# The 6th AISB Symposium on Computing and Philosophy: The Scandal of Computation - What is Computation?

Mark Bishop and Yasemin J. Erden (editors)

AISB Convention 2013 • University of Exeter • 3rd–5th April, 2013

# Foreword from the Convention Chairs

This volume forms the proceedings of one of eight co-located symposia held at the AISB Convention 2013 that took place 3rd-5th April 2013 at the University of Exeter, UK. The convention consisted of these symposia together in four parallel tracks with five plenary talks; all papers other than the plenaries were given as talks within the symposia. This symposium-based format, which has been the standard for AISB conventions for many years, encourages collaboration and discussion among a wide variety of disciplines. Although each symposium is self contained, the convention as a whole represents a diverse array of topics from philosophy, psychology, computer science and cognitive science under the common umbrella of artificial intelligence and the simulation of behaviour.

We would like to thank the symposium organisers and their programme committees for their hard work in publicising their symposium, attracting and reviewing submissions and compiling this volume. Without these interesting, high quality symposia the convention would not be possible.

Dr Ed Keedwell & Prof. Richard Everson
AISB 2013 Convention Chairs

Published by
The Society for the Study of Artificial Intelligence and the Simulation of Behaviour
http://www.aisb.org.uk

ISBN: 978-1-908187-31-4

# The 6th AISB Symposium on Computing and Philosophy: *The Scandal of Computation - What is Computation?*

What is computation? Society builds and uses millions of computers each year so at first sight the answer seems trivial. A computer is merely a general purpose, typically electronic device, that can be programmed to carry out a finite set of arithmetic or logical operations. These days they announce their ubiquity to the world in phones, desktop devices, washing machines, even lawn mowers.

Historically, however, the etymology of the word (from the OED) informs us that the notion of computation was identified with the action of humans who make calculations, often with the aid of calculating machines. In the 1940s this definition was refined with that of an "effective method" (a procedure that reduces the solution of problems to a series of rote steps which is bound to give the correct answer in finite time for all possible inputs), to yield the notion of the algorithm an effective method for calculating the values of a function and the notion of the effective calculability of functions with an effective method (algorithmic solution). In this way, the notion of computation came to be identified with the actions [steps] carried out by [automated] computers to produce definite outputs [in finite time]. This notion frames computation in terms of an agent, which raises the questions of what computation is per se - merely the dynamics of information flow? And in this scenario, how can computational data be meaningful? How can meaningful data acquire truth-values?

For a long time our ideas about computations (or about the underlying computational models) were more or less rigid, fixed, established in the middle of the twentieth century. In the centre there was the model of a classical Turing machine, with its scenario of a finite computation defining a fixed mapping from the inputs to the outputs. The computations of Turing machines served as a means for defining the complexity of computations, the notion of the universality of computations, and the notion of computability (historically, the lastly mentioned three notions should have been listed in a reversed order). Nevertheless, with the advent of modern computing technologies, networking, and advances in physics and biology, has emerged the ideas that computation is a far broader, far more common, and more complex phenomenon than that modelled by Turing machines. It has been increasingly more difficult to see newly emerging models of computations through the optics of Turing machine computations. Examples include biologically inspired models—such as neural nets, DNA computing, self-assembled structures, molecular computers, cognitive computing, brain computing, swarm computing, etc., or physically inspired models, such as quantum computing, relativistic computers, hyper-computers, and, last but not least, "technologically enabled" models, with the prominent example of the Internet, but also various (also mobile) networks.

In order to further explore these and related questions, the papers in this Symposium cover key related issues including, but not limited to:

Computationalism and Neural Systems; Models of Computation; Natural Computing; Computation as Knowledge Generating Processes; Computational Complexity Theory; Quantum Computing; Trivialization Arguments; Natural Computation; Dynamical Systems Theory; Computation and Pragmatics; Dynamics of Information; Interactive Computation; Intentional and Functional Concepts; Hypercomputers; Pancomputationalism; Digital Systems; Type and Token; Computational Universe; Special Relativity; Turing Machines; Stochastic Diffusion Search; Monte-Carlo Tree Search; Computational Platform; Game-Playing; Observer-Relativity; Phenomena and Noumena.

On behalf of the Organising Committee of this Sixth AISB Computing and Philosophy Symposium, we would like to thank all the members of the Programme Committee for their generous support, and for the excellent work in refereeing submissions. We hope that participants will find the event stimulating and enjoyable.

Mark Bishop (Dept. of Computing, Goldsmiths, University of London, UK)
Symposium Chair and AISB Chair
Yasemin J. Erden (Philosophy, St Mary's University College, UK)
Symposium Co-Chair and AISB Committee Member

**Symposium Organising Committee:**

Slawomir Nasuto (University of Reading, UK)
Stephen Rainey (St Mary's University College, UK)
Jiri Wiedermann, (Academy of Sciences of the Czech Republic, CZ)

**Symposium Programme Committee:**

Ron Chrisley (University of Sussex, UK)
S. Barry Cooper (University of Leeds, UK)
José Félix Costa (IST Technical University of Lisbon, PT)
George F. R. Ellis (University of Cape Town, SA)
Peter beim Graben (Humboldt-Universität zu Berlin, DE)
Yuri Gurevich (Microsoft Research, USA)
Phyllis Illari (University College London, UK)
Robert W. Kentridge (Durham University, UK)
Jan van Leeuwen (Universiteit Utrecht, NL)
Matthias Scheutz (Tufts University, USA)
Oron Shagrir (The Hebrew University of Jerusalem, IL)
Mariarosaria Taddeo (University of Hertfordshire, UK)
Mario Villalobos (The University of Edinburgh, UK)

# Contents

# Rethinking Computations [1]

**Jiří Wiedermann** and **Jan van Leeuwen** [2]

**Abstract.** Unlike the classical view of computations that considers them as processes transforming information, we will consider computations as processes generating knowledge. We present arguments supporting this view of computations. These arguments are based on the past and present trends in the use of information technologies, where a steadily growing emphasis on knowledge generation and exploitation is clearly visible. The view of computations–as–knowledge–generators naturally extends to non-man-made systems such as living organisms, brains, social networks and the Universe, and to non-Turing computations. If accepted, this epistemological view will lead to an important shift in our understanding of computations.

## 1 Introduction

**1.1 An attempt to describe the status quo** Computation is not what it used to be. Until relatively recent times – almost until the first half of the twentieth century – computation was an activity ascribed largely to humans making calculations, with or without the help of mechanical calculators. Notions of effective computability only gradually developed in the early 20th century.

By introducing his model of an *a-machine* (short for "automatic machine") Turing has changed this view forever. All of a sudden, humans no longer were the only subject doing computations: the idea of "computational machinery" was born. Turing's idea [18] of computation machines was so simple, yet so profound and influential that until now it has more or less fixed our ideas about computations and about the underlying computational models. J. Copeland [8] captured it concisely when he wrote that *"to compute is to execute an algorithm"*.

Nevertheless, with the advent of modern computing technologies, networking, and advances in physics and biology, the idea emerged that computation is a far broader, far more common, and more complex phenomenon than that modeled by Turing machines. It has been increasingly more difficult to see newly emerging models of computations through the optics of Turing machine computations (cf. [23]). Examples include biologically inspired models – such as neural nets, DNA computing, self-assembled structures, molecular computers, cognitive computing, brain computing, swarm computing, amorphous computing, etc., or physically inspired models, such as quantum computing, relativistic computers, hyper-computers, and, last but not least, "technologically enabled" models, with the prominent example of the Internet, but also various (also mobile) networks.

Have these new computing technologies changed our views on what is computation? Unfortunately, there no single, generally agreed upon view of computing. We briefly illustrate this by inspecting the recent opinions of some prominent fellow computer scientists, physicists, and philosophers on the subject.

Not surprisingly, it appears that the majority of them agrees that computation is a process.The most radical in this opinion seems to be D. Frailey [14], saying: *"Computation in its broadest sense, to me, is anything that happens (as opposed to things that are static). If so, then the principles of computation are, in fact, the principles of processes."*

Then there are researchers who see computations basically as information processing. For instance, P.S. Rosenbloom [15] defines computation in terms of information transformation. R. Bajcsy [3] claims that computation is a transformation/function applied to information.

Other scientists add more conditions for a process to be a computation. For instance, according to L. Fortnow [13], *"Computation is about process, about the transitions made from one state of the machine to another. Computation is not about the input and the output, point A and point B, but the journey."* P.J. Denning [9] sees computation as a sequence of transitions: *"A computation is an information process in which the transitions from one element of the sequence to the next are controlled by a representation."* J.S. Conery [7] also states that *"Computation is symbol manipulation. Computation is a discrete process, a sequence of distinct transitions.* Philosopher J. Searle [16] agrees by saying that *"on the standard textbook definition, computation is defined syntactically in terms of symbol manipulation."*

Next, there are scholars requiring that there must be a relation between the computation and some formal model of computation. According to A.V. Aho [1], *"Mathematical abstractions called models of computation are at the heart of computation and computational thinking. Computation is a process that is defined in terms of an underlying model."* J. Searle [2] adds a further requirement: the process must be physically realizable: *"Computation does not name a machine process. It names an abstract mathematical process that we have found ways to implement in machines."* On the other hand, physicist D. Deutsch [10] maintains that computation is a physical process: *"A computation is a physical process in which physical objects like computers, or slide rules or brains are used to discover, or to demonstrate or to harness properties of abstract objects – like numbers and equations."*

[1] This work was partially supported by RVO 67985807 and the GA ČR grant No. P202/10/1333.

[2] Institute of Computer Science of AS CR, Prague, Czech Republic, email: jiri.wiedermann@cs.cas.cz and Center for Philosophy of Computer Science, Utrecht University, the Netherlands, email: J.vanLeeuwen1@uu.nl

Finally, H. Zenil [26] has placed programmability at the center of the discussion and definition of computation. B. Cantwell Smith [17], on the other hand, believes that computation is not be captured adequately by anything formal. He argues that three key criteria are crucial, namely empirical, conceptual, and cognitively adequacy.

**1.2 The inadequacies of the current approaches** Unfortunately, it appears that each of the previous definitions falls short in some aspect of computation that intuitively seems to be important. For instance, definitions seeing a computation as an arbitrary process will also include objects that are generally not considered as computers. From such a stance it follows that anything, (e.g., a rock, a river) can be viewed as instantiating any computation. Similar objections hold for seeing computation-as-information-processing. Namely, information processing is the change (processing) and communication of information, and hence any change in the Universe can be seen as a computation.

These notions of computing are restricted by the requirement that transitions (changes of information) are controlled by some finite mechanism, defined by the "underlying model of computation". This leads to a somewhat circular definition, since computation then is whatever the model of computation, or real computers do. Moreover, insisting on the underlying models excludes brains (and other devices operating on unknown principles) from doing computations. Computation-as-symbol manipulation neglects analog computing, or non-Turing computing, such as (abstract) computing with reals, or computing by ruler and compass.

Insisting on computation being a physical process pulls the notion of computation down to a concrete level which might be miles away from a reasonable abstraction level (think, e.g., of theorem proving). Last but not least, considering issues of programmability moves the focus from general computation to so-called uniform computations, avoiding the larger class of non-uniform computations, e.g. by circuits, the Internet, mobile computing, amorphous computing, etc.

**1.3 The way out** There is one thing that all the previous definitions of computation have in common which seems to get little attention sa far. This is the fact that they all tend to express computation as *what the underlying hardware is doing*, or, in other words, *HOW the process of computation is realized.* This leads to little insight because what the hardware does is performing operations upon data; this forces us to see as computation whatever meaningless operations with data. All our experience with computation points in other direction: we are primarily interested in *WHAT the computation does* for us, i.e., for the designers, users, observers. What a computation does is expressed by the design of a computer system (or, possibly, by the software of the underlying system, if it is programmable). Knowing how a computation does what it does is less interesting – again, all our experience says that what the computation does can be implemented in many equivalent ways on a different hardware. So what is it what the computation does?

Our answer is simple: *computation generates knowledge.* It generates knowledge over the domain for which the underlying computational system was designed or evolved, or in which the system itself has evolved.

That being said, one question springs immediately into one's mind: what is knowledge, and what does it mean to generate it? The notion of knowledge is, similarly as computation, another notoriously elusive notion. Nevertheless, this makes our thesis that computation generates knowledge robust, in a sense. We will work with the following definition of knowledge (cf. Wikipedia [25]).

*Knowledge is a familiarity with someone or something, which can include facts, information, descriptions, skills, or behavior acquired through experience or education. It can refer to the theoretical or practical understanding of a subject. It can be implicit (as with practical skill or expertise) or explicit (as with the theoretical understanding of a subject); it can be more or less formal or systematic.*

As we shall later see in the examples of various computational systems, this definition will suit our needs. Obviously, knowledge according to our definition is *observer–dependent* and so is computation seen as a process generating knowledge. This is another difference with the approach computation-as-information-processing which tends to be an observer–independent notion and hence, too general as explained in Section 1.2.

What are the advantages of our definition? First, it allows a clear distinction of what is computation from what it is not, based on the ability of computation to produce knowledge. For instance, according to this definition, a rock (cf. [6]) computes if only an observer can give plausible reasons what knowledge is produced and that this production is caused by the processes occurring within that rock. This definition also allows to assign the computational ability to the brains, and possibly to other known and not-yet known gadgets that clearly produce knowledge. Second, the definition admits classification of computations based on the kind of knowledge that the underlying processes utilize and produce. Third, the independence of the definition on the underlying computational mechanisms is welcomed since it covers a host of known and not yet known instances of computing. Last but not least, our definition supports thinking on computation in high level abstract terms related to knowledge. In the case of artificial computing systems which are constructed by humans this is closely related to their design methodology that starts at the high level of abstract specifications. It is interesting from a philosophical perspective as well.

**1.4 Contents** In Section 2 we present our main thesis – computation as a process of knowledge generation – and support it by examples of contemporary, non-man-made and non-Turing computational systems seen as knowledge generating systems. Finally, Section 3 contains the conclusions.

## 2 Computation–as–knowledge–generation

The aim of this section is to provide evidence supporting our main thesis:

**Thesis:** Computation is the process of knowledge generation.

To this end we will survey a number of computational systems from the viewpoint of their ability to utilize and produce knowledge. We list the systems in the form of a table. With each system we will also specify the underlying knowledge domain over which the system generates knowledge and the type of generated knowledge. The entries in the table are supposed to be self-explanatory.

| | Computational system | Underlying knowledge domain | What knowledge is produced |
|---|---|---|---|
| | | **Contemporary computing systems** | |
| 1 | Acceptors | Formal languages | Membership |
| 2 | Recognizers | Formal languages | Membership function |
| 3 | Translators | Functions, relations | Function value |
| 4 | Scientific computing | Mathematics | Solutions |
| 5 | Theorem provers | Logic | Proofs |
| 6 | Operating systems | Computer's devices and peripheries | Management of computer's own activities |
| 7 | Database and information systems | Relations over structured finite domains | Answers to formalized queries |
| 8 | Control systems | Selected domains of human activity | Monitoring, control |
| 9 | Search engines | Relations over unstructured potentially unbounded domains | Answers to queries in a natural language |
| 10 | Artificial cognitive systems | Real world, science | Conjectures, explanations |
| | | **Natural computing systems** | |
| 11 | Living systems, cells | Real world | Life, behavior, intelligence |
| 12 | Brain, Mind | Knowable world | Knowledge of the world |
| 13 | Social networks | Knowledge of the world | Knowledge of the world |
| 14 | The Universe | The Universe, physics | Living systems |
| | | **Non-Turing computing systems** | |
| 15 | Compass and ruler | Euclidean geometry | Euclidean constructions |
| 16 | BSS machine | Theory of real numbers | Values of real functions |
| 17 | Oracles | A set $A \subseteq \Sigma^*$ | Characteristic function of $A$ |

Vertically, the table is split into three parts dealing with contemporary, natural, and non-Turing computing systems, respectively. In its first part, the systems in the table are listed roughly in the chronological order as they appeared in the timeline of history of modern computing.

In the sequel we will occasionally use the term *"wisdom"* in its basic philosophical meaning: the use of knowledge.

There also is a deeper reason why the contemporary systems in the table are ordered the way they are. Note that the items 1 to 5 all deal with an abstract, formal knowledge related to formal domains of formal languages or mathematical or logical theories. So is the nature of the generated knowledge. The underlying systems are endowed by wisdom embedded into their design by their creators.

Systems from 6 to 8 still deal with formalized knowledge, but this time the knowledge is related to selected segments of the real world. The formalization of knowledge as well as the way it is processed, has been embedded into the design of the underlying systems.

The man-machine interface in all previous cases was formalized and restricted to selected segments of human activities.

Finally, systems 9 and 10 differ from all previous systems because they deal with potentially unbounded amounts of knowledge, delivered to systems in unstructured, "raw" form. This knowledge is related to the real world and is either acquired by the system's own means (e.g., by system's sensory-motor activities) or it has the form of knowledge already acquired by other means (e.g., they contain knowledge in written form produced by people). They communicate with their users in more or less formalized language approaching (or even being equal to) a natural language. Systems in 10 are designed to discover implicit relations among data and make them explicit, or to produce "intelligent" behavior. The wisdom they possess is not given to them once for all times by their creators – their wisdom is subject of change and evolution by mechanisms they possess. These mechanisms alone can also be subjects of evolution.

Thus, we see that the nature of the underlying systems with respect to their ability to collect, accumulate and exploit, demonstrate or generate knowledge grows along several dimensions. These dimensions reflect the shift in the formalization level of knowledge processed/produced by the systems, in the way the systems interact with their environment, in their learning ability, in the utilization of the acquired knowledge for their own purposes, etc. Nevertheless, we are not going to follow this line of thinking.

Consideration of the systems under the heading *natural computing systems* shows that our thesis also extends to instances that are not man-made. Nonetheless, intuitively they should also be regarded as computations. In [11], it is argued that knowledge generation is inherent to natural (as opposed to artificial) computing systems. From our stance, this view is naturally subsumed by our general thesis.

Item 11 in the table are living systems. All of them are interactive systems – they glean information from their environment and turn it into knowledge that they demonstrate by their behavior and survival in their environment. In many cases interaction also means communication among the conspecies. Communication among computing systems is possible thanks to the property of *compositionality* obeyed by computing: (a representation of) knowledge produced by one computation can be processed by an other computation.

In [5] it is argued that the notions of communication and computation are conceptually quite close. This can be seen as consequence of our thesis, when we view communication as a process of knowledge passing among the interacting agents.

Interaction combined with communication leads to the emergence of social networks (item 13).

Including brains (or minds) among the computational systems is a natural consequence of our definition of what is computation. Classifying the Universe (item 14) as a computational system is also natural w.r.t. our definition: it is undeniable that, at least at some places, the Universe produces knowledge. At some places, life will evolve and we are back at item 11. It seems that the Universe is the terminal stage in the development of computing systems producing knowledge, indeed.

Now, as an extreme case, perhaps as an attempt to falsify the thesis, consider non-Turing computations, i.e., computations that cannot be realized adequately by Turing machines.

Item no. 15 – computations by (idealized) compass and ruler allowing the construction of geometrical figures with infinite precision – needs no further explanation. Item no. 16 deals with computations over reals. The corresponding model of computation – BSS machine – is the model by Blum, Shub and Smale [4] intended to describe computations of rational functions over reals. Finally, item no. 17 considers oracles in the spirit of Turing's original proposal from [19].

The last three computational systems all have in common is that they are not realizable in classical physics. This is because the first two models compute with real numbers, and the last model could deliver classically non-computable information. Yet the thesis classifies the respective processes as computations because physical realization is not a condition for our thesis to hold. This is in a good agreement with "practice" – in geometry, mathematics and computability theory it is useful to think about the respective processes as of computational processes. All three examples offer good arguments for not requiring in the thesis that the computation must be physically realizable.

The thesis as stated before — that computation is a process of knowledge generation — is quite general. However, in a forthcoming paper we show that one can go into more details and state conditions that are necessary for a computation to *provably* generate knowledge. These conditions will require that knowledge generated by a computation can be inferred within the knowledge domain underlying the computation, similarly as there must be an evidence that the knowledge at hand is generated by the generating process, indeed. Under this approach, the process generating knowledge becomes a parameter in the overall scheme.

Finally, let us show that our definition of computing has more potential than its application to computing itself. For instance, in analytic philosophy a problem emerged whether computing is a *natural kind*, rather than a *cultural kind*. It seems that computing systems operating autonomously within the real world and generating knowledge about it, have to internally create a model of this world (cf. [24]) (similar to the mind). The grouping of (the representations of) the objects within such models, and the respective observer-independent knowledge derived within these models, probably is a natural kind. An other use of our definition can simplify the often discussed question in cognitive science of what cognition is, if not computation (cf. [20]). If we see cognition as the ability to collect, accumulate and exploit, demonstrate or generate knowledge than this definition equals our definition of computation in its most developed form.

## 3 Conclusion

We have defined computation as an epistemological process generating knowledge. Using examples examples of contemporary, non-man-made and non-Turing computational systems we showed that computation performed by such systems can be seen as knowledge generating processes. The approach liberates the notion of computing from its dependence on the underlying computational model. In contrast to the majority of the previous approaches focusing on the procedural aspects of computing, our approach concentrates on what is computed.

Let us remind the recent idea of M. Vardi [22] that computation is a universal enabler of science, supporting both theory and experimentation. According to our thesis, *computation is the engine of science, for it creates knowledge.*

We believe that all the above mentioned facts warrant the new view of computation. If accepted, our definition of computation will change the way we think about computation.

## REFERENCES


[1] Aho, A.V.: Computation and computational thinking. *Magazine Ubiquity*, Volume 2011. Issue January, January 2011, Article no. 1

[2] Almond, P.: Machines like us, an interview by Paul Almond with John Searle. Machines Like Us, March 2009, http://machineslikeus.com/interviews/machines-us-interviews-john-searle-0

[3] Bajcsy, R.: Computation and information. *Comput. J.* 55(7): 825 (2012)

[4] Blum, L., Shub, M., Smale, S.: On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, *Bulletin of the American Mathematical Society* 21 (1), 1989.

[5] Broderick, P.B.: On communication and computation. *Minds and Machines* 14 (1):1-19, 2004

[6] Chalmers, D.J.: Does a rock implement every finite-state-automaton? *Synthese*, 1996, Vol 108, pp. 309-333, Kluwer

[7] Conery, J.S.: Computation is symbol manipulation. *Comput. J.* 55(7): 814-816 (2012)

[8] Copeland, B.J.: What is computation? *Synthese*, 1996, Vol 108, pp. 335-359, Kluwer

[9] Denning, P.J.: What is computation? (opening statement), *Magazine Ubiquity*, Volume 2010, Issue October, October 2010, Article No. 1

[10] Deutsch, D.: What is computation? (How) does nature compute? In: Zenil, H. (Editor): *A Computable Universe: Understanding and Exploring Nature as Computation*, World Scientific Publ. 2012, pp. 551-566

[11] Dodig-Crnkovic, G.: Knowledge generation as natural computation. *Journal of Systemics, Cybernetics and Informatics* 6 (2), 2008

[12] Ferrucci, D., et al.: Building Watson: An overview of the DeepQA Project. *AI Magazine,* Fall 2010, pp. 200-214.

[13] Fortnow, L.: The enduring legacy of the Turing machine. *Comput. J.* 55(7): 830-831 (2012)

[14] Frailey, D.J.: Computation is process. *Magazine Ubiquity*, Volume 2010, Issue November, November 2010 Article No. 5

[15] Rosenbloom, P.S.: Computing and computation, *Comput. J.* 55(7): 820-824 (2012)

[16] Searle, J.: Is the brain a digital computer? *Proceedings and Addresses of the American Philosophical Association* 64 (1990) 21-37

[17] Smith, B.C.: The foundations of computing. In: *Computationalism: New Directions*, M. Scheutz (ed.), pp. 23-58, MIT Press (2002)

[18] Turing, A.M.: On computable numbers, with an application to the *Entscheidungsproblem*, *Proc. London Math. Soc. Series 2*, 42 (1936) 230-265.

[19] Turing, A.M.: Systems of logic based on ordinals, *Proc. London Math. Soc. Series 2*, 45 (1939) 161-228.

[20] van Gelder, T.: What might cognition be, if not computation? *The Journal of Philosophy*, Vol. 92, No. 7, 1995, pp. 345-38
[21] van Leeuwen, J., Wiedermann, J.: A theory of interactive computation. In: *Interactive Computation: The New Paradigm.* Goldin, D., Smolka, S.A., Wegner, P. (Eds.), Springer Verlag, 2006, pp. 119-142
[22] Vardi, M.: Science has only two legs, *Comm. of the ACM* 53:9 (2010) 5
[23] Wiedermann, J., van Leeuwen, J: How we think of computing today. In: *Computability in Europe*, Proc. CiE 2008, LNCS 5028, Springer, 2008, pp. 579-593
[24] Wiedermann, J.: The creativity mechanisms in embodied agents: An explanatory model. To appear in: *Proc. 2013 IEEE Symposium Series on Computational Intelligence* (SSCI 2013), 2013
[25] Wikipedia, http://en.wikipedia.org/wiki/Knowledge, 2013
[26] Zenil, H.: What is nature-like computation? A behavioural approach and a notion of programmability, *Philosophy & Technology* (special issue on History and Philosophy of Computing), Springer, 2013

# What is a Digital State?

**Vincent C. Müller**[1]

**Abstract.** There is much discussion about whether the human mind is a computer, whether the human brain could be emulated on a computer, and whether at all physical entities are computers (pancomputationalism). These discussions, and others, require criteria for what is digital.

I propose that a state is digital if and only if it is a token of a type that serves a particular function – typically a representational function for the system. This proposal is made on a syntactic level, assuming three levels of description (physical, syntactic, semantic). It suggests that being digital is a matter of discovery or rather a matter of how we wish to describe the world, if a functional description can be assumed. Given the criterion provided and the necessary empirical research, we should be in a position to decide on a given system (e.g. the human brain) whether it is a digital system and can thus be reproduced in a different digital system (since digital systems allow multiple realization).

## 1. MOTIVATIONS: THE COMPUTATION–ALIST PROGRAM, ARTIFICIAL INTELLIGENCE, PANCOMPUTATIONALISM

Given that the ontology of digital states is hardly an established philosophical problem, it will be useful to briefly motivate its discussion. A clarification as to what constitutes a digital state is necessary primarily in the context where digital states are part of a certain kind of digital systems, namely digital computers. There is a significant confusion over which objects in the world are computers because there is no agreement on the criteria; Shagrir calls this the "problem of physical computation" (1, 394ff). For example, on the one hand there are the proponents of a computational representational theory of mind (CRM or "computationalism") who believe that the human mind is a functional computational mechanism operating over representations. These representational abilities are then to be explained naturalistically, either as a result of information-theoretical processes (2, 3), or as the result of biological function in a "teleosemantics" (4, 5).

On the other hand, the opponents of computationalism divide into two camps: those who think that some natural mechanisms may be computers, but the human mind is not one of these, and those who think that all systems can be *interpreted* as computers and so the human mind is just a computer like everything else: "every natural process is computation in a computing universe" (6, 10) – this position is now often called "pancomputationalism" (see 7). Finally, the question to what extent artificial intelligence (AI) is possible, requires an explanation what kinds of machines computers are and what they can do in principle – given that digital computers are currently the main kind of mechanism that is used for AI.

So, if the brain is a computer, could we perhaps reproduce the brain on different hardware? If we would scan the whole brain of a human and run it on a (different) Turing machine, *would it produce intelligence?* (And all the other cognitive features of humans?) If we could emulate the brain on different hardware it would show the same external behavior or output as does the emulation of earlier software or hardware, e.g. of a WWII 'Enigma' machine or of programs that ran on some of the first computers, like the 'Manchester Mark I'.

This might be a possibility if certain conditions are met, in particular: "Computability: brain activity is Turing-computable" (8) and "At present there is no convincing empirical evidence for uncomputability in the brain, although there is no shortage of claims for it." (9).

Several further problems for a computational theory of the mind would benefit from a resolution of what constitutes a digital state. Within the context of the discussion of the computationalism mental processes are traditionally understood as information processing through computational operations over *representations*. Is representation a necessary feature of computing? If yes, perhaps something can be called a digital state only on *presupposing* mental processes in the system, so there is a threat of a circle here (unless we are looking at a feedback circle). Another is the problem of "grounding" for computational systems: "How can the meanings of the meaningless symbol tokens, manipulated solely on the basis of their (arbitrary) shapes, be grounded in anything but other meaningless symbols?" (10, 335). We have argued in recent papers (11, 12) that a nonconceptual phenomenal content should be at the base of such grounding. If it were to turn out that such content is necessary but is analogue and cannot be present in purely digital systems, this would show that human cognition is not purely digital – and that AI on purely digital computers is impossible. In separate work, we argue that nonconceptual content is precisely non-digital content.

