Phylogeny, Ontogeny, and Epigenesis: Three Sources of Biological Inspiration for Softening Hardware

Eduardo Sanchez, Daniel Mange, Moshe Sipper, Marco Tomassini,
Andres Perez-Uribe, and André Stauffer

Logic Systems Laboratory, Swiss Federal Institute of Technology, IN-Ecublens,
CH-1015 Lausanne, Switzerland. E-mail: {Name.Surname}@di.epfl.ch

Abstract. Living beings are complex systems exhibiting a range of desirable qualifications that have eluded realization by traditional engineering methodologies. In recent years we are witness to a growing interest in Nature exhibited by engineers, wishing to imitate the observed processes, thereby creating powerful problem-solving methodologies. If one considers Life on earth since its very beginning, three levels of organization can be distinguished: the *phylogenetic* level concerns the temporal evolution of the genetic programs within individuals and species, the *ontogenetic* level concerns the developmental process of a single multicellular organism, and the *epigenetic* level concerns the learning processes during an individual organism's lifetime. In analogy to Nature, the space of *bio-inspired* systems can be partitioned along these three axes, phylogeny, ontogeny, and epigenesis, giving rise to the *POE* model. This paper is an exposition and examination of bio-inspired systems within the POE framework. We first discuss each of the three axes separately, considering the systems created to date and plotting directions for continued progress along the axis in question. We end our exposition by a discussion of possible research directions, involving the construction of bio-inspired systems that are situated along two, and ultimately all three axes. This presents a vision for the future which will see the advent of novel systems, inspired by the powerful examples provided by Nature.

1 Introduction: Biological inspiration as a bridge from the natural sciences to engineering

Traditionally, the development of the engineering disciplines (civil, electrical, computer engineering, etc') and that of the natural sciences (physics, chemistry, biology, etc') have proceeded along separate tracks. The natural scientist is a *detective*: faced with the mysteries of nature, such as meteorological phenomena, chemical reactions, and the development of living beings, he seeks to *analyze* existing processes, to *explain* their operation, to *model* them, and to *predict* their future behavior. The engineer, on the other hand, is a *builder*: faced with social and economic needs, he tries to *create* artificial systems (bridges, cars, electronic devices) based on a set of *specifications* (a description) and a set of *primitives* (elementary components such as bricks, beams, wires, motors, transistors).

These two major branches of human endeavor have been drawing closer together during the past decades. It is nowadays common for scientists to use tools created by engineers; to cite one example of many, we are witness to the systematic use of electronics in the medical world for such tasks as decoding the human genome, visually representing highly complex chemical molecules, computerized tomography, and so on.

More recently, engineers have been allured by certain natural processes, giving birth to such thriving domains as artificial neural networks and evolutionary algorithms. Living beings are complex systems exhibiting a range of desirable qualifications, such as evolution, adaptation, and fault tolerance, that have proved difficult to realize using traditional engineering methodologies. Such systems are characterized by a genetic program, the genome, that defines their development, their functioning and their extinction. If one considers Life on earth since its very beginning, then the following three levels of organization can be distinguished [9, 10]:

Phylogeny The first level concerns the temporal evolution of the genetic program, the hallmark of which is the evolution of species, or *phylogeny*. The "multiplication" of living beings is based upon the reproduction of the program, subject to an extremely low error rate at the individual level, so as to insure that the identity of the offspring remains practically unchanged; this error rate is higher at the group or species level [43]. It is precisely these copying errors, due to mutation (asexual reproduction) or mutation along with recombination (sexual reproduction), that gives rise to the emergence of novel species or new organisms. The phylogenetic mechanisms are fundamentally non-deterministic, with the mutation and recombination rate providing a major source of diversity; this diversity is indispensable for the survival of living species, for their continuous adaptation to a changing environment, and for the appearance of new species.

Ontogeny Upon the appearance of multicellular organisms, a second level of biological organization manifests itself. The successive divisions of the mother cell, the zygote, with each newly formed cell possessing a copy of the original genome, is followed by a specialization of the daughter cells in accordance with their environment, i.e., their position within the ensemble; this latter phase is known as cellular differentiation. *Ontogeny* is therefore the developmental process of a multicellular organism; this process is essentially deterministic: an error in a single base within the genome can provoke an ontogenetic sequence which results in notable, possibly lethal, malformations.

Epigenesis The ontogenetic program is limited in the amount of information that can be stored, thereby rendering the complete specification of the organism impossible. A well-known example is that of the human brain with some 10^{10} neurons and 10^{14} connections, far too large a number to be completely specified in the four-character genome of length 3×10^9. Therefore, upon reaching a certain level of complexity, there must emerge a different process that permits the individual organism to integrate the vast quantity of interactions with the outside world. This process is known as *epigenesis*, and primarily includes the nervous system, the immune system and the endocrine system. These systems are characterized by the possession of a basic structure that is entirely defined by the genome (the innate part), which is then subjected to modification through interactions of the individual with the environment (the acquired part). The epigenetic processes can be loosely grouped under the heading of *learning* systems.

In analogy to Nature, the space of *bio-inspired* systems can be partitioned along these three axes: phylogeny, ontogeny, and epigenesis; we refer to this as the *POE* model (Figure 1). As an example, consider the following three paradigms, each of which is positioned along one axis: (P) evolutionary algorithms are the (simplified) artificial counterpart of phylogeny in Nature; (O) self-reproducing automata are based on the concept of ontogeny, where a single mother cell gives rise, through multiple divisions, to a multicellular organism; (E) artificial neural networks embody the epigenetic process, where the system's synaptic weights and perhaps topological structure change through interactions with the environment. Within the domains collectively referred to as *soft computing* [67], characterized by ill-defined problems coupled with the need for continual adaptation or evolution, the above paradigms yield impressive results, rivaling those of traditional methods.

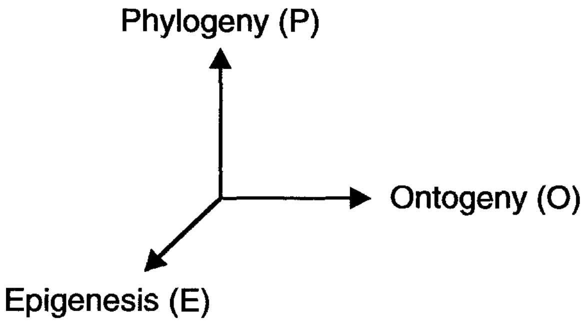

Fig. 1. The POE model. Partitioning the space of bio-inspired systems along three axes: phylogeny, ontogeny, and epigenesis.

