Systemic Computation, where natural computation and new technologies come together

Erwan Le Martelot
Department of Electronic & Electrical Engineering
University College London
e.le_martelot@ucl.ac.uk

Accompanying notes for the 18th September 2008 UCL EE Dpt Presentation

ABSTRACT

Natural systems provide unique examples of computation in a form very different from contemporary computer architectures. Biology also demonstrates capabilities such as adaptation, fault-tolerance, self-repair and self-organisation that are becoming increasingly desirable for our technology. Such computational properties are of great value for the increasingly complex systems our technologies require and new computer architectures are popular, whether distributed computing (or multiprocessing), computer clustering and grid computing and even ubiquitous computing and speckled computing. Thus, computation is increasingly becoming more parallel, decentralised and distributed. However, while hugely complex computational systems will be soon feasible, their organisation and management is still the subject of research. Ubiquitous computing may enable computation anywhere, and bio-inspired models may enable improved capabilities such as reliability and fault-tolerance, but there has been no coherent architecture that combines both technologies. To address these issues and unify notions of biological computation and electronic computation, Systemic Computation (SC) was introduced as a suggestion of necessary features for a computer architecture compatible with current processors, yet designed to provide native support for common characteristics of biological processes. Systemic computation is Turing Complete; I present here its working principle, state of the art and crucial natural properties like fault-tolerance and self-repair. Then its implications for new technologies and applications are tackled highlighting the compatibility with current electronics hardware and the relevance to emerging technologies like sensor networks, FPGA, bio-inspired hardware and their applications.

INTRODUCTION

With the increasing performance, potential and complexity in machines and software, it has become increasingly difficult to ensure reliability in systems. Software regularly

crashes, top of the line robots break down on the wrong kind of ground, power distribution networks fail under unforeseen circumstances (Bentley 2004).
While the theory of computation is well understood through the concept of the Universal Turing Machine (Turing 1965), practical issues of architecture remain problematical for computer science and computer-based technologies. The technology of computation is increasingly becoming more parallel, decentralised and distributed and while hugely complex computational systems will soon be feasible, their organisation and management is still the subject of research (Kephart, Chess 2003).

Biological systems are of great interest to computer science and new technologies as examples of highly complex systems that perform tasks in parallel with no centralised method of control and showing capabilities such adaptation, self-repair and as self-organisation. We can state that natural processes are stochastic, asynchronous, parallel, homeostatic, continuous, robust, fault tolerant, autonomous, open-ended, distributed, approximate, embodied, complex, have circular causality and compute locally (Bentley 2007a). Such characteristics are not natively present in current conventional paradigms and models of natural processes that run on conventional computers have to include a simulation of some of these features. Reciprocally new architectures providing some of these features are still lacking a formalised paradigm and there has been no coherent architecture that combines both natural and electronic computations (Bentley 2007a).

Computational systems and new technologies now require a formalisation of computation providing the power and reliability of natural computation while maintaining full compatibility with modern hardware. To address this, (Bentley 2007a) introduced Systemic Computation (SC) as a suggestion of necessary features for a computer architecture compatible with current processors, memories, sensors yet designed to provide native support for common characteristics of biological processes.

I present here the state of the art of the ongoing research in SC, its current limits, its potential implications for new technologies and their applications.

BACKGROUND

Systemic Computation (SC) is not the only model of computation to emerge from studies of biology. The potential of biology had been discussed in the late 1940s by Von Neumann who dedicated some of his final work to automata and self-replicating machines (von Neumann 1966). Cellular automata have proven themselves to be a valuable approach to emergent, distributed computation (Wolfram 2002). Generalisations such as constrained generating procedures and collision-based computing provide new ways to design and analyse emergent computational phenomena (Holland 1998; Adamatzky 2001). Bio-inspired grammars and algorithms introduced notions of homeostasis (for example in artificial immune systems), fault-tolerance (as seen in embryonic hardware) and parallel stochastic learning, (for example in swarm intelligence and genetic algorithms) (Bentley 2007a; Fogel and Corne 2003).

New architectures are also popular, whether distributed computing (or multiprocessing), computer clustering and grid computing and even ubiquitous computing and speckled computing (Arvind and Wong 2004). Thus, computation is increasingly becoming more parallel, decentralised and distributed. However, while hugely complex computational systems will be soon feasible, their organisation and management is still the subject of research. Ubiquitous computing may enable computation anywhere, and bio-inspired models may enable improved capabilities such as reliability and fault-tolerance, but there has been no coherent architecture that combines both technologies. Indeed, these technologies appear incompatible – the computational overhead of most bio-inspired methods is prohibitive for the limited capabilities of ubiquitous devices.

