Chapter 1

An Introduction To Bio-Inspired Machines

Moshe Sipper, Eduardo Sanchez, Daniel Mange, Marco Tomassini, Andrés Pérez-Uribe, and André Stauffer

1.1 Introduction: The POE model of bio-inspired systems

Living organisms are complex systems exhibiting a range of desirable characteristics, such as evolution, adaptation, and fault tolerance, that have proved difficult to realize using traditional engineering methodologies. Recently, engineers have been allured by certain natural processes, giving birth to such domains as artificial neural networks and evolutionary computation. If one considers life on Earth since its very beginning, then the following three levels of organization can be distinguished [25, 28]:

[1]The authors are with the Logic Systems Laboratory, Swiss Federal Institute of Technology, IN-Ecublens, CH-1015 Lausanne, Switzerland. E-mail: {name.surname}@di.epfl.ch. M. Tomassini is also with the Computer Science Institute, University of Lausanne. This chapter is based on the following paper: M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Pérez-Uribe, and A. Stauffer. "A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems." *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pages 83–97, April 1997.

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 organisms is based upon the reproduction of the program, subject to an extremely low error rate at the individual level, so as to ensure that the identity of the offspring remains practically unchanged. Mutation (asexual reproduction) or mutation along with recombination (sexual reproduction) give rise to the emergence of 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 surroundings, i.e., their position within the ensemble. This latter phase is known as cellular differentiation. *Ontogeny* is thus 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 approximately 3×10^9. Therefore, upon reaching a certain level of complexity, there must emerge a different process that permits the individual 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 lifetime 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* hardware systems can be partitioned along these three axes: phylogeny, ontogeny, and epigenesis; we refer to this as the *POE* model (Figure 1.1) [25, 28]. The distinction between the axes cannot be easily drawn where nature is concerned, indeed the definitions themselves may be subject to discussion. Sipper *et al.* [28] therefore *defined* each of the above axes *within the framework* of the POE model as follows: the phylogenetic axis involves *evolution*, the ontogenetic axis involves the *development* of a single individual from its own genetic material, essentially without environmental interactions, and the epigenetic axis involves *learning* through environmental interactions that take place after formation of the individual. As an example, consider the following three paradigms, whose hardware implementations can be positioned along the POE axes: (P) evolutionary algorithms are the (simplified) artificial counterpart of phylogeny in nature, (O) multicellular automata are based on the concept of ontogeny, where a single mother cell gives rise, through multiple divisions, to a multicellular organism, and (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* [33], often involving the solution of ill-defined problems coupled with the need for continual adaptation or evolution, the above paradigms yield impressive results, frequently rivaling those of traditional methods.

This chapter is an exposition and examination of bio-inspired hardware systems within the POE framework. We begin in the next section with an examination of the phylogenetic axis. In Section 1.3 we present the ontogenetic axis, followed by a discussion of the third axis, epigenesis, in Section 1.4. We end this chapter in Section 1.5 with conclusions and directions for future research, based on the POE model.

1.2 The phylogenetic axis

In this section we explore the phylogenetic axis of bio-inspired systems, also referred to as evolvable hardware. The main motivation is to attain adaptive systems that are able to accomplish difficult tasks, possibly involving real-time behavior in a complex, dynamical environment. We begin by briefly introducing two underlying themes, artificial evolution and large scale programmable circuits.

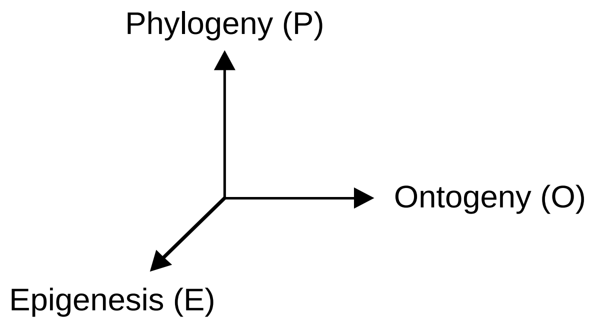

Figure 1.1: The POE model. Partitioning the space of bio-inspired hardware systems along three axes: phylogeny, ontogeny, and epigenesis. See text for definition of these terms.

1.2.1 Artificial evolution

The idea of applying the biological principle of natural evolution to artificial systems, introduced more than three decades ago, has seen 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 [9, 21]. As a generic example of artificial evolution, we briefly consider genetic algorithms [9]. An extensive introduction of evolutionary computation is given in Chapter **??**.

