects and events. A scientific theory is a structure suggested by these laws and is devised to explain them in formal manner.

Scientific theories generally consist of data, experiments, formalizations, and hypothesis testing. Data are obtained from experimental observations. Then, the formalization is a theory that provides hypotheses to explain data. Hypothesis testing then validates the theory.

Most scientific theories are based on such methodologies. For example, in physics, a physicist performs experiments and observes them to obtain data. Then, he will give the results of an experiment a theory to explain them, i.e., *model*.

A model is an abstraction of a phenomenon and is generally described mathematically. There are various models that are compatible to the phenomenon. For example, Newton formalized the equations of motion to describe data about the motion of bodies. Therefore, the equations of motion can be considered as one of the models of mechanics.

Models are used not only in the natural sciences but also in the social sciences. For example, population models are described by differential equations. There are also models that are described linear algebraically or stochastically.

Also, the formalization is hypothetical and should explain the actual data well. Here, whether or not it can be "explained well" is checked by a test. The method of testing differs depending on the field, and statistics, etc., are used.

There are traditionally three views about scientific theory, i.e., syntactic, semantic and pragmatic view. In the *syntactic view*, the structure of a scientific theory is reconstructed by means of sentences used in a metamathematical language. Thus, the basic tool is mathematical logic.

In the *semantic view*, the structure of a scientific theory is reconstructed by logic and set theory. It is to give apparatuses "meaning".

In the *pragmatic view*, the structure of a scientific theory is complex, thus implying scientists employ different kinds of theories for their purposes.

Summarizing these three views by Winther [29]. Theory structure is uninterpreted as an axiomatic system in the syntactic view; model-theoretic and set theoretic in the semantic view; internal and external pluralism in the pragmatic view.

Theory interpretation is done by correspondence rules in the syntactic view; by hierarchy of models, similarities, and isomorphism in the semantic view; by structure already inflected by practical function and application, and pragmatic virtues in the pragmatic view.

Is theory interpretation an aspect of theory structure? Yes in the syntactic view; No in the semantic view; Yes (but the distinction is hard to make) in the pragmatic view.

Traditional views about scientific theory may be incompatible to the theory of Artificial Life. Some of the thories of Artificial Life cannot adequately captured by the above three views. De we need different classifications of scientic theories?

Scientific theories can also be classified by *top-down theory* and *bottom-up theory*. In many scientific fields both approaches are used, and are compatible to Artificial Life. The fact is not surprising.

Top-down theory formalizes a theory and thereby explains a real phenomenon. i.e.. it makes a hypothesis and tests its validity with data. Thus, top-down theory can predict new phenomena. Bottom-up theory formalizes a theory based on real

phenomena. It formulates a hypothesis to formalize the obtained data. Therefore, bottom-up theory can describe existing phenomena

More intuitively, top-down theory is the theory from "abstract" to "concrete", Bottom-up theory is a theory from "concrete" to "abstract". Note that the ideas of "top-down" and "bottom-up" are common in computer science. For instance, the classification may be found in *system*.

A system consists of several elements that collectively provide a specific function. Here, the constituent elements are referred to as components or subsystems. For example, when developing software systems, there are top-down and bottom-up development methods.

In the top-down development method, we first design the entire system and then design subsystems. Then, we realize the whole system. In contrast, in the bottom-up development method, we first design many subsystems and synthesize them to realize the whole system.

An important feature of Artificial Life is that it is a bottom-up theory. This point is contrasted with the feature of Artificial Intelligence, which is a related area and is a top-down theory.

That is, by compositing certain components in various ways, Artificial Life is realized. Here, although there is the question of what "life" is, to a certain extent the whole picture about the organisms like human is our general knowledge.

Therefore, a bottom-up approach is extremely important for Artificial Life research. This is because various theories are needed to explain the whole picture of life.

It should also be noted that the fact that Artificial Life is a bottom-up theory applies to soft ALife, hard ALife, and wet ALife. Of course, this does not mean in the research of Artificial Life we don't need top-down theory, and the top-down thinking should be explored for Artificial Life.

1.2.6 Related Areas

There are some areas related to Artificial Life. The main areas related to Artificial Life are as follows.

- Artificial Intelligence (AI)
- Philosophy
- Biochemistry

In fact, the methods and ideas in these areas have been incorporated into Artificial Life research.

Artificial Intelligence (AI) is the field of research into realizing human intelligence in computers. The themes of AI include knowledge representation, automated reasoning, machine learning, natural language understanding, etc. We know that AI exploits success in many areas.

In AI, neural network, evolutionary computation, agent etc. are also studied, so there are research fields that overlap with research fields in AL, However, the approaches to AI and AL seem to be fundamentally different.

In other words, AI is top-down, but AL is bottom-up. Therefore, in AL, individual components alone are not viewed as systems of AL. By integrating the necessary components, the whole AL systems are realized.

Also, the differences between AL and AI are often emphasized. The fact heavily depends on the interpretations of life and intelligence, and there are different opinions even by researchers.

Generally speaking, AL itself generates a "life imitation" (system). And the realized system, regardless of the form, implements the functions that life has. Also, AL exists as itself a system like natural life.

On the other hand, AI is consistent with intelligence of humans, etc., and generates something that can behave, that is, a "system". Also, not all components of AI necessarily exist.

Therefore, AI need not be considered part of AL. Of course, both fields use similar techniques, so they are closely related. It may be thought that they are complementary to each other.

Philosophy studies our thinking in general, And existence, concepts, relationships, etc. become objects of study. Therefore, in AL, philosophical discussions of life are not avoided. In Schrödinger [25], he gave interesting discussion on life, thus relating it to the philosophy of AL.

Philosophy is clearly of special importance for AL due to its characteristics. Some concepts like artificial, life, etc. are in fact philosophical, and they were discussed in number of works so far.

Historically, many AL researchers discussed the philosophy of AL. However, they do not seem to be sufficient. We will discuss some philosophical issues of AL in Chap. 7.

Biochemistry is an area that investigates life phenomena using chemical methods. The object of biochemistry is the molecular level to the DNA level. It can serve as the basis of AL, especially DNA computing.

Biochemistry also constitutes the foundations for AL including wetware. It is in fact related to some notions discussed in AL including synthesis, evolution, emergence and simulation.

Studying life by synthesizing and simulating leads us to experiment with different forms of life. We think that this gives sharp experimental answers to many general questions about the nature of (artificial) life.

What are we waiting for Artificial Life? This is nothing but a future the AL. It community should actively construct. Then, at 60 years, we expect to see celebrate our achievements.

References

1. Ackley, D., Hinton, G., Sejnowski, T.: A learning algorithm for Boltzmann machine. Cognit. Sci. **9**, 147–169 (1985)
2. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science **266**, 1021–1024 (1994)
3. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E.: An autonomous molecular computer for logical control of gene expre3ssion. Nature **414**, 430–434 (2004)
4. Benett, C.: The thermodynamics of computation-a review. Int. J. Theor. Phys. **21**, 905–940 (1982)
5. Boden, M. (ed.): The Philosophy of Artificial Life. Oxford University Press, Oxford (1996)
6. Conrad, M.: On design principles for a molecular computer. Commun. ACM **28**, 464–480 (1985)
7. Descartes, R.: Treatise on man. In: C. Clerselierand T. Girard (ed.), De l'homme et de la formation du foetus, Paris, Sloan (1677)
8. Gale, G.: Theory of Science: An Introduction to the History, Logic, and Philosophy of Science. McGraw-Hill, New York (1979)
9. Gardner, M.: Mathematical games. Sci. Am. **223**, 120–123 (1970)
10. Gibson, D., et al.: Creation of a bacterial cell controlled by a chemically synthesized genome. Science **329**, 52–56 (2010)
11. Hamann, H. et al: Flora robotica – An Architectural System Combining Living Natural Plants and Distributed Robots (2017)
12. Hobbes, T.: Leviathan, or, The Matter, Forme, and Power of a Common Wealth, Ecclesiasticall and Civil. London: Andrew Crooke, 16512
13. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
14. Hopfield, J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Acad. Sci. (USA) **79**, 2554–2558 (1982)
15. Koza, J.: Genetic Programming: A Paradigm For Genetically Breeding Populations of Computer Programs To Solve Problems. Stanford University (1990)
16. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press. Mass, Cambridge (1992)
17. Langton, C.: Self-reproduction in cellular automata. Phys. D **10**, 135–144 (1984)
18. Langton, C.: Studying artificial life with cellular automata. Physica D **22**, 120–149 (1986)
19. Lindenmayer, A.: Mathematical models for cellular interaction in development I: Filaments with one-sided inputs. J. Theor. Biol. **18**, 280–289 (1968)
20. Ray, T.: Evolution and optimization of digital organisms. In: Billingsley, K., et al. (eds.), Scientific Excellence in Supercomputing, pp. 489–531. The Baldwin Press (1991)
21. Reynolds, C.: Flocks, herds and schools: a distributed behavioral model. Comput. Graph. **21**, 25–34 (1987)
22. Rumelhart, D., McClelland, J. (eds.): Parallel Distributed Processing. MIT Press, Cambridge, Mass., two volumes(1986)
23. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**, 386–408 (1958)
24. Sakamoto, K., Gouzu, H., Komiyama, K., Kiga, D., Yokoyama, S., Yokomori, T., Hagiya, M.: Molecular computation by DNA hairpin formation. Science **288**, 1223–1226 (2000)
25. Schrödinger, E.: What is Life? The Physical Aspect of the Living Cell. Cambridge University Press, Cambridge (1945)
26. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana (1966)
27. Wiener, N.: Cybernetics. Wiley, New York (1948)
28. Wiener, N.: The Human Use of Human Beings. The Riverside Press (1950)
29. Winther, R.: The structure of scientific theories, The Stanford Encyclopedia of Philosophy (2021), https://plato.stanford.edu/archives/spr2021/entries/structure-scientific-theories/

30. Wolfram, S.: Universality and complexity in cellular automata. Phys. D **10**, 1–35 (1984)
31. Wolfram, S.: Theory and Applications of Cellular Automata. World Scientific, New York (1986)
32. Wolfram, S.: Random sequence generator by cellular automata. Adv. Appl. Math. **7**, 123–169 (1986)
33. Wilmut, L., et al.: Viable offspring derived from fetal and adult mammalian cells. Nature **385**, 810–813 (1997)
34. Xu, Z., Huechel, T., Irvine, and Scanna, S.: Transmembrane transport in inorganic colloidal cell-mimics. Nature **597**, 220–224 (2021)

Chapter 2
Cellular Automata

Abstract This chapter describes cellular automata, which are used to model the self-reproducing function of organisms. Here we introduce types of cellular automata, the Wolfram class, the Game of Life, and Langton's loop.

2.1 Backgrounds of Cellular Automata

2.1.1 What Are Cellular Automata?

Cellular Automaton was proposed by von Neumann and Ulam in the 1940s as one of the self-reproducing discrete computational models; see von Neumann [12], Wolfram [13].

In 1945, von Neumann gave a lecture providing his goal, i.e., designing a machine whose complexity can grow automatically similar to biological organisms by means of natural selection. For this purpose, he could reach the idea that one must construct machines which can evolve.

Undoubtedly, cellular automaton is one of the theories that has had the greatest impact on Artificial Life, although he did not suppose it. The main features of cellular automata are self-regenerable and evolvable by means of simple rules. Cellular automata can also be applied to games.

Cellular automaton is also called *self-reproducing automaton* from the title of von Neumann's book. For the history of cellular automata; see Sarkar [10].

We do not know whether von Neumann aimed to give a basis of something like Artificial Life. But he was interested in the notion of self-reproduction which is an important function of life.

Von Neumann said:

> The a priori argument against the possibility of self-production is that it is natural to expect the constructing automaton to be more complex than the constructed one-i.e., the primary will be more complex than the secondary. (p. 118)

We believe that his idea may change the history of Artificial Life due to its originality and novelty. As will be discussed below, the area of Artificial Life has emerged by Langton's work on cellular automata.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Akama, *Artificial Life*, Intelligent Systems Reference Library 262,
https://doi.org/10.1007/978-3-031-74422-8_2

A cellular automaton consists of an infinite grid of *cells*, Each cell has a finite internal state. The internal state of a cell changes over time according to certain rules. Also, internal state changes are determined by the cell itself and its neighbours (surroundings).

Changes in internal state described in this way can describe self-reproducing. i.e. which is obtained by repeated application of simple rules. Cellular automata are studied as models of AL because they appear to simulate one of the basic functions of life.

Now, we look at the history of cellular automata. In the 1940s, Ulam at the Los Alamos Laboratory used a simple lattice network to study crystal growth. At the same time, the institute also had von Neumann, who was working on self-replicating systems.

Von Neumann seems to have had in mind the construction of self-replicating robots by means of his cellular automata, At Ulam's suggestion, he shifted his emphasis to develop mathematical theory. Then, von Neumann has succeeded in formalizing the basics of self-reproducing automata.

Additionally, von Neumann sought to investigate the question of life's origin by designing a self-reproducing automaton. This machine was to operate in a very simplified environment for seeing reproduction.

Von Neumann's cellular automaton is a two-dimensional cellular automaton with 29 states, capable of self-replication. In addition, von Neumann used cellular automata to develop a self-replicating machine called the *Universal Constructor*.

The Universal Constructor is based on a certain pattern of cell states in the cellular automaton. It contains one line of cells that serve as the description similar to the tape in Turing Machine, encoding a sequence of instructions that serve as a blueprint for the machine. The machine reads these instructions one by one and performs the corresponding actions.

To build a universal self-reproducing machine for evolving more complex machines, von Neumann recognized that he extended the Turing machine-like machine to output another machine rather than a ribbon of tape with a bunch of ones and zeroes printed on it.

Von Neumann conceived a machine composed of three components: (1) a blueprint describing the machine, which in analogy to the Turing tape, carries instructions for how to build another machine; (2) a universal constructor which decodes the instructions to construct the machine; (3) a universal copying machine which makes a copy of the instructions.

A machine builds a copy of itself using the instructions, then makes a copy of the instructions, and feeds them into the new machine, and so on.

After his death, von Neumann's theory of cellular automata was published as a book "Theory of Self-Reproducing Automata"; see von Neumann [12]. It seems that his later years were spent for exploring cellular automata.[1]

[1] Von Neuman's book was completed by Burks in 1966 and it contains his detailed introduction.

2.1.2 History of Cellular Automata

In 1968, Codd proposed an eight-state cellular automaton; see Codd [5]. While von Neumann's cellular automaton requires 29 states, Codd's cellular automaton can be considered as a simplification of von Neumann's.[2]

In 1970, Conway proposed the *Life of Game* by using cellular automata. The Game of Life was introduced and made famous by the mathematician Gardner; see Gardner [3, 4]. It reveals the possibility of cellular automata for games.

Since 1980, research on cellular automata has been conducted by Wolfram, etc. and it went fast. In particular, Wolfram wrote a number of articles on cellular automata; see Wolfram [13–16].[3]

In 1984, Langton proposed the so-called *loop* by using von Neumann's cellular automaton. He used the term "Artificial Life", and his work was the starting point of Artificial Life; see Langton [6, 7].

2.1.3 Classification of Cellular Automata

Next, we quickly review a classification of cellular automata. A cellular automaton consists of any number of cells arranged in 1, 2, 3 or more dimensions. Each has a state associated with it (on or off) and each cell and therefore the entire automaton transitions from one state to the next over time by using a rule or set of rules.

The types of cellular automata are as follows:

- one-dimensional cellular automaton
- two-dimensional cellular automaton
- reversible cellular automaton
- totalistic cellular automaton

One-dimensional cellular automaton, also known as an elementary cellular automaton, consists of a row of cells, each of which can be "on" or "off". And there are two immediate neighbors.

Two-dimensional cellular automaton consists of a two-dimensional grid of cells, and there are eight immediate neighbours. The cellular automaton, as proposed by von Neumann in 1952, is a two-dimensional cellular automation.

It also gives a model of games such as the Game of Life. It is known that it is Turing complete in the sense that it can simulate Turing machines.

Reversible cellular automaton is a cellular automaton whose rule is reversible in the sense that it has only one previous configuration. Thus, it is used to investigate the thermodynamic behavior of systems by means of computer simulation.

[2] Codd is also known as the founder of relational databases.

[3] Wolfram is also famous as the developer of computer algebra system *Mathematica*.

Totalistic cellular automaton is a cellular automaton in which each cell's state is defined by a finite set of integer values, and the cell's value at time t depends on the sum of the values of its neighbors at time $t - 1$. The Game of Life can be interpreted as a totalistic cellular automaton whose values are 0 and 1.

2.2 Wolfram Class

One-dimensional cellular automata are the most basic cellular automata. And, their properties have been studied in detail by Wolfram. According to Wolfram, one-dimensional cellular automata can be classified into his four classes, class 1 to class 4, depending on how time passes; see Wolfram [14].

His classification of one-dimensional cellular automaton rules into four types. It is base on the results of evolving the system from a disordered initial state: (class 1) Evolution leads to a homogeneous state. (class 2) Evolution leads to a set of separated simple stable or periodic structures. (class 3) Evolution leads to a chaotic pattern. (class 4) Evolution leads to complex localized structures, sometimes long-lived.

This classification is generally called the *Wolfram class*. In the sequel, we describe the details of these classes.

2.2.1 Classes 1–4

In class 1, cellular automata rapidly converge to a uniform state. In class 2, cellular automata rapidly converge to a repetitive or stable state. In class 3, cellular automata appear to remain in a random state. In class 4, cellular automata form areas of repetitive or stable states, but also form structures that interact with each other in complicated ways.

2.2.2 Some Rules

There are many interesting rules for one-dimensional cellular automata. Here, we introduce the *rule 90*. Considering two neighbours, the internal state of the one-dimensional cellular automaton at time t are the following eight states:

111, 110, 101, 100, 011, 010, 001, 000

Here, the value of the center cell is the one to be focused on, and the neighboring cells are neighbors. The value of the center cell at time $t + 1$ can be computed from the sum of the two neighborhoods.

Note that if the cell value is 0 or 1, we need to calculate the remainder when the sum of the neighborhoods is divided by 2. So the rule 90 can be written as Table 2.1.

Table 2.1 Rule 90

Time t	111	110	101	100	011	010	001	000
Time $t+1$	0	1	0	1	1	0	1	0

If the values at time $t+1$ are arranged side by side, it will be 01011010, but if converted to decimal, it will be 90, so this rule is called the rule 90.

For example, if the internal state at time t is 111, then the internal state of the cell of the center at time $t+1$ is $\mathrm{mod}(1+1,2) = \mathrm{mod}(2,2) = 0$.

Sierpinski's *Gasket* (cf. Sierpinski [11]) is famous as a fractal and can also be considered as a cellular automaton generated by the rule 90. The gasket using the rule 90 looks like Fig. 2.1.[4]

Next, the *rule 30* will be explained. According to Wolfram, the rule 30 generates a cellular automaton belonging to the class 4, It is considered useful for elucidating natural phenomena.

In fact, cellular automata generated by the rule 30 are known to resemble the pattern of a potato shell. It is surprising that a simple rule like rule 30 can show *complexity*. The rule 90 can be written as Table 2.2.

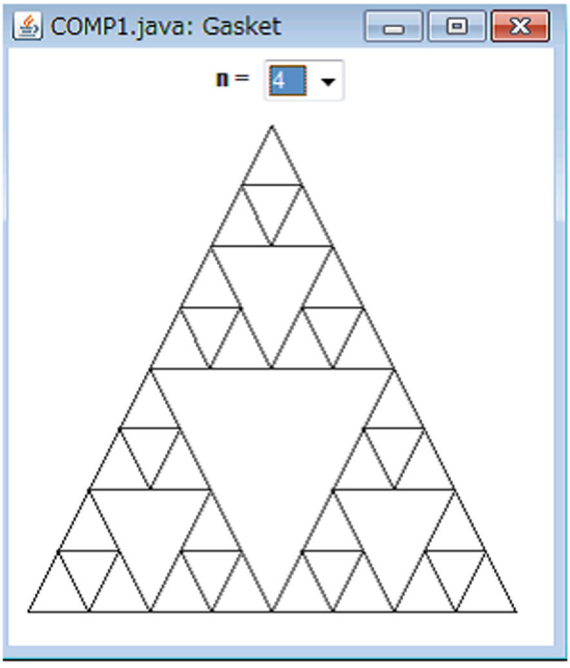


Fig. 2.1 Cellular automaton (Gasket)

[4] The program `COMP1.java` is written in Java application. It is available by the author on request.

Table 2.2 Rule 30

Time t	111	110	101	100	011	010	001	000
Time $t+1$	0	0	0	1	1	1	1	0

Similar to the rule 90, the rule 30 has an internal state of 00011110 at the central time $t+1$, which is converted to a decimal number to 30. This is the origin of the name.

Now, let the ith bit string at time t be $x(t, i)$. Then, the following holds.

$$x(t+1, i) = x(t, i-1) \text{ xor } (x(t, i) \text{ or } x(t, i+1))$$

Here, "xor" stands for exclusive logical sum, and "or" stands for logical sum, respectively. Such relationships can be obtained by simplification of Boolean functions.

The rule 30 produces the *random sequence generator*; see Wolfram [16]. For example, consider the internal state 100 at time t in Table 2.2. Then,

$$x(t, i) = 0,$$
$$x(t, i+1) = 0,$$
$$x(t, i-1) = 1$$

holds. Thus, we have:

$$\begin{aligned} x(t+1, i) &= x(t, i-1) \text{ xor } (x(t, i) \text{ or } x(t, i+1)) \\ &= 1 \text{ xor } (0 \text{ or } 0) \\ &= 1 \text{ xor } 0 \\ &= 1 \end{aligned}$$

The cellular automaton using the rule 30 looks like Fig. 2.2.[5]

Cellular automata play an important role in complex systems. In future work, we plan to discuss cellular automata in connection with complex systems.

2.3 Game of Life

The *Game of Life* is a game invented by Conway in the late 1960s. It can be considered an example of a two-dimensional cellular automaton. But unlike normal games, there are no players. In other words, it is a 0-person game. It was later popularized by Gardner's seminal papers; see [3, 4].

In the Game of Life, we start with a simple configuration of counters (organisms), one to a cell, then observe how it changes as you apply Conway's genetic laws

[5] The program `COMP8.java` is written in Java application. It is available by the author on request.

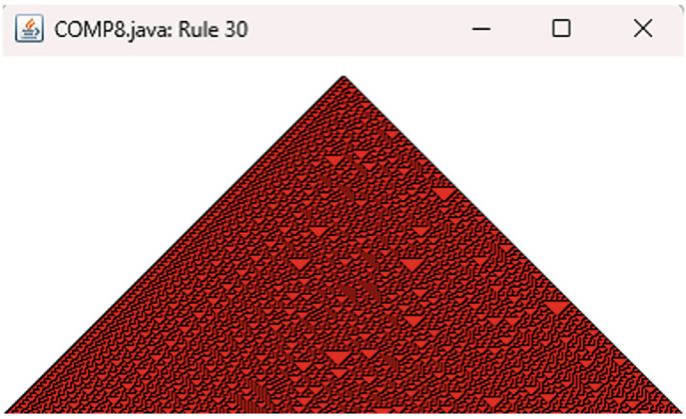


Fig. 2.2 Cellular automaton (rule 30)

described below: births, deaths, and survivals. It can be interpreted as a kind of cellular automata.

2.3.1 Is Life a Game?

Gardner [3] says about the features of the Game of Life:

> Because of its analogies with the rise, fall and alternations of a society of living organisms, it belongs to a growing class of what are called "simulation games"–games that resemble real-life processes. (p. 120)

Here, we briefly describe the Game of Life. Its basic idea is to start with a simple configuration of counters (organisms), one to a cell, then observe how it changes as you apply Conway's "genetic laws" for births, deaths, and survivals.

There are three rules, deciding cell's life or death.

(1) Survivals
Every counter with two or three neighboring counters survives for the next generation.

(2) Deaths
Each counter with four or more neighbors dies (is removed) from overpopulation. Every counter with one neighbor or none dies from isolation.

(3) Births
Each empty cell adjacent to exactly three neighbors–no more, no fewer–is a birth cell. A counter is placed on it at the next move.

Here, a 'cell' has the value 'black' (live) or 'white' (dead). The game repeats the alternation of generations according to the above rules. It is here important to understand that all births and deaths occur simultaneously.

Together they constitute a single generation or, as we shall call it, a "move" in the complete "life history" of the initial configuration.

Intuitively, these rules say that evolution is influenced by the environment. If there are suitable companions around the cell, it can survive and a new cell can be born.

However, if there are too few or too many friends around you, you cannot survive. The Game of Life therefore fits loosely with an organism's evolutionary process.

Figure 2.3 is an example of the state change of the Game of Life from Gardner [3].

Here, "0", "1", and "2" represent generations. Note that "0" represents the present. Now, let (0, 0) be the position of the upper left corner of the 5×5 grid. It shows the five triplets that do not fade on the first move.

The first three $[a, b, c]$ vanish on the second move. In connection with c it is worth noting that a single diagonal chain of counters, however long, loses its end counters on each move until the chain finally disappears.

The speed a chess king moves in any direction is called by Conway (for reasons to be made clear later) the "speed of light." We say, therefore, that a diagonal chain decays at each end with the speed of light.

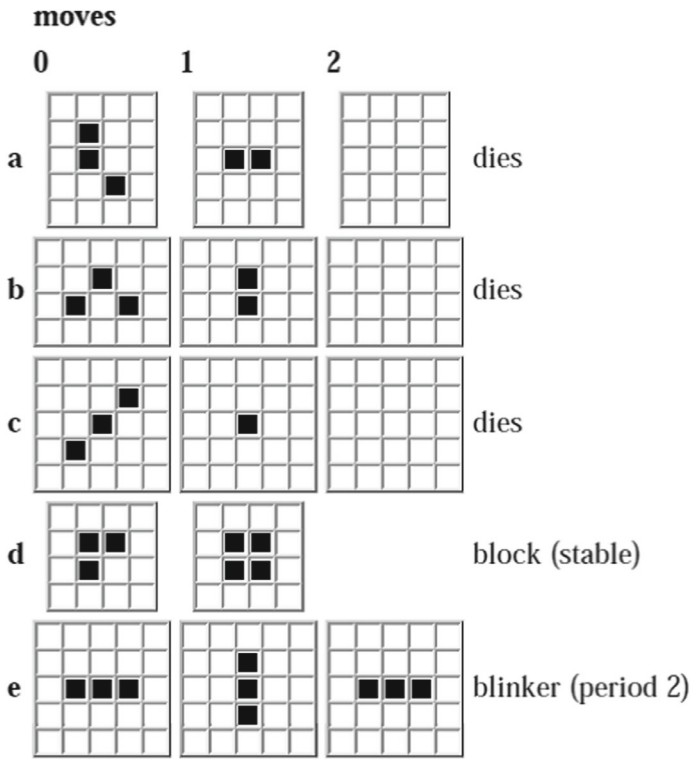


Fig. 2.3 The game of life

2.3.2 Computability of the Game of Life

The Game of Life can be interpreted as a kind of cellular automata. This suggests the duality of life and computers. And it reveals that the Game of Life is one of important methods of AL.

It turns out the Game of Life is Turing complete, i.e., it is also capable of universal computation. The famous Church-Turing Thesis states that everything computable is computable by a Turing Machine (TM). This statement is not proved, but is almost universally accepted. If it is true, then TMs may have power greater than modern (standard) computers.

Generally, if the behavior of cells in the Game of Life would be either repetitive (still life or oscillators cycle through 1 or more patterns) or chaotic, it would be hard to perform any computation. But gliders move and can interact with each other, thus enabling some non-chaotic processes.

2.4 Langton's Loop

The relationship between cellular automata and life was the starting point of von Neumann's research. Research into cellular automata also led to the establishment of the field of AL.

This may have been expected, but it took more than 30 years from von Neumann's research. If von Neumann had lived longer, the field of AL might have been founded earlier.

However, Langton has been researching cellular automaton since the late 1970s; see Langton [6, 7], and it seems that he was already aware of the potential for Artificial Life (based on cellular automata) in the mid-1980s.

2.4.1 Langton's Original Definition

In 1984, Langton proposed a cellular automaton, which is now called *Langton's loop*, simplifying von Neumann's two-dimensional cellular automaton; see Langton [6].

Langton's loop can be considered an AL in that it can self-replicate. Furthermore, it can be said that the possibility of cellular automata as a basic theory of AL was strongly demonstrated.

Langton had studied *self-reproduction* on the basis of von Neumann and Codd's cellular automata. Their cellular automata were proposed as a universal computer with the ability to self-reproduction. Observe that Langton also seems to have been influenced by Wolfram's work.

However, conventional cellular automata have a complicated structure. For example, von Neumann's Universal Constructor requires 29 states and 102 rules. Langton

considered the power of a universal computer to be a sufficient, but not a necessary, condition for self-reproduction. That is, self-reproduction does not necessarily require the computational power of a universal computer.

With this shift in thinking, Langton simplified the cellular automata of von Neumann and others and showed that they could self-reproduce. Cellular automaton, which was proposed by Langton, is now called *Langton's loop*.

In fact, many living organisms are capable of self-producing, although they cannot be considered to have computational power equivalent to a universal computer. Langton's interpretation is therefore reasonable, and it makes the connection between cellular automata and artificial life much clearer.

This idea led Langton to start the research area now called Artificial Life (AL). In 1986, Langton first used the term "Artificial Life" in Langton [7].

Langton says:

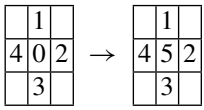
> We show that cellular automata are capable of supporting virtual automata that are equivalent to Turing machines and can thus perform any computable task. On this basis, we propose that the notion of the 'molecular logic of the living state' can be captured by the interactions of virtual automata and thus that the existence of artificial life within cellular automata is a distinct possibility. (p. 120)

From these remarks, he addressed the importance of cellular automata for investigating AL.

Now, we outline Langton's loop. It can be generally formalized by 8 states {0, 1, 2, 3, 4, 5, 6, 7} and 219 rules. The rule uses *von Neumann neighborhood*. Also, the total number of cells is generally 86.

Von Neumann neighborhood was used by von Neumann for his theory, considering the states of the north, south, east, and west cells to determine the next state of the reference cell. It should be noted that each cell can take his eight different values.

Here, the notation is specified as follows. For instance, 012345 used for rules denotes that if center cell value is 0, north cell value is 1, east cell value is 2, south cell value is 3, west cell value is 4, then center cell value will become 5.



Then, the rules for Langton's loop are defined as follows.

000000 000012 000020 000030 000050 000063 000071 000112
000122 000132 000212 000220 000230 000262 000272 000320
000525 000622 000722 001022 001120 002020 002030 002050
002125 002220 002322 005222 012321 012421 012525 012621
012721 012751 014221 014321 014421 014721 016251 017221
017255 017521 017621 017721 025271 100011 100061 100077
100111 100121 100211 100244 100277 100511 101011 101111
101244 101277 102026 102121 102211 102244 102263 102277
102327 102424 102626 102644 102677 102710 102727 105427
111121 111221 111244 111251 111261 111277 111522 112121
112221 112244 112251 112277 112321 112424 112621 112727

113221 122244 122277 122434 122547 123244 123277 124255
124267 125275 200012 200022 200042 200071 200122 200152
200212 200222 200232 200242 200250 200262 200272 200326
200423 200517 200522 200575 200722 201022 201122 201222
201422 201722 202022 202032 202052 202073 202122 202152
202212 202222 202272 202321 202422 202452 202520 202552
202622 202722 203122 203216 203226 203422 204222 205122
205212 205222 205521 205725 206222 206722 207122 207222
207422 207722 211222 211261 212222 212242 212262 212272
214222 215222 216222 217222 222272 222442 222462 222762
222772 300013 300022 300041 300076 300123 300421 300622
301021 301220 302511 401120 401220 401250 402120 402221
402326 402520 403221 500022 500215 500225 500232 500272
500520 502022 502122 502152 502220 502244 502722 512122
512220 512422 512722 600011 600021 602120 612125 612131
612225 700077 701120 701220 701250 702120 702221 702251 702321 702525 702720

Langton's loop self-reproducings itself is shown with step $t = 151$ as shown in Fig. 2.4 from Langton [6].

