Adaptation in a System Metamodel for Evolutionary Computation

Patrik Christen*[1] and Olivier Del Fabbro[2]

[1] *Institute for Information Systems, FHNW University of Applied Sciences and Arts Northwestern Switzerland, Riggenbachstrasse 16, 4600 Olten, Switzerland*
[2] *Chair for Philosophy, ETH Zurich, Clausiusstrasse 49, 8092 Zurich, Switzerland*

31 August 2020

Abstract

Complex systems are difficult to model and analyse with traditional mathematics, e.g. analytically solving differential equations. To better understand such systems, we recently presented a system metamodel to provide an algorithmic alternative to analytical methods and were able to create simulations of complex systems modelled as cellular automata and artificial neural networks. In this study we extend our system metamodel with the concept of adaption in order to integrate evolutionary computation in our so-called allagmatic method. Adaption is described in the context of our system metamodel and the allagmatic method, defined and implemented, and computational experiments with cellular automata and artificial neural networks performed. We find that the system metamodel of the allagmatic method integrates adaptation with an additional operation called adaptation function that operates on the update function, which encodes the system's dynamics. It allows the creation of evolutionary computations by providing an abstract template for adaptation and guidance for implementation with the system metamodel. The creation of the system metamodel was first inspired by abstract concepts of the philosophy of individuation of Gilbert Simondon. The theoretical background for the concept of adaptation in this study is taken from the philosophy of organism of Alfred North Whitehead.

*Corresponding author: patrik.christen@fhnw.ch. The authors contributed equally to this work.

1 Introduction

The history of evolutionary computation is an entangled one. In the 1960s the Germans Ingo Rechenberg, Hans-Paul Schwefel and the Americans Lawrence J. Fogel and John H. Holland developed simultaneously methods for evolutionary computation, calling them different names such as evolution strategies (Evolutionsstrategie) in the case of Rechenberg, evolutionary programming in the case of Fogel, and genetic algorithms in the case of Holland [1, 2].

Over time the different methods were developed further, improved, and became more efficient. By the beginning of the 1990s the field of evolutionary computation was more and more instantiated with scientific societies and journals being created. Contributing to this scientific enterprise was not only the increasing efficiency of the methods but also the applicability thereof. E.g. genetic algorithms are used in all sorts of fields including computational materials discovery to search for certain nanoparticle alloys [3], palaeontology to resolve the influence of environmental change on biodiversity [4], and molecular biology to classify and explore high throughput sequencing data [5].

The applicability of evolutionary computation coincides with the purpose of these methods: general concepts that are being used to solve computational problems [2]. This also led to a different understanding of the concept of intelligence, i.e. instead of believing that the rules underlying intelligence can be encoded in a top-down fashion, a bottom-up approach is being used in which very simple rules are encoded and let to evolve over time [6]. This also leads to the observation of complex behaviour: self-organisation into patterns, chaotic behaviour, adaptive interaction and so on [7].

One key element in evolutionary computation and which allows to grasp the behaviour of systems in the first place is the concept of *adaptation*. That is to say, adaptation consists of a certain target or end at the global level, an optimisation or fitness function to achieve that end, and a fitness measure. Finding that end is an iterative process. From a systems theoretic point of view however and this will be the main issue in this paper adaptation can happen at different levels and in different manners: i.e. either the set of rules can be adapting or the agents, cells or elements or simply the boundary conditions of a system.

We recently presented a system metamodel to model systems [8, 9, 10], especially complex systems, that allows to follow the process of creation of computer models and thereby to better understand that process. Based on philosophical concepts borrowed from the philosophy of Gilbert Simondon [11], the generic model building blocks entity representing an element or agent, milieu representing the interaction of entities within a neighbourhood, and update function representing the rule or law each entity is following in a system were defined and implemented in computer code. In this study, we want to extend our system metamodel by explicitly accounting for adaptive agents. We already defined an adaptation function in addition to the update function that changes the rules or law of the update function [10], however, in the present study we provide a description of it in the context of adaptation and complex adaptive systems as well as its implementation in computer code within the system metamodel. To give adaptation a theoretical grounding, we rely on the philosophy of organism of Alfred N. Whitehead. We show that Whitehead's philosophy of organism is quite similar to Simondon's philosophy of individuation and at the same time Whitehead deepens his thoughts on the concept of adaptation suitable for implementation in our system metamodel. The goal of this system metamodel is not necessarily optimisation or efficiency but the better understanding of computational processes in evolutionary computation.