I will argue in the following that being a digital state is to be a state of a type or category, but that we should not conclude from this that being a digital state is "description-dependent". In particular, a state can be digital if it fulfills a particular function in a system of which it is a part – e.g. a representational function.

## 2. BASIC CHARACTERISTICS

In a first approximation, being digital means being in a discrete state, a state that is strictly separated from another, not on a continuum. Prime examples of digital states are the states of a digital speedometer or watch (with numbers as opposed to an analog hand moving over a dial), the digital (binary) states in a conventional computer, the states of a warning light, or the states in a game of chess. Some digital states are binary, they have only

[1] Anatolia College/ACT, Pylaia & FHI, Dept. of Philosophy, University of Oxford. vmueller@act.edu, www.sophia.de

two possible states, but some have many more discrete states, such as the 10 numbers of a digital counter or the 26 letters of the standard English alphabet.

### 2.1. Multiple Realization

Goodman had pointed out in his early theory of representation that digital "marks" (physical entities) are "differentiated", as he called it, precisely if they can have an exact replica: one can write the same letter "A" twice, since "A" is differentiated from any other letter. Analog marks, in contrast, are "dense", meaning that for any pair of similar but non-identical marks, there is space for another mark in between (13, cf. 14). So, the states of an analog speedometer with a hand moving in analogy to the speed of the vehicle are continuous, just as the speed it represents, and for any two places where the hand can be, there is a third in between.

As we already pointed out with reference to Goodman, it is characteristic of a digital mark that it can be realized several times. So, one can write the same word twice, even if one cannot make exactly the same mark on paper twice. John Haugeland usefully explains this phenomenon with games: chess is a digital game because we can reproduce an earlier position precisely; we can even resume the same game with different pieces. Billiards, on the other hand, is analogue, because we can reproduce an earlier position only to a certain degree of measured precision, and if we were to reproduce the same position with different physical objects, it would not be the same position (15, 57, earlier in 16). The possibility of multiple realization is a *result* of digital states being discrete: Since a white bishop in chess can be clearly on field C3, we can move it back to C3, or replace it with a different bishop; it does not matter that it is not identical to the earlier one, provided it is clearly a white bishop on C3.

### 2.2. Discrete vs. Continuous

But which of the two characteristics is crucial for an analog state, the analogy to the represented, or the continuous movement?

This question becomes relevant in the case of analogous representations that proceed in steps, e.g. a clock the hands of which jump from one discrete state to another. Zenon Pylyshyn argues that the underlying process is analog, and this is what matters: "an analog watch does not cease to be analog even if its hands move in discrete steps" (17, 200, 18, 332 agrees). James Blachowicz also thinks that being on a continuum is sufficient for being analog, taking the view that "differentiated representations may also be analog – as long as they remain *serial*", his example is a slide rule with "clicks" for positions (19, 71). (Note how these authors assume a functional description, an issue to which we shall return later.)

These views ultimately fail to differentiate between analogue and digital representations. Note that the very same underlying mechanism could give a signal to a hand to move one step and to a digit to go one up (this is actually how clocks are controlled in centralized systems, e. g. at railway stations). In any case, some classic examples of digital states are clearly in a series, indeed a series of infinitely many steps: the series of the natural numbers. These two points rule out Blachowicz' proposal to take being serial as a criterion. Pylyshyn, on the other hand, would presumably say that the underlying mechanism is already digital, so the clock is digital in this case – but surely there are systems where a digital signal is converted into an analogue one (the speedometer in most modern cars) and where an analogue signal is converted into a digital one (an analogue central clock that controls several digital clocks), so we should then say that the system has digital and analog parts. I conclude that the first crucial feature of a digital state is indeed that of being a discrete state – not excluding that of being in a series, even in a series that is analogue to what is represented.

## 3. EVERYTHING IS ANALOGUE – AND DIGITAL, TOO?

A given blob of ink on a piece of paper might be in a particular digital state but it has several analogue properties, too, such as a color, a shape, a history, a value, etc. In fact, all digital states we have seen so far are states of physical entities, and thus have analogue properties as well. (For our purposes, we can leave aside the question whether abstract objects can be digital.) Being digital is a property of certain physical entities that are also analogue – though they might not be analogue representations. But of which entities? Negroponte puts it nicely: "A bit has no color, size or weight, … It is a state of being: on or off, true or false, up or down, in or out, black or white." (20, 14) But, which of the black things are in the state of being of a bit? What determines whether something is a bit?

It may seem that *we* can just define what counts as digital as we please, so everything is digital. Say, for example, the two of us agree that if I light a fire on a particular hill that means "the King is out of town". Is the hill henceforth in a binary digital state? Is anything not in any number of digital states, then?

For a given physical thing (say, my desk lamp), there are descriptions as continuous (where is the light, what is its shape, what its color?) and as digital (is it on/off?), so a natural response is to say that being a digital state is relative to a particular description: Under one description the light is digital, under another it is not, so we have at least a "relativity of descriptions" (21, 29).

This consequence is very tempting for digital computation and its algorithmic procedures. Alan Turing already seems to have already gone in this direction: "The digital computers […] may be classified amongst the 'discrete state machines', these are the machines which move by sudden jumps or clicks from one quite definite state to another. […] Strictly speaking there are no such machines. Everything really moves continuously. But there are many kinds of machine, which can profitably be thought of as being discrete state machines." (22, 439).

John Searle takes it one step further: "The electrical state transitions are intrinsic to the machine, but the computation is in the eye of the beholder." (23, 64). Oron Shagrir concurs: "… to be a computer is not a matter of fact or discovery, but a matter of perspective" (1, 393), and about algorithms: "… whether a process is algorithmic depends on the way we describe the process." "… processes are not really step-satisfaction [algorithmic]. It is simply useful to describe them this way." "Whether a system is digital depends not only on its natural properties, but chiefly on the context in which it is described." (18, 321, 331, 335).

It now seems that not only do we have a relativity of descriptions, but that a *description dependence of facts*: it would then be constitutive of being a digital state that its existence is

dependent on contingent social interests, namely the interest in a particular feature that makes a digital state. To illustrate this with a classical example: Being 'digital' is more like the word 'constellation' than the word 'star'. What is part of a stellar constellation depends on what *we* make part of it. What is a star depends on the world and is a matter of astronomic discovery (cf. 21, 18, 28, 24)

## 4. CLARIFICATION I: TYPE/TOKEN

Understanding the true nature of relativity here requires some further clarifications. Haugeland defines as follows: "A *digital system* is a set of positive and reliable techniques (methods, devices) for producing and reidentifying tokens, or configurations of tokens, from some prespecified set of types … A *positive* technique is one that can succeed absolutely, totally, and without qualification; … Many techniques are positive and reliable. Shooting a basketball at the basket is a positive method (for it getting through), for it can succeed absolutely and without qualification;" (15, 53f). Demopoulos thus calls being a digital mechanism of a certain type being a member of an "equivalence class" (25). Harnad talks about "symbol tokens" – but not of types (10, 1.2). The characteristic of "multiple realization" (see above 2.1) is crucial, so there must be a "positive technique" to produce perfect realizations that are clearly of *this* digital state. Multiple realization, however, this is not a feature of *certain* types, it is a feature of types, quite generally. For example, a transistor can be in a voltage state that is clearly of type "on" or "off", but it can also be on the borderline between the two – it just so happens that our computing machines are made with systems that do not usually get stuck in intermediate states. Every digital state is also on a continuum: the digital speedometer might change quickly from one number to another, but it does have intermediate states – just that these are not states of numbers, not states of these types, of these descriptions. Being of a digital state is thus not a statistical question of whether "all or none" states occur often, since what counts as "all" depends on the type. Just looking at the physical distribution will not tell us anything.

What is crucial here, therefore, is that a digital state is *of a type*. If it is of a type, then there can be multiple perfect realizations of it: No matter how many borderline cases a type happens to have (some have many, some have none), there is always the possibility of *clear* cases, and that is what is needed for being a digital state; we need to fulfill the implied semantic normativity of the "token of a type". A digital type can be vague, it just needs possible clear cases. So, we require in a first instance that *a digital state is a state that is a token of a type*. What we need to see now is which tokens of a type are the digital states.

## 5. CLARIFICATION II: LEVELS OF DESCRIPTION

In a next step, it is helpful to differentiate at least three levels of description of a proposed candidate for being in digital or digital computational states: (a) *physical*, (b) *syntactic* and (c) *semantic* levels – something only very few people do, despite the tradition of functionalism (26, 57, 27, 402f, cf. 28) (29).

The *physical level* (a) is that of the physical 'realization' of the computation – this is presumably what Searle had in mind with his "electrical state transitions" (above).

That physical state is in (b) a particular digital state on the *syntactic level* (a binary state, or a number, a letter, a word). It is at this level that a particular mathematical function is computed, it is fully specified by specifying it on this level.

(c) That digital state in turn may represent something else, e.g. a truth value, a time, or a color, let us call this the *semantic level*. What is represented at this level may, again, have representational functions on several levels (the color can represent a political opinion, etc.).

A digital computer works because it is constructed in such a fashion that its physical states cause other physical states in a systematic way, and these physical states are also digital states on the syntactic level. (The physical states need not be of the same physical type, they can be voltages and magnetic fields, for example.) The semantic level is not necessarily present and is not necessary for the digital system or digital mechanism. Contrary to popular belief, a computer does not require semantic content to function, (e.g. 30, 31, 1414ff) and (15, 66, 32, 385).

Given this clarification of levels, we can re-evaluate the understanding of digital states. The *semantic* level allows for a true relativity of facts, not just of descriptions: The same computer following the same algorithm can be said to compute different things. This is hardly surprising. For example, it may well be that what a computer does with the same binary sequence is to add two numbers or to change one letter to another. Whether we want to regard the binary sequence as the one or the other will depend on the context. So, these syntactic binary states can have many different contents, on a semantic level (just like "2 + 2 = 4" can add apples or pears). The semantic level, however, is irrelevant to the specification of the digital states, which reside on the syntactic level. – Contrary to popular belief, relativity on the semantic level does not show that there is a relativity of facts on the *syntactic* level. (Note that I am not thereby claiming that "digital state" is a natural kind, i.e. roughly a kind where belonging to the extension is determined wholly by criteria that are themselves natural kinds [recursively], that is, their existence is independent of any conceptual system that has a label for them.)

Remember that the digital states in a system often represent other digital states (e.g. the binary states represent numbers), which relates to the discussion over whether our mental computation is computation over material symbols, as is writing down a mathematical proof. (Do I think in words?)

## 6. CLARIFICATION III: A DIGITAL STATE IS A TOKEN OF A FUNCTIONAL TYPE

It is useful to note that not all systems that have digital states are digital *systems*. We can, for example, consider the male and female humans entering and leaving a building as digital states, even as a binary input and output, but in a typical building these humans do not constitute a digital system because a relevant causal interaction is missing. In the typical digital system, there will thus be a *digital mechanism*, i.e. a causal system with a *purpose*, with parts that have *functions*. Digital mechanisms in this sense may be artifacts (computing machines) or natural objects (perhaps the human nervous system). However, it seems

clear that not all digital states are parts of computational systems: the words in this paper are digital states, but their function is not computational.

If being of a type was the criterion for being digital, then everything would be in any number of digital states, depending on how it is described. However, what we really should say is that something is digital because that is its particular *function*. My desk lamp is always in a digital state, because being on/off is part of its function. The first letter of this sentence is in the digital state of being a "T" because that is its function – it is not an accidental orientation of ink or black pixels. The sun, on the other hand, is not in a digital state at present, though it can be shining or not shining at some place.

We make artifacts where some physical states cause other physical states such that these are physical states of the same set of types, e.g. binary states. (Note that one machine might produce binary states in several different physical ways, e.g. as voltage levels and as magnetic fields.) If someone would fail to recognize that my laptop computer has binary digital states, they would have failed to recognize the proper (non-accidental) function of these states for the purpose of the whole system – namely what it was made for. The fact that a logic gate in my laptop is a binary state depends on whether it *has* that function and is not description dependent. (And the fact that it computes is crucial to its function, but not to that of, say, my shaving brush – so pancomputationalism seems misleading here.)

I conclude that we should say a state is digital if and only if it is a token of a type that serves a particular function.

So, the function is that determines whether something is a token of a type or not. The normativity of having or fulfilling a function generates the normativity of being of a type. The type has the function, being of the type allows to fulfill the function. (Even though *we* would prefer to design digital *systems* such that they have very few borderline cases and almost only clear cases, this is not a criterion for being a digital system.)

### 6.1. Which Function?

At this point, it is clear that the description dependence of being digital depends on that of having a function. Functions are a very large issue, let me just indicate why one might think that there may be some facts here that are not description dependent.

In the case of an artifact, we *assume* a functional description. If the oil-warning light on a car dashboard is off, is it in a digital state? Yes, if its function is to indicate that nothing is wrong with the oil level. (It may serve all sorts of other accidental functions for certain people, of course.) But if the light has no electricity (the ignition is off), or if it was put there as a decorative item, then the lamp is not in a digital state "off". It would still be off, but this state would not be digital, would not be a token of the same functional kind.

In the case of a natural object, the allocation of proper function is dependent on teleological and normative description of systems (33, esp. 2.2) – a problematic but commonplace notion. The function of a human's legs seems to be locomotion (and kicking balls), but we are not tempted to say that the leg is in digital states, while perhaps the muscle cells are – with respect to *their* function. Whether or not the legs are digital, they can be simulated (to an arbitrary degree of precision) on digital systems – only that the simulation will not walk, it will just "walk in the simulation".

The description of an artifact in terms of function is to say that something is a means to an end, it serves the function to achieve that end – a function that can be served more or less well. (Note that serving a function does not mean being used for that function; there may well be no agent that can properly be said to be using the artifact, e.g. if it is part of a large and complex system.)

The dependence of being a digital type on having a proper function can be illustrated by looking at a digital information channel. Even if that channel is already defined as digital, it is not clear which parts are relevant and carry a function. For example, in computer security, there is the question which aspects of the information are used to convey information, searching for possible "covert channels", for example in the time delays between signals (see 34). These time delays can be used to convey information, similar to the delays in Morse code. So, given a time sequence of digital signals, it is still not clear which digital signals are present, unless function is specified, e.g. by stating whether particular time delays are significant or not.

### 6.2. Too many Functions, too Many Types

At this point, we need to see whether the account so far is sufficient to identify the digital states. A little reflection will reveal that it captures the standard samples of digital states, such as the states inside a digital computing system, or the states of a digital clock or indicator. However, it is very hard to see how the account as stated can be prevented from incorporating all too many states that are, intuitively, not digital states. For example, it will not only include the oil indicator lamp on my dashboard, but any lamp. After all, whatever the proper function of a particular lamp may be (such as to shed light on a desk), it fulfills that function by being "on" and does not fulfill it when "off". So, all lamps are in digital states. But so are all hats: Whatever the proper function of a particular hat may be (such as to shade the head), it fulfills that function by being "on the head" and does not fulfill it when "off the head". Wherever we look, we seem to find functionally determined clearly discrete tokens of types – but the flood is too hard to stem with just the sloppy notion of "function" that we used so far.

One diagnosis of the situation is the following: What we explained so far is really not *digital type*, but *type* in general.

The notion of function is really necessary for any type/token distinction. Take the word "tree". Which sounds or graphical shapes are tokens of that type? This is not just determined by some particular sound or graphical pattern but by whether a given sound or shape serves the function of being of that type. What is more, the type is functionally individuated: Which words are of the type "tree"? Those that serve the function of talking about trees.

So, the notion of function is not characteristic of digital types only, but of types quite generally. Of course, one might want to say that all types are digital, but it will become clear presently that this would constitute a deviation from current usage.

## 7. WHICH FUNCTIONAL TYPES ARE THE DIGITAL ONES?

We will discuss briefly some proposals for restricting the functional types to the desired digital ones. Each of these proposals contains a grain of truth that will be used in the final theory.

### 7.1. Computational

Since computational systems is what we are interested in, it seems natural to say that digital states are those that serve a computational function in a computational system. On this proposal, the notion of computation would be explained first, in formal, mathematical terms, and being digital is dependent on that explanation.

It so happens that all digital states can be part of a digital computational system, but do not have to be. It is irrelevant for its being digital whether a single warning light is or is not part of a computational system; it is still accurate to call it a digital representation. A digital clock can be just the output system of a purely analog time-keeping device. It appears that the class of digital non-computational states is not empty, so we cannot use computation to distinguish the digital types from the others. (And it follows from our previous discussion of digital vs. discontinuous that these are cases of digital states.) Having said that, it is a virtue of this proposal that it stresses the formal, syntactic, nature of computing – a feature that we will use presently.

### 7.2. In the System

In specifying the function of a state, one is often required to take recourse to the system of which it is a part, in particular to the *digital system*. This is clearly the case in conventional digital computers, so could this not provide a narrowing down of the right functional types?

In the case of conventional computing machines this proposal would work just like the one to start with "computing"– so there is clearly some unity here. However, if we allow systems to cover other systems that include the digital clock or warning light, how are we then going to limit the notion of "system" in the necessary way? It appears that a "system" will suffer from just the same defect as the "function" itself: anything is a system if it has some function or other. So, yes, all digital states are states in a system – but so are too many other states.

### 7.3. Representational

Perhaps we really have *two* notions here, the discrete states in general, and those discrete states that are representational, which we could call "digital"?

This has two disadvantages: It explains one obscure notion with an even more obscure one ("representation"). More importantly, it restricts digital states to those that represent, which is just what we should not do. It is precisely characteristic of the digital states in our prime example, the binary digital computer, that they do *not* represent. Just being of the state is what allows the system to perform its operations. The basic units have no meaning or representation – though they can be so *interpreted*, if desired, and in various ways, as we indicated above. So, we should not assume that all digital states are representational, but we must keep in mind that many are.

### 7.4. Pre-Defined

In at least some digital systems, notably in binary computers, the set of digital types is an explicitly pre-defined finite set. Is this the characteristic feature?

Is this necessary, however, in order to make a digital system, or a system with digital states? And *how* are they pre-defined? The proposal appears unnecessarily narrow, because it excludes digital states in systems that are not formally constructed. It would show a priori, for example, that the human brain does not have digital states (except if a creator pre-defined them).

### 7.5. Syntactical

The peculiar distinction of those types that are digital is that they are so devoid of content; it really does not matter at all how they are realized, what properties they have, provided that they are generally recognizable as being of the particular type. This aspect is probably best described by saying that digital types are *syntactic* types. A token of a syntactic type thus only contributes its *being of that type* to a larger syntactic system, nothing else. In particular, it cannot be said to have a meaning. In this sense, binary code is syntactic, and so are letters of an alphabet. Already words or lexemes are not syntactical, since they are semantically defined. Of course, lamps or hats are not syntactic items, so it appears that the requirement of syntactic definition of the type does our job to narrow down the many functional types.

## 8. WHICH STATES ARE DIGITAL?

In the case of the human nervous system, there are the questions whether it is a digital system on the level of mental functions, and whether it is a digital system on the level of cell properties and interactions. Many neuroscientists think of the latter in digital computational terms. [Piccinini now argues that "spikes" in neural activity do not constitute digital states. (35-37)] Computationalists think that representational function makes the mental level a digital computational system as well. Reproducing it in an AI computer system would thus yield mental properties.

## REFERENCES


1. Shagrir O. Why we view the brain as a computer. Synthese. 2006;153(3):393-416.
2. Dretske F. Knowledge and the flow of information. Cambridge, Mass.: MIT Press; 1981.
3. Dretske F. Naturalizing the mind. Cambridge, Mass.: MIT Press; 1995.
4. Millikan RG. Language: A biological model. Oxford: Oxford University Press; 2005.
5. Macdonald G, Papineau D, editors. Teleosemantics: New philosophical essays. Oxford: Oxford University Pres; 2006.
6. Dodig-Crnkovic G. Epistemology naturalized: The info-computationalist approach. APA Newsletter on Philosophy and Computers. 2007;6(2):9-14.
7. Dodig-Crnkovic G, Müller VC. A dialogue concerning two world systems: Info-computational vs. mechanistic. In: Dodig-Crnkovic G,

Burgin M, editors. Information and computation: Essays on scientific and philosophical understanding of foundations of information and computation. Boston: World Scientific; 2011. p. 149-84.
8. Sandberg A, Bostrom N. Whole brain emulation: A roadmap. Oxford: Future of Humanity Institute; 2008.
9. Sandberg A. Feasibility of whole brain emulation. In: Müller VC, editor. Theory and Philosophy of Artificial Intelligence. Berlin: Springer; 2013. p. 251-64.
10. Harnad S. The symbol grounding problem. Physica D. 1990;42:335-46.
11. Raftopoulos A, Müller VC. The phenomenal content of experience. Mind and Language. 2006;21(2):187-219.
12. Raftopoulos A, Müller VC. Nonconceptual demonstrative reference. Philosophy and Phenomenological Research. 2006;72(2):251-85.
13. Goodman N. Languages of art. Indianapolis: Bobbs-Merril; 1968.
14. Lewis D. Analog and digital. Nous. 1971;5(3):321-7.
15. Haugeland J. Artificial intelligence: The very idea. Cambridge, Mass.: MIT Press; 1985.
16. Haugeland J. Analog and analog. Philosophical Topics. 1981;12:213-26.
17. Pylyshyn ZW. Computation and cognition. Cambridge, Mass.: MIT Press; 1984.
18. Shagrir O. Two dogmas of computationalism. Minds and Machines. 1997;7:321-44.
19. Blachowicz J. Analog representation beyond mental imagery. The Journal of Philosophy. 1997;94(2):55-84.
20. Negroponte N. Being digital. New York: Vintage; 1995.
21. Boghossian PA. Fear of knowledge: Against relativism and constructivism. Oxford: Oxford University Press; 2006. 139 p.
22. Turing A. Computing machinery and intelligence. Mind. 1950;LIX:433-60.
23. Searle JR. Mind: A brief introduction. Oxford: Oxford University Press; 2004. 224 p.
24. McCormack P, editor. Starmaking. Cambridge, Massachusetts: MIT Press; 1996.
25. Demopoulos W. On some fundamental distinctions of computationalism. Synthese. 1987;70:79-96.
26. Pylyshyn ZW. Computing in cognitive science. In: Posner MI, editor. Foundations of cognitive science. Cambridge, Mass.: MIT Press; 1989. p. 49-91.
27. Harnish RM. Minds, brains, computers: An historical introduction to the foundations of cognitive science. Oxford: Blackwell; 2002.
28. Floridi L. The method of levels of abstraction. Minds and Machines. 2008;18(3):303-29.
29. Müller VC. Representation in digital systems. In: Briggle A, Waelbers K, Brey P, editors. Current issues in computing and philosophy. Amsterdam: IOS Press; 2008. p. 116-21.
30. Boden MA. Escaping from the Chinese room. In: Boden MA, editor. The philosophy of artificial intelligence. Oxford: Oxford University Press; 1990. p. 89-104.
31. Boden MA. Mind as machine: A history of cognitive science. Oxford: Oxford University Press; 2006. 1704 p.
32. Haugeland J. Syntax, semantics, physics. In: Preston J, Bishop M, editors. Views into the Chinese room: New essays on Searle and artificial intelligence. Oxford: Oxford University Press; 2002. p. 379-92.
33. Krohs U. Der Funktionsbegriff in der Biologie. In: Bartels A, Stöckler M, editors. Wissenschaftstheorie: Texte zur Einführung. Paderborn: Mentis; 2007. p. forthcoming.
34. Berk V, Giani A, Cybenko G. Detection of covert channel encoding in network packet delays. Dartmouth College Department of Computer Science, Technical Reports. 2005(TR 536).
35. Piccinini G. The first computational theory of mind and brain: A close look at McCulloch and Pitts's logical calculus of ideas immanent in nervous activity. Synthese. 2004;141(2):175-215.
36. Piccinini G. Digits, strings, and spikes: Empirical evidence against computationalism. NA-CAP Conference; 28.07.2007; Chicago2007.
37. Piccinini G, Bahar S. Neural Computation and the Computational Theory of Cognition. Cognitive Science. 2012:1-36.

# The 'simple-minded' metaphor: Why the brain is *not* a computer, via a defence of Searle

**Dr Yasemin J. Erden** [1]

**Abstract.** This paper offers a defence of Searle's argument that computation is observer relative. In so doing it will show why observer relativity is a necessary requirement for all applications of concepts to objects. A consequence of this account will be to critique the metaphor of brain as digital computer, and highlight why metaphors of this sort lack epistemic content. This paper offers a defence of Searle's original account of computation, by presenting a Kantian position on the necessary unity of consciousness in relation to objects and concepts.

## 1 INTRODUCTION

In 1990 Searle gave his Presidential Address to the APA. In this talk he attempts to answer the controversial question: 'Is the Brain a Digital Computer?' [1]. Unsurprisingly, his answer (which was of course negative) is widely disputed [2-3], and the debate rages on. This paper takes up the baton by challenging both the parameters of the question, and the assumptions upon which it relies. I will take Searle's critique as the starting point for this paper, but in so doing I will go on to offer further philosophical grounding, via Kant, for the arguments at the heart of the matter.

The central tenet of the paper will thus be twofold. First will be to show why the analogy between brain and computer is little more than a poorly constructed, yet highly creative and seductive metaphor. Second will be to extend Searle's defence of the *observer standpoint* with regards computation, by showing why it is impossible to identify any such idea of computation without the *identifiers*. Were there to be no metaphor-writing, question-posing, interpretative beings, who *can* identify patterns and systems, there could not be any judgements of any sort, whether about computation or anything else.

## 2 'THE BRAIN AS COMPUTER' AS METAPHOR

The idea of the brain as computer permeates modern thinking, culture, science and theory. My wish is not to engage directly with the content of that debate, but rather to show why the metaphor is problematic. To this end, Searle's reply to the question of whether *the operations of the brain can be simulated on a digital computer* is useful. He accepts, as do I, that such operations can indeed be *simulated*, but by this he accepts no more than he would when accepting that weather systems too, can be simulated. The relation is comparative. In both cases the simulations stand in place of the events that we either believe or predict have happened or could happen (whether they are likely or not).

In this way, simulations function in a manner akin to *metaphors*, where words, ideas, concepts, and images stand in place not of actual happenings in their entirety, but as we predict or understand them to be. What is different is that we expect a simulation to have a higher degree of accuracy than a metaphor, which by its very nature is sometimes taken to have little content beyond its literary or creative powers. In actual fact, metaphors pervade our language, and offer us the tools for understanding the world in ways that would otherwise be inaccessible to us. As I explain elsewhere [4], metaphors offer a method of comparison by which we may view new aspects or insights.

The metaphor is developed by us, and used in our understanding, and is thereby recognised as additional information, rather than as information about the intrinsic properties of an object or concept. So, if I claim to *wander lonely as a cloud*, I do in fact make a make a comment about my passage through time and space. This does not commit me to any position regarding the putatively intrinsic qualities of either myself or of clouds, where any such attempt would require focus on supposed internal properties of myself or of clouds, independent of external conditions, or even just as separate from the metaphor. The metaphor exists in the shared space now created between these otherwise disparate concepts or objects.

In a similar vein, Searle notes that the identification of a process *as* computational 'does not identify an intrinsic feature of the physics, it is essentially an observer relative characterization,' adding that 'nothing is intrinsically a digital computer solely in virtue of its physical properties' [1]. Thus to recognise that something is functioning *as* something, is to use the *as* in just this sort of comparative or metaphorical sense. This works in practical terms too, since in using my hand *as a plate*, is to do little more than recognise that in eating food from it, it is now functioning in a similar way as a plate in such circumstances. In this, as in other such observations, the comparison between them is relative to my having observed the relation. The reason this does not engender any danger of a multiplicity of inaccessible first-person subjective experiences of the world, is that when I utter such comparisons (e.g. about the loneliness of clouds), it would be easy enough for others to understand what I mean by this. Or at least, to imaginatively engage with what I might mean. For similar reasons the accusation of relativity can be avoided, but I will come to this later.

Searle's comment that 'we could not discover objects in nature which were functioning as chairs, except relative to some agents who regarded them or used them as chairs' [1] is instructive at this juncture. As too is Kant's [5, B137/138] explanation that,

> The first pure knowledge of understanding, then, upon which all the rest of its employment is based, and which also at the same time is completely independent of all conditions of sensible intuition, is the principle of the original *synthetic* unity of apperception. Thus

[1] St Mary's University College, London, UK
✉ erdenyj@smuc.ac.uk

> the mere form of outer sensible intuition, space, is not yet [by itself] knowledge; it supplies only the manifold of *a priori* intuition for a possible knowledge. To know anything in space (for instance, a line), I must *draw* it, and thus synthetically bring into being a determinate combination of the given manifold, so that the unity of this act is at the same time the unity of consciousness (as in the concept of a line); and it is through this unity of consciousness that an object (a determinate space) is first known. The synthetic unity of consciousness is, therefore, an objective condition of all knowledge. It is not merely a condition that I myself require in knowing an object, but is a condition under which every intuition must stand in order *to become an object for me*. For otherwise, in the absence of this synthesis, the manifold would *not* be united in one consciousness.