Note that the internal states are described as numbers, but can also be as colors. The initial state is the same as the right loop. And the sheath (the part on the right side) grows in a loop.

When a loop is formed in $t = 151$, it separates from the original loop and is duplicated in two as shown in Fig. 2.4. In addition, each grows to produce duplicates, the process is repeated, and creating more and more copies.

In Langton's loop, where the loop is formed, there is a new no loops can be formed. That is, a loop surrounded by other loops cannot self-replicate, and such a part is called the *colony*.

Colonies can also be considered "dead" parts in a sense, but they are also found in nature. For example, corals become coral reefs by forming colonies.

Langton's loop is notable for its ability to self-reproduce without full computational power. This seems to apply to organisms in nature as well, as mentioned above.

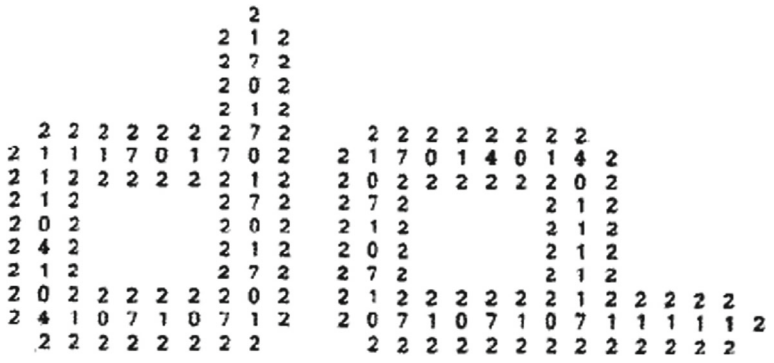


Fig. 2.4 Langton's loop (self-reproduction completed)

Thus, Langton's loop suggests that artificial life systems can be constructed without using complex systems such as universal constructors. The fact may be surprising because a life has to have the ability which is equivalent to the power of the Turing Machine.

A branch of Artificial Life research developed from his work, resulting in better insight into self-replicating processes, is of very importance. It has obvious relevance to Biology and living systems, thus having some connections with Artificial Life.

There is also a problem with the occurrence of loops. It is not clear why there is an organization (creature) like a loop. It can only be said that it is only for using the characteristics of cellular automaton.

This problem may not be a major problem considering the peculiarity of artificial life. In fact, related systems of Langton's loop have a similar self-reproduction function.

2.4.2 Some Simplifications

There are several attempts to simplify Langton's loop. We give some such approaches. In 1989, Byl simplified Langton's loop by removing the inner sheaths of loops; see Byl [1]. Byl's cellular automaton is simplified with 6 states and 12 cells in total, and self-reproduces in 25 steps.

In 1993, Reggia et al. proposed a similar model of cellular automaton; see Reggia et al. [9]. They discussed their approach by showing that self-replicating structures can emerge from non-replicating components and that genetic algorithms can also be applied to automatically program simple but arbitrary structures for replicating.

They also described recent work in which self-replicating structures are successfully programmed to do useful problem solving as they replicate. We conclude by identifying some implications and important research directions for our future works.

Finally, we mention two examples of variants of cellular automata. *Finite cellular automata* have a finite number of cells, and are closely related to the state transition diagram (STD). Finite cellular automata can be applied to VLSI design; see Chaudhuri et al. [2] for details.

Probabilistic cellular automata, also called the stochastic cellular automata, have probabilistic updating rules, and are helpful for the study of dynamic systems; see Louis and Nardi [8] for details.

We have finished the exposition of cellular automata. We believe that cellular automata are viewed as the area which is one of the most interesting one relating to Artificial Life. However, we could not currently see major developments of cellular automata in relation to Artificial Life.

References

1. Byl, J.: Self-reproduction in small cellular automata. Phys. D **34**, 295–299 (1989)
2. Chaudhuri, P., et al.: Additive Cellular Automata Theory and Applications, vol. 1. IEEE Press, Piscataway, NJ (1997)
3. Gardner, M.: Mathematical games. Sci. Am. **223**, 120–123 (1970)
4. Gardner, M.: Mathematical games. Sci. Am. **224**, 112–117 (1971)
5. Codd, F.: Cellular Automata. Academic, New York (1968)
6. Langton, C.: Self-reproduction in cellular automata. Phys. D **10**, 135–144 (1984)
7. Langton, C.: Studying artificial life with cellular automata. Phys. D **22**, 120–149 (1986)
8. Louis, P.-Y., Nardi, F.: Probabilistic Cellular Automata. Springer, Cham (2018)
9. Reggia, J., Armentrout, S., Chou, H., Peng, Y.: Simple system that exhibits self-directed replication. Science **259**, 1282–1287 (1993)
10. Sarkar, P.: A brief history of cellular automata. ACM Comput. Surv. **32**, 80–107 (2000)
11. Sierpinski, W.: On a curve every point of which is a point of ramification (Polish). Prace Mat. Fiz. **27**, 77–86 (1916)
12. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana (1966)
13. Wolfram, S.: Statistical mechanics of cellular automata. Rev. Mod. Phys. **55**, 601–644 (1983)
14. Wolfram, S.: Universality and complexity in cellular automata. Phys. D **10**, 1–35 (1984)
15. Wolfram, S.: Theory and Applications of Cellular Automata. World Scientific, New York (1986)
16. Wolfram, S.: Random sequence generator by cellular automata. Adv. Appl. Math. **7**, 123–169 (1986)

Chapter 3
Neural Networks

Abstract This chapter describes neural networks, which provide the theoretical basis for machine learning. Neural networks are computational systems based on the biological neural networks. After explaining their basic idea, we introduce perceptrons, back propagation networks, Hopfield networks, Boltzmann machines, and Parallel Distributed Processing Models.

3.1 Ideas of Neural Networks

Neural Networks are computational systems based on the biological neural networks. Neural Networks mimic the complex functions of the human brain. They consist of interconnected nodes or neurons that process and learn from data. In the field of Artificial Intelligence (AI), neural networks have been studied as a model alternative to symbolic processing using logic, etc., and are particularly applied to *machine learning*.

3.1.1 What Are Neural Networks?

Neural networks are sometimes simply called neural nets. They are also known as *artificial neural networks* (ANNs). In the 1980s, they are frequently referred to as *connectionism* or *Parallel Distributed Processing* (PDP).

A computer based on a neural network is called *neuro computer*. Neural networks are also attracting attention in the field of AL because they simulate the functions of the human brain. In fact, the brain is considered as an important part of life, and neural networks themselves can be interpreted as AL.

A neural network consists of components which include neurons, connections, weights, biases, propagation functions, and a learning rule. Neurons receive inputs, governed by thresholds and activation functions. Connections involve weights and biases regulating information transfer. Learning, adjusting weights and biases, occurs

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Akama, *Artificial Life*, Intelligent Systems Reference Library 262,
https://doi.org/10.1007/978-3-031-74422-8_3

in three stages: input computation, output generation, and iterative refinement enhancing the network's proficiency in diverse tasks.

From the similarities with those found in human information processing, neural networks have the following advantages. These features are clearly lacking in symbolic theory.

- Adaptive learning
- Self-organization
- Real time computation
- Fault tolerance.

Adaptive learning is learning something from given data. Living things can in fact learn various things from information from the external world. Adaptive learning has something to do with e-learning, personalized education, etc.

Self-organization means the ability to organize themselves from the information acquired during learning. Living things can organize themselves by using external signals.

Real-time computation is the ability to perform high-speed computation using parallelism. Real-time computation has to be done in a time frame that is relatively imperceptible to the user.

By contrast, other types of computation can be done on a delayed basis. Humans are generally able to make inferences instantaneously from given information.

Fault tolerance is the ability to calculate as a whole even if a problem occurs in part of the network. It is one of capabilities of a computer system, electronic system or network to deliver uninterrupted service, despite one or more of its components failing.

Humans can think even if they are injured. From these advantages, it can be seen that neural networks perform calculations in a manner that is very close to human information processing.

3.1.2 History of Neural Networks

Next, we simply introduce the (long) history of neural networks. In 1943, McCulloch and Pitts proposed a logical calculus of the ideas in nervous activity in [5], considering how the human brain could produce complex patterns through neurons. Their model mimics the functionality of a biological neuron. Thus, there is a sense in which it is a precursor of neural networks.

The significance of their logical calculus consists in comparing of neurons with a binary threshold to propositional (boolean) logic. Therefore, we can see that their work can be seen as a pioneer of the origins of neural networks although it has not been fully appreciated.

In 1958, the first neural network called the *perceptron* was developed by Rosenblatt [6]. Furthering the ideas of McCulloch and Pitts's work, Rosenblatt introduced weights to the equation into his perceptron.

Rosenblatt said:

> By the study of systems such as the perceptron, it is hoped that those fundamental laws of organization which are common to all information handling systems, machines and men included, may eventually be understood. (p. 407)

He implemented a program using perceptron for learning how to distinguish cards marked on the left versus cards marked on the right. His remarks above suggest applicability neural networks for AL.

In 1969, Minsky and Papert pointed out that simple perceptrons cannot discriminate linearly inseparable patterns; see Minsky and Papert [5]. Minsky and Papert showed that a perceptron can't solve the XOR problem. With their work, the boom in neural networks died down for a while.

In 1982, Hopfield proposed *Hopfield network* based on associative memory; see Hopfield [2]. Hopfield networks were the first instance of associative neural networks capable of producing an emergent associative memory. With the appearance of the Hopfield network, neural networks came to be re-evaluated.

In 1985, Ackley, Hinton, and Sejnowski proposed a *Boltzmann machine*, which is used to model probability distributions; See Ackley, Hinton, Sejnowski [1].

In 1986, Rumelhart and McClelland proposed *parallel distributed processing model* and solved the problem of Minsky and Papert's simple perceptron; see Rumelhart and McClelland [7]. Since then, neural networks have been studied as an important field of AI and related areas.

In the 1990s, due to many applications, neural networks saw a boom. And further developments have been expected.

In the 2000s, we can handle large datasets and innovative structures, and computers enhance computational power. With the advent of deep learning, the effectivenesses of neural network have been appreciated.

We turn to an overview of the human brain. The brain is an organ that coordinates nervous system function in the human and most invertebrate animals, It is usually located inside the head. It directs our body's internal functions.

It also integrates sensory impulses and information to form perceptions, thoughts, and memories. Therefore, the various parts of the brain are responsible for personality, movement, breathing, and other crucial processes for keeping people alive.

The brain is primarily made up of *neurons*, which send nerve impulses and store information. Innumerable neurons are connected by *synapses*. If neurons are excitable, then an electric current flows through the nerve fiber, and the signal is transmitted to the other neuron by the synapse. By receiving an electric current. neurons perform operations and transmit the resulting information to other neurons.

Neurons are considered as information messengers. They use electrical and chemical signals to send information between different areas of the brain, as well as between the brain, the spinal cord, and the entire body. Everything we think, feel, and do would be impossible without the work of neurons and their support cells. Therefore, in the human brain, many neurons interact to acquire knowledge and solve problems. A neural network imitates such information processing, and can realize massively parallel distributed processing.

A neuron in neural networks is described as a mathematical model (or function), which roughly several accepts inputs and gives relevant outputs by means of intended computing.

A neural network typically consists of *layers* (or nodes) of neurons. They are of the following three types:

1. Input layer
2. Hidden layer
3. Output layer.

Input layer is single, *hidden layer* is one or more than one, and *output layer* is single. Each node connects to another and is characterized by an associated weight and threshold.

If the output of any individual node is above the specified threshold value, then the node gets activated, and sends data to the next layer of the network. Otherwise, no data is transmitted to the next layer of the network. Thus, a typical neural network has the following structure (Fig. 3.1).

Here we explain the roles of these three layers. Input layer accepts the input data and passes it to the first hidden layer. It does not perform any transformations on the data.

Hidden layers are layers of mathematical functions that each designed to produce an output specific to an intended result. Output layer produces the final output.

Several classifications of types of neural networks may be found. For example, based on the direction of information flow, there are two sorts of neural networks, i.e., feedforward neural networks and feedback neural networks. In the sequel, we explain some of famous neural networks mentioned above.

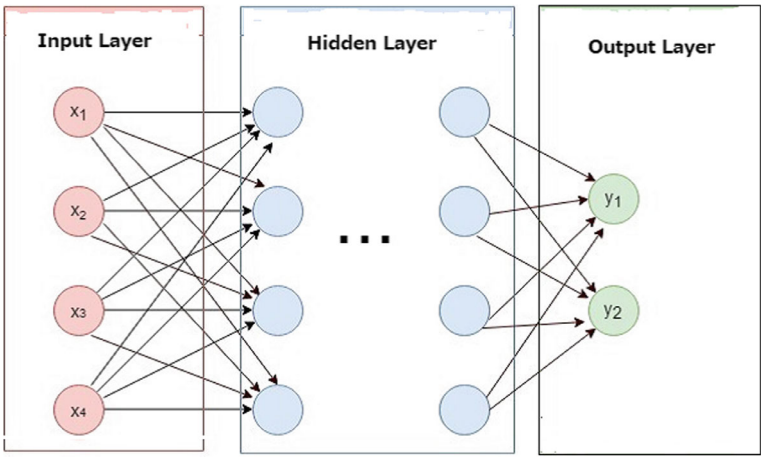


Fig. 3.1 Structure of neural network

3.1.3 Advantages and Disadvantages

Neural networks have both advantages and disadvantages. There are advantages including adaptability, pattern recognition, parallel processing, non-linearity. For adaptability, neural networks are useful for activities in which the link between inputs and outputs is complex or not well defined.

For pattern recognition, neural networks reveals the efficacious in tasks. Thus, they can also be applied to audio and image identification, natural language processing, and other intricate data patterns.

For non-linearity, neural networks can model and understand complicated relationships in data by using the non-linear activation functions in neurons, which overcome the drawbacks of linear models.

There are disadvantages including computational cost, black boxness, overfitting, and need for large datasets. For computational cost, large neural network training can be a hard process that demands a lot of computing power.

For black boxness, neural networks have a problem for important applications because it is difficult to understand how they make decisions.

For overfitting, neural networks commit training material to memory rather than identifying patterns in the data. Although regularization approaches help to alleviate this, the problem can still be found.

For the need of large datasets, neural networks frequently need sizable, labeled datasets for efficient training; otherwise, their performance may suffer from incomplete or skewed data.

3.2 Perceptrons

Perceptron is an early neural network proposed by Rosenblatt in 1958; see Rosenblatt [6]. It should be noted that perceptron is applied to pattern recognition, etc., and plays an essential role in Machine Learning.

Rosenblatt who is American psychologist and was inspired by the biological neuron and its ability to learn. Rosenblatt's perceptron consists of one or more inputs, a processor, and only one output.

Perceptron is used to classify data, or as an algorithm to simplify or supervise the learning capabilities of binary classifiers. It is clear that perceptron is one of the important methods for AL.

Rosenblatt says:

> The perceptron is designed to illustrate some of the fundamental properties of intelligent systems in general, without becoming too deeply enmeshed in the special, and frequently unknown, conditions which hold for particular biological organisms. The analogy between the perceptron and biological systems should be readily apparent to the reader. (p. 387)

Rosenblatt's purpose was to create a physical machine that behaves like a neuron. Its first implementation was a software that had been tested on the IBM704. Later. he implemented its software on other platforms for image recognition.

Rosenblatt gave the following three questions about perceptual recognition; (1) How is information about the physical world sensed, or detected, by the biological system? (2) In what form is information stored, or remembered? (3) How does information contained in storage, or in memory, influence recognition and behavior? We believe that his perceptron was the result which is the answer to these three fundamental questions.

Rosenblatt's main conclusions are as follows. First, in an environment of random stimuli, a system consisting of randomly connected units, subject to the parametric constraints can learn to associate specific responses to specific stimuli.

Second, in an "ideal environment", the probability of a correct response diminishes towards its original random level as the number of stimuli learned increases. Third, in such an environment, no basis for generalization exists.

Fourth, the asymptote can be made arbitrarily close to unity by increasing the number of association cells in the system. Fifth, in the differentiated environment, the probability that a stimulus which has not been seen before will be correctly recognized and associated to its appropriate class.

Sixth, the performance of the system can be improved by the use of a contour-sensitive projection area, and by the use of a binary response system.

Seventh, trial-and-error learning is possible in bivalent reinforcement systems. Eighth, temporal organizations of both stimulus patterns and responses can be learned by a system.

Nineth, the memory of the perceptron is distributed, Tenth, simple cognitive sets, selective recall, and spontaneous recognition of the classes present in a given environment are possible.

3.2.1 Basics of Perceptron

Perceptron is a hierarchical neural network composed of *S layer*, *A layer*, and *R layer*. Each layer consists of multiple units that are unidirectionally connected (Fig. 3.2).

Here, S layer is the sensory layer, A layer is the association layer by the threshold function, and R layer is the response layer, respectively. Each layer is composed of several units (circled portions).

However, there is no connection between units within the same layer. The input and output of each unit of Perceptron is "0" or "1". Inputs flow only in the order of S layer, A layer, and R layer.

S layer, the output is the input. In A layer and R layer, the input is the weighted values of the connections of outputs of the previous layers, and the output is obtained by the sum by threshold function.

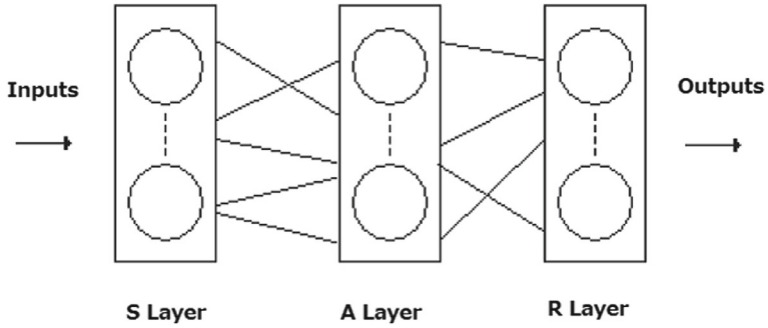


Fig. 3.2 Structure of perceptron

Note that the weight between S layer and A layer is fixed, but the weight between A layer and R layer is variable. In R layer, the outputs of each unit is considered to be outputs of Perceptron.

Now, we give a formal exposition of Perceptron. As Rosenblatt's original exposition is biologically and complicated, we give an exposition based on standard mathematics.

The perceptron is formally defined by

$$y = \text{sign}\left(\sum_{i=1}^{N} w_i x_i - \theta\right) \tag{3.1}$$

where w is the weight vector and θ is the *threshold* defined by as follows:

$$f(x) = \begin{cases} 1 & (x \geq 0) \\ 0 & (x < 0) \end{cases}$$

We turn to the algorithm of Perceptron denoted PLA. Given a set of input vectors $\{x^1, ..., x^P\}$ and a set of desired labels $\{y_0^1, ..., y_0^P\}$, PLA starts from some initial weight vector, modifies on the basis of the examples ultimately yielding set of weights w that classify correctly all the examples,

$$\forall \mu(\text{sign}(w^T, x^\mu) \tag{3.2}$$

T is a transposition. Note that PLA is incremental.

For PLA, examples are given one by one at each time step, and a weight update rule is applied. Once all examples are given the algorithms cycles again through all examples, until convergence.

PLA begins with an arbitrary set of weights w^0. At each subsequent step, i.e. the n th step, we have the weights arrived at in the previous step, i.e., w^{n-1}, and a new example (x^n, y_0^n). Then, the weights are then updated by the following rules: If

$$y_0^n({w^{n-1}}^T x^n) > 0 \tag{3.3}$$

which shows that the present weight vector classifies correctly the new example. Then,

$$w^n = w^{n-1} \tag{3.4}$$

and the algorithm moves to the next step. On the other hand, if

$$y_0^n ({}^{n-1}{}^T x^n < 0 \tag{3.5}$$

showing that error has occurred, then,

$$w^n = w^{n-1} + \eta y_0^n x^n \tag{3.6}$$

The algorithm halts when all the examples are classified correctly by w. If there was no error, then the algorithm halts, and all the training examples are classified correctly. Whether the algorithm can find w realizing the task depends on whether the data is linearly separable.

Here, the question arises: Does the perceptron algorithm will converge or not? Fortunately, this is ensured by *Perceptron Convergence Theorem* which says that For any finite set of linearly separable labeled examples, PLA will halt after a finite number of iterations.

3.2.2 Perceptron and Machine Learning

As said before, the most effective application of neural networks is it machine learning and there are several classifications. On one classification, it is divided into supervised learning and unsupervised learning.

Supervised learning is a method of machine learning, which is defined by its use of labeled datasets to train algorithms classifying data or predicting outcomes accurately. On the other hand, *unsupervised learning* is a method of machine learning, which analyzes and clusters unlabeled datasets.

It is obvious that Perceptron is considered to be a model for supervised learning. This is because the input becomes the training pattern, the output becomes the teacher pattern, and the weight of the connection that converts the input to the output is automatically calculated.

Perceptrons were originally interesting since they demonstrated how a system could learn to recognize or categorize patterns. They are later applied a number of problems because there are many complex pattern recognition problems that they cannot learn to solve, and this motivated the development of new Connectionist networks described below.

3.3 Back Propagation Networks

Back propagation networks are considered to be hierarchical neural networks, which are proposed by Rumelhart et al. They are regarded as an improvement of Perceptron; see Rumelhart and McClelland [7] in that they utilize three or more layers and that weights of all connections are learnable.

3.3.1 Basics of Back Propagation Networks

Roughly speaking, Perceptron models digital-type cells, while back propagation networks model analog-type cells. Rumelhart et al. formalized back propagation networks in which continuous functions with monotonically are allowed.

By using continuous functions, their learning ability is superior to perceptrons based on discrete functions. In the sequel, we review back propagation networks.

We now fix the notation. If a particular layer has $J \in \mathbf{N}$ nodes in it (where $\mathbf{N}$ is the set of all natural numbers), then we refer to an arbitrary node in that layer as the jth node, where $j \in \{0, 1, ..., J\}$. Similarly, the ith node is in a layer with I nodes, where each layer makes use of its own index variable.

Here, two remarks. The layers are (not necessarily) labeled in reverse alphabetical order. Because the layers are not in general fully connected, the nodes from layer k which innervate the jth node of layer j will in general be only a subset of the K nodes which belong to the kth layer. We denote this subset by K_j.

3.3.2 McCulloch-Pitts Neuron

We here introduce a single McCulloch-Pitts (MP) neuron (cf. McCulloch and Pitts [5]), transforming the weighted sum of its inputs via a function, usually non-linear, into an activation level, i.e., the output. The neurons consist of two parts, the weighted sum and the activation function.

Observe that all MP neurons perform the weighted sum of the inputs but activation functions vary. The weighted sum is

$$x_j = \sum_{k \in K_j} w_{kj} y_k, \tag{3.7}$$

where K_j is the set of nodes from the kth layer which feed node j, and the activation is the function of the form:

$$y_j = f(x_j) \tag{3.8}$$

We take up four (possible) activation functions; linear

$$f(z) = \beta z \tag{3.9}$$

threshold (used also for Perceptron),

$$f(z) = \begin{cases} 1 \ (z \geq \theta) \\ 0 \ (z < \theta) \end{cases} \tag{3.10}$$

sigmoid

$$\frac{1}{1 + e^{-\gamma z}} \tag{3.11}$$

and radial basis

$$f(z) = \exp\left(-\frac{(z-\mu)^2}{\sigma^2}\right). \tag{3.12}$$

Here, β, θ, γ, and μ are free parameters controlling the shape of the function. For back propagation networks, sigmoid function is frequently used.

We now describe the algorithm of back propagation networks. For a weight connecting a node in layer k to a node in layer j, the change in weight is given by

$$\Delta w_{kj}(n) = \alpha \Delta_j y_k + \eta \Delta w_{kj}(n-1) \tag{3.13}$$

where α is the learning rate (a real value on (0, 1], y_k is the activation of the node in layer k, n and $n-1$ is the iteration through the loop, η is the momentum (a real value on (0, 1], δ_j is the error term associated with the node after the weight.

If j is the output layer, then

$$\delta_j := (t_j - y_j) y_j (1 - y_j). \tag{3.14}$$

If j is a hidden layer, then

$$\delta_j := \left(\sum_{i \in I_j} \delta_i w_{ji}\right) y_j (1 - y_j). \tag{3.15}$$

Computation of the error terms must proceed backwards through the network, beginning with the output layer and terminating with the first hidden layer. In this way, changes of weights are repeated and if errors are less than a certain value, then the algorithm ends. The name of back propagation networks is after such a process described.

We can claim that back propagation networks have several advantages over Perceptrons. However, they rely on complicated notions in calculus and may have some difficulties with their implementation.

3.4 Hopfield Networks

Hopfield Networks (HPN) consist of one layer of n fully connected recurrent neurons. HPN is generally used in performing automatic association and optimization tasks. HPN is calculated by means of a converging interactive process and it generates a different response than our normal neural networks; see Hopfield [2].

HPN can serve as a model for human memory. It realizes the connection with the so-called *associative memory*.

Hopfield said:

> This paper shows that the important properties of the original model remain intact when these two simplifications of the modeling are eliminated. Although it is uncertain whether the properties of these new continuous "neurons" are yet close enough to the essential properties of real neurons (and/or their dendritic arborization) to be directly applicable to neurobiology, a major conceptual obstacle has been eliminated. (p.3088)

3.4.1 Basics of Hopfield Networks

Hopfield studied a model for a large network of "neurons" with a graded response (or sigmoid input-output relation). He adopted *content-addressable memory* (CAM) as one of the simplest collective properties of a system that is highly interconnected neurons.

Hopfield's original model used two-state threshold "neurons" that followed a stochastic algorithm. Each model neuron i has two states, characterized by the output V_i of the neuron having the values V_i^0 or V_i^1 (which may often be taken as 0 and 1, respectively).

The input of each neuron came from two sources, external inputs I_i and inputs from other neurons. The total input to neuron i is then

$$\text{Input to } i = H_i = \sum_{j \neq i} T_{ij} V_j + I_i. \tag{3.16}$$

The element T_{ij} can be biologically viewed as a description of the synaptic interconnection strength from neuron j to neuron i.

CAM in this system involves the change of state of the system with time. A model must describe how the state evolves in time, and the original model describes this in terms of a stochastic evolution.

Each neuron samples its input at random times. It changes the value of its output or leaves it fixed according to a threshold rule with thresholds U_i.

$$\begin{aligned} V_i &\rightarrow V_i^0 \text{ if } \textstyle\sum_{j\neq i} T_{ij}V_j + I_i < 0 \\ &\rightarrow V_i^1 \text{ if } \textstyle\sum_{j\neq i} T_{ij}V_j + I_i > 0 \end{aligned} \tag{3.17}$$

The interrogation of each neuron is a stochastic process, taking place at a mean rate W for each neuron. The times of interrogation of each neuron are independent of the times at which other neurons are interrogated. Thus, the algorithm is asynchronous, in contrast to the usual kind of processing done with threshold devices.

3.4.2 Associative Memory

Associative memory refers to the ability to remember relationships between concepts, and not just the individual concepts themselves. We can recognize such relationships by visual and verbal information.

For example, we can relate two words like man and woman by verbally. We can also relate an object and its alternative name like a violin visually.

The original HPN model behaves as an associative memory when the state space flow generated by the algorithm is characterized by a set of stable fixed points. If the location of a particular stable point in state space is thought of as the information of a particular memory of the system, states near to that particular stable point contain partial information about that memory.

From an initial state of partial information about a memory, a final stable state with all the information of the memory can be found.

The essential feature of CAM operation is convergent flow to stable states. For instance, consider

$$E = -\frac{1}{2}\sum\sum_{i\neq j} T_{ij}V_iV_j - \sum_i I_iV_i * \sum_i U_iV_i. \tag{3.18}$$

The change ΔE in E due to changing the state of neuron i by ΔV_i is For neurons exhibiting action potentials, it could be thought of as the mean soma potential of a neuron from the total effect of its excitatory and inhibitory inputs. V_i can be viewed as the short term average of the firing rate of the cell i.

$$\Delta E = -\left[\sum_{i\neg j} T_{ij}V_j + I_i - U_i\right]\Delta V_i. \tag{3.19}$$

But according to the algorithm, ΔV_i is positive only when the bracket is positive, and similarly for the negative case. Thus any change in E under the algorithm is

negative. E is bounded, so the iteration of the algorithm must lead to stable states that do not further change with time.

For neurons exhibiting action potentials, u_i could be thought of as the mean soma potential of a neuron from the total effect of its excitatory and inhibitory inputs. V_i can be viewed as the short term average of the firing rate of the cell i.

The typical input-output relation $g_i(u_i)$ is sigmoid with asymptotes V_i^0 and V_i^1. For neurons exhibiting action potentials, u_i could be thought of as the mean soma potential of a neuron from the total effect of its excitatory and inhibitory inputs. V_i can be viewed as the short term average of the firing rate of the cell i.

In terms of electrical circuits, $g_i(u_i)$ represents the input-output characteristic of a nonlinear amplifier with negligible response time. It is convenient also to define the inverse output-input relation, $g_i^{-1}(V)$.

In a biological system, u_i will lag behind the instantaneous outputs V_j of the other cells because of the input capacitance C of the cell membranes, the transmembrane resistance R, and the finite impedance T_{ij}^{-1} between the output V_j and the cell body of cell i. Thus there is a resistance-capacitance (RC) charging equation that determines the rate of change of u_i.

$$C_i(du_i/dt) = \sum_i T_{ij}V_j - u_i/R_i + I_i, \quad u_i = g_i^{-1}(V_i). \tag{3.20}$$

Here, $T_{ij}V_j$ represents the electrical current input to cell i due to the present potential of cell j, and T_{ij} is thus the synapse efficacy. Linear summing of inputs is assumed. T_{ij} of both signs should occur. Ii is any other (fixed) input current to neuron i.

The same set of equations represents the resistively connected network of electrical amplifiers. It appears more complicated than the description of the neural system because the electrical problem of providing inhibition and excitation requires an additional inverting amplifier and a negative signal wire.

The magnitude of T_{ij} is $1/R_{ij}$, where R is the resistor connecting the output of j to the input line i, while the sign of T_{ij} is determined by the choice of the positive or negative output of amplifier j at the connection site. R_i is now

$$1/R_i = \rho_i + \sum_i 1/R_{ij}, \tag{3.21}$$

where ρ_i is the input resistance of amplifier i. C_i is the total input capacitance of the amplifier i and its associated input lead. We presume the output impedance of the amplifiers is negligible.

Consider the quantity

$$E = -\frac{1}{2}\sum\sum_{i,j} T_{ij}V_iV_j + \sum_i (1/R_i)\int_0^{V_i} g_i^{-1}(V)dV + \sum_i I_iV_i. \tag{3.22}$$

Its time derivative for a symmetric T is

$$dE/dt = -\sum_i dV_i/dt \left(\sum_i T_{ij} V_j - u_i/R_i + I_i \right). \tag{3.23}$$

The parenthesis is the right-hand side of the first of (3.20), so

$$dE/dt = -\sum C_i (dV_i/dt)(du_i/dt) = -\sum C_i g_i^{-1'}(V_i)(dV_i/dt)^2. \tag{3.24}$$

Since $g_i^{-1}(V_i)$ is a monotone increasing function and C_i is positive, each term in this sum is non-negative. Therefore

$$dE/dt \leq 0,\ dE/dt = 0 \rightarrow dV_i/dt = 0\ for\ all\ i. \tag{3.25}$$

Together with the boundedness of E, (3.25) shows that the time evolution of the system is a motion in state space that seeks out minima in E and comes to a stop at such points. E is a Lyapunov function for the system.

This deterministic model has the same flow properties in its continuous space that the stochastic model does in its discrete space. Hopfield also discusses the relation between the stable states of the two models. This completes the presentation of the two models for Hopfield networks.

