The Many Facets of Natural Computing

Lila Kari[0]
Department of Computer Science
University of Western Ontario
London, ON, N6A 5B7, Canada
lila@csd.uwo.ca

Grzegorz Rozenberg
Leiden Inst. of Advanced Computer Science
Leiden University, Niels Bohrweg 1
2333 CA Leiden, The Netherlands
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309, USA
rozenber@liacs.nl

"Biology and computer science - life and computation - are related. I am confident that at their interface great discoveries await those who seek them." (L.Adleman, [3])

1. FOREWORD

Natural computing is the field of research that investigates models and computational techniques inspired by nature and, dually, attempts to understand the world around us in terms of information processing. It is a highly interdisciplinary field that connects the natural sciences with computing science, both at the level of information technology and at the level of fundamental research, [98]. As a matter of fact, natural computing areas and topics come in many flavours, including pure theoretical research, algorithms and software applications, as well as biology, chemistry and physics experimental laboratory research.

In this review we describe computing paradigms abstracted from natural phenomena as diverse as self-reproduction, the functioning of the brain, Darwinian evolution, group behaviour, the immune system, the characteristics of life, cell membranes, and morphogenesis. These paradigms can be implemented either on traditional electronic hardware or on alternative physical media such as biomolecular (DNA, RNA) computing, or trapped-ion quantum computing devices. Dually, we describe several natural processes that can be viewed as information processing, such as gene regulatory networks, protein-protein interaction networks, biological transport networks, and gene assembly in unicellular organisms. In the same vein, we list efforts to understand biological systems by engineering semi-synthetic organisms, and to understand the universe from the point of view of information processing.

This review was written with the expectation that the reader is a computer scientist with limited knowledge of natural sciences, and it avoids dwelling on the minute details of various natural phenomena. Thus, rather than being overwhelmed by particulars, it is our hope that the reader will see this article as simply a window onto the profound relationship that exists between nature and computation.

There *is* information processing in nature, and the natural sciences are already adapting by incorporating tools and concepts from computer science at a rapid pace. Conversely, a closer look at nature from the point of view of information processing *can* and *will* change what we mean by computation. Our invitation to you, fellow computer scientists, is to take part in the uncovering of this wondrous connection.[1]

2. NATURE AS INSPIRATION

Among the oldest examples of nature-inspired models of computation are the **cellular automata** conceived by Ulam and von Neumann in the 1940s. John von Neumann, who was trained in both mathematics and chemistry, investigated cellular automata as a framework for the understanding of the behaviour of complex systems. In particular, he believed that self-reproduction was a feature essential to both biological organisms and computers, [120].

A cellular automaton is a dynamical system consisting of a regular grid of cells, in which space and time are discrete. Each of the cells can be in one of a finite number of states. Each cell changes its state according to a list of given transition rules that determine its future state, based on its current state and the current states of some of its neighbors. The entire grid of cells updates its configura-

[0]Corresponding author.

[1]A few words are in order about the organization of this article. The classifications and labels we use for various fields of research are purely for the purpose of organizing the discourse. In reality, far from being clear-cut, many of the fields of research mentioned here overlap, or fit under more than one category. The very general audience for whom this paper is intended, our respective fields of expertise, and especially the limited space available for this review affected both the depth and breadth of our exposition. In particular, we did not discuss some fields of research that have large overlaps with natural computing, such as bioinformatics, computational molecular biology, and their roles in, for example, genomics and proteomics. In addition, our explanations of various aspects, themes, and paradigms had to be necessarily oversimplified. As well, the space we devoted to various fields and topics was influenced by several factors and, as such, has no relation to the respective importance of the field or the relative size of the body of research in that field.

tion synchronously according to the *a priori* given transition rules.

Cellular automata have been applied to the study of phenomena as diverse as communication, computation, construction, growth, reproduction, competition, and evolution. One of the best known examples of cellular automata, the "game of life", invented by Conway, [45], was shown to be computationally universal. Cellular automata have been extensively studied as an alternative explanation to the phenomenon of emergence of complexity in the natural world, [125], and used, among others, for modeling in physics, [16], [118], and biology [36].

In parallel to early comparisons, [119], between computing machines and the human nervous system, McCulloch and Pitts, [73], proposed the first model of artificial neurons. This research eventually gave rise to the field of **neural computation**, and it also had a profound influence on the foundations of automata theory, [59]. The goal of neural computation was two-fold. On one hand, it was hoped that it would help unravel the structure of computation in nervous systems of living organisms (How does the brain work?). On the other hand, it was predicted that, by using the principles of how the human brain processes information, neural computation would yield significant computational advances (How can we build an intelligent computer?). The first goal has been pursued mainly within the neurosciences under the name of brain theory or computational neuroscience, while the quest for the second goal has become mainly a computer science discipline known as artificial neural networks or simply *neural networks*, [6].

An artificial neural network consists of interconnected artificial neurons, [94]. Modeled after the natural neurons, each artificial neuron A has n real-valued inputs, $x_1, x_2, \ldots, x_n$, and it computes its own *primitive function* f_A as follows. Usually, the inputs have associated weights, $w_1, w_2, \ldots, w_n$. Upon receiving the n inputs, the artificial neuron A produces the output $f_A(w_1x_1 + w_2x_2 + \cdots + w_nx_n)$. An artificial neural network is a network of such neurons, and thus a network of their respective primitive functions. Some neurons are selected to be the output neurons, and the *network function* is a vectorial function that, for n input values, associates the outputs of the m output neurons. Note that different selections of the weights produce different network functions for the same inputs. Based on given input-output pairs, the network can "learn" the weights $w_1, \ldots, w_n$. Thus, there are three important features of any artificial neural network: the primitive function of each neuron, the topology of the network, and the learning algorithm used to find the weights of the network. One of the many examples of such learning algorithms is the "backwards propagation of errors". Backpropagation is a supervised learning method by which the weights of the connections in the network are repeatedly adjusted so as to minimize the difference between the actual output vector of the net and the desired output vector, [100]. Artificial neural networks have proved to be a fruitful paradigm, leading to successful novel applications in both new and established application areas, [6].

While Turing and von Neumann dreamt of understanding the brain, [115], and possibly designing an intelligent computer that works like the brain, **evolutionary computation**, [7], emerged as another computation paradigm that drew its inspiration from a completely different part of biology: Darwinian evolution, [26]. Rather than emulating features of a single biological organism, evolutionary computation draws its inspiration from the dynamics of an entire species of organisms. An artificial evolutionary system is a computational system based on the notion of simulated evolution. It features a constant- or variable-size population of individuals, a fitness criterion according to which the individuals of the population are being evaluated, and genetically-inspired operators that produce the next generation from the current one. In an evolutionary system, the initial population of individuals is generated at random or heuristically. At each evolutionary step, the individuals are evaluated according to a given fitness function. To form the next generation, firstly offspring are generated from selected individuals by using operators such as mutation of a parent, or recombination of pairs or larger subsets of parents. The choice of parents for recombination can be guided by a fitness-based selection operator, thus reflecting the biological principle of mate selection. Secondly, individuals of the next generation are selected from the set of newly created offspring, sometimes also including the old parents, according to their fitness - a process reflecting the biological concept of environmental selection.

