OPAALS

Contract N^{o} IST-034824

WP1:
Cell Biology, Autopoiesis and Biological Design Patterns

D1.5:
Integration of gene expression computing and operational closure in the software ecosystem architecture

Project funded by the European Community under the "Information Society Technology" Programme.

Contract Number: IST-034824
Project Acronym: OPAALS

Deliverable No: D1.5
Due date: 31/05/2010
Delivery date: 10/10/2010

Short Description:
A key challenge in modern computing is to develop systems that address complex, dynamic problems in a scalable and efficient way, because of the increasing complexity of software. Biological systems are thought to possess robust, scalable processing paradigms, which are addressed by autopoiesis. We therefore pursue two approaches, **biological computing** (biologically-based computing) and **natural computing** (nature-inspired computing), to the integration of autopoietic properties and behaviour in computing paradigms.

Authors: Gerard Briscoe and Paolo Dini (LSE); Chrystopher Nehaniv, Nicolas Oros, Ayoola Yinusa, and Moritz Buck (UH)
Partners contributed:
Made available to: Public distribution

Version	Date	Author, organisation
1	31/06/10	Gerard Briscoe (LSE)
2	23/09/10	Chrystopher Nehaniv - Sec. 4 (UH)

Quality Check:
Internal Reviewer: Alastair Munro (UNIVDUN)

Dependencies

Achievements*	Done: • *"Continue developing the top-down ecosystem-inspired approach."* • *"Bottom-up gene expression-inspired approach. Define theoretical computing models and conceptual architectures for gene expression-inspired computing."* • *"Examine if autopoiesis is a property that can be attributed to architecture."* • *"Advance the Community Cloud Computing interpretation of Digital Ecosystems."* Not done: • *"Is there an architecture style (design pattern) that gives rise to autopoiesis, which may be achievable in the remaining time in the project."* Somewhat expectedly, fully achieving this within the remaining time of the project proved too ambitious. We have nonetheless established the *ground work* with our autopoietic inspired extension of autonomic computing.
Workpackages	WP1
Partners	LSE, UH
Domains	Natural Computing, Autopoietic Computing, Autonomic Computing
Targets	WP1, WP3, natural computing

Publications*	**The Computing of Digital Ecosystems**. G Briscoe and P De Wilde. Accepted to *International Journal of Organizational and Collective Intelligence: Special Issue on Collectively Intelligent Information and Knowledge Management.* **Towards Autopoietic Computing**. G Briscoe and P Dini. In *Springer Digital Ecosystems Conference*, 2010. **Data Centres vs Community Clouds**. G Briscoe and R Chitchyan. In *Object Oriented Programming, Systems, Languages and Applications Conference: Software Research and Climate Change Workshop*, 2009. **Community Cloud Computing**. A Marinos and G Briscoe. In *Springer Cloud Computing Conference*, 2009. **Digital Ecosystems in The Clouds: Towards Community Cloud Computing**. G Briscoe and A Marinos. In *IEEE Digital Ecosystems and Technologies Conference*, 2009.
PhD Students*	M BUCK, N OROS
Outstanding features*	Hardware considerations for our *biological computing* based form of *autopoietic computing.* Form of *autopoietic computing* from a *natural computing* based extension of *autonomic computing.*
Disciplinary domains of authors*	Briscoe: computer science, natural computing Dini: applied mathematics, physics, computer science Nehaniv: mathematics, computer science, complex adaptive systems Oros: computer science Yinusa: computer science Buck: computer science

* Indicates information requested by reviewers

Contents

Executive Summary

Biological systems are thought to possess robust, scalable processing paradigms that can automatically manage complex, dynamic problem spaces, possessing several properties that may be useful in computer systems. These biological properties are addressed by autopoiesis, a descriptive theory of the cell founded on the concept of a system's circular organisation. We therefore pursue two approaches, biological computing (biologically-based computing) and natural computing (nature-inspired computing), to the integration of autopoietic properties and behaviour in computing paradigms. First, we continue our investigations into cellular and intra-cellular systems to define an autopoietic form of computing, focusing on what kind of new **hardware paradigms** will be necessary. Next, we pursue an autopoietically-inspired form of computing based upon extending the concept of **autonomic computing**.

For the **biological computing** approach we have further considered formalisations of autopoiesis, implementability, computability and potential implications, including consideration of the potential hardware requirements for our autopoietic form of computing. As integration with our Software Ecosystem at this stage is not feasible, we instead considered its integration with business models that adopt cellular organisation. Specifically, synergy between cellularly organised businesses and autopoietic software. Our **natural computing** approach presents a vision of autopoietic computing. Given the differences between the autonomic nervous system and the autopoietic immune system, we extended the self-CHOP (Configuring, Healing, Optimising, Protecting) properties of autonomic computing to include the self-R (Replicating) property, creating CHOPR for our (natural computing-based) autopoietic computing.

1 Introduction

A key challenge in modern computing is to develop systems that address complex, dynamic problems in a scalable and efficient way, because the increasing complexity of software makes designing and maintaining efficient and flexible systems increasingly difficult. Biological systems are thought to possess robust, scalable processing paradigms that can automatically manage complex, dynamic problem spaces, possessing several properties that may be useful in computer systems. The biological properties of self-organisation, **self-replication**, self-management, and scalability are addressed by **autopoiesis**, a descriptive theory of the cell founded on the concept of a system's circular organisation to define its boundary with its environment.

Autopoiesis as a source of biological inspiration for new models of computing sits alongside gene expression and symbiosis, as shown in Figure 1 (reproduced from D1.3 [26]), we consider all three expressions of the more fundamental concept of Interaction Computing. Whereas symbiotic computing is relevant to e.g. the concept of symbiotic security [131] and gene expression computing is discussed in the companion deliverable D1.4 [27], this report is primarily concerned with autopoietic computing.

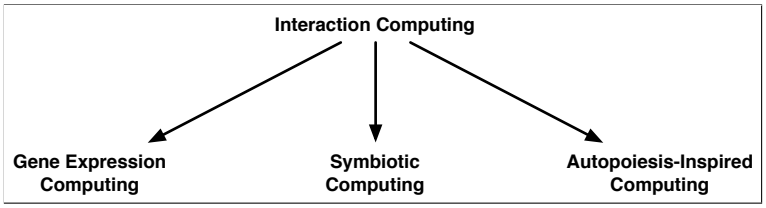

Figure 1: *Interaction computing and its derivatives.*

So, in this report we pursue two approaches to autopoietic computing, biological computing (biologically-based computing) and natural computing (nature-inspired computing), for the integration of autopoietic properties and behaviour in computing paradigms. **Section 2**, biological computing, continues our investigations to sufficiently understand the fundamental processes occurring in cellular and intra-cellular systems to define an autopoietic form of computing. Here we consider what kind of new hardware paradigms, beyond von Neumann, will be necessary to achieve this kind of computation. In **Section 3**, natural computing, we pursue an autopoietic inspired form of computing based upon extending the concept of *autonomic computing* to include certain autopoietic behaviour, specifically that of autopoietic replication. Autonomic computing started with a seminal paper that defined a vision for it, and so we similarly propose a vision for this form of autopoietic computing.

Understanding how to harness self-replicating entities in autopoeitic organisations requires advances in relating self-replication to organisational level emergence of functionality. Foundational investigations of self-replication carried out therefore focus on the achievement of and roles of three key phenomena: inheritable variation, sex (i.e. transfer of inheritable information) and its impact on the evolution in self-replicators, and the trade-off between complexity and pay-off in the emergence of higher levels of organisation requiring cooperation. These areas are treated in **Section 4**.

In **Section 5**, we conclude by discussing the potential for integration with our *Software Ecosystems*, including Digital Business Ecosystems, which are complementary Software and Business Ecosystems.

2 Biological Computation - Autopoietic (form of) Computing

Biologically-oriented computation is even more aspirational than quantum computing, which merely wishes to define and create a computing paradigm upon our current understand of quantum physics, specifically quantum mechanics. Instead, biological computation seeks to define a form of computing upon the operation of biological systems, which is many magnitudes of **complexity** greater, and for which our current understanding (despite considerable progress) is much more infantile. Furthermore, we seek not only to be inspired based on a relatively superficial understanding, an approach we will consider in the next section, but to have sufficient understanding to essentially copy, at a sufficient level of details, the fundamental processes occurring. However, the reverse or compliment of the problem, computing paradigms to apply in understanding biological systems (bioinformatics), is the key **progenitor** of biological computation. The seemingly ever increasing limitations of current computing paradigms in simulating and modelling biological systems has spurred the need for biological computation, which would serve not only to achieve its own goals, but improve the applicability of bioinformatics. So, it makes sense to at least consider initially the origins of this duality, and the more relevant aspect of bioinformatics, before returning to our focus to biological computation.

Since the early 1950s discoveries of the structure of complex biomolecules [106, 103, 104, 105, 152] which defined the cornerstones of **molecular genetics**, there has been a long line of research towards decoding the information content of macromolecules. During the past 50 years computers have been intensively used to model and analyse complex organic structures. The age of **bioinformatics** began with the collection and analysis of large sets of data. Every new advancement revealing the contiguity of the underlying biological substances suggested new hypotheses, theories and methods for experimentation with them while paying tribute to their growing complexity.

The discovery of the double helix of DNA [152] which ascertained the primary structure of the complex biomolecules underpinning information encoding and inheritance, in terms of a specific bonding among pairs of bases (adenine with thymine and guanine with cytosine) was followed by more elaborated research on the formation of **macromolecules** such as the genetic code relating information encoded in DNA to the structure of proteins [21], and predicting the secondary RNA structure [25, 110, 160, 150, 43]. These studies on the molecular basis of heredity were expanded towards exposing the biochemical and biophysical details of the tertiary and the **quaternary** structure of nucleic acids and proteins [47, 58, 138, 57, 48, 156]. At the same time, complex biological structures such as those of nucleic acids have inspired research in biomolecular and biological computation. In 1987 Tom Head introduced a new method of relating formal language theory to studying the information content of macromolecules [45]. He associated a language with each pair of sets such that the first set consists of double stranded DNA molecules and the second one corresponds to recombinatorial behaviours allowed by specific classes of enzymatic activities.

In 1994 Leonard Adleman used a set of simple DNA operations to solve an NP-complete problem as a node instance of the Directed Hamiltonian Path [2]. The performance of this device significantly **outperformed** the traditional silicon of the time for this specialised problem. This work also set out the beginning of a new era of scientific exploration at the edge between biology and mathematics, known as **DNA computation** which became an instance of biocomputation [52].

Furthermore, relations to other scientific fields such as control and chaos theory were investigated [73]. During this period, experimental studies of the mechanisms for the manipulation of

biomolecules, such as kinetic partitioning [139], were also pushed forward. **Biomolecular computation** ventured into the domains of game theory, neuroscience and design automation to investigate utilization aspects such as RNA solutions to chess problems [31], experimental DNA neural computation [87] and bottom-up circuit patterning based on DNA self-assembly [29]. Other prominent developments were autonomous DNA nano mechanical computation and motion devices [158], as well as autonomous DNA cellular automata [157]. Inevitably, qualities such as **powerful parallelisation**, small size with an enormous storage capacity and efficient energy utilisation are now driving this research towards evolutionary biological, biomolecular and organic computing [127, 30, 46, 135]. However, it must be recognised that true encoding, storage (inheritance) and biocomputation involves not only DNA nucleotides, but also protein sequences, non-protein coding RNAs, mitochondria, microtubules, lipid membranes, nuclei, cells,endoplasm, organs and the entire organism [94]. We must also consider epigenetic phenomena, i.e. the same DNA code can deliver different products at different times and under different conditions. The whole process is tuned by genetic buffering the restraints of present and past interactions at each level with the surrounding environment [147]. In other words, *the program is the system itself* [95]. While this was not the only path of progression [133, 82], and this is by no means an exhaustive account [133], it does exemplify one of the paths that lead to questions about the the fundamental autopoietic nature of the computation of biological systems. However, not only for the need to understand it for the purposes of modelling and simulation in bioinformatics, but also the need to understand and explore it as a computing (information processing) **paradigm** of biological systems in its own right, and its potential as a generic computing paradigm for human-made computing. Unfortunately, autopoiesis is still a *theory-at-work* which is very general and undifferentiated both in terms of mathematical formalisation and technical implementation [133].

2.1 Formalisations

Here we consider in greater detail the lack of a formalisation for autopoiesis, leading us to consider more deeply the implications. A category theoretical framework for formalisation of living systems was presented by Rosen after his studied of them over three decades [118, 119]. He concluded that living systems, which are essentially metabolism/repair (M, R)-systems [120, 121, 122, 123], are not realisable in computational universes. If correct, this conclusion would imply that a form of artificial life, akin to real life, cannot exist at all, or at least not in any computational spaces as they are known now. Letelier et al. [68] analysed (M, R)-systems from the viewpoint of autopoietic theory, and provided an algebraic example on defining **metabolic closure** while suggesting a relationship to autopoiesis.

In a series of works [96, 97, 98], reflecting the contributions of Rosen [123, 124], Luhman [59] and Kawamoto [53, 54], Nomura reviewed the formal roots of autopoiesis in the light of **category theory** [88] and proposed a mathematical model of autopoiesis based on Rosen's definition [96]. After having examined some of the central postulates of the autopoietic theory, Nomura concludes that previous research has not delivered an **unambiguous** description of the phenomenon and proposes a more general and strict formal definition.

Another category theoretical argument and revision of Rosen's theorem was provided by Chu and Ho [19]. These authors review the essence of Rosen's ideas leading up to his rejection of the possibility of simulating a form of artificial life, akin to real life, in computing systems. They argue that the class of machines Rosen distinguished from closed systems is not equivalent to a realistic formalisation of artificial life and concludes that Rosen's central proof, stating that living systems are not mechanisms, is wrong. However, Rosen himself warned *there is no*

property of an organism that cannot at the same time be manifested by inanimate systems [124]. As a result, Rosen's claim remains an **open issue**.

The realisation that some of Rosen's central notions were most likely ill defined provides some interesting theoretical concerns which warrant further investigation. Including, that from the viewpoint of contemporary logic, one cannot take for granted that organisms are merely mechanisms that they can be constructed in ways used to build machines. Indeed, his model does not fail with regards to particular aspects of natural systems which have **mechanical representations**, but instead failed to represent whole developing systems [133]. It is this last issue which is central for a comprehensive model of living systems, and serves as a reminder of the impracticality of reductionist approaches for autopoeisis, at least until the **phenomenon** is more significantly if not completely understood.