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. The algorithm sets out with an initial population of individuals that 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, and then transformed via genetically inspired operators, of which the most well known are *crossover* and *mutation*. Iterating this procedure, the genetic algorithm may eventually find an acceptable solution, i.e., one with high fitness.

1.2.2 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. In this chapter we consider solely programmable *logic* circuits, where the programmable function is a logic one, ranging from simple boolean functions to complex state machines. The programmed function is coded as a string of bits representing the *configuration* of the circuit. Note that there is a difference between programming a standard microprocessor chip and programming a programmable circuit – the former involves the specification of a sequence of actions, or instructions, while the latter involves a configuration of the machine itself, often at the gate level.

The most commonly used device in the past few years is the field-programmable gate array (FPGA) (for a full treatment of this subject see Chapter **??**). An FPGA is an array of logic cells placed in an infrastructure of interconnections, which can be programmed at three distinct levels: (1) the function of the logic cells, (2) the interconnections between cells, and (3) the inputs and outputs. All three levels are configured 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 suite of tools in order to design a system. Essentially, these generate the configuration bit string, given such inputs as a logic diagram or a high-level functional description.

1.2.3 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 [26] (recently, Koza *et al.* [15] studied analog systems as well). From this perspective, evolvable hardware is simply a sub-domain of artificial evolution, where the final goal is the synthesis of an electronic circuit. The work of Koza [13], which includes the application of genetic programming to the evolution of a three-variable multiplexer and a two-bit adder, 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. Sipper *et al.* [28] argued that the term *evolutionary circuit design* would be more descriptive of such work than that 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 1.2.5.

Taken as a design methodology, evolvable hardware offers a major advantage over classical methods. The designer's job is reduced to constructing the evolutionary setup, which involves specifying the circuit requirements, the basic elements, and the testing scheme used to assign fitness (this latter phase is often the most difficult). If these have been well designed, evolution may then (automatically) generate the desired circuit. Currently, most evolved digital designs are sub-optimal with respect to traditional methodologies, however, improved results are regularly demonstrated [28].

One important distinction that has been made is that between *offline* and *online* evolvable hardware. In the former, evolution is carried out by simulation, with only the final solution actually implemented in hardware. In online evolution one uses real hardware during the evolutionary process.

1.2.4 Common features of current phylogenetic hardware

Examining work carried out to date we find a number of common characteristics that span most current systems, often differing from biological evolution:

- 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 usually results in a completely independent fitness calculation, contrary to nature which exhibits a coevolutionary 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, that must be stored somewhere. This casts some doubts as to the utility of applying 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 executed outside the resulting circuit. This includes the operators (selection, crossover, mutation) as well as fitness calculation. As for the latter, while what has been 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.

1.2.5 Categories of phylogenetic hardware

The phylogenetic axis admits four qualitative sub-divisions (Figure 1.2):

- 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.
- Moving upward along the axis, one finds research in which a real circuit is used during the evolutionary process, though most operations are still carried out offline, in software. It is important to note that while experiments belonging to this category have been referred by some as online evolution, there is a prominent offline aspect since the population is stored in an external computer, which also controls the evolutionary process. It would probably be more appropriate to reserve the term *online* for the next sub-division.
- Still further along the phylogenetic axis, one finds systems in which *all* operations (selection, crossover, mutation), as well as 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 to speak of.

- The last sub-division, situated at the top of the phylogenetic axis, involves a *population* of hardware entities evolving in an *open-ended* environment. When the fitness criterion is imposed by the user in accordance with the task to be solved (currently the rule with artificial evolution techniques), one attains a form of *guided*, or *directed* evolution. This is to be contrasted with *open-ended* evolution occurring in nature, which admits no externally imposed fitness criterion, but rather an implicit, emergent, dynamical one (that could arguably be summed up as survivability). Open-ended, undirected evolution is the only form of evolution known to produce such devices as eyes, wings, and nervous systems, and to give rise to the formation of species. Undirectedness may have to be applied to artificial evolution if we want to observe the emergence of completely novel systems.

We have argued in [28] that only the last category can be truly considered evolvable hardware, a goal which still eludes us at present. We point out that a more correct term would probably be *evolving* hardware. 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 [2]. A related application domain is that of controllers for noisy, changing environments.