Evolutionary systems have first been viewed as optimization processes in the 1930s, [126]. The basic idea of viewing evolution as a computational process gained momentum in the 1960s, and evolved along three main branches, [29]. *Evolution strategies*, [88], [103], [89], use evolutionary processes to solve parameter optimization problems, and are today used for real-valued as well as discrete and mixed types of parameters. *Evolutionary programming*, [40], originally aimed at achieving the goals of artificial intelligence via evolutionary techniques, namely by evolving populations of intelligent agents modeled, for example, as finite-state machines. Today, these algorithms are also often used for real-valued parameter optimization problems. *Genetic algorithms*, [53], [54], originally featured a population of individuals encoded as fixed-length bit strings, wherein mutations consisted of bit-flips according to a typically small, uniform mutation rate, the recombination of two parents consisted of a cut-and-paste of a prefix of one parent with a suffix of the other, and the fitness function was problem-dependent. If the initial individuals were to encode possible solutions to a given problem, and the fitness function were designed to measure the optimality of a candidate solution, then such a system would, in time, evolve to produce a near-optimal solution to the initial problem. Today, genetic algorithms are also modified heavily for applications to real-valued parameter optimization problems as well as many types of combinatorial tasks such as, for example, permutation-based problems. As another application, if the individuals were computer programs, then the genetic algorithm technique would result in "the fittest" computer programs, as is the goal of *genetic programming*, [62].

Cellular automata, neural computation, and evolutionary computation are the most established "classical" areas of natural computing. Several other bio-inspired paradigms emerged more recently, among them swarm intelligence, artificial immune systems, artificial life, membrane computing, and amorphous computing.

A computational paradigm straddling at times evolutionary computation and neural computation is **swarm intelligence**, [35]. A swarm is a group of mobile biological organisms (such as bacteria, ants, termites, bees, spiders, fish,

birds) wherein each individual communicates with others either directly or indirectly by acting on its local environment. These interactions contribute to distributive collective problem solving. Swarm intelligence, sometimes referred to as *collective intelligence*, is defined as the problem-solving behavior that emerges from the interaction of such a collection of individual agents. For example, in research simulating flocking behavior, [93], each individual was endowed with three simple possible behaviors: to act as to avoid collision, to match velocity with neighbors, and to stay close to nearby flock mates. The simulations showed that flocking was an emergent behavior that arose from the interaction of these simple rules.

Particle swarm optimization was introduced, [58], as a new approach to optimization that had developed from simple models of social interactions, as well as of flocking behavior in birds and other organisms. A particle swarm optimization algorithm starts with a swarm of "particles", each representing a potential solution to a problem, similar to the population of individuals in evolutionary computation. Particles move through a multidimensional search space and their positions are updated according to their own experience and that of their neighbors, by adding "velocity" to their current positions. The velocity of a particle depends on its previous velocity (the "inertia" component), the tendency towards the past personal best position (the cognitive, "nostalgia", component), and the move towards a global or local neighborhood best (the "social" component), [35]. The cumulative effect is that each particle converges towards a point between the global best and its personal best. Particle Swarm Optimization algorithms have been used to solve various optimization problems, and have been applied to unsupervised learning, game learning, scheduling and planning applications, and design applications, [35].

Ant algorithms were introduced, [32], to model the foraging behavior of ant colonies. In finding the best path between their nest and a source of food, ants rely on indirect communication by laying a pheromone trail on the way back to the nest if they found food, and following the concentration of pheromones in the environment if they are looking for food. This foraging behavior has inspired a large number of ant algorithms used to solve mainly combinatorial optimization problems defined over discrete search spaces, [35].

Artificial immune systems are computational systems devised starting in the late 1980s and early 1990s, [37], [28], as computationally interesting abstractions of the natural immune system of biological organisms. Viewed as an information processing system, the immune system performs many complex computations in a highly parallel and distributed fashion, [27]. It uses learning, memory, associative retrieval, and other mechanisms to solve recognition and classification problems such as distinction between self and nonself cells, and neutralization of nonself pathogenic agents. Indeed, the natural immune system has sometimes been called the "second brain", [97], because of its powerful information processing capabilities, [27].

The natural immune system's main function is to protect our bodies against the constant attack of external pathogens (viruses, bacteria, fungi, and parasites). The main role of the immune system is to recognize cells in the body and categorize them as self or nonself, [28]. There are two parts of the immune system: innate (non-specific) and adaptive (acquired). The cells of the innate immune system are immediately available to combat against a wide variety of antigens, without requiring previous exposure to them. These cells possess the ability of ingesting and digesting several "known" pathogens. In contrast, the adaptive immune response is the antibody production in response to a specific new infectious agent. Our body maintains a large "combinatorial database" of immune cells which circulate throughout the body. When a foreign antigen invades the body, only a few of these immune cells can detect the invaders and physically bind to them. This detection triggers the primary immune response: the generation of a large population of cells that produce matching antibodies which aid in the destruction or neutralization of the antigen. The immune system also retains some of these specific-antibody-producing cells in immunological memory, so that any subsequent exposure to a similar antigen can lead to a rapid, and thus more effective, immune response (secondary response).

The computational aspects of the immune system, such as distinguishing of self from nonself, feature extraction, learning, memory, self-regulation, and fault tolerance, have been exploited in the design of artificial immune systems that have been successfully used in applications. The applications are varied and include computer virus detection, [41], anomaly detection in a time series of data, fault diagnosis, pattern recognition, [27], machine learning, bioinformatics, optimization, robotics, and control, [116]. Recent research in immunology departs from the self-nonself discrimination model to develop what is known as the "danger theory", [72], wherein it is believed that the immune system differentiates between dangerous and non-dangerous entities, regardless of whether they belong to self or to nonself. These ideas have started to be exploited in artificial immune systems in the context of computer security, [49].

While artificial immune systems (a.k.a. immunological computation, immunocomputing) constitute an example of a computational paradigm inspired by a very specific subsystem of a biological organism, artificial life takes the opposite approach. **Artificial life (ALife)** attempts to understand the very essence of what it means to be alive by building *ab initio*, within *in silico* computers and other "artificial" media, artificial systems that exhibit properties normally associated only with living organisms, [64]. *Lindenmayer systems* (*L-systems*), introduced in 1968, [68], can be considered as an early example of artificial life. L-systems are parallel rewriting systems that, starting with an initial word, proceed by applying rewriting rules in parallel to all the letters of the word, and thus generate new words, [99]. They have been most famously used to model plant growth and development, [87], but also for modeling the morphology of other organisms.