More recently Nomura [99] analyses the possibilities of an algebraic description of living systems and clarifies the differences between the aspects of **closedness** required in (M, R)-systems and autopoiesis. He discovers two essential differences between autopoietic and (M, R)-systems. The first one is the difference in their forms of closedness under entailment of the components and categories required for the description of closedness. The second one is the distinction between organisation and structure. Nomura points out that the first difference depends on the assumption that completely closed systems, modelled as an isomorphism from the space of operands to the space of operators, are necessary conditions of autopoiesis. However, this requirement has not yet been proved in a mathematically strict way. Furthermore, the definition of autopoiesis itself deserves a special attention. There were differences in the **interpretation** of autopoiesis between Maturana and Varela. When Varela collaborated with McMullin on computational models of autopoiesis, the original algorithm was revised within the same year [80, 83]. Varela together with Joseph Goguen have considered some of the implications of autopoietic theory on the axiomatisation via discrete mathematics for modelling biological processes in relations to Brownian algebra [143], but no one appears to have yet pursued this direction toward computational implementations.

In summary, the following two **conclusions** can be made: (i) a more precise and formal definition of autopoiesis and its relationship to (M, R)-systems, as outlined by [99], is required (as we proposed previously); (ii) novel mathematical techniques and tools which adequately describe and simulate biological processes [112, 114, 113] are required. In other words, in order to make progress in this area, it is necessary to either provide further means for formalisation, or invent new formal approaches that best suit the original definition, or to redefine (refine, extrapolate) the theory itself, such as the tangible approach provided in [11]. Rosen tried to distinguish life systems from machines with his original definition. Although his proof was incomplete, Chu and Ho confirmed the importance of Rosen's idea. Nomura also agrees with them on this point [99]. Therefore, in order to provide the **engineering base** for artificial organisms and systems in physical space, it is advisable to further investigate the necessary conditions for modelling characteristics of living systems and provide more stringent definitions of life systems and machines based on Rosen's attempt.

2.2 Implementability

Implementation is naturally hampered by a lack of suitable and viable formalisations. However, there are questions inherently raised from the issues consistently applicable to the majority of existing formalisation attempts. For example, is autopoietic computing another name for **evolvable hardware**? During the last decade the concept of evolvable hardware has gained

an ever increasing popularity, exemplified by the most intriguing eagerness of organisations such as NASA. Essentially, evolvable hardware uses the techniques of evolution (mutations and recombination) within the framework of genetic algorithms to design, control, and configure electronic hardware. But the basic idea of evolvable hardware lacks the essential component of structural coupling. In effect, in each *generation* all the versions of a given hardware *lineage* are generally screened according to the criterion of an outside observer or predefined fitness function, that selects the best versions and uses them as *seeds* to the next generation. Thus evolvable hardware devices are unable to construct a congruent relationship with their environment, as they are continuously tested by an agent (the experimenter who performs the selection) that is outside the circuit or outside the medium. Therefore, autopoietic computing has a crucial difference with respect to evolvable hardware, which is that no outside observer is needed, but the hardware must have true **plasticity**, and not only simulate plasticity as evolvable hardware does [67]. However, the process of evolution might yet in essential in realising autopoietic computing, as we shall discuss.

Stalbaum makes the point that it is necessary to differentiate between the **implementation** of computational autopoiesis as proof of concept (computational autopoiesis is possible), and the potential practical applications of such a system [136]. He supposes that Varela's early work [83] makes a strong case for the former, but that **designing** autopoietic automata implementing computing applications such as a self-managing database system is a substantially different problem.

Stalbaum further states that the challenge in finding **engineering congruency** between autopoietic systems and problems that yield solutions is enormous. Indeed, the demonstration that autopoiesis can be used for computational and communication processes using a minimal implementation, e.g. of an artificial chemistry model, is relatively simple. However, such possible uses of computational autopoiesis do not necessarily imply that autopoiesis can be **effectively implemented** to perform work. This is because the internal purpose of an autopoietic system is restricted to the ongoing maintenance of its own organisation. So, this goal becomes a problem for anyone intending to use autopoiesis for the purpose of computation as it is currently understood, i.e. to deliver an output result from a given input within a limited number of steps.

Furthermore, computation may be limited to being a by-product of ongoing **structural coupling** between a collection of autopoietic elements and their environment, but which cannot be deterministically defined as a purposeful task for the solution of a specific problem or class of problems as is expected from current computational and engineered systems. However, goal alignment with lineage **self-preservation** is the essence of evolutionary computing, and so it may be possible to evolve an autopoietic computing artefact, in which process of congruency (adaption to relatively small changes) is complemented by the process of evolution (adaption to relatively large changes).

Additionally, decision making is not the fundamental intent of biological computing. In other words, the natural drift inherent in living systems cannot be counted on to directly solve problems that do not primarily serve for conservation, adaptation and maintenance of those systems. In addition, the multiple orders of structural coupling in autopoiesis define an even more complex picture of the interacting units. In this respect, autopoietic computing has similarities to quantum computing [32, 33]. Quantum computing is computation that makes direct use of **quantum mechanical phenomena**, such as superposition[1] and entanglement[2],

[1] The potential simultaneous occurrence of all the possible states of a system (for example, the possible positions of a subatomic particle).

[2] A property of a quantum mechanical state of a system containing two or more objects, with non-classical correlations between observable physical properties of remote systems.

to perform operations on data, rather than the transistor of traditional computers. The basic principle behind quantum computation is that quantum properties can be used to represent data and perform operations on these data [40]. Although quantum computing is in its infancy, experiments have been carried out in which quantum computational operations were executed on a very small number of qubits (quantum bits). Both practical and theoretical research continues, and many national government and military funding agencies support quantum computing research to develop quantum computers for both civilian and national security purposes, such as **cryptanalysis** [50]. If large-scale quantum computers can be built, they will be able to solve certain problems much faster than any current classical computers (for example Shor's algorithm[3]). Quantum computers do not allow the computations of functions that are not theoretically computable by classical computers, i.e. they do not alter the Church-Turing thesis. The gain is only in **efficiency**, with its theoretical model known as the quantum Turing machine, also known as the universal quantum computer. However, if the non-Turing computability of autopoietic computing proves to be possible, then at the very least a different form of quantum computing would be required to support autopoietic computing paradigms.

2.3 Computability

Although being now a special case in modern physics, Newton's world picture still dominates computer science [14], engineering [66] and biology [15]. Turing Machines are **Newtonian** in the broader sense that they deal exclusively with syntax and inference rules based on discrete logic in absolute space and time to deliver predictable behaviour. Yet, the review of the diverse non-classical computation approaches beyond the Turing frame leads to the conclusion that these models hypothesise different post-Newtonian laws of physics as a precondition for their implementation.

Although modern computer science has not really entered the relativistic sub-nuclear age of modern physics yet, the abundance of computational ideas and approaches indicates that researchers are well aware of the limitations of the **Turing** computation model. They are certainly going to use every discovery and invention in physics to realise their concepts. Perhaps the most significant aspect of this finding from a historical perspective is the comeback of analog computing now based not on mechanical components and electronic circuits, but on artificial neural networks and **continuous quantum computation**. This is a very interesting fact which shows that computation is now closer to biology than to classical physics. Indeed, computer science and biology maintain today a similar relationship to that of 19th and 20th century mathematics and physics. Progress in one field will influence progress in the other and vice versa, for it is not possible to avoid analogy and semantics (in terms of the formal logic known today).

The development of new integral computational **paradigms** reflect the emergence and organisation of information for living systems in a more adequate way than traditional formal approaches based on binary logic and the Church-Turing thesis, [20, 140]. This represents an idealised, but not necessarily unique, model for computation, where representations are formal, finite and definite [72]. Exposing these characteristics itself, the Church-Turing theory of discrete states excludes by definition alternative approaches to computation. Even worse, as it has been successful for the past 70 years, the Church-Turing model evokes the conviction in the majority of contemporaries that it is universal, as demonstrated in [132], and that there is no other option for computation at all. Thus, **binary logic** remains a good working, and

[3] A quantum algorithm for integer factorisation (Given an integer N, find its prime factors).

of course, economic model for everyday discourse, just as Newtonian celestial mechanics was useful for global navigation until the arrival of Einstein's theory of relativity (1905-1915) which made interstellar journeys conceivable. However, Newtons mechanics is perfectly adequate for everyday tasks, as is the Church-Turing thesis is for everyday computing. While the universality of Church-Turing cannot justify excluding all others, going beyond we may construct something that initially has minimal everyday utility (e.g. Einsteinian physics, quantum computing). Yet, the quest for alternative approaches to computation continues.

MacLennan's investigation into these **alternative models** of information processing led to similar conclusions for computation as those of Feynman for quantum systems [32, 33] and Rosen for artificial biology [124, 125]. MacLennan argues that conventional digital computers are inadequate for realising the full potential of biological computation, and therefore that alternative and more brain-like technologies should be developed. He views the Turing Machine model as unsuited to address the class of questions defined by biological computing [71]. Furthermore, MacLennan argues that biological computation requires continuous and **non-Turing** models of computation [69]. Although, these processes cannot be regarded as *computation* in terms of the Turing machine definition [141].

However, Turing's thesis is not the ultimate verdict about computing, and we need only to refer to Euclid's geometry in a historical context. There are numerous examples about how new generalisations in mathematics and physics emerged out of apparent axioms and postulates. By shifting the perspective, the latter turned out to be **conceptual deadlocks** as they were not able to deal with new ideas or factual observations. So, when the new theories were finally reconciled with the established world order, the apparent *paradoxes* turned out to be new, more general facts and the old frame of thought became a special case within the new one [124].

It is a well known fact that science requires sometimes a few iterations of denial and rediscovery over the centuries until a new idea or paradigm is accepted by the dominating majority [155, 111, 60, 126]. The situation with the **dominant** computing paradigm today, the Turing machine model, is similar. This concept leads to a discrepancy when applying computer science methods in systems biology. Indeed there is nothing wrong with the Turing machine model, as long as it is exploited within its frame of relevance which deals with questions about formal **derivability** and the limits of effective calculability [70]. So, when computing moves to the domain of biology, it should be realised that it does not have the mathematics to describe living systems [99, 91].

So, a major paradox with adopting the Turing machine model for biological computing was pointed out in [72]. Correspondingly, formal logic considers a function computable if for any input data the corresponding output would be produced after finitely many steps, i.e. a proof can be of any finite length. For the sake of completeness and consistency, the Turing Machine model imposes no bounds on the length of the individual steps and on the size of formulas, so long as they are finite. The concept of **time** assumed in the Turing machine model is not discrete time in the familiar sense that each time interval has the same duration. Therefore, it is more accurate to call Turing machine time a sequential time [70]. Since the individual steps are of **indefinite** duration, there is no use to count the number of steps to translate that count into real time. Temporal formalisms based on the Turing machine model such as TLA[4] [61, 62] do not change this situation. Consequently, the only reasonable way to compare the time required by computational processes is in terms of their asymptotic behaviour. Therefore, once the observers have chosen to ignore the speed of the individual steps, all they have as complexity measure is the size of the formulas produced during the computation or the growth degree of the number of steps with the size of the input.

[4]Temporal Logic of Actions: a combined form of logic used to describe the behaviour of concurrent systems.

Turing **sequential time** is reasonable in a model of formal derivability or effective calculability, since the time duration required for individual operations was irrelevant to the research questions of formal mathematics. However, this perspective leads to the result that any polynomial-time algorithm is *tractable*[5] and *fast* (e.g. matrix multiplication) and that exponential-time algorithms are *intractable* (e.g. Travelling Salesman), whereas problems which are polynomial-time reducible to each other are virtually identical. This duration independent situation is indeed peculiar for biological systems. MacLennan ironically describes it as one where *an algorithm that takes N^{100} years is fast, but one that takes 2^N nanoseconds is intractable.* However, one cannot afford to ignore the duration of the steps in biological computing, where instant response has the value of **survival** for a living organism. Similarly, Rhodes [117](1969) argues that Turing computation is *off-line* considers models that reflect *on-line* real-time, interaction such as interacting sequential machines. This is in line with viewpoint of Wegner that a modern notion of computation must incorporate **interactivity**, going beyond the Church-Turing thesis [153].

Additionally, adaptation and reaction to unpredicted stimuli, large scale cluster optimisations and continuity of input and output values in space and time, such as those occurring in neural networks, immune systems and insect swarms are other characteristics of biological systems that lie outside the scope of the Turing machine model and cannot be addressed appropriately with **traditional** analytical approaches. Finally, Turing models do not at all address **evolvability**, that capacity to evolve appropriate, fitter solutions despite phenotypic/genetic variations, and the closely related issue of robustness [149], i.e. **continuous** development and effective operation in the presence of noise, uncertainty, imprecision, error and damage complete the set of characteristics of biological systems, [112, 114, 115, 85, 86]. Therefore, it is necessary to establish different criteria for evaluating biological computation based on its primary purposes.

2.4 Implications

Almost a decade before carrying out the first computer simulation experiments of autopoiesis [83], Varela and Letelier tested Sheldrake's theory of morphic resonance in silicon chips using a microcomputer simulation [142]. Briefly, they let a crystal grow as a microcomputer simulation. It was expected that if the morphic field theory were correct, the synthesis of the same crystal pattern would be **accelerated** after millions of iterations. However, Varela and Letelier found that there was no detectable acceleration of the growing process at all. They concluded that either Sheldrake's hypothesis was falsified or that it does not apply to silicon chips. However,there might be other explanations of this result. One of them could be that conventional Turing machines were used to simulate the above experiments. In this case, new computation models beyond the Turing machine concept are needed to address such issues as autopoiesis.

So, biological computation approaches advocate a shift towards a computation paradigm closer to the domains of quantum physics and biology [9, 8, 10]. In contrast, traditional engineered systems are expected to evolve and adapt according a limited set of production rules to maintain continuous system equilibrium. Such behaviour can be described, for example, by using the equations of classical dynamics. Unfortunately, this is a fairly limited case in biology. Most physical and chemical processes exhibit alteration, irregularity and contingency along with consistency, orderliness and pattern formation. Besides, biological systems are not conservative, but **dissipative** with fluctuations, unsteadiness and irreversible processes playing a central role [93]. In addition, contradictions and paradoxes (such as the ones known in quantum mechanics)

[5]Problems that can be solved but not fast enough for the solution to be useful are called intractable [49].

cannot be banned from the biological sciences, being an integral part of life and cognition [109]. The more classical engineering and computation enter the domains of the biological sciences, the more these phenomena and anomalies must be faced. Science does not know life systems well enough to make the claim of being able to create them error-free.