1.3 The ontogenetic axis

The ontogenetic axis involves the *development* of a single individual from its own genetic material, essentially without environmental interactions. As can be seen in Figure 1.3 (based on [4]) ontogeny can be considered orthogonal to phylogeny. The main process involved in the ontogenetic axis can be summed up as *growth*, or *construction*. Ontogenetic hardware exhibits such characteristics as replication and regeneration which find their use in many applications. For example, replicating systems have the ability to self-repair upon suffering heavy damage [17] and have been proposed as an economical means of space exploration [6]. Replication can in fact be considered a special case of growth – this process involves the creation of an *identical* organism by duplicating the genetic material of a mother entity onto a daughter one, thereby creating an exact clone. It is important to distinguish between two distinct terms, *replication* and *reproduction,* which are often considered synonymous. Replication is an ontogenetic, developmental process, involving no genetic operators, resulting in an exact duplicate of the parent organism. Reproduction, on the other hand, is a phylogenetic process, involving genetic operators such as crossover and mutation, thereby giving rise to variety and ultimately to evolution (note that reproduction has been justly placed on the vertical axis of Figure 1.3).

Research on ontogenetic systems began with von Neumann's work in the late 1940s on self-replicating machines. This work was later extended by others, and more recently we have seen the emergence of systems that exhibit other ontogenetic mechanisms, such as cellular division and cellular differentiation. This line of research can be divided into four stages, placed along the ontogenetic axis (Figure 1.4):

- Von Neumann [31] and others developed self-replicating automata capable of universal computation (i.e., able to simulate a universal Turing machine) and of universal construction (i.e., able to construct any automaton described by an artificial genome). While the complexity of these automata is such that no full physical implementation has yet been possible, the von Neumann cell has recently been implemented in hardware [27].

- Langton [16] and others developed self-replicating 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-replication.

- Tempesti [29], and Perrier *et al.* [22] developed self-replicating automata inspired by Langton's work, yet endowed with finite [29] or universal [22] computational capabilities.

- One of the defining characteristics of a biological cell concerns its role as the smallest part of a living being which carries the complete plan of the being, that is its genome [19]. In this respect, the above self-replicating automata are *unicellular* organisms: there is a single genome describing (and contained within) the entire machine. Mange *et al.* [17, 18, 19] and Marchal *et al.* [20] 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. The embryonics self-replicating machines are *multicellular* artificial organisms, in the sense that each of the several cells comprising the organism contains one copy of the complete genome. The embryonics framework is detailed in the second part of this book.

Several other works can be placed along the ontogenetic axis, including, e.g., L-systems [23] (see Chapter **??**), cellular encoding [7], graph generation systems [12], and self-replicating programs [14, 24]. Most of these are not considered at length in this book as they are currently implemented solely in software, while our emphasis is on hardware systems.

1.4 The epigenetic axis

The epigenetic axis involves *learning* through environmental interactions that take place after formation of the individual. To the best of our knowledge, there exist three major epigenetic systems in living multicellular organisms: the nervous system, the immune system, and the 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 [32], controllers for mobile robots [10], and immune systems for computers [11]. 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 human endocrine system is made up of a large number of glandular tissues that have in common the fact that they secrete directly into the blood stream chemical messengers, known as hormones, that regulate and integrate bodily functions (such as reproduction). This system resembles in some of its functionalities the nervous system in that both help the individual cope with changes in its environment.

The nervous system remains the most popular epigenetic source of inspiration for engineers (see Chapter **??**). A predominant approach in the field of artificial neural networks consists of applying a learning algorithm to the modification of synaptic weights, using a predesigned network topology. A prime difference between simple *rote* learning and *intelligent* learning is the generalization process taking place in the latter. One can view a predesigned network as an implementation of a *learned* system that exhibits instinctive behavior [30]. Indeed, there is growing evidence that the human brain has many more such instinctive networks than is usually acknowledged, possibly due to their being faster and less resource-demanding with respect to *learning* systems, which adapt continuously within a dynamic environment. Learning networks exhibit the plasticity necessary to confront complex, dynamical tasks, and must be able to adapt at two distinct levels, changing the dynamics of inter-neuron interactions (usually through changes in synaptic weights) as well as modifying the network topology itself. Topology modification has proven to be a successful solution to a problem known as the stability-plasticity dilemma, i.e, how can a learning system preserve what it has previously learned, while continuing to incorporate new knowledge [3]. Evolution may ultimately replace such learning networks by instinctive ones, e.g., via the Baldwin effect, whereby a learned (acquired) behavior becomes embedded within the organism's genome (i.e., its innate behavioral repertoire) through evolution [1, 8] (this may be considered a melange of phylogeny and epigenesis, an issue which shall be expanded upon in Section 1.5).