As said above, HPN is based on a model for human memory. Associative memory is the ability to remember relationships between concepts, and not just the individual concepts themselves. Correlation associative memory and auto associative memory belong to associative memory, and HPN can deal with the latter.

However, there are tow types of associative memory; implicit and explicit. Implicit associative memory is an unconscious process relying on priming, whereas explicit associative memory involves conscious recollection. To handle these two kinds of associative memory in HPN, we may need some extensions.

3.5 Boltzmann Machines

A *Boltzmann machine* (BM) is a neural network symmetrically connected, neuron-like units as proposed by Ackley, Hinton and Sejnowski [1]. BMs can learn the probability density from the input data to generating new samples from the same distribution.

They said:

> We believe that the Boltzmann Machine is a simple example of a class of interesting stochastic models that exploit the close relationship between Boltzmann distributions and information theory. (p. 160)

3.5.1 Basics of Boltzmann Machines

BM is composed of primitive computing elements called units that are connected to each other by bidirectional links. A *unit* is always in one of two states, on or off. The structure of BM is related to HPN.

A unit adopts these states as a probabilistic function of the states of its neighboring units and the weights on its links to them. The weights can take on real values of either sign.

The global state of the BM can be assigned a single number called the energy of the state. With the right assumptions, the individual units can be made to act so as to minimize the global energy.

The energy of a global configuration is defined as

$$E = -\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i \tag{3.26}$$

where w_{ij} is the strength of connection between units i and j, s_i is 1 if unit i is on and 0 otherwise, and θ_i is a threshold.

A simple algorithm for finding a combination of truth vales that is local minimum is to switch each hypothesis into whichever of its two states yields the lower total energy given the current states of the other hypotheses.

Because the connections are symmetric, the difference between the energy of the whole system with the kth hypothesis rejected and its energy with the kth hypothesis accepted can be determined locally by the kth unit, and this energy gap is just

$$\Delta E_k = \sum_i w_{ki} s_i - \theta_k \tag{3.27}$$

Therefore, the rule for minimizing the energy contributed by a unit is to adopt the *on* state if its total input from the other units and from outside the system exceeds its threshold. The threshold terms can be eliminated from (3.26) and (3.27). Thus these can be written as:

$$E = -\sum_{i<j} w_{ij} s_i s_j \tag{3.28}$$

$$\Delta E_k = \sum_i w_{ki} s_i \tag{3.29}$$

If the energy gap between the on and off states of the kth unit is ΔE, then regardless of the previous state set $s_k = 1$ with probability

$$p_k = \frac{1}{1 + e^{-\Delta E_k / T}} \tag{3.30}$$

where T is a parameter that acts like temperature. The decision rule in (3.30) is the same as that for a particle which has two energy states.

A system of such particles in contact with a heat bath at a given temperature will eventually reach thermal equilibrium and the probability of finding the system in any global state will then obey a *Boltzmann distribution*. BM is named after this distribution.

Recall the Boltzmann distribution (BD) is often used in physics. It is written as

$$p_i \propto e^{-E_i/kt}$$

where p_i is the probability of the system in state i, E_i is the energy of the state i, k is the Boltzmann constant, and T is the temperature, Thus, BD gives the probability of a system in a certain state with an energy and a temperature.

Similarly, a network of units obeying this decision rule will eventually reach "thermal equilibrium" and the relative probability of two global states will follow the Boltzmann distribution:

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T} \tag{3.31}$$

where P_α is the probability of being in the αth state and E_{alpha} is the energy of that state.

By using BD, it implies that the difference in the log probabilities of two global states is just their energy difference (at a temperature 1) from (3.31).

A learning algorithm for BM can achieve a domain-independent learning algorithm that modifies the connection strengths between units. The units of BM are divided into a non-empty set of visible units and a possibly empty set of hidden units. The visible units are the interface between the network and the environment. The hidden units are no always necessary and can be used to explain various constraints.

An information-theoretic measure of the discrepancy between the network's internal model and the environment is

$$G = \sum_\alpha P(V_\alpha) \ln \frac{P(V_\alpha)}{P'(V_\alpha)} \tag{3.32}$$

where $P(V_\alpha)$ is the probability of the αth state of the visible units when their states are determined by the environment, and $P'(V_\alpha)$ is the corresponding probability when the network is running freely with no environment unit.

The partial derivative of G is

$$\frac{\partial G}{\partial w_{ij}} - \frac{1}{T}(p_{ij} - p'_{ij}) \tag{3.33}$$

where p_{ij} is the average probability of two units both being in the *on* state when the environment is clamping the states of visible, and P'_{ij} s the corresponding probability

when the environmental input is not present and the network is running freely. Both these probabilities must be measured at equilibrium.

To minimize G, it is therefore sufficient to observe p_{ij} and p'_{ij} when the network is at thermal equilibrium, and to change each weight by an amount proportional to the difference between these two probabilities:

$$\Delta w_{ij} = \varepsilon(p_{ij} - p'_{ij}) \tag{3.34}$$

where ε scales the size of each weight change.

3.5.2 Advantages of Boltzmann Machines

BM has advantages over Hopfield network by introducing the notions of energy and temperature. BM can learn the patterns and correlations inside the data. Notice that BM can be used for both supervised and unsupervised learning. However, it is known that learning in BM is very slow with many hidden layers.

3.6 Parallel Distributed Processing Models

In 1986, Rumelhart and McClelland proposed *Parallel Distributed Processing (PDP) Models* which are a class of neurally inspired information processing models attempting to model information processing actually taken place in the brain; see Rumelhart and McClelland [7].

A PDP model was developed by the observation that neural connections are appeared in a parallel way. In fact, different types of mental processing are interpreted as distributed throughout a highly complex neural network.

3.6.1 Basics of Parallel Distributed Processing Models

There are three feature of the PDP model. First, its representation of information is *distributed* rather than local. Second, memory and knowledge for specific things are not stored explicitly, but stored in the connections between units. Third, learning can be done with gradual changes by experience.

A PDP system consists of a large number of neuron-like computing elements called *units*. Each unit can have an activation value between some maximum and minimum values, e.g., 1 and 0. In such systems, the representation of something that we are currently thinking about is a pattern of activation over the computing elements.

Processing is done by the propagation of activation from one unit to another by connections among the units. A connection may be excitatory (positive-valued) or inhibitory (negative-valued).

If the connection from one unit to another is excitatory, then the activation of the receiving unit increases whenever the sending unit is active. If the connection is inhibitory, then the activation of the receiving unit decreases.

In such a system, the knowledge that governs processing is stored in the connections among the units, for it is these connections that determine what pattern will result from the presentation of an input. Learning is done by adjustments of connection strengths. Memory storage is just a form of learning, and also occurs by connection weight adjustment.

3.6.2 Simple Parallel Distributed Processing Model

A simple PDP network consisting of one pool of units, and one projection, such that each unit receives connections from all other units in the same pool. Each unit also can receive external input (shown coming in from the left).

If this were a pool in a larger network, the units could receive additional projections from other pools (not shown) and could project to units in other pools as illustrated by the arrows proceeding out of the units to the right as shown by Fig. 3.3 from [4].

We finish the concise presentation of neural networks by giving the outline of major theories. These are used for many applications.

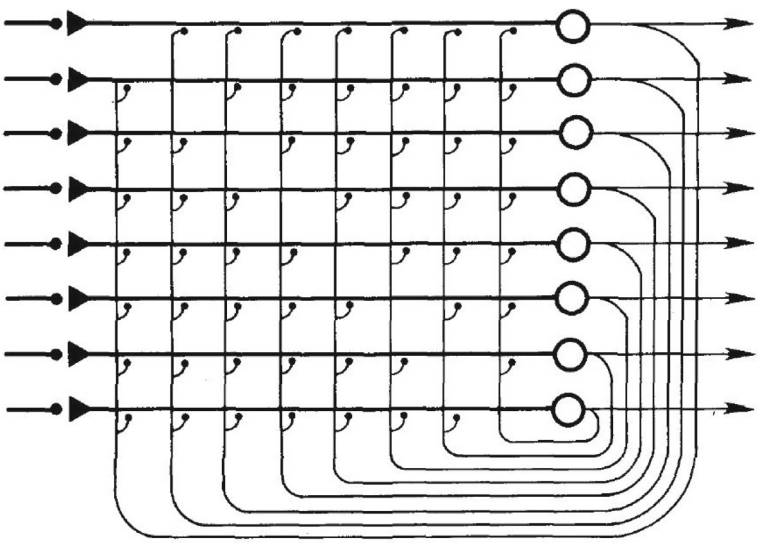


Fig. 3.3 A simple PDP network

3.7 Deep Learning

3.7.1 What Is Deep Learning?

Recently, neural networks become to play important roles in AI. In particular, the so-called *deep learning* led to great success of practical AI systems called the generative AI.

The first such a system is famous ChatGPT, and similar systems have been later developed. We thus think that some words on deep learning are needed here.

Deep learning is a family of methods rather than theories. It is a branch of machine learning which is based on neural networks. It can learn complex patterns and relationships within data.

In deep learning, we do not need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets.

Because it is based on neural networks which are inspired by the structure and function of the human brain's biological neurons, it is designed to learn from large amounts of data.

The key feature of deep learning is the use of deep neural networks, which have multiple layers of interconnected nodes. These networks can thus learn complex representations of data by suitably discovering hierarchical patterns and features in data.

Therefore, deep learning algorithms can automatically learn and improve from data without the need for manual. The fact is a merit for developing practical AI systems.

Although both machine learning and deep learning are methods of AI, there are many similarities and differences between them. Machine learning uses statistical algorithms to learn the hidden patterns and relationships in the dataset.

It can work on the smaller amount of dataset. It is thus better for the low-label task, and takes less time to train the model.

In contrast, deep learning uses neural network architecture to learn the hidden patterns and relationships in the dataset. This means that it requires the larger amount of dataset compared to machine learning.

It is thus better for complex task like image processing, natural language processing, etc. But, it need to take more time to train the model.

3.7.2 Advantages and Disadvantages

Deep learning has also both advantages and disadvantages. This means that deep learning is our final answer. We must correctly evaluate them, and work out further developments for intended applications.

First, it has high accuracy. Deep learning algorithms can achieve high performance in various tasks. The promising applications include image recognition and natural language processing.

Second, it ensures automated feature engineering. For deep Learning algorithms can automatically discover and learn relevant features from data without the need for manual feature engineering.

Third, it has scalability. Deep learning can scale to handle large and complex datasets, and can learn from large of data. In some sense, the point is useful.

Fourth, is it flexible. Deep learning models can be applied to a variety of tasks and can handle various types of data including images, text, and speech. The flexibility may be crucial.

Fifth, it can be improved. Deep learning models can continually improve their performance by using more available. Thus, such improvements are of special importance.

As for disadvantages of deep learning, we can point out some points. These reveal the limitation of deep learning, and we must carefully consider them.

Firstly, high computational requirements are needed. In fact, deep learning requires large amounts of data and computational resources to train and optimize. This reveals deep learning's crucial defect.

Secondly, it requires large labeled data. Deep learning models need a large amount of labeled data for training which can be expensive and time consuming to acquire. Therefoe, in view of possible applications. The problem should be suitably resolved.

Thirdly, it requires interpretability. Deep learning models can be challenging to interpret, but it may be difficult to understand how they make decisions. Although deep learning can give an answer, it does not proper explanation to the given problem.

Fourthly, it may be overfitting. Deep Learning models can sometimes overfit to the training data, and it may result in poor performance on new data. We need to investigate overfitting of deep learning.

Fifthly, it is blackbox-like. Deep Learning models are viewed as black boxes. Therefore, it may be difficult to understand how they work and how they arrived at appropriate predictions. The problem is closely related to the explanation of the results obtained by deep learning for various areas.

In summary, deep learning can be applied to any theory (or model) of learning using a neural network. In fact, it can provide the basis of practical AI systems for various areas. The fact is also useful for AL systems, because the ideas of deep learning are essential to our learning.

We believe that there may also be many possibilities of deep learning for AL. For example, it could improve design and implementation of DNA computers. Also observed is that it could also be of use to create new living things.

It is also noted that deep learning has some merits for quantum computing. Indeed quantum computing is theoretically attractive, but may be faced with some difficulties with an implementation. If we overcome difficulties of quantum computing, deep learning algorithms will more faster.

Based on the above discussion, a lot work remains to be seen. In fact, there are many hard problems. However, we expect to see further developments of deep learning in connection with AL.

References

1. Ackley, D., Hinton, G., Sejnowski, T.: A learning algorithm for Boltzmann machine. Cognit Sci **9**, 147–169 (1985)
2. Hopfield, J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Acad. Sci. (USA), **79**, 3088–3092 (1982)
3. Minsky, M., Papert, S.: Perceptrons. MIT Press, Cambridge, Mass., 1969, Expanded Version (1988)
4. McClelland, J., Rumelhart, D.: Distributed memory and the representation of general and specific information. J. Exp. Psychol.: General **114**, 159–197 (1985)
5. McCulloch, S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**, 115–133 (1943)
6. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**, 386–408 (1958)
7. Rumelhart, D., McClelland, J. (eds.): Parallel Distributed Processing. MIT Press, Cambridge, Mass., two volumes (1986)

Chapter 4
Evolutionary Computation

Abstract This chapter presents evolutionary computation. It is a field studied in the areas including AI and soft computing based on the ideas of biological evolution. It is mainly applied to solve complex optimization problems. After reviewing the basic ideas of evolutionary computation, we introduce Genetic Algorithm, Genetic Programming, Evolutionary Programming, and Evolutionary Strategy.

4.1 Ideas of Evolutionary Computation

4.1.1 What is Evolutionary Computation?

Evolutionary Computation (EC) is a computational method that applies the *evolution* of living organisms. Although evolutionary computation has traditionally been studied for optimization problems in areas like AI.

However, since evolution is based on the basic functions of living organisms, it has recently attracted attention in the field of AL. The fact implies EC plays a crucial role for the study of AL.

In evolutionary computation, the process of natural evolution is modeled for a strategy for finding optimal (or near optimal) solutions for a given problem. For instance, in genetic algorithms explained below, candidates for a solution are encoded in a string, often a binary string only containing 0 s and 1 s. Evolution takes place by modifying the genetic code of a candidate.

Because evolutionary computation simulates natural evolution, it can deal with optimization algorithms that iteratively enhance the quality of the solution set until an optimal solution is obtained.

Many problems handled by evolutionary computation are known as combinatorial optimization problems, which are computationally hard (NP-hard). The fact may suggest that evolution of living things can be properly simulated by computers.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Akama, *Artificial Life*, Intelligent Systems Reference Library 262,
https://doi.org/10.1007/978-3-031-74422-8_4

4.1.2 Approaches to Evolutionary Computation

There are the following major theories of evolutionary computation.

- Genetic Algorithm
- Genetic Programming
- Evolutionary Programming
- Evolutionary Strategy.

Genetic Algorithm (GA) was proposed by Holland in 1975. (He studied GA since the 1960s.) It studies various algorithms for finding approximate solutions of complex problems; see Holland [5, 6].

GA is considered as one of the attractive methods used in AI. GA is in fact inspired biological evolution. Thus, its potentials should be discussed for various areas of AL.

Genetic algorithms are based on the ideas of natural selection and genetics. Thus, they are considered to be intelligent realization of random search provided with historical data to direct the search into the region of better performance in solution space.

Genetic Programming (GP), proposed by Koza in 1990, is a method of *automatic programming*; see Koza [7]. GP applies the methods of GA to program evolution to automatically generate programs. In other words, GP deals with programs as a sort of living things.

Genetic Programming is a variant of the Genetic Algorithm. GP is an evolutionary algorithm relying on genetic operators such as fitness proportionate reproduction, crossover, and mutation to drive a population of encoded programs, or individuals, by successive generations toward a solution.

Evolutionary Programming (EP) is a method for simulating evolution. EP was developed from the viewpoint of AI by Fogel since the 1960s; see Fogel, Owens and Walsh [3].

Evolutionary Programming addresses mutation, and does not use cross-over. Classical EPs have been developed to evolve finite-state automata in capable of solving time series prediction problems and more generally evolving learning machines.

Evolutionary Strategy (ES) was developed in the 1960s by Rechenberg and Schwefel; see Rechenberg [11] and Schwefel [12]. ES provides algorithms for solving the non-linear optimization problems for real functions. It was shown that GA and ES are essentially equivalent algorithms.

Unlike other evolutionary computation techniques, Evolutionary Strategy uses floating-point numbers to encode continuous variables used in different equations. For example, GA does not usually encode variables by floating-point numbers. Note that EP also uses floating-point numbers for encoding.

Before reviewing the above mentioned theories of Evolutionary Computation, we simply give an exposition of biological evolution. Evolution can be seen as the process of change in all forms of life over generations, and it is descent with inherited modification.

Therefore, evolution is helpful for us to understand our living world and its history. Many theories of evolution have been proposed, and one of the most influential theories is Darwin's theory of *natural selection* published in 1859; see Darwin [2] for details.

According to Darwin, evolution is essentially brought about by the interplay of the following three principles. The first is *mutation*, the second is *inheritance*, and the third is *selection*.

The book led many people to think that species change over time—a lot of time. It suggested that the planet was much older than what was commonly believed at the time, namely six thousand years. At the time of publication of the book, Darwin's theory was popular but controversial.

We now return to the basics of his theory. By mutation, it means changes in the information contained in genetic material. Inheritance means the genetic traits which are passed down from parent to offspring. Selection refers to the preferential survival and reproduction.

Based on the above features of living things, Darwin reached the idea of *natural selection*. It can be regarded as the process through which populations of living organisms adapt and change, and is the core of his theory.

It is also observed that since the 1950s molecular biology have had a growing influence on the theory of evolution. The influence seems to be closely related to the developments of wetware as will be discussed in Chap. 5.

Evolutionary Computation obviously relies on the notion of evolution in various ways. In addition, Evolutionary Computation and evolution are also very important to the study of AL.

4.2 Genetic Algorithm

4.2.1 What is Genetic Algorithm?

In 1975, Holland proposed *Genetic Algorithm* (GA) in Holland [6]. Since the 1960's, he investigated the mechanisms of inheritance in nature as the *adaptation system*.

GA is an algorithm which mimics Darwin's theory of natural selection, and is used to solve many problems of optimization and search. In GA, the mechanism of evolution corresponds to search of the solution of a given problem.

A *chromosome* (or a genotype) is a set of parameters which define a proposed solution of the problem to be solved by GA. A chromosome is a material with inheritance information. A practical search in GA is carried out by mutation and selection.

4.2.2 Basics of Genetic Algorithm

Now, we give the outline of GA following McCall [9]. GA's main components are the *chromosome encoding*, the *fitness function*, *selection*, *recombination* and the *evolution scheme*.

The chromosome encoding is to represent a chromosome. GA manipulates populations of chromosomes, which are string representations of solutions to a particular problem. A chromosome is an abstraction of a biological DNA chromosome. It can be viewed as a string of letters from the alphabet $\{A, C, G, T\}$.

The fitness function is a computation evaluating the quality of the chromosome as a solution to a particular problem. By analogy with biology, the chromosome is referred to as the *genotype*, whereas the solution it represents is known as the *phenotype*.

The selection component of GA uses fitness to guide the evolution of chromosomes. Chromosomes are therefore selected for recombination on the basis of fitness. There are several methods for selection, e.g. roulette wheel selection, random stochastic selection, and tournament selection.

Recombination is the process by which chromosomes selected from a source population are recombined for forming members of a successor population. It can simulate the mixing of genetic material that can occur when organisms reproduce. There are two operators, i.e., crossover and mutation operators, to carry out recombination.

Evolution scheme is to generate the successor population after recombination. Selection and recombination are then iterated until a complete successor population is produced.

Then, the successor population is replaced by a new source population, i.e., the next generation. Such iterations produce a number of generations until suitable criteria are reached.

A standard algorithm for GA can be written as follows:

(1) Randomly generate an initial source population of P chromosomes.
(2) Calculate the fitness, $F(c)$, of each chromosome c in the source population.
(3) Create an empty successor population and then repeat the following steps until P chromosomes have been created.
    (a) Using proportional fitness selection, select two chromosomes, c_1 and c_2, from the source population.
    (b) Apply one-point crossover to c_1 and c_2 with crossover rate pc to obtain a child chromosome c.
    (c) Apply uniform mutation to c with mutation rate pm to produce c'.
    (d) Add c' to the successor population.
(4) Replace the source population with the successor population.
(5) If stopping criteria have not been met, return to Step (2).

Here are some words. Genotype can be seen as a coding mechanism for giving a natural representation for the given problem. The genotype representation is decided by the evaluation and the genetic operators (Fig. 4.2).

The simplest encoding is to use binary representation in which the genotype is represented as the array of genotypes and fenotype gives integer, real number, etc. A general coding is to represnet the chromosome as a real valued vector.

Let $\mathbf{R}$ is the set of real numbers. Then,

$$\vec{X} = [x_1, ..., x_n]$$

where $x_i \in \mathbf{R}$.

The evaluation function associates a real value to a vector:

$$f : \mathbf{R}^n \rightarrow \mathbf{R}$$

Fitness function is generally steps with high time cost. To reduce cost, we may be able to use an approximate function.

For (chromosomes) selection, it should be done in such a way that best individuals have a major possibility for being parents. However, worse chromosomes should have an opportunity for reproduction. They can provide useful information for the reproduction process.

There are two operators for recombination. The crossover operator mixes genetic material from two selected parent chromosomes to produce one or two child chromosomes. We know several crossover operators.

For instance, the one point crossover due to Holland selects a random crossover point and the tails of its two parents are swapped to get new children.

In Fig. 4.1, Parents A, B are divided into 5 bits and 3 bits. For children C, D, after 6 bits are swapped.

The mutation operator changes part of genes.

Here, the underlined genes are changed.

Stopping criteria have been satisfied, GA stops. There are some criteria. One of the most important is that a solution is found. In this way, Holland specified Darwin's theory as the practical algorithm.

Fig. 4.2 shows an example of mutation.

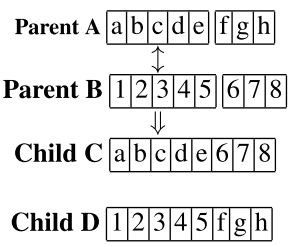


Fig. 4.1 One point crossover

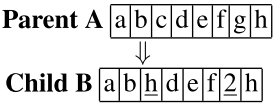


Fig. 4.2 Mutation

4.3 Genetic Programming

Genetic programming (GP), due to Koza [7, 8], is to describe a program based on evolutional computation. His Genetic Programming System was implemented by the programming language *LISP*, which is suitable for AI programming.

4.3.1 What is Genetic Programming?

Koza [7] says:

> The new "genetic programming" paradigm described herein provides a way to search for this most fit individual computer program. In this new "genetic programming" paradigm, populations of computer programs are genetically bred using the Darwinian principle of survival of the fittest and using a genetic crossover (recombination) operator appropriate for genetically mating computer programs. (p. 1)

From the above words, we can understand that Koza's aim was to develop evolutionary computation as a concrete computational system for solving various problems.

LISP is based on the idea of λ-calculus, which is a formal system for abstraction and application proposed by Church. Therefore, LISP can be classified as functional programming language.

However, LISP is suited to Koza's aim because it is appropriate for list processing. Note that LISP, developed by McCarthy, is a famous programming language for AI; see McCarthy [10].

4.3.2 LISP

Koza's GP system was implemented by Common LISP. The GP paradigm has several features. First, the structures that undergo adaptation in the GP paradigm are hierarchically structured computer programs.

Second, the search space of the GP paradigm is the hyperspace of valid LISP S-expression that can be recursively created by compositions of the available functions and available terminals for the problem.

Third, generation of the initial random population begins by selecting one of the functions from the set F = {AND, OR, NOT} at random to be the root of the tree.

Fourth, each individual in a population is assigned a fitness value as a result of its interaction with the environment.

Fifth, the two primary operations for modifying the structures undergoing adaptation are Darwinian fitness proportionate reproduction and crossover (recombination).

Sixth, the state of the hierarchical genetic algorithm system at any generation consists only of the current population of individuals in the population.

Seventh, the solution produced by this algorithm at any given time can be viewed as the entire population of disjunctive alternatives (presumably with improved overall average fitness) or, more commonly, as the single best individual in the population at that time ("winner takes all").

Eighth, the algorithm is controlled by various parameters, including two major parameters and five minor parameters.

Now, we describe the details of the GP paradigm. We assume some familiarity with LISP. The *structures* that undergo adaptation in the genetic programming paradigm are hierarchically structured computer programs whose size, shape, and complexity can dynamically change during the process.

The set of possible structures that undergo adaptation in the genetic programming paradigm is the set of all possible composition of functions that can be composed recursively from the available set of n functions $F = \{f_1, f_2, ..., f_n\}$ and the available set of m terminals $T = \{a_1, a_2, ..., a_m\}$. Each particular function f in F takes a specified number z(f) of arguments $b_1, b_2, ..., b_{z(f)}$.

The choice of functions depends on the problem and such functions include standard arithmetic operations, standard mathematical functions, Boolean operations, domain-specific functions, logical operators and iterative operators, etc. The structure can easily be written by *S-expressions* of LISP.

The *search space* created by S-expressions can be interpreted as the hyperspace of rooted point-labeled trees with ordered branches having internal points labeled with the available functions and external points (leaves) labeled with the available terminals.

The *initial structure* is the random population by selecting one of the functions from the set F at random to be the root of the tree.

The *fitness function* assigns a fitness value to each individual in a population, where the environment is defined as a set of cases which provides a basis for evaluating the fitness of the S-expressions in the population.

For most of the problems described herein, the raw fitness of any LISP S-expression is the sum of the distances (taken over all the environmental cases) between the point in the range space returned by the S-expression for a given set of arguments and the correct point in the range space.

If the S-expression is integer-valued or real-valued, the sum of distances is the sum of absolute values of the differences between the numbers involved. In particular, the raw fitness r(i, t) of an individual LISP S-expression i in the population of size M at any generational time step t is

$$r(i,t) = \sum_{j=1}^{N_e} S(i,j) - C(j)$$

where S(i,j) is the value returned by S-expression i for environmental case j (of N_e environmental cases) and where C(j) is the correct value for environmental case j.

If the S-expression is Boolean-valued or symbolic-valued, the sum of distances is equivalent to the number of mismatches. If the S-expression is complex-valued, or vector-valued, or multiple valued, the sum of the distances is the sum of the distances separately obtained from each component of the vector or list. The closer this sum of distances is to zero, the better the S-expression.

If the S-expression (or each component of a vector or list) is real-valued or integer-valued, the square root of the sum of squares of distances can, alternatively, be used to measure fitness. This means that increasing the influence of more distant points.

Next, we explain two operations for modifying the structures undergoing adaptation are Darwinian fitness proportionate reproduction and crossover (recombination).

The operation of fitness proportionate reproduction for the genetic programming paradigm is the basic engine of Darwinian reproduction and survival of the fittest. It operates on only one parental S-expression and produces only one offspring S-expression each time it is performed. Namely, it is an asexual operation.

If $f(s_i(t))$ is the fitness of individual s_i in the population at generation t, then, each time this operation is performed, each individual in the population has a probability of

$$\frac{f(s_i(t))}{\sum_{j=1}^{M} f(s_j(t))}$$

of being copied into the next generation by the operation of fitness proportionate reproduction.

The crossover (recombination) operation for the genetic programming paradigm creates variation in the population by producing offspring that combine traits from two parents. The crossover operation starts with two parental S-expressions and produces at least one offspring S-expression. Namely, it is a sexual operation.

The state of the hierarchical genetic algorithm system at any generation consists only of the current population of individuals in the population. There is no additional memory or centralized bookkeeping used in directing the adaptive process.

The algorithm can be terminated when either a specified total number of generations have been run or when some performance criterion is satisfied. In many problems, this performance requirement for termination may be that the sum of the distances reaches a value of zero.

If a solution can be recognized when it is encountered, the algorithm can be terminated at that time and the single best individual can be considered as the output of the algorithm.

We described basic ideas of evolutionary computation by focusing on GA and GP. We can find some connections of EC and AL. In particular, the so-called the *Boldwin effect* due to Boldwin [1] supports possible applications of EC to AL. The

Baldwin effect is an evolutionary mechanism which transforms a culturally invented and acquired trait into an instinctive trait by using natural selection.

The effect is about the influence on intraspecies evolution of behavior change and learning. It can be understood by the capability of an organism to be flexible and creative in adapting its behavior to a changing environment.

Hinton and Nowlan [4] insisted that Boldwin effect is seen in the evolution in GA. Namely, adaptive processes within the organism can be very effective in guiding evolution.

They said

> Learning can provide an easy evolutionary path towards co-adapted alleles in environments that have no good evolutionary path for non-learning organisms. This type of interaction between learning and evolution was first proposed by Baldwin (1896) and Lloyd Morgan (1896) and is sometimes called the Baldwin effect. Waddington (1942) proposed a similar type of interaction between developmental processes and evolution and called it "canalization" or "genetic assimilation." So far as we can tell, there have been no computer simulations or analyses of the combinatorics that demonstrate the magnitude of the effect.

As described, the Boldwin effect can be found in their experiment, but theoretical justification on the point remains to be worked out.

4.4 Evolutionary Strategy

Finally, we introduce *evolutionary strategy* (ES), developed by Rechenberg. It can be seen as an optimization technique on the basis of the idea of evolution. Mimicking evolution in order to optimize systems was found in German workers like Rechenberg and Schwefel; see [11, 12].

Note that ES has very little to do with biological evolution. Early versions of ES may have been inspired by biological evolution and the approach can be seen as sampling a population of individuals and allowing the successful individuals to dictate the distribution of future generations.

4.4.1 What is Evolutionary Strategy?

An ES algorithm can be described as an iterative fashion in the following:

1. samples a batch of candidate solutions from the search space.
2. evaluates them.
3. discards the ones with low fitness values.

The sampling for a new iteration (or generation) happens around the mean of the best scoring candidate solutions from the previous iteration.

Note that ESs operate at a phenotype or behavioral level. There is no mapping from an individual's genes to its physical expression, as the physical expression is coded directly.

The reason for the approach is to encourage a strong causality. That is, a small change in the coding leads to a small change in the individual. Similarly, a large change in the coding has associated with it a large change.

4.4.2 Applications of Evolutionary Strategy

ESs are classical variants of evolutionary algorithms which are frequently used to heuristically solve optimization problems, in particular in continuous domains. ESs generally evolve a search distribution, usually Gaussian distribution, and follow the scheme of sampling-and-learning.

ESs repeat the procedures of generating a population of candidate solutions from the search distribution and learning the distribution parameters from the generated samples.

Because ES uses floating-point numbers to encode continuous variables used in different equations, it has many applications. It is one of the advantages over other theories of EC.

After the proposal of ESs, *evolutionary stable strategy* was proposed. It is impermeable when adopted by a population in adaptation to a specific environment. It can be seen as a Nash equilibrium that is evolutionarily stable. An evolutionary stable strategy describes tactics used by individual organisms when competing with one another for a given resource.

Note that these tactics can be behavioral or structural. The organism does not consciously choose them, but adopts them as a natural consequence of evolution. Both structures and behaviors are heritable, and as some are successful and some fail, only the better ones are adopted.

Also notice that ES can be competitive with the so-called *reinforcement learning* (RL), which is one of machine learning techniques addressing the optimal behavior in an environment to obtain maximum reward.

RL is separated from supervised and unsupervised learning focusing on solving problems through a sequence or sequences of decisions optimized by maximizing the accrual of rewards received by taking correct decisions.

The main features of RL are: (1) Learning by trial-and-error, (2) Learning with optimal control, and (3) Learning considering temporal difference. Historically, RL was developed by Sutton and Barto since the 2010s; see Sutton and Barto [13]. It is thus important for the developments of AL.

We finish the exposition of evolutionary computation. The above mentioned methods are inspired by the mechanisms of biological evolution. However, the creativity of evolution is not limited to the natural world.

Artificial organisms evolving in computational environments give the subject of research. In fact, the process of evolution is an algorithmic process. Therefore, we believe that evolutionary computation can serve as one of the promising foundations for AL. However, a lot of work should be done to justify it.