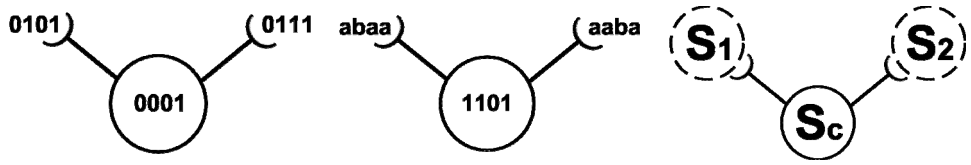
To unify notions of biological computation and electronic computation, (Bentley 2007a) introduced SC as a suggestion of necessary features for a computer architecture compatible with current processors, yet designed to provide native support for common characteristics of biological processes. Systemic Computation provides an alternative approach to computation. With SC, organisms and software programs now share a common definition of computation.

I now review the state of the art of SC, then discuss its integration into new technologies and some applications that would fit the SC paradigm.

REVIEW OF SYSTEMIC COMPUTATION

SC (Bentley 2007a) is a new model of computation and corresponding computer architecture based on a systemics world-view and supplemented by the incorporation of natural characteristics (previously listed). This approach stresses the importance of structure and interaction, supplementing traditional reductionist analysis with the recognition that circular causality, embodiment in environments and emergence of hierarchical organisations all play vital roles in natural systems. Systemic computation makes the following assertions:
-Everything is a system.
-Systems can be transformed but never destroyed.
-Systems may comprise or share other nested systems.
-Systems interact, and interaction between systems may cause transformation of those systems, where the nature of that transformation is determined by a contextual system.
-All systems can potentially act as context and affect the interactions of other systems, and all systems can potentially interact in some context.
-The transformation of systems is constrained by the scope of systems, and systems may have partial membership within the scope of a system.
-Computation is transformation.
In systemic computation, everything is a system, and computations arise from interactions between systems. Two systems can interact in the context of a third system. All systems can potentially act as contexts to determine the effect of interacting systems. A system is divided into three parts: two schemata and one kernel. These three parts can be used to hold anything (data, typing, etc.) in binary as shown in Figure 1(a). The kernel defines the result of two systems interacting in its context (and may also optionally hold data if it is interacting with another system). The two schemata define which subject systems may interact in this context as shown

in Figures 1(b) and 1(c). A system can also contain or be contained by other systems. This enables the notion of scope. Interactions can only occur between systems within the same scope. An SC program therefore comprises systems that are instantiated and positioned within a hierarchy (some inside each other). It thus defines an initial state from which the systems can then randomly interact, transforming each other through those interactions and following an emergent process rather than a deterministic algorithm. For full details see (Bentley 2007a) and (Le Martelot, Bentley and Lotto 2007a).



(a) Data system (b) Context system (c) Interaction

Fig. 1. 1(a): A system used primarily for data storage. The kernel (in the circle) and the two schemata (at the end of the two arms) hold data. 1(b): A system acting as a context. Its kernel defines the result of the interaction while its schemata define allowable interacting systems. 1(c): An interacting context. The contextual system Sc matches two appropriate systems S1 and S2 with its schemata and specifies the transformation resulting from their interaction as defined in its kernel.

Systemic Computation has been used to model several bio-inspired systems, showing natural crucial properties.

In (Le Martelot, Bentley, and Lotto 2007a) the first complete SC platform was presented, including programming language, compiler and virtual machine. Then it was used to create a genetic algorithm (GA) for the Travelling Salesman Problem (TSP). The TSP is a classic problem in the fields of Computational Complexity Theory and Evolutionary Computation. The problem is, given a number of cities and the distance from one city to another, find the shortest round-trip route that visits each city exactly once and then returns to the city of origin. The work explored how simply a GA can be set up and how SC enables self-adaptation with the minimum of additional code.

The GA was later reused in (Le Martelot, Bentley, and Lotto 2008a) to demonstrate that SC computing is crash-proof: SC programs naturally show fault tolerance and can easily integrate self-maintenance.

Also (Le Martelot, Bentley, and Lotto 2007b) presented an artificial neural network (ANN) model implemented in SC and showed that the SC implementation naturally involves flexible neural structure without reducing performance.

Finally (Le Martelot, Bentley, and Lotto 2008b) showed using systemic computation how to create an artificial organism, a program with metabolism that eats data, expels waste, clusters cells based on the nature of its food and emits danger signals suitable for an artificial immune system. The implementation was tested by application to a

standard machine learning set and showed excellent abilities to recognise anomalies in its diet.