Artificial neural networks have been implemented many times, mostly in software rather than in hardware. Online learning is essential if one wishes to obtain learning systems as opposed to merely learned ones (Figure 1.5). While neural network hardware had already appeared in the 1980s, only today are we seeing the birth of the technology that enables true online learning. For example, Perez and Sanchez have developed a network architecture with an online dynamic topology (see Chapter **??**).

1.5 Toward novel bio-inspired hardware systems

We presented the POE model for classifying bio-inspired hardware systems, based on three axes: phylogeny, ontogeny, and epigenesis (Figure 1.1). We described each axis and provided examples of systems situated along them. A natural extension which suggests itself is the combination of two, and ultimately all three axes, in order to attain novel bio-inspired hardware (see Figure 1.6 and [28]).

Note that the framework presented in this chapter involves bio-*inspired* systems. This means that, while *motivated* by observations of nature, strict adherence to her solutions is not a *sine qua non*. 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 [5]. 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 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, reproductive, regenerative, and learning capabilities. Such systems could give rise to novel species which will coexist alongside carbon-based organisms.

This constitutes, perhaps, our ultimate challenge.

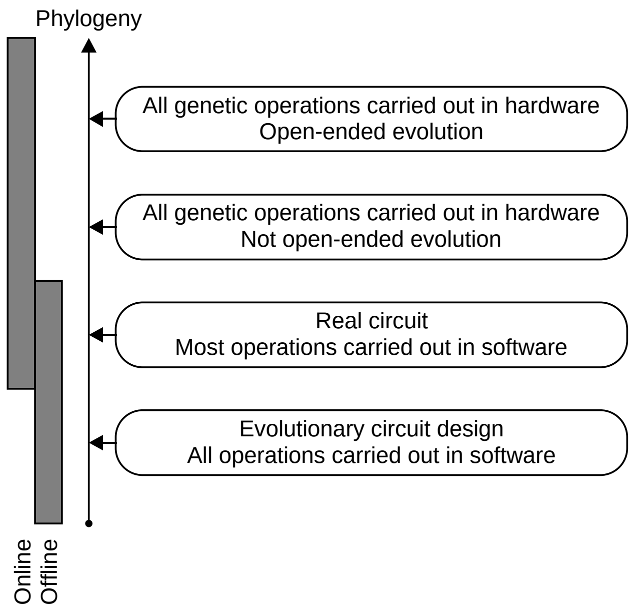
Phylogeny
All genetic operations carried out in hardware
Open-ended evolution
All genetic operations carried out in hardware
Not open-ended evolution
Real circuit
Most operations carried out in software
Evolutionary circuit design
All operations carried out in software
Online
Offline

Figure 1.2: The phylogenetic axis admits four sub-divisions, based on two distinguishing characteristics. The first involves the distinction between *offline* operations carried out in software, and *online* ones which take place on an actual circuit. The second characteristic concerns *open-endedness*. When the fitness criterion is imposed by the user in accordance with the task to be solved, one attains a form of *guided* evolution. This is to be contrasted with *open-ended* evolution occurring in nature, which admits no externally imposed fitness criterion, but rather an implicit, emergent, dynamical one (that could arguably be summed up as survivability).

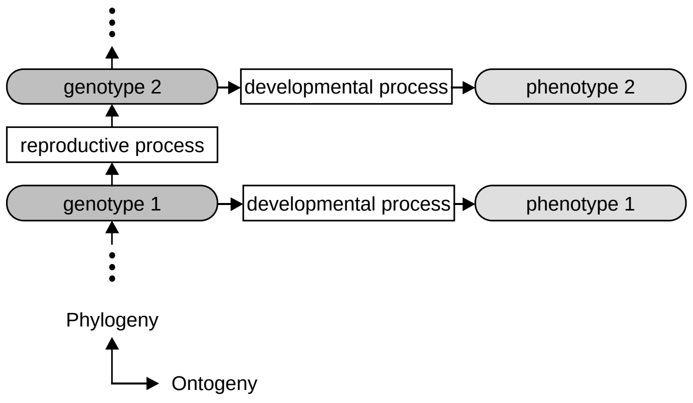

Figure 1.3: The phylogenetic and ontogenetic axes can be considered orthogonal. The figure shows two generations preceded and followed by an indefinite number of generations. Ontogeny involves the development of the phenotype in a given generation (horizontal arrows), while phylogeny involves the succession of generations through reproduction of the genotype (vertical arrows). Note that genes, the basic constituents of the genome, act on two quite different levels: they participate in the developmental process, influencing the development of the phenotype in a given generation, and they participate in genetics, having themselves copied down the generations (reproduction) [4].