Our goal in this paper is to introduce the basics of bio-inspired systems along each of the three axes; due to space restrictions, and in following the main theme of the conference, we shall concentrate on the phylogenetic axis (Section 2). In Section 3 we present a brief account of the ontogenetic axis, considering the role of self-reproduction within the scheme of bio-inspired systems. Section 4 considers the third axis, namely epigenesis. Our paper ends in Section 5 with our conclusions and directions for future research, based on the POE model; specifically, we shall consider the possibilities of combining two axes, along with the ultimate goal of combining all three. This presents a vision for the future which will see the construction of novel systems, inspired by the powerful examples provided by Nature.

2 The phylogenetic axis: Evolvable hardware

2.1 Artificial evolution

The idea of applying the biological principle of natural evolution to artificial systems, introduced more than three decades ago, has seen an impressive growth in the past few years. Usually grouped under the term *evolutionary algorithms* or *evolutionary computation*, we find the domains of genetic algorithms, evolution strategies, evolutionary programming, and genetic programming [2, 17, 22, 28, 32, 41, 42, 56]. As a generic example of artificial evolution, we consider genetic algorithms, originally invented by John Holland in the 1960s [28].[1]

A genetic algorithm is an iterative procedure that consists of a constant-size population of individuals, each one represented by a finite string of symbols, known as the *genome*, encoding a possible solution in a given problem space. This space, referred to as the *search space*, comprises all possible solutions to the problem at hand; generally speaking, the genetic algorithm is applied to spaces which are too large to be exhaustively searched. The symbol alphabet used is often binary due to certain computational advantages purported in [28] (see also [22]); this has been extended in recent years to include character-based encodings, real-valued encodings, and tree representations.

The standard genetic algorithm proceeds as follows [63]: an initial population of individuals is generated at random or heuristically. Every evolutionary step, known as a *generation*, the individuals in the current population are *decoded* and *evaluated* according to some predefined quality criterion, referred to as the *fitness*, or *fitness function*. To form a new population (the next generation), individuals are *selected* according to their fitness, by using a given selection procedure. Selection alone cannot introduce any new individuals into the population, i.e., it cannot find new points in the search space; these are generated by genetically-inspired operators, of which the most well-known are *crossover* and *mutation*. Crossover is performed with probability p_{cross} (the "crossover probability" or "crossover rate") between two selected individuals, called *parents*, by exchanging parts of their genomes (i.e., encodings) to form two new individuals, called *offspring*; in its simplest form, substrings are exchanged after a randomly selected crossover point. This operator enables the evolutionary process to move toward "promising" regions of the search space. The mutation operator is introduced to prevent premature convergence to local optima by randomly sampling new points in the search space. It is carried out by flipping bits at random, with some (small) probability p_{mut}. Genetic algorithms are stochastic iterative processes that are not guaranteed to converge; the termination condition may be

[1] For the purposes of this presentation, the differences with other evolutionary algorithms are inconsequential.

specified as some fixed, maximal number of generations or as the attainment of an acceptable fitness level.

Evolutionary algorithms are ubiquitous nowadays, having been successfully applied to numerous problems from different domains, including optimization, automatic programming, machine learning, economics, immune systems, ecology, population genetics, studies of evolution and learning, and social systems [42]. For recent reviews of the current state of the art, the reader is referred to [62, 63].

2.2 Large scale programmable circuits

An integrated circuit is called programmable when the user can configure its function by programming. The circuit is delivered after manufacturing in a generic state and the user can adapt it by programming a particular function; the programmed function is coded as a string of bits representing the configuration of the circuit. In this paper we consider solely programmable *logic* circuits, where the programmable function is a logic one, ranging from simple boolean functions to complex state machines.

The first programmable circuits allowed the implementation of logic circuits that were expressed as a logic sum of products; these are the PLDs (Programmable Logic Devices), whose most popular version is the PAL (Programmable Array Logic). More recently a new player has appeared on the scene, affording higher flexibility and more complex functionality: the Field-Programmable Gate Array (FPGA) [54]. An FPGA is an array of logic cells placed in an infrastructure of interconnections (Figure 2), which can be programmed at three distinct levels: (1) the function of the logic cell; (2) the interconnections between cells; (3) the input and outputs. All three levels are programmed via a string of bits that is loaded from an external source, either once or several times; in the latter case the FPGA is considered *reconfigurable.*

FPGAs are highly versatile devices that offer the designer a wide range of design choices. However, this potential power necessitates a plethora of tools in order to design a system; essentially, these generate the configuration bit string upon given such inputs as a logic schema or a high-level functional description.

2.3 Evolvable hardware: The present

If one carefully examines the work carried out to date under the heading 'evolvable hardware', it becomes evident that this mostly involves the application of evolutionary algorithms to the synthesis of digital systems [55] (recently, Koza has studied analog systems as well [33]). Taking this point of view, evolvable hardware is simply a sub-domain of artificial evolution, where the final goal is the synthesis of an electronic circuit. The work of [32], where genetic programming was applied to the evolution of a three-variable multiplexer, may be considered an early precursor along this line; it should be noted that at the time the main goal was that of demonstrating the capabilities of the genetic programming methodology, rather than designing actual circuits. We argue that the term *Evolutionary Circuit Design* would be more descriptive of such work than that

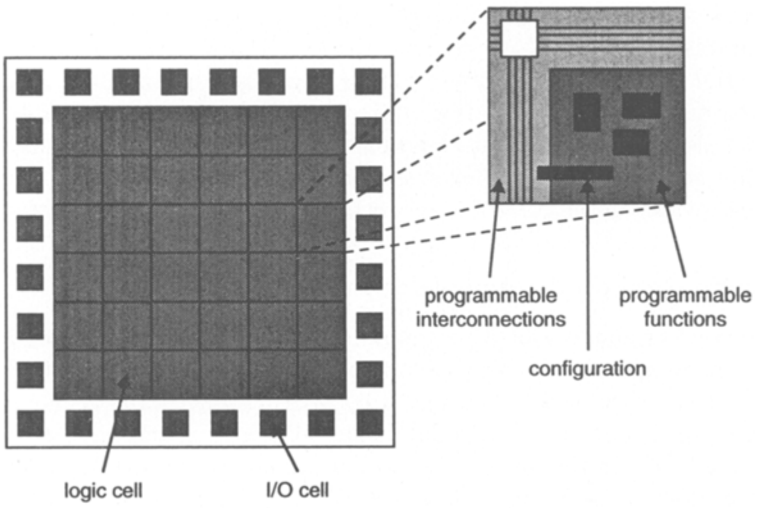

Fig. 2. A schematic diagram of a Field-Programmable Gate Array (FPGA).

of 'evolvable hardware'. For now, we shall remain with the latter (popular) term, however, we shall return to the issue of clarifying definitions in Section 2.5.