Building on the ideas of evolutionary computation, other pioneers of artificial life experimented with evolving populations of "artificial creatures" in simulated environments, [19]. One example, [111], was the design of evolving virtual block creatures that were selected for their ability to swim (or walk, or jump), and that competed for a common resource (controlling a cube) in a physically simulated world endowed with realistic features such as kinematics, dynamics, gravity, collisions, and friction. The result was that creatures evolved which would extend arms towards the cube, while others would crawl or roll to reach it, and some even developed legs that they used to walk towards the

cube. These ideas were taken one step further, [69], by combining the computational and experimental approaches, and using rapid manufacturing technology to fabricate physical robots that were materializations of their virtually evolved computational counterparts. In spite of the simplicity of the task at hand (horizontal locomotion), surprisingly different and complex robots evolved: many of them exhibited symmetry, some moved sideways in a crab-like fashion, and some others crawled on two evolved limbs. This marked the emergence of mechanical artificial life, while the nascent field of synthetic biology, discussed later, explores a biological implementation of similar ideas. At the same time, the field of Artificial Life continues to explore directions such as artificial chemistry (abstractions of natural molecular processes), as well as traditionally-biological phenomena in artificial systems, ranging from computational processes such as co-evolutionary adaptation and development, to physical processes such as growth, self-replication, and self-repair.

Membrane computing investigates computing models abstracted from the structure and the functioning of living cells, as well as from the way the cells are organized in tissues or higher order structures, [83], [84]. More specifically, the feature of the living cells that is abstracted by membrane computing is their compartmentalized internal structure effected by membranes. A generic membrane system is essentially a nested hierarchical structure of cell-like compartments or regions, delimited by "membranes". The entire system is enclosed in an external membrane, called the skin membrane, and everything outside the skin membrane is considered to be the environment. Each membrane-enveloped region contains objects and transformation rules which modify these objects, as well as specify whether they will be transferred outside or stay inside the region. The transfer thus provides for communication between regions. Various formal mechanisms were developed that reflect the selective manner in which biological membranes allow molecules to pass through them.

Another biologically-inspired feature of membrane systems as mathematical constructs is the fact that, instead of dealing with sets of objects, one uses multisets wherein one keeps track of the multiplicity of each object. The computational behavior of a membrane system starts with an initial input configuration and proceeds in a maximally parallel manner by the non-deterministic choice of application of the transformation rules, as well as of the objects to which they are to be applied. The output of the computation is then collected from an *a priori* determined output membrane. Next to the basic features indicated above, many alternatives of membrane systems have been considered, among them ones that allow for membranes to be dissolved and created. Typical applications of membrane systems include biology (modeling photosynthesis and certain signaling pathways, quorum sensing in bacteria, modeling cell-mediated immunity), computer science (computer graphics, public-key cryptography, approximation and sorting algorithms, and solving computationally hard problems), and linguistics, [24].

Amorphous computing is a paradigm that draws inspiration from the development of form (morphogenesis) in biological organisms, wherein interactions of cells guided by a genetic program give rise to well-defined shapes and functional structures. Analogously, an amorphous computing medium comprises a multitude of irregularly placed, asynchronous, locally interacting computing elements, [1]. These identically programmed "computational particles" communicate only with particles situated within a small given radius, and may give rise to certain shapes and patterns such as, for example, any pre-specified planar graph, [25]. The goal of amorphous computing is to engineer specified coherent computational behaviors from the interaction of large quantities of such unreliable computational particles interconnected in unknown, irregular, and time-varying ways, [1]. At the same time, the emphasis is on devising new programming abstractions that would work well for amorphous computing environments. Amorphous computing has been used both as a programming paradigm using traditional hardware, and as the basis for "cellular computing", discussed later, under the topics synthetic biology and computation in living cells.

3. NATURE AS IMPLEMENTATION SUBSTRATE

In the preceding section we saw cellular automata inspired by self-reproduction, neural computation by the functioning of the brain, evolutionary computation by the Darwinian evolution of species, swarm intelligence by the behavior of groups of organisms, artificial immune systems by the natural immune system, artificial life by properties of life in general, membrane computing by the compartmentalized organization of the cells, and amorphous computing by morphogenesis. All these are computational techniques that, while inspired by nature, have been implemented until now mostly on traditional electronic hardware. An entirely distinct category is that of computing paradigms that use a radically different type of "hardware". This category includes molecular computing and quantum computing[2].

Molecular computing (known also as biomolecular computing, biocomputing, biochemical computing, DNA computing), is based on the idea that data can be encoded as biomolecules – such as DNA strands–, and molecular biology tools can be used to transform this data to perform, for example, arithmetic or logic operations. The birth of this field was the 1994 breakthrough experiment by Leonard Adleman who solved a small instance of the Hamiltonian Path Problem solely by manipulating DNA strands in test-tubes, [2].

DNA (deoxyribonucleic acid) is a linear chain made up of four different types of nucleotides, each consisting of a base (Adenine, Cytosine, Guanine, or Thymine) and a sugar-phosphate unit. The sugar-phosphate units are linked together by covalent bonds to form the backbone of the DNA single strand. Since nucleotides may differ only by their bases, a DNA strand can be viewed as simply a word over the four-letter alphabet $\{A, C, G, T\}$. A DNA single strand has an orientation, with one end known as the 5' end, and the other as the 3' end, based on their chemical properties. By convention, a word over the DNA alphabet represents

[2]There are several research areas that, because of the limited space, we could not discuss here. Thus, for example, non-classical, unconventional computation, [113], focuses on carefully examining and possibly breaking the classical (Turing, von Neumann) computation assumptions, and developing a more general science of computation. A substantial part of this research is concerned with implementing computation on new physical substrates, exploiting in this way computational properties of various physical, chemical, and biological media. A majority of this research is entwined with, and motivated by, natural computing.

the corresponding DNA single strand in the 5' to 3' orientation, that is, the word GGTTTTT stands for the DNA single strand 5'-GGTTTTT-3'. A crucial feature of DNA single strands is their Watson-Crick complementarity: A is complementary to T, G is complementary to C, and two complementary DNA single strands with opposite orientation bind to each other by hydrogen bonds between their individual bases. In so doing, they form a stable DNA double strand resembling a helical ladder, with the backbones at the outside and the bound pairs of bases lying inside. For example, the DNA single strand 5'- AAAAACC - 3' will bind to the DNA single strand 5'-GGTTTTT-3' to form the 7 base-pair-long (7bp) double strand $\begin{matrix} 5' - AAAAACC - 3' \\ 3' - TTTTTGG - 5' \end{matrix}$. Another molecule that can be used for computation is RNA, ribonucleic acid. RNA is similar to DNA, but differs from it in three main aspects: RNA is usually single-stranded while DNA is usually double-stranded, RNA nucleotides contain the sugar ribose, while DNA nucleotides contain the sugar deoxyribose, and in RNA the nucleotide Uracil, U, substitutes for Thymine, which is present in DNA.