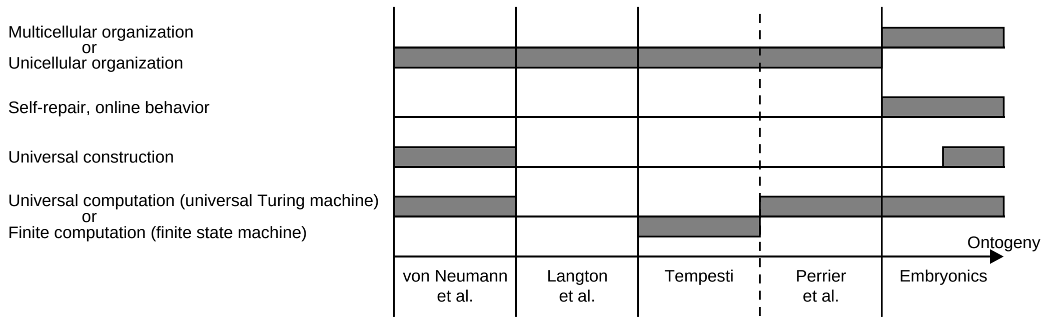

Figure 1.4: The ontogenetic axis admits four stages, based on a number of distinguishing characteristics: universal computation (the ability to simulate a universal Turing machine), universal construction (the ability to construct any automaton described by an artificial genome), self-repair capabilities, and unicellular or multicellular organization.

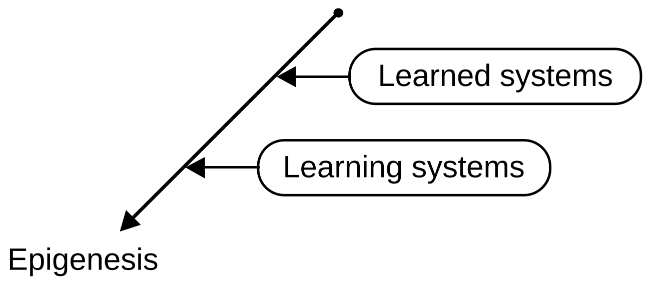

Figure 1.5: The epigenetic axis: moving from learned (instinctive) systems to online learning networks.

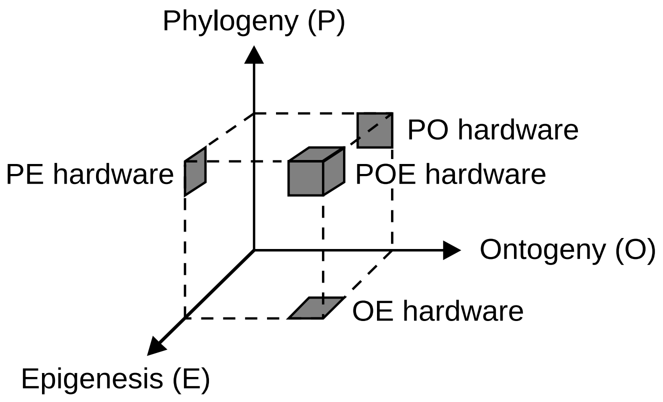

Figure 1.6: Combining POE axes in order to create novel bio-inspired systems: The PO plane involves evolving hardware that exhibits ontogenetic characteristics, such as growth, replication, and regeneration, the PE plane includes, e.g., evolutionary artificial neural networks, the OE plane combines ontogenetic mechanisms (self-replication, self-repair) with epigenetic (e.g., neural network) learning, and finally, the POE space comprises systems that exhibit characteristics pertaining to all three axes. An example of the latter would be an artificial neural network (epigenetic axis), implemented on a self-replicating multicellular automaton (ontogenetic axis), whose genome is subject to evolution (phylogenetic axis).