It is the observer who is active in both the *drawing* of the line, and in the *recognition* of it as such, and this is a condition for any knowledge we can have of lines. Without this experience there can be no line to which to refer. Without any human experience, there is no knowledge of *lines* (whether as concept, or as represented in space) in any way that we could make sense of. Searle makes a similar point when he notes that 'Computational states are not discovered within the physics, they are assigned to the physics' [1], and that 'There is no way you could discover that something is intrinsically a digital computer because the characterization of it as a digital computer is always relative to an observer who assigns a syntactical interpretation to the purely physical features of the system' [1].

The example offered by Searle, of the impossibility of an 'unknown sentence' in ones head, is key in this. As he rightly points out, a sentence requires active construction, use, and recognition. Rather like the drawing of a line. Repetition can certainly embed sentences within our languages, such that they require little apparent thought in their uttering (for example, 'how are you?' can elicit a 'fine, thanks, how are you?' with minimal genuine consideration to the question), but this shows little more than the habitual aspects of language use. Even if it sometimes seems otherwise, sentences are created, not found. This is true also for patterns, which, if not created, are identified as such by observers:

> the only sense in which the specification of the pattern by itself provides a causal explanation is that if you know that a certain pattern exists in a system you know that some cause or other is responsible for the pattern. So you can, for example, predict later stages from earlier stages [1].

What tricks us into confusing how the relation lies, is the *meaning* of the words,

> We are blinded to this difference by the fact that the same sentence, "I see a car coming toward me", can be used to record both the visual intentionality and the output of the computational model of vision. But this should not obscure from us the fact that the visual experience is a concrete event and is produced in the brain by specific electro-chemical biological processes. To confuse these events and processes with formal symbol manipulation is to confuse the reality with the model.

Thus the question with which Searle has engaged is not simply to be refuted. It needs to be discarded as 'ill defined'. Kant [5, A244/B302] once again proves instrumental here:

> So long as the definition of possibility, existence, and necessity is sought solely in pure understanding, they cannot be explained save through an obvious tautology. For to substitute the logical possibility of the concept (namely, that the concept does not contradict itself) for the transcendental possibility of things (namely, that an object corresponds to the concept) can deceive and leave satisfied only the simple-minded.

## 3 THE NECESSARY CONTINGENCY OF OBSERVER AND OBJECT

Key to Searle's account of computation is the claim that 'syntax is essentially an observer relative notion'. As he notes, 'The ascription of syntactical properties is always relative to an agent or observer who treats certain physical phenomena as syntactical' [1]. Furthermore,

> The multiple realizability of computationally equivalent processes in different physical media was not just a sign that the processes were abstract, but that they were not intrinsic to the system at all. They depended on an interpretation from outside. We were looking for some facts of the matter which would make brain processes computational; but given the way we have defined computation, there never could be any such facts of the matter. We can't, on the one hand, say that anything is a digital computer if we can assign a syntax to it and then suppose there is a factual question intrinsic to its physical operation whether or not a natural system such as the brain is a digital computer. [1]

This idea of the *requirement of interpretation from outside* is one that I will focus on here. The first point to note is that the definition of computation can only be applied to systems that we *recognise* as computational. While this insubstantial statement is not in itself controversial, disagreements hinge on the value of this term *recognition*, or its cousin *observation*. Opponents of Searle sometimes take his claim about the contingency of observation to be naive. Endicott [6, p. 104], for instance, rejects Searle's account of computation as too simplistic. In its place he claims that 'a system is a genuine computational device when there is a correspondence between its physical states and its formal states such that the causal structure of the physical system is isomorphic to the formal structure of the computational operations' [6, p. 104]. His refutation of Searle seems to hinge on a seemingly Platonic account of computation, whereby the system or computation has an identity as a thing-in-itself, which we then *discover*. As already noted above, this is problematic for a number of reasons.

For a start, Searle's claim is in fact far more complex than that, and points to the very core of the meaning that we understand by a term such as computation. The Platonic identity is flawed precisely because the expectation is that meaning is there to be *found*, rather than *determined.* For instance, where we recognise the pattern of some migrating bird formation as akin to a 'v' shape, we are not saying that there are shapes in the world that *are* 'v', which await our discovery. Instead, the comparison, or the metaphor, is drawn between the formation of birds in flight, and the letter 'v' that we use in language. It is clear that v-*ness* (where v is a symbol) is not somehow inherent to what it is to migrate, but rather that in our interpretative understanding of the world, we make sense of what we understand based on what we have already learned. Kant's [5, A258/B313-314] account of phenomena and noumena is useful here:

> When, therefore, we say that the senses represent objects *as they appear*, and the understanding objects *as they are*, the latter statement is to be taken, not in the transcendental, but in the merely empirical meaning of the terms, namely as meaning that the objects must be represented as objects of experience, that is, as appearances in thoroughgoing interconnection with one another, and not as they may be apart from their relation to possible experience (and consequently to any senses), as objects of the pure understanding. Such objects of pure understanding will always remain unknown to us; we can never even know whether such a transcendental or exceptional knowledge is possible under any conditions—at least not if it is to be the same kind of knowledge as that which stands under our ordinary categories.

It does not seem fair to attribute to Searle a position from which is offered an irrefutable definition of computation. By which I mean his point seems not to be centred on a declaration of what computation *is* but only to show what it *isn't*, i.e. an independently and objectively verifiable system that exists *independent* from our system of its interpretation as such. The added criticism offered by Endicott [6, p. 107] that there are 'multiply realised types within the domain of physical or natural science', which are not observer relative, is also refutable. Once again, the way in which we understand such systems to operate requires that we understand there to be a system as such. There is no noumenal objective reality about there being a system, and even if there were, we would know nothing about it. There is no way in which we can remove ourselves, and our interpretation of things, including our understanding of systems, such that we can *know* that a system simply *is*, in any objective sense of that term. Thus, simply claiming that there *are* these systems, which are somehow separate to our interpretation of them as such, makes no sense at all. If, for example, we recognise the system of flowers and bees and pollination, where do we recognise the system to lie? Do we recognise the system from the viewpoint of the flower, or that of the bee, or of the two as a combined system?

Yet none of this need engender accusations of relativism or anti-realism. The question of *how* anything can be thought to have an objective quality (such as mass, identity shape) is answered by Kant [5, B276-277] when he notes that it is only *by means* of 'outer experience' that 'inner experience' is possible. Rejecting claims of foundational Cartesian subjectivity, he notes [5, B277] that while 'the representation' *I am* may include the '*existence* of a subject' (emphasis added) it includes no '*knowledge* of that subject, and therefore also no empirical knowledge, that is, no experience of it'. To which he adds:

> For this we require, in addition to the thought of something existing, also intuition, and in this case inner intuition, in respect of which, that is, of time, the subject must be determined. But in order so to determine it, outer objects are quite indispensable; and it therefore follows that inner experience is itself possible only mediately, and only through outer experience.

Elsewhere he expands on the point with respect to our understanding [5, A820/B848]:

> The holding of a thing to be true is an occurrence in our understanding which, though it may rest on objective grounds, also requires subjective causes in the mind of the individual who makes the judgment. If the judgment is valid for everyone, provided only he is in possession of reason, its ground is objectively sufficient, and the holding of it to be true is entitled *conviction*…
>
> The touchstone whereby we decide whether our holding a thing to be true is conviction or mere persuasion is therefore external, namely, the possibility of communicating it and of finding it to be valid for all human reason.

To put this another way, if there is nothing about which your judgements *are*, then there is no particular *reason* to make one judgement over another [7]. Were there to be no objective grounds, explains McDowell [8, p. 67], all we would be left with would be a 'frictionless spinning in the void'. Another highly instructive metaphor, but not one which anyone would likely demand we take to be true in any apparent *objective* sense.

## 4 CONCLUSION

This paper sought to add further weight to Searle's arguments regarding the necessary relation between observer and computation. It has not spent much time engaging with the standard objections (of which there are many), since the ideas offered from Kant, cited throughout, should hopefully circumvent some of these more typical objections. A failure to recognise the metaphorical nature of the term *computation* is itself indicative of a broader failure to recognise how language of this sort relies on context for meaning (as I discuss elsewhere [9]). The origins of the term *computation* as applied to humans, and its later application to machines should have given some indication of this fluency of meaning, and employment of metaphor. Where this has failed, perhaps Kant will prove instructive. Especially since it should by now be clear that the very possibility of my recognition of this as such relies on my skills for recognising metaphors, systems and patterns, and in offering judgements and interpretations that (hopefully) may be agreeable to others who recognise the same. It is interesting therefore that Copeland is dismissive of this, when he notes:

'Searle is telling us no more than that if the brain is a computer, then it is so only in the sense in which all other computers are computers. This is hardly interesting' [9]. I disagree entirely. It is precisely because he does show the limitations of such an interesting metaphor that Searle's ideas are themselves most interesting indeed.

## ACKNOWLEDGEMENTS

I am extremely grateful to Stephen Rainey for his assistance in navigating the complex and often unsettling world of Kant. I am indebted to Mark Bishop, whose ideas and suggestions sparked the inception of this work, and whose constructive feedback on an earlier draft of this paper helped shape its direction. Thanks are also due to the reviewers of this paper who were highly instructive and generous in their comments. It goes without saying, but I shall say it anyway, that any and all faults and failures remain entirely my own.

## REFERENCES

[1] J. R. Searle, Is the brain a digital computer?, *Proceedings and Addresses of the American Philosophical Association*, 64: 21–37, 1990. Cited version online (accessed 05/02/2013): https://mywebspace.wisc.edu/lshapiro/web/Phil554_files/SEARLE-BDC.HTM

[2] D. J. Chalmers, Does a rock implement every finite-state automaton?, *Synthese*, 108(3): 309–333, 1996.

[3] R. L. Chrisley, Why everything doesn't realize every computation, *Minds and Machines*, 4(4): 403–420, 1995.

[4] Y. J. Erden, Wittgenstein on simile as the 'best thing' in philosophy, *Philosophical Investigations*, 35(2): 127-137, 2012.

[5] I. Kant, *Critique of Pure Reason* (trans. N. K. Smith), London: Macmillan press, 1933 (2nd ed.).

[6] R. P. Endicott, Searle, syntax, and observer relativity, *Canadian Journal of Philosophy,* 26(1): 101-122, 1996.

[7] S. Rainey, *A Pragmatic Conception of Rationality*, Doctoral Thesis, QUB, 2008 (unpublished).

[8] J. McDowell, *Mind and World*, Camb. Mass.: Harvard Uni. Press, 1994.

[9] Y. J. Erden, Could a created being ever be creative? Some philosophical remarks on creativity and AI development, *Minds and Machines*, 20(3): 349-362, 2010.

[10] B. J. Copeland, What is computation?, *Synthese*, 108(3): 335–359, 1996.

# Kinds and Limits of Computation

**John Preston**

**Abstract.** I focus on conceptual questions about how far the notion of computation extends, and whether there are importantly different kinds of computation. After specifying what 'computation' originally meant (a kind of intentional activity), and noting that Alan Turing modified that concept to give us the basic modern sense of 'computation', I argue that computation isn't a *natural* kind. The many things now called 'computation' can be divided into kinds in various ways, for most of which the reason why they count as computation is clear. When it comes to 'hypercomputers', though, whether *they* still count as computational should probably depend on whether their deliverances are checkable.

## 1 INTRODUCTION

There are various different sorts of questions about what kinds of computation there are, and what their limits might be. Notably, computer scientists ask some such questions, which tend to be of a technical kind. Some of these are questions of a *mathematical* nature, such as: Are all functions computable? If not, which kinds are computable, and which aren't? Which kinds of machines can compute which kinds of functions? (see, e.g., [1]). Others are technical questions of a *physical* nature, such as: How much energy must be expended to perform a particular computation? How quickly might it be performed? How small might the computing device be? How efficient might it be? How much information-storage capacity might it have (see [2], [3])?

The questions I have in mind today aren't these technical ones but rather they are *conceptual* questions, in particular the questions: How far does the concept of computation extend? That is, how far can we (e.g., computer scientists, mathematicians) legitimately extend the concept? Does doing so invoke importantly different *kinds* of computation? If so, how are those kinds related to one another? How do they constitute *different* kinds? Is there only *one* concept of computation, or are there different such concepts?

## 2 CONDITIONS ON APPLYING THE ORIGINAL NOTION OF COMPUTATION

Four important features of the strict and literal application of the notion of computation in its *original* mathematical sense (and cognate notions such as computing, computer, etc.) are (a) that inputs should be supplied to, and outputs delivered by, an identifiable system or process; there must be, as it were something that's *doing* the computing (b) that there should be some *function* (that is, some *mathematical* function) whose inputs are supplied and whose outputs are yielded by the system or process in question, (c) that the system or process in question deliver those outputs as the result of rule-governed symbolic manipulations (*by computing them*, as it were, not merely by causing them in some way or other), and (d) that the results of the operation are humanly *checkable*, that is, that we can, at least in principle, come to have confidence that the process in question has generated the *correct* results. Feature (d) is an important part of this original conception, and takes its place there because 'to compute' is a *success-verb* (see section 5, below), and success in mathematics is determined by *proof*.

I will argue that at least two of these four features are being relaxed by *something* that now counts, or something that some computer scientists would now *like* to count, as a computer.

## 3 HUMAN (CORE) COMPUTATION AS AN INTENTIONAL KIND

What humans do when they compute functions is what I shall call computing in the *core* sense. On my (non-computationalist) view, this is something that humans occasionally do, but digital electronic computers never literally do. If one had to specify what kind of activity computing in this core sense is, the answer would be that it's an *intentional* kind. That is, this kind of computation is an activity in which humans engage when they have an *aim* in mind (the computation of a particular function), and take their mathematical manipulations to be directed at this aim. Unsurprisingly (since they were made for one another) his kind of computation exhibits all four features of the strict and literal application of the notion, since the 'system' is the person plus any *aides-memoires* they are using, they are *trying* to compute a particular function, they arrive at its solution by moving from symbols to others symbols in a *rule-guided* way (whether mentally or on paper), and we can *check* the results of their activity.

## 4 TURING-STYLE COMPUTATION

The notions of computing or computation, and computations, though, have *changed*, and are no longer this same intentional kind. This change was initiated by Alan Turing. In his famous paper on the *Entscheidungsproblem* ([4]), Turing stipulated that the operations of his 'computing machines', and thus of all those machines we now think of as based on 'Turing-machines' can be broken down into elementary steps such as: scanning a square on a machine-tape, registering the contents of that square, erasing the symbol in the square, writing another such symbol, etc.. Turing's paper thus involves the idea that the sequence of states in a computation can be put into correspondence with the elementary pre-arithmetical operations of a human 'computer' (that is, a human being who is computing in the *core* sense). I will call this 'Turing-style' computation.

[1] Dept. of Philosophy, University of Reading, RG6 6UD, UK. Email: j.m.preston@reading.ac.uk.

Because computing machines are *causal* processors rather than *rule-followers* (in the original sense of *that* phrase), following Turing already involves relaxing feature (c) in my specification of the core sense of computation. Whether we should say (as the advert for this conference has it) that Turing 'refined' the concept of computation, or rather that his work had the effect of *redefining* it, the crucial thing is that he changed the concept (or, if you wish, created *another* concept of computation) in such a way that made computer *science* possible. And there can be no objection to his having done so, except insofar as we confuse ourselves if we think of 'computing' as a *single* kind of activity that people engage in (occasionally and ponderously) and that digital electronic computers engage in (most of the time, and very swiftly). The reasons I think this would be a confusion aren't crucial here (see [5]), but the point I want to make is that the notions of computing and computation, which used to pick out a single intentional kind, now pick out kinds of a *different* kind. What I will argue is that the current dispute about the possibility of 'hypercomputers' suggests that we have a *choice*, a choice between conceiving of computation as a purely *functional* kind (where 'function' no longer refers to *mathematical* functions, of course), and rejecting the very idea of hypercomputers because we can't quite reconcile ourselves to thinking of computation in this very 'thin' way, and we still want to insist on another important feature of the original conception.

## 5 COMPUTATION ISN'T A *NATURAL* KIND

Computationalists tend to treat the notion of computation as a *natural kind concept*. As a result, when they ask themselves about the limits of computation they try to specify what all such phenomena really have in common. Thus they often feel forced to look to *physics* for their account of computation, since physics is, as it were, the science of what every kind of computational system or process would have to have in common.

I don't think we should think that computation is a *natural* kind at all. One reason why computational kinds can't be natural kinds is that the concepts of computing and computation have a *normative* dimension: to 'compute (a function) *incorrectly'* is, strictly speaking, unsuccessfully to compute it, that is, *not* to compute it (just as to add two numbers together incorrectly isn't really to *add* them at all). The verbs in question here are *success-verbs* (like the verb 'to know', where one can't know that p unless p is the case). Natural kinds (like lemons, tigers, gold, planetary systems, etc.) can of course be subject to a distinction between what's typical or *normal* for that kind (tigers are four-legged, lemons are sour, planets move in ellipses) and what's atypical or abnormal. But the kinds in question aren't themselves *normative*. This is easy to see in the planetary systems example: if we found such a system whose objects weren't related to one another in roughly the way that Kepler's laws specify, we could *still* count it *as* a planetary system. It wouldn't fail to count as such just because it was misbehaving, as it were. Kepler's laws are putative *natural laws*, not *norms*.

## 6'BEYOND'TURING-STYLECOMPUTATION?

When one looks at the literature on computation today, one finds a large array of things called 'computation'. (And the first thing to say is that Turing himself envisaged several of them). Some of them (but only *some* of them) are in some way supposed to 'go beyond' the now ordinary Turing-style of computation. Here's a list I managed to generate from my own collection of books and articles on computer science:

Bio-molecular computation (including Biological computation, Evolutionary computation, Genetic algorithms, Molecular computation, Peptide computation, DNA computation)
Neural Computation (incl. parallel computation)
Computation by Dynamical Systems

Quantum Computation

Hypercomputation

Why should we think that all the phenomena covered by these terms are cases of *computation*? The answer is different in different cases. There are different ways of dividing this totality into kinds, but if one divides it into the kinds most significant to my purposes here, one finds that members of the first group (computation by biomolecular, neural, and dynamical systems) count as computation for *one* reason, whereas 'hypercomputation', *if* it counts as computation, does so for a different reason. (Quantum computation seems to be something of a hybrid).

## 7 NON-STANDARD TURING-STYLE COMPUTERS

I've already said that I think there's now more than one kind of computation (the Turing-style kind, which we use everyday, and the old-fashioned one). So I'm committed to there being *two* kinds of computation. Nevertheless, all the different things that count as *machine*-computation (Turing-style and other) might count as such for a *single* reason. Machine-computation itself could still be a single kind.

In fact, I think this isn't *quite* the case. That is, I think almost all of them count as computation in the post-Turing sense because machine-computation still exhibits features (a), (b), and (d) from our original conception.

Members of the *first* group ('biomolecular' computation (e.g., peptide computation, DNA computation, molecular computation)) count as computers performing computations because, roughly, they all involve systems or processes which, when supplied with inputs, can reliably generate the same outputs as those of our everyday computers. (And those everyday computers count as computers because, when supplied with inputs, they can reliably generate outputs of the same kind as *us*). So these count as computers because they *don't* really go beyond Turing-style computation. They involve no further relaxation of the conditions on computation, although they *do* feature unusual, non-standard *systems* performing the computations. That is, when one reads papers on computing using DNA, for example (like [6], [7]), what one finds amazing and impressive is that anyone could arrange a process like *that* in such a way that a search-problem gets solved.

## 8 *REALLY* BEYOND TURING-STYLE COMPUTATION: HYPERCOMPUTATION

The envisaged devices in the final group, though, 'hypercomputers', can't count as performing computations for the same reason as these non-standard Turing-style computers do, since by definition they compute functions *beyond* those that Turing-machines are capable of computing. This means that we can't think of their operations as being such that they all can be put into correspondence with the elementary pre-arithmetical operations of a human 'computer' (that is, a human being who is computing in the *core* sense). Turing dramatised this in his famous paper on ordinal numbers ([8]) by calling the devices in question 'oracles', thereby bringing out what I would like to call their *inscrutability*. He said of these oracles that they're *not* machines (i.e., not Turing-machines), although when combined with a machine they yield what he calls an '*o*-machine' (oracle-machine), and that they work in an 'unspecified' way ([9], p.156). He didn't call these oracles 'computers' and there's no evidence that he thought of them as computational. What he says about them amply reflects the fact that he named them '*oracles*': we have, and are supposed to have, no idea how they work.

We *could* count these hypercomputers, *o*-machines (if we had any) as computers merely because conditions (a) and (b) of my specification of the core sense of computation still hold good for them. That is, there's a *system* doing the processing, a *function* is being computed, and the system (inscrutably but) consistently delivers what we take to be the solution to the computation in question. *This* would be to think of computation in *purely* practical, functional terms.

It's notable, though, that computer scientists are strongly divided about this way of thinking. In one corner, some advocates of hypercomputation like Jack Copeland ([10], [11], [12], [13]) advertising *o*-machines as one kind of hypercomputer, speculate on what computation might be like when we actually devise such a thing. Others, less concerned with *o*-machines specifically, want to insist on the possibility of hypercomputers even if these aren't what Turing had in mind ([14], [15], [16], [17], [18]). In the other corner, though, traditionalists such as Martin Davis and Andrew Hodges ([19], [20], [21]) vigorously resist the idea that *o*-machines are a forgotten kind of machine that Turing envisaged, or even that there could be such a thing as hypercomputation at all. For them, the idea of the *o*-machine merely plays a role in Turing's paper. Its role there is to show that *if* we had a device that could solve any given number-theoretic problem, we could easily form a question about it that isn't number-theoretic, and thus which this machine couldn't itself decide. On this view, the idea of an *o*-machine is merely a 'what if…?', and in no way to be taken seriously as a kind of machine that Turing was proposing or envisaging.

When I said that the system delivers what we take to be the solution to the computation in question, my phrase 'what we take to be the solution to the computation' conceals an important issue, since it's not clear to me whether we could *check* the results of the hypercomputer's operation. If, as I've suggested, checkability is a still a core feature of our conception of computation, the concept of a hypercomputer may well be a contradiction in terms.

There are two scenarios to consider here. If we *can* somehow check the results of the hypercomputer's operation, conditions (a), (b), and (d) are still in place, it's just that the Turing-style conception of what's *involved* in those conditions has been revised. I suppose this would mean that elementary human arithmetical operations aren't the *only* ones that are involved in our computations. The idea might be that we have, in addition to these, a kind of mental process which is inscrutable by us, and *yet* whose deliverances we take to be reliable, perhaps what some people refer to as mathematical 'intuition' (this is an old-fashioned version of the view which I take Kurt Gödel and Roger Penrose to champion). Humans would, *pace* Turing, be able to compute *more* than Turing-machines can.

I have to say that I can't make sense of this possibility, since I think that any such faculty that humans have could only be the source of suggestions (intuitions) not a source of knowledge. Without the possibility of our being able to check the results of the alleged hypercomputation, it seems to me that the process in question should not really count as computation. So if there are 'computations' that can be carried out by hypothetical machines but which would be deeply *impossible* for humans to carry out *or* check, we can still ask computer scientists 'Why do you call *that* process "computation", and why should *we* call it that?". To rely here merely on the practical criterion of *functionality*: the machine delivers the goods, and we rely on them, or to rely merely on the fact that the machine's *other* operations are computations, would not be enough. Mathematics may contain intuition and speculation, but neither of these should be part of what we mean by the term *computation*.

So if we take seriously the four features of computation that I've suggested, and I think we still do, our sense that these are still *computations* ought to be fading out. They may be processes which will deliver outputs, but unless we have some way of knowing that (or at least some good evidence that) their outputs are the *correct* results of what would be computations, calling the processes 'computational' is at most a misleading honorific, not a serious classification of their nature.

However, if, as I've suggested, computation isn't a *natural* kind, the question 'Would *o*-machines (or other hypercomputers) be computers?' ultimately calls not for a discovery, but for a *decision*. Various factors may enter into that decision, but if the notion of computation is to retain its connection with mathematics, we should not allow its feature (d) to be quietly dropped. We may now be at the point in the stretching of our original concept where we're legitimately undecided between thinking that we're making merely another alteration to the concept of computation, and thinking that we've switched to (or are forging) *another* concept. I can't see that there is any fact of the matter to be captured here. For that to be the case, the concept of computation would have to be a *natural kind* concept, and it isn't. This emphatically *doesn't* mean that I think there are *no* factual issues between Copeland, Siegelmann, et al. on the one hand, and Davis, Hodges et al. on the other. Neither does it mean that the decision will be *arbitrary* (various sorts of considerations might be adduced on either side). All I want to suggest is that on the question 'Are these machines still doing *computations*?' a genuine decision would be called for, we can't somehow look to nature itself for the answer.

## 9. CONCLUSION: THE 'SCANDAL' OF COMPUTATION?

Immanuel Kant considered it a scandal in philosophy that we must accept the existence of things outside ourselves merely 'on trust', without proof (see the Preface to the 2nd edition of his *Critique of Pure Reason* (translated as [22]). While Martin Heidegger isn't a philosopher I normally have much time for, he got it right when he commented thus on Kant's supposed 'scandal': 'The "scandal of philosophy" is not that this proof has yet to be given, but that such proofs are expected and attempted again and again' ([23], p.249).

If one takes the view I recommend here, there is no 'scandal of computation', or even 'scandal of *hyper*computation'. These turn out to be like other philosophical scandals. However, many people *like* scandals, or the *idea* of scandals, so I won't be surprised to find a strong tendency to cling on to the idea that there is such a thing surrounding these notions.

## REFERENCES


[1] C. H. Papadimitriou. *Computational Complexity*, Addison-Wesley, USA, (1994).

[2] R. Landauer. Fundamental Physical Limitations of the Computational Process. In *Computer Culture: The Scientific, Intellectual, and Social Impact of the Computer*. H. R. Pagels (Ed.). New York: The New York Academy of Sciences (1984).

[3] C. H. Bennett & R. Landauer. The Fundamental Physical Limits of Computation, *Scientific American*, 253:38-46. (1985).

[4] A. M. Turing. On Computable Numbers, with an Application to the *Entscheidungsproblem. Proceedings of the London Mathematical Society, Series 2*, 42: 230-265 (1936-7). (Reprinted in [9]).

[5] J. M. Preston. What are Computers (If They're Not Thinking Things)?'. In *How the World Computes: Turing Centenary Conference and 8th Conference on Computability in Europe, CIE 2012, Cambridge, UK, June 2012, Proceedings*. S. Barry Cooper, A. Dawar & B. Löwe (Eds.). Berlin & Heidelberg: Springer-Verlag, (2012).

[6] L. M. Adleman. Molecular Computation of Solutions to Combinatorial Problems. *Science*, 266: 1021-1024 (1994).

[7] R. J. Lipton, DNA Solution of Hard Computational Problems. *Science*, 268:542-545 (1995).

[8] A. M. Turing, Systems of Logic Based on Ordinals. *Proceedings of the London Mathematical Society, Series 2*, 45: 161-228 (1938-9). (Reprinted in [9]).

[9] B. J. Copeland. *The Essential Turing: The Ideas that Gave Birth to the Computer Age*. Oxford University Press, Oxford, UK (2004).

[10] B. J. Copeland. Turing's O-Machines, Searle, Penrose and the Brain. *Analysis*, 58: 128-138 (1998).

[11] B. J. Copeland. Super-Turing Machines. *Complexity*, 4: 30-32 (1998).

[12] B. J. Copeland. Hypercomputation. *Minds and Machines*, 12: 461-502 (2002).

[13] B. J. Copeland. Hypercomputation: Philosophical Issues. *Theoretical Computer Science*, 317: 251-267 (2004).

[14] M. Stannett. X-machines and the Halting Problem: Building a Super-Turing Machine. *Formal Aspects of Computing*, 2: 331-341 (1990).

[15] M. Stannett. Computation and Hypercomputation. *Minds and Machines*, 13: 115-153 (2003).

[16] H. T. Siegelmann. Computation beyond the Turing Limit. *Science*, 268: 545-548 (1995).

[17] H. T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhäuser, Boston, USA (1999).

[18] H. T. Siegelmann. Neural and Super-Turing Computing. *Minds and Machines*, 13:103-114 (2003).

[19] M. Davis. The Myth of Hypercomputation. In *Alan Turing: Life and Legacy of a Great Thinker*. C.Teuscher (Ed.). Berlin: Springer Verlag (2003).

[20] M. Davis. Why there is no such Discipline as Hypercomputation, *Applied Mathematics and Computation*, 178: 4-7 (2006).