We still believe that the way to use autopoietic systems as computational tools lies in constructing the notion of temporally correlated **structural coupling** in computing. Structural coupling is a mechanism by which a lineage of autopoietic systems can change their structure (i.e. components and processes), which is congruent with the recurrent perturbations that arise in the medium. Also, autopoietic computing may require hardware that we do not currently produce. However, we believe we will be able to more definitely propose an autopoietic form of computing, but only once there has been further significant development into a more complete non-reductionist understanding of the **phenomenon** of autopoiesis, which will allow more viable and robust formalisations of autopoietic behaviour.

2.5 Cellular Business Models

In the same the same way that the Digital Business Ecosystem (DBE) would be composed of the Software Ecosystem and Business Ecosystems, we can consider the interaction of our biological-computing-based autopoietic computing with cellular business models. There are **organisational** innovations that allows companies to *bundle* their activities into discrete business units called *cells* that can expanding through the creation of clusters of these business cells. This cellular cluster organisational template is a dramatic departure from the conventional organisational structures, where subsidiaries are created under close supervision of headquarters, and the whole is held together through structures of hierarchy and control. The key point is that while firms expand from the *periphery* have many potential advantages, they must, in order to capture these advantages, display management and organisational capabilities of a high order. This offers the potential for more **organic** growth through partnership [75].

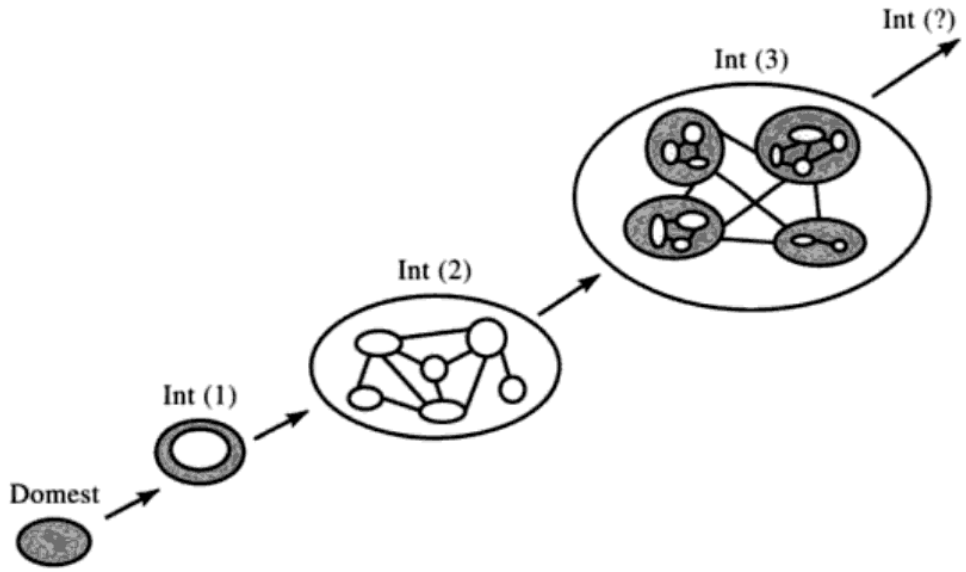

Figure 2: *Cellular Business Model [75]: An evolution through several stages of cellular expansion of the cellular business model. The first phase is of international expansion, with a centralised organisational structure. With its limits reached, it evolves to a second phase, in which cellular business units (BUs) become the primary site of a corporation's activities.*

An evolution through several stages of cellular **expansion** is depicted in Figure 2. A corporation starts as a purely domestic operation, while it acquires the necessary capabilities to make its launch on the world. It evolves from this to a first phase of international expansion, with a centralised organisational structure. This is a perfectly traditional organisational architecture, involving no more than a functional division of labor. However, when this reached its limits, it evolves to a second phase, in which cellular business units (BUs) become the primary site of a corporation's activities, interacting directly with each other (e.g. supplying components and sub-assemblies) and with external customers. This organisational architecture can reach its limits, then evolving to a third phase, involving four **overarching** BUs (cells), each again containing constituent BUs or cells. This phase can be expected to evolve to yet another form, or realisation of the cellular architecture, when the time is ripe. Indeed, a fourth phase can initiate with the cells (BUs) being spun off as a new entities [75].

While we have only considered a relatively microeconomic perspective, this could equally apply on macroeconomic scale. So, such cell-based business models could interact **symbiotically** with our biological-computing-based autopoietic computing, creating more cohesive socio-technical economic systems.

3 Natural Computation - Autopoietic (Inspired) Computing

Biomimicry in computer science is called Natural Computation, and the benefits of natural computation technologies often mimic those found in real natural systems, and include flexibility, adaptability, robustness, and decentralised control [23]. The increasing demands upon current computer systems, along with technological changes, create a need for more flexible and adaptable systems. The desire to achieve this has led many computing researchers to look to natural systems for **inspiration** in the design of computer software and hardware, as natural systems provide many examples of the type of versatile system required [23]. Their sources of inspiration come from many aspects of natural systems: evolution, ecology, development, cell and molecular phenomena, behaviour, cognition, and other areas [74]. The use of nature inspired techniques often results in the design of **novel** computing systems with applicability in many different areas [74].

All branches share the common characteristic of human-designed computing inspired by nature, the **metaphorical** use of concepts, principles, and mechanisms underlying natural systems. Thus, evolutionary algorithms use the concepts of mutation, recombination, and natural selection from biology; neural networks are inspired by the highly interconnected neural structures in the brain and the nervous system; molecular computing is based on paradigms from molecular biology; and quantum computing based on quantum physics exploits quantum parallelism [23]. There are however, important **methodological** differences between various sub-areas of natural computing. For example, evolutionary algorithms and algorithms based on neural networks are presently implemented on conventional computers. However, molecular computing also aims at alternatives for silicon hardware by implementing algorithms in biological hardware, using DNA molecules and enzymes. Also, quantum computing aims at non-traditional hardware that can make use of quantum effects [23]:

We are concerned with Biologically-Inspired Computing (BIC), which naturally relies heavily on the fields of biology and computer science. Briefly put, it is the study of nature to improve the usage of computers [34], and should not to be confused with *computational biology* [151], which is an **interdisciplinary** field that applies the techniques of computer science, applied mathematics, and statistics to address problems inspired by biology. BIC has produced Neural Networks, swarm intelligence, evolutionary computing [34] and autonomic computing [56]. So, we will now consider an autopoietic inspired form of computing which builds upon autonomic computing, which has come to epitomise self-managed information systems [37]. However, it was based upon autonomic systems, specifically the autonomic nervous system [56], and therefore lacks core functionality available to other systems in the body. Specifically, replication, which is a definitive property harnessed by the autopoietic immune system [89]. So, we will consider extending **autonomic computing** to include the the concept of component replication in self-management regimes, based primarily upon the autopoietic immune system. We shall therefore will start by considering the autonomic nervous system, the inspiration for autonomic computing, before then considering the autopoietic immune system as inspiration for an autopoietic inspired form of computing. Finally, we consider possible usage scenarios for this autopoietic computing.

3.1 Autonomic Nervous System

The autonomic nervous system is the part of the peripheral nervous system that acts as a control system functioning largely below the level of consciousness, and controls visceral functions. These affect heart rate, digestion, respiration rate, salivation, perspiration, diameter

of the pupils, micturition (urination), and sexual arousal. Whereas most of its actions are **involuntary**, some, such as breathing, work in tandem with the conscious mind. It is classically divided into two subsystems: the parasympathetic nervous system and sympathetic nervous system. Relatively recently, a third subsystem of neurons that have been named *non-adrenergic and non-cholinergic* neurons (because they use nitric oxide as a neurotransmitter) have been described and found to be **integral** in autonomic function, particularly in the gut and the lungs.

The autonomic nervous system regulates the life support systems of the body reflexively, i.e. without conscious direction. It **automatically** controls the muscles of the heart, digestive system, and lungs; certain glands; and homeostasisthat is, the equilibrium of the internal environment of the body. The autonomic nervous system itself is controlled by nerve centres in the spinal cord and brain stem and is fine-tuned by regions higher in the brain, such as the midbrain and cortex. Reactions such as blushing indicate that cognitive, or thinking, centres of the brain are also involved in autonomic responses [65].

With regard to function, the autonomic nervous systems is usually divided into sensory (afferent) and motor (efferent) subsystems. Within these systems, however, there are inhibitory and excitatory synapses between neurones. The sympathetic and parasympathetic divisions typically function in opposition to each other. But this opposition is better termed complementary in nature rather than antagonistic. For an **analogy**, one may think of the sympathetic division as the accelerator and the parasympathetic division as the brake. The sympathetic division typically functions in actions requiring quick responses. The parasympathetic division functions with actions that do not require immediate reaction. Consider sympathetic as *fight or flight* and parasympathetic as *rest and digest*.

However, many instances of sympathetic and parasympathetic activity cannot be ascribed to *fight* or *rest* situations. For example, standing up from a reclining or sitting position would entail an unsustainable drop in blood pressure if not for a compensatory increase in the arterial sympathetic tonus. Another example is the constant, second to second modulation of heart rate by sympathetic and parasympathetic influences, as a function of the respiratory cycles. More generally, these two systems should be seen as permanently modulating vital functions, in usually antagonistic fashion, to achieve **homeostasis** (through feedback loops, both positive and negative). Some typical actions of the sympathetic and parasympathetic systems are listed below.

This brief introduction serves to explain the fundamentals of the autonomic nervous system, and is still far greater consideration than it received in the creation of autonomic computing [56]. This is because autonomic computing was quite rightly based on the high-level abstract **principles** of the autonomic nervous system. However, we suspect that the lack of greater consideration of the autonomic nervous system was such that it was not fully appreciated, especially its uniqueness when compared to other systems of the vertebrate anatomy. Specifically, the lack of cellular **replication** as a core functionality. Equally, as biologically-inspired computing often originates from fulfilling an engineering need or problem, the scope of the form of computing envisaged did not necessitate any other additional properties than those found within the autonomic nervous system. Whether the latter or the former, or a combination of the two, be the reason, we now believe that an **opportunity** exists for a more inclusive inspiration of anatomical systems in defining computing paradigms in the context of current computational engineering challenges. However, we will first have to consider the current state-of-the-art of autonomic computing and the autopoietic immune system [134, 130].

3.2 Autonomic Computing

Autonomic computing refers to the **self-managing** characteristics of distributed computing resources, adapting to unpredictable changes whilst hiding intrinsic complexity to operators and users. An autonomic system makes decisions on its own, using high-level policies; it will constantly check and optimise its status and automatically adapt itself to changing conditions. The ultimate aim being to develop computer systems capable of self-management. So, overcoming the rapidly growing **complexity** of computing systems management, and reducing the barrier that complexity poses to further growth [55].

A general problem of modern distributed computing systems is that their complexity, and in particular the complexity of their management, is becoming a significant **limiting factor** in their further development. Large companies and institutions are employing large-scale computer networks for communication and computation. The distributed applications running on these computer networks are diverse and deal with many different tasks, ranging from internal control processes to presenting web content and to customer support. Additionally, mobile computing is pervading these networks at an increasing speed: employees need to communicate with their companies while they are not in their office. They do so by using laptops, PDAs, or mobile phones with diverse forms of wireless technologies to access their companies' data. This creates an enormous complexity in the overall computer network which is hard to control manually by one or more human operators. Manual **control** is time-consuming, expensive, and error-prone. The manual effort needed to control a growing networked computer system tends to increase very quickly.

A possible solution could be to enable modern, networked computing systems to manage themselves without direct human **intervention**. The Autonomic Computing Initiative (ACI) aims at providing the foundation for autonomic systems. It is inspired by the autonomic nervous system of the human body. This nervous system controls important bodily functions (e.g. respiration, heart rate, and blood pressure) without any conscious intervention. In a self-managing Autonomic System, the human operator takes on a new role: He does not control the system directly. Instead, he defines general **policies** and rules that serve as an input for the self-management process. For this process, IBM has defined the following four functional areas [56]:

- Self-**Configuration**: Automatic configuration of components
- Self-**Healing**: Automatic discovery and correction of faults
- Self-**Optimisation**: Automatic monitoring and control to optimise resource provisioning
- Self-**Protection**: Proactive identification and protection from arbitrary attacks

These self-CHOP properties, as they are called, are summarised in the well known autonomic computing image reproduced in Figure 3, in which each property takes up a quarter of the circle showing their equal importance.

In *The Vision of Autonomic Computing* [56], Kephart and Chess warn that the dream of **interconnectivity** of computing systems and devices could become the *nightmare of pervasive computing* in which architects are unable to anticipate, design and maintain the complexity of interactions. They state the essence of autonomic computing is system self-management, freeing administrators from low-level task management while delivering more optimal system behaviour. However, self-management means different things in different fields. Still, the number of computing devices in use is forecast to grow at 38% per annum and the average

complexity of each is increasing. Currently this volume and complexity is managed by highly skilled humans; but the demand for skilled IT personnel is already outstripping supply, with labour costs exceeding equipment costs by a ratio of up to 18:1. Computing systems have brought great benefits of speed and automation but there is now an overwhelming **economic** need to automate their maintenance.

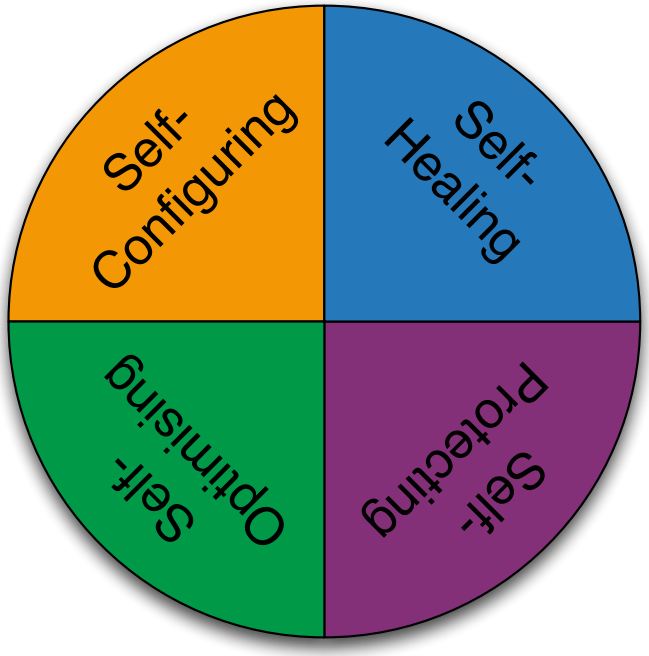

Figure 3: *Autonomic Computing: Each property takes up the quarter of the circle showing their equal importance. Self-configuring for increased responsiveness to adapt to dynamically changing environments. Self-healing for business resilience, to discover, diagnose, and act to prevent disruptions. Self-optimising for operational efficiency to tune resources and balance workloads to maximise use of IT resources. Self-protecting to secure information and resources, anticipating, detecting, identifying and protecting against attacks.*

IBM defined five levels for the autonomic **deployment** model. Level 1 deployment is the basic level that presents the current situation where systems are essentially managed manually. Levels 2 - 4 introduce increasingly automated management functions, while level 5 represents the ultimate goal of autonomic, self-managing systems.