SYSTEMIC COMPUTATION, NEW TECHNOLOGIES AND APPLICATIONS

The current limits

Systemic Computation has been designed to run bio-inspired systems natively, addressing the inappropriateness of standard computers for increasingly complex systems. Therefore even though the virtual machine enables us to work with the platform and models, it remains limited by the underlying conventional architecture.

Two major limits are imposed by such simulation: the execution speed and the natural features the platform can provide.

The first limit could be partially addressed by using computer clustering (grid computing), where each computer stands for one or more systems. The conventional client-server paradigm would need to be abandoned as SC does not centralise any data and each computer would then run a peer-to-peer SC platform.

However, although speed is an important aspect in computation, a far more important aspect for SC is the ability to provide natural features natively. If the aforementioned platform enabled the demonstration of fault-tolerance and self-repair ability, it still relies on the PC host's brittle architecture and operating system compared to the robustness of natural computation. Grid computing and any form of clustering relying on conventional computers are also facing the same issue.

Towards new implementations

SC implies a distributed, stochastic architecture, involving natural characteristics. (Bentley 2007b) suggested that a physical implementation of this architecture could be achieved through the use of wireless devices produced for sensor networks. The aim of this work was to provide an architecture for a useful, fault-tolerant and autonomous computer that could exploit all the features of sensor networks, providing benefits for our understanding of "natural computing" and robust wireless networking.

While still at the design stage, such biological computer architecture suggests that SC can potentially fit an impact the development of new technologies and their applications. Examples of such new technologies include advances such as speckled computing (Arvind, Wong 2004) which may enable "spray on" computer coatings. Predictions of the logical extension to the Internet – wireless dynamic networks of computers in almost every device around us – look set to become reality in a decade (Gershenfeld 2000). The technology of computation is increasingly becoming more parallel, decentralised and distributed, yet, such trends towards distributed and ubiquitous computing come at the cost of control and reliability. Here SC offers a novel way to control computation defining rules working locally and providing emergent global properties such as fault-tolerance and self-repair.

In conventional distributed computing, most of the work is adopting a traditional Client-Server interaction paradigm. However, for ubiquitous and pervasive computing applications this paradigm could be inadequate in particular when context awareness, auto-adaptive and emergence functionalities are required (Gaber 2007). Sensor networks are an example of distributed computing using a data-centric paradigm. In such case the sensor providing the data is irrelevant as only the data matters. A request for a temperature measure in an area is no longer a request to a specific device with an address but to an autonomous network able to process the request.
Similarly SC is also a data-based computational paradigm. Systems interact based on their content (the schemata define the interacting systems (Bentley 2007a)), they can be replicated to deal with the potential failure of others (programming with redundancy (Le Martelot, Bentley, Lotto 2008a)) and the outcome of a program is a behaviour emerging from a stochastic massively parallel interaction (Le Martelot, Bentley, Lotto 2007a).

However, sensors are not the only platforms that could benefit from SC and match such data-centric approach.

In SC, any system can be a context, and the interactions defined by contexts can change over computation, therefore a systemic computer requires on-line reprogrammable devices. The reconfigurable computing paradigm combines software flexibility and hardware high performance into devices commonly called reconfigurable hardware, one of the most powerful being known as the Field-Programmable Gate Array (FPGA). The FPGA is a programmable logic component that can be reconfigured into different circuits (the configuration stored on its own internal memory). An FPGA can be configured into a computer processor or any other logic-based electronic circuit, with the ability to reconfigure its own circuits as it runs (FPGAs range from tens of thousands to several million logic gates equivalent). If reconfigured as a massively parallel architecture where the routing between the logic blocks is dynamically changed and the computation reprogrammed on-line, the result is a platform suitable for SC with however a limit in the amount of systems defined by the amount of inner logic blocks. The ideal FPGA would have hundreds of millions of logic blocks, with for each the possibility to connect anywhere in 3 dimensions like the brain. Unfortunately, such technology is not available to date.

Similar limitations also apply to wireless devices, which are limited by range, bandwidth and interference (and also by the fact that most current wireless devices are little more than basic microcontrollers with only a few kilobytes of FLASH memory). There would therefore also be a restriction in the amount of systems, unless the previous issues are resolved, suggesting a technology not yet available either.
Both FPGAs and wireless sensor networks offer crucial features but also limitations. FPGAs would be more suitable for desktop computing, whereas sensor network are suitable for ubiquitous and pervasive computing.

