

Possibilities and Constraints of Basic Computational Units in Developmental Systems

Konstantinos Antonakopoulos and Gunnar Tufte

Norwegian University of Science and Technology

Department of Computer and Information Science

Sem Selandsvei 7-9, NO-7491, Trondheim, Norway

(kostas@idi.ntnu.no, gunnart@idi.ntnu.no)

Abstract

Artificial systems often target organisms or systems with some kind of functionality. Taking inspiration from the cellular nature of its biological counterpart or at a more abstract level, we are in a position to investigate developmental mappings through a cellular or a non-cellular approach. This would lead to a discrete separation of the intermediate phenotype and their computational structures. In this paper, we investigate sparsely connected, simple computational elements that exist in specific architectures (i.e., Boolean network, Cellular automata, Artificial neural network and Cellular neural network), used in developmental settings. In addition, we present an analysis of the possibilities and constraints involved in the development of such computational elements in the targeted architectures focusing on the form, functionality and the inherent biological properties.

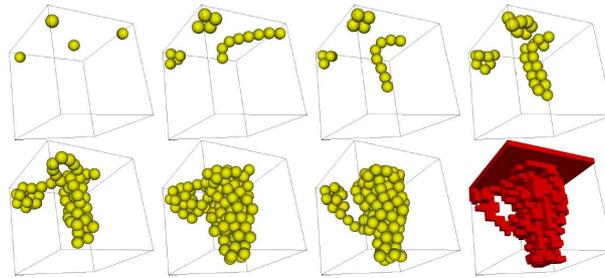
1 Introduction

Recently developmental approaches have once again come to the front in the area of bio-inspired systems. The motivation for moving towards developmental system is multitudinous ranging from early work on self-reproduction [30], to more recent attempts in overcoming limitations in Evolutionary Computation (EC) (e.g., scaling) [14] or towards a desire to include properties inherent in living organisms that may be favorable for artificial systems (e.g., robustness)[18].

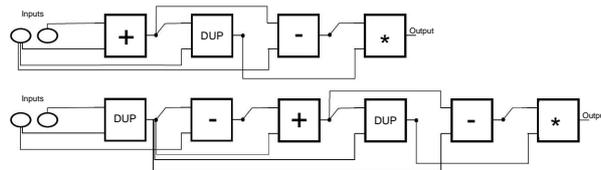
A developmental approach in EC includes an indirect mapping from genotype to phenotype. In biological development an initial unit—a cell, holds the building plan (DNA) for an organism. It is important to note that this plan is generative—it describes how to build the system not what the system will look like. It is of equal importance to note that the information carried in the initial cell is not the only source of information determining the outcome of the developmental process. Furthermore, the emergence of a possible organism is a product of the information in the genome, the environment in which is currently present and possible intermediate phenotypic and behavioral properties that can be exploited by the developmental process [21].

This paper was presented at the NIK-2009 conference; see <http://www.nik.no/>.

The developmental mapping from genotype to phenotype can include various techniques and methods ranging from rewriting systems (e.g., L-systems) [15], to cellular approaches where the genome is present and processed autonomously in every cell [24]. Furthermore, the information available to the mapping process can be varied from systems depending only on the genotype and an axiom [2], to systems including environmental information where exploitation of this information may be included in the creation of the phenotypic structure and behavior [29].



(a) Development of a structure with functional requirements for width and mechanical stress. Eight snapshots of the developmental process shown. Taken from [25].



(b) Development of a computational function. Evaluation is based on the computational function rather than the form of the organism. Two intermediate developmental steps shown. Taken from [11].

Figure 1: Development of structures

Artificial systems including developmental approaches mainly target organisms or systems with some kind of functionality. Functionality in this context may not be restricted to traditional/untraditional computational tasks. Target functionality may include systems aiming to solve e.g., a structural problem. Such problems may be mechanical structures with a weight/stress function [25]. Figure 1, illustrates the difference between the development of structural functions and computational functions.

Figure 1(a), shows the development of a mechanical structure. Specifically, it demonstrates how a cellular structure can grow to an artifact that can support weights placed on the added top plate; shown in the last developmental step. In this approach, the structure itself is the target. Figure 1(b), shows a system aiming for a given computational function. The example is taken from the work on self-modifying functions [11]. The goal herein is a computational function, i.e., modification of data. As such, the functionality of the system is given by the computational function of the nodes and their connections.