Taken as a design methodology, evolvable hardware offers a major advantage over classical methods; the designer's job is reduced to that of specifying the circuit requirements and the basic elements, whereupon evolution "takes over" to "design" the circuit. Currently, most evolved digital designs are sub-optimal with respect to traditional methodologies, however, improved results are continuously attained. Examining work carried out to date, one can derive a rough classification of current evolvable hardware, in accordance with the genome encoding (i.e., the circuit description), and the calculation of a circuit's fitness.

Genome encoding

- *High-level languages.* The first works carried out used a high-level functional language to encode the circuits in question, a representation far-removed from the structural (schematic) description. In [32], the evolved solution is a program describing the (desired) multiplexer rather than an interconnection schema of logic elements (the actual hardware representation). The work of [25] uses a high-level hardware language to represent the genomes. [31, 33] used the rewriting operation, in addition to crossover and mutation, in order to enable the formation of a hierarchical structure; this is still within the framework of a high-level language.
- *Low-level languages.* The idea of directly incorporating within the genome the bit string representing the configuration of a programmable circuit was

expressed early on by [13], though without demonstrating its actual implementation. As a first step one must choose the basic logic gates (e.g., AND, OR, and NOT), and suitably codify them, along with the interconnections between gates, to produce the genome encoding. An example of this approach is the work of [61]. [26] used a low-level bit string representation of the system's logic schema to describe small-scale PALs, where the circuit is restricted to a logic sum of products. The limitations of the PAL circuits have been overcome to a large extent by the introduction of FPGAs, as used, e.g., by [60].

The use of a low-level circuit description that requires no further transformation is an important step forward since this potentially enables placing the genome directly in the actual circuit, thus paving the way toward truly evolvable hardware. However, up until recently, FPGAs had introduced their own share of problems: (1) the genome's length was on the order of tens of thousands of bits, rendering evolution practically impossible using current technology; (2) one still had to extend the genome into a logic schema, a phase for which automatic methods do not exist; (3) within the circuit "space", consisting of all representable circuits, a large number were invalid (e.g., containing short circuits).

With the introduction of the new family of FPGAs, the Xilinx 6200, these problems have been attenuated [60]. As with previous FPGA families, there is a direct correspondence between the bit string of a cell and the actual logic circuit, however, this now always leads to a viable system. Moreover, as opposed to previous FPGAs where one had to configure the entire system, the new family permits the separate configuration of each cell, a markedly faster and more flexible process. [60] has employed this latter characteristic to reduce the genome's size, without, however, introducing real-time, partial system reconfigurations.

Fitness calculation

- *Offline evolvable hardware.* The use of a high-level language for the genome representation means that one has to transform the encoded system in order to evaluate its fitness. This is carried out by simulation, with only the final solution found by evolution actually implemented in hardware. This form of simulated evolution is known as *offline* evolvable hardware [55].
- *Online evolvable hardware.* As noted above, the low-level genome representation enables a direct configuration (and reconfiguration) of the circuit, thus entailing the possibility of using real hardware during the evolutionary process; this has been called *online* evolution by some of the works found in [55].

2.4 Common features of current phylogenetic hardware

Examining work carried out to date we find a number of common characteristics that span both online and offline systems, often differing from biological evolution:[2]

[2] This is not necessarily disparaging, as discussed in Section 5.

- Evolution pursues a predefined goal: the design of an electronic circuit, subject to precise specifications; upon finding the desired circuit, the evolutionary process terminates.
- The population has no material existence; at best, in what has been called online evolution, there is one circuit available, onto which individuals from the (offline) population are loaded *one at a time*, in order to evaluate their fitness.
- The absence of a real population in which individuals coexist simultaneously entails notable difficulties in the realization of interactions between "organisms". This results in a completely local fitness calculation, contrary to nature which exhibits a co-evolutionary scenario.
- If one attempts to resolve a well-defined problem, involving the search for a specific combinatorial or sequential logic system, there are no intermediate approximations; fitness calculation is carried out by consulting a lookup table which is a complete description of the circuit in question. This casts some doubts into the utility of using an evolutionary process, since one can directly implement the lookup table in a memory device, a solution which may often be faster and cheaper.
- The evolutionary mechanisms are carried out outside the resulting circuit. This includes the genetic operators (selection, crossover, mutation) as well as fitness calculation. As for the latter, while what is currently advanced as online evolution uses a real circuit for fitness evaluation, the fitness values themselves are stored elsewhere.
- The different phases of evolution are carried out sequentially, controlled by a central software unit.

2.5 Evolvable hardware: A look ahead along the phylogenetic axis

The phylogenetic axis admits a number of qualitative sub-divisions (Figure 3):

- At the bottom of this axis, we find what is in essence *evolutionary circuit design*, where all operations are carried out in software, with the resulting solution possibly loaded onto a real circuit. Though a potentially useful design methodology, this falls completely within the realm of traditional evolutionary techniques. As examples one can cite the works of [25, 26, 31, 33].
- Moving upward along the axis, one finds works in which a real circuit is used during the evolutionary process, though most operations are still carried out offline, in software. An example is the work of [60], where fitness calculation is carried out on a real circuit. It is important to note that while this has been referred to as online evolution, it would probably be more appropriate to reserve this term for the next sub-division.

