Ubiquity Symposium

Evolutionary Computation and the Processes of Life

Towards Synthesis of Computational Life-Like Processes of Functional and Evolvable Proto-Systems Via Extending Evolutionary Computation

by Darko Roglic

Editor's Introduction

Current evolutionary computation concepts and methods are too narrow and outdated. In this article, Darko Roglic demonstrates that evolutionary computation encounters essential difficulties in projecting these biological concepts by conventional algorithms bounded by the Church-Turing thesis. Such models give an oversimplified description of biological processes. Roglic states super-recursive algorithms can eliminate these limitations and provide adequate models of biological processes.

Editor

Ubiquity Symposium

Evolutionary Computation and the Processes of Life

Towards Synthesis of Computational Life-Like Processes of Functional and Evolvable Proto-Systems Via Extending Evolutionary Computation

by Darko Roglic

Evolutionary computation (EC) is a well-established branch of computer science with a strong engineering focus. However, in order to explore EC in relation to life processes (i.e. reproduction, maintenance and evolvability), which is the principal aim of this symposium, current EC concepts and methods are too narrow and outdated. It is worth to point out the efforts and approaches that extend EC throughout several examples.

Groups of computer scientists and biologists have recently presented the limitations of EC (they used the term “artificial evolution”) and suggested a research agenda toward computational evolution (CE). Actually, they identified specific biological concepts of complexity and robustness that should be incorporated more fully into existing algorithms [1]. Yet, while the first part of their proposal leads us obviously toward the extension of EC, we may encounter some essential difficulties in projecting suggested biological concepts using existing algorithms bounded within currently prevailing notion of computation of Church-Turing thesis (CTT). For example, consider genotype-phenotype relation, which is one of the mechanisms of evolution suggested in the mentioned reference, it is hard to find anything comparable to the “structure of logical determination specific to the formal notion of program” [2]. Analysis of different characteristics of gene expression processes and ultimately the development of phenotype (stability-variability interplay, norm of reaction, also including epigenetic and extragenic contexts within which the genome finds expression) led Longo and Tendero to suggest the programming paradigm cannot capture the relations that should link the genome to the phenotype. Two additional requests for mechanisms of evolution suggested in the CE study are exaptation (i.e.Darwinian preadapation) and enabling variety. However, it seems obvious, that

we cannot finitely prestate Darwinian preadaptations of any particular species, nor can we prestate all relational features of one organism or many that might turn out to be preadaptations, nor can we prestate or list all possible selective environments. This is Kauffman's claim that strongly weakens CTT [3]. Astonishing diversity of biological systems results from the gradual accumulation of spontaneously arisen variations (mutations) sorted out by natural selection. It is difficult to conceive how to enable variety according to CTT, since in the current computational machines there is no place for mutations (nucleotide misinsertions as mistakes in the genetic instruction set that survive actions of repair mechanisms and become possible source of inventions), which is one of the central concepts in biology [4]. The notion of mutation has no meaning for programming theory [2]. In order to fulfill our framework of EC, I want to point out with this discussion the following: It is undoubtedly important to extend oversimplified descriptions of biological processes, but at the same time it is necessary to rethink our conventional approaches and concepts of computation. Under the assumption that biological systems have many properties desired in computational systems, which reflect higher computational power (adaptation, robustness) and which we still don't know how to harness.

In a similar sense, considering the science of computing and proposal of biological computation, Melanie Mitchell has recently argued "a number of fundamental advances must be made." She continued, "not just in our understanding of biology but also in the development of theoretical computer science itself" [5].

Additionally, evolutionary computation of life processes could incorporate within its study possible forces of evolution that extend the conventional Neo-Darwinian paradigm. Comprehensive integrative work in this direction can be found in *Evolution in Four Dimensions: genetic, epigenetic, behavioral, and symbolic variation in the history of life* [6]. This book is aimed at anyone interested in evidence that undermines a strictly gene-based perspective.

We may consider that Darwin's theory starts once we have life (see Figure 1a). But, does Darwinian evolution have roots in prebiotic chemical systems? Finally, extended evolutionary computation could capture chemical information systems and early life processes—metabolism, catalysis, and information—that preceded biological phase. Several authors have developed in vitro evolutionary procedures and revealed replicative chemical systems that emphasize natural selection at the molecular level [7, 8]. Recently, Addy Pross has drawn the outlines for the general (extended) theory of evolution [9]. He derived underlying chemical terms for Darwinian concepts: "kinetic selection" (natural selection), "dynamic kinetic stability" (fitness) and "drive toward greater dynamic kinetic stability" (survival of the fitness).