This work is part of developing an artificial organism capable of performing a complex computation. To achieve this goal, there is a need for a developmental mapping that can produce phenotypic structures, consisting of connected computational units. To be able to develop such computational structures, we need to acquire further knowledge about the possibilities and constraints of the target architectures. The technology considered is based on circuits implementable in VLSI semiconductor materials.

The choice of technology targets phenotypes that consist of computational units and a connection network. Even though developmental approaches have been successfully applied to what might be considered close to digital circuits [9], other computational paradigms/architectures may be better suited for a developmental approach. Thus, inspiration is taken from the Cellular Computation paradigm [23] towards an approach where behavior emerges from cell interactions in a sparsely connected network. This kind of architecture is appealing from a hardware point of view, offering limited wiring and simple quite uniform computational elements.

The motivation for this work is to acquire further knowledge about the possibilities and constraints of the target architectures with a focus in the development of systems including explicitly, sparsely connected computational elements. The systems investigated herein are Boolean Networks (BN), Artificial Neural Networks (ANN) [1, 7], and Cellular Automata (CA) [28]. In addition to the previous systems already exploited in developmental approaches, we also investigate Cellular Neural Networks (CNN) [5].

The above proposed cellular architectures offer the possibility for massive parallel exploitation. The concept of relative simple, sparsely connected computational elements is favorable for parallel hardware implementations [16, 28]. However, it is hard to exploit the computational power using traditional design and programming methods [22]. As such, an approach towards adaptive methods [23], (e.g., Genetic Algorithm (GA)), with a developmental mapping is sought herein, (i.e., an EvoDevo approach) [21].

An evaluation of the chosen architectures is further motivated by the need to get an overview of the requirements while exploiting developmental mappings, as part of an automatic design tool. The target designs are highly parallel computational circuits. The common property of all architectures (i.e., sparsely connected networks), motivates a further investigation on how a mapping process should work on a class of architectures in a general way. The latter is elaborated by examining how universal properties and processes may be included in artificial development.

The article is laid out as follows: Section 2 introduces the roles of developmental models. The chosen architectures are described in Section 3. The developmental approach to the different architectures are presented in Section 4. Finally, a discussion takes place in Section 5 to be followed by the conclusion in Section 6.

2 A Cellular Approach

In nature the genotype (i.e., the zygote), intermediate phenotypes and the finalized phenotype (i.e., adult), are all part of the interweave processes of evolution and development. The artificial domain is quite different. In an artificial developmental system, evolution and development are not an emergent intermediate result (i.e., the current status of a species at a given point in evolutionary time). The process of evolution is given by some Evolutionary Algorithm (EA), where the developmental process represents a computational mapping process. As such, artificial development may take inspiration from the cellular nature of its biological counterpart or at a more abstract level as an indirect mapping process.

Figure 2, illustrates two different developmental mappings. The left part shows the concepts of a cellular developmental approach, where the right part gives a developmental approach using a non-cellular approach. In the cellular developmental process, the phenotype consists of cells that include the developmental process and the genome. The computational property of the system may be the cell states, resulting from extracellular communication. In the non-cellular approach, the computational property is a result of

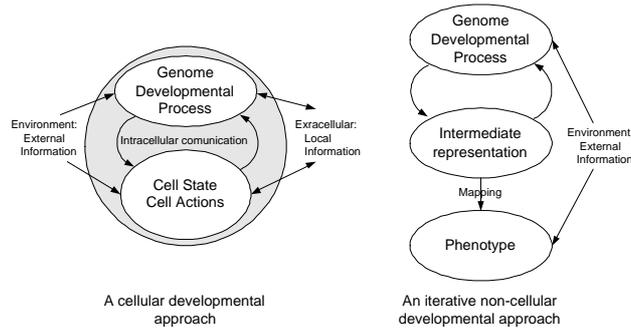


Figure 2: Developmental mapping

a mapping from the developmental process (i.e., the final phenotype), to a technology capable of computation. Therefore, the intermediate phenotypes play different roles. In the cellular mapping the phenotype consists at any time of a set of basic cells. That is, the phenotype and the computational structure is not separated. It includes a number of cells, that is, the result of cell division and some kind of a differentiation process. In this imaginary developmental process, the final and all intermediate phenotypes are structures with computational properties. On the other hand, in the non-cellular approach the developmental process operates on an intermediate representation of the phenotype. Here, only the final phenotype is mapped to a computational structure.