2 System Metamodel

Using the recently developed and implemented metamodel for modelling systems, we directly [8] and automatically [9] created cellular automata and artificial neural networks with the basic building blocks of the metamodel. The system metamodel is inspired by the philosophical concepts structure and operation as proposed by Simondon to describe a system [11]. Each system, natural or artificial, consists of at least one structure capturing the spatial dimension and at least one operation capturing the temporal dimension. Operation is thereby acting on the structure over time. We formally defined this at the most abstract level as follows [10]:

Definition 2.1. *Let $\mathcal{SM}$ denote a model of a system, S its set of structures or spatial domains, and O its set of operations or temporal domains. Then $\mathcal{SM}$ is a $(s+o)$-tuple consisting of s structures and o operations, where $s, o \geq 1$ and therefore:*

$$\mathcal{SM} \coloneqq (\hat{s}_1, \hat{s}_2, \hat{s}_3, \ldots, \hat{s}_s, \hat{o}_1, \hat{o}_2, \hat{o}_3, \ldots, \hat{o}_o), \tag{1}$$

where $\hat{s}_i \in S \wedge \hat{o}_j \in O$.

More concretely, structure S and operation O are described in more detail by further dividing them into more specific but still general concepts. Guided by the systems view, systems are made up by interacting entities defined with an entity e-tuple $\mathcal{E} = (\hat{e}_1, \hat{e}_2, \hat{e}_3, \ldots, \hat{e}_e)$, where $\hat{e}_i \in Q$ with Q being the set of k possible states [10]. These entities are interacting with each other within a certain neighbourhood or milieu defined with the milieus e-tuple $\mathcal{M} = (\hat{\mathcal{M}}_1, \hat{\mathcal{M}}_2, \hat{\mathcal{M}}_3, \ldots, \hat{\mathcal{M}}_e)$, where $\hat{\mathcal{M}}_i = (\hat{m}_1, \hat{m}_2, \hat{m}_3, \ldots, \hat{m}_m)$ is the milieu of the i-th entity $\hat{e}_i$ of $\mathcal{E}$ consisting of m neighbours of $\hat{e}_i$ [10]. Additionally, there can be structures for storing some information used by operations. The most basic operation updates the states of an entity depending of the states of neighbouring entities and some local rules. We defined it as an update function $\phi : Q^{m+1} \to Q$ and introduced the structure $\mathcal{U}$ storing the rules or logic of the function [10]. Within the context of evolutionary computation, we introduce here a further operation acting on the structure $\mathcal{U}$ and that is capable of adapting or changing them. We defined it as an adaptation function ψ with its own logic stored in the structure $\mathcal{A}$ [10]. We can now give a more complete definition of structure and operation as follows:

Definition 2.2. *The set of structures S consists of the entities e-tuple $\mathcal{E} = (\hat{e}_1, \hat{e}_2, \hat{e}_3, \ldots, \hat{e}_e)$, where $\hat{e}_i$ is in the set Q of k possible states, the milieus e-tuple $\mathcal{M} = (\hat{\mathcal{M}}_1, \hat{\mathcal{M}}_2, \hat{\mathcal{M}}_3, \ldots, \hat{\mathcal{M}}_e)$, where $\hat{\mathcal{M}}_i = (\hat{m}_1, \hat{m}_2, \hat{m}_3, \ldots, \hat{m}_m)$ is the milieu of the i-th entity $\hat{e}_i$ of $\mathcal{E}$ consisting of m neighbours of $\hat{e}_i$, the update rules u-tuple $\mathcal{U}$, the adaptation rules a-tuple $\mathcal{A}$, the adaptation end p-tuple $\mathcal{P}$, and possibly further structures $\tilde{s}_i$, leading to the following definition [10]:*

$$S \coloneqq \{\mathcal{E}, Q, \mathcal{M}, \mathcal{U}, \mathcal{A}, \mathcal{P}, \ldots, \tilde{s}_s\}. \tag{2}$$

Definition 2.3. *The set of operations O consists of, at least, an update function $\phi(\hat{e}_i, \hat{\mathcal{M}}_i, t, \mathcal{U})$, and optionally an adaptation function $\psi(g, \mathcal{A}, \mathcal{P}, l)$ as well as possible further operations $\tilde{o}_j$, leading to the following definition [10]:*

$$O \coloneqq \{\phi(\hat{e}_i, \hat{\mathcal{M}}_i, t, \mathcal{U}), \psi(g, \mathcal{A}, \mathcal{P}, l), \ldots, \tilde{o}_o\}, \tag{3}$$

where t is the number of time steps, g the number of adaptation iterations, and l the loss tolerance.