A basic concept to be applied in autonomic systems is that of closed **control** loops. This well-known concept stems from Process Control Theory. Essentially, a closed control loop in a self-managing system monitors some resource (software or hardware component) and **autonomously** tries to keep its parameters within a desired range. According to IBM, hundreds or even thousands of these control loops are expected to work in a large-scale self-managing computer system. Driven by such vision, a variety of architectural frameworks based on *self-regulating* autonomic components has been recently proposed. A very similar trend has recently characterised significant research work in the area of **multi-agent systems**. However, most of these approaches are typically conceived with centralised or cluster-based server architectures in mind and mostly address the need of reducing management costs rather than the need of enabling complex software systems or providing innovative services. So, an autonomic computing framework can be seen as **composed** of autonomic components interacting with each other [154]. An autonomic component can be modelled in terms of two main control loops (local and global) with sensors (for self-monitoring), effectors (for self-adjustment), knowledge and adapter for exploiting policies based on self- and environment awareness.

3.3 Autopoietic Immune System

An immune system is a system of biological structures and processes within an organism that **protects** against disease by identifying and killing pathogens and tumor cells. It detects a wide variety of agents, from viruses to parasitic worms, and needs to distinguish them from the organism's own healthy cells and tissues in order to function properly.

Autopoietic theory describes a model of self-organisation [145, 76] in which a system's organisation is defined by its *structure* (its components(nodes) and their relations (links)) and the processes that this structure performs, which continuously *regenerate* the structure that produces them. So, in the context of autopoietic theory, the immune system extends the *idiotypic network theory*[6]. The immune system is no longer considered to be antigen driven. Rather the immune system is an organisationally closed network that reacts autonomously in order to define and preserve the organism's identity, in what is called self-assertion [144]. The autopoietic view of the immune system stresses self-reactiveness in contrast to other views/approaches [77]. The immune system produces different responses depending on the external *stimulus* and its current state [12]. As the immune system changes over time (e.g. during development) the same stimulus can trigger different responses [89].

An **autopoietic view** of the immune system is that of a self-organising antibody network. The primary job of the immune system is to protect the host organism from foreign molecules. This is done in part by antibody molecules which are manufactured by B-cells (B-lymphocytes). Antibodies have specific 3-D structures that bind to foreign antigen molecules and thus signal their removal by other cells. The portion of the antibody molecule that can identify other molecules is called the paratope. When the paratope matches an antigen molecule sufficiently, the two attach leading to the antigen's removal. Paratopes can attach to specific regions on another molecule (antigen or antibody) that are called epitopes. Since the match between a paratope and an epitope does not have to be exact, a paratope may attach to an entire class of epitopes. So given enough antibody diversity, the probability that an epitope is not recognised is very small [107]. In addition, when an antibody attaches to an antigen, the corresponding B-cell is stimulated to clone and secrete more antibodies. This **process** which results in increased concentrations of useful antibodies is called clonal selection. The cloning process is not perfect, but is subjected to intense mutation, called somatic hypermutation, that results in slightly different B-cells and hence antibodies, which can be a better match to the invading antigen. Further diversity of antibody repertoire is maintained through replacement of a percentage of B-cells by new B-cells manufactured in the bone marrow.

The immune system is particularly good at maintaining and boosting diversity. This is achieved in two ways: with heterostasis, the preservation of diversity and heterogenesis, the introduction of diversity. On one hand, the diversity of the immune repertoire is preserved due to the way immune cells (lymphocytes in particular) are triggered to *clone*. According to clonal selection, a lymphocyte is triggered to clone when its antibodies recognise an antigen. The cloning rate is a function of the affinity between antibody and antigen and is usually high. This is a local selection

[6] A network of antibodies capable of neutralising self-reactive antibodies exists naturally within the body. The thrust of the idiotypic theory is that an antigen generates antibodies whose serologically unique structure, called an idiotype, results in the production of anti-idiotypic antibodies. The original antibody is designated Ab_1, and the anti-idiotypic antibody Ab_2. The Ab_2 antibodies recognise the antigen-binding site of Ab_1, and therefore share a motif or structural similarities with the original antigen. The cascade then perpetuates, with the generation of anti-anti-idiotypic antibodies (Ab_3) that recognise Ab_2, and so on. The end result can be the production of a chain of autoantibodies that recognise each other and that may modulate the immune system by stimulating or suppressing it [51].

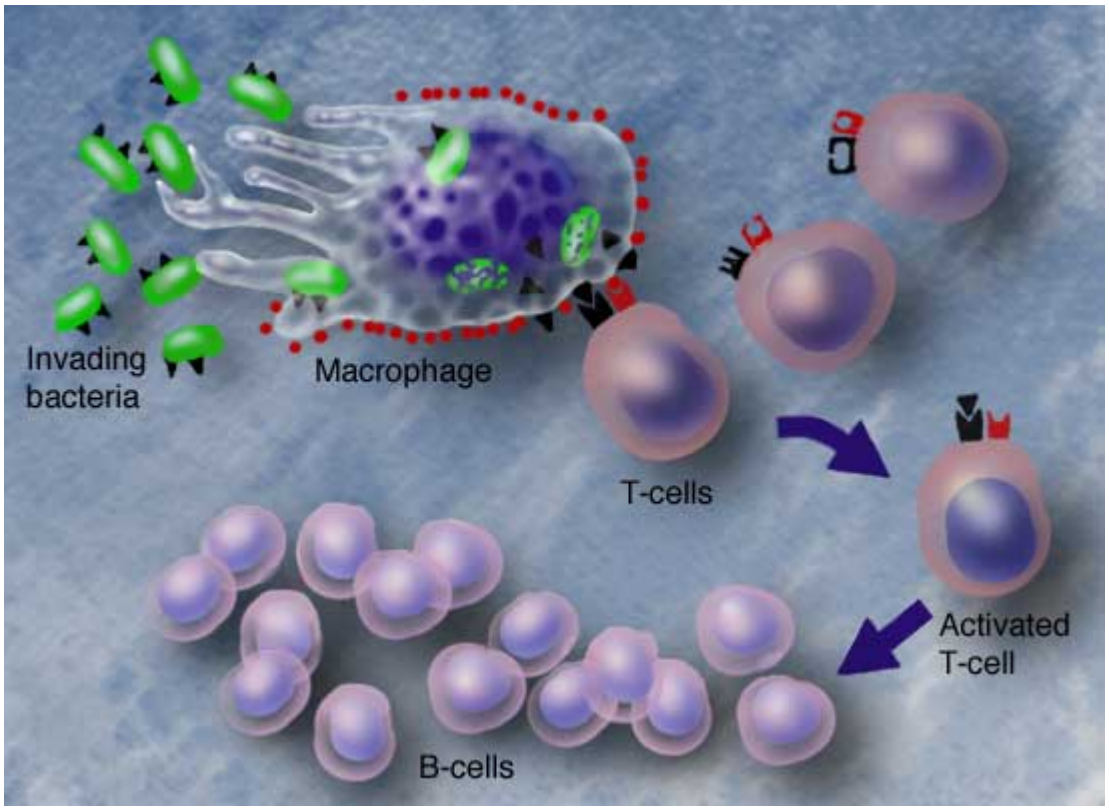

Figure 4: *T-Cell Response: Invading bacteria are phagocytosed (taken into the body of the macrophage). The macrophage then removes the specifics identifying markers, or antigens, of the invading bacteria. It places these antigens on its own surface. Helper T-cell lymphocytes which are specially adapted to fight this invading bacteria have on their own surface a complementary antigen marker, much like a puzzle piece. When these specialty T-cell lymphocytes notice a macrophage presenting the complement antigen they initiate a response which results in B-cell proliferation.*

scheme that chooses cells to **proliferate** not according to their relative fitness (fitness relative to one another), but rather according to the absolute value of their affinity to an invading antigen, because it provides greater effectiveness at suppressing an infection. So even lymphocytes that have not been successful so far in recognising antigens can be triggered to clone when the antigenic environment changes and new matching antigens invade the organism. In addition, long-lived memory cells that have been successful in the past and idiotypic effects, contribute further to heterostasis.

While dynamics play the role of reinforcement learning, a metadynamics functions as a distributed control mechanism which aims at maintaining the viability of the network through an on-going shift of immune repertoire [13]. One significant aspect of the immune network's **metadynamics** is that the system itself is responsible for selecting new cells for recruitment. New cells are constantly produced by the bone marrow, but only those with structure that matches that of existing cells in the immune network survive. As has been pointed out [146], it is this process of endogenous selection that results naturally in **self-assertion** and, as a complement, allows the dynamic control of the repertoire size.

So, while the autonomic nervous systems is primarily made of non-replicating components, some of which are never replaced should they fail (e.g. nerve cells of the spine [3]), the autopoietic immune system is primarily made of components which undergo constant **replacement**, partly to achieve the necessary diversity required in immune system response, but also to support the

dynamic resource provisioning required for the immune response. It is this ability, made possible by the self-replication principles of autopoiesis, which we can make use of in extending autonomic computing to define autopoietic computing.

3.4 Autopoietic Computing

Considering *autonomic computing* relative to the *autonomic nervous system*, and *autopoietic computing* relative to the *autopoietic immune system*, then we must consider the differences of the *autopoietic* immune system relative to the *autonomic* nervous system. We observe that the autopoietic immune system demonstrates all of the same controlling behaviour of the autonomic nervous system, plus the ability of the self-replication of its components (e.g. B-cells). Therefore, we can postulate that while autopoietic systems demonstrate the behaviour of an autonomic systems, they also demonstrate the behaviour of **self-replication**, and so we can consider autonomic systems a sub-class of autopoietic systems. So, we would expect autopoietic computing would include the self-CHOP (Configuring, Healing, Optimising, Protecting) properties of autonomic computing, plus a self-Replication property, creating CHOPR.

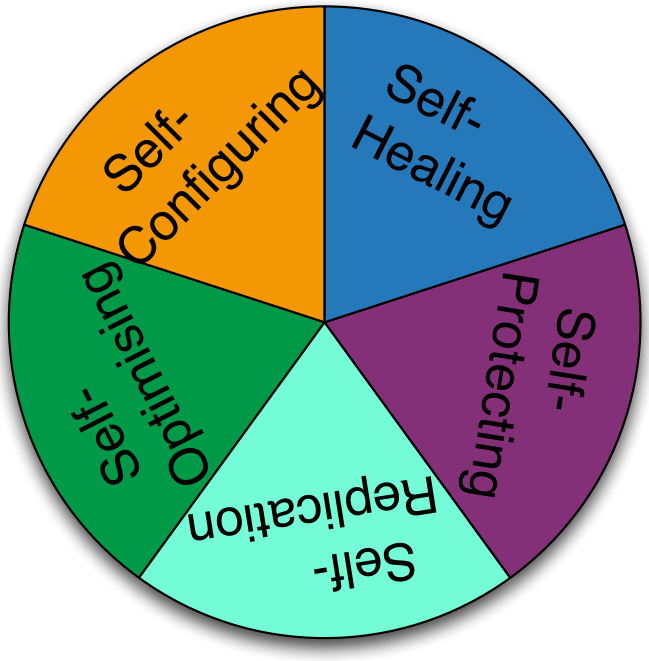

Figure 5: *Autopoietic Computing: Each property takes up a fifth of the circle showing their equal importance. Most important is the addition of self-replication for the greater distribution of control and dynamic resource scaling.*

The **vision** of autopoietic computing proposed is one of dynamic resource scaling, which has many potential applications. One obvious one is the provisioning of resources in Cloud Computing, where there is an inherent necessity for the **dynamic** scaling of resource provisioning. This would include our Community Clouds, which is a realisation of Digital Ecosystems in the context of Cloud Computing, with a key differentiator being the distribution of **control**.

3.5 Applicability

Cloud Computing is the use of Internet-based technologies for the provision of services [44], originating from the *cloud* as a metaphor for the Internet, based on how it is depicted in computer network diagrams to abstract the complex infrastructure it conceals [129]. It can be seen as a commercial evolution of the academia-oriented Grid Computing [36], succeeding where Utility Computing struggled [35, 100]. It is being promoted as the cutting edge of scalable web application development [5], in which dynamically scalable and often virtualised resources are provided as a service over the Internet [41, 44, 38, 39], with users having no knowledge of, expertise in, or control over the technology infrastructure of the Cloud supporting them [22]. Here then, Cloud Computing derives characteristics from autonomic computing, specifically comprising computer systems capable of **self-management**. However, instead of the centrally controlled resource replication that is common place in data centres, we could instead follow our autopoietic computing approach to make the components (services) self-replicating, and so beyond the self-CHOP properties of autonomic computing.

A *data centre* is a facility, with the necessary security devices and environmental systems (e.g. air conditioning and fire suppression), for housing a *server farm*, a collection of computer servers that can accomplish server needs far beyond the capability of one machine [6]. While there are various models for Cloud Computing, we shall consider the one of Amazon Web Services, which is the current **incumbent** of the industry. In our opinion, their success comes primarily from offering Infrastructure as a Service (IaaS) [92], which is at the most basic level of the Cloud Computing offerings. They provide *machine instances* to developers. These instances essentially behave like dedicated servers that are controlled by the developers, who therefore have full responsibility for their operation. So, once a machine reaches its performance limits, the developers have to **manually** instantiate another machine and scale their application out to it. This service is intended for developers who can write arbitrary software on top of the infrastructure with only small compromises in their development methodology.