There are many possible DNA bio-operations that one can use for computations, [56], such as: cut-and-paste operations achievable by enzymes, synthesizing desired DNA strands up to a certain length, making exponentially many copies of a DNA strand, and reading-out the sequence of a DNA strand. These bio-operations and the Watson-Crick complementarity binding have all been used to control DNA computations and DNA robotic operations. While initial experiments solved simple instances of computational problems, more recent experiments tackled successfully sophisticated computational problems, such as a 20-variable instance of the 3-Satisfiability-Problem [17]. The efforts towards building an autonomous molecular computer include implementations of computational state transitions with biomolecules, [101], and a DNA implementation of a finite automaton with potential applications to the design of smart drugs, [106].

More importantly, since 1994, research in molecular computing has gained several new dimensions. One of the most significant achievements of molecular computing has been its contribution to the massive stream of research in nanosciences, by providing computational insights into a number of fundamental issues. Perhaps the most notable is its contribution to the understanding of self-assembly, [124], which is among the key concepts in nanosciences, [92]. Recent experimental research into programmable molecular-scale devices has produced impressive self-assembled DNA nanostructures, [105], such as cubes, [23], [105], octahedra, [108], Sierpinski triangles, [96], DNA origami, [95], or intricate nanostructures that achieve computation such as binary counting, [8], or bit-wise cumulative XOR, [71]. Other experiments include the construction of DNA-based logic circuits, [104], and ribozymes that can be used to perform logical operations and simple computations, [114], [66]. In addition, an array of ingenious DNA nanomachines, [12], were built with potential uses to nanofabrication, engineering, and computation: molecular switches that can be driven between two conformations, [67], DNA "tweezers", [127], DNA "walkers" that can be moved along a track, [109], [107], and autonomous molecular motors, [91], [11], [48].

A significant amount of research in molecular computing has been dedicated to the study of theoretical models of DNA computation and their properties. The model of DNA computing introduced by Head, [51], based on splicing (a combination of cut-and-paste operations achievable by enzymes), predates the experimental proof-of-principle of DNA computing by almost ten years. Subsequently, studies on the computational power of such models proved that various subsets of bio-operations can achieve the computational power of a Turing machine, showing thus that molecular computers are in principle possible, [85], [56]. Overall, molecular computing has created many novel theoretical questions, and has considerably enriched the theory of computation.

Quantum Computing is another paradigm that uses an alternative "hardware" for performing computations, [52], [80], [57]. Already in 1980 Benioff, [13], introduced simulations of classical Turing Machines on quantum mechanical systems. However the idea of a *quantum computer* that would run according to the laws of quantum physics and operate exponentially faster than a deterministic electronic computer to simulate physics, was first suggested by Feynman in 1982, [38], [39]. Subsequently, Deutsch introduced a formal model of quantum computing using a Turing machine formalism, and described a universal quantum computer, [30].

A quantum computer uses distinctively quantum mechanical phenomena, such as superposition and entanglement, to perform operations on data stored as quantum bits (qubits). A *qubit* can hold a 1, a 0, or a quantum superposition of these. A quantum computer operates by manipulating those qubits with quantum logic gates. The notion of information is different when studied at the quantum level. For instance, quantum information cannot be measured reliably, and any attempt at measuring it entails an unavoidable and irreversible disturbance.

The 1980s saw an abundance of research in quantum information processing, such as applications to quantum cryptography, [14], which, unlike its classical counterpart, is not usually based on the complexity of computation, but on the special properties of quantum information. Recently, [117], an open air experiment was reported in quantum cryptography (not involving optical cable) over a distance of 144 km, conducted between two Canary islands.

The theoretical results that catapulted quantum computing to the forefront of computing research were Shor's quantum algorithms for factoring integers and extracting discrete logarithms in polynomial time, obtained in 1994, [110] – the same year that saw the first DNA computing experiment by Adleman. A problem where quantum computers were shown to have a quadratic time advantage when compared to classical computers is quantum database search that can be solved by Grover's algorithm, [50], [18]. Possible applications of Shor's algorithm include breaking RSA exponentially faster than an electronic computer. This joined other exciting applications, such as quantum teleportation (a technique that transfers a quantum state, but not matter or energy, to an arbitrarily distant location), [15], in sustaining the general interest in quantum information processing.

So far, the theory of quantum computing has been far more developed than the practice. Practical quantum computations use a variety of implementation methods such as ion-traps, superconductors, nuclear magnetic resonance techniques, to name just a few. To date, the largest quantum computing experiment uses liquid state nuclear magnetic resonance quantum information processors that can

operate on up to 12 qubits, [79].

4. NATURE AS COMPUTATION

The preceding sections describe research on the theory, applications and experimental implementations of nature-inspired computational models and techniques. A dual direction of research in natural computing is one in which the main goal becomes understanding nature by viewing processes that take place in nature as information processing.

This dual aspect can be seen in **systems biology**, and especially in **computational systems biology**, wherein the adjective "computational" has two meanings. On one hand it means the use of quantitative algorithms for computations, or simulations that complement experiments in hypothesis generation and validation, [31]. On the other hand, it means a qualitative approach that investigates processes taking place in cells through the prism of communications and interactions, and thus of computations. We shall herein address mostly the second aspect, whereby systems biology aims to understand the complex interactions in biological systems by using an integrative as opposed to a reductionist approach. The reductionist approach to biology tries to identify all the individual components of functional processes that take place in an organism, in such a way that the processes and the interactions between the components can be understood. In contrast, systems biology takes a systemic approach in focusing instead on the interaction networks themselves, and on the properties of the biological systems that arise because of these interaction networks. Hence, for example, at the cell level, scientific research on organic components has focused strongly on four different interdependent interaction networks, based on four different "biochemical toolkits": nucleic acids (DNA and RNA), proteins, lipids, carbohydrates, and their building blocks (see [22], whose categorization we follow here).

The genome consists of DNA sequences, some of which are genes that can be transcribed into messenger RNA (mRNA), and then translated into proteins according to the *genetic code* that maps 3-letter DNA segments into amino acids. A protein is a sequence over the 20-letter alphabet of amino acids. Each gene is associated with other DNA segments (promoters, enhancers, or silencers) that act as binding sites for proteins which activate or repress the gene's transcription. Genes interact with each other indirectly, either through their gene products (mRNA, proteins) which can act as transcription factors to regulate gene transcription – either as activators or repressors –, or through small RNA species that directly regulate genes.

These gene-gene interactions, together with the genes' interactions with other substances in the cell, form the most basic interaction network of an organism, the **gene regulatory network**. Gene regulatory networks perform information processing tasks within the cell, including the assembly and maintenance of the other networks. Research into modeling gene regulatory networks includes qualitative models such as random and probabilistic Boolean networks, asynchronous automata, and network motifs, [22].