It follows that $\mathcal{SM}$ can be described more precisely [10]:

Definition 2.4. *Let $\mathcal{SM}$ denote a model of a system, S its set of structures or spatial domains capturing entities in $\mathcal{E}$ and their possible states in Q and local neighbourhood or milieu in $\mathcal{M}$ and structures related to update function in $\mathcal{U}$ and to the adaptation function in $\mathcal{A}$ and $\mathcal{P}$, and O its set of operations or temporal domains capturing the dynamic state transitions of the entities $\mathcal{E}$ with the function ϕ and possible adaptations* *or evolutions of the system with the function ψ. Then $\mathcal{SM}$ is a $(s+o)$-tuple consisting of s structures and o operations, where $s, o \geq 1$ and therefore:*

$$\mathcal{SM} := (\mathcal{E}, Q, \mathcal{M}, \mathcal{U}, \mathcal{A}, \mathcal{P}, \ldots, \hat{s}_s, \phi, \psi, \ldots, \hat{o}_o), \tag{4}$$

where $\hat{s}_i \in S \wedge \hat{o}_j \in O$.

The system metamodel is applied in the so-called allagmatic method[8] following three different regimes (Fig. 1): In the virtual regime, the system metamodel is described abstractly. In the metastable regime, structure and operation are combined and fed with model parameters. This creates an actual system that can be run in the actual regime.

In this study, we added the adaptation operation to the system metamodel implementation as well as reimplemented the rest of the metamodel in accordance with the definitions above. As in the previous implementation [8] and guided by the philosophy of Simondon [11], object-oriented programming and template meta-programming in C++ was used. Classes are used to implement structures as member data and operations as methods in the virtual regime. Since the model parameters become concrete only through parametrisation in the metastable regime, generic types used in the virtual regime implemented with a vector template.

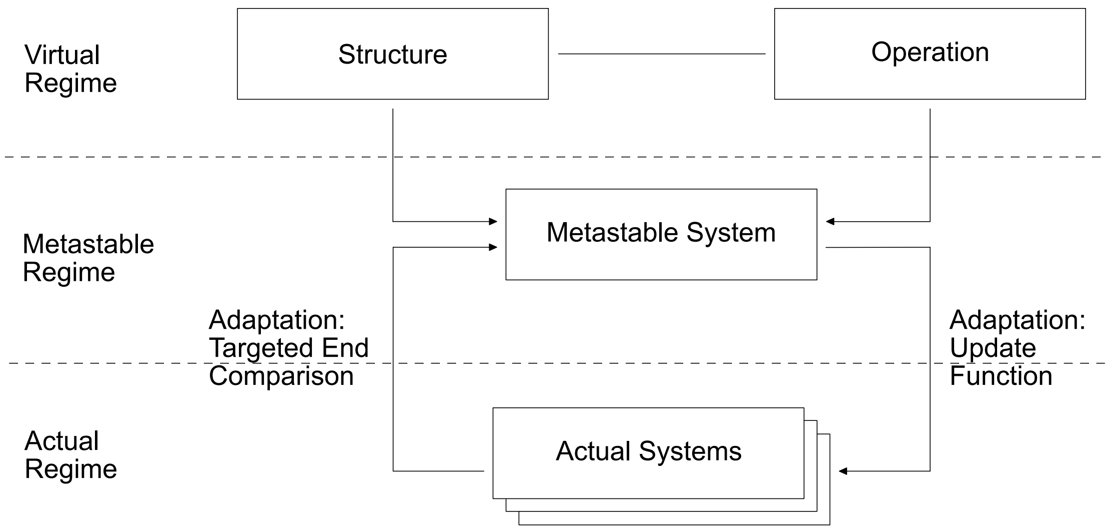

Figure 1: The allagmatic method consisting of a system metamodel that is abstractly described in the virtual regime, concretised with parameters in the metastable system, and run or executed in the actual regime. Adaptation as introduced in this study is occurring between the metastable and actual regime.