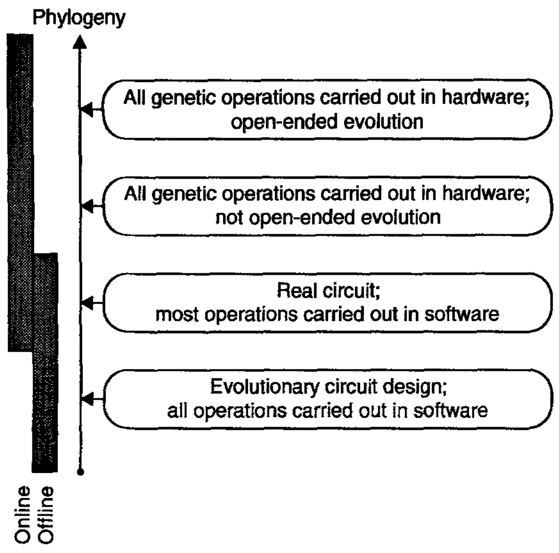
Phylogeny
All genetic operations carried out in hardware; open-ended evolution
All genetic operations carried out in hardware; not open-ended evolution
Real circuit; most operations carried out in software
Evolutionary circuit design; all operations carried out in software
Online
Offline

Fig. 3. The phylogenetic axis.

- Still further along the phylogenetic axis, one finds systems in which *all* genetic operations (selection, crossover, mutation, fitness evaluation) are carried out *online*, in hardware. The major aspect missing concerns the fact that evolution is not open-ended, i.e., there is a predefined goal and no dynamic environment so to speak of. An example is the work of [21].
- This represents the top of the phylogenetic axis, where a *population* of hardware entities evolves in an *open-ended* environment.

We argue that only the last category can be truly considered evolvable hardware, a goal which still eludes us at present. A natural application area for such systems is within the field of autonomous robots, which involves machines capable of operating in unknown environments without human intervention [4]. Another interesting example would be what we call "Hard-Tierra"; this involves the hardware implementation of the Tierra "world" [51], which consists of an open-ended environment of evolving computer programs. A small-scale experiment along this line was undertaken in [18]. The idea of Hard-Tierra is important since it demonstrates that 'open-endedness' does not necessarily imply a real, biological environment.

3 Ontogeny and self-reproducing hardware

The fundamental principle of embryology in real life is illustrated in Figure 4 (based on [11, 12]) which covers a period of two generations preceded and followed by an indefinite number of generations. The first condition is that there must be replicators, entities capable, like DNA molecules, of self-replication. The second condition is our main concern: there must be an embryonic process. The genome should influence the development of the external characteristics of the being, the phenotype; and the replicators must be able to wield some phenotypic power over their world, such that some of them are more successful at replicating themselves than others (this point is crucial for the phylogenetic process). It is important to understand that genes, the basic constituents of the genome, act on two quite different levels: they participate in the embryonic process, influencing the development of the phenotype in a given generation, and they participate in genetics, having themselves copied down the generations (reproduction). This is epitomized by an empirical separation between the disciplines of genetics and embryology; genetics is the study of the vertical arrows in Figure 4, i.e., the relationship between genotypes in successive generations, while embryology is the study of the horizontal arrows, i.e., the relationship between genotype and phenotype in any one generation [12].

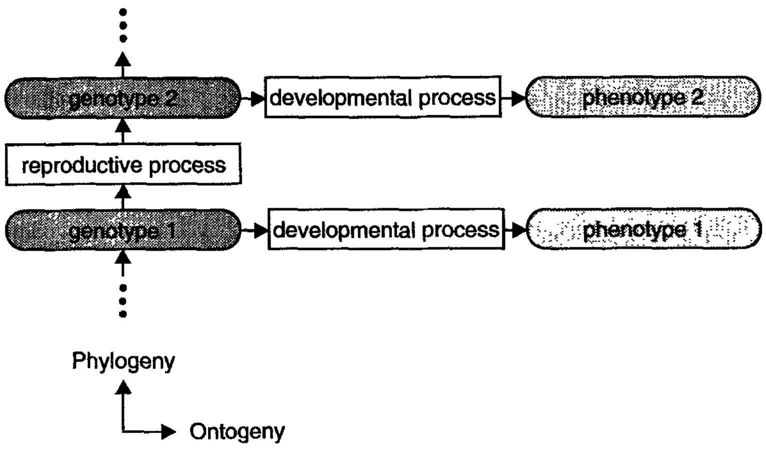

Fig. 4. The embryonic process in Nature: An interplay between phylogeny and ontogeny.

Research into self-reproducing machines, inspired by the ontogeny of living beings, began with von Neumann in the late 1940s. This line of research can be divided into five stages, placed along the ontogenetic axis (Figure 5):

1. Von Neumann [64] and his successors Banks [3], Burks [5], and Codd [8] developed self-reproducing automata capable of universal computation (i.e., able to simulate a universal Turing machine [29]) and of universal construction (i.e., able to construct any automaton described by an artificial genome). Unfortunately, the complexity of these automata is such that no physical im-

plementation has yet been possible, and only partial simulations have been carried out to date.

2. Langton [34] and his successors Byl [6], Reggia *et al.* [52], and Morita *et al.* [45] developed self-reproducing automata which are much simpler and which have been simulated in their entirety. These machines, however, lack any computing and constructing capabilities, their sole functionality being that of self-reproduction.
3. Tempesti [59], and Perrier *et al.* [49] developed self-reproducing automata inspired by Langton's work, yet endowed with finite ([59]) or universal ([49]) computational capabilities.

   In biological terms, a cell can be defined as the smallest part of a living being which carries the complete plan of the being, that is its genome [39]. In this respect, the above self-reproducing automata are unicellular organisms: there is a single genome describing (and contained within) the entire machine; their reproduction is then analogous to the *asexual* reproduction of *unicellular* living beings.
4. Mange *et al.* [36, 38, 39] and Marchal *et al.* [40] proposed a new architecture called *embryonics*, or embryonic electronics. Based on three features usually associated with the ontogenetic process in living organisms, namely multicellular organization, cellular differentiation, and cellular division, they introduced a new cellular automaton, complex enough for universal computation, yet simple enough for a physical implementation through the use of commercially available digital circuits; in addition to self-reproduction, this multicellular "organism" also exhibits self-repair capabilities, another biologically-inspired phenomenon. In order to embed universal construction, they are designing the basic cell with a molecular organization, similar to that of the transcription-translation mechanism (ribosome) [37]. These self-reproducing machines are clearly multicellular artificial organisms, in the sense that each of the several cells comprising the organism contains one copy of the complete genome; their reproduction is analogous to that of asexual multicellular living beings (as in the budding process of the hydra, described by [65]).