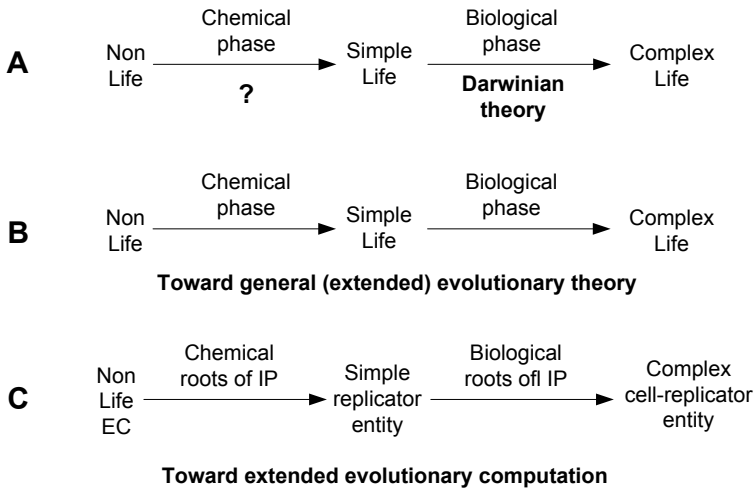

Figure 1: a) Currently, biological evolution (the science of changes) is separated from natural sciences of abiogenesis (or origin of life, that is the study of how the life on Earth could have arisen from inanimate matter) **b)**Recently suggested framework for general theory of evolution is rooted in the Darwinian landscape, but reformulated in physicochemical terms (Pross,2011) Chemical and biological phase could be conceptually merged into one single continuous physicochemical process. **c)**Similar framework could be formulated toward extended evolutionary computation based on the concepts of information and computation (see text)

Competing reactions of two replicating molecules, where one of the replicators is "driven into extinction," is a well-understood chemical kinetic phenomenon and is an example for the kinetic selection rule [10]. Other examples of selection rules include a kinetic advantage of autocatalytic network over the single replicator and cross-catalysis in a self-replicating reaction [11]. These examples demonstrate Darwinian type-behavior (of competition and cooperation, respectively) on the molecular level. However, the essence of Pross's theoretical framework relies on the nature of stability in replicator systems.

The stability of a system can arise for either thermodynamic or kinetic reasons, but both arise from lack of change. Pross's framework emphasizes the simple idea that in nature there is a special stability associated with entities that can self-replicate, this is dynamic kinetic stability —stability that is actually achieved through change, *i.e.* through continual turnover of individual replicative systems as the members of their own population that is stable [9]. Once some relatively simple self-replicating entity would have emerged, whether a single molecule or a minimal molecular network, the drive toward greater dynamic kinetic stability would have induced that minimal replicating system to further complexity.

The simple fact that complex (biological) replicators are kinetically more stable than simpler ones leads Pross to suggest the ongoing process of complexification appears to be not only a biological (evolutionary) property, but a kinetically-driven chemical property. This framework directs abiogenesis and biological evolution into a single continuous physicochemical process (see Figure1b).

Whether we consider chemical or biological roots of information processes, unified framework of extended evolutionary computation could include the life processes expressed in terms of information and information processes (computation) considering simple or complex replicative entities as well as their emergence (see Figure1c).

Super-recursive Algorithms as the Basis for Imperfect Replicative (Evolvable) Systems

Self-replication appears to be one of the fundamental principles without which life could not exist. However, as Woese makes it clear, "Machines are not made of parts that continually turn over, renew" [12]. Now in a similiar vein, as Pross dicussed the cases wherein replicative system could depart from the thermodynamic constraints by enhancing its dynamic kinetic stability, we may consider that certain computational replicative systems could arise if we depart from the completely disciplined behavior involved in traditional computation, and at the same time, the one which does not give rise to pointless repetitive loops. Additionally, by analogy with dynamic kinetic stability (fitness), we may distinguish computation where results appear in relation to *static stability* of the halting state and on-going computation of imperfect replications where selectable results appear correspoding with its *dynamic stability* hence without halting. We may consider, that in the first case, computation is based on recursive algorithms; in another case, computation could be rather based on super-recursive algorithms. Computation based on super-recursive algorithms (SRA) presents a transition from terminating computation to intrinsically emergent computation. They are defined as classes of algorithms that can compute more than Turing machines [13]. Emergent processes become in a certain way infinite but results appear in finite time. Hence, the requisite for halting of computation after achievement of result is no longer required. This is an essential feature of SRA. Additionally, super-recursive algorithms are the basis of computation that changes computational procedures.