The above difference in the approaches is stated only as an example of possible developmental mappings. As such, there is a variety of ways in how these principles can be utilized.

3 Computational Structures

In Figure 2, the actual computational units and interconnections may be any circuit or structure capable of computation. However in this work, the computational structures or architectures of interest are typical structures that consist of sparsely connected simple computational elements. This type of structure may be achieved by developmental mappings that are close to the cellular approach by e.g., a cellular computation architecture [28], or by systems close to the non-cellular developmental approach e.g., developing neural networks [8]. Even though the approaches are different, a computational structure of sparsely connected elements is achieved.

The computational structures of boolean networks, cellular automata, neural networks and cellular neural networks are presented as computational architectures included in a developmental setting. Figure 3, presents these computational units together with an example of possible interconnections.

BOOLEAN NETWORKS (BN): They have mainly been used towards modelling and understanding gene regulatory mechanisms and in predicting gene expression data. In figure 3(a), a node of a boolean network is shown on top. The node in the example includes three inputs $k = 3$, where the input can be taken from any node in the network. At the bottom of Figure 3(a) a three node $n = 3$, boolean network with $k = 3$ nodes is shown. Each element can be active or inactive (ON or OFF), the number of combinations of states of the K inputs is 2^K [12]. The Boolean function must specify whether the regulated element is active or inactive for each of these combinations. The total number of Boolean functions F of K inputs is $F = 2^{2^K}$. The output result of a boolean network

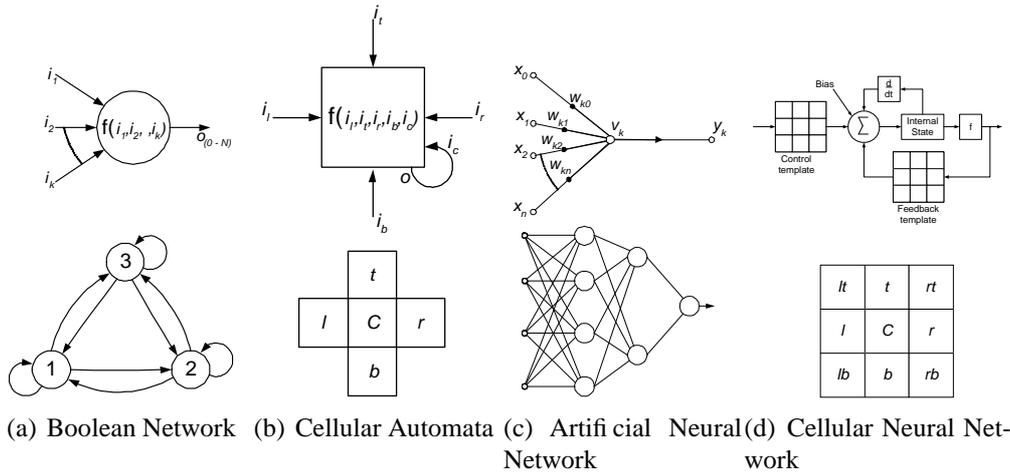


Figure 3: Basic computational units and interconnections.

is given by the states of all nodes in the network. The dynamic behavior of the network is the set of states in the total state space that are visited. The operation of a boolean network requires an initial state i.e., a boolean state, given to each node. All nodes of the network are usually updated synchronously in accordance with the functions assigned to them. This is a repetitive process which continues until the network reaches some predefined criteria or converges to a stable state (i.e., a point attractor), or a repetitive sequence of states (i.e., a cyclic attractor). The behavior of the network including the initial state and all visited states until the system converges can be illustrated as a trajectory. As such, all possible behaviors of a given BN can be represented by the set of trajectories emerging from all possible initial states (i.e., a basin of attractions). The workings of a boolean network is closely connected to trajectories in the basin of attraction. The behavior of the network is measured by the set of states in the total state space that are visited. The visited states along a trajectory is given by the output value, (i.e., ON or OFF), of all nodes in the network at a given time of the dynamic operation of the system. Development of a boolean network requires a developmental mapping that can generate structures that resemble the described structure. In such a mapping, the computational elements (i.e., the boolean function), for each node must be given. In addition, the mapping process need to connect the computational element so as to form the network structure. This requirement calls for a developmental mapping that can grow this kind of network structures. An EvoDevo system targeting computation by BNs, is challenged in the domains of evolution and development. First, a representation for the evolutionary process enabling a fruitful search landscape is needed. Further, the representation and genetic operators must take into account, if the intermediate phenotypes are going to be working BN structures. This implies that all phenotypes - from the first developmental action to the finalized network - include a valid boolean function in all nodes. In addition, the phenotype should have such a network connectivity where all connections are linked to existing nodes. The latter is a requirement for both evolution and development. The representation for the EA needs to provide genetic information to the developmental process that can enable an addressing scheme based on the existing phenotype at any step in developmental time, so as to generate a new valid intermediate phenotype.