Other specific hardware have been designed to integrate natural features like in the POEtic project (Tempesti et al. 2002), aiming at creating a platform organised with a similar hierarchy as found in biological systems, and capable of implementing systems inspired by all the three major axes (phylogenesis, ontogenesis, and epigenesis) of bio-inspiration in digital hardware. An improved version of this platform, currently under development, is known as Perplexus (Upegui et al. 2007).

The Embryonics project (Tempesti et al. 2007) is also another project that investigated such hardware for reliability, involving self-replicating hardware.

Some potential applications

Ideas such as the Internet of Things, Smartdust, and Ambient Intelligence based on a wireless network between objects or sensors, could also be implemented and benefit from such computation with appropriate hardware. In the same way, phone communications, radio transmission, digital audio players' music would be transmitted from everywhere to everywhere across the "everyware" (Greenfield, Adam 2006).

From a bio-inspired computing view, a SC hardware computer could provide fast bio-inspired computation (e.g. for NP problems) (Bentley 2007a). Indeed, in computational modelling and analysis for the biosciences, even our supercomputers struggle to model or analyse biological behaviour – Moore's Law cannot keep up with the data produced by advances in DNA synthesis and sequencing productivity (Carlson 2003). Large search space problems (NP-problems, e.g. the TSP mentioned above) would also greatly benefit from such new architecture. For such problems, even the fastest computer can need an amount of time in the scale of the age of the universe to solve rather small problems with brutal force (exploring all potential solutions). The alternatives to brute force algorithms commonly involve bio-inspired exploration techniques for which traditional computation is not best suited. The systemic computer however has been specifically designed for such techniques, incorporating natively all the characteristics they need to function where conventional computers rely on emulation of these properties.

Bio-inspired computation with SC can also offer good efficiency for data flow analysis and anomaly detection (e.g. Le Martelot, Bentley, and Lotto 2008b). Secure networks and reliable data transmission or control protocols (e.g., cognitive radio) are thus another field of application where bio-inspired computation implanted using SC can provide significant benefits.

Other applications and suitable platforms for SC include technologies closely linked to biological computation, like nanotechnologies, or self-assembling devices evolving towards artificial molecular-level devices and machines (Balzani et al. 2002), computation using biology or natural systems like DNA-computing (Adleman 1994), bacteria-computing (Gardner 2000), or reaction-diffusion computing (Adamatzky et al. 2002).
Similarly, the world of robotics, such as swarm robotics, using emerging collective behaviour like in an insect or an ant colony is another field where the SC paradigm would fit.

Finally, SC offers a clearly defined framework for complex systems; it is anticipated that it will improve the analysis and the understanding of complex systems (Bentley 2007a). Work is undergoing in this direction to provide a visualisation of computation over time, allowing computation to be watched over the evolution of a complex system, thus providing a crucial tool for the analysis of emerging patterns.

CONCLUSION

Biological computation has many significant characteristics that help give it desirable properties. This paper reviewed Systemic Computation, a novel computation paradigm that incorporates those characteristics and suggests a non-von Neumann architecture compatible with conventional hardware and able to support biological characteristics natively.

With SC, programs possess native natural rules leading to natural properties like fault-tolerance and straightforward self-maintaining programs. Also bio-inspired systems can be designed with minimal code while providing competitive results compared to standard approaches.

We first reviewed SC and the limits of the current platform, opening the way to its the implications for new hardware and technologies, highlighting the compatibility with current electronics hardware, whether for desktop or ubiquitous computing. We also reviewed some applicative potential of SC from biological computation or bio-inspired computation to nanotechnologies. Some daily applications like the internet or network security were also commented considering the potential impact of such technological progress, opening a way to a significant renewal of their integration in our daily life.

REFERENCES

Adamatzky, A., Computing in Nonlinear Media and Automata Collectives (Bristol, UK: Institute of Physics Publishing, 2001).

Adamatzky, A., and De Lacy Costello, B., Experimental logical gates in a reaction-diffusion medium: The xor gate and beyond, Physical review. E, Statistical, nonlinear, and soft matter physics, 2002

Adleman, L. M., Molecular Computation Of Solutions To Combinatorial Problems, Science 266 1021–1024,1994

Arvind, D.K., Wong, K.J., Speckled Computing: Disruptive Technology for Networked Information Appliances, Proc. of the IEEE International Symposium on Consumer Electronics (ISCE'04), Edinburgh, UK, 2004, 219-223.

Balzani, V., Credi, A., and Venturi, M., Controlled disassembling of self-assembling systems: Toward artificial molecular-level devices and machines, PNAS 2002;99;4814-4817; 2002

Bentley, P.J., Climbing Through Complexity Ceilings, Invited presentation at Symposium on Distributed Form: Network Practice, Berkeley, USA., 2004.