References

1. Boldwin, M.: Heredity and instinct. Science **3**(438–441), 558–561 (1896)
2. Darwin, C.: On the Origin of Species. John Murray, London (1859)
3. Fogel, D., Owens, A., Walsh, M.: Artificial Intelligence Through Simulated Evolution. John Wiley and Sons (1966)
4. Hinton, G., Nowlan, S.: How learning can guide evolution. Complex Syst. **1**, 495–502 (1987)
5. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
6. Holland, J.: Hidden Order: How Adaptation Builds Complexity. Addison-Wesley, New York (1995)
7. Koza, J.: Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems. Stanford University (1990)
8. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Massachusetts, Cambridge (1992)
9. McCall, J.: Genetic algorithms for modelling and optimisation. J. Comput. Appl. Math. **184**, 205–222 (2005)
10. McCarthy, J.: LISP 1.5 Programmer's Manual. MIT Press, Massachusetts, Cambridge (1962)
11. Rechenberg, I.: Evolutionstrategie. Frommann-Holzboog, Stuttgart (1973)
12. Schwefel, H.-P.: Numerische Optimierung von Computer-Modellen Mittels der Evolutions-Strategie. Birkäuser, Basel (1977)
13. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction, 2nd edn. MIT Press, Massachusetts, Cambridge (2018)

Chapter 5
Wetware

Abstract This chapter is an exposition of Wetware. By Wetware, we mean Artificial Life by using biochemical methods. Thus, it is considered to be any hardware or software systems with a biological component or biological systems which act like software and hardware. After providing general discussion on Wetware, we say more about DNA computers, artificial bacteria, clones, and artificial cells.

5.1 From Hardware and Software to Wetware

5.1.1 What Is Wetware?

By *Wetware*, we mean Artificial Life by using biochemical methods. Thus, it is considered to be any hardware or software systems with a biological component or biological systems which act like software and hardware.

It is also used as a general term for biochemical technologies related to life. Thus, wetware usually meansv the mixture of software, hardware, and biology. In fact, it is based on the idea between software and genetic codes.

"Wetware" is a slang term for organic parts integrated into a computer system, and is also a term used to describe the system's biological components, in comparison to "hardware" (the actual parts of a computer system) and "software" (the programs and applications that run on the system). These systems were mainly theoretical until reasonably recently.

The prosthetics and hearing aids that have been used in the past are clearly wetware. And with the advancement of microchips, the potential of wetware is spread even wider.

Therefore, the development of humanoid robots formed by artificial organs, i.e. *android*, seems to be our dream. Needless to say, Android is one of the ultimate Artificial Life systems.

However, "wetware", which is discussed in the field of Artificial Life, often refers to systems for biochemically synthetic life. Therefore, unlike other approaches to Artificial Life, wetware artificially synthesizes the cellular mechanisms of existing living things.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Akama, *Artificial Life*, Intelligent Systems Reference Library 262,
https://doi.org/10.1007/978-3-031-74422-8_5

There is no hardware or software in wetware. Although the actual mechanisms of life may be partially simulated, Wetware can be said to be "life" in itself. In other words, wetware is in a sense similar to "living organisms" in nature, and it can be called "artificial life".

However, the wetware at present is at the level of bacteria, viruses, and cells. When wetware is realized, several problems arise. For example, it could substantially affect the natural world.

Also, artificial viruses may be exploited to damage humans. It may be possible to affect the nature. The possibility of artificial viruses and bacteriological weapons being used by terrorists for war cannot be ruled out.

5.1.2 Computer Virus

Note here that *computer virus* is also considered to be "artificial life" in a broad sense. Computer virus invades a computer and causes damage, but in reality it is some kind of "program".

Most computer viruses are self-reproduced, so they could be called artificial life. However, computer viruses are not wetware but soft ALife. It is therefore necessary to distinguish wetware and computer virus.

Currently, research on computer technology based on biochemical theories is also being conducted. Such an approach is believed to enable ultra-fast computation. Also, these technologies are also expected to contribute significantly to the advancement of wetware.

5.2 DNA Computers

Biocomputer is a computer that uses the functions of brain and nerve of living organisms. It is also called a *biological computer*. Biocomputer is a new idea for computing, and is considered to be useful for research on AL.

Biocomputer is based on biological elements like *DNA* instead of electrical elements like semiconductors. Therefore, DNA computer is viewed as a type of biocomputer. In a wider sense, neurocomputer may be seen as a biocomputer.

One of the most promising biocomputers is the *DNA computer* and it has the following features.

(1) It is possible to perform ultra-high-speed computation by massively parallelism.
(2) It is possible to downsize and save energy in computers.
(3) It provides a new computational model.
(4) It is promising to apply to the medical field.

(1) implies the following, i.e., DNA computer is different from the conventional von Neumann type computer. Since massively parallel is possible, ultra-high-speed computation is possible.

Here, *massively parallel* plays an important role. Roughly, it refers to the parallel processing by using a large number of processes. Note that this advantage may also be found in neurocomputers and quantum computers.

(2) suggests the two points. First, the DNA computer constitutes hardware at the molecular level, computers can be miniaturized. On the other hand, conventional hardware uses "machines", there is a limit to miniaturization.

Second, since the computation is performed as a chemical reaction, energy saving is possible. In other words, it can perform calculation without consuming much energy. This feature was also noted by Bennett in the 1980s; see Bennett [5].

(3) means that the DNA computer has the potential to give rise to a quite different computational model. The idea is in fact different from other computers and computational models. Currently, the computability DNA computer is not entirely clear, however.

To solve this problem, a new formalization of the computational model like biological Turing machines, is required. As will be discussed below, molecular automata has been proposed.

Molecular automata employ molecular-scale phenomena of binding, dissociation, and catalytic action to achieve predetermined functions. Note that different chemical implementations are possible like proteins and inorganic reactions.

(4) addresses that DNA computer can be applied to the medical field. The principles and features like miniaturization of DNA computer are suitable to the design of ideal medical systems.

Before presenting DNA computers, we give the basics of DNA shortly. *DNA* (Deoxyribonucleic acid) is a molecule that contains the biological instructions that make each species unique. DNA, along with the instructions it contains, is passed from adult organisms to their offspring during reproduction.

A *genome* is all the genetic information of an organism. It is the entire set of DNA instructions found in a cell. Virtually every single cell in the body contains a complete copy of the approximately 3 billion DNA base pairs, or letters, for making up the human genome. For instance, in humans, the genome consists of 23 pairs of chromosomes in the cell's nucleus as well as a small chromosome in the cell's mitochondria. By 2003, we knew the DNA sequence of the entire human genome.

Nucleic acids are large biomolecules that play essential roles in all cells and viruses. They play an important role in the storage and expression of genomic information. *DNA* (Deoxyribose nucleic acid) is made of chemical building blocks called *nucleotides*.

DNA encodes the information cells need to make proteins. A related type of nucleic acid, called *RNA* (ribonucleic acid) comes in different molecular forms that play multiple cellular roles, including protein synthesis.

The four types of nitrogen bases found in nucleotides are: adenine (A), thymine (T), guanine (G) and cytosine (C). The order of these bases determines what biological instructions are contained in a strand of DNA.

Fig. 5.1 Structure of DNA

DNA, displayed as Fig. 5.1, forms a double helix structure by the four types of bases, i.e., *adenine* (A), *guanine* (G), *cytosine* (C), and *thymine* (T).

In 1953, Watson and Crick discovered that DNA forms the *double-helix structure*; see Watson and Crick [24]. Later, Watson and Click, together with Wilkins, awarded a Nobel Prize in Physiology or Medicine for their work in 1962.

The human genome includes the coding regions of DNA, which encode all the genes (between 20,000 and 25,000) of the human organism, as well as the noncoding regions of DNA, which do not encode any genes. By 2003 the DNA sequence of the entire human genome was known by the Human Genome Project (HGP).

The *central dogma* was proposed by Crick in the late 1950s. The theory suggested that genetic information flows primarily from nucleic acids in the form of DNA and RNA to functional proteins during the process of gene expression. It states that genetic information flows only in one direction, from DNA, to RNA, to protein, or RNA directly to protein (Fig. 5.2).

Later, researchers have discovered several exceptions to the theory. The famous example is that of prions. Prions are infectious proteins which replicate without going through DNA or RNA intermediates. Prions are responsible for the rare but devastating neurologic disease.

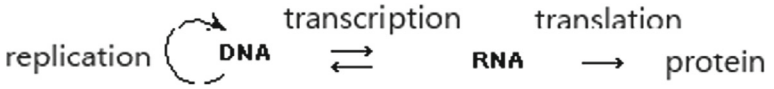


Fig. 5.2 Central dogma

We turn to biological computers. The theoretical possibilities of the biocomputer have been already suggested in the 1950s by Feynman; see Feynman [11]. Feynman insisted that by using biological systems it may be possible to miniaturize computers.

Feynman said:

> The biological example of writing information on a small scale has inspired me to think of something that should be possible. Biology is not simply writing information; it is *doing something* about it.

Feynman implicitly suggested *biological computers* by addressing miniaturization. Feynman also proposed the idea of a quantum computer in the 1960s.

Unfortunately, Feynman's ideas of molecular computers and quantum computers are unlikely to be realized. The main reason is that researchers did not pay attention until decades later in the 1990s.

In 1982, Bennett proposed a thermodynamics-based computer; see Benett [5]. Bennett described the Brownian computer based on the principle of Brownian motion. The work revealed that computation based on molecules are theoretically possible.

In fact, Bennett argues that the Brown-Turing machine can be constructed from macromolecules such as RNA. Therefore, we could call Bennett the pioneer of the DNA computer.

In 1985, Conrad showed an outline of a molecular computer; see Conrad [10]. This makes it theoretically possible to realize a molecular computer. Conrad gave some discussions supporting that molecules instead of digital circuits can sever as a basis of computing.

However, Conrad argues that every computer including molecular computers has a trade-off between programmability, computational efficiency and evolutionary applicability.

Conrad said:

> Here the study of molecular- computer designs has immediate practical significance for computer science independent of whether or not such designs can be concretized either in the near or distant future. (p. 479)

In 1994, Adleman of the University of Southern California developed an experimental implementation of a DNA computer, and solved the Hamilton path problem; see Adelman [1, 2]. He is also famous as one of the founders of the RSA code. His work raised interest of DNA computer in the 1990s.

In 2004, Shapiro et al. of the Weizmann Institute in Israel succeeded in developing a DNA computer with the input-output function; see Benenson et al. [3]. Their

DNA computer can perform both input and output and do computation by means of molecules.

Also noted are the fact that both hardware and software are constructed by molecules. Therefore, their research is considered a major step towards the practical application of DNA computers.

It is hoped that DNA computers, along with *quantum computers*, will be realized as soon as possible as next-generation computers. In addition, DNA computers are expected to play an important role in research on AL in the future.

Now, we give the details of DNA computers. There are several ways of thinking about DNA computers, but the basic principle is to use the properties of the double helix structure of DNA and the manipulation of DNA by enzymes for computation. An *enzyme* is a biological catalyst and is almost a protein. It speeds up the rate of a specific chemical reaction in the cell.

In DNA, as mentioned above, bases of adenine, guanine, cytosine, and thymine form the double helix structure, and various enzymes cause reactions to DNA. Therefore, basically, if we think of DNA as a state and enzymes as instructions, then we can describe computation using DNA.

In other words, the DNA reactions caused by enzymes can be interpreted as computation. It is also recognized that the hardware of a DNA computer itself can be constructed by means of DNA.

When Adelman devised a DNA computer, he focused on an enzyme called *DNA polymerase*, which has the function of replicating and repairing DNA. The function of DNA polymerase to replicate DNA is similar to the basic operation of computational models like Turing machines and computers. Adelman used this idea to solve the Hamilton path problem by using a DNA computer.

The *Hamilton path problem* is one of the benchmark problems of computational complexity, i.e., NP complete problem. It is to find a path that visits each city once, returns to the starting city, and minimizes the distance traveled.

The only known general solution algorithm guaranteeing the shortest path requires a solution time that grows exponentially with the problem size (i.e., the number of cities).

By developing the study of DNA computers, many operation on DNA were proposed. The following are main operations.

---

(1) melting

(2) annealing

(3) merge

(4) amplification.

Melting is an operation that separates double-stranded DNA into single strands by heating the solution to a certain temperature.

Annealing is a reverse operation of melting, in which two complementary DNAs combine to become one.

Merge is an operation of pouring them into one, in which it combines solutions from two test tubes into one test tube.

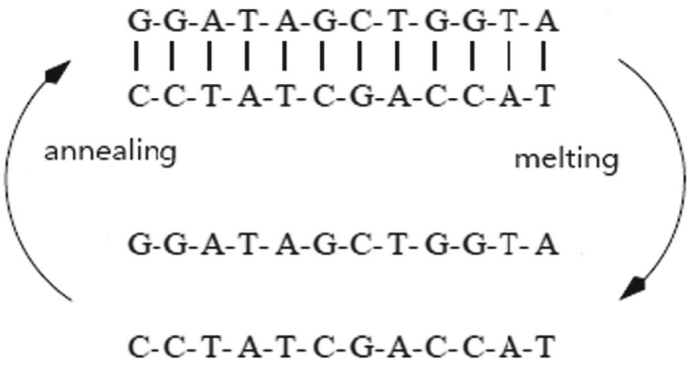


Fig. 5.3 Melting and Annealing

Amplification is operation to replicate DNA, by using PCR (Polymerase Chain Reaction).

For example, the operations melting and annealing are shown in Fig. 5.3.

Some experts seem to predict DNA computing would beat silicon-based technology, particularly with massively parallel computing. Later, researchers concluded that DNA computing is not fast enough to do that. Currently, the conclusion in fact is true.

5.2.1 Adleman's Approach

Here, we give the outline of Adleman's Approach. First of all, it should be noted that Adleman did not present a general-purpose DNA computer, but a DNA computer dedicated to solving the Hamilton path problem mentioned above.

A directed graph G with designated vertices v_{in} and v_{out} is said to have a Hamiltonian path if and only if there exists a sequence of compatible "one-way" edges $e_1, e_2, ..., e_n$, i.e., a path that begins at v_{in}, ends at v_{out}, and enters every other vertex exactly once.

For example, the directed graph of Fig. 5.4 (from Adleman [1]), the unique Hamiltonian path with $v_{in} = 0$ and $v_{out} = 6$ is $0 \to 1, 1 \to 2, 2 \to 3, 3 \to 4, 5 \to 6$.

The algorithm of solving the Hamiliton path problem is as follow:

(1) Generate random paths through the graph.
(2) Keep only those paths that begin with v_{in} and end with v_{out}.
(3) If the graph has n vertices, then keep only those paths that enter exactly n vertices.
(4) Keep only those paths that enter all of the vertices of the graph at least once.
(5) If any paths remain, say "Yes", otherwise, say "No".

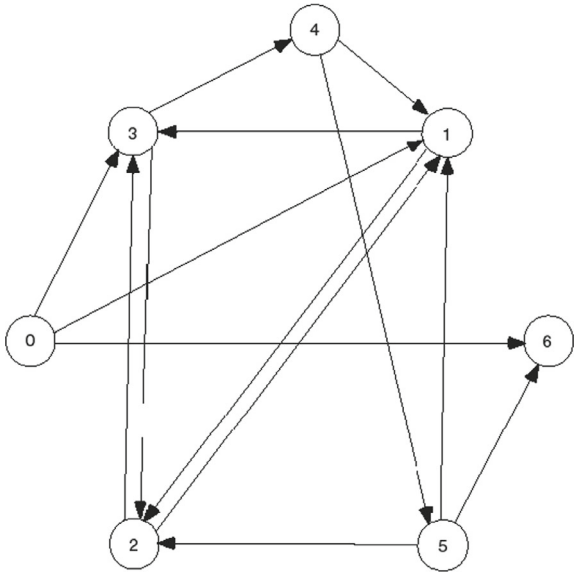


Fig. 5.4 Directed Graph

The DNA computer implements this algorithm by using DNA.

To implement Step 1 of the algorithm, each vertex i in the graph was associated with a random 20-mer sequence of DNA denoted O_i. For each edge $i \rightarrow j$ in the graph, an oligonucleotide $O_{i \rightarrow j}$ was created.

Adleman's experiment shows as follows:

O_2 : TATCGGATCGGTATATCCGA
O_3 : GCTATTCGAGCTTAAAGCTA
O_4 : GGCTAGGTACCAGCATGCTT
$O_{2 \rightarrow 3}$: GTATATCCGAGCTATTCGAG
$O_{3 \rightarrow 4}$: **CTTAAAGCTAGGCTAGGTAC**
$\overline{O_3}$: CGATAAGCTCGAATTTCGAT

Here, $\overline{O_i}$ denotes the Watson-Click complement of O_i.

To implement Step 2 of the algorithm, the product of Step 1 was amplified by polymerase chain reaction (PCR). Thus, only those molecules encoding paths that begin with vertex o and end with vertex 6 were amplified.

To implement Step 3 of the algorithm, the product of Step 2 was run on an agarose gel.

To implement Step 4 of the algorithm, the product of Step 3 was affinity-purified with a biotin-avidin magnetic heads system.

To implement Step 5 of the algorithm, the product of Step 4 was amplified by PCR and run on a gel.

The above computation required approximately 7 days of lab work. In general, with use of the above algorithm, the number of procedures required should grow linearly with the number of vertices on the graph. For large graphs, alternative procedures might be needed.

Although Adleman's work was very important as the initial work of DNA computer, he did not address its efficiency. Nevertheless, his approach opened the possibilities of DNA computers.

Adleman [1] said the following at the end of his paper:

> Research in chemistry may allow for the development of synthetic designer enzymes. One can imagine the eventual emergence of a general purpose computer consisting of nothing more than a single macromolecule conjugated to a ribosome like collection of enzymes that act on it. (p. 1023)

We think that Adleman's work is very suggestive for the foundations for DNA computing. Since his work, many approaches have been proposed as described below.

Computational Complexity

Now, we say something about *computational complexity* or complexity. Generally, algorithms with small complexity are considered as "good algorithms". This is the reason to theoretically.

There are two types of computational complexity: *time complexity*, which refers to the speed of computation, and *space complexity* which refers to the memory required for computation.

Ideally, the algorithm should have as little space as possible for computation and as fast as possible. In practice, the former is mainly discussed as a theory of computational complexity; see Garey and Jhonson [12].

When considering the efficiency of an algorithm, the size of the problem n (e.g., the length of the input data) comes into play. As the n increases, it becomes difficult to formulate the evaluation for strict computational complexity.

Therefore, we consider only the dominant term of the evaluation and ignore the constant coefficient. Which is called the *order* of computational complexity. For example, if the term $n - 1$ is large for n, then -1 can be ignored.

Theoretically, for a real number a and a natural number n_0, and if

$$\forall n \geq n_0 (f(n) \leq ag(n))$$

is satisfied, the order $f(n)$ of the computational complexity becomes $g(n)$, denoted $O(g(n))$. This notation is called the *big O notation*.

From this definition, it follows that the algorithm with the higher the order, the more the complexity. Also noted is that depending on the nature of $g(n)$ in $O(g(n))$, the definition of computational complexity will be different.

Using the concept of computational complexity, we can classify problems for which an algorithm exists as practical or not. In other words, algorithms can be classified into problems that can be solved theoretically but practically, and problems that can be solved theoretically but not practically.

The former problem is called the *tractable problem*, and the latter the *intractable problem*, respectively. For example, the problems with the computational complexity of polynomials such as $O(n)$.

$O(log_2 n)$ are generally considered to be easy to handle. The problems with the computational complexity of exponential functions such as $O(n^p)$ are considered to be intractable problems.

In theory of algorithms, a problem in which the computational complexity is $O(poly(n))$ for a polynomial $poly(n)$ is called the *P problem* (polynomial time computable problem). Note that the class of the P problems is written as **P**. Therefore, **P** is the class of problems that are tractable.

Algorithms generally the output results for a unique input data. Such an algorithm is called the *deterministic algorithm*. However, some algorithms have multiple possibilities for operations to be performed at the time of calculation, and some are non-deterministic to be selected and executed. Such a non-deterministic algorithm is called the *non-deterministic algorithms*.

NP Problem (non-deterministic polynomial time computable problem) is a problem that can be solved by a non-deterministic algorithm in polynomial time in the best case. The class is written as **NP**. From these definitions, the following relationship holds:

$$\mathbf{P} \subseteq \mathbf{NP}$$

Here, an interesting question is whether $\mathbf{P} = \mathbf{NP}$. However, this problem has not yet been proven, and it is known as one of the most important unsolved problems in computer science as the *P = NP problem*.

To prove $\mathbf{NP} = \mathbf{P}$, you only need to prove $\mathbf{NP} \subseteq \mathbf{P}$. However, this is currently not proven. The problem is considered one of the most well-known and important unsolved problems in computer science.

$P = NP$ problem is one of the unsolved problems in computer science selected by the Clay Mathematics Institute in 2000. Several years ago, there was a news that this problem has been solved, but we found that it has not been really solved. Readers still have a chance of the challenge. Currently, the author expects to see that $\mathbf{P} \neq \mathbf{NP}$.

For the **P = NP** problem, Cook proposed the so-called the *NP complete problem*, see Cook [9]. The NP complete problem is defined as the most difficult of the NP problems.

So, the **NP** complete problem is considered to be one of the keys to solving the **P = NP** problem. In other words, if you prove that the NP complete problem belongs to the P problem, then you can prove **P = NP**.

Here are some of the NP complete problems:

- Propositional (Boolean) satisfiability problem (SAT)
- Hamiltonian cycle problem
- Traveling salesman problem
- Knapsack problem
- Integer linear programming problem
- Set partition problem.

In fact, many of the hard-to-compute problems are the NP-complete problems. Often mentioned NP complete problems are the propositional (Boolean) satisfiability problem and the Hamiltonian cycle problem.

Therefore, the fact that Adleman actually built a DNA computer solving the Hamilton path problem is very significant from the perspective of computational complexity. This is because it demonstrated the computation using a DNA computer. Unfortunately, his computer could not perform efficient computation.

5.2.2 Lipton's Approach

After Adelman's experiment, Lipton demonstrated a solution to the *SAT problem* (satisfiability problem) in Lipton [16]). The SAT problem is to determine satisfiability of Boolean (or propositional) formula.

The problem is known as one of the NP complete problems. To solve the Hamilton path problem and the SAT problem, the usefulness of DNA computers was clearer.

As mentioned above, Lipton, like Adelman, interpreted the satisfiability problem as a Hamiltonian path problem and built a DNA computer. Although Lipton solved the two-variable satisfiability problem, he also discussed its generalization.

Adelman in fact showed a DNA computer that solves the Hamiltonian path problem, but the question is whether can be extended for important other NP-complete problems.

The problem is that Adelman's method does not mean that all NP-complete problems can be solved efficiently. Therefore, Lipton's research is very significant in that it demonstrates the potential of a DNA computer.

As Lipton points out, the speed of all computers including biological computers is determined by two factors. That is, (1) the number of parallel processes and (2) the number of steps each can perform per unit time. For biological systems, the first factor can be very large.

As described above, SAT is a simple search problem that is NP complete problem. We explain Lipton's example. SAT is the problem whether a propositional (Boolean) formula is *satisfiable*.

By a propositional and a Boolean formula, we mean formulas in propositional logic and Boolean algebra. A *truth assignment* is to assign the truth-value either 0 or 1 to a propositional variable x.

Logical symbols are interpreted in the following way. OR is interpreted as: $x \vee y = 0$ only if $x = y = 0$. AND is interpreted $x \wedge y = 1$ only if $x = y = 1$. NOT is interpreted as: $\overline{x} = 0$ if $x = 1$ and 1 if $x = 0$. The SAT problem is to find Boolean values for x and y that makes the formula F true.

A propositional (Boolean) formula is *satisfiable* iff there is a truth assignment making it true. We can easily show the satisfiability of a formula using *truth-tables* as standard textbooks of logic explain.

Lipton used a different method using a concept of clause. His example is the formula with two Boolean variables $F = (x \vee y) \wedge (\overline{x} \vee \overline{y})$, where $x \vee y, x \wedge y, \overline{x}$

denote disjunction (OR) of x and y, conjunction (AND) of x and y, negation (NOT) of x, respectively. We also use "0" and "1" to stand for the values "false" and "true".

A *clause* is a formula of the form $v_1 \vee ... \vee v_k$, where each v_k is a variable or its negation. The formula F consists of two clauses: The first is $x \vee y$ and the second is $\overline{x} \vee \overline{y}$.

In general, a SAT consists of a Boolean formula of the form $C_1 \wedge ... \wedge C_m$, where each C_i is a clause. The question is to find values for the variables so that the whole formula has the value 1. This is the same as finding values for the variables that make each clause have the value 1.

Strands of DNA are just sequences $\alpha_1, ..., \alpha_k$ over the alphabet {A, C, G, T}. *Double strands* is two DNA sequences, $\alpha_1, ..., \alpha_k$, $\beta_1, ..., \beta_k$. β_i is Watson-Crick complement of α_i satisfies A $\leftrightarrow$ T or C $\leftrightarrow$ G. That is,

$$\begin{array}{l} 5' - \alpha_1 - \alpha_2 - \alpha_3 \ ... - 3' \\ \quad - \updownarrow \quad \updownarrow \ \updownarrow \\ 3' - \beta_1 - \beta_2 - \beta_3 \ ... - 5' \end{array}$$

holds. Here, 5′ and 0-3′ denote different endpoints of the DNA strand.

There are a number of simple operations that can be performed on tesr tubes that contain DNA strands.

---

(1) It is possible to synthesize large number of any short strand (here short as at least 20 nucleotides).

(2) It is possible to create a double strand of DNA from complementary single strands by allowing them to anneal.

(3) Given a test tube of DNA, one can extract those sequences that contain some consecutive pattern of length l.

(4) Given a test tube of DNA, one can perform detect operation.

(5) Given a test tube of DNA, one can amplify all of the DNA strands.

In (3), assuming the pattern $\delta_1, ..., \delta_l$, where each δ_i is in {A, C, G, R}, a DNA strand $\alpha_1, ..., \alpha_k$ will be removed only if, for some i,

$$\delta_1 = \alpha_1, \delta_2 = \alpha_{i+1}, ..., \delta_k = \alpha_{i+k-1},$$

which is called "contract.".

For (4), the operation can be done by PCR.

In the computations, one fixed text tube will be used for all computations. The set of DNA in this test tube correspond to the following simple graph G_n. The test tube is formed in the same way Adleman formed the test tube of all paths to find the Hamiltonian graph.

The graph G_n has nodes $a_1, x_1, x'_1, a_2, x_2, x'_2, ..., a_{n+1}$ with edges from a_k to both x_k and x'_k and from both x_k and x'_k to a_{k+1} (Fig. 5.5).

The paths of length $n + 1$ that start at a_1 and end at a_{n+1} are assumed to be in the initial test tube. This graph is constructed so that all paths that start at a_1 and end at a_{n+1} encode an n-bit binary number.

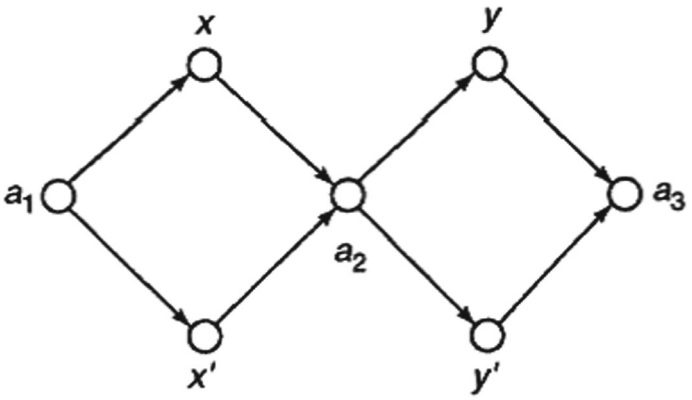


Fig. 5.5 The Graph G_n which encodes two-bit numbers

At each stage, a path has exactly two choices. If it takes the vertex with an unprimed label, it will encode a 1; if it takes the vertex with a primed label, it will encode 0. Therefore, the path $a_1 x' a_2 y a_3$ encodes the binary number 01 (Fig. 5.5).

The graph is encoded into a test tube of DNA as follows. Each vertex of the graph is assigned a random pattern of length l from {A, C, G, T}. The length of $l = 20$ used in Adleman's paper, should suffice here.

This "name" of the vertex has two parts. The first half is denoted by p_i, and the second half by q_i. Thus, $o_i q_i$ is the name associated with the i-th vertex. Then, a test tube is filled with the following kinds of DNA strands.

---

(1) For each vertex, put many copies of a $5' \to 3'$ DNA sequence of the form $p_i q_i$ into the test tube.

(2) For each edge from $i \to j$, place many copies of a $3' \to 5'$ DNA sequence of the form $\hat{q}_i \hat{p}_j$ ($\hat{x}$ denotes the sequence that is the Watson-Crick complement to x).

(3) Add a $3' \to 5'$ sequence of length $l/2$ complementary to the first half the initial vertex to the test tube. Similarly, add a $3' \to 5'$ sequence complementary to the last half of the final vertex to the test tube (that is, add $\hat{p}_1$ and $\hat{q}_n$).

Here, every legal path in G_n corresponds to a correctly matched sequence of vertices and edges. Operations are performed only on the DNA sequences from the graph G_n. We use $E(t, i, a)$ to denote all of the sequences in test tube t for which the i-th bit is equal to $a \in \{0, 1\}$.

Now, we construct a series of test tubes for $F = (x \vee y) \wedge (\bar{x} \vee \bar{y})$. The first one t_0 is just the test tube of all two-bit sequences. Then, operate as follows.

---

(1) Let t_1 be the test tube that corresponds to $E(t_1, 1, 1)$. Let the remainder be t'_1 and let t_2 be $E(t'_1, 2, 1)$. Pour t_1 and t_2 together to form t_3.

(2) Let t_4 be the test tube that corresponds to $R(t_3, 1, 0)$. Let the remainder be t'_4. Let t_5 be $E(t'_4, 2, 0)$. Again pour t_4 and t_5 together to form t_6.

Table 5.1 Values encoded by the DNA in the last tube during the biological solution of F

Test tube	Values present
t_0	00, 01, 10, 11
t_1	10, 11
t'_1	00, 01
t_2	01
t_3	01, 10, 11
t_4	01
t'_4	10, 11
t_5	10
t_6	01, 10

(3) Check to see if there is any DNA in the last test tube t_6. The satisfying assignments are exactly those in the final test tube.

Consider Table 5.1 to understand how this works,

Here, tube t_3 consists of all those sequences that satisfy the first clause: 01, 10, 11. In the same way, t_6 consists of all from t_3 that satisfy the second clause: 01, 10. The latter are exactly the correct answer to the given SAT problem.

As described above, Lipton constructed the DNA computer by interpreting the SAT problem as Adleman did for the Hamilton problem. Lipton solved the SAT problem with two variables, and he also discussed the general case.

Any SAT problem on n variables and m clauses can be solved with at most order m extract steps and one detect step. By order m, we mean that the number of steps is linear in m.

This assumes, as usual, that each clause consists of a fixed number of variables or their negations. Let $C_1, ..., C_m$ be the clauses. A series of test tubes $t_0, ..., t_m$ are constructed so that t_k is the n-bit numbers x so that $C_1(x) = C_2(x) = ... = C_k(x) = 1$, where $C_i(x)$ is the values of the clause C_i on the settings of the variables to x.

For t_0, use the set t_{all} of all possible n-bit numbers. Assuming t_k has been constructed, we construct t_{k+1}. Let C_{k+1} be the clause

$$v_1 \vee ... \vee v_l$$

where each literal v_i is a literal or a complement of a literal.

For each literal v_i, operate as follows: If v_i is equal to x_j, then form $E(t_k, j, 1)$; if v_i is equal to $\bar{v}_j$, then form $E(t_k, j, 0)$. As in the example, the remainder of each extraction is used for the next step.