CELLULAR AUTOMATA (CA): They are perhaps the quintessential example of cellular computing, conceived in the late 1940s by Ulam and von Neumann. A cellular automaton consists of an array of cells, each of which can be in one of a finite number

of possible states, updated synchronously in discrete time steps, according to a local interaction rule. The spatial lattice of the cells can vary from its simplest form of being a linear chain with N lattice sites to 2-D or 3-D lattice sites of fixed size. In lattices with higher dimension different schemes can be considered, namely, rectangular, triangular or hexagonal. The most commonly studied square (or so-called Euclidean) lattice neighborhood includes the von Neumann neighborhood, consisting of the four size horizontally and vertically adjacent to the center site of interest, and the Moore neighborhood consisting of all eight sites, immediately adjacent to the center site. The neighborhood is not restricted to include information from the closest cells the neighborhood can span over several neighbors.

In Figure 3(b), a 2-D cellular automata cell is shown at the top. The cell includes a simple computational function updated synchronously. The updated output at $t + 1$, of the cell is distributed to all cells in its defined neighborhood, a von Neumann type, shown at the bottom of Figure 3(b). As for BNs, CAs include the same metric of n giving the number of cells and k giving the number of inputs to the computational function of the cell. In Figure 3(b), $k = 5$.

Another property of a cellular automata model is uniformity, that is, the degree of regularity of the model's other properties. Such properties may involve cell issues like, if cells are of the same type (i.e., execute the same function).

The computational operation of CAs as for BNs, start from an initial state, providing an initial value to all cells. The computational output is given by the change in states by an iterative process of updating all cells synchronously. The final state of the system may be reached after an apportioned number of iterations. However, a CA of finite size will always reach a point or cyclic attractor. Taking the computational paradigm of CAs into a developmental system, may exploit the cellular structure to the development of growing structure of CA cells. Such approaches, where the actual CA is the computational structure, explicit development of connectivity is not needed, since the CA neighborhood is specified and all cells share the definition of which cells to communicate with. As such, evolution (e.g., the genome), does not need information storing mechanisms for the wiring process whereas for the developmental process, is also not needed to be able to express such connectivity since it is predefined.

ARTIFICIAL NEURAL NETWORKS (ANN): They have been used extensively in a wide variety of applications namely, image processing, pattern recognition and modeling complex systems. ANNs can be generally divided into feedforward and recurrent, based on their connectivity. An ANN is feedforward if the connectivity is only directing signals from the inputs towards the output. An ANN is recurrent if feedback connections are present. In Figure 3(c), an artificial neuron is shown on top. The neuron includes x_n inputs where $n = 4$, along the weights w_{nk} . The weighted sum of the inputs is forwarded to the activation (or transfer) functions v_k , so as to give the resulting output y_k . At the bottom of Figure 3(c), a four input $n = 4$ neural network with two hidden layers and an output is shown. The architecture of an ANN is determined by its topological structure (i.e., the overall connectivity and transfer function of each node in the network). Additionally, the operation of a neural network requires an input for each neuron along with its respective weight as an initial value (state). The output of a neural network is evaluated in a quite straightforward way; reading the value of the output neuron (y_k).

ANNs have the ability to learn; a feature not present in previous architectures. Learning in ANNs is performed in two steps: training, where the network is trained by adjusting its weights and actual learning which can roughly be either supervised or

unsupervised [31].