[21] A. Hodges. Did Church and Turing have a Thesis about Machines?. In *Church's Thesis after 70 Years*. A. Olszewski (Ed.). Berlin: Logos Verlag (2006).

[22] I. Kant. *Immanuel Kant's Critique of Pure Reason*. N. Kemp Smith (Trans.). London: Macmillan (1929).

[23] M. Heidegger. *Being and Time*. J. Macquarrie & E. Robinson (Trans.) Oxford: Blackwell (1962).

# Abstract Platforms of Computation

**Matthew C. Spencer**[1] and **Etienne B. Roesch** and **Slawomir J. Nasuto**[2]
**Thomas Tanay** and **J. Mark Bishop**[3]

**Abstract.** Computational formalisms have been pushing the boundaries of the field of computing for the last 80 years and much debate has surrounded what computing entails; what it is, and what it is not. This paper seeks to explore the boundaries of the ideas of computation and provide a framework for enabling a constructive discussion of computational ideas. First, a review of computing is given, ranging from Turing Machines to interactive computing. Then, a variety of natural physical systems are considered for their computational qualities. From this exploration, a framework is presented under which all dynamical systems can be considered as instances of the class of abstract computational platforms. An abstract computational platform is defined by both its intrinsic dynamics and how it allows computation that is meaningful to an external agent through the configuration of constraints upon those dynamics. It is asserted that a platform's computational expressiveness is directly related to the freedom with which constraints can be placed. Finally, the requirements for a formal constraint description language are considered and it is proposed that Abstract State Machines may provide a reasonable basis for such a language.

## 1 INTRODUCTION

Over the last 80 years, computing has developed considerably, but there is still much debate about what "computing" is. When Alan Turing broached the field in the 1930's, he provided a very precise understanding of computing machines and computable problems. However, as variations on Turing's original mechanisms have been explored, the original definitions have become less appropriate and modern computational praxis now bares little resemblance to the original conception. The gap between practice and theory in computer science has been noted in other works [8, 6, 5, 16] which have strived to provide broad reviews of the field and suggest modern approaches to the discourse of computing. This discourse will be further explored here with an aim to provide a unified framework for defining computation and positioning popular computing formalisms.

There appear to be two central issues within the computing community: computational expressiveness and the scope of what can be called computation. The former issue is a discussion about the scope of problems that can be addressed with different computational formalisms, beginning with classical Turing Machines [23] and discussing other, more advanced concepts [12, 24, 6, 15, 16, 9]. The latter issue accepts the varying scopes of computational expressiveness and the proliferation of computational paradigms and explores the boundaries of what computation might be [5]. Both of these issues will be explored here to help establish the depth and breadth of the proposed framework.

Finally, if there were a unified framework of computation, a formal abstract language for describing programs would be valuable. Modern programming languages are largely platform dependent but a number of abstract formalisms [19, 11, 14, 10] might provide possibilities for platform-agnostic program descriptions. Such program descriptions will be considered here.

The following paper will first review computational paradigms, starting with Turing Machines. This discussion will address notions of super-Turing Machine expressiveness with a specific discussion on modern views on interaction. Interactive computing discusses concurrent partially coupled systems which, when shifted to the continuous time domain, begin to resemble natural physical systems. To differentiate computation from physical processes, a constructive view on the boundaries of computation will be explored. From these previous ideas, a framework of abstract computational layers will be proposed. Notionally, this framework will encompass the breadth of computational paradigms. Finally, the requirements of an abstract program description language for this framework will be discussed.

## 2 CLASSICAL COMPUTING

### 2.1 Functions and algorithms

A class of functions, called "effectively computable functions", contains those functions that can be worked out through finite sequences of simple mechanistic operations. Prior to the 1930s, such sequences (known as algorithms), lacked a formal definition and it was poorly understood which functions were effective and which were not. In the 1930s, both Alonzo Church and Alan Turing approached this problem whilst addressing the *Entsheidungsproblem*. Aiming to introduce a formal description of algorithms, Church produced the $\lambda$-calculus and went on to show the unsolvability of the *Entsheidungsproblem* [3]. Simultaneously, Turing approached this same problem, but from a different angle. Taking seriously the notion of sequences of mechanical operations, Turing formalised algorithms with his "abstract machines" (which are now commonly known as Turing Machines (TMs)) and went on to provide what is generally accepted to be a more convincing proof of Church's result [23]. Ultimately, Turing Machines and the $\lambda$-calculus are equivalent formalisms for representing the class of effectively computable functions, a point which is embodied in the Church-Turing Thesis (CTT).

When initially conceptualized, the CTT described this class of effective functions and the algorithms used to solve them. These algorithms necessarily had the very particular property that they processed mechanically, from some input to some output through a finite sequence of simple, ordered operations. By "mechanistic pro-

[1] email: m.c.spencer@reading.ac.uk
[2] University of Reading, UK
[3] Goldsmiths, University of London, UK

cessing", it was understood that the sequences were carried out deterministically, and that no insight or agency could interfere in the processing. By "simple operations", it was understood that each operation in the algorithm's description was trivially performed by a human mathematician. It was on this basis that Turing conceived of his machines.

## 2.2 Turing machines

Turing's *abstract machine* involved a tape (a bidirectional, arbitrary length, linear string of symbols), a read/write head pointing to a location on the tape, a set of possible configurations in which the machine could exist, and a table of instructions that could move the head, read and write the tape, and change the configuration of the machine. The machine would proceed mechanically, looking up an instruction in the table using the current configuration and the input symbol on the tape. Then, this instruction would be executed, potentially resulting in modifying the symbol on the tape, changing the machine's configuration, or shifting the tape one step to the left or right. The machine then proceeded to look-up the next operation from the table and so on until the machine would finally halt. Between the instructions in the table and the set of available configurations, each machine could embody a single algorithm. To execute the algorithm, the input was provided on the tape and the machine processed until halting. Once the machine had terminated, the tape would contain the output of the algorithm. While termination was not an absolute necessity (specifically in the case of computing numbers to arbitrary precision), the full answer could not be known unless the machine halted.

For the sake of the topic of this paper, the Turing Machine can be thought of as machine providing a certain intrinsic and deterministic dynamics. These dynamics are predicated on the ability to execute a small set of elementary operations, specifically, reading and writing tape symbols, moving the tape, reading the current configuration, and changing the configuration. If any of these operations were non-trivial for the executor, a Turing Machine would not, on its own, provide a satisfactory formal representation of an algorithm. Turing Machines are, by their very conception, abstract entities. They were not designed to be actually implemented, and yet, if each of the elementary operations is appropriately simple, implementation as a physical machine could be done by embedding the TM design in a physical form.

Further to the intrinsic dynamics of TMs, they provide a flexible approach to customizing the constraints on those dynamics through the set of configurations and the writing of the instruction table. It is through this customization that the general class of TMs can instantiate a TM for a specific algorithm. However, there are natural limitations intrinsic to the TM formalism which cannot be overcome through this customization (further discussed in Section 2.3). These limitations were not an oversight, but were necessary for formally representing effectively computable functions. However, modern digital computers do not exclusively describe this class of functions.

Later in his career, Turing considered several extensions to the TM concept. An important extension was that of the Universal Turing Machine (UTM) – a Turing Machine capable of simulating any other Turing Machine. From this idea, universal computers (for computing more than a single function) were devised, ultimately resulting in modern computers. However, since modern computers have a finite memory, it is argued that they are strictly less powerful than UTMs, which may have a tape of arbitrary length. Nonetheless, it has been argued [5] that modern computers can be (and regularly are) used for more than the computation of functions, and are therefore *more* expressive than UTMs. These views will be discussed in Section 3.1.

Other extensions that Turing considered were those that incorporated non-mechanical components into the machine's structure; in other words, including non-trivial operations. Two such components that are popularly discussed are so-called "oracles" and human users. In Oracle Machines, the TM is able to consult an all-knowing oracle, which is capable of delivering a non-trivial answer in finite time, and while such Oracles do not exist in practice, they do serve to describe an abstract computing paradigm of consulting an outside expert. Such consultation is also represented in Turing' Choice Machines, in which the TM might pause occasionally to query a human user for additional input. In both of these cases, the TM architecture has been extended to include non-mechanical components and would not be what Turing considered to be "automatic machines" or "purely mechanical".

## 2.3 Restrictions of Church-Turing formalisms

While Turing Machines provide a robust and formal description of algorithms, they feature a number restrictions which define the space of their applicability. However, owing to these in-built restrictions, other, more computationally expressive paradigms can be imagined. One of the central restrictions to the TM paradigm is the notion of operating on a finite alphabet of symbols. Since the alphabet is finite, it cannot represent all real numbers or other continuous concepts. This has implications for certain types of scientific computing where arbitrary precision is desirable. This also has implications for considering TMs as capable of intelligence, owing to limited and inflexible nature of the alphabet. Further, TMs receive input and emit output encoded by the same alphabet, reducing the types of functions that can be implemented (such as are found in the broader set of bijective mappings).

To consider other restrictions, one must take a larger view of the TM as a computational machine embedded within an environment containing at least one other agent. This other agent, the user of the TM, is implicit in the TM's definition since the input has to arrive on the tape from somewhere and the output must be requested for some purpose that is not the TM's own. In this wider view, other restrictions become clear, namely the temporal insensitivity and the synchronicity of the input and output. In other words, once the user has provided input to the TM, they must wait some time for the TM to complete its operation before it delivers output; additional input cannot be supplied as the TM operates. Also, the input and output must be specified in the same alphabet. Furthermore, for every input there is guaranteed to be a single output, so concepts including multi-input/multi-output or streaming interaction with the environment cannot be modelled. Finally, a TM lacks memory that persists between inputs, which prevents it from modelling a learning system.

# 3 HYPER-COMPUTABILITY

It is from the class of effectively computable functions, which the notion of "computability" takes its generally understood meaning. If a function is effectively computable, it is computable and has an associated algorithm for computing it. Likewise, it also has a Turing Machine or $\lambda$-calculus representation. Similarly, if it is not computable, none of these other representations apply.

The formal definition of "computable" is generally taken to be "that which is computable by a Turing Machine". Thus, any machine formalism that is more expressive or capable than a Turing Machine

is generally called a "hyper-computer" and is capable of "hyper-computation", though perhaps adhering to the term "super-Turing" is clearer. One example is the Zeno-machine, which performs each subsequent operation in half the time of the previous one, appealing to the Zeno paradox to perform an infinite number of operations in a finite span of time. Likewise, most hyper-computers include the notion of infinity into their construction (infinite length alphabet, infinite speed, infinite knowledge, etc) [15] and as such, are unsuitable models for practical computing or investigating artificial intelligence or human cognition [16]. However, two types of hyper-computing are notable for their practical importance: machines that are capable of continuous/analog information manipulation and machines that incorporate real-time interaction with their environments.

In the first case, the TM's finite alphabet of discrete symbols prevents it from computing problems that require the infinite precision of real numbers. Specifically, if all numbers must be represented using a finite length string of symbols from a finite length alphabet, the range of numerical values is at most a countable. For instance, while a single symbol, such as $\pi$, can represent an irrational number, a finite alphabet of symbols might not have symbols to represent the values of *e* or *i* and, if it does, then the inter-symbol relationships would require a look-up table since infinite precision calculations would take infinitely long to complete. Thus, exact quantities cannot always be defined or manipulated. Since most empirical measurements involve such quantities, digital computers are forced to approximate the real numbers instead. Aside from the purely pragmatic desire to process on real numbers, digital computers restrict precision when modelling continuous dynamical systems, which is problematic when considering the sensitivity of chaotic dynamics. Also as computers are used to investigate intelligence which is arguably embedded in a continuous physical space, their inability to deal with real numbers limits the extent of this field of research. However, while the processing of real numbers is a clearly practical and desirable capability for computers to have, it is unlikely that it will ever be realized on digital computers (in which higher precision entails greater space requirements).

On the other hand, interactive computation has been a mainstay of the software industry for decades; and while it is ubiquitous, it has only been widely recognized as a super-Turing paradigm within the last ten years. By now, it is quite clear that many features of a TM (synchronous input-output, the unified alphabet, and necessity of termination, to name a few) do not describe systems like autonomous robots, word processors, operating systems, or the internet [26, 12, 6, 5, 16]. For each of these systems, the time-sensitive input is streamed to the machine as the machine works, and the machine's output at any given time is potentially a product of its entire history of operation. Similarly, there is no single ultimate output that these machines produce, but rather they are expected to continuously operate, remaining responsive to the environment.

The internet represents another form of interaction, in which the environment does not simply provide operational input to the machine, but may also alter the machine's very construction at any time. Whether or not these alterations enhance or reduce the machine's capability, they may fundamentally change how the machine will be able to respond to future input, including gaining the capacity to operate on a newer or larger alphabet [12].

What makes interactive computing more expressive than TM-equivalent paradigms is that interactive computing can express more than functions. Granted, interactive computing may not be able to express all functions, but the ones that are TM-computable exist as a special case where the input-output relationships of the interactive machine are restricted to the TM definition. Essentially, this generalizes the purpose of the machine from computing functions to generally performing a wider set of tasks [6, 5].

## 3.1 Interaction as hyper-computation

There are many types of the interaction that can be considered, but not all of them entail a system with super-TM capability. For instance, a TM itself defines interaction between the tape and tape-head. In the weakest sense, "interaction" merely suggests the presence of more than one non-independent entity in a system. However, under specific stronger notions of interaction, the restrictions on the Turing Machine formalism are relaxed and super-TM expressiveness ensues. This stronger form of interaction is minimally defined by the following two characteristics:

**Definition 1.** Coupling: *Current output of an interactive computer affects its future input.*

**Definition 2.** Persistence: *Current output of an interactive computer is affected by more than current input.*

Without these two characteristics added to the current capabilities of a TM, the interactive machine would have no more expressiveness than repeated calls to a TM. However, with these two characteristics, the machine is capable of modifying its environment in a meaningful way, partially affecting its own future input and therefore making decisions based on potentially all of the input it has ever received. While the first characteristic could be considered a property of the machine's environment (rather than of the machine itself), it can be argued that this property provides a type of sensorimotor coupling within the machine, thus making the machine not just reactive, but active and proactive as well. It could also be argued that a TM might have this property in the reactive sense, but without Persistence, lacks the capability for a long-term action strategy.

It is easy to show at this point that TMs are a special case of this type of interactive computers where the environment is ambivalent about the machine's output and the machine is completely amnesiac. Thus, interactive computers form a proper superset of Turing Machines and are therefore more expressive [24].

There may remain some question, however, about whether these interactive machines are "automatic" in Turing's sense of the word. The initial TM was automatic in the sense that once the input was finalized on the tape, the machine would deterministically produce the output such that each subsequent configuration of the machine (and tape) was entirely determined by the previous configuration. It has been argued that if the machine had paused to query an outside source for input (from a human or an all-knowing oracle, for instance), then the machine would not have been purely mechanical. It could be argued that since interaction with the environment forms a crucial part of an interactive computer's function, that an interactive computer is not purely mechanical. However, there is no reason to suspect that, given the current configuration of an interactive computer and the value of the current input from the environment, that the subsequent operation of the computer would not be entirely deterministic (until the next input arrived). In fact, given a sequence of input symbols from the environment, $S$, if $S$ were fed to two identical interactive computers that are initially in the same state, both computers ought to produce identical streams of output. While this example neglects the organic role of the environment (namely neglecting the first principle above), it shows that interactive computers may be automatic machines. The key to this argument lies in the distinction that the environment, while a part of the interaction, is not a part of the machine with which it interacts.

## 3.2 Models of interactive computing

### *3.2.1 Persistent Turing Machines*

A common model of interactive computing that expresses the properties of Coupling and Persistence is the Persistent Turing Machine (PTM) [7]. The PTM formalisms builds from the standard TM by altering the input-output mechanics and by incorporating internal memory. This is done by giving the PTM three tapes: one for read-only input, one for write-only output, and an internal, read/write tape for working memory. The PTM functions at two time scales: during each macro-step, the environment synchronously provides input on the input tape and consumes output from the output tape, while the PTM produces the appropriate output for the given input over a sequence of micro-steps. Thus, while the above two properties of interaction have been added to the TM construction, the input and output are still completely synchronous. However, because the internal tape is persistent across macro-steps, identical input may not always produce the same output.

### *3.2.2 Interactive Machines*

A similar construct has also been investigated by Van Leeuwen and Wiedermann [13], referred to as "Interactive Machines" (IMs). While IMs possess both Coupling and Persistence, it also has a weaker input-output relationship, defined as the "interactiveness" property, which states:

**Definition 3.** Interactiveness: *Any time an IM receives a non-null input symbol from the environment, it must provide a non-null output symbol to the environment some finite time later, and* vice versa*.*

Thus, rather than have synchronous input and output, such that input and output can be described in pairs, IMs have asynchronous input and output, such that any number of inputs can be fed to the automaton so long as some output is given some time later. While automata that have interactiveness may have their inputs and outputs interleaved, there is not even the stipulation that there be a one-to-one input-output relationship. This weaker input-output relationship positions PTMs as a special case of IMs. Further, unlike PTMs, IMs do not operate at multiple time scales, though blank symbols are defined for both input and output which may allow the simulation of multiple time scales.

### *3.2.3 Lineages of automata*

Another model of interactive computing, Site Machines [24], represent the notion of a physical machine, such as a desktop computer. Aside from interacting with the machine through its conventional channels of input and output (such as a keyboard and monitor), one might also alter the machine's physical construction (by adding more memory or a new communication interface). These interactions may serve to augment the machine's capabilities, or to diminish them (such as enacting physical harm on the machine's components), but either way they have important ramifications for the future of the machine's conventional operation. This same concept can be extended to the Internet as a whole, where new components are regularly added and removed, not simply changing the computational power, but fundamentally altering the machine architecture.

Site or Internet Machines can, at any time, be conceived as complex, multi-dimensional $\omega$-transducers (automatons that process an infinite input streams into infinite output streams). Then, at specific times when physical modifications occur, these transducers can be replaced by other ones with other capabilities. Thus, Site and Internet machines are appropriately represented as sequences or lineages of $\omega$-transducers, and ultimately, the complexity of the input stream is only limited by the complexity of the most complex transducer. Since the set of situations in which the input stream evolves its alphabet of symbols as time progresses cannot be modelled by a Turing Machine, lineages of automata present a strictly more expressive class of machines [25]. The notion of lineages where preceding transducers physically construct their successors have been explored and are referred to as autopoietic automata [27], though their computational expressiveness derives entirely from the notion of lineages.

# 4 BOUNDARIES OF COMPUTING

Concurrency is a central notion in interactive computing. In the simplest sense, the environment and the automaton are processing in parallel and their behaviour is mutually coupled through the input and output ports of the automaton. However, as more automata are included within the environment, more parallel, semi-enclosed systems are available for direct or indirect interaction. Further, each automaton itself may be a multi-scale system, with high-level operations delegating to lower-level components for execution details (such as is explicitly described by the $\lambda$-calculus). Thus, this entanglement of semi-decoupled interactive dynamical systems begins to resemble discrete-time versions of natural physical processes. In fact, any physical process could be described as an interactive computer, given the Coupling and Persistence properties of interactive computing.

In fact, a school of thought called *pancomputationalism* would argue that all physical processes are intrinsically computing. This idea suggests that the laws of physics are computational rules which are processed continuously as time progresses, shifting the current state of the universe to the next. However, this view devalues the concept of computation and there might be more constructive ways to discuss physical systems as computers. To begin this constructive discussion, it is necessary to understand the boundaries of computing.

In his 1992 book, John Searle paraphrased this notion by saying "For any object there is some description of that object such that under that description the object is a digital computer" [21] and Jack Copeland has referred to this as Searle's Theorem[4], for the sake of argument [4] . Copeland rephrases this statement to clarify it, by saying that any object, $e$, can be described by mapping its states by some labelling, $L$, such that the pair $< e, L >$ is a computer. Both Searle and Copeland agree that under this definition, $e$ is not intrinsically computational but that it can be described as such with an appropriate selection of $L$. However, Copeland argues, not just any $L$ is valid.

In Searle's initial statement, he went on to say that there existed a description of a wall such that the wall was implementing a word processor. Copeland shows that for this to be possible, $L$ must be time-sensitive and defined after the fact. In other words, a labelling could be constructed to map the states of an observed history of the wall to the states of a single observed computation of a word processor, but both sequences would have to be completely observed first and the labelling would have to be applied afterwards. This means that not only could $L$ not be defined *before* the wall "executed" the word processor program, once $L$ was defined, it would only apply to the wall at a specific set of time instances, and would be invalid at

[4] Though Searle himself only states the theorem as an absurd logical extension of some of the contemporary theoretical computational discourse, it serves as a concise statement of the central problem.

any other time or for any other run of the program. While $< e, L >$ could be called a computer implementing a word processor in this case, Copeland argues that $L$ constitutes a "non-standard" description of the wall. Alternatively, he argues that only so-called "honest" descriptions ought to be valid for $< e, L >$ to form a valid computer.

The crux of Copeland's argument lies in the dynamics and semantics of $e$ and $L$. He argues that for the description of a wall to describe a computer implementing a word processor, $L$ would contain all of the computational power, in other words, all of the dynamics *and* semantics of the system; in fact, the wall could be swapped out for any other object. He argues that an "honest" description of an object requires that the majority (if not all) of the dynamics exist solely within $e$. Meanwhile, $L$ provides an arbitrary, time-invariant symbolic representation of the states of $e$ while respecting the natural dynamics of $e$.

An element necessary to this discussion has merely been implied thus far: for there to be a semantic description of an object, there must be a subject doing the describing. If the semantics are not an intrinsic part of a physical system, then they have to be bestowed upon that system from somewhere else. This represents an instance of an epistemic cut [17], where syntactic manipulation of symbols by a dynamic process (for which the symbols have no meaning) produces meaningful information or performs a meaningful task for a subject (by whom the symbols have their meaning bestowed). Even a common digital computer is not a discrete symbolic machine, but a severely constrained dynamical system in which states of certain components are given specific meanings (eg. 5V stored in a flip-flop may be interpreted as a binary 1, while 0V is interpreted as a binary 0). While there is a general question about how to bridge epistemic cuts in nature [17], it is clear that in the case of practical computation, the physical system is constrained by human hands and given meaning by human minds, such that when a human symbol is fed to the physical system it will produce another human symbol, while remaining oblivious to the meaning of its action.

In fact, another element of computation that is not discussed by Searle or Copeland in this context is that of programming or algorithms. While a dynamical process can have its states labelled to express a single computation (eg. a single pass through an algorithm), there has been no discussion about how to generalize these dynamics to multiple computations (ie. an algorithm run multiple times with different inputs). For a dynamical system to be useful for computation, it must provide a number of degrees of freedom for placing custom constraints upon the dynamics. Thus, even if a wall could be honestly mapped to a computational process, it would still severely restrict the types of tasks that could be computed. This implies that, while any physical system might be described as a computer, not every physical system is a generalized computer, and there will be limits to that system's computational expressiveness.

## 5 PHYSICAL SYSTEMS AS COMPUTERS

A wide variety of physical systems can be described as useful computational platforms; systems that provide intrinsic and predictable dynamics and a formalized approach to placing constraints on those dynamics. Several examples of such systems will be explored in this section, to demonstrate the flexibility of this framework for conceiving of computation.

First, one could consider the natural physical system of light and occlusion, whereby opaque objects block light and cast shadows. When appropriately constrained, this system can provide meaningful computation through the shape and location of shadows. For instance, a sundial positions a labelled face and a gnomon such that the shadow cast by the gnomon's edge aligns with the labels on the face to indicate the time of day (Figure 5). In a sense, a sundial represents a program for telling time on the platform of the physical system of occlusion and celestial mechanics.

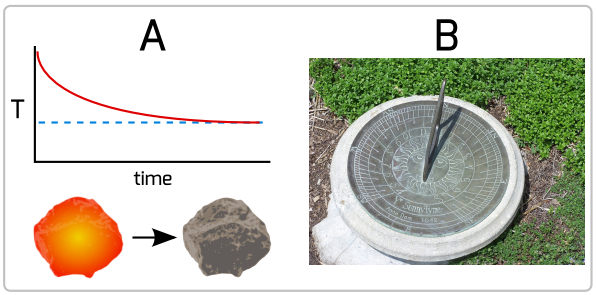



**Figure 1.** Two examples of natural computation. In A, a heated rock cooling to ambient temperature is used to calculate a geometric curve approaching an asymptote. In B, a sundial is used to compute time [22].

Similarly, non-opaque obstacles can be placed in the path of a ray of light (such as a laser) to reflect, refract, and filter the light. Again, these obstacles will act as constraints on the dynamical system of optics and could be configured to perform meaningful computation. For instance, a series of such obstacles could be arranged such that, depending on the position of the light source, the ray is redirected to one of two target faces, computing a decision problem where the light position and the targets have semantic meaning.

A third example might be dropping a ball in the physical system governed by gravity and collision mechanics. In this system, constraints may be physical obstacles which may be placed at various positions and angles such that a dropped ball may tumble and bounce along a path depended on the position from which it is released (not unlike a pinball machine). If a mapping is devised between the ball's physical positions and some meaningful states of a computational process, the physical obstacles could be configured to represent logical conditions or other predicates.

Finally, the physical system could be that of electrical potential energy, which drives electrons to flow through conductive materials. Constraints can be placed on this system by redirecting or impeding the current, as is done with wires and resistors. The currents can also be manipulated with doped semi-conductors, such as transistors and diodes. Since transistors are the building blocks of transistor-transistor logic (TTL) which yields logic gates (AND gates, OR gates, NOT gates, etc) and ultimately flip-flops, there can be no doubt that the constraints on this system ultimately yield a computational platform.

## 6 ABSTRACT COMPUTATIONAL PLATFORMS

The above examples have demonstrated a variety of physical systems that can provide varying degrees of general computation. The computational expressiveness of each system relies heavily on the degrees of freedom for setting constraints on the intrinsic dynamics. The key to computing with each of these dynamical systems is to place constraints on the dynamics such that the states of the system can support an honest semantic mapping. For instance, the constraints on electrical potential to produce logic gates are specifically designed to support the semantic mapping of predicate logic. Digital computers are so valuable because of the breadth of what can

be expressed in predicate logic, though even predicate logic has its limitations.

Observing the above physical systems, some generalization may be drawn for defining the characteristics of an *abstract computational platform*. First, such a platform must have its own intrinsic dynamics. As in the above examples, these dynamics can be those of the physical world, but they do not have to be. For instance, the dynamics of a Turing Machine are not natural, though they are intrinsic to the platform and are predictable.

Second, such a platform must provide customization of constraints on its dynamics (Figure 2). Essentially, this provides a way to "program" on the platform, redirecting the dynamics to perform some meaningful computation. In other words, it is this property of a computational platform that allows a programmer to map dynamical states to semantic values and place constraints to process the semantics. In the case of a Turing Machine, this is afforded by the design of the machine's configurations and the instruction table.

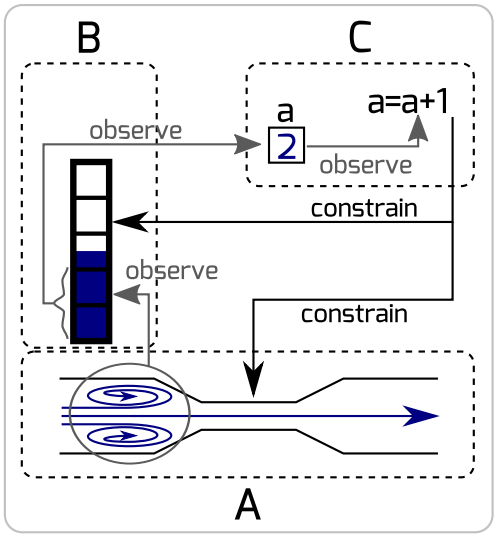



**Figure 2.** Abstract platforms of computation. In A, the dynamics of some flowing system is constrained to manipulate the quantity of fluid on the left hand side. This quantity is reinterpreted in B, into a real-valued number, and constrained again through discrete quantization. This discrete number is represeted in C as an integer value. Ultimately, a simple equation controls the constraints in A and B to produce the semantic outcome of the equation.

A natural result of these two characteristics is that the dynamics and the degrees of freedom on customization limit the computational expressiveness of a platform. For instance, the light-and-occlusion system might be able to compute more than time of day, but may not be as expressive as a Turing Machine. Thus, while there may exist systems with severely limited computational expressiveness (such as Searle's wall or a heated rock cooling to the ambient temperature), even these systems may serve as platforms for some few computational tasks.

Another result of this definition of a computation platform is that it naturally gives rise to the concept of layers of abstraction (Figure 3). Each computational platform exists as a single layer of abstraction and is capable of supporting computation for another layer (eg. a digital computer supports an operating system which, in turn, supports a web server). Saying that a computational platform is "computing" does not stipulate *what* is being computed. Thus, for every platform, there requires some observer to remap its states and place constraints to establish some semantic meaning of the dynamics before computation can occur. However, this observer need not be a human, but could, instead be another machine, or even another computational layer.