   All the above machines are characterized by an *asexual* reproductive process; the genome is therefore haploid.
5. The use of a diploid genome was discussed by [22], and more recently by [27]. This idea, coupled with the recombination of genetic material from two parents, could be introduced within the embryonics framework, representing an ultimate phase with respect to reproducing machines.

4 Epigenesis: Learning through interactions with an environment

To the best of our knowledge, there exist three major epigenetic systems in living multicellular organisms: the nervous system, the immune system and the

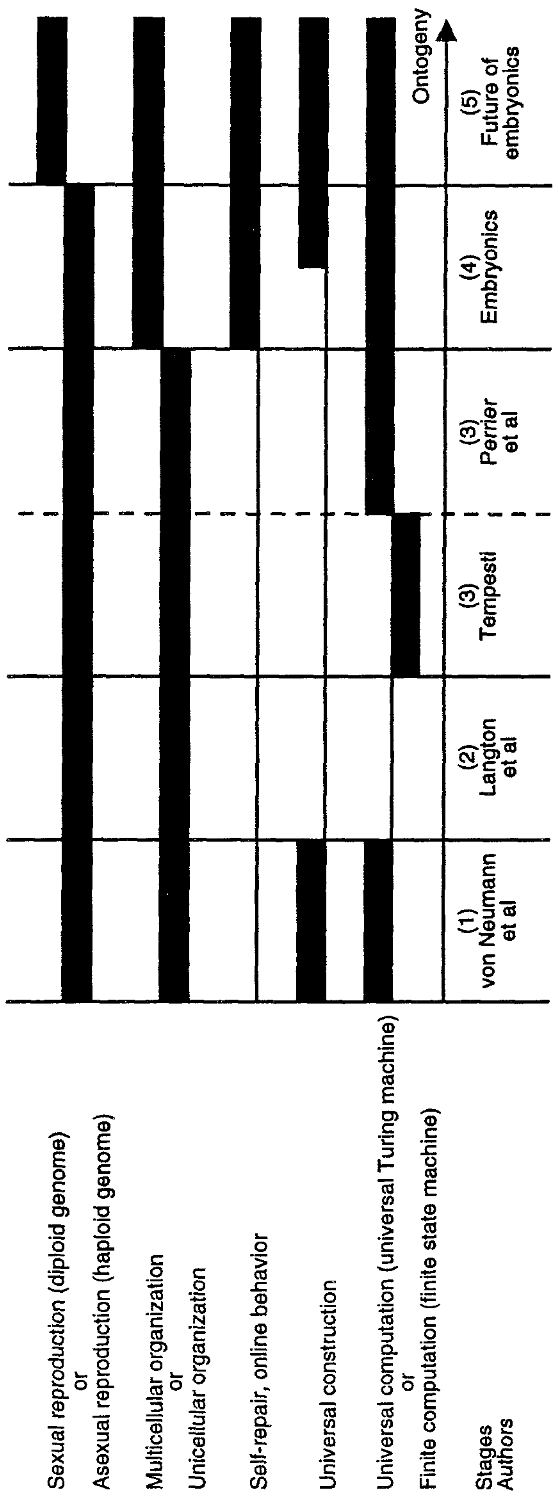

Fig. 5. The ontogenetic axis.

endocrine system, the first two having already served as inspiration for engineers. The nervous system has received the most attention, giving rise to the field of artificial neural networks; this will be the focus of our discussion below. The immune system has inspired systems for detecting software errors [66], as well as immune systems for computers [30]. Immunity of living organisms is a major domain of biology; it has been demonstrated that the immune system is capable of learning, recognizing, and, above all, eliminating foreign bodies which continuously invade the organism. Moreover, when viewed from the engineering standpoint, it is most interesting that immunity is maintained when faced with a dynamically changing environment. This feature leads us to surmise that the immune system, if implemented as an engineering model, can provide a new tool suitable for confronting dynamic problems, involving unknown, possibly hostile, environments.

The nervous system remains the most popular epigenetic model used by engineers. From a biological point of view, it has been determined that the genome contains the formation rules that specify the outline of the nervous system [9, 10]. It is primarily the synapses, the zones of contact between two neurons, where learning takes place, through interactions with the environment during the organism's lifetime. The nervous system of living beings thus represents a mixture of the innate and the acquired, the latter precluding the possibility of its hereditary transmission.

Artificial neural networks have been implemented many times over, mostly in software rather than in hardware, though only the latter concerns us here. Online learning is essential if one wishes to obtain learning systems as opposed to merely learned ones; such systems must learn rapidly from examples with no external guidance. Thus, while neural-network hardware had appeared already in the 1980s [1, 23], only today are we seeing the birth of the technology that enables true online learning. From a hardware point of view, it seems that the possibilities of using memory to retain training examples [53], and the use of adaptable connectivity structures have been widely under-explored; this can be due to the lack of appropriate technology. FPGAs have been used to implement neural networks with a dynamically reconfigurable structure, where neurons and connections may be added or removed, in accordance with environmental changes; this increases the online learning capabilities of the network, coupled with high-speed, parallel operation [47, 48]. The work of [44] has also investigated the possibility of restructuring the network, online. Another recent system of interest is that of Field-Programmable Interconnection Circuits (FPICs), which can be used in conjunction with FPGAs to further improve the network's online capabilities.

Other interesting paths are those that combine two or three axes of the POE model, as discussed in Section 5.

5 Conclusions: Softening hardware by combining phylogeny, ontogeny, and epigenesis

We presented the POE model for classifying soft hardware, based on three axes found in nature: phylogeny, ontogeny, and epigenesis (Figure 1). It is relatively straightforward to place the works presented at ICES'96 along these axes (Figure 6). Taking a look at the results obtained to date reveals a particular emphasis on the phylogenetic axis (fifteen papers, among which six concern offline evolution and nine concern partially or fully online evolution). This conforms with the prime theme of the conference, namely evolvable hardware. The epigenetic axis exhibits seven works on neuronal hardware, which can be grouped into three groups: brainware, learning and control processors for autonomous robots, and artificial neural networks. Finally, the ontogenetic axis is represented by two works concerning uni and multicellular, self-reproducing hardware. In addition, there are a number of overview papers, as well as works concerning specialized hardware systems.