However, a vision for automatic and dynamic scaling is lacking in their model. With our autopoietic computing, and its **self-replication** specifically, services would have a model for dynamically scaling through the nodes of a data centre. Beyond the necessity of task or service duplication for redundancy, services would literally replicate across nodes to match capacity requirements, in the same way that the autonomic immune system replicates entities to fight infection. Then, as **capacity** requirements fall, service replicants, upon the completions of tasks and the significant lack of additional tasks, would not only cease replicating, but would shutdown clearing nodes for other use, in the same way that the autopoietic immune system scales down after fighting off an infection.

While the scenario we presented was in the context of a data centre for Cloud Computing, it would be equally applicable in the context of Community Clouds, which utilise the available resources of networked personal computers to provide the facilities of a virtual *data centre*. The server farms within data centres are an intensive form of computing resource provision, while the Community Cloud is more **organic**, growing and shrinking in a symbiotic relationship to support the demands of the community, which in turn supports it. So, this symbiotic growth of Community Clouds maps even more strongly to the self-replication process of entities (services) of our autopoietic computing.

4 Self-Replication: Foundational Studies

Understanding how to harness self-replicating entities in autopoietic organisations (e.g. cells in the immune system) requires advances in relating self-replication to organisational level emergence of functionality (cf. also [1] for an introduction to self-replication). Foundational investigations of self-replication carried out therefore focus on the achievement of and roles of three key phenomena: inheritable variation, sex (i.e. transfer of inheritable information) and its impact on the evolution in self-replicators, and the trade-off between complexity and pay-off in the emergence of higher levels of organisation requiring cooperation.

4.1 Self-Replicators and Complexity

Von Neumann [148] initiated the study of self-replication with consideration of the question of how it could be mechanistically possible for living things or machines to produce something of equal or of greater complexity than themselves. In the late 1940s, still before the discovery of the structure of DNA by Watson and Crick [152], he considered several abstract solutions including a cellular automaton in which he exhibited the design of a self-reproducing configuration called a universal constructor (capable also of universal computation) capable of constructing any bounded configuration according to a description on a tape, and copying the instruction tape, and attaching the copied description to the constructed configuration. In this way, he presented an elegant solution to problem of self-replication: by providing the universal constructor and copier with a description specifying how to construct them, the description is copied and attached to a newly constructed copy of the universal constructor/computer. The result is then the original automaton still intact with an offspring copy.

In principle, he pointed out this indicates the path to the evolution of more and more complex automata: mutations on the description, if not lethal, would be copied along with this description, to the offspring, and inherited by subsequent generations. Thus inheritable mutations could result in the evolutionary growth in complexity over several generations. McMullin [81] has argued that this was von Neumann's key interest and contribution in founding the studying of self-reproducing automata. Mutations drive evolution: without variation, arising from mutation or other sources of variability such as recombination, natural selection has no substrate upon which to act. Von Neumman's work can form a basis for understanding how self-reproducing structures could lead to evolutionary adaptation and increases in complexity in autopoeitic organizations based on self-reproducing components, but this must be accompanied by a study of inheritable mutation and variability as basic for such evolvability.

Pesevanto [108] with the help of collaborators and subsequent researchers succeeded in implementing von Neumann's self-reproducer, which is extremely large and complex, and it has recently, only in the last few years, become feasible to carry out the replication, and moreover, inheritance of single mutation has been demonstrated. To achieve the evolutionary potential of this system, complexity change — in particular, complexity increase – over several generations must be demonstrated. This has been accomplished by Yinusa at the University of Hertfordshire with the addition of a description for a new component to the self-replicator in a mutation to its tape description, followed by the addition of yet another component in the tape of the offspring. Thus successive generations from the initial von Neumann self-replicators arguably show increase in complexity over successive generators [159].

However von Neumann's system and its derivatives are very brittle with the chance of

spontaneously occurring viable mutation being very small. The increase in complexity demonstrated just mentioned relied on human design of the modified description tape, and no autonomous process of evolutionary change (with intrinsically generated heritable mutations and differing reproductive success, i.e. Darwinian natural selection) has to date been exhibited with the complexity of von Neumann self-replicators.

If one gives up the universal construction and computation capacity of the von Neumann replicators then there do exist a class of small self-replicators introduced and studied by H. Sayama [128], based on Langton's self-reproducing loops [64], that do exhibit Darwinian evolution (cf. [90]), however apparently these are still without convincing examples of complexity increase.[7]

In a study of inheritable mutations in the von Neumann self-replicator, making use of the GOLLY simulator, we carried out multi-point mutations on the tape which holds the description of the self-replicator so as to observe the effects over subsequent generations. Von Neumann [148] emphasised that *Self reproduction includes the ability to undergo inheritable mutations as well as the ability to make another organism like the original.* Thus subsequent offspring generations may inherit complex attributes from the parent generation during reproduction. Harmful and neutral mutations were documented, and we also give the first demonstration of cumulative complexity increase based on inheritable mutations to the von Neumann self-replicator. The latter achievement is summarised here and in [159].

4.2 Inheritable Mutation in the Von Neumann Self-Replicator

This project was implemented using the GOLLY simulator and the Von Neumann self replicator with the variant Nobili 32-state rule for the von Neumann self-replicator. The method used for studying inheritable mutations here is the insertion method. This involves the injection of a sequence of codon(s)[8] into unoccupied spaces on the tape which is occupied by the *zero* state, whereas the substitution method involves flipping or changing bit(s) of codons. To achieve inheritable mutation, single-point and multi-point mutation was done on the tape.

EXPERIMENT 1: Single-point mutation was done on the tape of the self replicator. This was done within the first 2,669 cell spaces of the tape. A codon of a quintuplet identified with a *zero* bit representing an unoccupied space within the cellular space was injected with a *one* bit. Upon insertion of the *one* bit into the tape, the information on the tape was changed. After self-reproduction, the structure appears almost the same but with a little difference which is a single *dot*, slightly out of place on the left side of the automaton. Figure 6 shows the universal constructor with the effect of the mutation which is not lethal but also not very interesting. This effect is of no benefit to subsequent generation(s). This is called a Neutral mutation. Neutral mutations survive or sometimes disappear during evolution [42] and they have little or no effect on the fitness of offspring. According to Von Neumann [148], *Self reproduction includes the ability to undergo inheritable mutations as well as the ability to make another organism like the original.* Though previous work had been done on mutation of Von Neumann's self-replicator, this was not fully documented. The project carried out shows that inheritable mutation in the Von Neumann self-replicator does exist and hence can be implemented. To

[7] Instead, generally speaking these self-replicators tend to decrease in size over the course of evolution, allowing them to replicate faster.

[8] In this case, these *codons* are 5-tuples of symbols on the description tape.

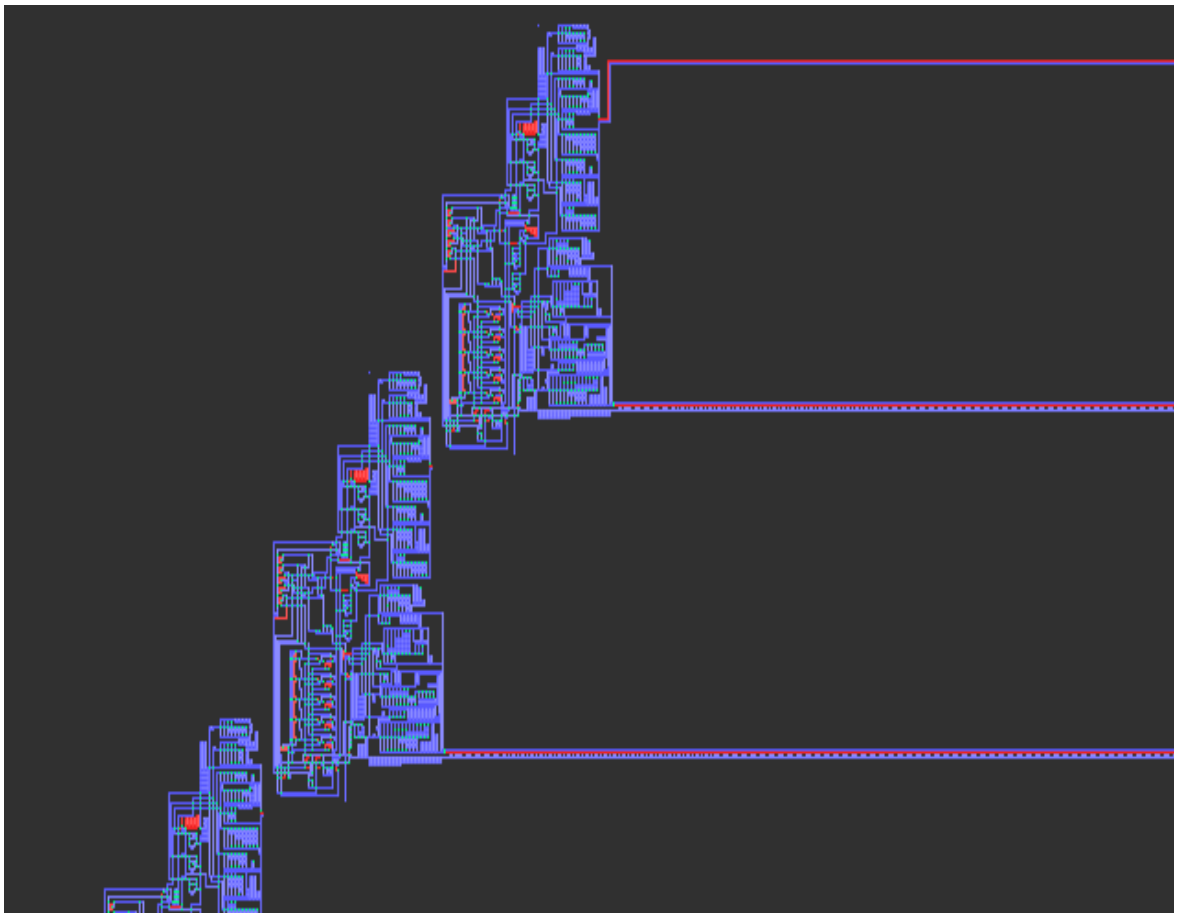

Figure 6: *Neutral mutation from Experiment 1.*

guarantee evolvability in the self-replicator, Von Neumann included a universal constructor in the body of the self-replicator thereby enabling it to make a copy of itself as well as making anything the tape instructs it to do hence potentially creating a more complex systems than itself. Mutation on the tape allows the next generation of a system to behave in a more complex way than the parent thereby guaranteeing the evolvability of the lineage. Not only must it behave in a complex manner, but it must also be able to produce complex organisms such as itself. With respect to Von Neumann's work, important mutation types have been established as either *inheritable* or *lethal*. A lethal mutation involves a random change of an element in B of a self-replicating automaton (A+B+C) hence failure in the system to make connections and/or stimulate themselves when they have to while act. Inheritable mutation involves, e.g. an automaton (A+B+C) with description $\varphi(A+B+C)$ altered such that $\varphi(A+B+C)$ is $\varphi(A+B+C+D)$ and the system reproduces a modified copy of itself $(A+B+C+D)$ with a new copy of the description $\varphi(A+B+C+D)$ [148].

EXPERIMENT 2: Different stages of self-replication from one generation to another are shown in the figures below. In Figure 7 (a) and (b), multi-point mutation was done on the tape of the first generation of the universal constructor. The effect of this mutation was observed in the second generation and subsequent generations and it was observed that the mutation inherited in the second generation was also inherited by the third and fourth generations. This is called the *banana* mutation because the shape of the a new component in the offspring replicator. It was observed that the mutation inherited by the second generation made it different in structure and hence, in behaviour thereby making it a more complex structure than its parent. This mutation

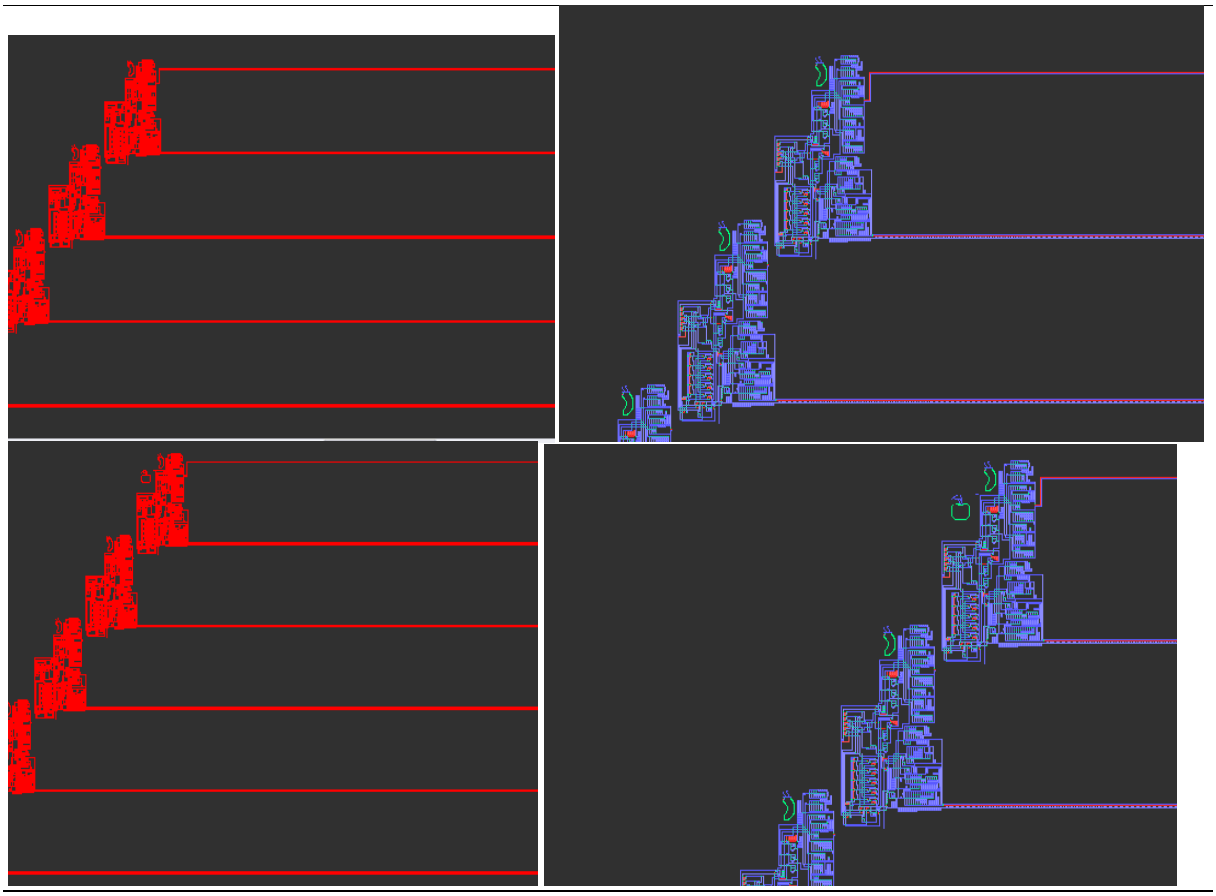

Figure 7: *Cumulative complexity increased over the course of successive inheritable mutations in self-replicators. Experiment 2: (a) Upper left: Fourth generation of replicators exhibiting the inheritable banana mutation (b) Upper right: Close-up of the self-replicators with inheritable banana mutation, (c) Lower left: Screen shot of the first generation with both the inheritable apple mutation and banana mutations (d) Lower right: Close-up of the self-replicators with cumulative genome change exhibiting both the apple and banana inheritable mutations.*

was further inherited by subsequent generation(s) making them also complex systems like the parent. Such complexity underpins evolution which involves a system producing another system like itself which has the ability to behave in a more complex manner and also the ability to produce a complex system like itself. As a result of the above positive effect of the multi-point mutation on the universal constructor, a further mutation experiment was performed on the resulting self-replicator so as to establish the feasible of successive complexity increase.