Digital computers represent several, layered, computational platforms before the first line of software is even available. Semiconductors are arranged to constrain the natural dynamics of electricity to emulate logic functions. These logic gates are then arranged as flip-flops which can store one of two stable voltage states. Here, the first layer reinterprets the flow of electrical current (and the state of the electrical potential across the circuit) as logical predicates. The second layer reinterprets the combination of logical predicates into binary values. While this might appear to represent a system with a discrete time step and a discrete state space, it is in fact a constrained continuous system, with each layer of constraints introducing new semantics on the previous layer. Likewise, while a register of 8 flip-flops can store up to 8 binary digits and these digits can have numerical meaning to a human, this semantic meaning is not intrinsic to the register. This same trend continues upwards through the arithmetic-logic unit and the instruction pipeline. Thus, at every layer, the states of the previous layer are remapped with new semantics (eg. voltage $\rightarrow$ binary $\rightarrow$ letters $\rightarrow$ stories).

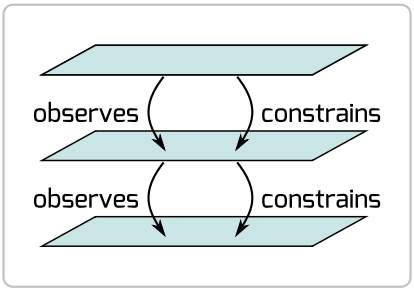



**Figure 3.** Abstract computational platforms can be layered, each reinterpreting and constraining the states and dynamics of the lower layers. However, the reinterpretation must be a static mapping and the constraints must be afforded by the underlying dynamics, such that not all platforms are suitable for all tasks.

However, computational layers are abstract concepts. While they have an intrinsic dynamics, those dynamics need not be natural or even deterministic. Also, for computational platforms to accommodate layering, a platform must have some notion of what is required for another platform to support it. For instance, a Turing Machine's dynamics can be described as follows:

1. Look up entry in instruction table for current configuration and current tape symbol
2. Change configuration, modify tape symbol, move tape depending on the entry in the table

Also, a Turing Machine provides to the programmer the following operations:

- Read/write tape
- Move tape left/right
- Read configuration
- Change configuration

For a TM program to run, these operations must be guaranteed by the platform. For a platform to support a TM, it would have to provide the ability to change and read the configuration of the machine, and store and manipulate information that can be represented as a linear symbol tape of arbitrary length.

## 7 FORMAL CONSTRAINT LANGUAGE

Under the definition of computers as layers of abstract computational platforms, programs are described by a collection of specific constraints placed upon the dynamics of the underlying platform. To match the generality of this framework, it would be ideal if there were a formal, platform independent constraint description language (CDL) which could be used to describe constraints on any platform using the elementary operations provided by that platform. Such a language would have to operate at the natural level of abstraction of whichever computational platform it is currently targeting, describing input, output, and all constraints in a natural way for that platform. Further, the language should be flexible to the wide potential diversity in implementations, including interaction rules.

### 7.1 Abstract State Machines

Seeking a formalism for modern software, Gurevich et al have created Abstract State Machines (ASMs) [8] to be a high-level conceptual model which expresses algorithms and other, non-algorithmic programs at their native level of abstraction. One such ramification of this idea is that data types and operations may all maintain their semantic meanings, delegating the implementation details to a lower layer of abstraction. The abstract layering approach to ASMs naturally fits with the notion of computational layers and thus may provide the basis of an ideal CDL.

Briefly, an ASM defines the state of the system at any time as a structure or group (in the abstract algebra sense) which contains all of the values and potential operations that may be performed on those values. Then, each step of the ASM transforms the state based on which of the operations is valid at each step, performing every valid operation in parallel. To more intuitively grasp the functioning of an ASM, it can be expressed as a set of conditional assignment statements. Each step, every statement is evaluated in parallel and, where the conditions are met, assignments take place to modify the values stored by the system state. To preserve the level of abstraction of the procedure, implementation details are often replaced by semantically named functions, as is regularly done in object-oriented programming, which become the elementary operations of the program. The intuition is that each of these sub-functions might be described by an ASM at their native level of abstraction, recursively expressing more implementation detail.

### 7.2 Interactive ASMs

While ASMs are not intrinsically interactive, they can include interaction through their abstracted function calls. In the ASM literature, two types of interaction are defined: inter- and intra-step [8, 2]. In inter-step interaction, input is injected into the state of the computer between computational steps. This can be viewed as the environment modifying the state of the computer, an idea that is undesirable in both ASMs and software engineering. The alternative is for the computer to specifically request and then receive input, deliberately inverting the input-output sequence and giving the computer control of the interaction. In interactive ASMs, all interaction (synchronous or asynchronous) is performed through intra-step queries, in which querying the environment (or another agent in the environment) replaces the standard output stream of an interactive automaton and the environment's response replaces the input stream. This querying semantics is reminiscent of Turing's Choice Machines, which could pause to request further input from a user.

The distinction between inter- and intra-step interaction is one largely of perception, as it depends on where one draws the boundary between the environment and the embodiment of the computing agent. For instance, if one considers the sensor buffer of a robot to be external to a program that the robot is running, then the program must deliberately fetch any data that is waiting there. However, the sensor buffer is a physical component of the same robot that is running the program and, by taking this slightly larger view of the system, it might appear that the environment is writing data directly to the robot's computational state. While any instance of inter-step interaction can be inverted by shrinking the program definition in some way to exclude the input ports that are directly coupled with the external environment, this also blurs the line between the automaton and its surroundings. As ASMs directly discuss levels of abstraction, this should not necessarily be a problem, as it can be argued, following the previous example, that the robot's physical body occupies a different abstraction layer from its "mind". Another way of considering the coupling is that the robot's body is a part of the environment and is interacted with by the robot's programming.

Ultimately, since the difference between intra- and inter-step interaction is largely a matter of perspective, and inter-step interaction can be performed by an Interactive Machine, there is no reason to suspect that ASMs cannot be purely mechanical. Thus, the interactiveness of ASMs is as expressive as that of Van Leeuwen and Wiedermann's IMs.

### 7.3 Continuous Time ASMs

There is, however, one feature of ASMs that bares some consideration for their selection as a formal CDF: the discrete time step. While continuous values might be adequately represented by the symbolic abstraction (in the same way as mathematics represents real numbers), ASMs operate in discrete time steps. However, one could conceive of a continuous time ASM, in which each parallel operation occurs in continuous time, though the properties of such a construction would have to be further explored. If ASMs could be adapted to model constraints on continuous time, concurrent, interactive dynamics, it would likely provide a satisfactory description language for constraints on any computational platform.

## 8 CONCLUSION

Computation has evolved greatly since the formulation of the Turing Machine in 1936. Many computational paradigms have been envisioned that reach beyond the limitations of the original conception, though, many of these must remain as abstract concepts owing to their inclusion of concepts of infinity. Other paradigms, which are represented ubiquitously in operating systems, word processors, and other embedded or interactive software, also supercede Turing Machine expressiveness. With the repetitive relaxing of the definition of computing, some have come to speculate that all of physics is inherently computational. Aiming to reserve the term "computation" for a special subset of dynamics, this paper has defined computation with a framework of abstract computational platforms.

Computational platforms provide a constructive and unified way to express the computational nature of any natural or abstract dynamical system. It has been argued that any dynamical system can be described as a computer, but every dynamical system has natural restrictions to how constraints can be placed upon its dynamics so as to accomplish meaningful computational tasks. Computation then exists as these constrained dynamics. Under this description a wall could not execute a word processor, but a light-occlusion based system could compute time.

The framework of computational platforms seeks to describe a computational system as a dynamical system, whose states maybe acquire semantic meaning and whose dynamics may be constrained to uphold those semantics. Technically, any dynamical system can be described as such, but the degree to which the constraints can be placed are dependent on each system. With less flexibility on the constraints comes less computational expressiveness.

The final requirement for this approach is to formalize a language for describing constraints for any arbitrary computational platform. It has been shown here that perhaps Abstract State Machines could serve as a useful direction, provided that a continuous time version is feasible. Further insight might be gained by exploring related models of concurrent computation, including the Actor Model [10] and Interaction Nets [11].

Ultimately, this approach aims to incorporate all conventional computing paradigms, including Turing Machines, Interactive Computing, which have been discussed here, as well as many less conventional paradigms. In particular, this approach seeks to include distributed computing paradigms (for instance, artificial neural networks [20] and Stochastic Diffusion Search [1]), each interacting with the environment and/or each other. Each of these swarms could be seen as a computational platform with intrinsic dynamics. While each swarm agent might be considered as a lower computational layer, constraints can be described on the swarm as a whole by modifying the content or protocol of their communications. By allowing this framework to describe populations of interacting processes, a bridge might be build to constructively discussing the computational capacity of the brain.

The work presented here parallels the framework for computating and information processsing discussed in [18], but with a specific emphasis on incorporating distributed, continuous, interactive, and reflective systems. Further, the work presented here views semantic mappings as crucial for distinguishing computation from other physical processes.

As much as this approach aims to include all dynamical systems under the aegis of computing, it also aims to position these systems such that their computational expressiveness is understood to be limited. While all physical systems could be said to support computation, they are not themselves computational nor are they all capable of expressing the same class of programs. Also, abstract dynamics can be said to support computation, and their computational capacity can be understood alongside their natural counterparts. With such broad applicability, hopefully, the framework of computational platforms will help to constructively focus the future discourse of computing.

## 9 ACKNOWLEDGEMENTS

The authors would like to convey their gratitude to the John Templeton Foundation for providing the funding for this work and to the anonymous reviewers for their helpful comments.

## REFERENCES

[1] J M Bishop, 'Stochastic Searching Networks', in *Artificial Neural Networks, 1989., First IEE International Conference on (Conf. Publ. No. 313)*, pp. 329–331, (1989).
[2] Andreas Blass and Y Gurevich, 'Ordinary interactive small-step algorithms, I', *ACM Transactions on Computational Logic (TOCL)*, **V**(July), 1–55, (2006).
[3] Alonzo Church, 'An unsolvable problem of elementary number theory', *American journal of mathematics*, **58**(2), 345–363, (1936).
[4] BJ Copeland, 'What is computation?', *Synthese*, (1996).
[5] Gordana Dodig-Crnkovic, 'Significance of Models of Computation, from Turing Model to Natural Computation', *Minds and Machines*, **21**(2), 301–322, (2011).
[6] Dina Goldin and Peter Wegner, 'The interactive nature of computing: Refuting the strong ChurchTuring thesis', *Minds and Machines*, 1–26, (2008).
[7] Dina Q. Goldin, Scott A. Smolka, and Peter Wegner, 'Turing machines, transition systems, and interaction', *Information and Computation*, **194**(2), 101–128, (2004).
[8] Yuri Gurevich, 'Interactive algorithms 2005', in *Mathematical foundations of computer science*, pp. 26–38, (2005).
[9] Yuri Gurevich, 'Foundational Analyses of Computation', *How the World Computes*, 264–275, (2012).
[10] C Hewitt, 'Actor Model of Computation: Scalable Robust Information Systems', *arXiv preprint arXiv:1008.1459*, 1–32, (2012).
[11] Yves Lafont, 'Interaction nets', in *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 95–108. ACM, (1989).
[12] Jan Van Leeuwen and Jírí Wiedermann, 'On the power of interactive computing', *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics*, **14186**(201), 619–623, (2000).
[13] Jan Van Leeuwen and Jírí Wiedermann, 'A computational model of interaction in embedded systems', *Computer Science*, (January), (2001).
[14] Robin Milner, Joachim Parrow, and David Walker, 'A calculus of mobile processes, i', *Information and computation*, **100**, 1–40, (1992).
[15] Vincent C. Müller, 'On the Possibilities of Hypercomputing Supertasks', *Minds and Machines*, **21**(1), 83–96, (2011).
[16] Aran Nayebi, 'Plausible hypercomputability', *arXiv preprint arXiv*, 1–55, (2012).
[17] Howard H. Pattee, 'The physics of symbols: bridging the epistemic cut', *Biosystems*, **60**(1-3), 5–21, (2001).
[18] Gualtiero Piccinini and Andrea Scarantino, 'Information processing, computation, and cognition.', *Journal of biological physics*, **37**(1), 1–38, (2011).
[19] Wolfgang Reisig, 'Abstract state machines for the classroom', *Logics of Specification Languages*, 15–46, (2008).
[20] D.E. Rumelhart and J.L. Mcclelland, 'Parallel distributed processing: explorations in the microstructure of cognition. Volume 1. Foundations', (January 1986).
[21] John R. Searle, *The Rediscovery of the Mind*, MIT Press, 1992.
[22] SEWilco. Garden sundial mn 2007. Creative Commons Attribution-Share Alike 3.0.
[23] A. Turing, 'On computable numbers, with an application to the Entscheidungsproblem (1936)', *B. Jack Copeland*, 58, (2004).
[24] J van Leeuwen and Jírí Wiedermann, 'On algorithms and interaction', *Mathematical Foundations of Computer Science 2000*, 99–113, (2000).
[25] Peter Verbaan, J van Leeuwen, and Jírí Wiedermann, 'Lineages of automata', *UU-CS*, (2004).
[26] Peter Wegner, 'Interactive foundations of computing', *Theoretical computer science*, **3975**(97), (1998).
[27] Jírí Wiedermann, 'Autopoietic automata: Complexity issues in offspring-producing evolving processes', *Theoretical Computer Science*, **383**(2-3), 260–269, (2007).

# Stochastic Diffusion Search applied to Trees: a Swarm Intelligence heuristic performing Monte-Carlo Tree Search

**Thomas Tanay** and **J. Mark Bishop**[1]
**Matthew C. Spencer** and **Etienne B. Roesch** and **Slawomir J. Nasuto**[2]

**Abstract.** In this paper, we introduce Stochastic Diffusion Search applied to Trees (SDST), a swarm intelligence heuristic inspired from Stochastic Diffusion Search able to solve the complex and general problem of forward planning. In SDST, each individual agent processes information concerning a unique action without "awareness" of the way in which actions are being compared and combined. Yet the dynamics of the entire population of agents lead to a high level "reasoning" about successions of actions analogous to Monte-Carlo Tree Search (MCTS). In its functioning, SDST is argued to introduce a meta-level in the swarm intelligence paradigm.
This result is presented in the context of Abstract Platforms of Computation (APCs), a concept introduced in an accompanying paper in an attempt to clarify and broaden the notion of computation. In particular, the concept of APC is used to draw a distinction between classical sequential algorithmic models of computation and nature-inspired parallel distributed ones. It is argued that the understanding of (at least human) cognition requires the study of decentralised emergent systems (fundamentally parallel and distributed), whose computational properties cannot be reduced to their Turing power.

## 1 INTRODUCTION

In [12], we introduced the concept of Abstract Platforms of Computation (APCs) in order to characterise in a unified framework various computational paradigms and we discussed in this context the computational nature of both abstract and natural dynamical systems. The concept is defined in terms of intrinsic dynamics and customisation of constraints on these dynamics in a way that is consistent with other recent works. For example, [18] defines computation as "the processing of medium-independent vehicles according to rules", the notion of medium-independent vehicle being further detailed in the following way: "a given computation can be implemented in multiple physical media (e.g. mechanical, electromechanical, electronic, magnetic, etc), provided that the media possesses a sufficient number of dimensions of variation (or degrees of freedom) that can be appropriately accessed and manipulated."

Although our framework proposed in [12] based on the concept of APCs agrees in the broad sense with the taxonomy of computation proposed in [18], we do not believe that cognition is merely a computational phenomenon (as it is claimed to be in [18]: we may conclude that cognition is computation in the generic sense) even though the APC framework admits perfectly viable and a posteriori structurally isomorphic computational descriptions of cognitvce processes. Thus ultimately, one of the most important problems in cognitive science is to determine and characterise the set of dynamical systems that have the right computational properties to provide such descriptions. In the case of biological systems, cognition results from the activity of the brain which is inherently a parallel and distributed system. Surprisingly however, current parallel and distributed computational models (such as neural networks or swarm intelligence heuristics) appear rather limited when facing some complex problems. For example, artificial neural networks are particularly adapted to solve pattern recognition problems but they have difficulty in sequentially processing high arity predicates (they can be conceived as fundamentally building learning mappings in complex high dimensional Euclidean spaces). Consequently, the dominant paradigm in artificial intelligence has historically been and is still the sequential algorithmic one.

In the present paper, we introduce Stochastic Diffusion Search applied to Trees (SDST), a swarm intelligence heuristic (inherently parallel and distributed) inspired from Stochastic Diffusion Search (SDS) and able to solve the complex and general problem of forward planning in a way analogous to Monte-Carlo Tree Search (MCTS). Although some previous attempts have been made to apply decentralised methods to forward planning tasks, such methods did not reach the same degree of generality as SDST. For example Tesauro developed in 1989 a neural network program playing Backgammon (a finite two-person zero-sum game with imperfect information) better than any other program (the program called Neurogammon won the backgammon competition of the First Computer Olympiad). However, Tesauro explicitly expressed in the introduction of [17] that "the game of backgammon in particular was selected because of the predominance of judgement based on static pattern recognition, as opposed to explicit look-ahead or tree-search computations."

By presenting SDST, our objective is to extend the applicability of parallel and distributed models of computation (and in particular SDS) to solve problems that were historically exclusively addressed with a sequential algorithmic approach requiring centralised control and access to the data. Here, it is important to contrast this result with what constitutes a universality proof[3]. Indeed, proving that a model is Turing-equivalent consists in showing that any Turing machine can be simulated by an instance of this model. But in such a case, there

[1] Goldsmiths, University of London, UK, email: thomas.tanay@gmail.com
[2] University of Reading, UK

[3] Note that there does not currently exist any such proof for SDS, but for example neural nets [19] and the rule 110 cellular automaton [6] (which are parallel and distributed models) are proven to be Turing-equivalent.

exists an abstract level at which the instance in question can be described as implementing a sequential algorithmic APC. On the contrary in SDST, the solution to the problem faced is fundamentally emerging from the decentralised interaction of simple computational agents, and the functioning of the heuristic fundamentally cannot be abstracted to the functioning of a sequential algorithm APC (See Table 1 for an illustration of the different levels of abstraction). In practice, as described in section 3, SDST has been "programmed" from the intrinsic dynamics defining SDS (the alternation of test and diffusion phases in a population of communicating agents) by customising the test and diffusion protocols.

**Table 1**: Nature of the APCs in two example implementations of MCTS: SDST and a hypothetical implementation of classical MCTS via a cellular automaton (CA).

| APC where MCTS occurs | SDST: Parallel Distributed | Classical MCTS: Sequential Algorithmic |
|---|---|---|
| Underlying APC | Digital computer: Sequential Algorithmic | Cellular automaton: Parallel Distributed |

For the sake of simplicity and clarity, and because it is the problem for which it was originally conceived, SDST is presented in the context of combinatorial games (finite two-person zero-sum games with perfect information such as Chess). However the discussion is entirely consistent with any planning task that can be represented as a tree of sequential decisions.

## 2 BACKGROUND

The work presented here rests on two pillars: the swarm intelligence metaheuristic for search and optimisation called Stochastic Diffusion Search (SDS) and the Monte-Carlo based search method for tree structures called Monte-Carlo Tree Search (MCTS). These two techniques are briefly described in the following subsections.

### 2.1 Stochastic Diffusion Search (SDS)

SDS is an efficient probabilistic swarm intelligence global search and optimisation technique that has been applied to diverse problems such as site selection for wireless networks [20], mobile robot self-localisation [2], object recognition [11] and text search [3]. Additionally, a hybrid SDS and n-tuple RAM [1] technique has been used to track facial features in video sequences [11, 8]. Previous analysis of SDS has investigated its global convergence [13], linear time complexity [17] and resource allocation [14] under a variety of search conditions.

SDS is based on distributed computation, in which the operations of simple computational units, or agents are inherently probabilistic. Agents collectively construct the solution by performing independent searches followed by diffusion of information through the population. SDS relies on two principles: partial evaluation of hypotheses and direct communication between agents. The SDS algorithm is characterised by three phases: Initialisation, Test and Diffusion—the test and diffusion phases are repeated until a Halting criterion is reached. During the initialisation phase each agent formulates a hypothesis, i.e. chooses a potential solution in the search space. During the test phase each agent partially evaluates its hypothesis: agents for which the partial evaluation is positive become active, and the others become inactive. During the diffusion phase, agents exchange information by direct communication: each inactive agent X contacts an agent Y at random. If Y is active, X takes its hypothesis, otherwise X formulates a new hypothesis at random (procedure called passive recruitment). In practice a halting criterion needs to be defined to stop the algorithm running: the properties of convergence of SDS led to the definition of two criteria, a weak and a strong version [13].

### 2.2 Monte-Carlo Tree Search (MCTS)

MCTS "is a recently proposed search method that combines the precision of tree search with the generality of random sampling" [4]. Since 2006, over 200 papers related to MCTS have been published, with applications ranging from computer Go to Constraints Satisfaction problems through Reinforcement Learning and Combinatorial Optimisation. [4] offers a complete survey of the published work on MCTS (until 2011) and argues that "it has already had a profound impact on Artificial Intelligence (AI) approaches for domains that can be represented as trees of sequential decisions, particularly games and planning problems".

MCTS has originally been developed in the context of computer game playing and finds its roots in B. Abramson's 1990 paper *Expected-outcome: a general model of static evaluation*. In this paper is introduced the idea to evaluate a game position by playing a great number of random games from that position, assuming that a good move must increase the expected outcome of the player[4]. The second decisive step in the development of MCTS was the publication in 2006 of Kocsis and Szepesvári's paper *Bandit based Monte-Carlo Planning*. In this paper is introduced Upper Confidence bound applied to Trees (UCT), a method that "applies bandit ideas to guide Monte-Carlo planning". The crux in UCT is to choose the moves to be evaluated at each node of the game-tree according to the information already collected during previous evaluations, in order to exploit more the most promising areas of the tree. Standard MCTS consists in iteratively building a "search-tree" (the root node of which is the current position) and is outlined in [5] as a succession of four phases: Selection, Expansion, Simulation and Backpropagation. In practice, the four phases are repeated until a given computational budget is spent (usually the time), at which point a decision is made and a move is played. The moves to be evaluated are first chosen in the existing search-tree from the root in a way that balances between exploration of the available moves and exploitation of the most promising ones (selection): the policy used to choose the moves during this phase is called the "tree policy" and this is where [9] introduced the analogy between a node of the search-tree and a multi-armed bandit. When a leaf of the search-tree is reached, the rest of the game is played up to a final state (simulation). The policy used during this phase is called the "default policy" and can be purely random in the simplest implementations of MCTS. The first move chosen by the default policy is then added to the search-tree (expansion). Finally, the statistics of each node crossed during the selection phase are updated according to the outcome of the simulated game (backpropagation). The way MCTS works is rather intuitive and it is argued in [4] that "the forward sampling approach is, in some ways, similar to the method employed by human game players, as the algorithm will focus on more promising lines of play while occasionally checking apparently weaker options." An important property of MCTS is its asymptotic

[4] This assumption is not necessarily a good one due to the distinction between random play and optimal play.

convergence to Minimax, i.e. it is assured to select the best move available if enough time is given (the convergence to Minimax can be very long in practice).

## 3 STOCHASTIC DIFFUSION SEARCH APPLIED TO TREES (SDST)

The initial motivation for the work on SDST was to extend the applicability of Stochastic Diffusion Search (SDS) to more complex search spaces, and combinatorial games were chosen as a first study case. Then, Monte-Carlo Tree Search (MCTS) came naturally as a good framework for several reasons. First, MCTS does not rely on domain knowledge but rather on a large number random game simulations and the notion of random game simulation fits well with the concept of partial evaluation in SDS. Second, the strength of MCTS relies on the tree policy balancing between exploration of the search space and exploitation of the promising solutions and SDS is a metaheuristic precisely conceived to solve this "exploration-exploitation dilemma" in the management of the computational resources. Finally, MCTS has proven very successful in a wide range of problems—not only game playing—and is still under active study. Conceptually, the application of SDS to game-tree exploration is a two step process. First, each node is being attributed a distinct and independent *local population* of agents to solve the problem of move selection on that node. Second, a reallocation policy is used to move the uncontacted agents toward more interesting regions of the game-tree—thus leading to the formation of a dynamically moving *metapopulation*[5] of agents.

### 3.1 First step: use of multiple populations of agents

The first step toward implementing SDST is to use SDS to solve the "exploration-exploitation dilemma" appearing during the selection phase of MCTS at each node of the search-tree. An algorithm detailing this idea is given in Table 2 (in SDS terms).

The operation of this algorithm is illustrated in Figure 2 on the small game-tree presented in Figure 1. The studied game-tree has been specifically designed to reveal the ability of the algorithm to converge to minimax and escape local optima: while a monte-carlo evaluation of the left and right moves for Max at the first ply would respectively lead to 50% and 75% chances to win—thus suggesting that the right move is better—the minimax resolution of the game-tree actually shows that, *if the players play optimally*, the left move leads to a win for Max (whatever Min plays at the second ply, the right move for Max at the third ply leads to a win) while the right move leads to a loss (if Min plays his left move at the second ply, whatever Max plays for the third ply leads to a loss with Min playing the left move at the fourth ply).

Figure 2 shows that during iterations 1 and 2, most of the agents in the root node population point toward the right move. Then during iterations 3 and 4, the selection of Min's left moves at plies 2 and 4 changes this tendency and at iteration 5 all the agents in the root node point toward Max's left move—the best move in the minimax sense. Figure 2 simply illustrates that, as any other MCTS with a different tree policy, the algorithm presented here converges to minimax (provided that every non-terminal node of the game-tree is being attributed a population of agents).

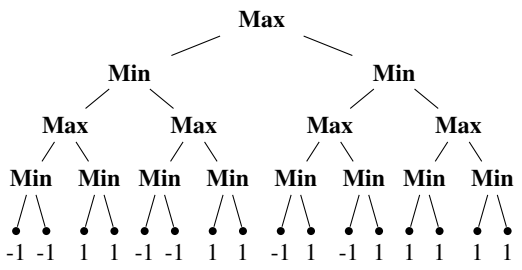
[5] The term was coined by Levins in [10] to describe the dynamics of interacting populations of social insects.

**Table 2**: First application of SDS to game-tree exploration: use of multiple populations of agents.

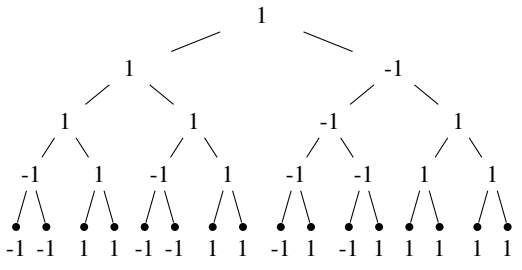
**Initialisation** During the initialisation phase, a local population of agents is generated for each node of the game-tree up to a fixed depth D. For each local population, agents' hypotheses are initialised to a possible move of the corresponding node.

**Test** During the test phase, a complete hypothesis is formed for each agent in the local population corresponding to the root node (later called root node population). This is done by combining agents from different local populations in a way analogous to the selection phase in MCTS: for each agent X in the root node population, an agent Y in the local population pointed by X's hypothesis is selected. Then an agent in the local population pointed by Y's hypothesis is selected, etc, until depth D is reached. Once a hypothesis is formulated, a simulation is run (in the MCTS sense) and the activities of the agents forming the hypothesis are updated according to the node they belong to (step corresponding to the backpropagation in MCTS): if the simulation leads to a win for Max, the agents in populations corresponding to Max's nodes become active and the agents in populations corresponding to Min's nodes become inactive (if it leads to a loss, it is the contrary).

**Diffusion** During the diffusion phase, each local population acts independently, i.e. a diffusion phase is undertaken in the sense of Standard SDS without communication with other local populations.