A natural extension which suggests itself is the combination of two, and ultimately all three axes, in order to attain novel bio-inspired hardware (Figure 7). As examples we propose:

- *The PO plane.* This involves self-reproducing, evolving hardware, situated in the phylogenetic-ontogenetic plane. For example, [57, 58] have co-evolved non-uniform cellular automata to act as random number generators; [36] have shown that such evolved generators can be implemented by a multicellular automaton that exhibits self-reproduction and self-repair. Thus, the eventual combination of these two projects can be considered to be in the phylogenetic-ontogenetic plane.
- *The PE plane.* The architecture of the brain is the result of a long evolutionary process, during which a large set of specialized subsystems interactively emerged, carrying out tasks necessary for survival and reproduction [19]. Learning (epigenesis) in biological neural systems can be considered to serve as a mechanism for fine-tuning these broadly laid out neural circuits [24]. Although it is impossible that the genes code all structural information about the brain (Section 1), they may be the ultimate determinant of what it can and cannot learn [7].
  The idea of evolutionary, artificial neural networks, situated in the PE plane, has received attention in recent years; this involves a population of neural networks, where evolution takes place at the global (population) level, with learning taking place at the individual (neural network) level. Examples are the works of [35, 46, 68], though they are currently completely offline. The work of [14, 20] can also be situated in the PE plane, with a partially-online implementation; epigenetic learning takes place online, with the phylogenetic (population) existing offline.
- *The OE plane.* According to selectionism (e.g., [15]), selective pressures operate on epigenetic variation during the ontogeny of the individual (in "somatic" time), not on a phylogenetic time scale [50]. This suggests the pos-

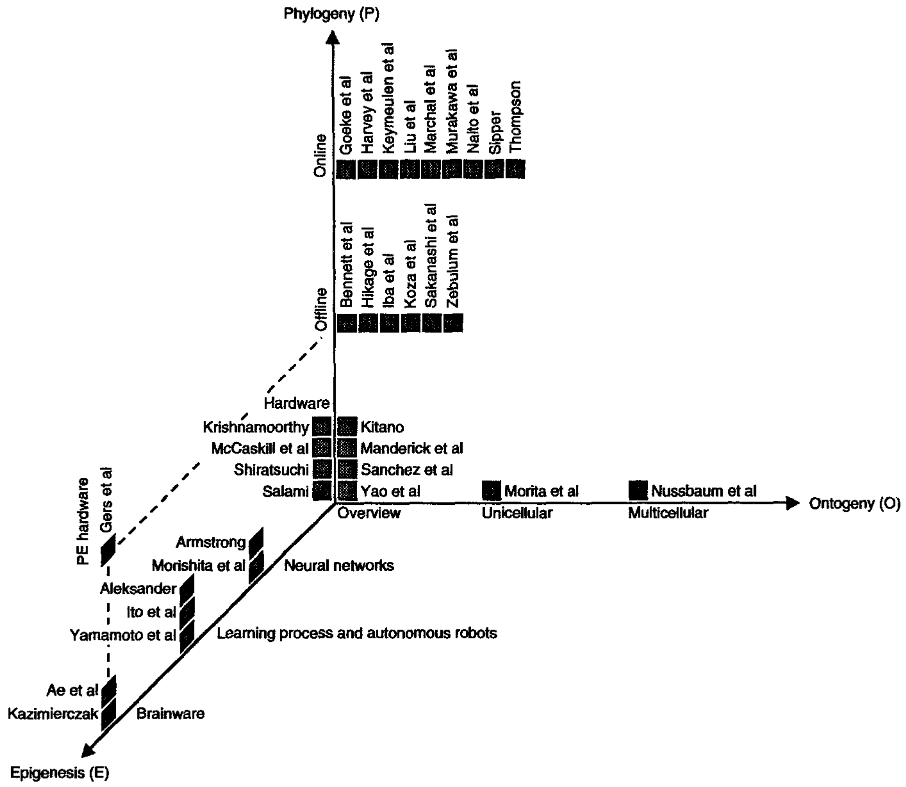

Fig. 6. Placing the works presented at the ICES'96 conference within the POE framework.

sibility of combining the ontogenetic mechanisms discussed above, with the epigenetic (neural network) learning algorithms.

Inductive learning can be interpreted as the capability to infer a response to an unknown situation, achieved through generalizing from previously-encountered, known situations. Engineers are perpetually confronted with a trade-off between generalization and robustness; while adding a multitude of neurons increases the system's fault tolerance, there is a risk of "learning nothing", if we do not attempt to generalize [53]. Implementing neurons in hardware is generally quite expensive, so it is imperative that cost-effectiveness be considered, trying to obtain the smallest possible network. Once a good generalization is obtained (with respect to a certain problem or situation), fault tolerance can be achieved through other self-repair mechanisms, e.g., those used by the embryonics system.

- *The POE space.* The development of an artificial neural network (epigenetic axis), implemented on a self-reproducing multicellular automaton (ontoge-

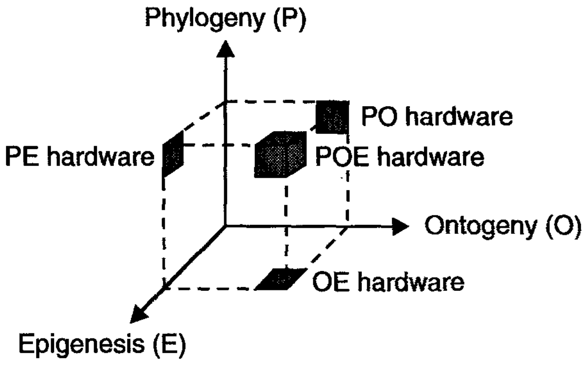

Fig. 7. Combining POE axes in order to create novel bio-inspired systems.

netic axis), whose genome is subject to evolution (phylogenetic axis), constitutes an ultimate example situated in the POE space (Figure 7).