Neural weights (or connection weights) play a major role for the mapping process, suggesting potentiality of an axon to grow. Evolution of connection weights introduce an adaptive and global approach to training. Learning rules can also be evolved through an automatic adaptive process for novel rules. Evolution of the connectivity scheme enable neural networks to adapt their topologies to different tasks providing an automated approach. It has been demonstrated that learning and evolution have a special relationship and that learning can be improved by evolution [20].

In a developmental setting, the mapping process needs to connect the computational elements (activation functions), so as to form the network structure. This is a prerequisite for a developmental mapping that is able of growing the outgoing axons of a neuron. Connectivity might be static (predefined) or evolving. As was the case for boolean networks, a developmental neural network is faced with the genotype representation issue and the selection of an efficient EA algorithm. The genotype representation may potentially provide 'working' phenotypes. All phenotypes will contain valid ANN architectures, as soon as the developmental process begins, implying efficient activation functions and weight adaptivity. The transfer function (e.g., sigmoid or Gaussian), of each node is usually fixed and predefined for the whole ANN but it may sometimes vary per layer, or become even more complex [26]. As applies for BNs, the behavior of the ANNs is measured also by the set of states in the total state space. The set of states is given by the output value y_k of the neural network at any time during its dynamic operation. Thus, behavioral capabilities (e.g., functionality), of a neural network are introduced as an emergent property of the interplay among its output neurons, weights and transfer functions.

CELLULAR NEURAL NETWORKS (CNN): They have been mainly used in pattern and character recognition and image processing. The basic architecture and operational principles are very similar to those in CAs. In Figure 3(d), a typical CNN is shown on top. It's basic functional unit is called a cell. Cells are laid down in equally spaces, forming a highly massive parallel computational architecture. Communication between adjacent cells is accomplished directly where information between far apart cells is transmitted through the propagation effects of continuous-time dynamics of CNNs [4]. The Control template initially receives input from any neighboring cell in the grid. The structure of a 3×3 template (be it Control or Feedback), is shown at the bottom of Figure 3(d). The states of the cells are evaluated according to the d/dt fraction. This fraction actually measures how frequent the states of the cells in the Control template will be evaluated. After state evaluation an activation function takes on, giving the output of the CNN. The function's output might processed by the Feedback template-an analogue of ANN's weights-which will adjust the input accordingly. The Feedback template gives the ability for the CNN to act like a recurrent ANN. The Control and Feedback templates can be considered of different neighborhood sizes with fixed number of cells. This is an continuous process, until the desired output is given or bias is reached.

As for CAs, the dynamic behavior of CNNs are largely depended upon the boundary conditions that define the neighborhood of the grid; it is evident that they operate in the edge-of-chaos regime. A CNN with a finite grid size of Control and Feedback templates will always reach an attractor that may be a point or cyclic attractor. A state equilibrium point of a cellular neural network is defined to be the state vector with all its components consisting of stable cell equilibrium states. It was proved that each cell will settle at a stable equilibrium point after the transient has decayed to zero [17]. The transient of a

cellular neural network is the trajectory starting from some initial state and ending at an equilibrium point. This implies that the circuit will not oscillate or become chaotic.

The computational operation of CNNs, as with the CAs, start from an initial state providing an initial value to all cells. The computational output is given by the change in states by a continuous process, of updating cells synchronously. The final state of the system is reached after the bias has been reached or a solution has been found. The activation function plays a role in the computability of the CNN. Although data and parameters in CNNs are typically continuous values, they can operate also as a function of time in discrete environments.

In a developmental setting, challenges with genotype representation and genetic operators are also the case here. If during development, intermediate 'valid' phenotypes is the aim, special concern has to be given to the values of Control and Feedback template, so as they are valid throughout the process. Also, the choice of the activation function is an extra challenge. Apart from the initial computational input of Control template, external input (i.e., environment), may also be applied to the initial input in terms of perturbations.

4 EvoDevo of Computational Structures

The different computational structures of Figure 3, give rise to high levels of parallelism. In every example given, simple computational elements can exceed their function autonomously, as long as valid input conditions are present.