3 Meta-Theoretic Definition of Adaptation

Before describing how the adaptation function is integrated into the allagmatic method and system metamodel, and how it operates in concrete models such as cellular automata and artificial neural networks, it is necessary to give philosophical definitions of adaptation on a meta-theoretical level and how it was implemented. These meta-theoretical definitions of adaptation in turn follow the philosophy of organism of Alfred North Whitehead, because his philosophy is on the one hand similar to that of Simondon [12], and on the other hand Whitehead has produced a more thorough analysis and deployment of the concept of adaptation.

Already in our previous studies [8, 9, 10] the concept of entity, which constitutes the very basic structure of any computer model, is borrowed from Whiteheads philosophy of organism, when he uses the concept of actual entity to denote the final real things of which the world is made up. [13]. Similar to our model in which entities form together a milieu or neighbourhood, Whitehead believes that actual entities form together a so-called nexus. But actual entities are in turn only capable of forming such a nexus because of their prehension of each other. Prehension is thus the abstract concept analogous to the change of rules in the update function that defines how entities enter into a relational setting of reciprocal influence.

The important point in regard to adaptation is that it gives the entities not only a goal, target, or end (a telos in philosophical terms) that needs to be attained (or a problem to be solved), adaptation also gradually allows to define order respectively disorder. This means that those entities which together form a first nexus are then fully ordered, when the attainment of an end, goal, or target is achieved. Hence, when the behaviour of entities is structured through an adaptive operation (an adaptation function) and the given end of that function is attained, a so-called society or an individual is formed. Society, as it is defined by Whitehead, should therefore not be understood in the narrow and common sense of a social gathering of humans studied by sociology. It is rather the computed states of the system of every iteration stored in the program, which is the actual system that reaches its end. In contrast to prehension, this is the structural view upon the system.

For both, a nexus of interacting entities and a society, all entities interact together in order to follow a certain path (update function) but only in a society an end is attained (adaptation function). This phenomenon can, according to Whitehead, also be observed in biology or physics, when molecules form a cell or waves and electrons an electromagnetic field. We will see in the next section that this conceptual difference between an individual or a society and a nexus is important because it allows to describe those computed iterations, which did not attain the pre-defined end. These results that did not attain the end, but nevertheless produced certain patterns, are in this sense disordered. Nexs (the plural of nexus) are thus defined as actual systems that did not reach the targeted end. When a society is ordered, i.e. when it has attained a certain end, it is self-sustaining; in other words, that it is its own reason. [13].

Notice at this point that order and disorder are not described structurally but rather operationally, i.e. they are defined by what the system does and not what it is. Due to its processual definition, order is a gradual concept that attains its complete definition when an end has been attained. Order is hence also present within disordered nexs, respectively ordered societies emerge from disorder. Chaos, or chaotic disorder, is therefore a nexus which lacks of compatibility between the entities to fulfil a certain targeted end. Adaptation is thus not solely a concept describing the behaviour of completed individuals, it allows to grasp the emergence of individuals – their individuation as Simondon would say. Individuation is for this reason nothing else than the processual-operational view of a society.

Whitehead's rather processual definition of adaptation and its difference from mainstream definitions in biology that focus on already given individuals can also be seen in his definition of survival [14]. If surviving means for entities to form a society that is capable of persisting over a certain amount of time, then this means that the society has become on the one hand a stable society and on the other hand capable of adapting to the given environment. The stable society has become specialised because it answers to specific conditions and features of the environment. In other words: the structure of the society becomes complex. Notice that in Whitehead's metaphysics the environment itself is nothing else than a constitution of mixed nexs and societies. But in order to survive or to persist, societies need to be flexible, i.e. they need to retain a certain degree of unspecialised behaviour due to the changing of the environment. Hence, societies need to find a midway between specialisation in order to cope with problems and flexibility in order to adjust to newly arising problems.

It might also be that entities within societies are themselves societies such as molecules within a living cell. These molecules are then called a subordinate society. This in turn means that the concept of hierarchy and control needs to be introduced. On the one hand some societies have their own autonomy and order, on the other hand these same societies can be subdued to the control of other societies. A society in turn that manages to bring together different subordinate societies is defined as a structured society.