(a) Studied game-tree




(b) Minimax resolution

**Figure 1**: Studied game-Tree. The minimax resolution shows that Max is the winner if he plays optimally.

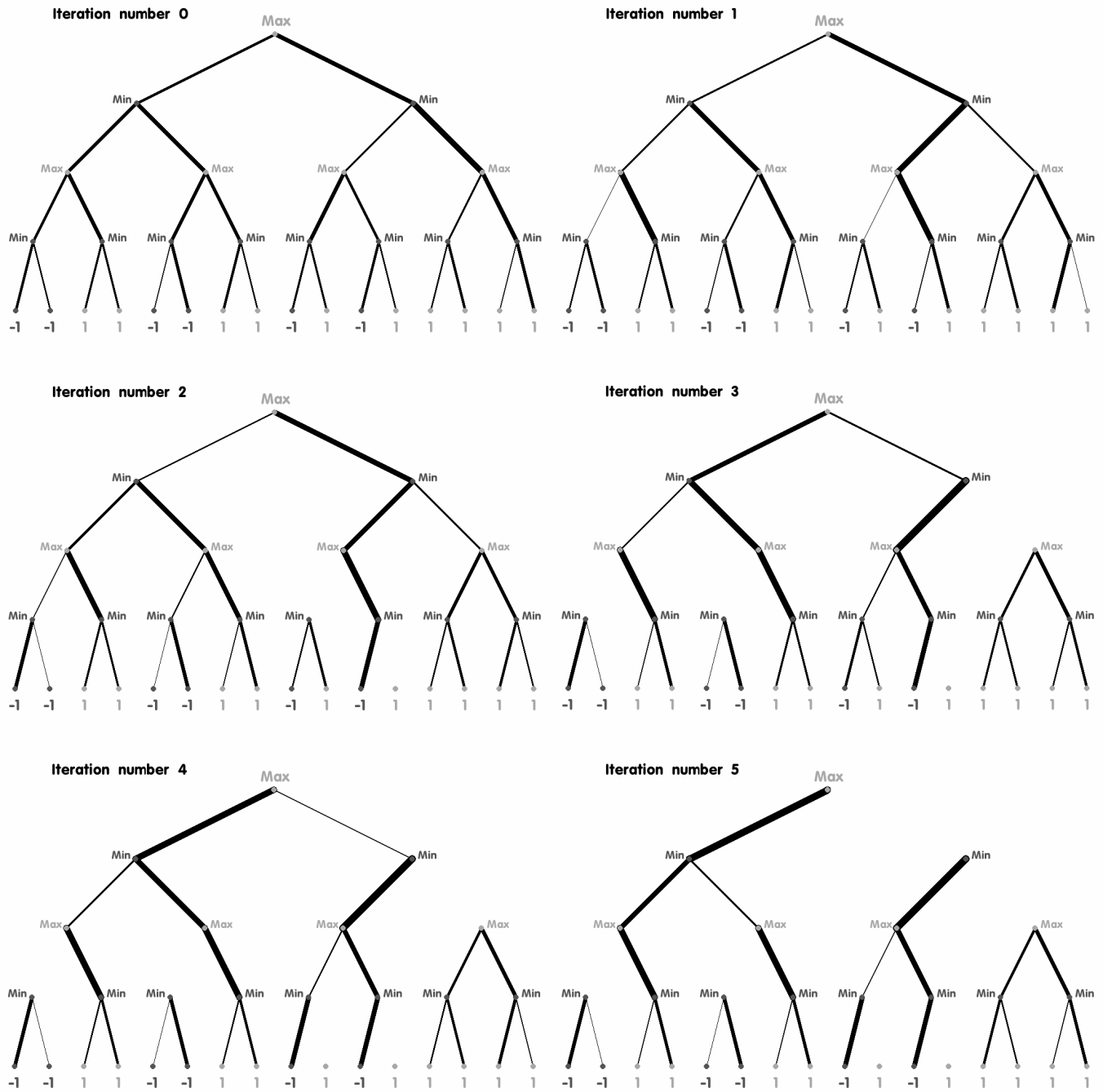



**Figure 2**: Illustration of the algorithm presented in Table 2: Evolution of the distribution of the agents in the different nodes of the studied game-tree (first 5 iterations shown, total number of agents = 175). Each branch has an area proportional to the number of agents in the parent node population supporting the move corresponding to the child node population.

### 3.2 Second step: use of a reallocation policy

Although the previously discussed algorithm is shown to solve the problem of game-tree exploration, it suffers from two main drawbacks. First, the number of studied nodes in the game-tree and the number of agents per node need to be fixed manually in a very artificial way. Second, a uniform repartition of the agents in the initialisation phase rapidly leads to many agents being uncontacted in some branches (for example, all the agents on the right side of the tree become useless after the fifth iteration in Figure 2).
These drawbacks can be solved with the use of a reallocation policy where agents are scattered in the tree from the root node and uncontacted agents are backscattered toward parent nodes. SDST uses such a reallocation policy, defined naturally as described in Table 3.

**Table 3**: Stochastic Diffusion Search applied to Trees (SDST).

**Initialisation** During the initialisation phase, all the agents are allocated to the root node population and their hypotheses are selected randomly among the available moves.

**Test** During the test phase, complete hypotheses are formed. For each agent X in the root node population, an agent Y in the local population pointed by X's hypothesis is selected. Then an agent in the local population pointed by Y's hypothesis is selected, etc, until the local population pointed by the last agent is empty. Once a hypothesis is formulated, a simulation is run and activities of the agents forming the hypothesis are updated.

**Diffusion** For each local population, the diffusion phase is divided in three subphases:

1. *Backscattering:* the agents that were not contacted to form a hypothesis go back in the parent node population. In order to preserve the hypotheses distribution among the different moves in the parent node population, a backscattered agent chooses its new hypothesis not randomly but by copying the hypothesis of a chosen agent in that population.
2. *Scattering (by active recruitment):* every active agent X selects another agent Y at random; if Y is inactive, it is sent in the local population pointed by X's hypothesis. Similarly to the backscattering subphase, in order to preserve the hypotheses distribution in the host node population, the scattered agent selects its new hypothesis not randomly but by copying the hypothesis of a chosen agent in that population (if there are no agents at all in the host node population, then the new hypothesis is chosen randomly).
3. *Internal diffusion (by passive recruitment):* every inactive agent X selects another agent Y at random; if Y is active, X takes Y's hypothesis.

SDST is illustrated in figure 3 on the studied game-tree. As for the previously discussed algorithm, a majority of agents in the root node population first points toward the right move (best move in a purely Monte-Carlo sense) before reorienting toward the left move (best move in the minimax sense). However, the distribution of the agents in the entire metapopulation is now dynamically regulated: most of the agents diffuse in the right part of the game-tree in the first four iterations, and then diffuse back to the left part of the tree in the following iterations. Also, only the regions of interest are visited: for example the entire region after Max's right move at the first ply and Min's right move at the second ply is ignored because the entire subtree leads to a win for Max (no agent becomes active in Min's node population to send inactive agents in this area).
Under normal conditions, an equilibrium between the scattering and backscattering forces eventually appears, leading to a statistically stable metapopulation. A very interesting property of SDST is that this equilibrium depends on the number of agents used. Asymptotically if enough agents are used, the equilibrium is equivalent to minimax. This is the case of the simulation presented in figure 3: at iteration 12 the metapopulation stabilises in the left part of the game-tree.

## 4 DISCUSSION

In the previous sections, we have introduced Stochastic Diffusion Search applied to Trees (SDST), a swarm intelligence heuristic performing forward planning. SDST is very similar to classical Monte-Carlo Tree Search (MCTS) algorithms in its functioning, but is conceptually radically different. While classical MCTS requires a central processing unit executing the algorithm in a sequential way (with a permanent and complete access to the data), the problem solving ability in SDST emerges from the collaboration of homogeneous agents with limited computational capacities. This distinction can be expressed differently by saying that classical MCTS and SDST are implemented on Abstract Platforms of Computation of fundamentally different nature (sequential algorithmic vs. parallel distributed). Importantly, the facts that classical MCTS could be implemented on a Turing equivalent decentralised system (such as a cellular automaton), or that SDST is being executed on a digital computer are not relevant. The concept of APC allows layered levels of abstraction, and what matters is the nature of the APC at the level at which the forward planning problem is being solved. This last remark suggests that the broad notion of computability that emerged in recent works ([12], [18]) needs new tools to be studied and in particular, a characterisation of computational systems in terms of their Turing power is not sufficient any more.

In addition to the main result, our work introduces a meta-level in the Swarm Intelligence paradigm: SDST relies on emergence both at the level of the agents forming local populations and at the level of the local populations forming a dynamically moving metapopulation. Individual agents are themselves unable to compare the different moves available to them, but their *interaction* leads to the exploitation of the most promising branches at each node of the game-tree. Similarly, local populations have a weak level of play when taken independently (branches are chosen without tactical sense), but their *interaction* makes a high level of play emerge (SDST is asymptotically equivalent to Minimax). Interestingly, the concept of metapopulation (a population of populations) exists in biology to refer to the dynamical coupling that appears between different populations of social insects [10].

Finally, the work presented here takes on its full meaning only if one recognises that it might have some interesting insights to provide about cognition. In fact, SDS has already been proposed as a model for neural activity: the one-to-one communication makes it a plausible candidate, and there exists a connectionist spiking neuron version of SDS called NESTER (for NEural STochastic nEtwoRk) [16]. Also in SDS, contrary to most of the other swarm intelligence heuristics[6], the meaning is embedded in the entire population instead of being simply supported by individual agents. This property is due to the partial evaluation of solutions: in the case of string matching for example, the position of the solution after convergence is indicated by the formation of a cluster of agents, possibly dynamically fluctuating (in the case of a partial match, agents will keep exploring the text while the cluster will globally stay on the best match).

[6] Ant Colony Optimisation also shares this property

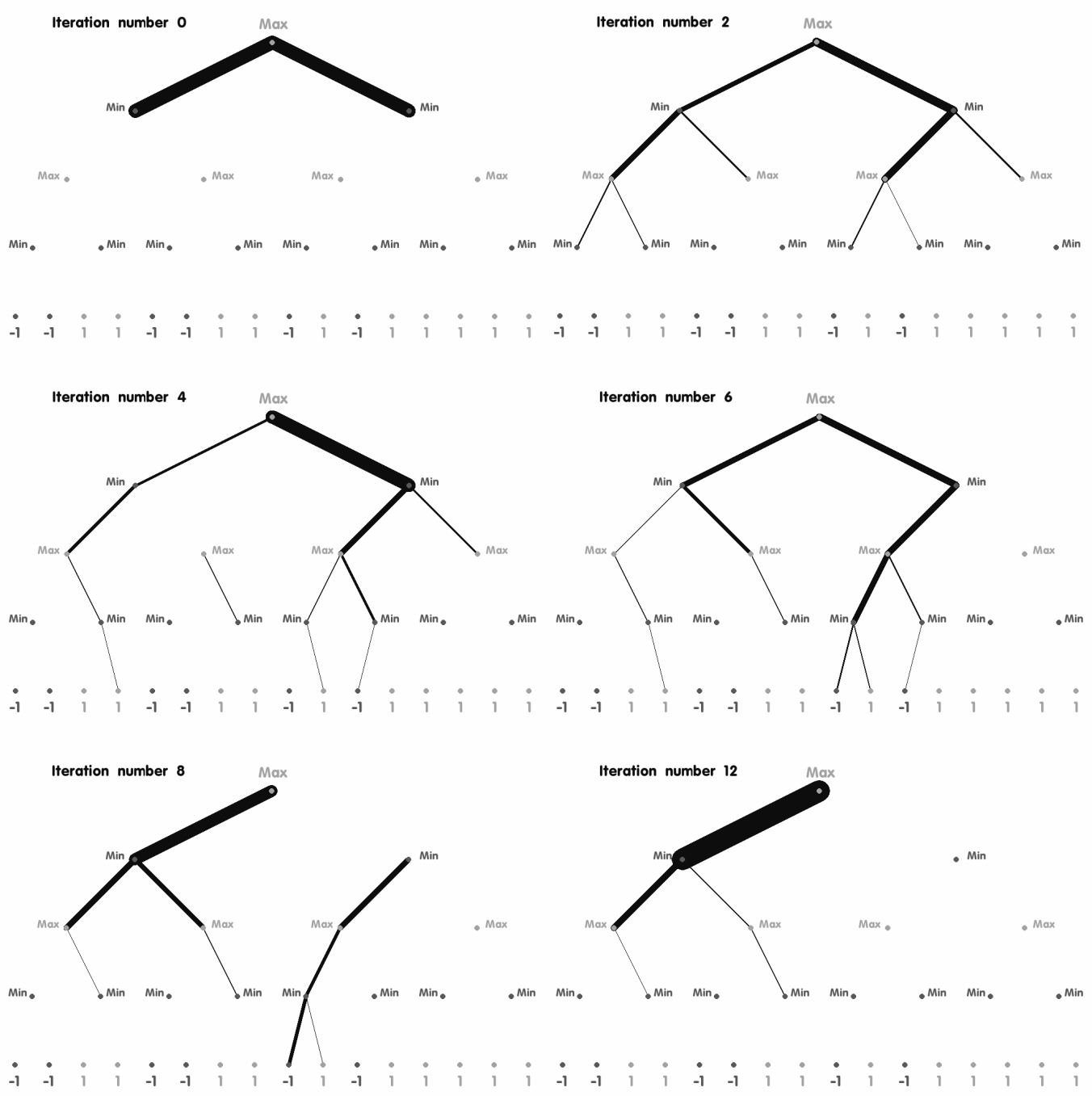



**Figure 3**: Illustration of SDST: Evolution of the distribution of the agents in the entire game-tree (iterations 0, 2, 4, 6, 8 and 12 shown, total number of agents = 100). Each branch has an area proportional to the number of agents in the parent node population supporting the move corresponding to the child node population.

In the neural model NESTER, this property leads to the synchronisation of the firing of neurons at convergence; "*hence in this model oscillatory behaviour may be a result of, rather than a cause of, the binding of features belonging to the same object*" [16]. In addition to giving a new theoretical solution to the binding problem [15], the ability to allocate efficiently and dynamically the cognitive resources to the search task has been proposed as a model for neural attention [7].

In their survey [4], Browne et al. concluded that:

> "Over the next five to ten years, MCTS is likely to become more widely used for all kinds of challenging AI problems. We expect it to be extensively hybridised with other search and optimisation algorithms and become a tool of choice for many researchers. In addition to providing more robust and scalable algorithms, this will provide further insights into the nature of search and optimisation in difficult domains, and into how intelligent behaviour can arise from simple statistical processes."

Although it was not conceived for practical AI purposes, we believe that SDST pertains to the type of hybridised algorithm Browne et al. had in mind. In particular, by integrating MCTS into the swarm intelligence paradigm, we believe that SDST indeed manage to "provide further insights (...) into how intelligent behaviour can arise from simple statistical processes."

## ACKNOWLEDGEMENTS

We would like to thank the John Templeton Foundation for supporting this research, and the reviewers for their useful comments.

## REFERENCES

[1] I. Aleksander and T.J. Stonham, 'Guide to pattern recognition using random-access memories', *Computers and Digital Techniques, IEE Journal on*, **2**(1), 29–40, (1979).
[2] P.D. Beattie and J.M. Bishop, 'Self-localisation in the 'SENARIO' Autonomous Wheelchair', *Journal of intelligent & robotic systems*, **22**(3), 255–267, (1998).
[3] J.M. Bishop, 'Stochastic Searching Networks', in *Artificial Neural Networks, 1989., First IEE International Conference on (Conf. Publ. No. 313)*, pp. 329–331. IET, (1989).
[4] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, 'A survey of monte carlo tree search methods', *Computational Intelligence and AI in Games, IEEE Transactions on*, **4**(1), 1–43, (2012).
[5] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, 'Monte-carlo tree search: A new framework for game ai', in *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, pp. 216–217, (2008).
[6] Matthew Cook, 'Universality in elementary cellular automata', *Complex Systems*, **15**(1), 1–40, (2004).
[7] K. De Meyer, J.M. Bishop, and S.J. Nasuto, 'Attention through Self-Synchronisation in the Spiking Neuron Stochastic Diffusion Network', *Consc. and Cogn*, **9**(2), 81–81, (2000).
[8] H.J. Grech-Cini and G.T. McKee, 'Locating the mouth region in images of human faces', volume 2059, (1993).
[9] L. Kocsis and C. Szepesvári, 'Bandit based Monte-Carlo Planning', *Machine Learning: ECML 2006*, 282–293, (2006).
[10] R. Levins, 'Some demographic and genetic consequences of environmental heterogeneity for biological control', *Bulletin of the ESA*, **15**(3), 237–240, (1969).
[11] R. Linggard, DJ Myers, and C. Nightingale, 'The Stochastic Search Network', in *Neural Networks for Images, Speech, and Natural Language*, 370–387, Chapman & Hall, Ltd., (1992).
[12] C. S. Matthew, J. M. Bishop, J. S. Nasuto, E. B. Roesch, and T. Tanay, 'Abstract platforms of computation', *AISB*, (2013). submitted.
[13] S. Nasuto and M. Bishop, 'Convergence analysis of stochastic diffusion search', *Parallel Algorithms and Applications*, **14**(2), 89–107, (1999).
[14] S.J. Nasuto, *Resource Allocation Analysis of the Stochastic Diffusion Search*, Ph.D. dissertation, University of Reading, 1999.
[15] S.J. Nasuto and J.M. Bishop, 'Neural Stochastic Diffusion Search Network-a theoretical solution to the binding problem', in *Proc. ASSC2, Bremen*, volume 19, (1998).
[16] S.J. Nasuto, J.M. Bishop, and K. De Meyer, 'Communicating neurons: A connectionist spiking neuron implementation of stochastic diffusion search', *Neurocomputing*, **72**(4), 704–712, (2009).
[17] S.J. Nasuto, J.M. Bishop, and S. Lauria, 'Time complexity analysis of the stochastic diffusion search', *Neural Computation*, **98**, (1998).
[18] Gualtiero Piccinini and Andrea Scarantino, 'Information processing, computation, and cognition', *Journal of biological physics*, **37**(1), 1–38, (2011).
[19] Hava T Siegelmann and Eduardo D Sontag, 'On the computational power of neural nets', *Journal of computer and system sciences*, **50**(1), 132–150, (1995).
[20] R.M. Whitaker and S. Hurley, 'An agent based approach to site selection for wireless networks', in *Proceedings of the 2002 ACM symposium on Applied computing*, pp. 574–577. ACM, (2002).

# Toward a Unified View of Computation in Neural Systems: A Reply to Shagrir and Piccinini

**Frank R. Faries**[1]

**Abstract.** Among those who would call themselves computationalists about the mind, there is not a clear sense of the notion of computation most appropriate for neural systems. The contrasting works of Oron Shagrir and Gualtiero Piccinini are one illustration of the divide among philosophers. The dispute turns on an ontological distinction regarding whether or not computation is an observer-independent phenomenon. This seems an incommensurable difference. However, if we conceive of neural computation in a generic sense—that is, as neither analog nor digital, but *sui generis*—and fix our concepts accordingly, these contrasting views are shown to be closer than they appear.

## 1 INTRODUCTION

Investigations in cognitive science attempt to bridge the gap between mental functions and the activities of cells in the brain that make up their neurobiological underpinnings. What is needed for this inquiry is an intermediate vocabulary of the processes occurring below the level of folk psychology and above the level of neural processes. A well-received view in cognitive science posits computation as one such process. Yet even among those who employ this term in explanations of mentality, there is a division regarding just what is taken to be meant by computation as it regards the brain. Among philosophers the term is most frequently invoked in discussions of computationalism, that is, whether or not computation is necessary and sufficient for a mind. Yet even among those who might call themselves computationalists, there is not a clear sense of the notion of computation most appropriate for neural systems.

An illustration of this divide lies in the contrasting work of Oron Shagrir and Gualtiero Piccinini. While there is a great deal of overlap in the two authors' positions on computation, they part ways at a critical point. Each would call himself a computationalist. Both employ a notion of computation growing from the tradition of Alonzo Church and Alan Turing [1], and influenced by Fodor [2]. Both *might* be willing to admit that "the brain is a computer", but they would not agree over what such a claim amounts to. For Shagrir the brain's status as a computer is merely a matter of convention, relevant only to particular observer's interests to explain certain semantic tasks. For Piccinini, however, computation in the brain is an observer-independent phenomenon individuated entirely without the need for semantics. Shagrir does not go so far as to agree with John Searle's claim that computation "is not discovered within the physics" [3], but he does concede that whether something is a computer is interest-relative. By contrast, Piccinini wishes to assert that computation is an intrinsic part of reality—something that happens in certain physical systems regardless of the interests of the observer. It appears that due to the enormous chasm between the consequences of the author's views their positions would be incommensurable. However, if we fix our concepts appropriately, as Piccinini does, we might find a way to span this deep divide, or at least draw the opposing sides more closely in alignment. Specifically, to reconcile these positions we must (i) keep a firm grasp on the norms of mechanistic explanation, of which computational explanation is a sub-species, (ii) conceive of computation in a generic sense, and (iii) recognize computation in the brain as neither analog nor digital, but as a kind all its own.

The remainder of this paper will proceed as follows. In Section 2 I will provide a précis of Shagrir's position and elaborate the pitfalls he wishes to avoid, and his positive claims about computation. In particular, Shagrir acknowledges the unique successes of computational modeling and explanation in the brain, but recognizes that digital computation is unsuited for explaining cognition. Furthermore, Shagrir's notion of computation necessarily involves representation, and he is skeptical about the prospect of a notion of representation capable of distinguishing computing from non-computing systems. Thus he concludes that while physical systems really do implement certain computations, which computations are the right ones is up to the interests of the observer. Section 3 offers Piccinini's adjustments to computational taxonomy, in particular his generic sense of computation—of which digital and analog computation are both sub-species—and his amended notions of information-processing, as well as further distinctions within the realm of computing systems. Neural computation, he contends, is neither digital nor analog, but is itself a unique kind of computation that is neither continuous nor discrete. Computations are individuated by appeal to their functional properties, without the need for external semantic content. Section 4 deploys this amended taxonomy in Shagrir's account, and finds favor with his position. This will show that a weaker form of computationalism satisfies the requirements of both authors, while avoiding the dangers to which computational theories can succumb. Section 5 addresses objections to this new taxonomy, and offers the most charitable account of Shagrir's resistance to it. Even on this reading, however, if neither author will grant that their respective positions can be unified, I will at least show them to be much closer in alignment than initially thought. Furthermore, I hope to highlight the common elements of these contrasting views which underscore the fundamental precepts of a unified view of computation in neural systems.

## 2 SHAGRIR ON COMPUTATION

[1] Dept. of Philosophy, University of Missouri – St. Louis, St. Louis, MO 63121, USA. Email: frank.r.faries@umsl.edu.

Shagrir [4] attempts to provide a meaningful, principled sense of computation that is relevant for explaining activity in the brain. He claims that there are two conditions typically involved in defining computation. These conditions reflect the classical sense of computation which Shagrir uses as the foil for his argument. That is, computation is information-processing, and formal in the sense espoused by Fodor, who famously holds (as Shagrir does) that there can be no computation without representation. These conditions are invoked for the purpose of establishing a principled definition of computation—one that marks off systems that truly compute from those that do not. Shagrir contends that neither of these conditions is adequate to establish an observer-independent sense of what it means to compute. The targets in Shagrir's attack are the concepts of representation, which he takes to be necessary for computation, and algorithmicity, one of three constraints composing the formality condition.

The formality condition is borrowed from the work of Jerry Fodor [5], and has typically been associated with three features: mechanicalness, abstractness, and algorithmicity. Mechanicalness can have two connotations, neither of which is ultimately helpful in demarcating computing from non-computing systems. First, mechanicalness can refer to a tradition in the philosophy of science regarding the concept of a mechanism. Here mechanisms are taken to mean "a set of entities and activities organized such that they exhibit the phenomenon to be explained" [6]. Computational systems are indeed mechanisms, on this view, but then so are scores of other physical systems which are not computing systems. The second kind of mechanicalness simply means a process that is sensitive only to structure, but not to meaning. The rules of inference are taken to be a prime example, as they hold regardless of the content of the axioms being adduced. Again, this second sense of mechanicalness fares no better than the first, as there are many physical processes (e.g., digestion) which are formal or structural in the way described above, yet are not computing anything.

The second feature of the formality condition is abstractness, or a description of the system strictly in terms of its formal (i.e., mathematical, logical, syntactical) characteristics. That is, the abstractness feature requires that the system be subject to description that abstracts away from its physical characteristics. Of course, when talking about physical systems, it is required that the concrete system implements the abstract description. The meaning of implementation is by no means clear and uncontroversial (see [7], [8]). Roughly speaking we can say a physical system implements an abstract description if there is a "mirroring" between the states and operations of the physical system and the states and operations of the abstract description. Here again, this feature does not amply serve to distinguish computing from non-computing systems, as there are mathematical descriptions which non-computing physical systems implement. The solar system, for example, is subject to a mathematical description of the movement of its planets, yet we do not wish to say that the solar system is computing.

Lastly, as Shagrir has insisted in earlier works, "Being algorithmic is not necessary for being computational" [9]. Shagrir's uneasiness about algorithms is familiar to defenders of computationalism. When speaking of the algorithm being implemented in a physical computing system, the computationalist desires a sense of algorithm that is not so permissive as to be vacuous or trivial, but not so restrictive that it does not capture a relevant or intuitive sense of what it means to compute. And in any case the algorithmicity constraint, as Shagrir sees it, is still inadequate. He cites the existence of non-algorithmic computers to show that the requirement of an algorithm is insufficient. Ultimately, he concludes that because each of the three features of the formality condition is itself insufficient to establish a satisfactory definition of computation, the condition as a whole is inadequate.

Returning to the first condition, Shagrir describes Fodor and Pylyshyn's notion of computation as the processing of information and necessarily involving representations. In doing so, Shagrir plans to attack computation in its classical form. Computation on this view is digital, or roughly, defined over discrete variables. Yet as noted above, Shagrir fears that digital computation in this sense is inappropriate to philosophy of mind, as it does not mark a clear divide between computing and non-computing systems. Furthermore computation in the brain is information processing, which "roughly means that causal processes in the nervous systems are mappings from one brain state $B_1$, which represents some feature $W_1$, to another brain state, $B_2$, which represents $W_2$" [10]. Therefore the processing of information, on this account, requires representations. However, as Shagrir admits, the terms "information" and "representation" are dangerously vague. We want an entity to pick out the meaningful elements of causal commerce in implementing a computation, but it is not clear what the ontological commitment is. In any case there is no sense of either term which draws a principled division between systems that compute and those that do not. For this reason, these concepts, so integral to most forms of computation, do not in fact serve to define the very term to which they are so frequently attached.

Thus we have a snapshot of the trepidations Shagrir has about computation. While Shagrir plays down the threat of pancomputationalism, he stresses the need for a meaningful sense of computation relevant to philosophy of mind. But what an odd position to hold! The theoretical space which Shagrir carves out is strange indeed, as he agrees with Churchland, Koch, and Sejnowski [11] that "whether something is a computer depends on whether someone has an interest in the device's abstract properties and in interpreting its states as representing states of something else". But he will not go so far as Searle and say the notion of computation is not even capable of being true or false—computation is still "out in the world", and the relationships it describes are very real.

This is in line with Shagrir's positive account of computation. To call something a computer is a strategy to explain how a complex system performs a semantic task. This strategy is successful when we identify the computational structure of a system. That structure is identified as a correlation between the mathematical properties of the target system and mathematical properties of the objects being represented. This structure is very real, but the system is only computing when we apply the computational approach. The success of a computational explanation lies in explaining how the central nervous system represents an object in the visual field. As such, the correlation between properties of the CNS and properties of the object will account for the explanatory force of the explanation.

All of this makes sense if, like Shagrir, your concept of computation necessarily involves content and representation. As he says, "the notion that lies at the heart of computation is content: computers are systems whose dynamics are individuated

by the content of the representations over which these dynamics are defined" [12]. Indeed, if the brain *is* a computer, it is only because cerebral activity represents external entities by reflecting or simulating the formal relations of those entities. Thus, for Shagrir, the brain is an analog model (AM) computer. This means it satisfies two conditions associated with AM-computing: (i) it is engaged in information-processing in that it maps one set of representations to another, and (ii) its formal relations mirror the formal relations of what is being represented [13]. All of this points to what Shagrir sees as the true utility of a computational approach to the brain: explaining how neural activity tracks the environment.

But what, precisely, is the meaning of "representation", as Shagrir uses it? All we need, it seems, for one entity to represent another, is simply a mapping, or reliable causal correlation, between the entities. This correlation will occur at the level of the formal descriptions of the objects and activities contained therein.

In summation, Shagrir stresses the need to find a principled definition for computation adequate to separate computing from non-computing systems. However, the received views on computation—as a digital information-processing task requiring representations, or formal in the sense of being mechanical, abstract, and algorithmic—are inadequate. Digital computation simply does not seem appropriate to explain cognition. The most appropriate sense in which a brain is described as a computer is analog model computing. On this view, computing in the brain is an information-processing task that maps one set of representations to another, and the internal relations of brain processes have reliable causal correlations to the external relations of the objects they represent. Individuating computations requires identifying a semantic task, and thus computing is an observer-relative phenomenon. However, the syntactic relationships picked out by computations have truth-value independent of the interests of observers. Therefore, describing something as a computer is an explanatory strategy whose force comes from identifying reliable causal relations between the formal properties of the target system and the formal properties of the object being represented.