Pour all of these together to form t_{k+1}. Then, do one detect operation on t_m to decide whether or not the clauses are satisfiable.

This process assumes that the operations are perfect, that the operations are performed without error. This definitely needs to be studied. This definitely needs to be studied.

The assumption that the extract gets all of the sequences is not needed. If the original test tube has many copies of the desired sequence, then all that is necessary is a reasonable probability that it is correctly extracted to make everything work properly.

These methods can be used to solve a generalization of SAT. This generalization includes most examples of NP problems. The key is to generalize the class of Boolean formulas that we consider.

Recall that a SAT problem corresponds to a formula of the restricted form

$$C_1 \wedge \ldots \wedge C_m.$$

A natural generalization of this is to consider problems that correspond to any Boolean formula. Thus, we allow formulas to be unrestricted: They can use the logical operations of negation, OR and AND without any restrictions.

More precisely, formulas are defined by the recursive definition:

(1) Any variable x is a formula.

(2) If F is a formula, then so is $\bar{F}$.

(3) If F_1 and F_2 are formulas, then so are $F_1 \wedge F_2$ and $F_1 \vee F_2$.

The size of a formula is measured by the number of operations used to build the formula. The SAT problem for formulas is, given a formula F, find an assignment of Boolean values to the variables so that F is true. Because this problem includes normal SAT, it is still NP-complete.

This SAT problem for formulas can be solved in a number of DNA experiments that are linear in the size of the formula. The key to proving this statement is to actually prove more; we show how to solve not just any formula, but any contact network; see Shannon [22].

A *contact network* is a directed graph with a single special source s and a single special sink t. Each edge is labeled with either x or $\bar{x}$, where x is some variable. Given any assignment of values to the variables, an edge is considered to be connected if the edge's formula evaluates to 1. If the edge is labeled with $\bar{x}$, then it is connected only if $x = 0$.

Therefore, the network in Fig. 5.6 is equal to 1 only if $w = 1$ or $x = y = z = 1$.

The network in Fig. 5.6 is satisfied only if $w = 1$ or $x = y = z = 1$.

The SAT problem for contact networks is to determine whether or not there is an assignment of values to the variables such that there is a directed connected path

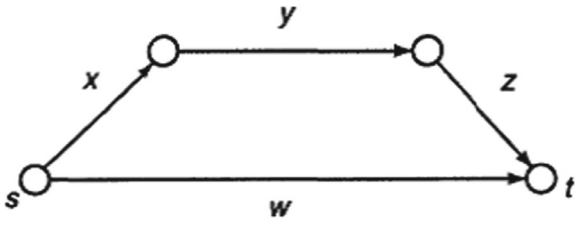


Fig. 5.6 A simple contact network

from s to t. If two edges have the same label, then one is connected if and only if the other is. Put another way, all values of x or $\bar{x}$ are consistent.

Our result follows from two simple claims:

(i) Given any formula of size S, there is a contact network of size linear in S such that the set of assignments that satisfy the formula also satisfy the network.

(ii) Given any contact network of size S, the SAT problem for the network can be solved in order S DNA experiments. These two claims will prove our assertion about formulas.

The first claim is due to Shannon [22]. Two formulas are equivalent if they always give the same value for any assignment to the variables. Any formula can be placed into a normal form with De Morgan's Laws.

$$x \bar{\vee} y = \bar{x} \wedge \bar{y}$$
$$x \bar{\wedge} y = \bar{x} \vee \bar{y}.$$

Through these identities, any formula equivalent to one where all the negations are on variables. Assuming that our formulas are so restructured, we build a contact network that simulates the formula inductively.

If the formula is a variable or its negation, then there is a single-edge contact network that is equivalent. For example, the formula $\bar{x}$ is equivalent to the network with an edge from s to t with the label $\bar{x}$.

In the general case, the formula is equal to either $E \vee F$ or $E \wedge F$, where E and F are simpler formulas. Assuming that G is the network for E and that H is the one for F, the network for $E \vee F$ is constructed by placing G and H in parallel (Fig. 5.7a).

Clearly, there is a connected path from s to t provided that there is either a path from 5 to t through G or through H. The network for $E \wedge F$ is constructed by p1acing them in series (Fig. 5.7b). In this case, there is a connected path from s to t provided there is one through both G and H.

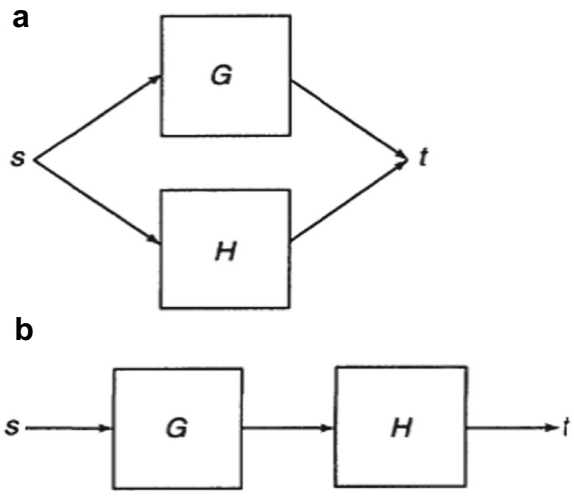


Fig. 5.7 The Networks for $E \vee F$ (**a**) and $E \wedge F$ (**b**)

Here, G is the network for E and H is that for F **(A)**. The OR case is constructed by connecting the networks in parallel and **(B)** the AND case is built by placing the networks in series.

It is quite simple to show how DNA experiments can be used to solve the SAT problem for any contact network. Associate a test tube P_v with each node v in the contact network. The DNA in each test tube should encode in the usual way the set of assignments to the variables that connect s to v.

The test tube P, associated with the sink t is the "answer". Suppose that $v \rightarrow u$ is an edge with the label $x(\bar{x})$ and that P_v is already constructed. Then, construct P_u simply by doing the extraction $E(P_v, x, 1)[E(P_v, x, 0]$.

If several edges leave a vertex v, then use an amplify step to get multiple copies of the DNA in test tube P_v. Also, if several edges enter a vertex v, then pour the resulting test tubes together to form P_v.

Lipton said:

> I expect that in the near future, experiments will be performed that will determine whether or not DNA-based computers are a practical means of solving hard problems. (p.545)

5.2.3 Shapiro's Approach

Shapiro's research group is changing the situation of DNA computers. Research by Shapiro et al. in 2004 brought the practical application of DNA computers; see Benenson et al. [4]. Shapiro believes that it is beneficial for DNA computers to develop medical work.

In 2001, Shapiro et al. [3] proposed the *molecular automata* which correspond to the finite automata. A finite automaton3 is a notional computing machine that operates on finite sequences of symbols. The machine can be in one of a finite number of internal states, of which one is designated an initial state and some are designated accepting states.

Its software consists of transition rules, each specifying a next state based on the current state and the current symbol. It is initially positioned on the leftmost input symbol in the initial state.

In each transition the machine moves one symbol to the right, changing its internal state according to one of the applicable transition rules. Alternatively, it may 'suspend' without completing the computation if no transition rule applies.

A computation terminates on processing the last input symbol. An automaton is said to accept an input if a computation on this input terminates in an accepting final state.

Recall that an *automaton* is a model for computation, and is applied to describe various systems. Since automata are general, they can also be used for DNA computers.

An automaton can describe computation by means of inputs, internal states, and outputs. It is used as models for hardware, software, algorithms, processes, languages, etc. For automata theory, the reader is referred to Hopcroft, Motwani and Ullman [14].

The main automata in order of computational power are described as follows:

- Finite automaton (FA)
- Pushdown automaton (PDA)
- Linear bounded automaton (LBA)
- Turing machine (TM).

A *finite automaton* (FA) is an automaton that has an input device but no output device. Finite automaton is also called *finite state machine* (FSM).

A *pushdown automaton* (PDA) is an automaton which extends FA with a stack.

A *linear bounded automaton* (LBA) is an automaton which can be seen as a restricted form of Turing machine.

A *Turing machine* (TM) is the most powerful automaton simulating any type of computation.

There are close connections of automata and formal grammars. A *formal language* is generated by a *formal grammar*. The all strings accepted by an automaton correspond to the formal language generated by the corresponding formal grammar.

There are some types of formal grammars. A *regular grammar* has the rewriting rules of the form either $A \to a$, or $A \to aB$, where $A, B \in V_N, a \in V_T$.

A *context-free grammar* has the rewriting rules of the form $A \to \alpha$, $A \in V_N$.

A *context-sensitive grammar* has the rewriting rule of the form $\alpha \to \beta$, where $|\alpha| \leq |\beta|$. ($|\alpha|$ denotes the length of sequences of α).

A grammar without any restriction is the *unrestricted grammar*.

The languages generated by the grammars above are *regular language*, *context-free language*, *context-sensitive language*, and *unrestricted language*, respectively.

Chomsky studied the syntactic structure of natural languages; see Chomsky [7]. In connection with the work, Chomsky also proposed the hierarchy of grammars, i.e., type i grammar ($i = 0, 1, 2, 3$), now known as *Chomsky hierarchy*; see Chomsky [8].

Therefore, the unrestricted grammar is the *type 0 grammar*, the context-sensitive grammar is the *type 1 grammar*, the context-free grammar is the *type 2 grammar*, and the regular grammar is the *type 3 grammar*, respectively.

As described above, a finite automaton is a notional computing machine that operates on finite sequences of symbols. The machine can be in one of a finite number of internal states, of which one is designated an initial state and some are designated accepting states.

Molecular automata's software consists of transition rules, each specifying a next state based on the current state and the current symbol. It is initially positioned on the leftmost input symbol in the initial state. In each transition the machine moves one symbol to the right, changing its internal state according to one of the applicable transition rules.

Alternatively, it may 'suspend' without completing the computation if no transition rule applies. A computation terminates on processing the last input symbol. An

automaton is said to accept an input if a computation on this input terminates in an accepting final state.

Because living organisms process information, their materials and mechanisms lend themselves readily to computing. The DNA molecule exists to store data, written in an alphabet of nucleotides. Cellular machinery reads and modifies that information using enzymes and other molecules.

Central to this operating system are chemical affinities among molecules allowing them to recognize and bind with one another. Making molecules into a Turing-like device, therefore, means translating his concepts into their language.

A simple conceptual computer called a finite automaton can move in only one direction and can read a series of symbols, changing its internal state according to transition rules.

A two-state automaton could thus answer a yes-no question by alternating between states designated 1 and 0. Its state at the end of the calculation represents the result.

Raw materials for a molecular automaton include DNA strands in assorted configurations to serve as both input and software and the DNA-cleaving enzyme Fokl as hardware. Nucleotides, whose names are abbreviated A, C, G and T, here encode both symbols and the machine's internal state.

Shapiro's molecular automaton can be seen as a finite automaton with two states. In DNA computers based on molecular automata, all components are DNAs. The input is a DNA base sequence. Then, the four types of short DNA are transition rules, i.e. become the software. In addition, two types of enzymes can serve as the hardware.

Furthermore, Shapiro et al. proposed an ambitious concept called the *DNA Doctor*. Which are embedded in cells, detects illness and outputs appropriate medical treatment.

Since two-state finite automaton was too simple to be of any use in solving complex computational problems, they considered it nothing more than an interesting demonstration of the concept of programmable, autonomous biomolecular computers, and we decided to move on.

Focusing their efforts for a while on trying to build more complicated automata, however, we soon ran up against the problem recognized by Adleman: the "designer enzymes" he had yearned for a decade earlier still did not exist.

No known naturally occurring enzyme or enzyme complex can perform the specific recognitions, cleavages and ligations, in sequence and in tandem, with the flexibility needed to realize the Turing machine design. Natural enzymes would have to be customized or entirely new synthetic enzymes engineered.

Because science does not yet have this ability, we found ourselves with a logical design specification for a biomolecular Turing machine but forced to wait until the parts needed to build it are invented.

That is why we returned to our two-state automaton to see if we could at least find something useful for it to accomplish. With medical applications already in mind, we wondered if the device might be able to perform some kind of simple diagnosis, such as determining whether a set of conditions representing a particular disease is present.

For this task, just two states suffice: we called one state "yes" and the other "no". The automaton would begin the computation in the yes state and check one condition at a time.

If a condition on its checklist were present, the yes state would hold, but if any condition were not present, the automaton would change to the no state and remain that way for the rest of the computational process. Thus, the computation would end in yes only if all the disease conditions held, but if one condition were not met the "diagnosis" would be negative.

To make this logical scheme work, we had to find a way to connect the molecular automaton to its biochemical environment so that it could sense whether specific disease conditions were present.

The general idea that the environment could affect the relative concentrations of competing transition molecules—and thus affect the computation—had already been suggested in the blueprint for the molecular Turing machine.

To apply this principle to sense disease symptoms, we had to make the presence or absence of a disease indicator a determinant of the concentration of software molecules that testify for the symptom.

Shapiro and Benenson [23] said:

> Human physicians make this kind of decision whenever they weigh the risk to a patient of a possible disease against the toxicity of the treatment and the certainty of the diagnosis. In the future, if our molecular automaton is sent on a medical mission, it can be programmed to exercise similar judgment. (p. 51)

Their remark is very interesting, because DNA computers are useful for doctors and medical applications. If their approach is realized, then Artificial Life (in particular, the DNA computers) will cure the disease and save our lives.

5.2.4 Hagiya's Approach

In 2000, Hagiya's group in University of Tokyo proposed the DNA computer for solving the satisfiability of three variables (3-SAT); see Sakamoto et al. [20]. They constructed the DNA computer using the method called the *hairpin*.

They encode the given formula in "literal strings" which are conjunctions of the literals selected from each SAT clause (one literal per clause). A formula is satisfiable if there exists a literal string that does not contain any variable together with its negation.

If each variable is encoded as a DNA subsequence that is the Watson-Crick complement of its negation then any strands containing a variable and its negation self-anneal to form "hairpin" structures.

These can be distinguished from non-hairpin structure-forming strands, and removed. The advantage of this approach is that it does not require physical manipulation of the DNA, only temperature cycling. The disadvantage is that it requires 3^n literal strings for n clauses, thus facing with the scalability argument.

Sticker Systems

DNA computers, such as Edleman and Lipton, were devised to solve specific problems, and their computation were performed by manipulating DNA. However, these DNA computers are not versatile. Therefore, we need a computational model of DNA computers.

In 1998, the *sticker systems* were proposed as the first computational model of DNA computers; see Kari et al. [15]. The sticker systems are based on formal language theory for describing processes on DNA by means of sticker operations.

Kari et al. [15] said:

> The *sticker systems* introduced here are language generating devices based on the *sticker operation*, which, in turn, is a model of the techniques used by L. Adleman in his successful experiment of computing a Hamiltonian path in a graph by using DNA. (p.401)

As described before, DNA sequences are in fact double stranded (helicoidal) structures composed of four nucleotides, A (adenine), C (cytosine), G (guanine), and T (thymine), paired A–T, C–G according to the so-called Watson-Crick complementarity.

If we have a single stranded sequence of A, C, G, T nucleotides, together with a single stranded sequence composed of the complementary nucleotides, the two sequences will be "glued" together (by hydrogen bonds), forming a double stranded DNA sequence.

Using this biochemical reaction, Adleman has proceeded to searching Hamiltonian paths in a graph. The sticker operation abstracts the manipulation of DNA used in Adelman's experiments.

The sticker systems are considered to be formalized based on formal language theory. However, hey are more complex than ordinary formal grammars.

Now, we explain sticker operations. Let V be an alphabet (a finite set of abstract symbols) endowed with a symmetric relation ρ (of complementarity), $\rho \subseteq V \times V$. Let # be a special symbol not in V, denoting an empty space (the blank symbol).

Using the elements of $V \cup \{\sharp\}$, we construct the composite symbols of the following sets:

$$\binom{V}{V}_{\rho} = \left\{ \binom{a}{b} \mid a, b \in V, (a, b) \in \rho \right\}$$

$$\binom{\sharp}{V} = \left\{ \binom{\sharp}{a} \mid a \in V \right\}$$

$$\binom{V}{\sharp} = \left\{ \binom{a}{\sharp} \mid a \in V \right\}$$

Here, we denote

$$W_{\rho}(V) = \binom{V}{V}_{\rho}^{*} S(V).$$

where

$$S(V) = \binom{\sharp}{V}^* \cup \binom{V}{\sharp}^*$$

and we call the elements of this set *well-started sequences* (in general, $\{X - *\}$ is the set of all strings, including the empty one denoted by λ, composed of elements of X, and X^* is the set $X^* - \{\lambda\}$).

Stated otherwise, the elements of $W_\rho(V)$ start with pairs of symbols in V, as selected by the complementarity relation, and end either by a suffix consisting of pairs $\binom{\sharp}{a}$ or with a suffix consisting a pair for either a suffix consisting of pairs $\binom{b}{\sharp}$, for $a, b \in V$) (the symbols $\binom{\sharp}{a}, \binom{b}{\sharp}$ are not mixed.)

The sticker operation, denoted by μ, is a partially defined mapping from $W_\rho(V) \times S(V)$ to $W_\rho(V)$, defined as follows. For $x \in W_\rho(V)$, $y \in S(V)$, $z \in W_\rho(V)$, we write

$$\mu(x, y) = z$$

if and only if one of the following cases holds.

(1) $x = \binom{a_1}{b_1} \ldots \binom{a_k}{b_k} \binom{a_{k+1}}{\sharp} \ldots \binom{a_{k+r}}{\sharp} \binom{a_{k+r+1}}{\sharp} \ldots \binom{a_{k+r+p}}{\sharp}$,
$y = \binom{\sharp}{c_1} \ldots \binom{\sharp}{c_r}$,
$z = \binom{a_1}{b_1} \ldots (a_k\ b_k) \binom{a_{k+1}}{c_1} \ldots \binom{a_{k+r}}{c_r} \binom{a_{k+r+1}}{\sharp} \ldots \binom{a_{k+r+p}}{\sharp}$
Here, $k \geq 0, r \geq 1, p \geq 1$, $a_i \in V, 1 \leq i \leq k, b_i \in V, 1 \leq i \leq k, c_i \in V, 1 \leq i \leq r$, $(a_{k+1}, c_i) \in \rho, 1 \leq i \leq r$.

(2) $x = \binom{a_1}{b_1} \ldots \binom{a_k}{b_k} \binom{a_{k+1}}{\sharp} \ldots \binom{a_{k+r}}{\sharp}$,
$y = \binom{\sharp}{c_1} \ldots \binom{\sharp}{c_r} \binom{\sharp}{c_{r+1}} \ldots \binom{\sharp}{c_{r+p}}$,
$z = \binom{a_1}{b_1} \ldots \binom{a_k}{b_k} \binom{a_{k+1}}{c_1} \ldots \binom{a_{k+r}}{c_r} \binom{\sharp}{c_{r+1}} \ldots \binom{\sharp}{c_{r+p}}$
Here, $k \geq 0, r \geq 0, r + p \geq 1, a_i \in V, 1 \leq i \leq k + r, b_i \in V, 1 \leq i \leq k, c_i \in V, 1 \leq i \leq r + p$, $(ak + i, c) \in \rho, 1 \leq i \leq r$.

(3) $x = \binom{a_1}{b_1} \ldots \binom{a_k}{b_k} \binom{\sharp}{b_{k+1}} \ldots \binom{\sharp}{b_{k+r}} \binom{\sharp}{b_{k+r+1}} \ldots \binom{\sharp}{b_{k+r+p}}$,
$y = \binom{c_1}{\sharp} \ldots \binom{c_r}{\sharp}$,
$z = \binom{a_1}{b1} \ldots \binom{a_k}{b_k} \binom{c_1}{b_{k+1}} \ldots \binom{c_{k+r}}{b_{k+r}} \binom{\sharp}{b_{k+r+1}} \ldots \binom{\sharp}{b_{k+r+p}}$
Here, $k \geq 0, r \geq 1, p \geq 1$, $a_i \in V, 1 \leq i \leq k, b_i \in V, 1 \leq i \leq k + r + p, c_i \in V, 1 \leq i \leq r$, $(c_i, b_{k+i}) \in \rho, 1 \leq i \leq r$.

(4) $x = \binom{a_1}{b1} \ldots \binom{a_k}{b_k} \binom{\sharp}{b_{k+1}} \ldots \binom{\sharp}{b_{k+r}}$,
$y = \binom{c_1}{\sharp} \ldots \binom{c_r}{\sharp} \binom{c_{r+1}}{\sharp} \ldots \binom{c_{r+p}}{\sharp}$,
$z = \binom{a_1}{b_1} \ldots \binom{a_k}{b_k} \binom{c_1}{b_{k+1}} \ldots \binom{c_r}{b_{k+r}} \binom{c_{r+1}}{\sharp} \ldots \binom{c_{r+p}}{\sharp}$,

Here, $k \geq 0, r \geq 0, p \geq 0, r + p \geq 1, a_i \in V, 1 \leq i \leq k, b_i \in V, 1 \leq i \leq k + r, c_i \in V, 1 \leq i \leq r + p, (c_i, b_{k+i}) \in \rho, 1 \leq i \leq r$.

Using the sticker operation, we can define a generating/computing mechanism as follows:

A *sticker system* is a construct

$$\gamma = (V, \rho, A, B_d, B_u)$$

where V is an alphabet, $\rho \subseteq V \times V$ is a symmetric relation on V, A is a finite subset of $W_\rho(V)$ (of axioms), and B_d and B_u are finite subsets of $\begin{pmatrix} \sharp \\ V \end{pmatrix}^+$ and $\begin{pmatrix} V \\ \sharp \end{pmatrix}^+$, respectively.

In sticker systems, we start with the sequences in A and we prolong them to the right with the strings in B_d, B_u according to the sticker operations (the elements of B_d are used on the lower row, down, and those of B_u are used on the upper row).

When no blank symbol is present, we obtain a string over the alphabet $\begin{pmatrix} V \\ V \end{pmatrix}_\rho$. The language of all such strings is the language generated by γ.

Formally, we define this language as follows. For two strings $x, z \in W_\rho(V)$, we write

$$x \Longrightarrow z \text{ iff } z = \mu(x, y) \text{ for some } y \in B_d \cup B_u.$$

We denote by $\Longrightarrow^*$ the reflexive and transitive closure of the relation $\Longrightarrow$.

A sequence $x_1 \Longrightarrow x_2 \Longrightarrow \ldots \Longrightarrow x_k, x_1 \in A$, is called a *computation* in γ (of length $k - 1$). A computation as above is *complete* if $x_k \in \begin{pmatrix} V \\ V \end{pmatrix}_\rho^*$ (no blank symbol is present in the last string of composite symbols).

The language generated by γ, denoted by $L(\gamma)$, is defined by

$$L(\gamma) = \{w \in \begin{pmatrix} V \\ V \end{pmatrix}_\rho^* \mid x \Longrightarrow^* w, x \in A\}.$$

Therefore, only the complete computations are taken into account when defining $L(\gamma)$. Note that a complete computation can be continued since we allow prolongations starting from blunt sequences.

Note that the operations used in Adleman's experiment, B_d corresponds the codes of the graph nodes, and B_u corresponds to the complementary strings identifying the arrows in the graph (or conversely), respectively.

By this setting, we can simulate DNA computing as generating mechanisms investigated in formal language theory.

A complete computation $x_1 \Longrightarrow x_2 \Longrightarrow \ldots \Longrightarrow x_k, x_1 \in A, x_k \in \begin{pmatrix} V \\ V \end{pmatrix}_\rho^*$, with respect to γ, is said to be:

primitive if no properly initial part of it is complete.
balanced if in each step $x_i \longrightarrow x_{i+1}$ one uses a sticker operation corresponding to cases 2 or 4 as above. Moreover, cases 2 and 4 alternate from a step to the next one.

Thus, in a primitive computation we do not use sticker operations as in cases 1–4 with $p = 0$, except in the last step. In a balanced computation we allow $p = 0$, but from a step to the next one we have to change the set B_d, B_u from which we take the string to be used.

Let us denote by $L_p(\gamma)$, $L_b(\gamma)$, $L_{pb}(\gamma)$ the languages of the languages of the strings $w \in \binom{V}{V}_p^*$ obtained by a complete computation of γ that is primitive, balanced, both primitive and balanced, respectively.

Assume now that the strings in the sets B_d, B_u are labelled in a one-to one manner by natural numbers from 1 to card(B_α), $\alpha \in \{d, u\}$; denote by $e_\alpha \longrightarrow \{1, ..., \text{card}(B_\alpha)\}$, $\alpha \in \{d, u\}$, the labellings.

For a computation

$$D : x_1 \Longrightarrow x_2 \Longrightarrow, ..., \Longrightarrow x_k, x_i \in A, x_k \in \binom{V}{V}_\rho^*$$

and for $1 \leq j \leq k-1$, we denote

$$e_\alpha(x_j \Longrightarrow x_{j+1}) = \begin{cases} e_\alpha(y) \text{ if } (x_{j+1} = \mu(x_j, y), y \in B_\alpha) \\ \lambda \qquad \text{otherwise} \end{cases}$$

and we define

$$e_\alpha = e_\alpha(x_1 \Longrightarrow x_2)e_\alpha(x_2 \Longrightarrow x_3)...e_\alpha(x_{k-1} \Longrightarrow x_k),$$

for $\alpha \in \{d, u\}$. We say that $e_d(D)$ is the d-control word and $e_u(D)$ is the u-control word associated with D.

A computation D such that $e_d(D) = e_u(D)$ is called *coherent*.

When $|e_d(D)| = |e_u(D)|$ (where $|x|$ is the length of the string x), we say that D is a *fair computation*.

We denote by $L_c(\gamma)$ and $L_f(\gamma)$ the languages of the strings in $\binom{V}{V}_\rho^*$ that are obtained by a coherent complete computation and, respectively, by a fair complete computation in γ. Clearly, each coherent computation is also fair.

By definition, $L_\alpha(\gamma) \subseteq L(\gamma)$, for all $\alpha \in \{p, b, pb, c, f\}$.

We denote by SL, PSL, BSL, $PBSL$, CSL, FSL the families of languages of the form the form $L(\gamma)$, $L_p(\gamma)$, $L_b(\gamma)$, $L_{pb}(\gamma)$, $L_c(\gamma)$, $L_f(\gamma)$, respectively, defined as above. (By REG and RE, we denote the families of regular and recursively enumerable languages, respectively.)

We now begin our investigations concerning the generative capacity of sticker systems. We will first show that many of the basic variants yield only regular languages.

Then, we show that each regular language can be represented as a weak coding of a language generated by a sticker system of one of these basic types. (A *weak coding* is a morphism $h : V_1^* \to V_2^*$ such that $h(a) \in V_2 \cup \{\lambda\}$ for all $a \in V_1$. If $h(a) \in V_2$ for all $a \in V_1$, then h is called a *coding*.)

We give some technical results whose proofs may be found in Kari et al. [15].

Lemma 5.1 $SL \subseteq REG$.

Lemma 5.2 $PSL \subseteq REG$.

Lemma 5.3 $BSL \subseteq REG$.

Lemma 5.4 $PBSL \subseteq REG$.

Modulo a weal coding, the opposite inclusions are also true.

Lemma 5.5 *Every regular language can be represented as a weak coding of a language in* $SL \cap PSL \cap BSL \cap PBSL$.

For a family F of languages, we denote by $wcode(F)$ the family of languages of the form $g(L)$, for $L \in F$ and g a weak coding.

Theorem 5.1 $REG = wcode(SL) = wcode(PSL) = wcode(BSL) = wcode(PBSL)$.

Thus, the Adleman way of computing cannot transgress the power of finite automata.

Next, we give our main result: the family CSL is computationally universal, in the sense that $wcode(CSL) = RE$. Here, we denote by RE the *recursively enumerable language*.

From the definitions, it is clear that all languages generated by sticker systems are context-sensitive. Moreover, from the *Turing-Church thesis* we have the following lemma:

Lemma 5.6 $wcode(CSL) \subseteq RE$.

In view of the fact that, at the first sight, the operation of prolongation to the right based on matching symbols related by a complementarity relation does not look very powerful, the following result is rather surprising.

Lemma 5.7 *Every recursively enumerable language can be represented as a weak coding of a language in the family* CSL.

Lemma 5.8 *For each recursively enumerable language* $L \subseteq T^*$, *there exist two* $\lambda-$*free morphisms* $h_1, h_2 : \Sigma_2^* \to \Sigma_*^1$, *a regular language* $R \subseteq \Sigma_1^*$, *and a projection* $\Sigma_1^* \to T^*$ *such that* $L = h_T(h_1(E(h_1, h_2)) \cap R)$.

From Lemmas 5.6 and 5.7 we get:

Theorem 5.2 $RE = wcode(CSL)$.

For the fair computations we have:

Theorem 5.3 $REG \subset wcode(SL) \subset RE$.

Gari et al. work is considered to be important in that computability of DAN computers is explored based on formal language theory.

We could extend sticker systems in various ways. One of the interesting extensions is *probabilistic sticher systems*. They use probabilistic treatments of axioms and rules, and may be usful for DNA computing.

5.3 Artificial Bacteria

Artificial bacteria are artificially synthesized bacteria and are interpreted as wetware. *Bacteria* refer to unicellular organisms with prokaryotic cells, which are single cells that do not have organelles or a true nucleus and are less complex than eukaryotic cells. Bacteria have an asexual mode of reproduction, called *binary fission*. Namely, a single bacterium divides into two daughter cells.

Bacteria are alive, though unusually shaped and reproducing slowly. However, their cells operate by means of a set of biological rules, producing familiar proteins with a reconstructed genetic code.

There are bacteria that can cause illnesses. They are responsible for many infectious diseases However, not all bacteria are harmful to humans. There are some bacteria which are beneficial.

We also know the harmful agent called *virus*. Viruses are microscopic biological agents that invade living hosts and infect their bodies by reproducing within their cell tissue. There are differences between bacteria and viruses.

A virus is not a living organism and can only grow and reproduce in the cells of a host. On the other hand, bacteria are single-celled organisms that produce their own energy and can reproduce on their own.

Advances in the Genome Project have made it possible to decipher the genomes of many organisms. However, it was not obvious whether a complete genetic system could be reproduced through chemical synthesis. If this were possible, it would be possible to synthesize bacteria purely artificially.

In May 2010, a team from the Craig Venter Institute in the United States succeeded in synthesizing an artificial bacterium that can self-replicate; see Gibson et al. [13]. The Craig Venter Institute is a research institute founded by Craig Venter. Furthermore, Venter is known as a researcher in genome research.

Their goal was to synthesize the smallest cells containing an essential genome. Actually, they linked together DNA to synthesize artificial genomes. This artificial genome was modeled after the bacterium called the *Mycoplasma mycoides*, which has

few base sequences corresponding to genetic information and is easy to manipulate. Mycoplasma mycoides is a bacterium that exists in the bodies of livestock and causes diseases such as pneumonia in cattle.

They mimicked the bacterial genome and chemically synthesized the DNA fragments that make up the genome. They then combined this into one in coli and created an artificial genome. They then transplanted this artificial genome into the cells of another bacterium. The new genome can be thus designed by using a computer.

As a result, the bacteria transplanted with the artificial genome were transformed by the artificial genome and produced Mycoplasma mycoides proteins. In other words, the transplanted artificial genome worked, produced proteins, and began self-replication. Furthermore, the cells of this artificial bacterium had the expected phenotypic properties and were capable of repeating self-replication.

They called such cells the *synthetic cells*. The properties of synthetic cells are similar to those of all cells that are produced synthetically. Venter et al.'s results took more than 20 years to achieve.

But, their results were the first step toward the practical application of artificial bacteria. We believe that Artificial bacteria are the ultimate wetware and are expected to have a major impact on research into AL.

If their method can be generalized, it will be possible to synthesize a variety of bacteria by designing a certain type of genome and incorporating it into a microorganism.

Therefore, artificial bacteria will be useful in the development of new medicines in the future. Also, it is thought that the technology of artificial bacteria will be applied to elucidate the function of genes.

However, concerns about safety also arise. This is because we do not know how bacteria that never existed before will interact with other organisms. In particular, it affects human safety and the possibility of this occurring cannot be denied.