4 Adaptation and the Allagmatic Method

Here, we suggest to integrate adaptation into the system metamodel and the allagmatic method with the guidance of Whitehead's metaphysics. However, we first explain the details of the update function that was created and implemented according to the philosophy of Simondon [11] in earlier studies [8, 9, 10].

The distinction of structure and operation in a system is borrowed from Simondon but he also suggests that these operations act upon structures in three different regimes (Fig. 1), which provides a general framework for describing the functioning of a system in terms of structure and operation. At the most abstract level in the virtual regime, the update function is an operation of the system model and is defined as a mathematical function $\phi(\hat{e}_i, \hat{\mathcal{M}}_i, t, \mathcal{U}) : Q^{m+1} \rightarrow Q$, where all its parameters remain abstract in that regime. Therefore, ϕ is implemented as a method of the class `SystemModel` referring to $\mathcal{SM}$. The function itself is specific to the application and is thus not defined abstractly. Several existing and concrete update functions as used in elementary cellular automata [15] and simple feedforward artificial neural networks [16] are thus implemented in the class `SystemModel`. Please note that the parameters are the same for each of these methods and are implemented as generic type and with dynamic size to achieve the abstract description in the virtual regime. In the metastable regime, the main program instantiates a `SystemModel` object, where the types are defined and therefore concretised through template meta-programming. A vector with entity objects of the class `Entity` is created next and assigned to the member data `entities` of the class `SystemModel`. With that, the system is further concretised with the number of entities. In the same way, $\mathcal{M}$ and $\mathcal{U}$ are created and assigned to the member data `milieus` and `updateRules` of the class `SystemModel`, respectively. With that creation, the system is further concretised with the specific neighbourhood and thus topology of the entities as well as the specific update rule. By providing the number of time iterations t, the concretisation of the `SystemModel` object in the metastable regime is finished. In the actual regime, the `SystemModel` object is run or executed with the specified parameters for t iterations.

To run evolutionary computations that are capable of adapting, we extended our system metamodel with a further model building block called *adaptation function* in the present study guided by the metaphysics of Whitehead [13]. It is now described in a similar way as the update function in the previous paragraph in the context of the allagmatic method and its implementation in the system metamodel as well as the guidance from Simondon's and Whiteheads metaphysics with respect to adaptation. Again we start with the distinction of structure and operation acting in the three different regimes.

At the most abstract level in the virtual regime, the adaptation function is an operation of the system model and is defined as a mathematical function $\psi(g, \mathcal{A}, \mathcal{P}, l)$, where all its parameters remain abstract in that regime. Therefore, ψ is implemented as a method of the class `SystemModel` referring to $\mathcal{SM}$. The function itself is specific to the update function and is thus not defined abstractly. We implemented specific adaptation functions for cellular automata [15] and simple feedforward artificial neural networks [16] in the class `SystemModel`. There are 2^8 possible rules in the update function of an elementary cellular automata. We thus implemented an adaptation function that randomly selects one of these possible rules without considering any error or fitness measure. In contrast, for the update function of artificial neural networks, an adaptation function that adapts the weights of the network taking into account the error according to the so-called perceptron learning rule [16]. Both adaptation functions are implemented in the class `SystemModel`. Please note that although some parameters might not be used in a specific method, the ones that are used are the same for each of these methods and are implemented as generic type and with dynamic size to achieve the abstract description in the virtual regime. In the metastable regime (Fig. 1), identical to the update function description above, the main program instantiates a `SystemModel` object, where the types are defined and therefore concretised through template meta-programming. A vector with entity objects of the class `Entity` is created next and assigned to the member data `entities` of the class `SystemModel`. With that, the system is further concretised with the number of entities. In the same way, $\mathcal{M}$ and $\mathcal{U}$ are created and assigned to the member data `milieus` and `updateRules` of the class `SystemModel`, respectively. With that creation, the system is further concretised with the specific neighbourhood and thus topology of the entities as well as the specific update rule. By providing the number of time iterations t, the concretisation of the `SystemModel` object in the metastable regime is finished if no adaptation is considered.