Form and Function

Development with autonomous computational units may exploit the possible computational properties (i.e., functionality) of the structure. This applies from the first developmental stage, (i.e., the zygote) and throughout development, to the end of the life time of the artificial organism. The possibility of monitoring the functional properties of the organism at any time during development, opens for a life-time fitness evaluation [27]. As such, evaluation feedback of the EvoDevo system to the EA can be based on the behavior of the emerging computational structure.

The two developmental options mentioned may be viewed as two extremes of development. If related to Figure 2, the first option can be considered close to a cellular developmental process, where the computational units including connectivity emerge as an interplay among the genome, the intermediate phenotypes and the environment. Using this approach, a phenotype structure could be developed by growth of simple units using, e.g., CA cells. Differentiation could also be applied in order to express the different functional definitions of CA cells. In [28], this approach is applied to develop 2-D non-uniform CA. Further, the developmental process can be included in every cell exploiting simultaneously the dynamic operation of the CA (i.e., the computational and developmental processes).

The non-cellular approach of Figure 2, can use any intermediate representation of the emergent phenotype and map this representation to a target architecture or media. As such, the property of a distributed development where the phenotype emerges out of local cellular communication can be omitted. Departing from a developmental process based only on local properties, allows us to include global properties directly into the process of development. Such global properties may include the order of executing changes in the intermediate structure, solving possible developmental conflicts globally, (e.g., overgrowth) [10], or exploiting global spatial properties (e.g., connections of inputs and

outputs to the developed structure) [3]. Furthermore, global properties may be exploited to control and alter the workings of a developmental process over time (e.g., introducing developmental stages) [6].

The basic computational operations of the chosen architectures are presented in Section 3. Even though they are chosen with the property of being simple computational units, sparsely connected, they may be further separated into three different categories. First, boolean networks, cellular automata and cellular neural networks, all share an emergent expression of the computational result (i.e., the result is given by measuring the output state of all units at some given point or over a period of time). Second, artificial neural networks may be said to share an emergence as the computational result is given by the states of all (or most), of the neurons in the network. However, the output measured is taken from the output neuron(s). As such, the computational result is given by monitoring a single or few of the computational units involved. Third, the dynamics of the architecture in the former category differs in discrete time for boolean networks, cellular automata and artificial neural networks versus continuous time for cellular neural networks. Despite the difference in continuous and discrete operations with regard to time, the architectural way of presenting a result may be considered as the main difference, if such structures are to be exploited in EvoDevo systems.

The connection between the different architectures and the way of presenting results, influences the way evaluation functions may be applied. ANNs open for an explicit evaluation of the output nodes or possibly by applying the output signal directly to parameter control. This can be implicitly evaluated by monitoring the artifact's behavior. Architectures with emergent type output may need to be interpreted so as to be able of measuring and evaluating if the sought emergent behavior is present. As such, the interpretation of the state(s) of the system influences the search space, hence the evolvability depends on the chosen interpretation of the system's output.

If applications are considered, there are obvious tasks that better suit into architectures where the computational result emerges as expressed states by all units, (e.g., gene patterns for boolean regulatory networks [12]). For ANN like architectures, tasks that explicitly require a given number of outputs (e.g., actuator output control signals [7]), are more suitable within this architecture.

Inherent Biological Properties

The properties of adaptivity and robustness are traits of biological organisms that are desired in artificial systems. Adaptivity is one of the key-properties in evolution. However, adaptation related to evolution is population-based adaptation (i.e., changes and emergence of species). In artificial systems, the term adaptation, often relates to a single device and its ability to adjust its properties, to meet some external influence. In an artificial developmental system, the same processes that are exploited to generate a phenotype, can also be exploited for adaptation. The phenotype can be shaped/reshaped to fit changing external conditions (i.e., phenotypic plasticity) [29]. As such, including a developmental process paves the way for achieving adaptation, as this can be an inherent property of development. Biological development includes inherent processes to regenerate or cope with damage of the organism. Such inherent properties may also be part of artificial developmental systems. Similar parts of an organism can be regenerated after damage by the same processes exploited, to develop the artificial organisms [18].

5 Discussion

The computational structures proposed are favorable for exploitation in a evolutionary design approach, that includes a generative or developmental mapping. The two principal features relates to composition and specification of computation. First, all the presented architectures have an inherent composition of small, relatively identical units that represent the principal building blocks of all emerging structures and substructures. Second, the specification of behavior (e.g., computation), is distributed among the computational units.