Figure 7 (c) and (d) show the effects of successive multi-point mutations done on the fourth generation of the banana self-replicator. The tape in the fourth generation of the banana self-replicator was mutated. The fifth generation shows the effect of the mutated tape which was inherited. Another mutation apart from the banana mutation was inherited from modifying the description in the tape of the banana self-replicator parent (which is the previous generation) with a further insertion. The previous generation which is a banana self-replicator reproduced a more complex system than itself with the system having in addition to the banana component, also an apple automaton. If observed further, this apple-banana self-replicator breeds true and reproduces a structure like itself unless the tape is again further mutated by deleting, inserting or substituting codons on the tape.

Mutations have been done and studied in different generations and over several generations. The tape encoding has also been described up to a certain point so it is easier for others to understand in our work [159]. According to McMullin in [81], Von Neumann was trying to model a system that can behave live a biological organism such that during evolution, there is an increase in complexity of the system and complex systems can give rise to complex systems. This basis for evolution in Von Neumann's self-replicator has been established here by exhibiting successive inheritable mutations and the resulting increasing complexity phenotypes.

4.3 SexyLoop: Replicators Sharing Information

It has been argued that sex can speed up evolutionary change by allowing the recombination of components that could together provide a useful function, to come together and realise such functions. In biological terms, sex is just the transfer of inheritable genetic material between individuals. There is a long debate on how sex could have evolved or can be maintained, and what roles it may play in evolution [79, 78, 84]. In artificial self-replicators, sporadic occurrences of sex could be observed in Tom Ray's Tierra system [116] in which programs self-replicate in a virtual Darwinian operating system. Building on work of Oros and Nehaniv [101, 102], we created a model in a variant of Sayama's evoloop system [128] that allows self-replicating individuals to carry a heritable gene that enables them to transmit genetic information to others they collide with in an evolving population. As a model system, we studied the persistence of sex in evolving self-reproducing structures in such a cellular automaton space.

In previous work [102], we created a simple evolutionary system, F-sexyloop, on a deterministic twelve-state five-neighbour cellular automaton (CA) where self-reproducing loops have the capability of sex. This work was based on the sexyloop [101], which was transformed by adding two new states and new rules. In the F-sexyloop, the loops could carry a sex gene used to facilitate the transfer of genetic material from a loop to another. This gene is analogous to the F factor plasmid in bacterial conjugation, which confers the capacity to act as a donor of genetic material (including the gene itself). Therefore, the sex gene could potentially be maintained in the population during evolution or disappear. In the F-sexyloop, the loops could transfer their genes only if they had a sex gene in their genome. We have shown that in most of the experiments, sex resulted in greater diversity, faster evolution, and also the sex gene persisted in time during the evolutionary process and was also established in dominant species. However, sex may have hitched-hike on selection pressure for ability to detach when loops interact. In the F-sexyloop, when a loop collided with another one and did not have a sex gene, it remained attached to it until another loop collision killed it (Figure 8). Therefore, there was a high selection pressure to have sex, intrinsic to the system.

A natural next step from this work was the development of a null model that would make it possible to distinguish between random drift and selection for or against the sex gene. Therefore, we developed a new version of the F-sexyloop where the loops could carry a sex gene (like in the F-sexyloop) or they could carry a *dissolver* gene. This new gene allows the loops that do not have a sex gene to disconnect their arm from another loop. The effect of both sex and dissolver genes do not differ during self-replication, however they trigger different mechanisms when interaction occurs, i.e. a collision and thus potential connection is made, between two loops. With such a model, we could study the persistence of the sex gene and the dissolver gene during the evolutionary process and see if the sex gene persists only due to the benefits (faster evolution, increased diversity) of transfer of genetic material, or to due allowing loops to detach when they interact. Figure 9 shows one loop (bottom) that grows its arm towards another loop

then binds to it. Once this sex connection is made, the loop at the bottom can transfer its genes into the other loop using its sex gene to dissolve the sheath blocking the signal (genes) to move in. The mechanism used here is similar to the one used in the original F-sexyloop.

Figure 10 shows the arm of one loop (bottom) colliding with another loop. This time, the loop at the bottom has a dissolver gene instead of a sex gene. This gene enables the loop to detach and dissolve its arm. There is no transfer of genetic material in this case. The loop will then be able to self-replicate into another location.

At this stage, we have observed no significant difference in the persistence of the two different genes over evolutionary time between loops that were carrying the sex gene and the ones carrying the dissolver gene. This could mean that the sex gene persisted in the original F-sexyloop mainly due to the selective pressure to be able to detach sex connections in order to survive and reproduce. This is slightly unexpected as previous results suggested that sex has a strong effect during the initial stages of the evolutionary process, increasing the diversity and accelerating evolution. However, more experiments are necessary in order to confirm these results.

4.4 Trade-off between evolvability and system level functionality

Cooperation between self-replicators to achieve system level properties like CHOPR, will require an understanding of reliable mechanisms to generate higher level fitness in populations of interacting components. Artificial genetic regulatory networks (GRNs) provide another model to study the emergence of higher level functionality and cooperation in populations of self-replicators. In biology one finds higher-level individuals, i.e. clonal multicellular populations (each cell with its own controller GRN) grown from the replication of single cell – just as the cells in a human body all descend from a single zygote (fertilised egg cell) — in which the cells take on different roles in a self-organised division of labour, where the entire organisation is viable at every stage in its development [7]. Evolution has given raise to these complex living systems in a way that rewards fitness of the higher level organisation at the expense of freedom at the lower level [18, 84]. We have developed formal measures to detect the emergence of a higher level in systems where no explicit fitness function is used to drive this emergence, and use

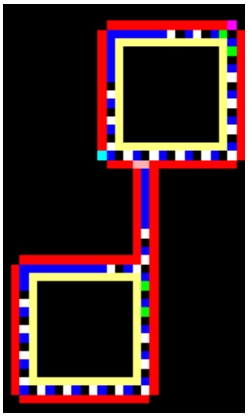

Figure 8: *Example of a loop connected to another one. The bottom loop does not have a sex gene to dissolve the blocker in order to transfer its genes so it will remain attached to the other one until another loop destroys it.*

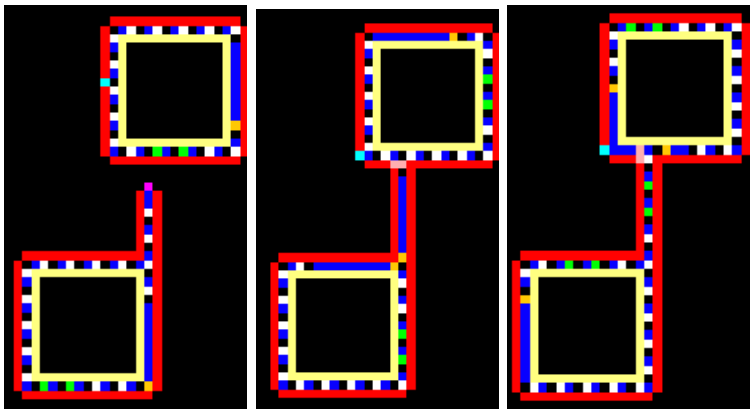

Figure 9: *Example of a loop (bottom) growing its arm towards another loop then creating a connection between the two. This loop has a sex gene (yellow) allowing it to dissolve the blocker and transfer its genes into the other one.*

of these measure suggests some hints of the emergence of higher level regulation of individual replicator dynamics in a GRN-model [16]. Furthermore, using evolution of artificial GRNs with a specially tailored, paramterisable fitness function, we showed that a trade-off exists with respect to evolvability between the benefit of higher level functionality and the computational complexity (difficulty) of carrying out the higher level functionality [17].

While the model systems considered in our foundational studies of self-replication are relatively simple, their behaviour at the individual and group levels proves quite complex and not easy to predict. Harnessing self-replicators in computational autopoiesis will require a sophisticated understanding of the theory, properties, and potential of self-replicators, an endeavour for which we and other researchers have only made a modest start.

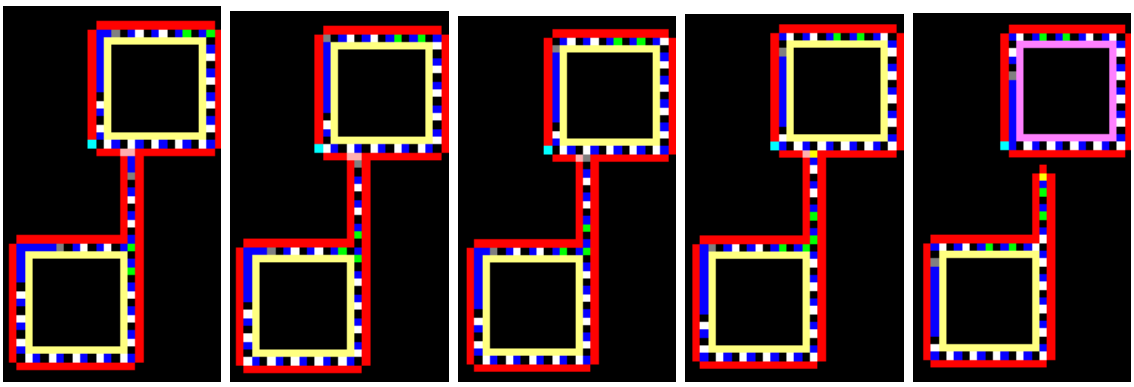

Figure 10: *Example of a loop (bottom) connected to another one. The bottom loop has a dissolver gene (grey) allowing it to detach from the other loop and dissolve its arm. There is no transfer of genetic material in this case.*

5 Conclusion

For the **biological computing** approach we have further considered formalisations of autopoiesis for the developing of an autopoietic form of computing, concluding on need for more rigorous non-reductionist ones to progress further. However, based upon general autopoietic properties, we have considered implementability, computability and potential implications. This has included consideration of the potential hardware requirements for our autopoietic form of computing, which may prove a greater limiting factor than the lack of suitable formalisations. Simply put, modern computing bears as much relation to biological computing as the ordered world of sudoku does to the statistical chaos of quantum mechanics [137].

Our **natural computing** approach presents a vision of autopoietic computing, at this stage focusing on the what, rather than the how, because we seek inspiration and not imitation. Natural computing when done well is inspiration using the **principles** which nature has demonstrated to be successful design strategies. For example, in the early days of mechanised flight the best designs were not the ornithopters, which most completely imitated birds, but the fixed-wing craft that used the principle of aerofoil cross-section in their wings [4].

Regarding integration with our Software Ecosystem, the self-CHOPR (Configuring, Healing, Optimising, Protecting, Replicating) behaviour of entities, of our natural-computing-based autopoiesis-inspired extension to autonomic computing, maps easily to the agents (services) of our Software Ecosystem. However, the integration of our biological-computing-based form of autopoietic computing will prove more challenging, because of the radically different way in which it will operate. While an over-arching **Ecosystem-Oriented Architecture** will be essential in providing a framework for not only evolution, but distributed evolution, its realisation will be fundamentally different. Thus, its realisation will be such that speaking about the integration of entities defined by temporally correlated structural coupling with that of agents (services) from our Software Ecosystem would be premature, and will very much depend on the formal computational definition of autopoietic computing to be created, which itself will depend on the formalisations of autopoiesis to be developed. For this reason, we instead considered their integration with business models that adopt **cellular organisation**, which would be synergistic to autopoietic software, in the same way that business ecosystems are complementary to our Software Ecosystem.