As a final remark we note that the systems considered in this paper are bio-*inspired*; this means that, while motivated by our observations of Nature, we do not have to strictly adhere to its solutions. As an example, consider the issue of Lamarckian evolution, which involves the direct inheritance of acquired characteristics. While the biological theory of evolution has shifted from Lamarckism to Darwinism, this does not preclude the use of artificial Lamarckian evolution [16]. Another example concerns the time scales of natural processes, where phylogenetic changes occur at much slower rates than either ontogenetic or epigenetic ones, a characteristic which need not necessarily hold in our case. Thus, "deviations" from what is strictly natural may definitely be of use in our bio-inspired systems.

Looking (and dreaming) toward the future, one can imagine nano-scale (bioware) systems becoming a reality, which will be endowed with evolutionary, self-reproducing, self-repairing, and neural capabilities; such systems could give rise to novel *species* which will coexist along with carbon-based living beings.

This constitutes, perhaps, our ultimate challenge.

Acknowledgment

We are grateful to Antoine Danchin of the Pasteur Institute, Paris for his careful reading of this manuscript and his many helpful remarks and suggestions.

References

1. L. E. Atlas and Y. Suzuki. Digital systems for artificial neural networks. *IEEE Circuits and Devices magazine*, pages 20–24, November 1989.

2. T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, New York, 1996.
3. E. R. Banks. Universality in cellular automata. In *IEEE 11th Annual Symposium on Switching and Automata Theory*, pages 194–215, Santa Monica, California, October 1970.
4. R. A. Brooks. New approaches to robotics. *Science*, 253(5025):1227–1232, September 1991.
5. A. Burks, editor. *Essays on cellular automata.* University of Illinois Press, Urbana, Illinois, 1970.
6. J. Byl. Self-reproduction in small cellular automata. *Physica D*, 34:295–299, 1989.
7. J. Changeux and A. Danchin. Selective stabilisation of developing synapses as a mechanism for the specification of neural networks. *Nature*, 264:705–712, 1976.
8. E. F. Codd. *Cellular Automata.* Academic Press, New York, 1968.
9. A. Danchin. A selective theory for the epigenetic specification of the monospecific antibody production in single cell lines. *Ann. Immunol. (Institut Pasteur)*, 127C:787–804, 1976.
10. A. Danchin. Stabilisation fonctionnelle et épigénèse: une approche biologique de la genèse de l'identité individuelle. In J. -M. Benoist, editor, *L'identité*, pages 185–221. Grasset, 1977.
11. R. Dawkins. *The Blind Watchmaker.* W.W. Norton and Company, 1986.
12. R. Dawkins. The evolution of evolvability. In C. G. Langton, editor, *Artificial Life*, volume VI of *SFI Studies in the Sciences of Complexity*, pages 201–220. Addison-Wesley, 1989.
13. H. de Garis. Evolvable hardware: Genetic programming of a Darwin machine. In R. F. Albrecht, C. R. Reeves, , and N. C. Steele, editors, *Artificial Neural Nets and Genetic Algorithms*, pages 441–449, Berlin, 1993. Springer-Verlag.
14. H. de Garis. "Cam-Brain" ATR's billion neuron artificial brain project: A three year progress report. In *Proceedings of IEEE Third International Conference on Evolutionary Computation (ICEC'96)*, pages 886–891, 1996.
15. G. M. Edelman. *Neural Darwinism: The Theory of Neuronal Group Selection.* Basic Books, New York, 1987.
16. J. D. Farmer and A. d'A. Belin. Artificial life: The coming evolution. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 815–840, Redwood City, CA, 1992. Addison-Wesley.
17. D. B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence.* IEEE Press, Piscataway, NJ, 1995.
18. P. Galley and E. Sanchez. A hardware implementation of a Tierra processor. Unpublished internal report (in French), Logic Systems Laboratory, Swiss Federal Institute of Technology, Lausanne, 1996.
19. M. S. Gazzaniga. Organization of the human brain. *Science*, 245:947–952, 1989.
20. F. Gers and H. de Garis. CAM-Brain: A new model for ATR's cellular automata based artificial brain project. In *Proceedings of The First International Conference on Evolvable Systems: from Biology to Hardware (ICES96)*, Lecture Notes in Computer Science. Springer-Verlag, Heidelberg, 1996.
21. M. Goeke, M. Sipper, D. Mange, A. Stauffer, E. Sanchez, and M. Tomassini. Online autonomous evolware. In *Proceedings of The First International Conference on Evolvable Systems: from Biology to Hardware (ICES96)*, Lecture Notes in Computer Science. Springer-Verlag, Heidelberg, 1996.

22. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.
23. H. P. Graf and L. D. Jackel. Analog electronic neural network circuits. *IEEE Circuits and Devices magazine*, pages 44–49, July 1989.
24. B. Happel and J. M. Murre. Design and evolution of modular neural network architectures. *Neural Networks*, 7(6/7):985–1004, 1994.
25. H. Hemmi, J. Mizoguchi, and K. Shimohara. Development and evolution of hardware behaviors. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 250–265. Springer-Verlag, Berlin, 1996.
26. T. Higuchi, M. Iwata, I. Kajitani, H. Iba, Y. Hirao, T. Furuya, and B. Manderick. Evolvable hardware and its application to pattern recognition and fault-tolerant systems. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 118–135. Springer-Verlag, Berlin, 1996.
27. T. Hikage, H. Hemmi, and K. Shimohara. Hardware evolution system: Introducing dominant and recessive heredity. In *Proceedings of The First International Conference on Evolvable Systems: from Biology to Hardware (ICES96)*, Lecture Notes in Computer Science. Springer-Verlag, Heidelberg, 1996.
28. J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, Michigan, 1975.
29. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory Languages and Computation.* Addison-Wesley, Redwood City, CA, 1979.
30. J. O. Kephart. A biologically inspired immune system for computers. In R. A. Brooks and P. Maes, editors, *Artificial Life IV*, pages 130–139, Cambridge, Massachusetts, 1994. The MIT Press.
31. H. Kitano. Morphogenesis for evolvable systems. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 99–117. Springer-Verlag, Berlin, 1996.
32. J. R. Koza. *Genetic Programming.* The MIT Press, Cambridge, Massachusetts, 1992.
33. J. R. Koza, F. H Bennett III, D. Andre, and M. A. Keane. Automated WYWIWYG design of both the topology and component values of electrical circuits using genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 123–131, Cambridge, MA, 1996. The MIT Press.
34. C. G. Langton. Self-reproduction in cellular automata. *Physica D*, 10:135–144, 1984.
35. Y. Liu and X. Yao. Evolutionary design of artificial neural networks with different nodes. In *Proceedings of IEEE Third International Conference on Evolutionary Computation (ICEC'96)*, pages 670–675, 1996.
36. D. Mange, M. Goeke, D. Madon, A. Stauffer, G. Tempesti, and S. Durand. Embryonics: A new family of coarse-grained field-programmable gate arrays with self-repair and self-reproducing properties. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 197–220. Springer-Verlag, Berlin, 1996. Also available as: Technical Report 95/154, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, Switzerland, November, 1995.
37. D. Mange, D. Madon, A. Stauffer, and G. Tempesti. Von Neumann revisited: A Turing machine with self-repair and self-reproduction properties. Technical Report