As we discussed previously, there is no conceivable way to pre-determine the space of possible evolutions (Darwinian pre-adaptations). Part of this issue could lie in the very notion of biological function, or we may say selectable function in some selective environment. Classical example is the function of a human heart. The heart makes sounds and jiggles fluid in pericardial suck, but the "function" of the heart is to "pump blood"—an organ that pumps

blood has been a selective advantage for our ancestors. So, the central issue according to Kauffman's claim is that there is no finite prestatable set of biological functions. For instance, by taking his example, we cannot come up with finite list of all possible uses of screwdriver for untold purpose since we can use it to drive the screw, or to open a can of paint, or to wedge a door closed, or open, and so on. Importantly, although we do not know ahead of time what relational feature will be useful for certain unknown and new purpose, or function, we use such relational features all the time to find new uses for screwdrivers. Independently, Kugel has considered the same example from the perspective of computer science [14]. He has emphasized the difference between machines and machinery. Turing machines include machinery (originally a tape, a read-write head, and a control unit) limited to computing where we take its first output as a result. Then he introduced the Gold-Putnam machine, as he called it, according to Gold and Putnam's algorithmic models, which are one type of super-recursive algorithms [15, 16]. This machine uses only the machinery of Turing, but in a different way. Instead it performs limiting computation where we take its last output as a result. Kugel suggested computers can apply powerful algorithms, but more crucial is their ability to apply algorithms that were not developed at the time they were built. Actually, it is the ability to acquire them and to adapt them to new situations in ways that are more powerful than ones that can be achieved by computing alone. So, as a computing machine, a screwdriver is limited to driving screws, but as a piece of machinery a screwdriver can be used to drive screws, pry open paint cans, and so on [14]. From this point of view, we may notice in contrast to Woese's remark, mentioned above, machines could be made of parts that continually turn over; at least in relation to changing specification of how machinery may be used.

In comparison with regular computational systems, replicative systems achieve their results through changes rather than through lack of changes. So, the emergent computation of replicative entities based on super-recursive algorithms, once “freed” of halting state, could possibly lead to express teleonomic character; a character that is totally unrelated to teleological directed programming and computation. Accordingly SRA theory could provide suitable formalism to study replicative systems, whether we consider their chemical or biological roots. An evolvable computer version of this idea might be worth exploring.

The Open-Ended Synthesis and Complex Systems Engineering

Extending evolutionary computation forces us to rethink our conventional engineering design and synthesis in order to build replicative systems. As Russ Abbott argues: “To date we don’t know how to build systems that persist on their own” [17]. Scientific endeavor is now oriented to explore and adopt processes traditionally confined to the natural world and complex

systems, principal among them evolutionary and developmental processes, as well as the concepts that address self-organization [1, 17].

However, evolution could be viewed as the process of open-ended synthesis and biological "machine" with its complex program—which comprises the mechanisms of developmental processes (cell growth, differentiation, morphogenesis) and governs its maintenance, function, and behavior— is the product of such a process. Fundamentally, it addresses how modifications of development and developmental processes lead to the production of novel functions and organisms, or we may say briefly, how genetic changes have given rise to difference. An extended approach toward complex systems engineering (e.g. Banzhaf and Pillay [1]) has to take into consideration the engineering programming paradigm as one deeply related with the problem of synthesis for which we don't have any formal model of how it can be done automatically. Automatic programming uses computers to turn finite programs a programmer has developed into programs a computing machine can run. In contrast, extending evolutionary computation could be interested in looking for procedures that would generate programs. Implementing dynamical entity, called here evolvable program (by which we could simulate replicative systems), that generates finite programs from infinite inputs could be one way to do that.