## 3 PICCININI ON COMPUTATION

In much the same way as Shagrir, Piccinini attempts to provide a clear account of computation. This concept should do justice to computer science and the computational theory out of which it was borne, and should also be appropriate to philosophy of mind. He believes the mind is a computing system, and attempts to discern exactly what kind of putative computational process is appropriate for explaining it. Contrary to Shagrir, however, Piccinini is concerned about the threat of pancomputationalism. The fear is "if everything is a computing system, it is unclear how computation could be interestingly related to inference, rationality, executing instructions, following rules, or anything else specific to explaining mental phenomena" [14]. The mind is no more special in its ability to be described as computing spatial memory as the wall is in its description as computing a Facebook status update. But we want to describe some things, accurately, as computers. Calling a wall a computer in any meaningful sense scrapes against common sense intuitions about what computers are, and does not do justice to the field of computational theory, a robust branch of mathematics, which is certainly not in the business of giving vacuous descriptions of walls.

Piccinini claims that there is a distinction to be drawn between those computational descriptions that do not make reference to the computations of the system, and those that explain the behavior of a system by appealing to the computations it performs. This distinction marks the difference between computational modeling and computational explanation. To construct a computational model which describes the wall as computing therefore satisfies the weak sense in which a computational description can be given of any sufficiently complex object, while doing no injustice to computability theory. Genuine computational explanation, however, is a special kind of mechanistic explanation, applying only to systems with special functional properties. That is, computational explanations gain their explanatory force to the extent that they identify mechanisms—the functional organization of a system's components—underlying the phenomena.

But the question remains, when genuine computational explanation appeals to this certain type of activity—computation—what exactly is it referencing? Piccinini's most recent work with Sonya Bahar offers an adjustment to the taxonomy of computation. This new meaning of "computation" diverges from classical digital computation and thus eludes the criticisms leveled by Shagrir. Computation in the brain, on this new account, is computation in a *generic* sense; "Computation in the generic sense is the processing of vehicles (defined as entities or variables that can change state) in accordance with rules that are sensitive to certain vehicle properties and, specifically, to differences between different portions (i.e., spatiotemporal parts) of the vehicles" [15]. This definition marks a change from [16] in which digits, strings, and symbols were invoked to describe a computing mechanism. However, it does still satisfy the two conditions set out in that previous work: computation does not require representation, and computing systems are mechanisms. Furthermore, some features of this new account are similar in kind to the features of the previous account. Namely, computation, in the generic sense: (i) preserves the objectivity of computation, (ii) requires explanations to say how computation explains behavior, (iii) accounts for paradigmatic examples of computing mechanisms, (iv) excludes paradigmatic examples of non-computing systems, (v) explains miscomputation, and (vi) provides a taxonomy. The change is in abandoning the reduction of neural computation to a digital architecture. This move is motivated by empirical evidence and by the structural and temporal constraints of the brain. This is computation in the context of the causal nexus of the brain. Based on our known facts about the nervous system, this computation is most likely not (strictly) digital or (strictly) analog. That is, neural computation is sui generis, complete with its own mathematical theory. This leaves the ultimate nature of computation as it happens in the brain an empirical matter, but one with promise of an answer from computational neuroscience.

Accompanying this new sense of computation are further changes in the taxonomy. The first is a clarification of the relationship between computation, cognition, and information-processing. If a system is processing information, it is a computing system, but the converse does not hold. If a system is computing it need not be processing information. Cognition does

involve the processing of information, in at least three important senses: non-semantic, natural semantic, and non-natural semantic. Non-semantic information is merely mutual information, or statistical dependency, between a transmitter and a receiver: a stochastic signal. Cognition processes neural signals which are statistically dependent on other variables. Natural semantic information is causal correlation between a signal and its source. Any system that processes mutual information, by definition, processes natural semantic information, as a statistical dependency between signal and source is a causal correlation. Therefore, cognition processes information in this sense. Lastly, non-natural semantic information refers to internal states that can represent correctly or incorrectly, or "full-blown" representations (taken here to mean representations that have the capacity to *misrepresent*). At least some cognitive processes (e.g., those involving language) tend to deal with representations that can correctly or incorrectly represent external objects. Therefore cognition involves information in this last sense as well.

Furthermore, information processed by computations is medium-independent. Information processed by a concrete computation is a vehicle over which the computation ranges. As concrete computations and their vehicles are defined independently of the physical media that implement them, they are medium-independent. Thus the same computation can be implemented in various different physical media, provided the media has sufficient degrees of freedom, and the components of the mechanism are functionally organized in the right way.

Finally, computations are individuated by their functional properties, which are "specified by a mechanistic explanation without appealing to any semantic properties" [17]. This thesis about computational individuation must not be conflated with any thesis about computational causation. The only properties of a computation that are causally relevant are purely formal (or syntactic), non-semantic properties.

Under this new concept of computation, both digital and analog computation retain their distinctiveness, and both fall under the umbrella of generic computation. Each has also taken on a slightly new meaning. Concrete digital computation manipulates strings of digits according to rules defined over those digits. Here 'digits' are taken to mean discrete state variables which can be concatenated and whose type can be unambiguously distinguished by the computing system. This definition of digital computation must be distinguished from three other varieties. Classical digital computation in the tradition of Fodor and Pylyshyn [18] is defined over language-like objects, and necessarily involves representations. Digits in this new digital computation need not represent. It is also distinct from algorithmic computation, or the requirement that computations follow an algorithm. But some Turing-computable computations do not follow algorithms. Lastly, it is distinct from mere Turing-computable computation, as digital computation may be Turing-computable or not. In fact, digital computation is a broader concept of which the above three are all subspecies. But in this manner paradigmatic examples of digital computers (e.g., Turing-machines, personal computers) can be properly taxonomized.

By contrast, the sense of analog computers being defined here rests on the notion of abstract analog computers put forth by Pour-El [19]. These systems are distinct from digital computers in that the variables they manipulate are continuous, and non-discrete. Admittedly, this is not the only line on which to draw a digital/analog distinction, as Maley [20] points out. However, as this account of computation does not involve representations, Maley's distinction is orthogonal to our current analysis.

Based on our best scientific evidence, the computation that occurs in the brain is not digital. Current evidence cites properties of neural spike trains as the functional elements of neural computation. Spikes were natural candidates for discrete elements of computation in the brain due to their "all-or-none" characteristics (i.e., individual neurons were either firing or not), yet research shows that spike trains are not actually viable as strings of digits. Without strings of digits there can be no digital computation. Thus computation in the brain is not digital computation.

In fact, current research indicates that "typical neural signals, such as spike trains, are graded like continuous signals but are constituted by discrete functional elements (spikes); thus typical neural signals are neither continuous signals nor strings of digits" [21]. That is, computation as it occurs in the brain is sui generis. This view carries with it some consequences.

First, if neural computation is not digital, there is no good reason to believe that formalisms from digital computability theory will give exhaustive characterizations of the functionally relevant properties of neural processes. Second, just because a digital computer can model a neural computation, that does not mean it can simulate it. Third, while it is unclear whether a human brain is more powerful than a Turing machine, if neural computation is not digital, "the question of whether neural computation is computable by Turing machines doesn't even arise" [22].

Piccinini's position provides a principled distinction between computing and non-computing systems, and a sense of computation appropriate for neural biological systems. However, this account also attempts to do justice to computer science and computability theory. As such, computation does not presuppose representations, and computing systems are mechanisms. Computation also satisfies the six features listed above. Computation is objective—it is "out in the world". It exists regardless of the interests of the observer. Computational explanation (as a kind of mechanistic explanation) will say how a computation explains a particular behavior. This will be done by uncovering the functionally organized entities and activities that exhibit the behavior in question. This definition of computation will contain paradigmatic examples of computing mechanisms, exclude paradigmatic examples of non-computing mechanisms, and furthermore, provide an accurate taxonomy in which to categorize the various sorts of computing mechanisms. The satisfaction of these requirements rests on the fact that vehicles over which a computation is defined are medium-independent, and either discrete or continuous. Therefore, because the inputs and outputs of a stomach, for example, are not medium-independent—their activities are not blind to physical properties of the states and activities of the system—digestion is not computation. Desktop computers, neural networks, Turing machines, and abstract analog computers, however, are all computers, and can be distributed into their more specific taxonomies accordingly. Lastly, this definition of computation can explain miscomputations. As computing systems are mechanisms, explained by the functional arrangement of activities and entities organized to process medium-independent vehicles according to rules defined over those vehicles, if the mechanism malfunctions, a miscomputation occurs. Thus

Piccinini has drawn a principled, intuitive distinction between computing and non-computing systems, and has done so without recourse to content or representations. We shall see below how the terminological adjustments Piccinini suggests can collapse the divide between his and Shagrir's positions, or at least draw them closer.

## 4 TOWARD A UNIFIED VIEW

Let us think back to Shagrir's original objections to computation. First, he claims, though computation is typically described as information processing which necessarily requires representations, it is not clear that there is a sense of "representation" which truly separates computing from non-computing systems. Observer-independence does not seem to draw the line in the right place, as some putative non-computing systems (e.g., planetary systems, stomachs, washing machines) could plausibly represent observer-independently, yet paradigmatic computing systems like desktop computers or chess computers are obviously operating on symbols whose content is observer-dependent. Another sense of representation refers to symbol systems like the kind developed by Newell and Simon [23]. However, this sense is still insufficient, as analog computers operating over non-symbolic representations, and connectionist computing systems operating over representations with no combinatorial structure can attest.

The question thus arises: "So why keep representations?" Presumably Shagrir feel he needs representations as his AM-computing necessarily operates over maps of one set of representations to another. But what sense of "representation" is this? It seems that Shagrir only requires a sense of "representation" strong enough to support the second condition of AM-computing—that of simulation. By simulation, you will recall, Shagrir means only a "mirroring" between the relations of the target system and another. This need not, therefore, be any strong sense of representation. Indeed, the weak sense of information posited by Piccinini as a mere stochastic signal between transmitter and receiver, given the complexity of the brain, entails natural semantic information, or roughly the reliable causal correlation that Shagrir seems to be looking for. This does not seem too weighty a commitment. By dropping the troubling term "representation", Shagrir can still get the "mapping" he desires without worrying about what processes count as representations in the relevant sense, or determining which representations are the "right" ones. This seems to be a reasonable compromise.

But what of Shagrir's other qualm with computation: the "formality" condition given by Fodor? Specifically, algorithmicity in both its weak and strong form is unsuitable for describing computation in the brain, and in any case is inadequate; non-algorithmic analog computers and neural networks show this to be the case. Again, this need not trouble the new taxonomy. As was shown above, algorithmic computation is a restricted subcategory of digital computation. Digital computation, even in its broadest sense, is still inadequate to capture the functionally relevant properties of computation in the brain. Therefore, the fact that there are cases of non-algorithmic digital computing is not troublesome, and neither are analog computers and neural networks. All are accommodated under the new taxonomy.

Lastly, what about the interest-relative aspect of computation: the semantic task which is necessary, on Shagrir's account, to individuate computations? On a view of computation that does not make essential reference to representational content, it is possible to skirt this semantic view of computational individuation in favor of a functional view. The confusion comes in that for most program-controlled computers, external semantic descriptions are the easiest and most accessible way to gain understanding, or to program or control them. It may be, in fact, that many of these computations do in fact have external semantic properties. However, all of this is consistent with the claim that computations do not have external semantic content essentially. Though an external semantic description may be given of a particular computation, this does not mean that the computation cannot be individuated without reference to external semantic content.

Take for example Shagrir's overly simplified example of the "brown-cow cell" [24]. Briefly, the cell is a logic gate capable of having different meanings assigned to the same implementation. The cell can be described as an AND gate, thus firing when something is both brown *and* a cow. Or, with a variation in the voltage sensitivity of the components, the cell can be described as an OR gate, thus firing when the object in the visual field is either brown *or* a cow. Which computation is the "right" one, it is argued, will be determined by the semantic task defined by the computational problem. However, though it may be easier to decide the computational structure of the cell by appeal to external contents, it does not follow that it is impossible to identify the computation without reference to semantic content. The cell is taken to be a cell in the visual cortex, and thus is embedded within other entities and activities making up the whole central nervous system. The functionally relevant aspects of a computing mechanism can be determined (non-semantically) by investigating how the inputs and outputs of the mechanism interact with its context in the environment. In turn the functions which serve to individuate computations are interpreted to be wide (but not too wide). This means that in determining the computational identify of a system, it may be necessary to determine its function by looking outside the system. In the case of the brown cow cell, this might require examining its interactions with other cell populations, and its place in the greater causal nexus of the central nervous system. However, none of this requires essential reference to external semantic contents. Piccinini acknowledges the divergence of Shagrir's view from his own here, and attributes the dissidence to Shagrir's narrow construal of functional properties. This is a separate discussion that runs far afield of our project here. Needless to say, Shagrir offers no principled reason for the construal of functional properties , and instead clings to a semantic view of computational individuation. This might be the biggest bullet Shagrir would have to bite to integrate his position with Piccinini's. However, as he offers no overt reason why he chooses the narrow view of functional properties that he has, it might be safe to presume he would at least consider dropping it.

In fact, under Piccinini's new taxonomy, Shagrir seems to get most of what he wants with little to no cost to his own view. We finally have a principled distinction between computing and non-computing systems, and have a generic sense of computation suitable for philosophy of mind as well as neuroscience. This principled definition of computation not only accommodates paradigmatic examples of computing and non-computing

systems, but further categorizes them to an extent unachievable by digital computation. We have eliminated the term "representation" and all of its ontological baggage, while still retaining the reliable causal correlation which caused Shagrir to invoke it in the first place. Lastly, we have a means of individuating computations without appeal to their external semantic properties. Though computations may in fact have external semantic contents, as Shagrir is quick to assume, we need not make essential reference to them in order to individuate computational tasks, or computations themselves. All of this is achieved at a low cost, and it entails an added benefit: with a principled distinction between computing and non-computing systems comes an answer to the threat of pancomputationalism. Now computation can remain an intrinsic fact about the world, and only the right kinds of functionally organized systems will be capable of implementing computations. With this, the explanatory value of computation is preserved.

## 5 SHAGRIR'S REPLY

One might argue that the move to this new taxonomy is too quick, and that it does violence to Shagrir's theory. Presumably, Shagrir could object on three separate bases: the removal of representations, the broadening of functional properties, and the apparent ad hoc-ness of this new taxonomy. However, even if we concede to Shagrir on any or all of these counts, it should still be clear that the chasm separating the positions of Shagrir and Piccinini is not as wide as it first appeared.

Regarding the first objection, Shagrir might claim that the notion of computing he finds appropriate for describing the brain requires representations, and the removal of representations from a theory of computations simply will not stand. Computation is being invoked to explain mental processes, and these typically refer to objects outside of the computing system. Therefore, it is necessary to espouse a theory of computation which makes essential reference to the representation of external objects.

Here again the same point will be stressed as above. The depth of Shagrir's commitment to representation for analog model computing need only go so far as to allow for a mirroring of relations between the brain and the objects the brain represents. This mirroring need only capture the relevant external relations, and is characterized as a reliable causal connection between brain states and states of the world. This sense of representation is captured in a very weak sense of information. If the brain processes stochastic signals, then there exists a statistical dependency of signal transmitters and signal receivers. This dependency can be seen as a reliable causal connection, and hence we can claim that neural processes carry natural semantic information by virtue of the presence of non-semantic information, in conjunction with the complexity of the brain. While this may not be entirely satisfactory for Shagrir, it does appear to offer the same (or only slightly weaker) sense of representation he requires, only with a different name.

Shagrir's second objection might be that the move to wide individuation of functional properties is unwarranted, or carries consequences which are unappealing. The only problem with this assessment is that, until this point, Shagrir's position has not made it clear what these consequences might be. Wide functional individuation, whatever its warrant or consequences, does not appear to be a substantial ontological commitment at first pass. It merely pays respect to the fact that complex computing systems are often caught in a complex causal hierarchy of other mechanisms. Indeed, the mechanist tradition of Craver [25] might readily admit this. If we wish to develop a sense of computation relative to neural biological systems, it might be best to take this fact into account. Otherwise, if Shagrir is still unwilling to budge on this point, the burden of explanation is on him as to why.

Finally, Shagrir might simply cry "ad hoc" to these adjustments, claiming them to be a reaction to the failure of digital computation in neural systems. However, rather than an admittance of failure, this new taxonomy seems to embrace the wealth of evidence neuroscience has brought to bear on neural systems. It may be that Piccinini pushes the empirical envelope a bit, but as the computational structure of the brain is ultimately an empirical question, this does not seem like such a bad idea. Rather than an ad hoc move, these adjustments reflect a considered appreciation of the complex structure of the human mind, and respect the differences between modeling a system, and really explaining it.

## 6 CONCLUSION

The preceding was a cogent presentation of the motivations behind a unified view of computation in the brain. In closing, it should be mentioned that it is by no means tautological that the respective positions of Shagrir and Piccinini are extensionally equivalent. Both authors advance the idea that the computations a computing system performs are very real, and have truth-values independent of the interests of observers. Shagrir wants to say that we do not find computers in the world, but we employ computational explanation as a means of identifying a correspondence in the formal relations of a target system and an object. Piccinini would agree that computational explanation, as a species of mechanistic explanation, identifies this correspondence to the extent that it uncovers entities and activities duly organized so as to exhibit the phenomenon in question. And all this can be done without the need for representations. If we recognize neural computation in a generic sense, as a class of computation all its own, we may yet find the precise computational structure of the mind. Semantics, it seems, are more trouble than they are worth. We can still be realists about computation, avoid semantic content, and do justice to computation as it is implemented in biological systems embedded in their environment.

## REFERENCES


[1] Turing, A. (1950). Computing Machinery and Intelligence. *Mind*, 236:433-460.

[2] Fodor, J.A. (1975). *The Language of Thought*. Cambridge, MA: Harvard University Press.

[3] Searle, J. R. (1990). Is the Brain a Digital Computer? *Proceedings and Addresses of the American Philosophical Association* 64 (November):21-37.

[4] Shagrir, O. (2006). Why we view the brain as a computer. *Synthese* 153:393–416.

[5] Fodor, J. A. (1980). Methodological solipsism considered as a research strategy in cognitive psychology. *Behavioral and Brain Sciences, 3*, 63–73.

[6] Craver, C. F. (2007). *Explaining the Brain*. Oxford: Oxford University Press.

[7] Chalmers, D.J. (1994). On implementing a computation. *Minds and Machines* 4(4): 391-402.
[8] Putnam, H. (1988). *Representation and Reality*. Cambridge, MA: MIT Press.
[9] Shagrir, O. (1999). What is Computer Science About? *Monist*, 82:131-149.
[10] Shagrir, O. (2010). Brains as Analog-Model Computers. *Studies in History and Philosophy of Science*, 41:271–279.
[11] Churchland, P. S., Koch, C., & Sejnowski T. J. (1990). What is computational neuroscience? In E. L. Schwartz (Ed.), *Computational neuroscience* (pp. 46–55.) Cambridge, MA: MIT Press.
[12] Shagrir, O. (1999). What is Computer Science About? *Monist*, 82:131-149.
[13] Shagrir, O. (2010). Brains as Analog-Model Computers. *Studies in History and Philosophy of Science*, 41:271–279.
[14] Piccinini, G. (2007). Computational modeling vs. computational explanation: Is everything a Turing machine, and does it matter to the philosophy of mind? *Australasian Journal of Philosophy*, 85(1), 93-115.
[15] Piccinini, G. and Bahar, S. (2012). Neural Computation and the Computational Theory of Cognition. Cognitive Science, doi: 10.1111/cogs.12012.
[16] Piccinini, G. (2007). Computational modeling vs. computational explanation: Is everything a Turing machine, and does it matter to the philosophy of mind? *Australasian Journal of Philosophy*, 85(1), 93-115.
[17] Piccinini, G. (2008). Computation without Representation. *Philosophical Studies* 137:205_241.
[18] Fodor J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture. Cognition 28:3-71.
[19] Pour-El, M. B. (1974). Abstract Computability and Its Relation to the General Purpose Analog Computer (Some Connections between Logic, Differential Equations and Analog Computers). *Transactions of the American Mathematical Society* 199: 1–28.
[20] Maley, C. (2011). Analog and digital, continuous and discrete. Philosophical Studies 155:117–131.
[21] Piccinini, G. and Bahar, S. (2012). Neural Computation and the Computational Theory of Cognition. Cognitive Science, doi: 10.1111/cogs.12012.
[22] Piccinini, G. and Bahar, S. (2012). Neural Computation and the Computational Theory of Cognition. Cognitive Science, doi: 10.1111/cogs.12012.
[23] Newell, A. and Simon, H. A. (1976). Computer science as an empirical enquiry: Symbols and search. *Communications of the ACM* 19: 113 – 26.
[24] Shagrir, O. (2006). Why we view the brain as a computer. *Synthese* 153:393–416.
[25] Craver, C. F. (2007). *Explaining the Brain*. Oxford: Oxford University Press.

# From Proactive to Interactive Theory of Computation

**Marcin J. Schroeder**[1]

**Abstract.** The paper answers question "What is computation?" presenting it as a special case of a more general natural process involving dynamics of interacting information systems. For this purpose, a more detailed description of the generalization of Turing machine called symmetric machine outlined in author's earlier paper is used as a model for natural computing in which pro-active character of the machine with differentiated functions of its components (head and tape) is replaced by symmetric interaction of functionally equivalent information systems of hierarchic structure. Turing's A-machine is a special case of such S-machine in which only local states of cell's are changing, but not instructions on the list. The importance of the interactive character of computation is analysed in the context of exclusively interactive processes of mechanics. Causality involved in physical implementations of Turing machine can be preserved in its interactive generalization at the higher, collective level of its structure.

## 1 INTRODUCTION

The question "What is computation?" presupposes identification of a more general concept, with computation being its special instance distinguished by some characteristics. Since the interest in computation had its main source in the concept of a Turing machine, it is frequently implicitly assumed that computation is what such a machine is doing, and the focus is shifted to the question "What can Turing machine do?" However, the issues of computability, or of the possibility to find effective methods of going beyond it in hyper-computation are not directly related to our original question, unless we restrict the concept of computation to the algorithmic description of the recursive functions defined for natural numbers.

Turing conceived his A-machine as an abstract description of the process performed by the human computer (non-human computers did not exist at that time) in simple arithmetical calculations. His next step was to generalize the work of the machine in modelling processes of the higher level of generality corresponding to logical reasoning in mathematics or manipulation of symbols in a more general context. However, his theoretical machine and its various technological realizations were artefacts in the sense that they were designed by humans to achieve some pre-designed goals. They were based on procedures derived from the use of language, in particular in the written form of language.

Universal Turing machine turned out to be surprisingly powerful in modelling a very wide range of processes performed by humans. This led to the fallacious conviction that modelling of the human way of describing reality with the use of language is equivalent to the modelling of human mind or alternatively to modeling of reality. Informal extensions of the Church-Turing thesis to all possible forms of computation going beyond its original statement about computability of recursive functions is an example of such conviction.

Thus, in the search for generalization of the concept of computation we should try to go beyond the realm of human artefacts, specifically beyond language. This justifies the choice of naturalization of computing as a way of generalization. Now we can formulate our original question in a more specific way. Is it possible to describe computation as a special instance of a more general natural process, such that its application in the realization of our human goals becomes the familiar model of computation?

The present paper is an attempt to present computation as such natural process involving dynamics of information. It is not a surprise that the conceptual framework for this purpose is based on the concept of information. After all, most frequently technological realizations of computation in the form of computers are used not for the purpose of finding values of recursive functions, but to process information in its diverse manifestations. Moreover, information became one of the most fundamental concepts in the description and study of natural phenomena.

Of course, the concept of information to be used here must be general enough to prevent re-imposing restriction of computation to the realm of language or other human artefacts. Although majority of theses of this article remain valid in several of the large variety of conceptualizations of information, the author will be using his own definition of information which thanks to its generality includes other ways of understanding information as special instances [1]. In particular, this definition is putting together as dual manifestations two competing and formerly considered contradictory concepts of information: more popular based on the concept of uncertainty or selection and another based on the concept of structure or form [2,3].

Thus, the concept of information is understood here as an identification of a variety, which presupposes only categorical opposition of one and many and nothing else. The variety in this definition, corresponding to the "many" side of the opposition is a carrier of information. Its identification is understood as anything which makes it one, i.e. which moves it into or towards the other side of the opposition. The preferred word "identification" (not the simpler, but possibly misleading word "unity") indicates that information gives an identity to a variety. However, this identity is considered an expression of unity or "oneness". We could interpret this formulation of the concept of information as a resolution of the one-many opposition.

There are two basic forms of identification. One consists in the selection of one out of many in the variety (possibly with a limited degree of determination described for instance by probability), the other in a structure binding many into one (with a variable degree of such binding reflected by decomposability of the structure). This brings together two manifestations of

[1] Akita International University, 010-1211, Akita, Japan.
Email: mjs@aiu.ac.jp.

information, the selective and the structural. The two possibilities are not dividing information into two types, as the occurrence of one is always accompanied by the other, but not on the same variety, i.e. not on the same information carrier.

In an earlier paper, the author presented his view on the role of the dualism between selective and structural manifestations of information in modelling its dynamic [4]. In order to fit computation into this model of information dynamic, it was necessary to reinterpret and generalize Turing machine as a composition of two fundamentally equivalent interacting information systems. This symmetric Turing machine was considered in that context as an example of mechanism in which information dynamics understood as a natural and universal phenomenon plays fundamental role comparable to the feed-back mechanisms or to the biological evolution.

In the present paper, the generalization of Turing machine outlined earlier serves different purpose. It provides us with a direction in which the concept of computation can be generalized in order to go beyond limitations of human artefacts. With the generalized and re-interpreted symmetric Turing machine we can attempt not only to answer the questions "What is computation?" or "How is computation related to other natural phenomena such as biological evolution?", but also we can try to find the limitations of traditional computation and the potential ways to overcome them.

## 2 SYMMETRIC TURING MACHINES

When we think about dynamics in terms of natural processes, the fundamental concept in its formulation is interaction. If we want to develop a naturalistic description of the information dynamic involved in computing, it is necessary to generalize computation, so that we have only mutual interactions. For this reason the author considered a generalization of Turing machine, called a symmetric Turing machine, or S-machine in place of the original A-machine [4]. The generalization considered before was minimal, in the sense that its only purpose was to fit the functioning of Turing machine into a dynamical process. Consequently, the S-machine was more different from the A-machine because of the re-interpretation, than because of the actual increase of generality. The description of the S-machine here is little bit different from the original in earlier paper, but the differences are of secondary importance.

Instead of the two very different functional systems of the Turing machine, a reading/writing head and a tape, we have here two information systems with basically the same structural and functional characteristics (this is why the machine is called symmetric). For the purpose of easy comparison with the A-machine, traditional names of the head and tape will be retained, although they lose their literal meaning. Both are compound systems characterized in terms of their components which are in respective local states, and of configurations of component states constituting their global states. To maintain as close as possible similarity with the A-machines, it was assumed that the components of both systems are numbered by natural numbers giving the structures of the system a linear, discrete partial ordering. This assumption is quite restrictive and more general configurations of components (e.g. with continuous indices in real numbers or with a partial order) could be considered. Components of the tape are called *cells*, of the head are called instruction list positions (*ilp*'s). Each of the cells can be in one of the $n$ states called *characters*, each of the instruction list positions in one of the $m$ states called *instructions*. The characters and instructions do not have any fundamentally different features. The distinction is only relative to the internal structures of the systems. What in the A-machines made the tape and head functionally different is here generalized into symmetric interaction in the computing dynamics of fundamentally similar systems.

S-machine is a two-part system equipped with a dynamical structure which in each state of the machine allows for an interaction of one particular cell with one particular instruction list position (*active cell* and *active list position*). Here too, we have a restriction, as the way these two composite parts interact could be more general allowing involvement of bigger number of active components. The choice of the pair of active components is a state of the S-machine, different from the global states of compound systems which form the two interacting parts of the machine, and of course different from the local states. Outcome of the interaction is a possible local change of the state of both active components (active cell and active *ilp*) and possible change of the state of the machine, i.e. the choice of the pair of an active cell and active instruction list position. The changes are described as possible, as in some cases the states may remain the same.

The three levels of computation dynamics require separate considerations. At the local level, each step of computation at the state of the machine (j,k), meaning that j-th cell and k-th *ilp* are active, is described by a function $(c'_j,i'_k) = \Phi_{loc}(c_j,i_k)$, where $c_j$ and $c'_j$ belong to the catalogue of n cell states (characters), and $i_k$ with $i'_k$ belong to the catalogue of m states of *ilp* (instructions). The function $\Phi_{loc}$ does not depend explicitly on j and k, but exclusively on the values of $c_j$, $i_k$.

The change of the state of machine is another function $(j',k') = \Phi_m(c_j,i_k)$, which in this case is in principle a compound function of j and k, but its dependence on j and k may be eliminated (by defining each j' as j+s where s is an integer function of the values of $c_j$ and $i_k$), and then it depends exclusively on the values of $c_j$, $i_k$.