Another point of view, [55], is that the entire genomic regulatory system can be thought of as a computational system, the "genomic computer". Such a perspective has the potential to yield insights into both computation as humans historically designed it, and computation as it occurs in nature. There are both similarities and significant differences between the genomic computer and an electronic computer. Both perform computations, the genomic computer on a much larger scale. However, in a genomic computer, molecular transport and movement of ions through electrochemical gradients replace wires, causal coordination replaces imposed temporal synchrony, changeable architecture replaces rigid structure, and communication channels are formed on an as-needed basis. Both computers have a passive memory, but the genomic computer does not place it in an *a priori* dedicated and rigidly defined place; in addition, the genomic computer has a dynamic memory in which, for example, transcriptional subcircuits maintain given regulatory states. In a genomic computer robustness is achieved by different means, such as by rigorous selection: non (or poorly)-functional processes are rapidly degraded by various feed-back mechanisms or, at the cell level, non (or poorly)-functional cells are rapidly killed by apoptosis, and, at the organism level, non (or poorly)-functional organisms are rapidly out-competed by more fit species, [26]. Finally, in the case of a genomic computer, the distinction between hardware and software breaks down: the genomic DNA provides both the hardware and the digital regulatory code (software), [55].

Proteins and their interactions form another interaction network in a cell, that of **biochemical networks**, which perform all mechanical and metabolic tasks inside a cell, [22]. Proteins are folded-up strings of amino acids that take three-dimensional shapes, with possible characteristic interaction sites accessible to other molecules. If the binding of interaction sites is energetically favourable, two or more proteins may specifically bind to each other to form a dynamic protein complex by a process called *complexation*. A protein complex may act as a catalyst by bringing together other compounds and facilitating chemical reactions between them. Proteins may also chemically modify each other by attaching or removing modifying groups, such as phosphate groups, at specific sites. Each such modification may reveal new interaction surfaces. There are tens of thousands of proteins in a cell. At any given moment, each of them has certain available binding sites (which means that they can bind to other proteins, DNA, or membranes), and each of them has modifying groups at specific sites either present or absent. Protein-protein interaction networks are large and complex, and finding a language to describe them is a difficult task. A significant progress in this direction was made by the introduction of Kohn-maps, [61], a graphical notation that resulted in succinct pictures depicting molecular interactions. Other approaches include the textual biocalculus, [77], or the recent use of existing process calculi (π-calculus), enriched with stochastic features, as the language to describe chemical interactions, [90].

Yet another biological interaction network, and the last that we discuss here, is that of **transport networks** mediated by lipid membranes. Some lipids can self-assemble into membranes and contribute to the separation and transport of substances, forming transport networks. A biological membrane is more than a container: it consists of a lipid bilayer in which proteins and other molecules, such as glycolipids, are embedded. The membrane structural components, as well as the embedded proteins or glycolipids, can travel along this lipid bilayer. Proteins can interact with free-floating molecules, and some of these interactions

trigger signal transduction pathways, leading to gene transcription. Basic operations of membranes include fusion of two membranes into one, and fission of a membrane into two. Other operations involve transport, for example transporting an object to an interior compartment where it can be degraded. Formalisms that depict the transport networks are few, and include membrane systems described earlier, and brane calculi, [21].

The gene regulatory networks, the protein-protein interaction networks, and the transport networks are all interlinked and interdependent. Genes code for proteins which, in turn, can regulate the transcription of other genes, membranes are separators but also embed active proteins in their surfaces. Currently there is no single formal general framework and notation able to describe all these networks and their interactions. Process calculus, [74], has been proposed for this purpose, but a generally accepted common language to describe these biological phenomena is still to be developed and universally accepted. It is indeed believed that one of the possible contributions of computer science to biology could be the development of a suitable language to accurately and succinctly describe, and reason about, biological concepts and phenomena, [43], [20].

While systems biology studies complex biological organisms as integrated wholes, **synthetic biology** is an effort to engineer artificial biological systems from their constituent parts. The mantra of synthetic biology is that one can understand only what one can construct. Thus, the main focus of synthetic biology is to take parts of natural biological systems and use them to build an artificial biological system for the purpose of understanding natural phenomena, or for a variety of possible applications. In this sense, one can make an analogy between synthetic biology and computer engineering, [4]. The history of synthetic biology can be arguably traced back to the discovery in the 1960s, by Jacob and Monod, [75], of mathematical logic in gene regulation. Early achievements in genetic engineering using recombinant DNA technology (the insertion, deletion, or combination of different segments of DNA strands) can be viewed as the experimental precursors of today's synthetic biology which now extends these techniques to entire systems of genes and gene products. One goal can be constructing specific synthetic biological modules such as, for example, pulse generator circuits that display a transient response to variations in input stimulus, [10].

Advances in DNA synthesis of longer and longer strands of DNA are paving the way for the construction of synthetic genomes with the purpose of building an entirely artificial organism, [128]. Progress includes the generation of a 5,386bp synthetic genome of a virus, by rapid (14 day) assembly of chemically synthesized short DNA strands, [112]. Recently an announcement was made of the near-completion of the assembly of an entire "minimal genome" of a bacterium, *Mycoplasma Genitalium*, [9]. Smith and others indeed found about 100 dispensable genes that can be removed individually from the original genome. They hope to assemble a minimal genome consisting of essential genes only, that would be still viable but shorter than the 528-gene, 580,000bp genome of *M.Genitalium*. This human-made genome could then be inserted into a *Mycoplasma* bacterium using a technique, [65], wherein a whole genome can be transplanted from one species into another, such that the resulting progeny is the same species as the donor genome. Counterbalancing objections to assembling a semi-synthetic cell without fully understanding its functioning, the creation of a functionally and structurally understood synthetic genome was proposed, [42], containing 151 genes (113,000bp) that would produce all the basic molecular machinery for protein synthesis and DNA replication.

A third approach to create a human-made cell is the one pursued by Szostak and others, [102], who would construct a single type of RNA-like molecule capable of self-replicating, possibly housed in a single lipid membrane, [76]. Such molecules can be obtained by guiding the rapid evolution of an initial population of RNA-like molecules, by selecting for desired traits, [82].

Lastly, another effort in synthetic biology is towards engineering multi-cellular systems by designing, e.g., cell-to-cell communication modules that could be used to coordinate living bacterial cell populations, [123].

Research in synthetic biology faces many challenges, some of them of an information processing nature. There arguably is a pressing need for standardization, modularization, and abstraction, to allow focusing on design principles without reference to lower-level details, [34].

Besides systems biology that tries to understand biological organisms as networks of interactions, and synthetic biology that seeks to engineer and build artificial biological systems, another approach to understanding nature as computation is the research on **computation in living cells**. This is also sometimes called *cellular computing*, or *in vivo computing*, and one particular study in this area is that of the computational capabilities of gene assembly in unicellular organisms called ciliates, [63], [33].