There is also the possibility that ethical debate regarding artificial bacteria will become more active. If artificial bacteria are recognized as AL, the difference between real life and AL will become smaller scientifically. However, as a practical matter, this opinions will be divided as to whether or not to accept it.

Furthermore, new problems have arisen in 2007, when researchers at the Craig Venter Institute applied for a U.S. patent on their technology for minimal bacterial genomes. Additionally, Venter's name does not appear to be on the patent application.

There are various criticisms regarding patent applications for artificial life. In fact, in response, Canadian civil society organization ETC Group immediately issued a statement criticizing the patent application.

Venter and his colleagues seem to be aiming to monopolize technology related to AL. So criticism may be justifiable. We hope that there will be no impact on the development of AL.

Recently, the use of AI, especially deep learning, plays an important role in handling artificial bacteria. For instance, MIT researchers have discovered a class of compounds that can kill a drug-resistant bacterium that causes more than 10,000 deaths in the United States every year.

They showed that these compounds could kill methicillin-resistant Staphylococcus aureus (MRSA) grown in a lab dish and in two mouse models of MRSA infection. The compounds also show very low toxicity against human cells, making them particularly good drug candidates; see Wong et al. [25].

A key innovation of the new study is that the researchers were also able to figure out what kinds of information the deep learning model was using to make its antibiotic potency predictions. This knowledge could therefore help researchers to design additional drugs that might work even better than the ones identified by the given model.

Degression

We are now tackling with various *computer virus* in our daily life. What is a computer virus? It is a type of *malware*, which includes viruses, spyware, ransomware, etc.

A computer virus is a type of malware that attaches to another program ike a document. Note that it replicates after we first run it on our system. For instance, you could receive an email with a malicious attachment, open the file unknowingly, and then the computer virus runs on your computer.

Therefore, a computer virus is harmful and can destroy data, slow down system, etc. In spite of the names, computer virus is not a wetware. A computer virus is a type of malicious software, or malware, spreading between computers and causes damage to data and software.

A computer virus gives many influences on our systems including personal computers and cellular phone. Its common sign includes the following. They make our system's running speed slow. It also causes unwanted pop-up windows appearing on our computer.

Is a computer virus AL? Ludwig argues that computer virus is really much closer to AL than anything else humans have produced so far; see Ludwig [18, 19]. This is because in view of the fact that such viruses have are seen as living things like the biochemical viruses.

His approach was to study the evolution of computer viruses and AL. This seems to be a unique idea. However, someone may object to his approach since computer virus has the features which can be found the work of AL.

Now, there are several means facing with computer viruses. For instance, scanning the files in our computer is important. The use of antivirus softwares is another. By the developments IT technology, the treatment of computer viruses is seen as crucial.

5.4 Clone

Cloning is a technique used by scientists to create exact genetic replicas of genes, cells, or animals. Organisms that have the same origin and uniform genetic information are called *clones*.

Cloning technology has advanced in recent years, and clones are also AL in a broad sense. Cloning is broadly defined to mean the duplication of any kind of biological material such as a piece of DNA or an individual cell.

For example, segments of DNA are replicated exponentially by a process known as *polymerase chain reaction* (PCR). it is a technique widely used widely in basic biological study. There are three different types of artificial cloning, i.e., gene cloning, reproductive cloning and therapeutic cloning.

Gene cloning produces copies of genes or segments of DNA. *Reproductive cloning* produces copies of whole animals. *Therapeutic cloning* produces embryonic stem cells for experiments, aiming at creating tissues to replace injured or diseased tissues.

Gene cloning, also known as DNA cloning, is a very different process from reproductive and therapeutic cloning. Reproductive and therapeutic cloning share many of the same techniques, but are done for different purposes.

Researchers routinely use cloning techniques to make copies of genes for their study. The procedure consists of inserting a gene from one organism, often referred to as foreign DNA, into the genetic material of a carrier called a vector.

Examples of vectors include bacteria, yeast cells, viruses or plasmids, which are small DNA circles carried by bacteria. After the gene is inserted, the vector is placed in laboratory conditions that prompt it to multiply, resulting in the gene being copied many times over.

There are many examples of cloning in nature. For example, cell division in unicellular organisms is considered to be an operation that produces clones. Plant cloning technology has been known for a long time. An example of this type is cuttings.

Animal cloning technology is also being researched, but it is more difficult than plant cloning technology. The history of cloning is very long. Reproductive cloning was originally carried out by artificial twinning, or embryo splitting, which was first performed on a salamander embryo in the early 1900s by German embryologist Spemann.

In the 1970s, by advancements of molecular biology, we could develop techniques that allowed scientists to manipulate cells and to detect chemical markers that signal changes within cells. Beginning in the 1980s mammals such as sheep were cloned from early and partially differentiated embryonic cells.

Here, we explain cloning of animals by reproductive cloning. Researchers remove a mature somatic cell, such as a skin cell, from an animal that they wish to copy. They then transfer the DNA of the donor animal's somatic cell into an egg cell, or oocyte, that has had its own DNA-containing nucleus removed.

Researchers can add the DNA from the somatic cell to the empty egg in two different methods. In the first method, they remove the DNA-containing nucleus of the somatic cell with a needle and inject it into the empty egg.

In the second method, they use an electrical current to fuse the entire somatic cell with the empty egg. In both processes, the egg can develop into an early-stage embryo in the test-tube and then is implanted into the womb of an adult female animal.

Ultimately, the adult female gives birth to an animal that has the same genetic make up as the animal that donated the somatic cell. This young animal is called

Fig. 5.8 Dolly and her surrogate mother

a clone. Reproductive cloning may require the use of a surrogate mother to allow development of the cloned embryo, as in the most famous cloned organism, Dolly, the sheep.

In 1996, Scottish scientists Wilmut and his co-workers cloned the first animal, a sheep named Dolly. Which was cloned using an udder cell taken from an adult sheep [26] (cf. Fig. 5.8). Since then, scientists have cloned several animals including cows, cats, deer, horses, and rabbits.

Besides cattle and sheep, other mammals that have been cloned from somatic cells include: cat, deer, dog, horse, mule, ox, rabbit and rat. In addition, a rhesus monkey has been cloned by embryo splitting. However, they still have not cloned human.

The reason is probably that it is difficult to produce a viable clone. Many scientists believe that it can never be safe. For instance, the U.S. has no federal law on human cloning, but several states, dozens of countries, and international agreements prohibit it.

In 2005, the United Nations passed a nonbinding Declaration on Human Cloning that calls upon member states "to adopt all measures necessary to prohibit all forms of human cloning inasmuch as they are incompatible with human dignity and the protection of human life".

United Nations passed a nonbinding Declaration on Human Cloning that calls upon member states "to adopt all measures necessary to prohibit all forms of human cloning inasmuch as they are incompatible with human dignity and the protection of human life".

In spite of such situations, cloning of animals has the potential applications for medicine and agriculture. For example, by cloned animals, we can produce the milk relevant to human. Other possible use of cloned animals is for testing new drugs and treatment strategies. These applications may have the potential drawbacks for human health.

Someone want to seek other applications. For example, it would be interesting in having their deceased pets cloned in the hope of getting a similar animal to replace the dead one. It may be desirable for pet owners.

We must also watch the trend of human cloning. However, it is believed that it can never be safe. Also observed is that the U.S. has no federal law on human cloning. Currently, cloning is not always connected with AL, but someone will find interesting connections.

5.5 Artificial Cells

Defining life has been challenged since Schrödinger's [21] book "What is Life?", giving one answer.

Schrödinger gave the following remarks about cells:

> The growth of an organism is effected by consecutive cell divisions. Such a cell division is called mitosis. It is, in the life of a cell, not such a very frequent event as one might expect, considering the enormous number of cells of which our body is composed. (p. 22)

His remark is very instructive for life science as well as AL. This is because it explains essential features of cells.

A (natural) *cell* is the most basic unit of life. Although cells are small, they are complicated. Cells have different components likes the cytoplasm, ribosomes, and genetic information in the form of DNA.

Artificial cells are constructed to imitate natural cells. It is important to study biological processes and the origin of life. The construction methods for artificial cells are both top-down or bottom-up ways, and have achieved great development over the past decades; see Buddingh and van Hest [6].

Artificial cells are constructed to imitate natural cells and allow researchers to explore biological process and the origin of life. The construction methods for artificial cells are by top-down or bottom-up approaches.

Synthetic biology takes two different stances. The first is to create a minimal cell either by genome-editing of a natural organism, i.e., top-down approach. The second is to construct from individual building blocks a structure that mimics the essential aspects of a natural cell, i.e., bottom-up approach.

As for the top-down approach, the Venter Institute have used it to knock out genes in a living organism to a minimum set of genes; see, e.g. Gibson et al. [13]. As for the bottom-up approach, it creates a cell by assembling a stack of non-biotic component; see, e.g., Liu [17].

In 2010, the Craig Vender Institute succeeded in creating a replicating strain (named Mycoplasma laboratorium) of Mycoplasma mycoides using synthetically created DNA deemed to be the minimum requirement for life which was inserted into a genomically empty bacterium; see Gibson et al. [13].

The work to create the first synthetic bacterial cell was not easy, and took this team approximately 15 years to complete.

In fact, they had to develop new tools and techniques to construct large segments of genetic code, and learn how to transplant genomes to convert one species to another.

Gibson et al., gave the following in Discussion:

> If the methods described here can be generalized, design, synthesis, assembly, and transplantation of synthetic chromosomes will no longer be a barrier to the progress of synthetic biology. We expect that the cost of DNA synthesis will follow what has happened with DNA sequencing and continue to exponentially decrease. Lower synthesis costs combined with automation will enable broad applications for synthetic genomics.

Seen from the quotation, we can understand that their work opened a new door for possible uses of artificial cells.

In 2021, Xu et al. developed artificial cell-like structures by inorganic matter that autonomously ingest, process, and push out material—recreating an essential function of living cells; see Xu et al. [27].

They proposed to create the so-called *cell mimic*, which has potential applications ranging from drug delivery to environmental science. Xu el al. remarked that their approach provide a blueprint for developing the next generation of smart materials, autonomous micromachinery and artificial cell-mimics.

The cell mimics can perform active transport tasks by autonomously capturing, concentrating, storing, etc. Liu et al. work gave the worth of cell mimic. Actually, cell mimics are studied by many researchers, and are applied to several issues.

It is clear that artificial cells have many applications. In particular, they are applied to medicine. Artificial cells are very important in the drug delivery system. Thus, they are used for therapeutics in hospitals.

As explained above, artificial cells are a non-living synthetic design developed to mimic biological functions and modulate them by means of a minimalistic setup. With relatively simple engineering principles, artificial cells are equipped with a membrane and membrane encapsulated functional core.

Including biosensing and drug delivery applications, artificial cells are designed and developed in various ways as portable bioengineering devices able to communicate and respond to the surrounding.

An artificial cell has huge potential in future biological applications. But, they currently lack customized membrane options and selective membrane permeability. Progress in membrane engineering might be able to circumvent this limitation in the near future. This would accelerate artificial cell applicability in biotechnology.

For future developments of artificial cells, it is important to ensure that life-like systems are provided with an effective metabolism to sustain the biomimetic processes performed within the compartment.

Furthermore, strategies should be designed allowing the replication of the genetic information and the membrane components. This is of use for designing artificial cells for medical applications.

Concluding this chapter, wetware can be considered as one of the promising technologies for AL, In fact, we can insist that wetware is a point of contact between life science and AL, and future research should be focused on it. However, it also involves other difficult problems.

Furthermore, DNA computers are expected to develop as one of the important computational models for AL. Also observed is the connection with AI, since the ideas in AI are applicable for DNA computers. But, a lot of work for DNA computing should be done for advancing AL researches.

References

1. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science **266**, 1021–1024 (1994)
2. Adleman, L.: Computing with DNA. Sci. Am. **279**, 54–61 (1998)
3. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machine made of biomolecules. Nature **414**, 430–434 (2001)
4. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E.: An autonomous molecular computer for logical control of gene expression. Nature **414**, 430–434 (2004)
5. Bennett, C.: The thermodynamics of computation-a review. Int. J. Theor. Phys. **21**, 905–940 (1982)
6. Buddingh, B., van Hest, J.: Artificial Cells: Synthetic Compartments with Life-like Functionality and Adaptivity. Acc. Chem. Res. 769–777 (2017)
7. Chomsky, N.: Syntactic Structures. Mouton, Hague. (1957)
8. Chomsky, N.: On certain formal properties of grammars. Inf. Control. **2**, 137–167 (1959)
9. Cook, S.A.: The complexity of theorem proving procedures. In: Proceeding of the 3rd Annual ACM Symposium on the Theory of Computing, pp. 151–158. ACM, New York (1971)
10. Conrad, M.: On design principles for a molecular computer. Commun. ACM **28**, 464–480 (1985)
11. Feynman, R.: There's plenty of room at the bottom. In: Gilbert, D. (ed.) Miniaturization, pp. 282–296. Reinhold (1961)
12. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness, 1st edn. W. H. Freeman and Company, New York (1979)
13. Gibson, D., et al.: Creation of a bacterial cell controlled by a chemically synthesized genome. Science **329**, 52–56 (2010)
14. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Pearson, Boston (2006)
15. Kari, L., Paun, G., Rozenberg, G., Salomaa, A., Yu, S.: DNA computing, sticker systems, and universality. Acta Inform. **35**, 401–420 (1998)

16. Lipton, R.: DNA solution of hard computational problems. Science **268**, 542–545 (1995)
17. Liu, A.: The rise of bottom-up synthetic biology and cell-free biology. Phys. Biol. **16**, (2019)
18. Ludwig, M.: The Little Black Book of Computer Viruses. American Eagle Publications, Tucson, AZ (1991)
19. Ludwig, M.: Computer Viruses, Artificial Life and Evolution: What Computer Viruses Can Teach Us about Life. Tucson, AZ (1993)
20. Sakamoto, K., Gouzu, H., Komiyama, K., Kiga, D., Yokoyama, S., Yokomori, T., Hagiya, M.: Molecular computation by DNA hairpin formation. Science **288**, 1223–1226 (2000)
21. Schrödinger, E.: What is Life? The Physical Aspect of the Living Cell. Cambridge University Press, Cambridge (1945)
22. Shannon, C.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**(656–715), 623–656 (1949)
23. Shapiro, E., Benenson, Y.: Bring DNA computers to life. Sci. Am. **294**, 44–51 (2006)
24. Watson, J., Crick, F.: Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. Nature **171**, 737–738 (1953)
25. Wong, F., et al.: Discovery of a structural class of antibiotics with explainable deep learning. Nature (2023)
26. Wilmut, L., et al.: Viable offspring derived from fetal and adult mammalian cells. Nature **385**, 810–813 (1997)
27. Xu, Z., Huechel, T., Irvine, W.T., Sacanna, S.: Transmembrane transport in inorganic colloidal cell-mimics. Nature **597**, 220–224 (2021)

Chapter 6
Artificial Life Systems

Abstract This chapter presents some examples of AL systems. As typical examples, we overview Boids, Tierra, and L-systems.

6.1 Boids

Currently, various AL systems are being constructed. Here, we introduce the theory and practice of typical systems, namely Boids, Tierra, and L-systems. We have already introduced traditional theories such as cellular automata, but many new ideas are being incorporated into such AL systems.

Therefore, by experiencing the AL systems introduced below, you will be able to intuitively understand what AL really is. *Boids* is an AL program developed by Reynolds in 1986 that simulates the movement of a flock of birds; see Reynolds [12].

Boids means "bird-like". It is an AL based on *flocking* which is the act of individuals forming a group, and is a fundamental characteristic of the behavior of living things such as birds and fish.

Reynolds gave the following remarks about the motivations of Boids.

> This approach assumes a flock is simply the result of the interaction between the behaviors of individual birds. To simulate a flock we simulate the behavior of an individual bird (or at least that portion of the bird's behavior that allows it to participate in a flock). To support this behavioral "control structure" we must also simulate portions of the bird's perceptual mechanisms and aspects of the physics of aerodynamic flight. If this simulated bird model has the correct flock-member behavior, all that should be required to create a simulated flock is to create some instances of the simulated bird model and allow them to interact. (p. 25)

We believe that the system to simulate flocking can be considered AL. Boids is the first flocking-based AL system. Boids also provided the basis for the new idea of *swarm intelligence*. It seems to have had a major impact on AL systems such as swarmbots, which will be introduced in Chap. 7.

We saw Boids' demonstration at an international conference more than twenty years ago and were surprised to its simulation of the behaviors of birds. If we had not heard that it was an AL system, we would have thought it was a video of actual

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Akama, *Artificial Life*, Intelligent Systems Reference Library 262,
https://doi.org/10.1007/978-3-031-74422-8_6

flock of birds. In addition, we were surprised to the fact that Boids was built on an extremely simple principle.

The behaviors that make up the flocking model are stated in terms of "nearby flockmates". Boids is based on the flocking model in which three types of rules are used: These rules describe the behavior of boids.

- separation
- alignment
- cohesion.

Separation means that a boid is separated to other nearby boids. In other words, boids fly so as not to collide with other boids.

Alignment means that a boids acts in the same direction as the nearby boids. That is, a boids flies towards the same place as the other boids.

Cohesion means that a boids behaves approximately equidistant from nearby boids. In other words, a boid flies together with other boids in flocks.

These are illustrated as Figs. 6.1, 6.2 and 6.3. The central boid moves in the direction of the thick line.

Here, the central boids moves away from others which are too close.

Here, the central boids changes direction the same as the other boids.

Here, the central boids is moved to a position equidistant from the other boids (black circles).

Note that each boids recognizes its own position in the overall landscape, but flocking also considers its own neighborhoods, as in Fig. 6.4.

The neighborhood is determined by the distance from the central boid and the angle to the direction of flight.

Note that boids outside the local neighborhood are ignored. Therefore, by the above three principles and neighborhoods, boids can be grouped. Neighborhoods

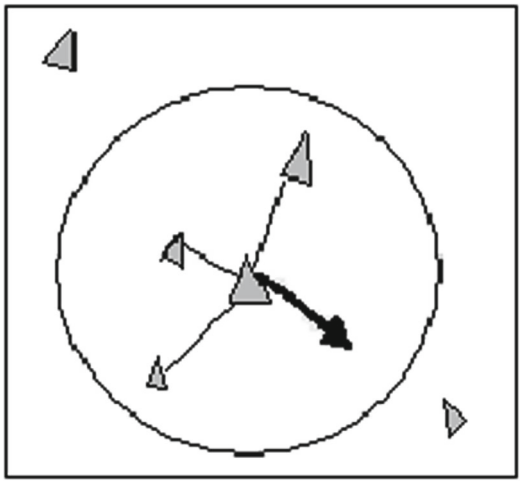


Fig. 6.1 Separation

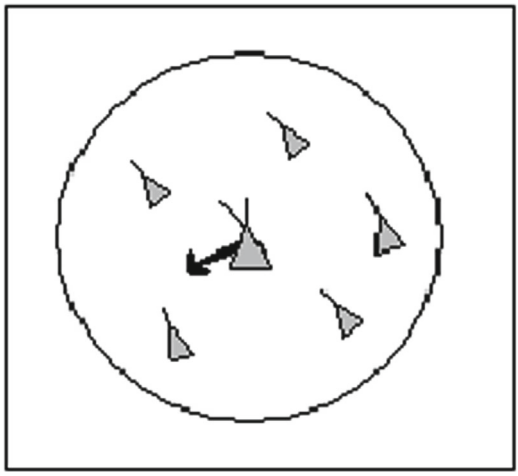


Fig. 6.2 Alignment

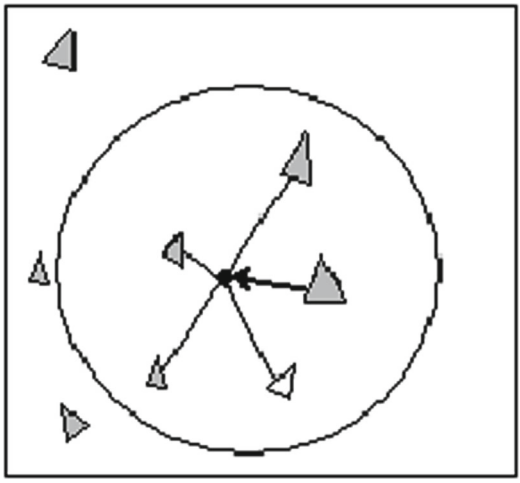


Fig. 6.3 Cohesion

can be thought of as a limited model of perception, but more rigorous formalizations are possible that can influence the behavior of boids.

Running the Boid Java applet in Fig. 6.5, you can see boids flying in flocks, like real birds (www.red3d.com/cwr/boids). Here, you can see that the three boids are grouped.

Boids is an AL system based on a simple principle, and we would think of it as swarms of birds. However, there are also some criticisms of Boids. First, Boids does not have a quantitative model.

Its movement is controlled by the position of boid. However, other factors should also be considered. For example, depending on the number of boids, that is, the size of the swarm, the movement of boids is likely to change.

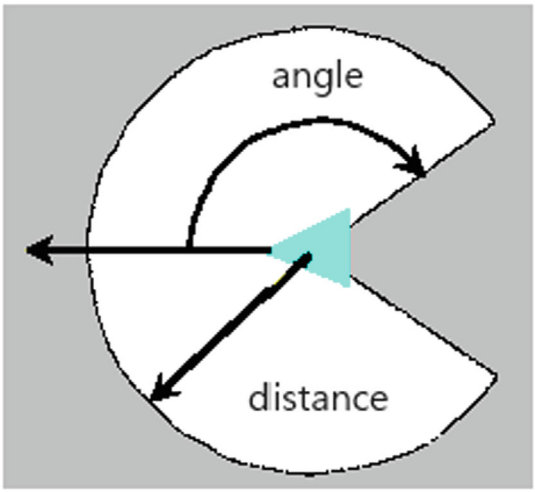


Fig. 6.4 Neighborhoods

Fig. 6.5 Result of Boids

Second, it is not clear when flocking will occur. The processing of Boids is only done as an algorithm by means of three types of flocking. However, it seems that actual birds behave in response taking into account the temporal factors.

Third, Boids does not seem to be able to handle swarms with multiple types and sizes. In nature, different species of birds form flocks in relation to other species of birds. In other words, behavior is thought to change depending on the given environment.

Renolds' work presented a model of polarized, noncolliding aggregate motion, such as that of flocks, herds, and schools. The model is based on simulating the behavior of each bird independently. Working independently, the birds try both to stick together and avoid collisions with one another and with other objects in their environment.

Artificial animals are now often called *animates*, which include physical robots and simulations. The term animat was coined by Wilson; see Wilson [13]. Thus, there is a sense in which Boids is a kind of animat using a simulating the swarm of birds.

6.2 Tierra

Tierra is an AL system developed by Ray; see Ray [9–11]. From both ecological and evolutionary views, Tierra was implemented whose details will be explained below. Thus, Tierra can also be interpreted as a model of AL. Observe that Tierra means the earth in Spanish.

Tierra is a system that simulates the dynamics and evolutionary processes of an ecosystem and proliferates in the memory of a computer. Consequently, Tierra has similarities with evolutionary computation.

However, the difference with evolutionary computation is essential. In Tierra, the so-called fitness function is not given externally, but is built in internally.

We turn to the structure of Tierra. It is a virtual parallel computer equipped with an operating system (OS). The OS is simulated by a computer. Therefore, when Tierra is executed, the program proliferates. Each proliferating program gains its own memory and CPU time.

Here, the memory is considered as a cell. Cells can read and execute all the instructions in the program, but can only write to themselves and their children. Tierra's OS then executes the instructions of each cell of memory. As a result, cells evolve through proliferation and mutation.

Therefore, Tierra is considered to be an AL in which organisms live in the memory of a computer and works with the help of OS. The structure of Tierra is shown in Fig. 6.6.

Here, cells are independent programs (gray areas). If a cell writes to itself or child cells, then cells in memory, i.e., programs, evolve like organisms. In Tierra, the program is described by a special machine language. It is a simple language consisting of 32 instructions.

The OS controls the survival of each organism through time sharing. The processor for time sharing is called a *slicer*. Also, the OS kills from the old cells when the memory is full. The processor for this is called a *reaper*. The information of cells is saved as data. The processor for this is called *genebank*.

In Tierra, a program corresponds to genome and an instruction to amino acids, respectively. And the energy for activity is provided as CPU time. That is, the basic

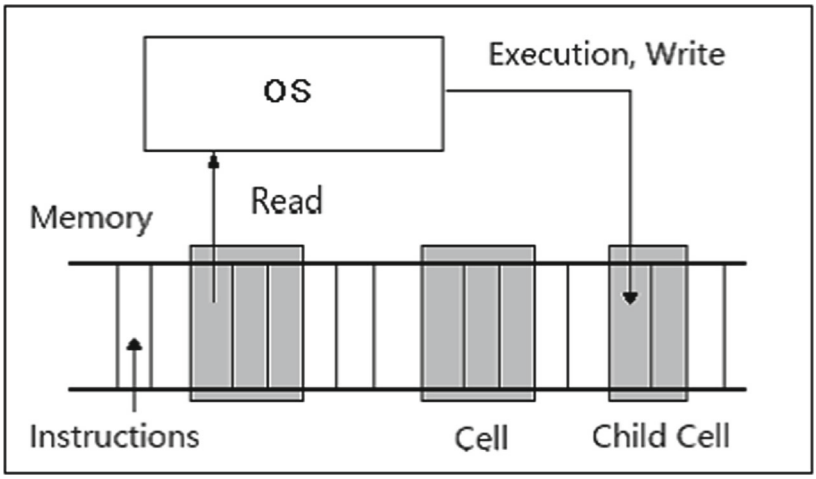


Fig. 6.6 Structure of Tierra

elements of Tierra's life are similar to those of other living organisms. The evolution is modeled in the world of programs.

The simulation begins with the execution of a single program, i.e., host, consisting of 80 instructions. And then, the process of self-propagation is started. As seen from the evolution of normal organisms, besides self-propagation mutations are also performed. In other words, when the host is executed, it evolves in different ways.

In addition, Tierra also supports *parasites*. The parasite replicates itself using the ancestor replication procedure. Parasitic organisms can also be parasitic by *hyper-parasites*. When a parasite attempts to utilize a hyper-parasite, the hyper parasite causes the parasite to replicate the hyper-parasite.

If parasitic organisms parasitize, then immune cells also occur and gradually increase. This means that the competition for survival of the theory of evolution will also take place in Tierra.

A system that implements Tierra on Mac is known as *MacTierra* by Fraser [5]. MacTierra was developed in 1994, but is independent of Ray's original Tierra.

Figures 6.7, 6.8 and 6.9 demonstrates processes of evolution in Tierra from Ray [11]. Figure 6.7 shows the initial state.

Here, hosts, red, are very common. Parasites, yellow, have appeared but are still rare.

Here, hosts, are now rare because parasites have become very common. Immune hosts, blue, have appeared but are rare.

Here, immune hosts now dominate memory, while parasites and susceptible hosts decline in frequency. The parasites will soon be driven to extinction.

Tierra simulates the process of unicellular organisms, and is one of AL systems. Therefore, it can be considered as a new model of evolutionary computation. Tierra

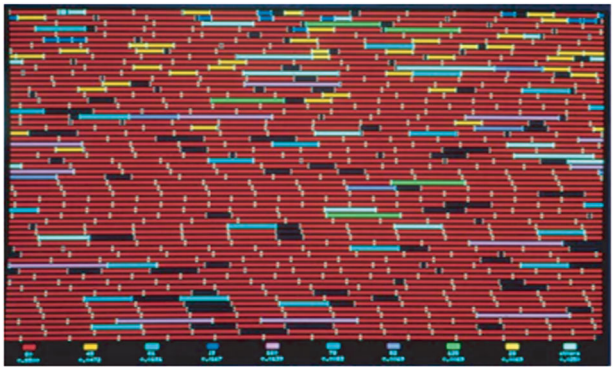


Fig. 6.7 Tierra (1)

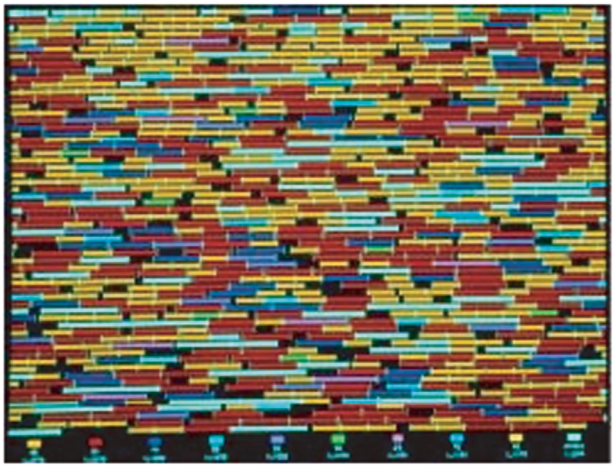


Fig. 6.8 Tierra (2)

presents a more sophisticated model than other theories of evolutionary computation, but there are also possibilities for some extensions.

First, an organism consisting of multiple cells can be introduced. This allows you to simulate the evolution of complex organisms which include multicellular organisms.

Second, we could handle different kinds of organims. In Tierra, there is no carnivorous/herbivorous distinction, but in nature these two species exist.

Third, cooperation with CG (Computer Graphics) will be necessary. In Tierra, evolution is only displayed by distribution. By using CG, more friendlier system could be built.

Finally, we can point out several difficulties with Tierra. For example, Tierra is an experimental AL system, and should be expanded by considering some points

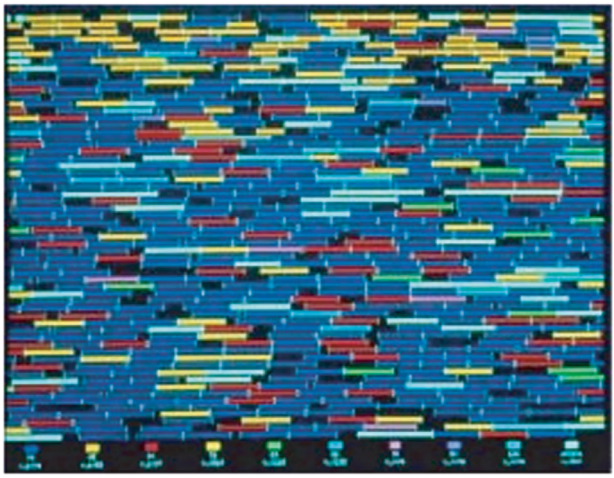


Fig. 6.9 Tierra (3)

mentioned above. However, as Tierra language is too restricted, we cannot simulate all that have happened in the history of life on earth.

An AL system similar to Tierra is *Avida*; see Lenski et al. [6]. Abida was developed by Lenski et al., and is strongly influenced by Tierra.

6.3 L-System

The *L-system* is a theory proposed by the Hungarian biologist Lindenmayer to mathematically formalize the growth process of organisms like plants, etc.; see Lindenmayer [7] and Prusinkiewicz and Lindenmayer [8].

Here, the "L" in "L-System" is the initial letter of "Lindenmeyer". The L-System is also called the Lindenmayer System. In the sequel, we give a quick review of L-system following Prusinkiewicz and Lindenmayer [8].

The L-system is a theory based on *formal grammar* due to Chomsky [2], and can be interpreted as a *rewriting system* with parallelism. Consequently, the L-system is more powerful and general than standard formal grammars.

Prusinkiewicz and Lindenmayer [8] addressed the features of the L-system:

> The essential difference between Chomsky grammars and L-systems lies in the method of applying productions. In Chomsky grammars productions are applied sequentially, whereas in L-systems they are applied in parallel and simultaneously replace all letters in a given word. This difference reflects the biological motivation of L-systems. Productions are intended to capture cell divisions in multicellular organisms, where many divisions may occur at the same time. Parallel production application has an essential impact on the formal properties of rewriting systems. (p. 2)

From this, the L-system can be considered to be not just a variant of Chomsky's grammatical theory, but a grammatical theory with mechanisms for dealing with living things.