96/180, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, Switzerland, March 1996. (submitted for publication).

38. D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, S. Durand, P. Marchal, and C. Piguet. Embryonics: A new methodology for designing field-programmable gate arrays with self-repair and self-reproducing properties. Technical Report 95/152, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, Switzerland, October 1995.
39. D. Mange and A. Stauffer. Introduction to embryonics: Towards new self-repairing and self-reproducing hardware based on biological-like properties. In N. M. Thalmann and D. Thalmann, editors, *Artificial Life and Virtual Reality*, pages 61–72, Chichester, England, 1994. John Wiley.
40. P. Marchal, C. Piguet, D. Mange, A. Stauffer, and S. Durand. Embryological development on silicon. In R. A. Brooks and P. Maes, editors, *Artificial Life IV*, pages 365–370, Cambridge, Massachusetts, 1994. The MIT Press.
41. Z. Michalewicz. *Genetic algorithms + data structures = evolution programs.* Springer, Berlin, third edition, 1996.
42. M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, 1996.
43. J. Monod. *Chance And Necessity: An Essay On The Natural Philosophy Of Modern Biology.* Vintage, New York, 1971.
44. J. M. Moreno. *VLSI Architectures for Evolutive Neural Models.* PhD thesis, Universitat Politecnica de Catalunya, Barcelona, 1994.
45. K. Morita and K. Imai. Logical universality and self-reproduction in reversible cellular automata. In *Proceedings of The First International Conference on Evolvable Systems: from Biology to Hardware (ICES96)*, Lecture Notes in Computer Science. Springer-Verlag, Heidelberg, 1996.
46. S. Nolfi, D. Parisi, and J. L. Elman. Learning and evolution in neural networks. *Adaptive Behavior*, 3(1):5–28, 1994.
47. A. Perez and E. Sanchez. FPGA implementation of an adaptable-size neural network. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN96)*, volume 1112 of *Lecture Notes in Computer Science*, pages 383–388. Springer-Verlag, Heidelberg, 1996.
48. A. Perez and E. Sanchez. Neural networks structure optimization through on-line hardware evolution. In *Proceedings of the World Congress on Neural Networks (WCNN96)*. INNS (International Neural Networks Society) Press, 1996. (to appear).
49. J. -Y. Perrier, M. Sipper, and J. Zahnd. Toward a viable, self-reproducing universal computer. *Physica D*, 97:335–352, 1996.
50. S. R. Quartz and T. J. Sejnowski. The neural basis of cognitive development: A constructivism manifesto. *Behavioral and Brain Sciences*, 1996. (to appear).
51. T. S. Ray. An approach to the synthesis of life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 371–408, Redwood City, CA, 1992. Addison-Wesley.
52. J. A. Reggia, S. L. Armentrout, H.-H. Chou, and Y. Peng. Simple systems that exhibit self-directed replication. *Science*, 259:1282–1287, February 1993.
53. A. Roy, S. Govil, and R. Mirand. A neural network learning theory and a polynomial time RBF algorithm. *IEEE Transactions on Neural Networks*, 1996. (to appear).

54. E. Sanchez. Field programmable gate array (FPGA) circuits. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, Berlin, 1996.
55. E. Sanchez and M. Tomassini, editors. *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1996.
56. H. -P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, New York, 1995.
57. M. Sipper and M. Tomassini. Co-evolving parallel random number generators. In H. -M. Voigt, W. Ebeling, I. Rechenberg, and H. -P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 950–959. Springer-Verlag, Heidelberg, 1996.
58. M. Sipper and M. Tomassini. Generating parallel random number generators by cellular programming. *International Journal of Modern Physics C*, 7(2):181–190, 1996.
59. G. Tempesti. A new self-reproducing cellular automaton capable of construction and computation. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *ECAL'95: Third European Conference on Artificial Life*, volume 929 of *Lecture Notes in Computer Science*, pages 555–563, Berlin, 1995. Springer-Verlag.
60. A. Thompson. Silicon evolution. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 444–452, Cambridge, MA, 1996. The MIT Press.
61. A. Thompson, I. Harvey, and P. Husbands. Unconstrained evolution and hard consequences. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 136–165. Springer-Verlag, Berlin, 1996.
62. M. Tomassini. A survey of genetic algorithms. In D. Stauffer, editor, *Annual Reviews of Computational Physics*, volume III, pages 87–118. World Scientific, 1995. Also available as: Technical Report 95/137, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, Switzerland, July, 1995.
63. M. Tomassini. Evolutionary algorithms. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 19–47. Springer-Verlag, Berlin, 1996.
64. J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Illinois, 1966. Edited and completed by A.W. Burks.
65. L. Wolpert. *The Triumph of the Embryo*. Oxford University Press, New York, 1991.
66. S. Xanthakis, R. Pajot, and A. Rozz. Immune system and fault-tolerant computing. In *Evolution artificielle 94*. Cepadues, cop., 1995.
67. R. R. Yager and L. A. Zadeh. *Fuzzy Sets, Neural Networks, and Soft Computing*. Van Nostrand Reinhold, New York, 1994.
68. X. Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(3):203–222, 1993.