Ciliates possess two copies of their DNA: one copy encoding functional genes, in the macronucleus, and another "encrypted" copy in the micronucleus. In the process of conjugation, after two ciliates exchange genetic information and form new micronuclei, they use the new micronuclei to assemble in real-time new macronuclei necessary for their survival. This is accomplished by a process that involves re-ordering some fragments of DNA (permutations and possibly inversions), and deleting other fragments from the micronuclear copy. The process of gene assembly is fascinating from both the biological and the computational point of view. From the computational point of view, this study led to many novel and challenging research themes, [33]. Among others, it was proved that various models of gene assembly have full Turing machine capabilities, [63]. From the biological point of view, the joint effort of computer scientists and biologists led to a plausible hypothesis (supported already by some experimental data) about the "bioware" that implements the process of gene assembly, which is based on the new concept of template-guided recombination, [86], [5], [81].

Other approaches to cellular computing include developing of an *in vivo* programmable and autonomous finite-state automaton within *E.Coli*, [78], and designing and constructing *in vivo* cellular logic gates and genetic circuits that harness the cell's existing biochemical processes, [60], [122], [121].

At the end of this spectrum of views of nature as computation, the idea was even advanced by Zuse and Fredkin in the 1960s that information is more fundamental than matter or energy. The Zuse-Fredkin thesis stated that the entire universe is some kind of computational device, namely

a huge cellular automaton continuously updating its rules, [44], [129], [46]. Along the same lines, it has been recently suggested, [70], that the universe is a quantum computer that computes itself and its own behavior.

5. NATURAL SCIENCES: OURS TO DISCOVER

Science advances in ever-widening circles of knowledge. Sometimes it meticulously crawls. Other times it leaps to a new dimension of understanding and, in the process, it reinvents itself. As the natural sciences are rapidly absorbing ideas of information processing, and the meaning of computation is changing as it embraces concepts from the natural sciences, we have the rare privilege to take part in such metamorphoses.

At this moment we and our natural scientist fellows are awash in wave after gigantic wave of experimental, especially biological, data. Just underneath this tumultuous surface lie ingenious algorithms waiting to be designed, elegant theorems waiting to be proven, natural laws waiting to be discovered that will put order into chaos. For, as Spinoza wrote, "nothing happens in nature that does not follow from her laws", [47].

Conversely, as this review shows, there is an abundance of natural phenomena which can inspire computing paradigms, alternative physical substrates on which to implement computations, while viewing various natural processes as computations has become more and more essential, desirable, and inevitable. All these developments are challenging our assumptions about computation, and indeed, our very definition of it.

In these times brimming with excitement, our task is nothing less than to discover a new, broader, notion of computation, and to understand the world around us in terms of information processing.

Let us step up to this challenge. Let us befriend our fellow the biologist, our fellow the chemist, our fellow the physicist, and let us together explore this new world. Let us, as computers in the future will, embrace uncertainty. Let us dare to ask afresh: "What is computation?", "What is complexity?", "What are the axioms that define life?".

Let us relax our hardened ways of thinking and, with deference to our scientific forebears, let us begin anew.

6. LITERATURE

The upper-bound placed on the number of references was a real limitation for this review, since the literature on natural computing is vast. For a more complete list of references the reader is referred to the full version of this article at www.csd.uwo.ca/~lila/Natural-Computing-Review.pdf.

Almost each of the areas we mentioned here has an extensive scientific literature as well as a number of specialized journals and book series. There are also journals and book series aimed at the general natural computing community, among them the journals *Natural Computing*, Springer, *Theoretical Computer Science, Series C: Theory of Natural Computing*, Elsevier, the *Natural Computing* book series, Springer, and the upcoming *Handbook of Natural Computing* (G.Rozenberg, T.Bäck, J.Kok, editors, Springer).

7. ACKNOWLEDGEMENTS

We gratefully acknowledge comments on early drafts of this paper by T. Bäck, D. Bentley, G. Brassard, D. Corne, M. Hirvensalo, J. Kari, P. Krishna, H. Lipson, R. Mercer, A. Salomaa, K. Sims, H. Spaink, J. Timmis, C. Torras, S. Watt, R. Weiss.

This work was supported by NSERC Discovery Grant and Canada Research Chair Award to L.K., and NSF grant 0622112 to G.R.

8. REFERENCES

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight Jr., R. Nagpal, E. Rauch, G. Sussman, and R. Weiss. Amorphous computing. *CACM*, 43(5):74–82, 2000.

[2] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.

[3] L. Adleman. Computing with DNA. *Scientific American*, 279(2):54–61, 1998.

[4] E. Andrianantoandro, S. Basu, D. Karig, and R. Weiss. Synthetic biology: new engineering rules for an emerging discipline. *Molecular Systems Biology*, 2:1–14, 2006.

[5] A. Angeleska, N. Jonoska, M. Saito, and L. Landweber. RNA-guided DNA assembly. *J. Theoretical Biology*, 248:706–720, 2007.

[6] M. Arbib, editor. *The Handbook of Brain Theory and Neural Networks*. MIT Press, 2003.

[7] T. Bäck, D. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. IOP Publishing, UK, 1997.

[8] R. Barish, P. Rothemund, and E. Winfree. Two computational primitives for algorithmic self-assembly: Copying and counting. *Nanoletters*, 5(12):2586–2592, 2005.

[9] P. Barry. Life from scratch: learning to make synthetic cells. *Science News*, 173(2):27, 2008.

[10] S. Basu, R. Mehreja, S. Thiberge, M. Chen, and R. Weiss. Spatiotemporal control of gene expression with pulse-generating networks. *PNAS*, 101:6355–6360, 2004.

[11] J. Bath, S. Green, and A. Turberfield. A free-running DNA motor powered by a nicking enzyme. *Angew.Chem.Int.Ed.*, 44(28):4358–4361, 2005.

[12] J. Bath and A. Turberfield. DNA nanomachines. *Nature Nanotechnology*, 2:275–284, May 2007.

[13] P. Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *J. Stat. Phys.*, 22(5):563–591, 1980.

[14] C. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proc. IEEE Int. Conf. on Comp., Syst., and Signal Processing*, pages 175–179, 1984.

[15] C. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Phys. Rev. Lett.*, 70:1895–1899, 1993.

[16] C. Bennett and G. Grinstein. Role of irreversibility in stabilizing complex and nonergodic behavior in locally interacting discrete systems. *Phys. Rev. Lett.*, 55:657–660, 1985.

[17] R. Braich, N. Chelyapov, C. Johnson, P. Rothemund, and L. Adleman. Solution of a 20-variable 3-SAT problem on a DNA computer. *Science*, 296:499–502, 2002.

[18] G. Brassard. Searching a quantum phone book. *Science*, 275:627–628, 1997.

[19] R. Brooks. Artificial life: From robot dreams to reality. *Nature*, 406:945–947, 2000.

[20] L. Cardelli. Bioware languages. In *Computer Systems: Theory, Technology, and Applications—A Tribute to Roger Needham*, pages 59–65. Springer, 2004.