To this end it would seem our first step is to build dynamical replicative entities. Without going into details concerning the structure of an evolvable program, suffice it to say, it is based on the sequences of program elements that carry operational and replicative sets of instructions. These instructions ensure the particular function of an individual system in its environment as well as its reproduction. Running the template-based replication, program elements are gathered from the environment of the system in order to synthesis new evolvable programs with ancestral characteristics. However, most of our current evolutionary simulations tend to deplete their resources for replication and development hence the system fails to produce a sustained evolution. For instance, resources in Tierra are not developed, only consumed. Hence, to ensure propagation of replication and development in self-sustained way enviromental resources of evolvable program elements are needed *ab initio*.

More basically, open-ended evolution lies on the capability of a replicative system to incorporate viable variations during the process of replication synthesis. In relation with evolvable program, we may consider that variation (as a non-harmful and possibly useful mutation) could appear at the operational part of the sequence. Importantly, such variations do not tend to represent the variants of old functionalities, but rather they could lead to new ones. So, these variations (mutations) of evolvable program sequences impose selectable (functional) differentials on the populations of evolvable program entities.

Further, the current simulations of an evolutionary process essentially terminates when the best result (fitness is explicit so far) is found. But, it is not sufficient for combinatorial operations for the new structures to be open-ended, if the system is closed to emerging opportunities and achievable goals. Natural selection produces not only adaptations but also new opportunities, and since the selection process continues; the cumulative adaptations (adaptations built on a set of previously fixed adaptations) can provide a new "survival" pathway. Hitherto, simulated adaptation produces results but stops afterward through lack of a new input from the selection. Further evolution is impossible, unless the system of evolvable programs is used interactively. At this point, we may consider the paradigm of selective (inductive) programming, which allows us to introduce new selection criteria during the ongoing computational processes of evolvable programs.

According to modern evolutionary synthesis, a successful variant must exist before it can be favored by the environment. Only then do the beneficial mutations favored by natural selection eventually become fixed as a newly incorporated string of DNA sequence. The resistance of bacteria to antibiotics is a consequence of this process. We may expect evolvable program entities could reach the results in an equivalent way, considering such adaptations that involve one or few genes and their variations. In order to explore this we could consider a more complex model called evolvable cell-program (ECP).

Choosing the model of a cell as a unit of function and organization allows us to incorporate molecular intracellular information processes and, at the same time, a network of interacting cells. In relation to unicellular organisms (e.g bacteria) ECP is also an organism model. ECP includes epuome as a set of evolutionary processable units (epu). Roughly, each single epu includes operational instructions that encode for proteator-execution program and stays as an analogical equivalent to a single gene as a hereditary unit. The term proteator (abbreviation constructed from *prot*ein-level-oper*ator*) alludes to protein (peptide) sequences and their working role that arises after the process of protein folding. Without going into detailed descriptions of ECP functioning a simplified exposition is shown in Table 1. Let us only shortly emphasize the computational processes in relation to development and evolutionary adaptation by analogy to bacteria achievements.

Table1.: Simplified overview of an Evolvable Cell-Program structure with corresponding processes

Step	Structure	Processes in relation to development and evolution		Results in relation to:	
		Type	Direction	Synthesis and Development	Evolutionary adaptation
1.	epuome	Replication	*vertical*	epuome	mutation
2.	epuome (with mut.)	Epu-expression	*horizontal*	proteators	new proteator
3.	proteators	Formation of macroproteator structures (moduls) and their interactions	*horizontal*	cell-program (ECP)	Novel feature of ECP
4.1.	ECP	4.1.1.Operational cycles	*horizontal*	Operational result	(selection Level)
	ECP (possibly with advantageous feature)	4.1.2.Reproduction and growth (via binary fission and steps 1-4)	*vertical*	New genartion of ECP pop. with certain diversity	
4.2.	ECP pop. (e.g. colony)	4.2.1.Intercellular interactions	*horizontal*	Collectively achieved results	
		4.2.2.Selection	*transition*	Implicitly selected ECP	
	Selected ECP individuals	4.1.2.Reproduction and growth (via binary fission and steps 1-4)	*vertical*	ECP population which is no longer susceptible to a particular selective pressure	