5.1 Future Work

We consider autopoiesis, gene expression and symbiosis expressions of the more fundamental concept of Interaction Computing. Gene expression computing refers to the ability to specify environments that, in turn, can generate software services in response to internal or external stimuli. Thus, gene expression computing aims to reproduce the ability of the biological cell to synthesise particular functional components such as enzymes in response to the needs of on-going metabolic processes or to signal transduction pathways from outside the cell. The signals to initiate the synthesis of such components are somehow *encoded* in the metabolic processes themselves, i.e. in prior biochemical reaction products. If the purpose of these reaction products is to trigger the synthesis of the enzymes, then the causal chain is linear and relatively straightforward. If, instead, the reaction products are needed for some other purpose and, *at the same time*, they encode the trigger for the expression of the enzyme(s), then we have a case of functional overloading that is the root of the concept of symbiotic computing. In any case, services generated in this manner should be considered *atomic* in some suitable sense and

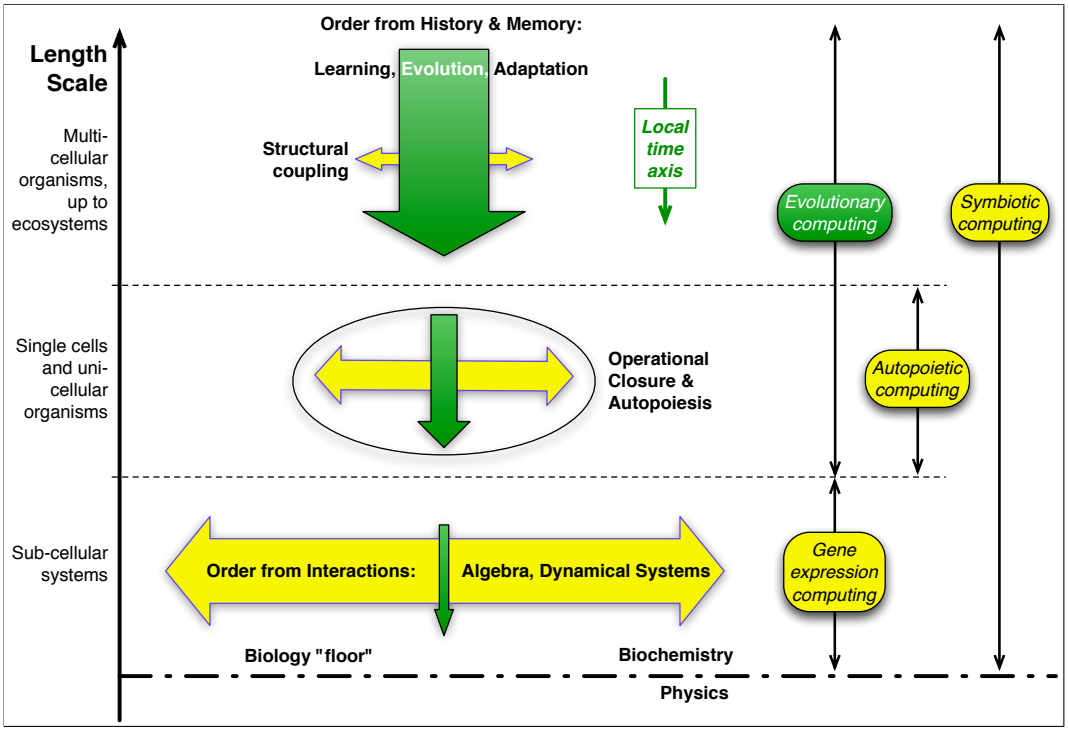

Figure 11: *Schematic showing the different relative importance of order construction mechanisms in biology, at different scales of description, and the corresponding relevance of different kinds of bio-computing. Green background indicates memory/history-dependent processes developing over time, yellow background indicates instantaneous dynamical interactions and reliance on a scale-invariant mathematical model of interaction computing.*

many of them need to be composed dynamically to produce the complex behaviour that we associate with a software service of average size and complexity. The dynamic composition process is similar in character to the sequences of biochemical reactions that make up metabolic and regulatory pathways. At an applied level symbiotic computing can be explained through the example of symbiotic security. As discussed in Schreckling et al. ([131]: 36), symbiotic security captures the balance between the encapsulation of a security function in a specialised security service with the interdependence between such a service and the service(s) it is meant to secure. In other words, in order to address the problem posed by the *ex post patching paradigm* in the development of security services and applications,[9] we felt it would be worth investigating a mode of generation of such paired services based on the integration of their most fundamental functions ex ante. This is similar to how the human immune system is inextricably integrated with the nervous and endocrine systems [24]. As a consequence, in symbiotic security a security service is assumed to be tightly coupled to the services it is meant to secure. Thus, neither can be executed by itself, both need the other service to do anything. Each service is functionally overloaded to perform its function *and* to trigger state transitions in the service it is coupled to. Since the concept of service applies at different scales, gene expression computing becomes a general framework that can support the more specific symbiotic computing, whose realisation requires the enforcement of additional suitable constraints.

Considering the integration of these three variants of Interaction Computing, once the interactions inside the cell, between cells, and between aggregates of cells have been translated into computer science rules subject to suitable algebraic constraints, we will be ready to build an

[9]The ex post patching paradigm does not develop new security services or applications but, rather, patches the corresponding applications after realising that they have a problem.

evolutionary framework around the whole construction and will be able to *launch* an autopoietic software ecosystem. Figure 11 (based on a similar figure in [28]: 119) shows the relative importance of different modelling perspectives at different scales. Although the time axis is technically a 3rd dimension, in this 2D diagram it is drawn parallel to the length scale axis; a 3D diagram would have been too difficult to draw and not very clear.

5.2 Socio-Technical Autopoietic Systems

In addition to the completion of the models presented, and beyond the cellularly organised (autopoietic) socio-economic systems we considered, we have observed that current **socio-technical systems** also demonstrate autopoietic behaviour, however, with the users performing much of this behaviour, e.g. a server is maintained in a functional state by a user compensating when errors occur. So, we can consider, that as a **framework of understanding**, autopoietic computing aims to move the necessary autopoietic behaviour for sustainable computing systems from the sociological to the technological.

References

[1] Freitas R. A. and Merkle R. C. *Kinematic Self-Replicating Machines*. Landes Bioscience, Georgetown, 2004.

[2] L.M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.

[3] S. Alters. *Biology: Understanding Life*. Jones & Bartlett Publishers, 1999.

[4] J. Anderson. *Introduction to Flight*. McGraw-Hill Professional, 2004.

[5] Michael Armbrust, Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley view of Cloud Computing. *University of California, Berkeley*, 2009. Available from: `http://d1smfj0g31qzek.cloudfront.net/abovetheclouds.pdf`.

[6] M. Arregoces and M. Portolani. *Data center fundamentals*. Cisco Press, 2003.

[7] Wallace Arthur. *The Origin of Animal Body Plans: A Study in Evolutionary Developmental Biology*. Cambridge Univ. Press, paperback edition, 2000.

[8] I. Baianu. Natural transformation models in molecular biology. In *Proceedings of SIAM & Society for Mathematical Biology Meeting*, pages 230–232. AMS (SIAM), 1983.

[9] I.C. Baianu. Natural transformations of organismic structures. *Bulletin of Mathematical Biology*, 42(3):431–446, 1980.

[10] IC Baianu. Quantum Genetics, Quantum Automata and Computation, 2004.

[11] R.D. Beer. Autopoiesis and cognition in the game of life. *Artificial Life*, 10(3):309–326, 2004.

[12] H. Bersini. Self-assertion versus self-recognition: A tribute to Francisco Varela. In *Proceedings of the First International Conference on Artificial Immune Systems (ICARIS-2002)*, pages 107–112. Citeseer, 2002.

[13] H. Bersini and FJ Varela. The immune learning mechanisms: reinforcement, recruitment and their applications. *Computing with Biological Metaphors*, 1, 1994.

[14] L. Blum. Computing over the reals: Where Turing meets Newton. *Notices of the American Mathematical Society*, 51(9):1024–1037, 2004.

[15] J.M. Bower. Looking for Newton: Realistic Modeling in Modern Biology. *Brains, Minds and Media, this volume*, 2005.

[16] M. Buck and C. L. Nehaniv. Looking for evidence of differentiation and cooperation: Natural measures for the study of evolution of multicellularity. *Advances in Complex Systems*, 12(3):255–271, 2009.

[17] M. Buck and C. L. Nehaniv. Evolution of cooperation and developmental constraints a GA-driven approach. In *Artificial Life XII, Odense, Denmark (19-23 August 2010)*, pages 437–444. MIT Press, 2010.

[18] Leo W. Buss. *The Evolution of Individuality*. Princeton University Press, 1987.

[19] D Chu and W K Ho. A Category Theoretical Argument Against the Possibility of Artificial Life: Robert Rosens Central Proof Revisited. *Artificial Life*, 12:117–134, 2006.

[20] A. Church. A set of postulates for the foundation of logic. *Annals of mathematics*, 33(2):346–366, 1932.

[21] F. H. Crick, L. Barnett, S. Brenner, and R. J. Watts-Tobin. General nature of the genetic code. *Nature*, 192:1227–1232, 1961.

[22] Krissi Danielson. Distinguishing Cloud Computing from Utility Computing. Technical report, ebizQ, 2008. Available from: `http://www.ebizq.net/blogs/saasweek/2008/03/distinguishing_cloud_computing/`.

[23] L. De Castro. *Fundamentals of Natural Computing: Basic Concepts, Algorithms, And Applications*. CRC Press, 2006.

[24] L N De Castro and J Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, London, 1 edition, November 2002.

[25] C. Delisi and D.M. Crothers. Prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences of the United States of America*, 68(11):2682, 1971.

[26] P Dini, G Briscoe, I Van Leeuwen, A J Munro, and S Lain. *D1.3: Biological Design Patterns of Autopoietic Behaviour in Digital Ecosystems*. OPAALS Deliverable, European Commission, http://files.opaals.eu/OPAALS/Year_3_Deliverables/WP01/D1.3.pdf, 2009.

[27] P Dini, A Egri-Nagy, C L Nehaniv, M J Schilstra, I Van Leeuwen, A J Munro, and S Lain. *D1.4: Mathematical Models of Gene Expression Computing*. OPAALS Deliverable, European Commission, 2010. Available from: `http://files.opaals.eu/OPAALS/Year_4_Deliverables/WP01/`.

[28] P Dini, D Schreckling, and G Horváth. Algebraic and Categorical Framework for Interaction Computing and Symbiotic Security. In E Altman, P Dini, D Miorandi, and D Schreckling, editors, *Paradigms for Biologically-Inspired Autonomic Networks and Services: The BIONETS Project eBook*, 2010. Available from: `http://www.bionets.eu`.

[29] C. Dwyer. Computer-aided design for DNA self-assembly: process and applications. In *IEEE/ACM International Conference on Computer-Aided Design, 2005. ICCAD-2005*, pages 662–667, 2005.

[30] E. Eberbach. Toward a theory of evolutionary computation. *BioSystems*, 82(1):1–19, 2005.

[31] D. Faulhammer, A.R. Cukras, R.J. Lipton, and L.F. Landweber. Molecular computation: RNA solutions to chess problems. *Proceedings of the National Academy of Sciences of the United States of America*, 97(4):1385, 2000.

[32] R.P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, 1982.

[33] R.P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6):507–531, 1986.

[34] N. Forbes. *Imitation of Life: How Biology Is Inspiring Computing*. MIT Press, 2004.

[35] Tom Foremski. Sun services CTO says utility computing acceptance is slow going. Technical report, ZDNet, CBS Interactive, 2006. Available from: `http://blogs.zdnet.com/Foremski/?p=33`.

[36] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud Computing and Grid Computing 360-degree compared. In *Grid Computing Environments Workshop*, pages 1–10, 2008.

[37] A.G. Ganek and T.A. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003.

[38] Gartner. Cloud Computing will be as influential as e-business. Technical report, Gartner, 2008. Available from: `http://www.gartner.com/it/page.jsp?id=707508`.

[39] Praising Gaw. What's the difference between Cloud Computing and SaaS? Technical report, Proofpoint, 2008. Available from: `http://blog.fortiva.com/fortivablog/2008/05/what-is-the-dif.html`.

[40] N. Gershenfeld and I.L. Chuang. Quantum computing with molecules. *Scientific American*, 278(6):66–71, 1998.

[41] Galen Gruman and Eric Knorr. What Cloud Computing really means. Technical report, InfoWorld Inc., 2008. Available from: `http://www.infoworld.com/article/08/04/07/15FE-cloud-computing-reality_1.html`.

[42] H. L. Hartwell, L. Hood, M. L. Goldberg, A. E. Reynolds, M. L. Silver, and Veres R.C. *Genetics: From Genes to Genomes*. McGraw-Hill, 3rd edition, 2008.

[43] T. Haynes, D. Knisley, E. Seier, and Y. Zou. A quantitative analysis of secondary RNA structure using domination based parameters on trees. *BMC bioinformatics*, 7(1):108, 2006.

[44] Mark Haynie. Enterprise cloud services: Deriving business value from Cloud Computing. Technical report, Micro Focus, 2009.

[45] T Head. Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49:737–759, 1987.

[46] C.V. Henkel and J.N. Kok. Towards evolutionary DNA computing. *Mechanisms, Symbols, and Models Underlying Cognition*, pages 242–257, 2005.

[47] K Henrick and JM Thornton. Trends in biochemical sciences; pqs: a protein quaternary structure file server. *Trends Biochem Sci*, 23(9):358–361, 1998.

[48] S.R. Holbrook. RNA structure: the long and the short of it. *Current opinion in structural biology*, 15(3):302–308, 2005.

[49] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-wesley Reading, MA, 1979.

[50] R. Hughes and T. Heinrichs. A quantum information science and technology roadmap. *Quantum*, 2:12, 2003.

[51] N.K. Jerne. Towards a network theory of the immune system. In *Annales d'immunologie*, volume 125, page 373, 1974.

[52] L. Kari. DNA computing: arrival of biological mathematics. *The Mathematical Intelligencer*, 19(2):9–22, 1997.

[53] H. Kawamoto. Autopoiesis: The Third Generation System. *Seido-sha Publishers*, 1995.

[54] H. Kawamoto. *The Extension of Autopoiesis*. Seido-sha Publishers, 2000.

[55] J.O. Kephart. Research challenges of autonomic computing. In *27th International Conference on Software Engineering, 2005. ICSE 2005. Proceedings*, pages 15–22, 2005.

[56] J.O. Kephart and D.M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[57] P.S. Klosterman, D.K. Hendrix, M. Tamura, S.R. Holbrook, and S.E. Brenner. Three-dimensional motifs from the SCOR, structural classification of RNA database: extruded strands, base triples, tetraloops and U-turns. *Nucleic acids research*, 32(8):2342, 2004.

[58] P.S. Klosterman, M. Tamura, S.R. Holbrook, and S.E. Brenner. SCOR: a structural classification of RNA database. *Nucleic Acids Research*, 30(1):392, 2002.

[59] G. Kneer and A. Nassehi. *Niklas Luhmanns Theorie sozialer Systeme: Eine Einführung*. W. Fink, 1994.

[60] T.S. Kuhn. The function of dogma in scientific research. *Scientific change*, pages 347–369, 1963.

[61] L Lamport. The Temporal Logic of Actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994. Available from: `citeseer.ist.psu.edu/lamport94temporal.html`.

[62] L. Lamport. *Specifying systems: The TLA+ language and tools for hardware and software engineers*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2002.

[63] C. Landauer and K.L. Bellman. Reflective Infrastructure for Autonomous Systems. In *Proc. EMCSR*, pages 671–676, 2000.

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

[65] E. Lawrence. *Henderson's dictionary of biological terms*. Pearson Education, 2005.

[66] S.H. Lee, J. Kim, FC Park, M. Kim, and JE Bobrow. Newton-type algorithms for dynamics-based robot movement optimization. *IEEE Transactions on robotics*, 21(4):657–667, 2005.

[67] J.C. Letelier, G. Marín, and J. Mpodozis. Computing With Autopoietic Systems. *Soft Computing and Industry Applications*, 2002.