[21] L. Cardelli. Brane calculi: Interactions of biological membranes. In LNCS *3082*, pages 257–280. Springer, 2005.

[22] L. Cardelli. Machines of systems biology. *Bulletin of the EATCS*, 93:176–204, 2007.

[23] J. Chen and N. Seeman. Synthesis from DNA of a molecule with the connectivity of a cube. *Nature*, 350:631–633, 1991.

[24] G. Ciobanu, G. Paun, and M. Perez-Jimenez, editors. *Applications of Membrane Computing*. Springer, 2006.

[25] D. Coore. *Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer*. PhD thesis, MIT, 1999.

[26] C. Darwin. *The Origin of Species by Means of Natural Selection*. Adamant Media Corp., 2001. Original 1859.

[27] D. Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer, 1998.

[28] L. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[29] K. De Jong. *Evolutionary Computation: A Unified Approach*. MIT Press, 2006.

[30] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London*, A400:97–117, 1985.

[31] B. Di Ventura, C. Lemerle, K. Michalodimitrakis, and L. Serrano. From *in vivo* to *in silico* biology and back. *Nature*, 443:527–533, 2006.

[32] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, 1992.

[33] A. Ehrenfeucht, T. Harju, I. Petre, D. Prescott, and G. Rozenberg. *Computation in Living Cells: Gene Assembly in Ciliates*. Springer, 2004.

[34] D. Endy. Foundations for engineering biology. *Nature*, 438:449–453, 2005.

[35] A. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley and Sons, 2005.

[36] G. Ermentrout and L. Edelstein-Keshet. Cellular automata approaches to biological modelling. *J. Theoretical Biology*, 160:97–133, 1993.

[37] J. Farmer, N. Packard, and A. Perelson. The immune system, adaptation, and machine learning. *Physica D*, 22:187–204, 1986.

[38] R. Feynman. Simulating physics with computers. *Int. J. Theoretical Physics*, 21(6/7):467–488, 1982.

[39] R. Feynman. Quantum mechanical computers. *Optics News*, 11:11–46, 1985.

[40] L. Fogel, A. Owens, and M. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley and Sons, 1966.

[41] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proc. IEEE Symp. on Res. in Security and Privacy*, pages 202–212, 1994.

[42] A. Forster and G. Church. Towards synthesis of a minimal cell. *Molecular Systems Biology*, 2(45), August 2006.

[43] E. Fox Keller and D. Harel. Beyond the gene. *PLoS ONE*, 2(11):e1231, 2007.

[44] E. Fredkin. Digital mechanics: An informational process based on reversible universal CA. *Physica D*, 45:254–270, 1990.

[45] M. Gardner. The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, pages 120–123, October 1970.

[46] M. Gardner. Mathematical games: On cellular automata, self-reproduction, the Garden of Eden and the game 'life'. *Scientific American*, pages 112–117, February 1971.

[47] C. Gebhardt, editor. *Spinoza Opera*. Heidelberg, Winters, 1925.

[48] S. Green, L. Lubrich, and A. Turberfield. DNA hairpins: fuel for autonomous DNA devices. *Biophysics J.*, 91:2966–2975, 2006.

[49] J. Greensmith, U. Aickelin, and J. Twycross. Articulation and clarification of the dendritic cell algorithm. In LNCS *4163*, pages 404–417. Springer, 2006.

[50] L. Grover. A fast quantum mechanical algorithm for database search. In *Proc. STOC*, pages 212–219. ACM, 1996.

[51] T. Head. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bull. of Math. Biol.*, 49:737–759, 1987.

[52] M. Hirvensalo. *Quantum Computing, 2nd ed.* Springer, 2004.

[53] J. Holland. Outline for a logical theory of adaptive systems. *JACM*, 9:297–314, 1962.

[54] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[55] S. Istrail, S. Ben-Tabou De-Leon, and E. Davidson. The regulatory genome and the computer. *Developmental Biology*, 310:187–195, 2007.

[56] L. Kari. DNA computing—the arrival of biological mathematics. *The Math. Intelligencer*, 19(2):9–22, 1997.

[57] P. Kaye, R. Laflamme, and M. Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 2007.

[58] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. IEEE Int. Conf. Neural Networks*, pages 1942–1948. IEEE Press, 1995.

[59] S. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies, Annals of Mathematical Studies 34*, pages 3–41. Princeton, 1956.

[60] T. Knight Jr. and G. Sussman. Cellular gate technology. In *Unconventional Models of Computation*, pages 257–272. Springer, 1998.

[61] K. Kohn. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Molecular Biology of the Cell*, 10(8):2703–2734, 1999.

[62] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[63] L. Landweber and L. Kari. The evolution of cellular computing: nature's solution to a computational problem. *Biosystems*, 52(1/3):3–13, 1999.

[64] C. Langton, editor. *Artificial Life*. Addison-Wesley Longman, 1990.

[65] C. Lartigue *et al.* Genome transplantation in bacteria: changing one species to another. *Science*, 317:632–638, 2007.

[66] H. Lederman, J. Macdonald, D. Stefanovic, and M. Stojanovic. Deoxyribozyme-based three-input logic gates and construction of a molecular full adder. *Biochemistry*, 45(4):1194–1199, 2006.

[67] T. Liedl, M. Olapinski, and F. Simmel. A surface-bound DNA switch driven by a chemical oscillator. *Angew.Chem.Int.Edn.*, 45(30):5007–5010, 2006.

[68] A. Lindenmayer. Mathematical models for cellular interaction in development, parts I and II. *J. Theoretical Biology*, 18:280–315, 1968.

[69] H. Lipson and J. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.

[70] S. Lloyd. *Programming the Universe: A Quantum Computer Scientist Takes on the Cosmos*. Knopf, 2006.

[71] C. Mao, T. LaBean, J. Reif, and N. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature*, 407:493–496, 2000.

[72] P. Matzinger. The danger model: A renewed sense of self. *Science*, 296:301–305, 2002.

[73] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[74] R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[75] J. Monod and F. Jacob. Teleonomic mechanisms in cellular metabolism, growth, and differentiation. *Cold Spring Harb. Symp. Quant. Biol.*, 26:389–401, 1961.

[76] G. Murtas, Y. Kuruma, P. Bianchini, A. Diaspro, and P. Luisi. Protein synthesis in liposomes with a minimal set of enzymes. *Biochem. and Biophys. Res. Comm.*, 363:12–17, 2007.

[77] M. Nagasaki, S. Onami, S. Miyano, and H. Kitano. Bio-calculus: Its concept and molecular interaction. *Genome Informatics*, 10:133–143, 1999.

[78] H. Nakagawa, K. Sakamoto, and Y. Sakakibara. Development of an *in vivo* computer based on Escherichia Coli. In LNCS *3892*, pages 203–212. Springer, 2006.