The architectures for development of computational structures presented herein, are based on the principle of sparsely connected computational units. All units include a possibility for exploiting massive parallel execution. In the presentation of the different computational paradigms, the information required to represent and develop the structures differed mainly in terms of connectivity. The computational units on the other hand, were similar and required information regarding the update function. Note that the actual amount of information that is needed may vary. However for the developmental process, the information required to express a computational unit depends on the set of different update functions apportioned. Thus, the amount of information needed to be expressed by the developmental process for a given set of CA update rules can be expressed by the same information, as an equal-sized set of boolean functions.

In the architectures presented, the computational structure was always sought by an EvoDevo design approach. There is no specific mapping process neither any EA proposed or discussed in detail. The proposed architectures are all used in adaptive design approaches and have shown to be exploitable for such purposes. However, to further expand the variety of designs obtained by EvoDevo, the similarities in these architectures may be further exploited. Instead of designing a mapping process towards a specific architecture (e.g., 2-D non-uniform CA), a mapping approach that can be applied to several architectures should rather be investigated.

Unlike in nature, where all multicellular organisms are a product of evolution and typically share a developmental process, defined by some evolved core processes [13], artificial development has no unified set of processes or mechanisms. The architectures presented are as presented not equal, but quite similar. As such and as a possible future work, the different architectures may be used as a testbed, towards examining how a simple developmental process can be applied to the architectures mentioned herein, where each architecture might be viewed as a species and the set of architectures may be seen as a family.

Towards such an approach, we might explore two possibilities. First, the determination of predefined developmental mechanisms in order to express the sought architecture's inherent properties on connectivity and functionality. The developmental mapping is here crafted and consists of predefined developmental mechanisms (e.g., how cell division, growth and differentiation are determined and executed). As such, investigating these mechanisms would lead to finding useful ways of carrying out the developmental actions and how such actions should be cued. The second possibility suggested, includes an actual search by evolution for the developmental mapping itself. The search is aimed at finding ways of mapping the information at hand (i.e., genome, environment, and intermediate phenotype), to be expressed in the emerging phenotype. This is radically different than the previous approach, as the latter aims towards the evolution of development by changing the view of the existence of the program metaphor [19], from the genome to the developmental mapping. As such, the genome and the

environment represent the input information to the developmental program sought.

Although the genome is the carrier of information from generation to generation in both approaches, the role and information of the genome is very different. In the former, the composition of the genome is sought by evolution as to be read and developed by the defined developmental mapping. In the latter, the evolved result sought is the developmental processes interweaved with information needed to drive the developmental processes, (i.e., genetic information). Pursuing the program metaphor to represent development, may lead towards uncovering common processes that are present in several of the architectures. Such common processes will hopefully help us identify and define useful ways for designing mapping processes that are more universal (i.e., core processes for sparsely connected computational structures).

6 Conclusion

In this ongoing research work, some of the most widely used architectures in a developmental setting have been presented and explored through a EvoDevo system, namely, the cellular and non-cellular approach. The latter included identification of the form and function of the basic computational units involved in such architectures along with their connectivity schemes. The inherent biological properties found in such architectures were also explored. Future work will include potential integration of several of the architectures expressed in a single phenotype. That would lead to new phenotypes and presumably novel functionalities under various environmental conditions.

References

- [1] J. C. Astor and C. Adami. A developmental model for the evolution of artificial neural networks. *Artificial Life*, 6(3):189–218, 2000.
- [2] P. J. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Genetic and Evolutionary Computation Conference (GECCO '99)*, pages 35–43, 1999.
- [3] M. Bidlo and Z. Vacek. Investigating gate-level evolutionary development of combinational multipliers using enhanced cellular automata-based model. In *Congress on Evolutionary Computation (CEC2009)*, pages 2241–2248. IEEE, 2009.
- [4] L. Chua and L. Yang. Cellular neural networks: applications. *IEEE Transactions on Circuits and Systems*, 35(10):1273–1290, October 1988.
- [5] L. Chua and L. Yang. Cellular neural networks: theory. *IEEE Transactions on Circuits and Systems*, 35(10):1257–1272, October 1988.
- [6] K. Clegg and S. Stepney. Analogue circuit control through gene expression. In *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, pages 154–163. Springer, 2008.
- [7] D. Federici. Evolving a neurocontroller through a process of embryogeny. In *Simulation of Adaptive Behavior (SAB 2004)*, LNCS, pages 373–384. Springer, 2004.
- [8] J. Gauci and K. Stanley. Generating large-scale neural networks through discovering geometric regularities. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 997–1004, New York, NY, USA, 2007. ACM.
- [9] T. G. W. Gordon and P. J. Bentley. Development brings scalability to hardware evolution. In *the 2005 NASA/DOD Conference on Evolvable Hardware (EH05)*, pages 272–279. IEEE, 2005.
- [10] P. C. Haddow, G. Tufte, and P. van Remortel. Shrinking the genotype: L-systems for ehw? In *4th International Conference on Evolvable Systems (ICES01)*, Lecture Notes in Computer Science, pages 128–139. Springer, 2001.