Bentley, P.J., Systemic computation, A Model of Interacting Systems with Natural Characteristics, International journal of parallel, emergent and distributed systems, 22, 2007, 103-121.

Bentley, P. J. (2007), Designing Biological Computers: Systemic Computation and Sensor Networks. Submitted chapter for Bio-Inspired Computing and Communication, notes arising from BIOWIRE 2007: A workshop on Bioinspired design of networks, in particular wireless networks and self-organizing properties of biological networks.

Carlson, R., The Pace and Proliferation of Biological Technologies. Biosecurity and Bioterrorism: Biodefense Strategy, Practice, and Science 1, 3 (August 2003).

Fogel, G. B., Corne, D. W. (Eds) (2003) Evolutionary Computation in Bioinformatics. Morgan Kaufmann Pub. ISBN: 1558607978.

Gaber, J., "Spontaneous Emergence Model for Pervasive Environments," Globecom Workshops, 2007 IEEE , vol., no., pp.1-4, 26-30 Nov. 2007

Gardner, T. S., Cantor, C. R., and Collins, J. J., Construction of a genetic toggle switch in Escherichia coli, Nature 403, 339--42, 2000

Gershenfeld, N., When Things Start to Think. Henry Holt and Company, New York (2000).

Greenfield, A., (2006). Everyware: the dawning age of ubiquitous computing. New Riders, p.11-12. ISBN 0321384016.

Holland, J. H., Emergence. From Chaos to Order (Oxford, UK: Oxford University Press, 1998).

Kephart, J., Chess, D., “The Vision of Autonomic Computing”, Computer Magazine, 0018- 9162/03 IEEE, 41-50 (2003).

Le Martelot, E., Bentley, P. J., and Lotto, R. B. (2007) Exploiting Natural Asynchrony and Local Knowledge within Systemic Computation to Enable Generic Neural Structures. In Proc of 2nd International Workshop on Natural Computing (IWNC 2007), December 10-13, 2007, Nagoya University, Nagoya, Japan.

Le Martelot, E., Bentley, P. J., and Lotto, R. B. (2007) A Systemic Computation Platform for the Modelling and Analysis of Processes with Natural Characteristics. In Proc of 9th Genetic and Evolutionary Computation Conference (GECCO 2007) Workshop: Evolution of Natural and Artificial Systems - Metaphors and Analogies in Single and Multi-Objective Problems, July 7-11, 2007, University College London, London, U.K.

Le Martelot, E., Bentley, P. J., and Lotto, R. B. (2008) Crash-Proof Systemic Computing: A Demonstration of Native Fault-Tolerance and Self-Maintenance. In Proc of 4th IASTED International Conference on Advances in Computer Science and Technology (ACST 2008), ACTA press, April 2-4, 2008, Langkawi, Malaysia.

Le Martelot, E., Bentley, P. J., and Lotto, R. B. (2008) Eating Data is Good for Your Immune System: An Artificial Metabolism for Data Clustering using Systemic

Computation. In Proc of 7th International Conference on Artificial Immune Systems (ICARIS 2008), August 10-13, 2008, Phuket, Thailand.

Tempesti, G., Roggen, D., Sanchez, E., Thoma,Y., Canham, R., Tyrell, A., and Moreno, J-M, A POEtic Architecture for Bio-Inspired Hardware, In Proc. of the 8th Intl. Conf. on the Simulation and Synthesis of Living Systems (Artificial Life VIII), MIT Press, Cambridge, 2002, 111-115

Tempesti, G., Mange, D., Mudry, P-A, Rossier, J, and Stauffer, A, Self-Replicating Hardware for Reliability: The Embryonics Project, ACM Journal on Emerging Technologies in Computing Systems (JETC), Volume 3 , Issue 2 , 2007

Turing, A., "On Computable Numbers, With an Application to the Entscheidungsproblem", Proc. of the London Mathematical Society, series 2, vol. 42, (1936); reprinted in M. David (ed.), The Undecidable, Hewlett, NY: Raven Press, (1965).

Upegui, A., Thoma, Y., Sanchez, E., Perez-Uribe, A., Moreno, J-M, Madrenas, J., The Perplexus bio-inspired reconfigurable circuit, In Proc of Adaptive Hardware and Systems, AHS 2007. Second NASA/ESA Conference, 2007, 600--605

von Neumann, J., The theory of self-reproducing automata (Champaign, IL: Univ. of Illinois Press, 1966).

Wolfram, S., A New Kind of Science (Champaign, IL: Wolfram Media, Inc., 2002).