[79] C. Negrevergne *et al.* Benchmarking quantum control methods on a 12-qubit system. *Phys. Rev. Lett.*, 96:art170501, 2006.

[80] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[81] M. Nowacki, V. Vijayan, Y. Zhou, K. Schotanus, T. Doak, and L. Landweber. RNA-mediated epigenetic programming of a genome-rearrangement pathway. *Nature*, 451:153–158, 2008.

[82] N. Paul, G. Springsteen, and G. Joyce. Conversion of a ribozyme to a deoxyribozyme through in vitro evolution. *Chemistry and Biology*, 13(3):329–338, 2006.

[83] G. Paun. *Membrane Computing: An Introduction*. Springer, 2002.

[84] G. Paun and G. Rozenberg. A guide to membrane computing. *Theoretical Computer Science*, 287(1):73–100, 2002.

[85] G. Paun, G. Rozenberg, and A. Salomaa. *DNA Computing: New Computing Paradigms*. Springer, 1998.

[86] D. Prescott, A. Ehrenfeucht, and G. Rozenberg. Template-guided recombination for IES elimination and unscrambling of genes in stichotrichous ciliates. *J. Theoretical Biology*, 222(3):323–330, 2003.

[87] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, 1990.

[88] I. Rechenberg. Cybernetic solution path of an experimental problem. *Royal AirCraft Establishment, Library Translation*, 1122, 1965.

[89] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog, 1973.

[90] A. Regev and E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419:343–343, 2002.

[91] J. Reif. The design of autonomous DNA nanomechanical devices: Walking and rolling DNA. In LNCS *2568*, pages 22–37. Springer, 2003.

[92] J. Reif and T. LaBean. Autonomous programmable biomolecular devices using self-assembled DNA nanostructures. *CACM*, 50(9):46–53, 2007.

[93] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[94] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer, 1996.

[95] P. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440:297–302, 2006.

[96] P. Rothemund, N. Papadakis, and E. Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2(12), Dec. 2004.

[97] G. Rowe. *The Theoretical Models in Biology*. Oxford University Press, 1994.

[98] G. Rozenberg. Computer science, informatics and natural computing—personal reflections. In *New Computational Paradigms: Changing Conceptions of What Is Computable*, pages 373–379. Springer, 2008.

[99] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*. Academic Press, 1980.

[100] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[101] K. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, and M. Hagiya. Molecular computation by DNA hairpin formation. *Science*, 288:1223–1226, 2000.

[102] P. Sazani, R. Larralde, and J. Szostak. A small aptamer with strong and specific recognition of the triphosphate of ATP. *J. Am. Chem. Soc.*, 126(27):8370–8371, 2004.

[103] H.-P. Schwefel. Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik. Dipl.-Ing. Thesis, Tech. Univ. Berlin, 1965.

[104] G. Seelig, D. Soloveichik, D. Zhang, and E. Winfree. Enzyme-free nucleic acid logic circuits. *Science*, 314:1585–1588, 2006.

[105] N. Seeman. Nanotechnology and the double helix. *Scientific American Reports*, 17(3):30–39, 2007.

[106] E. Shapiro and Y. Benenson. Bringing DNA computers to life. *Scientific American*, 294:44–51, May 2006.

[107] W. Sherman and N. Seeman. A precisely controlled DNA biped walking device. *Nanoletters*, 4:1203–1207, 2004.

[108] W. Shih, J. Quispe, and G. Joyce. A 1.7 kilobase single-stranded DNA that folds into a nanoscale octahedron. *Nature*, 427:618–621, 2004.

[109] J. Shin and N. Pierce. A synthetic DNA walker for molecular transport. *J. Am. Chem. Soc.*, 126:10834–10835, 2004.

[110] P. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proc. FOCS*, pages 124–134. IEEE Press, 1994.

[111] K. Sims. Evolving 3D morphology and behavior by competition. In *Proc. Artificial Life IV*, pages 28–39. MIT Press, 1994.

[112] H. Smith, C. Hutchison III, C. Pfannkoch, and C. Venter. Generating a synthetic genome by whole genome assembly: ϕX174 bacteriophage from synthetic oligonucleotides. *PNAS*, 100(26):15440–15445, 2003.

[113] S. Stepney *et al*. Journeys in non-classical computation I: a grand challenge for computing research. *Int. J. Parallel, Emergent and Distributed Systems*, 20(1):5–19, 2005.

[114] M. Stojanovic and D. Stefanovic. A deoxyribozyme-based molecular automaton. *Nature Biotechnology*, 21:1069–1074, 2003.

[115] C. Teuscher. *Turing's Connectionism: An Investigation of Neural Networks Architectures*. Springer, 2002.

[116] J. Timmis, P. Andrews, N. Owens, and E. Clark. An interdisciplinary perspective on artificial immune systems. *Evolutionary Intelligence*, 1(1):5–26, 2008.

[117] R. Ursin *et al*. Entanglement-based quantum communication over 144 km. *Nature Physics*, 3:481–486, 2007.

[118] G. Vichniac. Simulating physics with cellular automata. *Physica D*, 10(1/2):96–116, 1984.

[119] J. von Neumann. *The Computer and the Brain*. Yale University Press, 1958.

[120] J. von Neumann. *Theory of Self-Reproducing Automata*. U. Illinois Press, 1966. Edited and completed by A.W.Burks.

[121] R. Weiss and S. Basu. The device physics of cellular logic gates. In *Proc. The First Workshop on Non-Silicon Computation*, pages 54–61, 2002.

[122] R. Weiss, G. Homsy, and T. Knight, Jr. Toward in-vivo digital circuits. In *Evolution as Computation, Natural Computing Series*, pages 275–295. Springer, 2002.

[123] R. Weiss and T. Knight, Jr. Engineered communications for microbial robotics. In LNCS *2054*, pages 1–16. Springer, 2001.

[124] E. Winfree, X. Yang, and N. Seeman. Universal computation via self-assembly of DNA: some theory and experiments. In *DIMACS Series 44*, pages 191–213. AMS Press, 1999.

[125] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[126] S. Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proc. 6th International Congress of Genetics*, volume 1, pages 356–366, 1932.

[127] B. Yurke, A. Turberfield, A. Mills Jr., F.Simmel, and J.Neumann. A DNA-fuelled molecular machine made of DNA. *Nature*, 406:605–608, 2000.

[128] C. Zimmer. Tinker, tailor: can Venter stitch together a genome from scratch? *Science*, 299:1006–1007, 2003.

[129] K. Zuse. Rechnender Raum. *Elektronische Datenverarbeitung*, 8:336–344, 1967.

Lila Kari (lila@csd.uwo.ca) is Professor and Canada Research Chair in Biocomputing in the Department of Computer Science at the University of Western Ontario, London, Canada.

Grzegorz Rozenberg (rozenber@liacs.nl) is Professor at the Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands, and Adjunct Professor in the Department of Computer Science at the University of Colorado at Boulder, USA.