- [11] S. L. Harding, J. F. Miller, and W. Banzhaf. Self-modifying cartesian genetic programming. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1021–1028, New York, NY, USA, 2007. ACM.
- [12] S. A. Kauffman. *The Origins of Order*. Oxford University Press, 1993.
- [13] M. Kirschner and J. Gerhart. *The Plausibility of Life: Resolving Darwin's Dilemma*. Yale University Press, 2006.
- [14] H. Kitano. Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems*, 4(4):461–476, 1990.
- [15] A. Lindenmayer. Developmental Systems without Cellular Interactions, their Languages and Grammars. *Journal of Theoretical Biology*, 1971.
- [16] M. A. Maher, S. P. Deweerth, M. A. Mahowald, and C. A. Mead. Implementing neural architectures using analog vlsi circuits. *IEEE Transactions on Circuits and Systems*, 36(5):643–652, May 1989.
- [17] K. Mainzer. *Thinking in Complexity*, chapter 6. Complex Systems and the Evolution of Artificial Life and Intelligence, pages 227–310. Springer-Verlag, 2007.
- [18] J. F. Miller. Evolving a self-repairing, self-regulating, french flag organism. In *Genetic and Evolutionary Computation (GECCO 2004)*, Lecture Notes in Computer Science, pages 129–139. Springer, 2004.
- [19] J. F. Miller and W. Banzhaf. Evolving the program for a cell: from french flag to boolean circuits. In S. Kumar and P. J. Bentley, editors, *On Growth, Form and Computers*, pages 278–301. Elsevier Limited Oxford UK, 2003.
- [20] S. Nolfi and D. Parisi. Evolving artificial neural networks that develop in time. In *Advances in Artificial Life: Proceedings of the third European conference on Artificial Life*, pages 353–367. Springer-Verlag, 1995.
- [21] J. S. Robert. *Embryology, Epigenesis and Evolution: Taking Development Seriously*. Cambridge Studies in Philosophy and Biology. Cambridge University Press, 2004.
- [22] M. Sipper. *Evolution of Parallel Cellular Machines The Cellular Programming Approach*. Springer-Verlag, 1997.
- [23] M. Sipper. The emergence of cellular computing. *Computer*, 32(7):18–26, 1999.
- [24] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Pérez-Urbe, 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.
- [25] T. Steiner, J. Trommler, M. Brenn, Y. Jin, and B. Sendhoff. Global shape with morphogen gradients and motile polarized cells. In *Congress on Evolutionary Computation(CEC2009)*, pages 2225–2232. IEEE, 2009.
- [26] D. Stork, S. Walker, M. Burns, and B. Jackson. Preadaptation in neural circuits. *Proc. of International Joint Conf. of Neural Networks*, I:202–205, 1990.
- [27] G. Tufte. Cellular development: A search for functionality. In *Congress on Evolutionary Computation(CEC2006)*, pages 2669–2676. IEEE, 2006.
- [28] G. Tufte and P. C. Haddow. Towards development on a silicon-based cellular computation machine. *Natural Computation*, 4(4):387–416, 2005.
- [29] G. Tufte and P. C. Haddow. Extending artificial development: Exploiting environmental information for the achievement of phenotypic plasticity. In *7th International Conference on Evolvable Systems (ICES07)*, Lecture Notes in Computer Science, pages 297–308. Springer, 2007.
- [30] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, USA., 1966.
- [31] X. Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(3):203–222, 1993.