In addition to biological growth processes, the L-systems can also be applied to describe the *iterative function system* (IFS), such as *fractals*. IFS is a theory introduced for the realization for defining fractals; see Barnsley [1].

The L-system as a basic theory of AL is obviously one of the recent research topics of AL. Unlike conventional theories, the L-system is based entirely on symbolic ideas. For example, the L-system can be used to mathematically describe the growth process of life.

Lindenmeyer had been studied the growth patterns of various dried algae for many years. He devised the L-system to formally describe multicellular tissues. At that time, the formal grammar theory proposed by Chomsky has attracted a great deal of attention, and from such a situation the L-system has been originated as a grammatical theory.

The L-systems are defined as the grammar $\langle V, S, \omega, P \rangle$, where V is a set of variables, S is a set of constants, ω is the axiom (the variable defining the initial state, i.e., starting symbol, and P is a set of rewriting rules, which is also called production rules.

A *rewriting rule* replaces the variable a with the string b of variable in the form

$$a \to b$$

where $a \in V, b \in V \cup S$.

A rewriting rule is also called a *production rule*. The formalization of the L-system is basically the same as the formalization of Chomsky's formal grammar. Thus, it is possible to use the L-system for recursively generating infinite patterns by repeated applications of rewriting rules.

The L-systems were introduced in 1968 by Lindenmayer [7] as a theoretical framework for studying the development of simple multicellular organisms. This seemed to be a starting point of L-systems.

They were also applied to plant models for investigating higher plants and plant organs. Later, they allow the use of Computer Graphics for realistic visualization of plant structures and developmental processes.

For example, the L-system for describing the development process of cells are defined as $\langle V, S, \omega, P \rangle$. Here, $V = \{a, b\}, S = \emptyset, \omega = a, P = \{a \to ab, b \to a\}$. It belongs to the simplest class of L-systems which are deterministic and context-free, called *DOL-systems*.

The development process is described by repeatedly applying the production rules of P with ω as the starting symbol:

$$
\begin{aligned}
&n = 0 : a \\
&n = 1 : ab \\
&n = 2 : aba \\
&n = 3 : abab \\
&n = 4 : abaaba
\end{aligned}
$$

Here, the rewriting rule $a \to ab$ represents that the cell a divides into the cells a and b. In addition, $b \to a$ indicates that the cell b grows and becomes a divisible cell a.

The ideas of L-systems can be extended for dealing with various graphics including fractals and CG, animation. For instance, Sherpinski's gasket is defined as the L-system $\langle V, S, \omega, P \rangle$.

Here, $V = \{a, b\}, S = \{+, -\}, \omega = a, P = \{a \to b - a - b, b \to a + b + a \to\}$, and $+, -$ denote left and right turn with the angle 60°, respectively. By repeated applying ω to rewriting rules, we obtain the following:

$$
\begin{aligned}
&n = 0 : a \\
&n = 1 : b - a - b \\
&n = 2 : a + b + a - b - a - b - a + b + a \\
&n = 3 : b - a - b + a + b + a + b - a - b - a + b + a - b - a - b \\
&\qquad -a + b + a - b - a - b + a + b + a + b - a - b
\end{aligned}
$$

To date, many AL systems using the L-system have been developed. Of course, due to historical reasons, many of them deal with many plants. Among them, the *Virtual Laboratory* (vlab), developed at the University of Calgary in Canada, is particularly famous; see Federl and Prusinkiewicz [4].

The basic idea of Virtual laboratory is to use the L-systems to model plants and generate various graphics such as fractals. In addition, Virtual Laboratory has an interactive environment and does not require advanced knowledge about particular living things to use them.

Virtual laboratory is currently commercialized and used all over the world. Therefore, Virtual laboratory can also be applied to AL and CG. However, it may be better to think of it as an environment for AL research rather than an AL life system.

Figure 6.10 is the screenshot if Virtual Laboratory.

The Virtual Laboratory has several features. First, it has a consistent organization. Experiments are run by applying tools (programs) to objects (data files).

Second, it has inheritance of features. It is often the case that several objects differ only in details.

Third, it has version control. Interaction with an object during experimentation may result in a temporary or permanent modification. In the latter case, the user is able to decide whether the newly created object replaces the old one or should be stored as another version of the original object.

Fourth, it has interactive manipulation of objects. It provides a set of general-purpose tools for manipulating object parameters.

Fig. 6.10 Virtual laboratory

Fifth, it has flexibility in conducting experiments. The user may apply tools to objects in a dynamic way while an experiment is being conducted.

Sixth, it has guidance through the laboratory. A hypertext system imposes a logical organization on the set of objects, provides a textual description of the experiments, and makes it possible to browse through the experiments in many ways.

Figure 6.11 visualizes the growth of a chrysanthemum called wall lettuce.

Finally, we discuss several extensions of L-systems. We know that the original L-system is the so-called the *OL-System*, which is a (context-free L-system). This implies that basically only the processes corresponding to the context-free grammar can be described by the original L-system.

However, the growth process of living organisms varies greatly depending on the type. In order to describe these cases, various extended L-systems are required by accommodating to required types.

In fact, there are several extensions of L-systems. The *statistical L-system* uses a rewriting rule with the addition of a probabilistic component; see Eichhorst and Savitch [3].

That is, its rewriting rule is in the form $a \rightarrow_p b$, in which a is rewritable to b with the probability p. In a stochastic context-free L-system (S0L-system), every symbol may have one or more rewriting rule, each with an associated probability of selection.

The *context-sensitive L-system* allows a rewriting rules to be applied in specific contexts; see Prusinkiewicz and Lindenmayer [8]. That is, its rewriting rule is of the

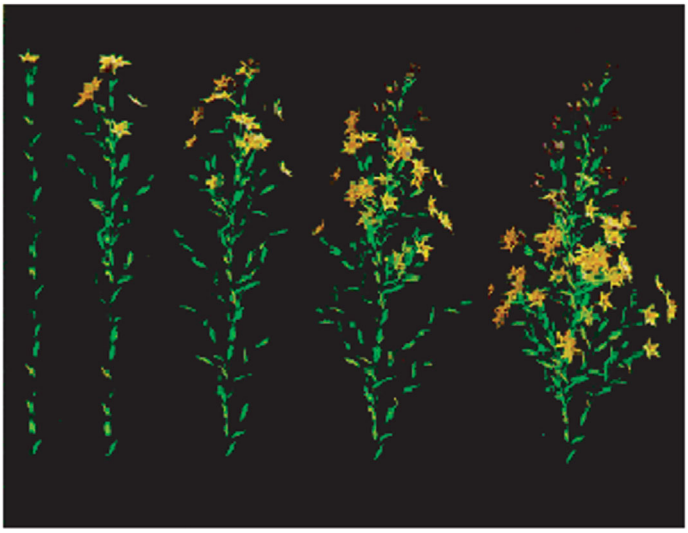


Fig. 6.11 Growth of wall lettuce

form $a_l\langle a\rangle a_r \rightarrow_p b$, in which a is a_l on the left, If there is a_r on the right, it will be rewritten as b.

Finally, we address some problems with L-systems. The L-system is a theory that describes processes in symbolic operations, but the growth of organisms is modeled in an iterative function system. In terms of practicality, the latter is better.

Then, the question arises as to whether there is an iterative function system corresponding to any L-system. If the solution to this problem is positive, the application of L-systems to graphics will be further expanded. However, this issue seems currently unresolved.

L-systems can describe processes iteratively and in parallel through rewriting rules. However, it is difficult to predict the original symbol sequence from the final symbol sequence. This represents the shortcomings of the L-system, because the process of evolution is almost always expected from current organisms.

For example, the form of an old organism can be identified from that of an organism today. Therefore, a new method for efficiently determining the initial symbol sequences from the generated symbol sequences is required. Such a method seems to be impossible only by pure grammatical formalization.

But, we believe that L-systems may be also helpful other media including music and video. For instance, L-systems can be used for music composition. If advanced stages of AL should do activities including music composition, then L-systems colud serve as one of the promised methods.

References

1. Barnsley, M.: Fractals Everywhere. Academic Press, San Diego (1988)
2. Chomsky, N.: Three models for the description of language. IRE Trans. Inf. Theory **2**, 113–124 (1956)
3. Eichhorst, P., Savitch, W.: Growth of functions of stochastic Lindenmayer systems. Inf. Control. **45**, 217–228 (1980)
4. Federl, P., Prusinkiewicz, P.: Virtual Laboratory: an interactive software environment for computer graphics. Proc. Comput. Graph. Int. **93–100**, 1999 (1999)
5. Fraser, S: MacTierra. smfr.org/work/sfi/mactierra/tierra/intro.html (1994)
6. Lenski, R., Ofrica, C., Pennock, R., Adami, C.: The evolutionary origin of complex features. Nature **423**, 139–145 (2004)
7. Lindenmayer, A.: Mathematical models for cellular interaction in development I: filaments with one-sided inputs. J. Theor. Biol. **18**, 280–289 (1968)
8. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer, Berlin (1990)
9. Ray, T.: Evolution and optimization of digital organisms. In: Billingsley, K., et al. (eds.) Scientific Excellence in Supercomputing, pp. 489–531. The Baldwin Press (1991)
10. Ray, T.: An evolutionary approach to synthetic biology: Zen and the art of creating Life. Artif. Life **1**, 179–209 (1994)
11. Ray, T.: tomray.me/tierra (the Tierra home page)
12. Reynolds, C.: Flocks, herds and schools: a distributed behavioral model. Comput. Graph. **21**, 25–34 (1987)
13. Wilson, S.: The animat path to AI. In: Meyer, J.-A., Wilson, S. (eds.) From Animals to Aimats 1, pp. 15–21. MIT Press, Cambridge, Massachusetts (1991)

Chapter 7
Prospects of Artificial Life

Abstract This chapter shows prospects of AL systems. We discuss several problems with AL. We also consider new trends of AL in connection with cybernetics, complex systems, agents and avatar. Finally, we provide remarks on some open problems.

7.1 Problems with Artificial Life

This book tries to introduce the main theories of AL and some specific examples of AL systems. Clearly, the field of AL is advancing rapidly, and it is expected to be applied to various fields. However, AL is also faced with many problems to be discussed.

7.1.1 Some Important Problems

Let's consider some of the problems associated with AL. One of the most important problems is about the **Problem 1**.

Problem 1
Is AL a life?

According to the general definition usually understood, AL is artificially synthesized life. In addition, wetware is life using existing life, but both soft Alife and hard Alife are purely AL.

Many AL systems satisfy the basic conditions described in Chap. 1. However, the theories and implementations of these AL systems are diverse. Therefore, the essential question of whether AL is life is raised.

In any case, AL is distinguishable from non-living by physical processes. For instance, hard Alife is interpreted in such a way that the base physical processes of AL can be simulated by a universal computer.

The criterion that life is a process is a natural interpretation of AL as a scientific theory, and this criterion is also implicitly assumed in biology.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Akama, *Artificial Life*, Intelligent Systems Reference Library 262,
https://doi.org/10.1007/978-3-031-74422-8_7

It is also true that there are various objections to recognizing hard Alife as a life. For example, the question arises why physical processes are necessary. If we accept this standard, then it follows that AL is a computer system. Considering the primacy and mystique of life, this conclusion may be unacceptable.

However, that any physical process is simulated by a universal computer is considered as the common sense in computer science. Of course, even in medicine and biology, living organisms are treated as processes.

And the basis of biological processes are molecules (or atoms). Therefore, it can be said that real life is interpreted as a simulation of physical processes. This means that the computational models of physical processes are diverse.

For example, DNA computers use DNAs which are the basic element of life. That is, their computation is performed according to the molecule itself and its properties. Then, the system actually developed by the DNA computer meets the standards of life.

Some people discussed the standard properties of a life. For instance, Farmer and Bellin [4] gave a list of properties that we associate with life.

1. Life is a pattern in spacetime rather than a specific material object.
2. Life has an ability of self-reproduction, if not in the organism itself, at least in some related organisms.
3. Life has the information storage of a self-representation
4. Life has a metabolism which converse matter and energy from the environment into the pattern and activities of the organism.
5. Life has functional interactions with the environment.

For Tierra, Emeche [3] points out that (2) and (4) are not fully satisfied. In fact, Tierra's self-replication is a formality, not consuming matter. Also, there is the question of whether metabolism should be considered as just a change in electrical state. Furthermore, stability in Tierra is weak in the sense of the distribution of programs. Of course, these points are not to reason that Tierra is not an AL.

However, Emeche's analysis defines an intuitive definition of AL. Obviously, the central issue of current research on AL is the construction of AL system as hard Alife.

Therefore, as long as the process is the essence of life, it can be said that there is no significant difference between real life and AL.

Farmer and Bellin's [4] work is very important for AL. They said:

> The advent of artificial life also has deep philosophical implications. It prompts us to reexamine our anthropocentric views and raises numerous questions about the nature and meaning of life. In addition, the study of artificial life may help us to understand, guide, and control the emergence of artificial life on earth, thereby averting a potential disaster and perhaps helping to create beautiful and beneficial new life-forms instead. (p. 817)

The following Problem 2 is closely related to Problem 1.

Problem 2
Is AL a machine?

If you affirm Problem 1, you will be faced with Problem 2. Here, a machine could be replaced with a computer. This is because a universal computer can be interpreted as a machine.

From the *Church–Turing principle*, any computable function can be computed by a universal computer. And in computer science, a process is considered a finite concrete operation that realizes a certain task.

Physical processes are processes that encompass processes in computer science accepted in computer science. However, the generally stated Church–Turing principle can also be interpreted in physics. That is, we may replace a universal machine with a universal physical machine.

In fact, the physical extension of the Church–Turing principle is the starting point for the so-called *quantum computers*. Therefore, it would be possible to view AL as a machine.

Of course, this interpretation is based on the assumption that the computability of a universal physical computer is the same as that of a universal computer. The claim that AL and a real life is more than machines would require a new interpretation of the Church–Turing principle.

Also, it cannot be said with certainty that Turing computable is computable. This is because the Church–Turing principle is an empirical hypothesis, not a proven one. In fact, the possibility of a theory with more computational power than a universal Turing machine cannot be ruled out; see Copeland and Sylvan [2].

Note that a similar problem to Problem 2 is also discussed by Wiener. In 1948, Weiner published the title "Cybernetics" [17]. Cybernetics is a field that deals with living organisms and machines in a unified manner using the methods of communication engineering and control engineering, and gives a great influence on current engineering.

Wiener's theory is also notable in that it was built before the birth of computers. And the 1950 book that explained cybernetics to ordinary person also encompassed the interpretation that man is a machine [18].

Therefore, cybernetics also seems to be the conceptual basis of AL and AI. However, Wiener did not discuss a life in detail as described above.

Problem 3
Does AL have intelligence?

Many forms of life are clearly considered to have some kind of intelligence. Human intelligence is indisputable, but intelligence in other forms of life is not. All animals may be intelligent, but whether plants and microorganisms are intelligent will be controversial because we do not know how to verify them.

However, it can be said that some organisms that exist today had some intelligence, so they were able to evolve by winning the competition for survival. If we assume that all life is intelligent, then artificial life should also be considered intelligent. So, what is the intelligence of artificial life? Furthermore, does the degree of intelligence differ depending on the form of artificial life?

As already mentioned, there is a field called AI, and its theory has been established to some extent. The difference between AL and AI is that the former is bottom-up, but the latter top-down.

In general, the object of AI is human intelligence. Therefore, the challenge is to have computers perform intellectual activities. Consequently, advanced AL has intelligence. Only with intelligence AL can be called a life.

However, this issue does not seem to be discussed much. The reason is that there is not active exchange between researchers in both fields.

In addition, the research stage of AL may not be at a level where this issue should be considered. However, we cannot deny that there is a great connection between AL and AI.

For example, the application of neural networks and evolutionary computation in both fields seems to speak for itself. In AI, if the applicable fields are limited, it is possible to build fairly sophisticated AI systems such as an *expert system*.

Therefore, it may be necessary to think that the level of intelligence of AL also varies depending on the form of life.

The next problem concerns the origin of AL.

Problem 4
How does AL emerge?

There are various theories about the origin of life in nature. The most promising theory is based on Darwin's theory of evolution. In other words, at first, simple life forms such as microorganisms appeared in some form, and various evolutions were repeated, resulting in the existence of many organisms. Such processes can also be applied to AL.

Here, "some form" is very important. Considering that the components of life are molecules, it is self-evident that there are various possibilities for the process of emergence.

For example, most organisms, including human, contain structures called *mitochondria*, which are organelles within eukaryotic cells that produce adenosine triphosphate. Mitochondria help sustain life by breathing and producing energy in cells.

Similarly, the origin of AL should be considered. Should we think of the origin of AL as a subsystem of cellular automata or a Turing machine? In any case, AL is an artifact created by human thought.

By analogy with life in nature, current AL is considered to be an evolution of cellular automata and Turing machine. It is, in a sense, correct. This is because these two theories are mathematically rigorous formalization of processes.

Therefore, the origin of AL is thought be when a specific theory was proposed. In fact, all scientific theories evolve by a lot of research. Unlike the birth of life in nature, the origin of AL is created by humans.

If we accept the theory of evolution, then AL life will also evolve. However, the current AL systems are unlikely to have evolved on their own. Of course, AL life is artificial, so such an interpretation may be commonplace.

Evolutionary computation is a theorization of the ideas of evolution, not the evolution of AL life itself. If there is a distinction between AL and life in nature, it may be possible to construct an evolutionary theory of AL.

Since the evolution of AL is artificial, it may therefore be called AL. It is undeniable that the problem of the origin and evolution of AL involves various philosophical problems that cannot be solved by simple discussions.

If AL is realized in the future, the next Problem 5 will inevitably be discussed by us.

Problem 5
Can AL coexist with real life?

In the nature, in addition to animals such as human, ecosystems are built with plants and microorganisms. And now, human coexists with other living things. However, this can be said from a human standpoint, and it is doubtful whether this is really the case.

In many countries, for example, traditional ecosystems have collapsed in recent years. As a result, some valuable animals and plants are facing extinction. However, it should be worrying that the problem of alien species is occurring due to human responsibility. This is because non-native species have a significant impact on existing lives.

When AL constitutes a part of a living organism, life in nature, especially human, must coexist with AL. Already, some wetwares coexist with human, and such AL contributes to the maintenance of human life and life.

However, this situation does not always continue. For example, malicious human could use AL. In addition, if AL life has intelligence, there will be a competition for survival with humans.

AL is like an alien species that we have created. The problem of alien species is also becoming more acute in nature, so we can get one lesson. One can only hope that in the books of AL decades later, Problem 5 is not one of the main themes but solved one.

7.1.2 Definitions of Artificial Life

Here, we give arguments of the definition of AL related to Langton [9]. Its review part, Langton described the state of investigations of AL by 1987.

> Artificial Life is the study of man-made systems that exhibit behaviors characteristic of natural living systems. It complements the traditional biological sciences concerned with the analysis of living organisms by attempting to synthesize life-like behaviors within computers and other artificial media.
>
> By extending the empirical foundation upon which biology is based beyond the carbon-chain life that has evolved on Earth, Artificial Life can contribute to theoretical biology by locating life-as-we-know-it within the large picture of life-as-it-could-be biology by locating life-as-we-know-it within the larger picture of life-as-it-could-be.

From these passages, we can interpret Langton's basic stance towards AL. The following is about the historical developments of biology.

> Biology is the scientific study of life-in principle anyway. In practice, biology is the scientific study of life based on carbon-chain chemistry. There is nothing in its charter that restricts biology to the study of carbon-based life; it is simply that this is the only kind of life that has been available for study. Thus, theoretical biology has long faced the fundamental obstacle that it is difficult, if not impossible, to derive general theories from single examples.
>
> Certainly life, as a dynamic physical process, could "haunt" other physical material: the material just needs to be organized in the right way. Just as certainly, the dynamic processes that constitute life - in whatever material bases they might occur- must share certain universal features - features that will show us to recognize life by its dynamic form alone, without reference to its matter. This general phenomenon of life - life writ-large across all possible material substrates - is the true subject matter of biology.
>
> Without other examples, however, it is extremely difficult to distinguish essential properties of life - properties that must be shared by any living system in principle - from properties that are incidental to life, but which happen to be universal to life on Earth due solely to a combination of local historical accident and common genetic descent.
>
> Since it is quite unlikely that organisms based on different physical chemistries will present themselves to us for study in the foreseeable future, our only alternative is to try to synthesize alternative life-forms ourselves - Artificial Life: life made by man rather than by nature.
>
> Whereas biology has largely concerned itself with the material basis of life, Artificial Life is concerned with the formal basis of life. Biology has traditionally started at the top, viewing a living organism as a complex biochemical machine, and worked analytically down wards from there - through organs, tissues, cells, organelles, membranes, and finally molecules - in its pursuit of the mechanisms of life.
>
> Artificial Life starts at the bottom, viewing an organism as a large population of simple machines, and works upwards synthetically from there - constructing large aggregates of simple, rule-governed objects which interact with one-another nonlinearly in the support of complex, global dynamics.

The following passages are very important in that they address the concept of emergent behavior. Obviously, both emergent behavior and bottom-up behavior are crucial for AL.

> The "key" concept in AL is emergent behavior. Natural life emerges from out of the organized interactions of a great number of non-living molecules, with no global controller responsible for the behavior of every part. Rather, every part is a behavior itself, and life is the behavior that emerges from out of all of the local interactions among individual behaviors.
>
> It is this bottom-up, distributed, local-determination of behavior that AL employs in its primary individual behaviors. It is this bottom-up, distributed, local-determination of behavior that AL employs in its primary methodological approach to the generation of life-like behavior.

Based on Langton's review described, he gave the first definition of AL as follows:

Langton's Definition 0

Artificial Life is Life made by man rather than by nature.

We need to interpret this naive definition. This is to explore what it means to be "made by man rather than by nature". "Artificial" generally means "made by

man rather than by nature", so **Definition 0** seems to be plausible. However, it rases another more fundamental question, i.e., "What is Life?".

There are several definitions of life in related areas. On biological perspective, we can show **Definition 1**.

Langton's Definition 1
Artificial Life is Life man-made animate objects.

Definition 1 reveal the definition of AL in which *manifest functions* (MF) holds. By manifest function, we mean the anticipated and intended goals of an action or social structure; the reason something is done.

The term "made-thing" has two meanings; i.e., (1) taking separate things and put them together to make another kind of thing, (2) a single thing divides into several identical things.

Langton's Definition 2
Artificial Life is Life made in sense (1) rather than in sense (2).

Based on **Definition 2**, Langton could reach the final definition.

Langton's Definition 3
Artificial Life is natural life, where "nature" is understood to include, rather than to exclude, human beings and their artifacts.

We think that **Definition 3** is appropriate as a definition of AL. Langton provided the following in the end of his article:

> Hence, finally, a major goal of Artificial Life should be to work toward removing "Artificial Life" as a phrase that differs in meaning **in any fundamental way** from the term "Biology".
>
> Although they are currently viewed, and pursued, a very different scientific endeavors. I believe that the day will come when science will **finally** acknowledge that which Darwin gave us the key to understand.
>
> Human beings, and all that they do, are a part of Nature.

Indeed Langton's new definition of AL is not conclusive, and we will be able to seek a new definition of AL related to its developments.

7.2 New Trends

The theoretical foundations of current AL are cellular automata, neural networks, genetic algorithms, etc. Biochemical methods are also used in wetware. However, depending on the diversity of life, the scope of AL research must be further expanded.

In addition to constructing a new theory, existing theories must be reevaluated. It is also necessary to conduct research from an interdisciplinary perspective. In the following, we discuss the new trends of AL, but they are not exhaustive.

We take up the following:

- Cybernetics
- Complex system
- Swarm bot
- Avatar
- Virtual Reality (VR)

7.2.1 Cybernetics

As already mentioned, *Cybernetics* proposed by Wiener [17, 18] refers to a field that studies organisms and machines in a unified manner by means of several engineering methods.

Therefore, cybernetics can offer theoretical foundations of any "system". However, due to the complexity and novelty, it has not received much attention in recent years.

The purpose of cybernetics clearly has something in common with AL. For example, *cyborg* is classified as an AL system in which the ideas of cybernetics are fully incorporated.

This is because a cyborg means (fictional) person whose physical abilities are beyond normal human limitations by mechanical elements built into the body.

Furthermore, cybernetics can be used to interpret not only life but also society, which is a collection of living things, as a "system". In fact, cybernetics can describe the development and evolution of any system including various living things.

Cybernetics is itself an almost established theory. As is well known it covers many traditional disciplines, including technology, philosophy, biology, mathematics, and social sciences.

Therefore, studying and understanding it will involve all aspects of life. By evaluating cybernetics from different perspectives, it may be possible to construct a unified theory of AL.

7.2.2 Complex Systems

We turn to *complex systems*. Unfortunately, we cannot simply interpret the field. We here view a complex system as a system which is difficult to model and predict. The area studying complex systems is generally called *complex systems science*.

Many natural and artificial systems are clearly considered as complex systems. In this sense, a living organism is a complex system and its study is closely related to AL. However, because the field of complex systems need interdisciplinary work, there are many possibilities of complex systems for applications to AL.

Complex systems generally satisfy properties such as non-determinism, non-tractability, distributivity, self-organization, and emergence, which are also characteristic of AL.

Non-determinism means that the behavior of the system is not deterministic. Therefore, the behavior of a system will not uniquely determined from past information.

Non-tractability means that we grasp the components of a system, but we cannot fully grasp its overall behavior. Conventional systems are formalized based on tractability.

Distributivity means that the information and representation of a system are distributive. That is, all functions of the system cannot be localized.

Self-organization means that a system autonomously generates organizations and structures. Clearly, it is an essential property of living organisms.

Emergence is to characterize the hierarchies of complexity of a system and one hierarchy of a system is irreducible to another. Therefore, the whole complex components of a system cannot be reduced to simple ones.

Complex systems have attracted a lot of attention in recent years. In a sense, living organisms such as human are complex systems. Therefore, it is thought that AL can be treated in a unified manner by using various methods of complex systems.

The following basic theories of complex systems are known (cellular automata have been already explained in Chap. 2).

- Cellular automata
- Fractals
- Chaos

Fractals

Fractals are figures with self-similarity, which were introduced and coined by Mandelbrot in 1975; see Mandelbrot [12]. Note that in Latin "fractus" means "broken". Geometry that studies fractals is called *fractal geometry*.

Many physical systems in nature and human organons have shapes that cannot be captured by ordinary Euclidean geometry. For example, the shape of the coastline is complex, but it is known that it has the same complex shape as expanding it.

Examples of such fractal figures include shapes such as mountains, trees, blood vessels, and intestines, etc. Since these are also related to life, fractal theory could be applied to some aspects of AL.

However, at present, fractal theory deal only with complex shapes. It is necessary to clarify not only the shape of life, but also the self-similarity of various properties. The linkage between fractal theory and AL should also be studied.

Chaos

Chaos is a phenomenon that occurs according to deterministic rules in dynamical systems but is interpreted irregularly in probabilistic systems. The name of "chaos" is due to Lorenz in the 1960s in the context of chaos theory.

Chaos is characterized by the following features:

- In chaos, small differences in initial conditions can make a big difference in results in the future.
- In chaos, prediction of long-term outcomes is impossible.
- Chaos does not have periodicity.
- Chaos has non-linearity
- In chaos, the Lyapunov exponent is greater than 0.
- Chaos has self-similarity.

Lyapunov exponent is a value representing an exponential increase in error, which allows the degree of unpredictability to be quantified. That is, when the Lyapunov index is greater than 0, chaotic property is expressed.

Chaos is found in many phenomena not only in natural sciences but also in social sciences. Conventionally, linear equations and differential equations have been considered effective for modeling scientific phenomena.

However, modeling chaos requires modeling with more complex non-linear equations. Non-linear equations are also studied in the field of mathematics.

The history of chaos research is not so long, but it is about to be established as a general theory that deals with unpredictable and complex phenomena. Chaos theory is an important theory of complex systems and is also expected to have a major impact on the study of AL.

For example, *logistic mapping* is used as a growth model in ecology, and is the starting point of chaos theory. The American biologist May studied the population of organisms using the difference system of logistic equations; see May [13, 14].

Logistic mapping is generally used in population biology to map the population at any time step to its values at the next time step. We can find additional applications including:

1. Genetics: change in gene frequency
2. Epidemiology: fraction of population infected
3. Economics: relationship between commodity quantity and price
4. Social Sciences: number of people to have heard a rumor

The *logistic equation* is the basis of the population model and is an equation of the form:

$$\frac{dx}{dt} = (b - cx)x.$$

where $x(t)$ is the number of populations at time t. b and c are constants, $b - cx$ represents the population growth rate, and cx represents the restraining force proportional to the number of populations.

Unlike human, dogs, cats, and others, insects are born from eggs. When they are born, their parents are dead and not in this world. Therefore, generations do not overlap. In this case, time can be measured in generations.

Here, we can consider differentiation in the limit of the time width of 0, and since the generation interval cannot be 0, the formula for modeling the ecology of living

organisms cannot be described by differential equations, but by difference equations. Note that the difference equation is an equation that discretizes the differential equation.

Now, the number of insects of generation n is $x(n)$. The rate of increase of insects in the next generation is (7.1).

$$x(n+1) = x(n) + bx(n) \tag{7.1}$$

In (7.1), if $a = 1 + b$.

$$x(n+1) = ax(n) \tag{7.2}$$

holds. If you know the number of insects in generation n, multiply it by a to get the number of insects in generation $n + 1$.

Converting input to output like (7.2) is called *mapping*, and a is said to be a *control parameter*.

For example, the logistic equation (7.3)

$$x(t) = \frac{bx(0)e^{bt}}{b + cx(0)(e^{bt} - 1)} \tag{7.3}$$

can be written as (7.4).

$$x(n+1) = (1 + (b - cx(n)))x(n) \tag{7.4}$$

Here, normalizing with the maximum value $x = (1 + b)/c$ of x and introducing $y(n) = x(n)/x$, the equation becomes (7.5).

$$y(n+1) = a(1 - y(n))y(n) \tag{7.5}$$

By the normalization, we have $0 \leq y(n) \leq 1$.

Because the maximum value on the right-hand side of (7.5) is $a/4$, $0 < a \leq 4$ holds.

$$f = a(1 - y)y \tag{7.6}$$

is called a *logistic mapping*, which is one of the models that simply generates chaos.

Equation (7.6) can be written in general as (7.7).

$$X_{n+1} = aX_n(1 - X_n) \tag{7.7}$$

If $0 \leq a \leq 4$, $0 \leq X_0 \leq 1$, then X_n can be interpreted as the number of organisms in the n generation, and a can be interpreted as the reproduction rate.

The behavior of the logistic mapping is heavily dependent on the value a.

- If $0 \leq a \leq 1$, then X_n converges monotonically to 0.
- If $1 < a \leq 2$, then X_n converges monotonically to $1 - (1/a)$.
- If $2 < a \leq 3$, then X_n converges, while, oscillating to $1 - (1/a)$.
- If $3 < a \leq 3.5699$, then X_n oscillates the periodic point of a power of 2.
- If $3.5699 < a \leq 4$, then the fluctuation of X_n is irregular and does not have a specific periodicity.

Here, 3.5699 is said to be *Feigenbaum point* and represents the boundary point of chaos.

Langton points out that the distinction between consistent and chaotic behavior plays a major role in evolution. It is in the range of variables λ, which evaluate the behavior of a cellular automaton called the *edge of chaos*.

When λ changes, the cellular automata exhibit a phase transition of behavior. This allows Langton to show the equivalence of computability of cellular automata and Turing machines; see Langton [8].

The term "edge of chaos" is due to Packard, Farmer etc., and has since been used to refer to an intermediate state between an order state and a chaotic state. That is, it is the space between, where the right balance exists between order and chaos, implying self-organization, innovation and creativity may be possible.

In fact, systems that are close to the edge of chaos behave in a complex manner. Thus, the relationship between AL and the edge of chaos is very important.

Kauffman mathematically shows that the rate of evolution is greatest near the edge of chaos; see Kauffman [5]. His result means that the biological world can also be regarded as a complex system.