ECP capable of imperfect replication could insert a mutation as a novel string within a epuome sequence during the process of replication (Step 1), which could lead to a synthesis of a new proteator as a new algorithmic procedure. After inducing a new selective criterion by selective programming (Step 4.2.2.) (by analogy with selective pressure, e.g. antibiotic), pre-existed epu-variation could possibly express novel proteator (Step 2) (e.g certain enzyme in bacteria) ,which gives advantage to a particular ECP entity (or few ECPs that have such variation). By definition, there is at least one such ECP entity in the population that already possess corresponding advantage. By conventionally running the ECP (i.e. operational modul of proteators of ECP), specific signal/data processing will reach the result by halting or coming in the final state (Step 4.1.1.). However, reaching the result reflects a certain trait or a functional capability of such ECP that becomes the subject of selection. Contrary to conventional halting-restriction of computation, ECP proceeds with its computation vertically, by reproduction. Hence, the fitness of individual ECP is implicit (i.e. how successful it is at reproducing) rather than explicit (currently, fitness is mostly based on numerical values of an individual entity as a solution to a given problem). Note that vertical evolutionary processes governed by selective programming do not directly operate on data (operational results), but on the structures that carry instructions. As it follows, ECP individuals reproduced (Step 4.1.2.) and, thus, the mutations providing previously non-existent variation undergo fixations by spreading through population over the generations. At the same time, new mutations occur (Step 1), contributing a new genetic variation to an existing genetic variation governed by non-terminating evolutionary processes.

References

[1] Banzhaf, W., and Pillay, N. Why Complex Systems Engineering Needs Biological Development. *Complexity* 13 (2007), 12-21.

[2] Longo, G., and Tendero, P.E. The Differential Method and the Causal Incompleteness of Programming Theory in Molecular Biology. *Foundations of Science* 12, 4 (2007), 337-366.

[3] Kauffman, S. Reinventing the Sacred: A new view of science, reason, and religion. *Basic Books*, 2008.

[4] Roglic, D. Super-recursive Features of Evolutionary Information Processing And the Models of Evolutionary Computers. In *Information and Computation*, Dr. G. Dodig-Crnkovic and Dr. Mark Burgin, eds. World Scientific Series in Information Studies, 2011.

[5] Mitchell, M. Biological Computation. *Ubiquity* (Feb. 2011).

[6] Jablonka, E., and Lamb, M.J. *Evolution in Four Dimensions: Genetic, Epigenetic, Behavioral, and Symbolic Variation in the History of Life.* MIT Press, 2005.

[7] Bartel, D.P., Szostak, J.W. Isolation of New Ribozymes from a Large Pool of Random Sequences. *Science* 261, 5127 (1993), 1411-1418.

[8] Wright, M.C., Joyce, G.F. Continuous In Vitro Evolution of Catalytic Function. *Science* 276, 5312 (1997), 614-617.

[9] Pross, A. Toward a General Theory of Evolution: Extending Darwinian theory to inanimate matter. *Journal of Systems Chemistry* 2, 1 (2011).

[10] Lifson, S. On the crucial stages in the origin of animate matter. *J Mol Evol* 44, 1 (1997), 1-8.

[11] Lincoln, T.A., and Joyce, G.F. Self-sustained replication of an RNA enzyme. *Science* 323, 5918 (2009), 1229-1232.

[12] Woese, C.R. A new biology for a new century. *Microbiol Mol Biol Rev* 68, 2 (2004), 173-186.

[13] Burgin, M. *Super-recursive Algorithms*. Springer, (2005).

[14] Kugel, P Computers Can't Be Intelligent (...Turing Said So). *Minds and Machines* 12, 4 (2002).

[15] Gold, E.M. Limiting Recursion. *Journal of Symbolic Logic* 30, 1 (1965), 28-48.

[16] Putnam, H. Trial And Error Predicates And The Solution Of A Problem Of Mostowski. *Journal of Symbolic Logic* 30, 1 (1965), 49-57.

[17] Abbott, R. Putting Complex Systems to Work. *Complexity* 13, 2 (2007).

About the Author

Darko Roglic's interests, in relation to evolutionary computation, include modeling and analysis of molecular infomation systems at distinct levels of biological organization. His long-term aim is to model a minimal self-replicative single-cellular system (as an interactive agent system) that will allow us to incorporate and explore three separate issues: how the genome-equivalent structure expresses information through protein-level functions integrated in the cell, how the cell structures interact in order to produce behavior, and an issue of evolvability as genome-equivalent structure contolled capacity to respond to directional selection.

DOI: 10.1145/2555235.2555236