To include adaptation, the number of adaptation iterations g, the structures for adaptation rules $\mathcal{A}$ and adaptation end $\mathcal{P}$, and loss tolerance l are defined and with that the system further concretised. Please note that $\mathcal{A}$ as well as $\mathcal{U}$ can be explicitly specified with the given structures or directly be implemented with the respective function. In the actual regime (Fig. 1), the `SystemModel` object is run or executed with the specified parameters, that is with concrete number of entities e, milieu topology $\mathcal{M}$, update function ϕ, and number of time iterations t. Once t is achieved, adaptation takes place (Fig. 1).

According to Whitehead, at this stage a nexus has been formed if the end $\mathcal{P}$ is not yet reached or a society if the end $\mathcal{P}$ is reached. Thereby, nexus and also society refer to all the system's states at all time iterations in one adaption iteration, respectively. These system states are stored as member data of the `SystemModel` class. To decide whether or not the end has been reached, first and in any case of the adaption process, that is for every specified adaptation function specified, a loss is calculated. Loss relates to different terms such as error and fitness. It is application specific and needs to be specified as such for the different adaptation functions. In our computational experiment, we are searching for a specific sequence called *end* captured by the structure $\mathcal{P}$ and consisting of zeros and ones. The loss is calculated by the mean square error when comparing each position of the resulting sequence after t iterations with the end. Second and again in any case of the adaptation process, the update function is adapted. Specifically, the rules according to which the update function is working are updated. With that, prehension of entities is adapted since the entities will update according to different rules based on the entities's states in the milieu. Thus, an entity's prehension of its milieu is at every adaptation iteration different. Additionally but not in any case of specific adaptation functions, the adaptation of the update function takes into account an error or learning. In our computational experiment, the adaptation function for elementary cellular automata does not take into account any error or learning, it randomly selects from a given pool of update rules. In contrast, the adaptation function of the artificial neural network, takes into account the error of each neuron and adapts the network weights according to a learning rule. Once adaptation is finished, a new actual system with the adapted update function is created in the metastable system which is then run or executed again in the actual regime until the number of adaptation iterations g or a certain loss is reached.

5 Discussion

The additional value of the introduction of adaptation as adaptation function in our allagmatic method is, as we already highlighted, important because it gives a processual definition of adaptation that better fits to the computational aspect of evolutionary computation. Adaptation is not a concept operating on completed individuals or societies but it evolves with the emergence of nexs and their possible individuation process. This is important because it shows that adaptation is not outside of the system. It is an integrative feature of systems themselves. Adaptation is thus not an obscure and single concept, but a concept part of a whole system, playing a specific role in the creation of societies, i.e. individuals.

By the means of our created system metamodel, the processual emergence of nexs and societies through the different regimes, which as we have shown, are completely compatible with object-orientated programming, can be followed meticulously with the allagmatic method. The method gives guidance for the creation of a complete system with functions, parametrisation etc. This in turn also gives a better understanding of systems, because specific behaviour and emergent structures can be traced back to meta-theoretical concepts implemented in object-orientated code. This again is one of Whitehead's and also Simondon's important reasons to do metaphysics. The abstract concepts such as prehension, society, nexus, structure, operation and so on are supposed to find concrete applications in daily experiences; scientific and non-scientific alike. This is what we have shown in regard to the practice of computer modelling: first these concepts are capable of finding an implementation in object-orientated programming; second, once the abstract concepts have been successfully implemented, the computations are operating within the context of a broad elaborated system metamodel of different regimes giving each element its specific place of behaviour and factual arrangement. Metaphysical concepts are implementable. Thus, we speak of metaphysical computation.

6 Outlook

Our implementation of adaptation only refers to the emergence of nexs or societies, i.e. individuation processes. It does not show how these nexs or societies behave or act in regard to an external environment composed of other nexs and societies. This is not only a further question of implementation but also of interpretation. We have addressed this in section III by highlighting two concepts: survival and control of subordinate societies.

First the question arises if only societies survive within a whole computational universe or if also nexs can survive. This question can be answered with the topic of storage. That is to say, are nexs simply overwritten and not stored and then dying or are they kept alive by storing them? This can be compared in evolutionary computation with the question whether survival is bound to efficiency in the sense that only the best solutions to a problem are surviving or a random mating of societies is capable of reproduction. Due to the observed necessity of individuals to be at the same time flexible and specialised in order to survive, it might be useful to design computer models on the basis of a mixture of nexs and societies as this is already done in some evolutionary computations.