Kauffman's claim may be that life cannot be explained by our current laws of physics. This suggests that his theory is based on molecules that reproduce only collectively, called autocatalytic sets. Therefore, we think that his theory is influential to AL in view of complex systems.

Next, we discuss a new robotic concept called *swarm-bots*, which is swarms of self-assembling artifacts. The idea of swarm-bots was proposed in the early 2000s by Dorigo etc. [15]. Later, based on the model of robots behaving in groups, the swarm-bots project started.

The project, that lasted 42 months, was successfully completed on March 31, 2005. Since October 1, 2006, the Swarmanoid project is extending the work done in the Swarm-bots project to three dimensional environments.

Boids is an AL system that models swarm behavior, but swarm-bots is a robot with real hardware. Therefore, swarm-bots can be regarded as more realistic system than other AL systems.

Swarm-bots is thought to be a reduced ecosystem of natural organisms. Therefore, it is considered promising as a new approach to AL. In fact, it incorporates traditional AL and AI methods in a variety of ways.

In Swarm-bots, a relatively simple individual, that is, a robot, evolves and acts cooperatively. Therefore, Swarm-bots as a whole is considered as a life form. The robot used in the Swarm-bots project is shown in Fig. 7.1 (www.swarm-bots.org).

Fig. 7.1 Swarm-bots

Problems that cannot be solved by a single robot could be easily solved by multiple robots working together. *Distributed AI* aims to solve problems by such an idea.

Swarm-bots and distributed AI are thought to be highly related. That is, individual intelligences work together like a single intelligence. Such swarmed intelligence is called *swarm intelligence*.

Swarm intelligence is one of the characteristics of swarms of organisms in nature. Therefore, swarm intelligence can be applied to AL in various ways. For example, the aforementioned Boids is an AL system that applies the simple principle of swarm intelligence.

Swarm-bots is also autonomously assembled and repaired. Therefore, Swarm-bots is more promising than other AL systems in that it is equipped both with hardware and software.

However, some problems have been pointed out with Swarm-bots. First, there is a problem of how to incorporate the functions of swarm intelligence into individual robots. Because Swarm-bots can be used at all levels in AL system, a corresponding formalization of swarm intelligence is necessary.

Second, if you think of Swarm-bots as an AL system, its architecture is not always clear. One question is whether the group of robots is uniform or diverse. In addition, there are several possibilities for how to control the robots that make up the swarm.

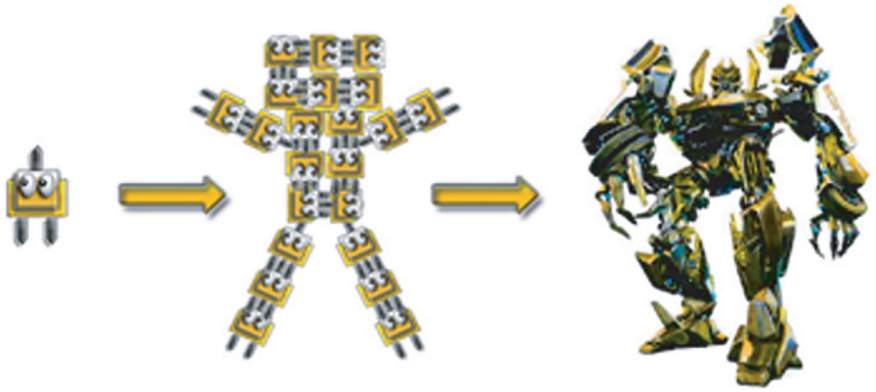


Fig. 7.2 Future of i-SWARM

Third, Swarm-bots require hardware and software. Therefore, it is necessary to determine specifications (for example, size, shape, etc.) to design the bots (robots). Also, since the swarm-bots is equipped with an OS, its function depends on the specifications of the OS. In order to make Swarm-bots a general-purpose technology, it is also necessary to standardize its OS.

A further development of the Swarm-bots is *i-SWARM*. which realizes an AL system that combines diverse robots, and it can be seen as more ambitious project.

In other words, i-SWARM tries to produce robots with many complex functions by evolving and grouping small robots. Here, these functions are conducted autonomously.

i-SWARM assumes that about 20 robots will become like one robot. Given the standard number of levels, the i-SWARM seems to be highly versatile. i-SWARM's robots are expected to be applied in a wide range of fields, from daily life to science and technology. Currently, the possibilities of its field of application are being studied. And the future of i-SWARM is reminiscent of the world of SF.

For example, Fig. 7.2 shows an image of the evolution of i-SWARM robots (www.i-swarm.org).

Here, the two on the left are the current i-SWARM levels. The third robot is the final form of i-SWARM. If it is in human form, it represents cyborg.

There are many potential merits of i-SWARM including greater flexibility and adaptability of the system to the environment, robustness to failures, etc. Their collective behavior opens up new application fields.

As found in Swarm-bots, I-SWARM, etc. needless to say that their methods are closer to AL than existing robotic technology. As described above, research on AL is underway by incorporating new ways of thinking.

It should be noted here that many ALs life strongly reflect von Neumann's philosophy and theory of cellular automata.

7.2.3 Avatars

Next, we simply discuss *avatars*. Although the concept is controversial to AL, we may be able to give some comments. While avatars are sometimes made to resemble the user, they also function present as a kind of play.

Examples include changing to a gender different from the real self or changing their personality to match their customized appearance to "pretend to be" another person. We could also find other problems.

Of course, this form of play and communication existed in Web before the advent of avatars, but with the advent of more visually appealing avatars, it became easier. Avatars obviously address the importance of "pretence" in digital worlds.

The history of avatars is closely connected to that of movies and games. In fact, the word was originated from "Avatar: The Way of Water" in the 2022 American movie. Avatars are often widely used for services provided by the company, such as being displayed on behalf of the user during chat.

For example, it is displayed in the screen of a webmail. Displayed on the Messenger screen. Appears on the bulletin board. In online games and SNSs, it is displayed as a character that briefly represents the participant. Avatars also gave many influences on the field like music, games and visual effects.

There are also some attempts to make new avatar with AI, i.e., avatar generators. Based on such applications, we can easily create avatars. Thus, in spite of its impacts on AI, avatars should also be discussed in the research on AL.

Indeed avators may be important for AL, but they lack theoretical basis. It was ignored for related fields. How do we formally interpret avatars? One attractive idea for realizing it is to use the theory of *counterfactuals*.

We know that the foundations for counterfactuals are now established both in philosophy and formal logic; see Lewis [10] and Stalnaker [16] for details. Their ideas use a *possible world* considering the similarities of actual worlds and counterfactual worlds (possible worlds) to deal with the semantics for counterfactual conditionals.

Probabilistic semantics can also be developed for counterfactuals. However, we know no serious work for theoretical foundations for avatars. It is thus expected to see some foundational studies for avatars in relation to AL.

7.2.4 Virtual Reality

Virtual reality (VR) is the research area, which uses computer modeling and simulation in which a person can interact with an artificial three-dimensional (3-D) visual or other sensory environment.

In this sense, avatar mentioned above is the subject of VR. Its applications enable the computer user to simulate reality through the use of interactive devices.

The idea of VR was already found in the 18th century. For example, many artists, entertainers and performers have tried to create imaginary worlds. For instance, *Illusion* is one of the techniques.

We can also point out that *panorama* is a visual art which has some connection with VR. Panora can be generated by paintings which are massive artworks that reveal a wide view of subjects including a landscape, military battle, historical event, etc.

The term VR was coined by Lanier in 1987; see Lanier [7]. VR has many applications for fields including education, entertainment, etc. At the period, some programming language like VRML, X3D for VR has been developed.

VRML (Virtual Reality Markup Language) was developed for 3D graphics in 1995. It is a file format used to create 3D interactive scenes and objects for the web. VRML scenes can be viewed using a VRML browser plug-in or a standalone VRML viewer.

Later, *X3D* has also developed in 1997. X3D is an improvement of VRML, and has a rich set of componentized features that can be used for the use in engineering and scientific visualization including CAD, architecture, animation, etc.

In the 1990s, VR gave the promise of a completely immersive experience just for games, but for completely reshaping the way we viewed the world. We knew that VR would allow us to see inside things.

Later, in relation to VR, the so-called *augmented reality* (AR) has been also studied. AR is the integration of digital information with the user's environment in real time. Unlike VR, which creates a totally artificial environment, AR users experience a real-world environment with generated perceptual information.

One of the most famous applications of AR is *heads-up-displays*(HUDs) used in military airplanes and tanks. Presently, AR is applied to several games by adding the information to players' viewpoints.

It is also applied to many applications in smart phone. displaying information such as building addresses, real estate signs, retail sales offers, etc. This enables us to intuitively understand reality by using smart phone.

We think that VR and AR have a possibility of advancing AL. VR can serve as a basis of creating a new artificial living things. AR can give us the way real and artificial organisms can be co-existed. However, no work has been done on the mentioned problems.

7.3 Open Problems

Finally, we will introduce the open problems of AL. In 2000, Bedau et al. raised 14 open problems; see Bedau [1]. They said:

> The following is a structured list of key open problems in artificial life. Some goals involve practical synthesis; others involve theoretical analysis. The aim is not necessarily to make each problem wholly independent of the others; some problems overlap or complement others, which may help to triangulate the deeper issues. (p. 364)

Some of these problems have already been touched upon, but still need to be studied. It consists of three major problems: A, B, and C, which are further subdivided in the sequel.

(A) How does life arise from the non-living?
- (1) Generate a molecular proto-organism in vitro.
- (2) Achieve the transition to life in an artificial chemistry in silico.
- (3) Determine whether fundamentally novel living organizations can exist.
- (4) Simulate a unicellular organism over its entire lifecycle.
- (5) Explain how rules and symbols are generated from physical dynamics in living systems.

(B) What are the potentials and limits of living systems?
- (6) Determine what is inevitable in the open-ended evolution of life.
- (7) Determine minimal conditions for evolutionary transitions from specific to generic response systems.
- (8) Create a formal framework for synthesizing dynamical hierarchies at all scales.
- (9) Determine the predictability of evolutionary consequences of manipulating organisms and ecosystems.
- (10) Develop a theory of information processing, information flow, and information generation for evolving systems.

(C) How is life related to mind, machines, and culture?
- (11) Demonstrate the emergence of intelligence and mind in an artificial living system.
- (12) Evaluate the influence of machines on the next major evolutionary transition of life.
- (13) Provide a quantitative model of the interplay between cultural and biological evolution.
- (14) Establish ethical principles for artificial life.

As noted by Bedau et al., the above list of challenges could be extended, of course. Some people may insist that some fundamental questions are missing. We think that the most notable is about the nature of life itself.

The above problems are classified as A, B, and C according to the level of abstraction in thinking about "life". In other words, A is concerned with "life", B with "system", and C with "world", respectively.

A is concerned with the relationship between living and non-living things, and is a problem that pursues the essence of living things. Therefore, these solutions will contribute to the development of wetware and DNA computing.

B is about the fundamentals of AL systems. Therefore, these are issues to be considered for the actual implementation of AL systems.

C deals with more general view of life in connection with mind, machines, and culture. These are the basis for the positioning of AL in the future.

The above 14 problems are clearly unsolved and conundrum. A comprehensive analysis of these can be said as follows. Conventional research on AL has mainly focused on B.

By advances of wetware like DNA computing, etc., we can recognize the importance of A and C. They raised many issues to be explored in the future. When AL reaches the practical stage, the importance of C will increase, but it does not seem to be much discussed at present.

Now, we analyze the above open problems. For (1), Bedau et al. claimed that Artificial life research has succeeded in creating both self-replicating (templating) molecules that show limited evolution, and self-replicating compartments (micelles, lipid aggregates).

It is thus necessary to work out the way of combining these ideas in an autonomous, evolvable, self-replicating autopoietic system. However, it seems to be non-trivial for developing such systems.

For (2), they said that few questions concerning living systems are as fundamental as the spontaneous generation of life, and a constructive context in which to explore this issue is artificial chemistry. To challenge (2), the computer technology and information theoretic foundations may play crucial role.

For (3), Artificial life calls special attention to the question of whether there can be forms of life that are fundamentally different from the familiar biological instances. This means that we need to explore varieties of forms of life mathematically as well as biologically.

For (4), it involves simulating an existing biological organism on the computer, and the goal is for all of the organism essential dynamics to emerge from a bottom-up simulation. To address the point, methods of statistics and complex systems should be incorporated.

For (5), a proper theory of evolving life must allow evolution to create new symbolic descriptions and rules to manipulate them. In general, these entities are formed by an underlying physical dynamics, and then evolution sharpens and reinforces this symbolic level.

We need to show how can such systems be modeled effectively where no state transition rules at the new symbolic level are specified in advance in contrast with current modeling practice.

For (6), the challenge concerns the contingency of life. Considering it, we face two related questions: (a) What are the features common to all evolutionary processes, or to broad classes of evolutionary processes? (b) Do different evolutionary processes contain fundamentally different evolutionary potential?

For (7), the challenge concerns the interplay between evolution and learning prior to mind (i.e., between hereditary and acquired information) in the evolutionary formation of response systems. Its solution involves investigating the evolution of artificial life models with short-term fluctuations in survival.

For (8), there are two problems to be resolved. The first problem is to create a formal framework for consistently describing hierarchical systems with some coarse-graining procedure for moving between levels. The second problem is how complex and robust functionalities are generated in biological (and protobiological) systems.

For (9), it is part of a larger issue involving the predictability of the effects of manipulating ecosystems of all sizes and kinds. Progress on the challenge should have an impact on guidelines for intervening in biological and technological contexts with evolutionary implications.

For (10), it is to develop a theory of information in evolving systems concerning with transmission, processing, and generation. For living systems, there appear to be two complementary kinds of information transmission.

One is the conservative hereditary transmission of information through evolutionary time. The other is transmission of information specified in a system's physical environment to components of the system.

For information processing, the unification with evolution is important and should be worked out. For example, information generation plays a crucial role during evolution.

For (11), it is to clarify the connection of life and mind. Two deep issues in this area arises for artificial life. The first is substantive: whether and, if so, how the natures of life and mind are intrinsically connected.

The second is methodological: whether it is most profitable to study mind and intelligence only when embodied in living systems. Obviously, the problem is closely related to theory and practices in AI.

For (12), it is to evaluate the role of machines on evolutionary transition of AL. Machines may well be the central players in the transition, as will be the case if autonomously evolving machines get established in the natural world.

Machines might also play a supporting role in the transition by providing an infrastructure that influences its rate or direction of change. They suggest to investigate the influence of machines in evolution of AL, because current AL systems are based on computer technology.

For (13), it is to give a quantitative model concerning the interplay between cultural and biological evolution. For example, cultural evolution can be found throughout the social sciences, including the development of economic markets.

Note also that psychological information transmission can involve speaking, writing, or non-linguistic forms of behavior.

For (14), it is to raise complicated ethical issues in AL. These issues fall into four broad categories: (a) the sanctity of the biosphere, (b) the sanctity of human life, (c) the responsible treatment of newly generated life forms, and (d) the risks of exploitation of AL.

AL's ethical issues somewhat resemble those concerning animal experimentation, genetic engineering, and AI. Several literatures on the ethical issues raised in those three fields may provide some interesting guidance for exploring the ethical issues in AL.

On the other hand, creating novel forms of life and interacting with them in novel ways will place us in increasingly uncharted ethical issues.

Bedau et al. said in Introduction in their paper as follows:

> As with Hilbert's problems, formulating a clear and fruitful challenge provides direction for, rather than replacing, shorter-term (e.g. five year) research goals, so the challenges reported

> here are unabashedly long term. However, the potential exists for some significant progress on at least the first ten in the next decade. (p. 364)

The quotation is about the research on the foundations of mathematics by Hilbert long time ago. However, the situation also is applied to AL research because the foundations of mathematics and AL have been progressed. In fact, AL is truly advancing day by day.

But, the analysis of open problems of AL is quite difficult, and little work has been done so far. One promising idea seems to analyze AL in relation to AI. Here, we want to address its significance.

In fact, AL and AI are viewed as very different fields. Thus, identifying a strong understanding of the similarities and differences between the two is very meaningful.

For instance, the problems in (A) above, i.e., How does life arise from the nonliving? shows some connections with AI. In particular, (A-3), i.e., Determine whether fundamentally novel living organizations can exist, is appropriate to AI.

All living things can be understood by using genetic information in DAN and RNA. However, some AL systems are developed by using software to realize "living things". This may involve the need of alternative representations of genetic information.

The problems in (C) above, i.e., How is life related to mind, machines, and culture?, are closely related to many ethical issues. AI also has similar ethical issues. For the case of AL, we can point out that there may be particular difficulties with ethical issues.

For example, the dignity of human life should be discussed in AL. Most people agree with the dignity of human life, but does not agree with the one of some inferior animals, How about AL? Do we consider the dignity of (computer) viruses? New life forms should be similarly discussed.

There seem to be some problems in AL to be discussed in relation to AI. We think that to solve these problems are very interesting for further developments of AL. However, the complete work of understanding AL in the context of AI is beyond the scope of the present book.

Finally, we want to give final remarks of the present book. Although it may not be long before the field of AL becomes an established branch of science, especially computer science, we are currently still faced with many problems.

Of course, we believe that most of the above open problems must be solved for further developments of AL. If we succeed in solving some of the above-mentioned open problems of AL, it will be obviously helpful to the future developments of the study of AL.

After we can properly verify these solutions, we will substantially have to rewrite the present book. This is our necessary task.

7.4 New Ideas from Related Areas

In Chap. 1, we introduce several areas closely related to AL. Since the 2020's, such areas gave rise to new ideas which are of special importance for AL (also AI). In the sequel, we discuss such ideas.

1. Neo-cybernetics
2. Transhumanism
3. New Pragmatism

The above ideas seem to be tentative, and they are mainly circulated in SNS. And they also play a role in AI, but contributions to AL are open. In spite of these situations, we give some words.

Neo-cybernetics is a new version of cybernetics for our ages. Although Wiener proposed cybernetic based on the idea human can be regarded as a machine, neo-cybernetics is based on the emerging multidisciplinary claim for navigating the systemic intricacies of our world.

According to Wiener, cybernetics is the scientific study of control and communication in the animal and the machine. In contrast, the subject of neo-cybernetics additionally deals with adaptive behavior.

By integrating systems science, design, ecological awareness and futures thinking, Neo-Cybernetics seems to be applied to rapidly evolving technologies. This means that neo-cybernetics is also be helpful for AL.

However, we are faced with many problems. Although cybernetics laid a robust foundation based, the theoretical basis of neo-cybernetics remains to be fully worked out.

Transhumanism is a philosophical claim that takes into account the current technology. Based on the developments of technology, humanism varies. It is also recognized that transhumanism broadens human potential.

Transhumanism refers to an optimistic belief in the enhancement of the human condition through technology in all its forms. Many supporters believe in fundamentally enhancing the human condition through applied reason and a corporeal embrace of new technologies.

We think that the starting points of AL and transhumanism are dully compatible. By this means, we have a possibility of sketching a new concept of Life (also Artificial Life), thus believing that transhumanism could serve as a conceptual basis for AL in future.

New pragmatism is a philosophical area that expands pragmatism whose main claim is that the meaning of words is a result of how they are used. As is well known, pragmatism has been developed by James. Periirce, etc. in USA in the early 20th century.

New pragmatism was proposed by Malachowski in 2010; see Malachowski [11]. It was also developed by Kolodko in 2022 in relation to economics; see Kolodko [6]. Approaches by both workers to new pragmatism appear to be similar.

In the context of AL, there is a sense in which several ideas in pragmatism and new pragmatism are considered as attractive. The fact seems to be very important for interpreting a life. Thus, we must incorporate new pragmatics' ideas for the research of AL.

Currently, we do not know whether the above ideas are of use for AL, although there is a promising possibility for AI. This means that it is worth studying these ideas in connection with foundational studies of AL.

Finally, we say that a life (natural or artificial) seems to be more mysterious than we have supposed. This means that there are many interesting problems to be worked out theoretically as well as practically. Therefore, the research of AL should be continued in relation to various areas including biology, philosophy, computer science, etc.

After completing the book, we obtained a surprising news. Hopfield and Hinton received the Nobel Prize in Physics 2024 for fundamental discoveries and innovations that enable machine learning with artificial neural networks.

As discussed before, their works are useful for AI as well as AL.

References

1. Bedau, M., McCaskill, J., Packard, N., Rasumussen, S., Adami, C., Green, D., Ikegami, T., Kaneko, K., Ray, T.: Open problems in Artificial Life. Artif. Life **6**(2000), 363–376 (2000)
2. Copeland, B., Sylvan, R.: Beyond the universal turing machine. Aust. J. Philos. **77**, 46–66 (1999)
3. Emmeche, C.: The Garden in the Machine. Princeton University Press, Princeton (1994)
4. Farmer, J., Belin, A.: Artificial Life: the coming evolution. In: C. Langton (ed.) Artificial Life II, pp. 815–840. Addison-Wesley, Redwood (1992)
5. Kauffman, S.: Origins of Order: Self-organization and Selection in Evolution. Oxford University Press, Oxford (1993)
6. Kolodko, G.: Political Economy of New Pragmatism. Springer, Cham (2022)
7. Lanier, J.: You Are Not a Gadget: A Manifesto, Google Books. Knopf, Alfred A (2010)
8. Langton, G.: Computation at the edge of chaos. Phisica D **42**, 12–37 (1990)
9. Langton, C.: A new definition of Artificial Life (1987). www.chairetmetal.com/cm03/intro.htm
10. Lewis, D.: Counterfactuals. Basil Blackwell, Oxford (1973)
11. Malachowski, A.: The New Pragmatism. Routledge, London (2010)
12. Mandelbrot, B.B.: Fractals: Form, Chance, and Dimension. Freeman, San Francisco (1977)
13. May, R.: Biological populations with nonoverlapping generations: stable points, stable cycles, and chaos. Science **186**, 645–657 (1974)
14. May, R.: Simple mathematical models with very complicated dynamics. Nature **261**, 459–467 (1976)
15. Sahin, E., Labella, T. Trianni, V., Deneubourg, J., Passe, P., Floreano, D., Gambardella, L., Mondana, F., Nolfi, S. and Dorigo, M.: SWARM-BOT: pattern formulation in a swarm of self-assembling mobile robots. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 1–6
16. Stalnaker, R.: A theory of conditionals. In: N. Rescher (ed.) Studies in Logical Theory, pp. 98–112. Blackwell, Oxford (1968)
17. Wiener, N.: Cybernetics. Wiley, New York (1948)
18. Wiener, N.: The Human Use of Human Beings. The Riverside Press (1950)

Index

A
Ackley, 8
Adaptation system, 55
Adaptive learning, 32
Adenine, 68
Adleman, 9, 69
Agent, 2
ALife in Art, 3
Alignment, 100
Amplification, 71
Android, 65
Animate, 103
Annealing, 70
Artificial bacteria, 90
Artificial cell, 95
Artificial Intelligence (AI), 12
Artificial Life, 1
Artificial Neural Network (ANN), 31
Association layer, 36
Associative memory, 41, 42
Augmented Reality (AR), 128
Automatic programming, 54
Automaton, 81
Avatar, 127
Avatar: The Way of Water, 127
Avida, 106
Axiom, 10

B
Back propagation networks, 39
Bacteria, 90
Balanced, 88
Barto, 62
Bedau, 128
Benenson, 81
Bennett, 7, 67
Big O natation, 73
Binary fission, 90
Biochemistry, 13
Biocomuputer, 66
Biological computer, 66
Blueprint, 18
Boden, 9
Boids, 8, 99
Boldwin, 61
Boldwin effect, 60
Boltzmann distribution, 46
Boltzmann machine, 8, 33, 44
Bottom-up theory, 11
Buddingh, 95
Burks, 18

C
Cell, 18, 95
Cell mimick, 10, 96
Cellular automaton, 5, 17
Central dogma, 68
Chaos, 121
ChatGPT, 10
Chomsky, 82, 106
Chomsky hierarchy, 82
Chromosome, 55
Chromosome encoding, 56
Church, 58
Church–Turing principle, 115
Clause, 76
Click, 68
Clone, 92

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

S. Akama, *Artificial Life*, Intelligent Systems Reference Library 262,
https://doi.org/10.1007/978-3-031-74422-8

Cloning, 92
Codd, 19
Coding, 89
Coherent, 88
Cohesion, 100
Colony, 27
Complex adaptive system, 3
Complex system, 120
Complex systems sciences, 120
Component, 2
Computational complexity, 73
Computer virus, 66, 92
Connectionism, 31
Connectionist models, 8
Conrad, 7
Contact network, 79
Content-addressable memory, 41
Context-free grammar, 82
Context-free language, 82
Context-sensitive gramma, 82
Context-sensitive language, 82
Context-sensitive L-system, 109
Control parameter, 123
Conuterfactual, 127
Conway, 6, 19, 22
Cybernetics, 5, 120
Cyborg, 120
Cytosine, 68

D
Darwin, 4
Deep learning, 49
Deoxyribonucleic Acid (DNA), 66, 67
Descartes, 1
Deterministic algorithm, 74
Distributed AL, 125
Distributivity, 121
DNA computer, 9, 66
DNA Doctor, 83
DNA polymerase, 70
DOL-system, 107
Dorigo, 124
Double-helix structure, 68
double strand, 76
Droplets, 3

E
Edge of chaos, 124
Emergency, 121
Emmeche-ch, 114
Enzyme, 70
Evolution, 53
Evolutionary computation, 6, 53
Evolutionary Programming, 54
Evolutionary strategy, 54, 61
Evolution scheme, 56
Evolvable, 4
Expert system, 116

F
Fair computation, 88
Fault tolerance, 32
Federl, 108
Feigenbaum point, 124
Finite automaton, 82
Finite cellular automata, 28
Finite state machine, 82
Fitness function, 56, 59
Flocking, 99
Fogel, 54
Folorarobotica, 3
Formal grammar, 82, 106
Formal language, 82
Formation rule, 10
Fractal, 107, 121
Fractal geometry, 121
Fraser, 104

G
Gale, 10
Game of Life, 6, 22
Gardner, 22
Gasket, 21
Genebank, 103
Genecloning, 93
Generative Artificial Intelligence, 10
Genetic Algorithm, 6, 54, 55
Genetic programming, 9, 54, 58
Genome, 67
Genotype, 56
Gibson, 96
Guanine, 68

H
Hagiya, 84
Hairpin, 9
Hamiltonian cycle problem, 74
Hamilton path Problem, 70
Hard Alife, 3
Heads-Up-Display (HUD), 128
Hidden layer, 34
Hinton, 8

Hobbes, 1
Holland, 6, 54, 55
Hopfield, 7, 33
Hopfield network, 7, 33, 41
Hyper-parasite, 104

I
Illusion, 128
Inheritance, 55
Initial structure, 59
Input layer, 34
Integer linear programming problem, 74
Intractable problem, 73
i-SWARM, 126
Iterated function system, 107

J
James, 133

K
Kari, 85
Kauffman, 124
Knapsack problem, 74
Kolodko, 133
Koza, 54, 58

L
Langton, 2, 19
Langton's loop, 7, 25, 26
Layer, 34
Learnable, 4
Lenski, 106
Lewis, 127
Life of Game, 19
Lindenmayer, 6, 106
Linear bounded automaton, 82
LISP, 58
Liu, 96
Living organism, 2
Living thing, 4
Logistic equation, 122
Logistic mapping, 122, 123
Loop, 19
L-system, 6, 106
Ludwig, 92
Lyapunov exponent, 122

M
Machine learning, 31, 38
MacTierra, 104
Malachowski, 133
Malware, 92
Mandelbrot, 121
Manifest function, 119
Mapping, 123
Massively parallel, 67
May, 122
McCarthy, 58
McCaskill, 128
McClelland, 33
McCulloch, 32
Melting, 70
Merge, 70
Minsky, 33
Mitochondria, 116
Model, 11
Molecular automata, 81
Molecular computer, 7
Mutation, 55
Mycoplasma mycoides, 90

N
Nardi, 28
Natural selection, 55
Neo-cybernetics, 133
Neural network, 5, 31
Neuron, 33
New pragmatism, 133
Newton, 11
Non-determinism, 121
Non-deterministic algorithm, 74
Non-tractability, 121
Nowlan, 61
Nucleic acid, 67
Nucleotides, 67
Nuro computer, 31

O
OL-System, 109
One-dimensional cellular automaton, 19
Order, 73
Output layer, 34
Owens, 54

P
Packard, 128
Panorama, 128
Papert, 33
Parallel Distributed Processing model, 8, 47

Parallel Distributed Processing (PDP), 31
Parasite, 104
PDP model, 33
Perceptron, 5, 32, 35
Perceptron Convergence Theorem, 38
Periirce, 133
Phenotype, 56
Philosophy, 13
Pitts, 32
$P = NP$ problem, 74
Polymerase chain reaction, 93
Possible world, 127
P problem, 74
Pragmatic view, 11
Premise, 10
Primitive, 88
Probabilistic cellular automata, 28
Probabilistic semantics, 127
Probabilistic sticker sytems, 90
Productionn rule, 107
Propositional (Boolean) satisfiability problem, 74
Prusinkiewicz, 106
Pushdown aautomaton, 82

Q
Quantum computer, 70, 115

R
Raggio, 28
Random sequence generator, 22
Rasumussen, 128
Ray, 103
Real time computation, 32
Reaper, 103
Rechenberg, 54
Recombination, 56
Recursively enumerable language, 89
Regular grammar, 82
Regular language, 82
Reinforcement learning, 62
Reproductive cloning, 93
Response layer, 36
Reversible cellular automaton, 19
Rewriting rule, 107
Rewriting system, 106
Reynolds, 99
Ribonucleic Acid (RNA), 67
Rosenblatt, 5, 32
Rule, 10, 20, 21, 90
Rule 30, 21
Rule 90, 20
Rumelhart, 33

S
Sahi, 124
Sakamoto, 9
Sarkar, 17
Satisfiable, 75
3-SAT, 9
SAT problem, 75
Schrödinger, 2
Schwefel, 54
Scientific theory, 10
Search space, 59
Sejnowski, 8
Selection, 55, 56
Self-organization, 32, 121
Self-regenerable, 4
Self-reproducing automaton, 17
Self-reproduction, 25
Semantic view, 11
Sensory layer, 36
Separation, 100
Set partition problem, 74
S-expression, 59
Shannon, 79
Shapiro, 9, 70
Slicer, 103
SNS, 133
Soft Alife, 3
Space complexity, 73
Stalnaker, 127
Sticker system, 85, 87
Stochastic L-system, 109
Strand, 76
Structure, 59
Supervised learning, 38
Sutton, 62
swarm-bots, 124
Swarm intelligence, 99, 125
Synapses, 33
Syntactic view, 11
Synthetic cell, 91
System, 12

T
Theorem, 10
Theory, 10
Therapeutic cloning, 93
Threshold, 37
Thymine, 68
Tierra, 103

Time complexity, 73
Top-down theory, 11
Totalistic cellular automaton, 20
Transhumanism, 133
Traveling salesman problem, 74
Truth assignment, 75
Truth-tables, 75
Turing-Church thesis, 89
Turing machine, 82
Two-dimensional cellular automaton, 19
Type 0 grammar, 82
Type 1 grammar, 82
Type 2 grammar, 82
Type 3 grammar, 82

U
Ulm, 17
Unit, 45, 47
Universal Constructor, 18
Unrestricted grammmar, 82
Unrestricted language, 82
Unsupervise learning, 38

V
Van Hest, 95
Venter, 10, 90
Virtual Laboratory, 108
Virtual Reality Markup Language (VRML), 128
Virtual Reality (VR), 127
Virus, 90
Von Neumann, 5, 17
Von Neumann neighborhood, 26

W
Walsh, 54
Watson, 68
Weak coding, 89
Web, 127
Wet ALife, 3
Wetware, 9, 65
Wiener, 5, 120
Wilmut, 9
Wilson, 103
Winther, 11
Wolfram, 7, 19
Wolfram class, 7, 20

X
X3D, 128