Bibliography

[1] R. W. Anderson. Learning and evolution: A quantitative genetics approach. *Journal of Theoretical Biology*, 175:89–101, 1995.

[2] R. A. Brooks. New approaches to robotics. *Science*, 253(5025):1227–1232, September 1991.

[3] G. Carpenter and S. Grossberg. The ART of Adaptive Pattern Recognition by a self-organizing neural network. *IEEE Computer*, pages 77–88, March 1988.

[4] 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.

[5] 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.

[6] R. A. Freitas, Jr. and W. P. Gilbreath, editors. *Advanced automation for space missions: Proceedings of the 1980 NASA/ASEE summer study*, chapter 5: Replicating Systems Concepts: Self-replicating Lunar Factory and Demonstration. NASA, Scientific and Technical Information Branch (available from U.S. G.P.O., Conference Publication 2255), Washington, D.C., 1982.

[7] F. Gruau. Artificial cellular development in optimization and compilation. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science*, pages 48–75. Springer-Verlag, Heidelberg, 1996.

[8] G. E. Hinton and S. J. Nowlan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* University of Michigan Press, Ann Arbor, Michigan, 1975. (Second edition, Cambridge, MA: MIT Press, 1992).

[10] A. Ishiguro, T. Kondo, Y. Watanabe, and Y. Uchikawa. Immunoid: An immunological approach to decentralized behavior arbitration of autonomous mobile robots. 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 666–675. Springer-Verlag, Heidelberg, 1996.

[11] 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.

[12] H. Kitano. Designing neural networks by genetic algorithms using graph generation systems. *Complex Systems*, 4:461–476, 1990.

[13] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, Massachusetts, 1992.

[14] J. R. Koza. Artificial life: Spontaneous emergence of self-replicating and evolutionary self-improving computer programs. In C. G. Langton, editor, *Artificial Life III*, volume XVII of *SFI Studies in the Sciences of Complexity*, pages 225–262, Reading, MA, 1994. Addison-Wesley.

[15] 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.

[16] C. G. Langton. Self-reproduction in cellular automata. *Physica D*, 10:135–144, 1984.

[17] D. Mange, M. Goeke, D. Madon, A. Stauffer, G. Tempesti, and S. Durand. Embryonics: A new family of coarse-grained field-programmable gate array 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, Heidelberg, 1996.

[18] D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, and C. Piguet. Embryonics: A new methodology for designing field-programmable gate arrays with self-repair and self-replicating properties. *IEEE Transactions on VLSI Systems*, 6(3):387–399, September 1998.

[19] 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.

[20] 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.

[21] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Heidelberg, third edition, 1996.

[22] J.-Y. Perrier, M. Sipper, and J. Zahnd. Toward a viable, self-reproducing universal computer. *Physica D*, 97:335–352, 1996.

[23] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[24] 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.

[25] E. Sanchez, D. Mange, M. Sipper, M. Tomassini, A. Pérez-Uribe, and A. Stauffer. Phylogeny, ontogeny, and epigenesis: Three sources of biological inspiration for softening hardware. In T. Higuchi, M. Iwata, and W. Liu, editors, *Proceedings of the First International Conference*

on Evolvable Systems: From Biology to Hardware (ICES96), volume 1259 of *Lecture Notes in Computer Science*, pages 35–54. Springer-Verlag, Heidelberg, 1997.

[26] E. Sanchez and M. Tomassini, editors. *Towards Evolvable Hardware*, volume 1062 of *Lecture Notes in Computer Science.* Springer-Verlag, Heidelberg, 1996.

[27] M. Sipper, D. Mange, and A. Stauffer. Ontogenetic hardware. *BioSystems*, 44(3):193–207, 1997.

[28] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Pérez-Uribe, and A. Stauffer. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation*, 1(1):83–97, April 1997.

[29] 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, Heidelberg, 1995. Springer-Verlag.

[30] P. Turney. Myths and legends of the Baldwin effect. In *Proceedings of the 13th International Conference on Machine Learning (ICML-96)*, pages 135–142, 1996.

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

[32] S. Xanthakis, R. Pajot, and A. Rozz. Immune system and fault-tolerant computing. In *Evolution artificielle 94.* Cepadues, cop., 1995.

[33] R. R. Yager and L. A. Zadeh. *Fuzzy Sets, Neural Networks, and Soft Computing.* Van Nostrand Reinhold, New York, 1994.