This in turn would mean to create so-called structured societies, i.e. societies composed of nexs and so-called subordinate societies, which are governed and controlled by the overarching structured society. This also means to introduce the concept of hierarchy and inside/outside boundaries: which society is from an operative and structural point of view part of another grander structured society and so on? Such definition of hierarchy, control, structured societies, and subordinate societies could for example also help to better define Stephen Wolfram's four classifications of cellular automata behaviour: stable, periodic, complex, chaotic [15] . It thus allows an analysis of pattern creation, also similar to the ones conducted by Stuart A. Kauffman [17] and Chris G. Langton [18].

References

[1] Kenneth De Jong, David Fogel, and Hans-Paul Schwefel. A history of evolutionary computation. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *The Handbook of Evolutionary Computation*, pages A2.3:1–12. Oxford University Press/IOP Publishing, New York, 1997.

[2] Thomas Bäck, Ulrich Hammel, and Hans-Paul Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[3] Paul C. Jennings, Steen Lysgaard, Jens Strabo Hummelshoj, Tejs Vegge, and Thomas Bligaard. Genetic algorithms for computational materials discovery accelerated by machine learning. *npj Computational Materials*, 5, 2019.

[4] Jun-xuan Fan, Shu-zhong Shen, Douglas H. Erwin, Peter M. Sadler, Norman MacLeod, Qiu-ming Cheng, Xu-dong Hou, Jiao Yang, Xiang-dong Wang, Yue Wang, Hua Zhang, Xu Chen, Guo-xiang Li, Yi-chun Zhang, Yu-kun Shi, Dong-xun Yuan, Qing Chen, Lin-na Zhang, Chao Li, and Ying-ying Zhao. A high-resolution summary of cambrian to early triassic marine invertebrate biodiversity. *Science*, 367(6475):272–277, 2020.

[5] Aubin Thomas, Sylvain Barriere, Lucile Broseus, Julie Brooke, Claudio Lorenzi, Jean-Philippe Villemin, Gregory Beurier, Robert Sabatier, Christelle Reynes, Alban Mancheron, and William Ritchie. Gecko is a genetic algorithm to classify and explore high throughput sequencing data. *Communications Biology*, 2, 2019.

[6] Melanie Mitchell and Charles E. Taylor. Evolutionary computation: An overview. *Annual Review of Ecology and Systematics*, 30(1):593–616, 1999.

[7] John H. Holland. *Complexity: A Very Short Introduction.* Oxford University Press, New York, 2014.

[8] Patrik Christen and Olivier Del Fabbro. Cybernetical concepts for cellular automaton and artificial neural network modelling and implementation. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 4124–4130, 2019. arXiv:2001.02037.

[9] Patrik Christen and Olivier Del Fabbro. Automatic programming of cellular automata and artificial neural networks guided by philosophy. In Rolf Dornberger, editor, *New Trends in Business Information Systems and Technology: Digital Innovation and Digital Business Transformation*, pages 131–146. Springer, Cham, 2020. arXiv:1905.04232.

[10] Patrik Christen. System metamodel formalism. *arXiv:2005.01192*, 2020.

[11] Gilbert Simondon. *L'individuation à la lumière des notions de forme et d'information.* Editions Jérôme Millon, Grenoble, 2013.

[12] Didier Debaise. *Speculative Empiricism: Revisiting Whitehead. Translated by Tomas Joseph Weber.* Edinburgh University Press, Edinburgh, 2017.

[13] Alfred North Whitehead. *Process and Reality: An Essay in Cosmology.* The Free Press, New York, 1978.

[14] Michael Hampe. *Die Wahrnehmung der Organismen: ber die Voraussetzungen einer naturalistischen Theorie der Erfahrung in der Metaphysik Whiteheads.* Vandenhoeck & Ruprecht, Gttingen, 1990.

[15] Stephen Wolfram. *A New Kind of Science.* Wolfram Media, Champaign, 2002.

[16] Stuart J. Russel and Peter Norvig. *Artificial Intelligence: A modern Approach.* Prentice Hall, Upper Saddle River, 2010.

[17] Stuart A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, New York, 1993.

[18] Chris G. Langton. Computation at the edge of chaos - phase-transitions and emergent computation. *Physica D*, 42(1-3):12–37, 1990.