[68] Soto-Andrade J. Abarzua F. G. Cornish-Bowden A. Cardenas M. L. Letelier, J.C. Metabolic closure in (m,r) systems. *Proc. 9th Int. Conf. Simulation and Synthesis of Living Systems (ALIFE9)*, pages 450–461, 2004.

[69] B. J MacLennan. Continuous information representation and processing in natural and artificial neural networks. Technical report, Tech. Report. UT-CS-03, 2003.

[70] B J MacLennan. Super-Turing or non-Turing? Invited presentation. In *Workshop Future Trends in Hypercomputation (Trends' 06)*, pages 11–13, 2006.

[71] BJ MacLennan. Transcending Turing computability. *Minds and Machines*, 13(1):3–22, 2003.

[72] BJ MacLennan. Natural computation and non-Turing models of computation. *Theoretical Computer Science*, 317(1-3):115–145, 2004.

[73] G. Manganaro and J.P. de Gyvez. DNA computing based on chaos. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97): April 13-16, 1997, University Place Hotel, Indianapolis, IN USA*, page 255. IEEE, 1997.

[74] P Marrow. Nature-inspired computing technology and applications. *BT Technology Journal*, 18:13–23, 2000.

[75] J.A. Mathews. *Dragon Multinational: a new model for global growth*. Oxford University Press, USA, 2002.

[76] H.R. Maturana and F.J. Varela. *Autopoiesis and cognition: The realization of the living*. Springer, 1980.

[77] P. Matzinger. The danger model: a renewed sense of self. *Science's STKE*, 296(5566):301, 2002.

[78] J. Maynard Smith and E. Szathmáry. *The Major Transitions in Evolution*. W. H. Freeman, 1995.

[79] John Maynard Smith. *The Evolution of Sex*. Cambridge Univ. Press, 1978.

[80] B McMullin. Computational autopoiesis: The original algorithm. Technical report, Working paper 97-01-001, Santa Fe Institute, Santa Fe, NM 87501, USA, 1997. Available from: `http://www.santafe.edu/sfi/publications/wpabstract/199701001`.

[81] B. McMullin. John von Neumann and the evolutionary growth of complexity: Looking backwards, looking forwards. *Artificial Life*, 6:347–361, 2000.

[82] B. McMullin. Thirty years of computational autopoiesis: A review. *Artificial Life*, 10(3):277–295, 2004.

[83] B. McMullin and F.J. Varela. Rediscovering computational autopoiesis. In *Fourth European Conference on Artificial Life*, pages 38–47. MIT Press, 1997.

[84] R. E. Michod. *Darwinian Dynamics: Evolutionary Transitions in Fitness and Individuality*. Princeton University Press, 1999.

[85] J.G. Miller. Living systems theory. *McGrow Hill*, 1978.

[86] J.G. Miller and J.L. Miller. Introduction: the nature of living systems. *Behavioral science*, 35(3):157–163, 1990.

[87] AP Mills Jr, M. Turberfield, AJ Turberfield, B. Yurke, and PM Platzman. Experimental aspects of DNA neural network computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 5(1):10–18, 2001.

[88] B. Mitchell. *Theory of categories*. Academic Pr, 1965.

[89] N. Nanas and A. De Roeck. Autopoiesis, the immune system, and adaptive information filtering. *Natural Computing*, 8(2):387–427, 2009.

[90] C. L. Nehaniv. Self-replication, evolvability, and asynchronicity in stochastic worlds. In Oleg B. Lupanov, Oktay M. Kasim-Zade, Alexander V. Chaskin, and Kathleen Steinhöfel, editors, *Proc. 3rd Symposium on Stochastic Algorithms, Foundations and Applications (Symposium Stochastische Algorithmen, Grundlagen und Anwendungen) - SAGA'05, Moscow, Russia, October 20-22, 2005*, volume 3777, pages 126–169, 2005. Available from: `http://homepages.stca.herts.ac.uk/~comqcln//nehaniv-SAGA05-withcorrections.pdf`.

[91] C. L. Nehaniv and G. P. Wagner. The right stuff: Appropriate mathematics for evolutionary and developmental biology (editors' introduction to the special issue). *Artificial Life*, 6(1):1–2, 2000.

[92] A. Newman, A. Steinberg, and J. Thomas. *Enterprise 2.0 Implementation*. McGraw-Hill Osborne Media, 2008.

[93] G. Nicolis and I. Prigogine. *Exploring complexity: an introduction*. Freeman, 1989.

[94] D. Noble. *The Music of Life*. Oxford University Press Oxford, 2006.

[95] D. Noble. Genes and causation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1878):3001, 2008.

[96] T. Nomura. An Attempt for Description of Quasi-Autopoietic Systems Using Metabolism-Repair Systems. *Proc. the Fourth European Conference on Artificial Life*, pages 48–56, 1997.

[97] T. Nomura. Formal description of autopoiesis for analytic models of life and social systems. In *Proceedings of the eighth international conference on Artificial life*, pages 15–18. MIT Press, 2002.

[98] T Nomura. Category Theoretical Formalization of Autopoieis from Perspective of Distinction between Organization and Structure. *Explorations in the Complexity of Possible Life: Abstracting and Synthesizing the Principles of Living Systems*, 2006.

[99] T. Nomura. Category theoretical distinction between autopoiesis and (M, R) systems. *Lecture Notes in Computer Science*, 4648:465, 2007.

[100] Andrew Orlowski. The Cell chip - how will MS and Intel face the music? Technical report, The Register, 2005. Available from: `http://www.theregister.co.uk/2005/02/03/cell_analysis_part_two/`.

[101] Nicolas Oros and C. L. Nehaniv. Sexyloop: Self-reproduction, evolution and sex in cellular automata. In *The First IEEE Symposium on Artificial Life (April 1-5, 2007, Hawaii, USA)*, pages 130–138, 2007.

[102] Nicolas Oros and C. L. Nehaniv. Dude, where is my sex gene? — persistence of sex over evolutionary time in cellular automata. In *Proc. 2nd International IEEE Symposium on Artificial Life (Nashville, Tennessee, USA - 30 March-2 April 2009)*, pages 1–8, 2009.

[103] L. Pauling and R.B. Corey. Atomic coordinates and structure factors for two helical configurations of polypeptide chains. *Proceedings of the National Academy of Sciences of the United States of America*, 37(5):235, 1951.

[104] L. Pauling and R.B. Corey. Configurations of polypeptide chains with favored orientations around single bonds: two new pleated sheets. *Proceedings of the National Academy of Sciences of the United States of America*, 37(11):729, 1951.

[105] L. Pauling and R.B. Corey. Structure of the nucleic acids. *Nature*, 171(346), 1953.

[106] Linus Pauling and Carl Niemann. The structure of proteins. *J. Am. Chem. Soc.*, 61(7):1860–1867, 1939.

[107] A.S. Perelson and G.F. Oster. Theoretical studies of clonal selection: minimal antibody repertoire size and reliability of self-non-self discrimination. *Journal of Theoretical Biology*, 81(4):645–670, 1979.

[108] U. Pesevanto. An implementation of von neumann's self-reproducing machine. *Artificial Life*, 2:337–354, 1995.

[109] H. Pietschmann. *Das Ende des naturwissenschaftlichen Zeitalters*. P. Zsolnay, 1980.

[110] JM Pipas and JE McMahon. Method for predicting RNA secondary structure. *Proceedings of the National Academy of Sciences*, 72(6):2017, 1975.

[111] K.R. Popper. Logik der Forschung. Vienna. *The logic of Scientific Discovery*, 1934.

[112] Rashevsky. Topology and life: in search of general mathematical principles in biology and sociology. *Bull. Math. Biophys.*, 16:317–348, 1954.

[113] N. Rashevsky. Mathematical biophysics: physico-mathematical foundations of biology. *Bull. Amer. Math. Soc. 45 (1939), 223-224. DOI: 10.1090/S0002-9904-1939-06963-2 PII: S*, 2(9904):06963–2, 1939.

[114] N Rashevsky. *Mathematical Biophysics and Physico-Mathematical Foundations of Biology Vol II*. Dover, New York, 1960.

[115] N Rashevsky. Models and mathematical principles in biology. In Waterman/Morowitz (Eds.), editor, *Theoretical and mathematical biology*, pages 36–53. New York, Blaisdell, 1965.

[116] T. S. Ray. An approach to the synthesis of life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II. Volume X of SFI Studies in the Sciences of Complexity*, pages 371–408. Addison-Wesley, Redwood City, CA, 1992.

[117] John Rhodes. *Applications of Automata Theory and Algebra via the Mathematical Theory of Complexity to Finite-State Physics, Biology, Philosophy, and Games.* World Scientific Press, 2010.

[118] R Rosen. A Relational Theory of Biological Systems. *Bulletin of Mathematical Biophysics*, 20:245–260, 1958.

[119] R Rosen. The Representation of Biological Systems from the Standpoint of the Theory of Categories. *Bulletin of Mathematical Biophysics*, 20:317–341, 1958.

[120] R. Rosen. Abstract biological systems as sequential machines. *Bulletin of Mathematical Biology*, 26(2):103–111, 1964.

[121] R. Rosen. On analogous systems. *Bulletin of Mathematical Biology*, 30(3):481–492, 1968.

[122] R. Rosen. Some realizations of (M, R)-systems and their interpretation. *Bulletin of Mathematical Biology*, 33(3):303–319, 1971.

[123] R. Rosen. Some relational cell models: The metabolism-repair systems. *Foundations of Mathematical Biology*, 2:217–253, 1972.

[124] R. Rosen. *Life itself: A comprehensive inquiry into the nature, origin, and fabrication of life.* Columbia Univ Pr, 1991.

[125] R. Rosen. *Essays on life itself.* Columbia Univ Pr, 2000.

[126] OW Sacks. Scotoma: Forgetting and neglect in science. *Hidden Histories of Science. New York, New York Review of Books*, pages 141–187, 1996.

[127] S.N. Salthe. *Evolving hierarchical systems: their structure and representation.* Columbia Univ Pr, 1985.

[128] H. Sayama. A new structurally dissolvable self-reproducing loop evolving in a simple cellular automata space. *Artificial Life*, 5(4):343–365, 1999.

[129] Jessie Scanlon and Brad Wieners. The internet cloud. Technical report, The Industry Standard, 1999. Available from: `http://www.thestandard.com/article/0,1902,5466,00.html`.

[130] L.W. Schindler. *Understanding the immune system.* DIANE Publishing, 1991.

[131] D Schreckling, M Brunato, P Dini, L Dóra, A Faschingbauer, J Golic, G Horváth, F Martinelli, and M Petrocchi. *D4.6: Security in BIONETS.* BIONETS Deliverable, European Commission, 2009. Available from: `http://www.bionets.eu`.

[132] C.E. Shannon. A universal Turing machine with two internal states. *Automata studies*, pages 129–153, 1956.

[133] P.L. Simeonov. Integral biomathics: A post-Newtonian view into the logos of bios. *Progress in Biophysics and Molecular Biology*, 103(1):1–156, 2010.

[134] L. Sompayrac. *How the immune system works.* Wiley-Blackwell, 2003.

[135] R. Stadler. Molecular, chemical, and organic computing. *Communications of the ACM*, 50(9):45, 2007.

[136] B. Stalbaum. Toward Autopoietic Database. http://www.c5corp.com/research/autopoieticdatabase.shtml, C5 Corporation, 2005.

[137] Matthew Swabey. The future of computing? the spinnaker million processor computer. Technical report, University of Southampton, 2010.

[138] M. Tamura, D.K. Hendrix, P.S. Klosterman, N.R.B. Schimmelman, S.E. Brenner, and S.R. Holbrook. SCOR: Structural Classification of RNA, version 2.0. *Nucleic acids research*, 32(Database Issue):D182, 2004.

[139] D. Thirumalai, DK Klimov, and SA Woodson. Kinetic partitioning mechanism as a unifying theme in the folding of biomolecules. *Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta)*, 96(1):14–22, 1997.

[140] A Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:2:230–265, 1936. A correction, ibid, 43, 1937, pp. 544-546.

[141] AM Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(1):230, 1937.

[142] F. Varela and JC Letelier. Morphic resonance in silicon chips: An experimental test of the hypothesis of formative causation. *Skeptical Inquirer*, 12:298–300, 1988.

[143] F. J. Varela. *Principles of Biological Autonomy*. North Holland, 1979.

[144] F.J. Varela and A. Coutinho. Second generation immune networks. *Immunology Today*, 12(5):159–166, 1991.

[145] F.J. Varela, H.R. Maturana, and R. Uribe. Autopoiesis: the organization of living systems, its characterization and a model. *Currents in modern biology*, 5(4):187, 1974.

[146] NM Vaz and FJ Varela. Self and non-sense: an organism-centered approach to immunology. *Medical hypotheses*, 4(3):231–261, 1978.

[147] C Venter. *A life decoded*. Pengiun, 2007.

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

[149] Andreas Wagner. *Robustness and Evolvability in Living Systems*. Princeton Studies in Complexity, 2005.

[150] S. Washietl, I.L. Hofacker, and P.F. Stadler. Fast and reliable prediction of noncoding RNAs. *Proceedings of the National Academy of Sciences*, 102(7):2454–2459, 2000.

[151] M. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. CRC Press, 1995.

[152] J. D. Watson and F. H. C. Crick. A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

[153] Peter Wegner. Why interaction is more powerful than algorithms. *Communiations of the ACM*, 40(5):80–91, 1997.

[154] S.R. White, J.E. Hanson, I. Whalley, D.M. Chess, and J.O. Kephart. An architectural approach to autonomic computing. *Autonomic Computing*, pages 2–9, 2004.

[155] A.N. Whitehead. *Adventures of ideas*. Free Pr, 1933.

[156] C. Winter, A. Henschel, W.K. Kim, and M. Schroeder. SCOPPI: a structural classification of protein-protein interfaces. *Nucleic Acids Research*, 34(Database Issue):D310, 2006.

[157] P. Yin, S. Sahu, A. Turberfield, and J. Reif. Design of autonomous DNA cellular automata. *DNA Computing*, pages 399–416, 2006.

[158] P.T. Yin and S. AJ. Design of an autonomous DNA nanomechanical device capable of universal computation and universal translation motion. *DNA10, Springer-Verlag, LNCS*, 3384:426–444, 2005.

[159] Ayoola R. Yinusa. Study of inheritable mutations in the Von Neumann self-replicator using the Golly simulator. Master's thesis, School of Computer Science, University of Hertfordshire, U.K., 2010.

[160] M. Zuker. Computer prediction of RNA structure. *Methods in Enzymology*, 180:262–288, 1989.