Finally, we have a function $(c',i') = \Phi_{glob}(c,i,t)$ describing the change of the global states of the two interacting systems which form the machine after t iterations starting from the global states $c = (c_i : j \in J)$ and $i = (i_k : k \in K)$.

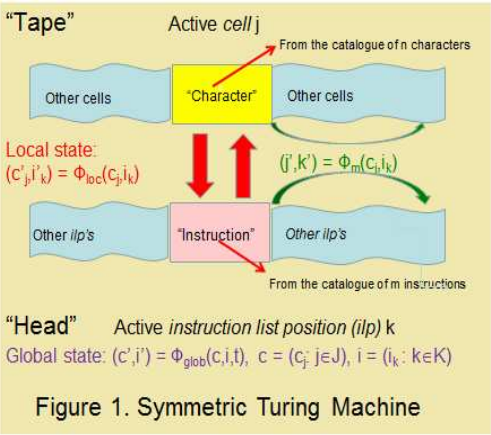



Figure 1. Symmetric Turing Machine

It is obvious, that the S-machine in which the state of each instruction list position (*ilp*) remains always the same allowing

identification of *ilp*'s with respective instructions is a usual A-machine. If we additionally assume that the local states of the tape remain unchanged, we get an automaton. The possibility of the change of instructions, i.e. local states of at least some instruction list positions makes the S-machine slightly more general than the A-machine. However, disassociation of the change of the local state of the tape from the head and describing it as an interaction between active local components of the compound systems is just a reinterpretation.

There is a natural question whether the result of every computation performed by S-machine on the tape can be computed by an A-machine. However, even if it can, S-machines open many interesting questions which simply cannot be stated for A-machines. As a result of computation, both interacting compound systems evolve. It is interesting (for many reasons) question whether it is possible to evolve a universal A-machine from simpler S-machines, and under what conditions it is possible.

Another, but related question is about the relationship between the S-machines and super-recursive algorithms introduced by Mark Burgin [5] or evolutionary computation considered by Eugene Eberbach [6] and others [7]. Here the answer is similar to that above. Since evolutionary computation presupposes the use of universal Turing machine for which a new type of algorithm with evolutionary characteristics is used, even if the outcome of computing on the tape can be reproduced, there is a wide range of problems which cannot be formulated, in particular, the problems of natural realization of computing in the biological systems.

## 3 INTERACTION vs. ACTION - CAUSALITY

S-machines with their dynamic based on interaction give us generalization going beyond traditional computing. This generalization can be understood as an interaction of two information systems, each of hierarchic character with two levels, the local one in which information has selective manifestation, and the global one in which information has structural manifestation.

Interaction is between components of the local level, but its outcome is exhibited in the evolution of structural manifestation of information at the global level. This type of process can be also realized in the form of a biological evolution of populations (interaction of the population and its environment), or as mechanisms of the feed-back type [4].

However, there is still a problem whether answering the question "What is computation?" in terms of the dynamics of interaction of composite systems is justified. Why is the dynamic of S-machines based on interaction in place of traditional one-way action of the head on the tape so important?

The answer to this question requires a broad perspective on the computing taking into consideration the fact that computing is performed by some actually existing machine, not by its theoretical model. The latter is only an interpretation created by human mind of what is actually happening in the machine. For instance, we can interpret a sequence of zeros and ones on a tape as a natural number presented in a binary numerical system, but without active participation of the human mind the sequence does not have any meaning beyond structural information in the form of configuration of the states of cells.

Any numerical interpretation is involving an external entity (mind) which escapes description of computation. The issue is similar to the recognition of the difference between a numeral and corresponding number. Without active participation of human mind there is no relationship between them. Actually, the relationship involves not only a conscious mind, but also its cultural load in the form of the numerical convention. Conscious, but uneducated mind would not be able to perform interpretation.

Since it is completely impractical to search for description of computation which involves as a component human mind with its all functions, we have to find conceptual framework in which mind's interpretative capability is absent, or is simplified to the degree permitting its modelling. It is natural to expect that the process of computation is autonomous and can be described as independent from its interpretation. Interpretation can be considered at a secondary stage of description.

The fact that computation is after all a fundamentally physical process is obscured by the old fallacy regarding the distinction between its digital and analogue forms [4]. In both types of computing the process is physical, involving a measurement of a physical system, but the interpretation of the outcome of the measurement is different. In digital computing only discrete values are assigned (typically 0 and 1), while in analog one the set of values is assumed to be continuous (interval of real numbers).

Some analogy can be found in the use of a slide rule. Both addition and multiplication of numbers involve sequential composition of material intervals of the rulers along a line. If we use linear scale on rulers, the outcome is the sum of numbers assigned to intervals. If we use logarithmic scale, the outcome is the product. Process here is identical, purely physical, the difference is in interpretation.

This means that computation of any type involves a physical process, which successively is interpreted. We can consider this interpretation as a secondary process of assigning the meaning to the states of the physical system and their transitions. To avoid problems of the meaning of meaning, we can use approach proposed by the author [8].

Meaning can be understood as a relationship between two information systems which preserves their structural characteristics (mathematically described as their respective logics in the form of lattices of closed subsets of appropriate closure space). In the linguistic context, information system of some fragment of reality is mapped into the information system of the language. Typically, the latter system has much smaller information capacity, allowing more economic handling of information.

Similar situation is in the case of computing. In the analogue computing, meaning is simply a process of assigning real number values to the states of a physical system, not much different from the usual concept of an observable.

In the case of digital computing, situation is more complicated. Here, physical measurements have only two values (or at least discrete and finite) and are performed at the local level of a cell of the memory or processing unit. Of course, we have here too some form of meaning relationship, but it is usually completely ignored. Nobody (except engineers working on the technological aspects of computer architecture) is usually interested in the meaning of the fact that some cell of memory has value 1 or 0.

Through the composition (typically multiple compositions) of functions, the meaning is associated with a system of cells according to the rules called "data structures". For Turing machines, typically meaning is an association of a global structure composed of the local states of cells with a natural number, more precisely with a numeral. This means, the meaning of a numeral is a global configuration of local states of cells.

Numeral can be associated with a natural number, but this association is outside of our study, because it requires involvement of human mind. It would be an absurd to claim that Turing machine itself is capable to associate numeral with a natural number.

Even if for typical issues of computation the engineering problem of the meaning of the local states of cells is of secondary interest, it is necessary to remember that it is the point where physics of computing is entering the picture. The original metaphorical image of a Turing machine as a head moving on an infinite tape and reading and printing characters cannot be considered literally. Thus, we have to take into account the physical picture of the world in which interactions play the fundamental role, as postulated by the third principle of Newtonian mechanics.

This brings us to the question about the relationship between the theoretical model of Turing machine and its physical world implementations. The former is based on logical relationships, which describe computation as a process only in the sense of a linear ordering of consecutive steps. This linear ordering and number of steps play the role of "time", while number of cells between the first nonempty character (typically the first 1) and the last is considered a measure in "spatial" dimension. However, both expressions are metaphorical.

Physical implementations of Turing machine operate with the assumption of involvement of causality and one-way actions. When the head is reading a character, we have one-way action from the cell to the head, when the head is printing we have reversed direction of action. Computation becomes a chain of causal relationships realized through one-way actions.

How can this picture of apparently physical realization of Turing machine be explained, when physics tells us that we have only interactions? The question is basically the same as that about the agreement between mechanical view of the world and causality.

Our sense of exercising free will makes causal relationship seemingly very natural. We believe that an apple falls on the ground, because Earth is acting on it with gravitational force. We also believe that the Earth is revolving around the Sun. But, it is not the physical view of these phenomena. Both the Earth and the apple are falling towards their centre of mass. Both the Earth and the Sun are revolving around their centre of mass. In the first case, the error in the common sense description is so small, that in practice we can always forget the actual physical description of the process.

It is much bigger problem, when the issue is not the magnitude of an error, but explanation of the mechanisms of phenomena. For instance, the concept of free will requires that the human subject has the ability to initiate and carry out an action towards something unilaterally. Thus, even if we overcome objections of Laplace regarding mechanical determinism in the form of his omniscient demon, there is still problem how to explain initiation of a causal chain.

Surprisingly, the concept of a causal relationship is incompatible with mechanics, in a similar way as kinematic theory of heat entered Loschmidt's "reversibility paradox". Every mechanical process is symmetric with respect to time inversion. What happens in one time-direction can happen in the other, under appropriate change of force direction (but not of the point where the force is applied!).

Causal relationship has as its necessary condition time precedence, and therefore is strictly dependent on time direction. Thus, we cannot derive causal relationship from the mechanical description of the world.

To resolve this paradox, we can follow Boltzmann in his resolution of the original paradox of Loschmidt, by giving the Second Law of Thermodynamics statistical status.

While mechanics describes dynamics of micro-components of the system (e.g. molecules of gas), thermodynamics describes the compound system in terms of macro-states, whose high value of entropy corresponding to the big number of microstates producing the same macro-state makes them preferable in dynamical evolution simply because of their higher likelihood.

In order to resolve our paradox, we can also assume that the causal relationship is a collective phenomenon governed by appropriate rules applicable to collectives. Thus, without getting into contradiction with the physical image of the world, we can say that lost control over the car caused collision with a tree, even if from the mechanical point of view both the car and the tree participated in the accident in exactly the same way.

The Second Law of Thermodynamics was later associated with more general rule for not necessarily mechanical complex systems described in terms of information. The connection is very clear through the association of physical entropy (which has physical dimension) with informational entropy (devoid any physical dimension) introduced by Shannon as a measure of information.

Since causality is a philosophical concept and does not belong to strictly scientific terminology, it is more difficult to associate it with the transfer of information in a formal manner. However, the description of information dynamics in an earlier paper of the author [4], together with the assumption of the collective character of causality allows its interpretation in this way. Actually, the view of causality as a transfer of information was already considered before, but in different context [9].

When we have interaction of complex systems with two hierarchically structured levels, local and global, associated by the duality of the selective and structural manifestations of information, the interaction (which lack direction) carried out at the local level can produce directed transformation of the structural information at the global level, which can be interpreted as a cause-effect relationship.

This very general outline requires further elaboration regarding justification of the association with causality, which however is beyond the scope of the present paper. Our objective is to present computation in its traditional form, as a special case of the more general process of interaction of two complex systems at the local level which results in a directed evolution at the global level. Computation with a symmetric Turing machine is an example of such process, which is fully consistent with physics. Moreover, computation in its naturalized form becomes a model for information dynamics, which can be useful in explanation of natural phenomena, such as cognition.

## 4 CONCLUSIONS & FUTURE WORK

Turing S-machines based on mutual interaction of their components as described above are minimal generalizations of A-machines satisfying the requirement of consistency with physics, and can have realizations in a large variety of natural systems.

Since general information systems can have multi-level hierarchic structures derived from the dualism of selective and structural manifestations of information, further generalization of computation may be considered in such architecture. An important example can be found in living objects.

The objective of the present paper is to answer the question "What is computation?" by putting computation in the context of natural processes, which involve interaction. S-machine generalization of Turing's A-machine served this purpose. On this occasion further generalizations appeared as possible ways to make computation similar to natural processes, but their study does not belong here.

Other generalizations worth future consideration mentioned above include processes which instead of iteration, i.e. indexing of steps in computation by natural numbers, can be described by parameters with continuous values, or with parameters belonging to a general partially (not necessarily linearly) ordered set. Yet another generalization can be considered with a variety of different ways active components interact.

The latter generalization is of special interest, as the assumption of a single pair interaction seems most artificial. The structure of cells and instruction list positions could have geometry very different from the linear and interactions of active components could depend on the geometry.

Finally, there is need for the theory of computing (with S-machines, or A-machines) in complex systems with "limited resources", i.e. with the limits on the number of components, either absolute, or "time"-dependent (i.e. with the limit as a function of the number of steps performed in calculation).

## REFERENCES


[1] M.J. Schroeder. Philosophical Foundations for the Concept of Information: Selective and Structural Information. In: *Procs. of the Third International Conference on the Foundations of Information Science, Paris 2005*, http://www.mdpi.org/fis2005 ,(2005).

[2] M.J. Schroeder. Quantum Coherence without Quantum Mechanics in Modeling the Unity of Consciousness, In: P. Bruza, et al. (Eds.*) QI 2009*, LNAI 5494, Springer, Berlin, Germany, pp. 97-112, (2009).

[3] M.J. Schroeder. From Philosophy to Theory of Information*, Intl. J. Information Theor. and Appl.*, 18 (1), 56-68, (2011)

[4] M.J. Schroeder. Dualism of Selective and Structural Manifestations of Information in Modelling of Information Dynamics, In: G. Dodig-Crnkovic, R. Giovagnoli (Eds.) *Computing Nature, SAPERE 7,* Springer, Berlin, Germany, pp. 125-137, (2013).

[5] M. Burgin. *Super-recursive Algorithms*. Springer, New York (2005).

[6] E. Eberbach. Toward a Theory of Evolutionary Computation. *BioSystems,* 82 (1), 1-19, (2005).

[7] D. Roglic. The universal evolutionary computer based on super-recursive algorithms of evolvability. arXiv:0708.2686 [cs.NE], (2007).

[8] M.J. Schroeder. Semantics of Information: Meaning and Truth as Relationships between Information Carriers, In: C. Ess & R. Hagengruber (Eds.) *The Computational Turn: Past, Presents, Futures? Procs. IACAP 2011, Aarhus University – July 4-6, 2011*. Monsenstein und Vannerdat Wiss., Munster, Germany, pp. 120-123, (2011).

[9] J. Collier. Causation Is the Transfer of Information. In: H. Sankey, (Ed.) *Causation, Natural Laws and Explanation.* Kluwer, Dordrecht, pp. 279-331, (1999).

# Computational Complexity: An Empirical View

**Maël Pégny** [1]

**Abstract.** Computational complexity theory (CCT) is usually construed as the mathematical study of the complexity of computational problems. In recent years, research work on unconventional computational models has called into question the purely mathematical nature of CCT, and has revealed potential relations between its subject matter and empirical sciences. In particular, recent debates surrounding quantum computing have raised the possibility of a new computational model, one based on quantum mechanics, which may be exponentially more efficient than any previously known machine. In this paper, I will show how those recent debates suggest an alternative, empirical view of CCT. I will then examine what are the fundamental arguments in the favor of this view, what are its consequences for our conception of CCT as a scientific field, and how further philosophical investigation should proceed.

## 1 Introduction: unconventional computing and philosophy

In the last decades, computer scientists have been showing a growing interest in the study of unconventional computational models. Those models stem from a broadened understanding of computation. In his original paper, A. Turing made it clear that this model was explicitly inspired by human, pen-and-paper computation (see [28], section 9). In many unconventional computational models, computation is no longer reduced to the discrete process of pen-and-paper computation: computation is any process that allows the encoding of an input value into the state of a given system, the processing of that information according to a given specification, and the reading of an output value. Physical, chemical and biological processes can be harnessed to perform computations in this broadened sense. Algorithms can no longer be conceived as abstract mathematical methods whose conception and theoretical properties are completely independent of implementation: algorithms now become abstract models of empirical processes.

This new approach to computing raises new challenges for computer scientists. In the case of Turing machines, and other historical models, empirical realizability was not a pending issue. The possibility of implementing these models by pen-and-paper computation was built into their very design. Many unconventional computational models, on the contrary, raise more serious issues of empirical realizability. Furthermore, defining the set of computable functions, or computational costs, according to these models, might depend on considerations of empirical realizability. In this broadened framework of computation, it becomes natural to think that not only logical and mathematical arguments, but also empirical ones, are relevant to decide whether a given model is realistic, and what its exact computational power is.

[1] Université de Paris-1, CEA-Larsim, France, email: Mael.Pegny@malix.univ-paris1.fr

To the extent of my knowledge, there is a striking assymetry in the philosophical litterature[2] between the treatment of computability issues and that of complexity issues, when it comes to analyzing the role of empirical considerations. Foundational issues of computability, especially the debate on the possibility of hypercomputation, have been the object of intensive scrutiny (see [12], [6], [27], [21], [9], to mention only a few references). The relations between computational complexity and empirical considerations have been the object of a somewhat lesser attention (see [22], [23], [16], [17], [30], [2], for a few remarkable exceptions). In particular, how this empirical turn might affect our conception of CCT as a discipline has not been the subject of a systematic philosophical treatment. The aim of this work is to synthetise the dispersed remarks present in the literature, and to provide such a systematic treatment.

After articulating the orthodox, mathematical conception of CCT (section 1), I use recent analysis of the Extended Church-Turing Thesis (ECTT) to present an empirical interpretation of CCT (section 2). I then examine how research on unconventional computational models, especially quantum computing, can support arguments in favor of this empirical interpretation (section 3). Finally, I discuss how this interpretation offers a new vision of the subject-matter of CCT, and its unity as a scientific field: CCT can be seen as an empirical science, founded on empirical hypotheses (section 4)

## 2 The subject-matter of CCT, and the nature of computation

What is the object of computational complexity theory (abbreviated as CCT[3])? In the literature, CCT is usually defined as a mathematical science quantifying and studying the difficulty of mathematical, computational problems ([13], [5]). Two remarks must be made to properly understand this definition.

First, complexity properties are, prima facie, properties of algorithms. They become intrinsic properties of a given problem through optimality results, which demonstrate that no algorithm can do better than a given, optimal one. It is this quantification over all possible algorithms, and the quest for optimality results, that distinguishes CCT from Algorithmic Analysis.

[2] By "philosophical literature", I do not only mean the literature written by professional philosophers, but also the foundational considerations published by physicists, computer scientists, and logicians.

[3] I will actually focus my attention on *uniform* computational complexity theory. The reason for this restriction is simply that I am only interested in computational models that can actually be implemented. If one thinks of the characterization of non-uniform complexity classes by resource-bounded Turing Machines receiving advice, one immediately faces the problem of the actual origin of such an advice. In full generality, it is unclear whether non-uniform computational models can be considered as empirically realizable models, or if they are to be considered as abstract tools used by the theorist. Even though that might be a very interesting philosophical issue, I do not think that such a short, synthetic work is the proper place to discuss such details.

Second, and foremost, CCT as a domain rests on three distinct hypotheses concerning the nature of its subject matter and the nature of its results:

- *Hardware-independence.* CCT concepts and results are independent of low-level hardware details. This is one of the central reasons why the number of steps needed by a given algorithm is described up to a linear factor, using the $O$ notation. The exact number of steps taken by an algorithm on a given machine could be sensitive to hardware details, but its gross magnitude should not show that sensitivity.
- *Model-independence.* To be well defined, computational costs must be considered within a given computational model, i.e. a mathematical model describing precisely how algorithmic instructions are to be written, and how they are to be executed. But the main concepts and results of CCT are independent from any particular choice of model, e.g. Turing machine, or register machine.
- *Relevance for implementation.* Despite hardware and model independence, CCT results are relevant to determine the concrete values of time-interval, and memory size, needed by a concrete device when it executes a given algorithm.

The first two hypotheses explain why CCT can be seen as a mathematical, abstract theory of computational problems. The third one states that its concepts and results remain relevant for concrete machine implementations. This is the reason why CCT can also sometimes be defined as a branch of engineering, one that quantifies and studies the resources needed by a computer to solve a given problem, without creating any philosophical controversy [20].

From these three hypotheses, a vision of computation, and its complexity, can be reconstructed. Computation is an abstract process that can be studied a priori by a mathematical theory. The results of this theory determine constraints relevant for any device that implements that abstract process: real runtime and memory size are thus determined to a certain order of magnitude by the logical steps described by the instructions of the algorithms.

This rather natural view of computation, and its complexity, has been called into question by recent work in unconventional computational models. Since this criticism mainly stems from recent challenges made to the hypothesis of model-independence, I will first take a closer look at it.

## 3 The core issue: model-independence and the Extended Church-Turing Thesis

The rigorous expression of model-independence is a specific hypothesis lying at the foundations of CCT, called the Extended Church-Turing Thesis (henceforth abbreviated as ECTT):

*All reasonable computational models are simulable by a Turing machine with at most polynomial overhead.*

All the proven relations of efficient simulation between models that have been explored so far provide evidence for this thesis. But as the list of computational models is open-ended, it currently remains a working hypothesis. It is not a mathematical proposition, and thus can be neither demonstrated nor taken as an axiom.

Despite its informal character, the ECTT is a fundamental hypothesis of CCT for at least two reasons. First, the ECTT allows complexity theoretists to reason within one particular model, most frequently Turing machines, knowing that any result regarding the polynomial or superpolynomial time-complexity of a given problem will remain untouched by a change of model. Without the ECTT, complexity properties could not be attributed to computational tasks in an absolute sense, but only relatively to a given model. The second reason is that the model-independence of polynomial time is one of the main arguments for the identification of polynomial-time complexity with tractability, and superpolynomial time-complexity with untractability (see, for instance, [19]).

The obvious escape in ECTT is the adjective "reasonable." It suggests the existence of computational models that go beyond the limits drawn by the ECTT, but are characterized as 'unreasonable.' Yet without any specification of what a reasonable computational model should look like, the ECTT is not only informal, but also not well-defined.

In recent years, several computer scientists and physicist, including P. Shor, U. Vazirani, E. Bernstein and D. Aharonov suggested that "empirically realizable"[4] should be substituted for "reasonable" in the above phrasing (see [26], [7], [4]). The suggestion is quite natural, since empirical realizability is a necessary condition of implementation.

A straightforward identification of "reasonable" with "empirically realizable" might nevertheless be questionable. As P. van Emde Boas noticed in [29], a computational model might violate the ECTT simply because of a bad choice of data representation. There is no relation of efficient simulation between an unary Turing machine and a Turing machine with a shorter data representation but it is yet empirically realizable. A reasonable computational model might be defined only up to a convenient choice of data representation. Other similar conditions, independent of empirical realizability, might be added to an improved understanding of what a reasonable model is. For the sake of caution, one might distinguish a special, empirical ECCT (*All empirically realizable computational models are simulable by a Turing machine with at most polynomial overhead)* from the original, general ECCT. The empirical ECCT is not an autonomous proposition but a simple subhypothesis of the general ECCT: if it turns out to be false, the general ECCT would be refuted. Even if the ECTT is not reduced to an empirical hypothesis, it would still depend on such an empirical hypothesis[5]. This being said, we can drop the distinction between the empirical ECTT and the general ECTT for the sake of conciseness.

We may then consider an empirical interpretation, or empirical view, of computational complexity, based on the following statement: *a valid definition of a reasonable computational model, and a valid definition of computational costs within a computational model, should take into account the empirical factors of empirical realizability.* The value of this proposition is essentially heuristic, since we do not have a definition of what a reasonable computational model is. It simply warns computer scientists, that they should take empirical factors into account, when they discuss the reasonableness of a given computational model.

This proposal opens a new vista on the subject-matter of CCT, since it lays at its foundations an empirical hypothesis, which might

[4] The most commonly used phrase is "physically realizable". But I prefer to avoid the common philosophical ambiguity associated with the adjective "physical," which refers both to any empirical entity or process, and to the entities or process that are explicitly modelized by physics. P. Shor clearly had the second acception in mind, since he writes [26] :

> Researchers have produced machine models that violate the above quantitative Church's thesis, but most of these have been ruled out by some reason for why they are not "physical;" that is, why they could not be built and made to work.

[5] Our inability to demonstrate the ECTT should not be attributed to a mere lack of definition, but to this very dependence on an empirical hypothesis, which cannot be proven.

be refuted by further empirical findings. The purely a priori nature of CCT as a discipline might thus be called into question. A proper account of computational complexity should take into consideration the empirical conditions of implementation. As P. Shor explicitly remarked, the success of our mathematical models of computation might have blinded us to this seemingly elementary fact, tricking us into thinking that computational complexity was a purely abstract, a priori topic.

With this interpretation in mind, it is easy to see the reason why the ECTT should not be considered an implicit definition, as P. van Emde Boas proposed [29]. If one thinks that the ECTT should be a mathematically well-defined proposition, then one can only be disturbed by the lack of definiteness of the notion of "reasonable model". At first sight, it would seem relevant to consider that statement as a implicit definition of the notion. But one would then be unable to use this concept to discuss the merits of original computational models, which is a quite unpleasant consequence.

An empirical view can help us out of this dead-end. The basic concepts of physics, for instance, are never defined with perfect mathematical rigor. Think for instance of the wave-packet reduction principle in quantum mechanics: when a measurement occurs, the wave function of the measured system collapses to an eigenstate of the measured observable. It does not tell us precisely what counts and what does not count as a measurement, and this has precisely been a topic of discussion in the interpretation of quantum mechanics. But it would nevertheless be foolish to think that it is an implicit definition of what a measurement is, and it does not keep physicists from believing that they would acknowledge a counter-example to that principle if they met one. In mathematics, you cannot speak of a counter-example to a proposition that is not rigorously defined. That is not the case in empirical sciences: physicists agree that they would acknowledge a counter-example to basic physical principles, even if a basic principle is never defined with perfect rigor, and there is always room for interpretation. In that sense, physical principles, if they cannot be demonstrated, can be falsified, and have to be construed as genuine propositions, not mere definitions.

In a similar fashion, we might not be able, and we should probably not try to reach mathematical rigor in the definition of what a reasonable computational model is. It should not keep us from discussing the issue, and it will not keep us from acknowledging a counter-example to the ECTT if we ever meet one. Thus, the empirical interpretation of the ECTT illuminates its present status, and the understanding that we should have of it. It is a empirical proposition, that cannot be demonstrated, but can both be falsified by a relevant counter-examples, and corroborated inductively by the rebutting of alleged counter-examples, just as our fundamental physical principles are corroborated by years of resistance to seemingly adversary experiments.

## 4 How empirical considerations are brought into CCT: unconventional computational models

Let us now see how one can argue in favor of this empirical interpretation. Before we come to the main argument, namely the current debate around quantum computing, let us put the problem into a larger perspective, by presenting quantum computing as one among many other unconventional computational models.

A fundamental distinction has to be drawn between two different questions: the relevance of empirical considerations in the definition of a computational model, and the success to implement this or that unconventional model. Many unconventional models are blatantly unreasonable, but the very reasons why they are deemed so make them relevant for the discussion of the empirical view of computation. If the arguments formulated to dismiss exceptional complexity-theoretic performances are empirical, then the empirical interpretation of complexity is supported by the following counterfactual argument: *if empirical facts of the matter have been different from what they actually are, then it would have been possible to achieve exceptional computational performances, such as the violation of the ECTT.* The truth-value of the empirical interpretation of complexity does not depend on the success or failure to implement some unconventional computational model, but on the specific reasons for this success or failure.

Depending on the computational model under scrutiny, the "empirical facts of the matter" mentioned in the antecedent can vary greatly in nature. In certain cases, such as quantum computing with non-linear variants of Schrödinger's equation [3], the computational model contradicts well-established principles of some physical theories. In other cases, such as computing with Closed Timelike Curves, a controversial aspect of some accepted theory, i.e. General Relativity, is put to use (see [10], for a critical appraisal). In still other cases, the computational model makes use of a formal possibility that would be deemed unrealistic by most physicists, such as arbitrarily precise control or measurement of a given system, even though for the time being no fully established principle explicitly forbids such a possibility (see [2] for a comprehensive review).

Among all unconventional computational models, quantum computing (abbreviated as QC) has drawn an exceptional amount of attention from the CCT and physics community. To this day, QC is the only computational model violating the ECTT that has been the object of intensive scientific scrutiny, with entire research groups working on both theoretical and experimental aspects of the problem.

What makes QC so special? In 1994, P. Shor published an efficient algorithm to factorize integers using a quantum computational model [25][6]. This result was striking mainly for two reasons. First, factorizing integers was, and still is believed to be hard on a Turing machine: it is thus likely that Shor's algorithm violates the ECTT. Second, quantum computing is not based on a blatantly unrealistic model, but on our best-confirmed physical theory, and it is not obvious to say that it misconstrues this theory.

For very small input sizes, prototypical quantum computers have already been implemented and shown to work. Nevertheless, a true demonstration of the extra computational power of quantum computers would require the use of large inputs to study how computational time scales with input size. This has not yet been achieved and would represent a considerable experimental challenge. Thus the possibility of a large-scale quantum computer is still a controversial topic.

From our philosophical perspective, one argument used by QC enthusiasts has special interest. The relevance of research in quantum computing has been defended on the basis of an "either way I win" argument. Trying to build a large scale quantum computer would be a topic of major scientific interest, because if we were successful, we would have shown that the ECTT could be violated, and if we were not, since quantum computing is compatible with quantum theory as we know it, we would have proven quantum theory wrong. The debates on the possibility of quantum computing exhibit the dependence of a complexity-theoretic question -does there exist a reasonable computational model violating the ECTT? - to non-trivial, open

[6] There exist other results indicating that a quantum computer might be more powerful than any conventional, "classical" model (see, for instance, [14]). But Shor's algorithm is the only known example that provides an exponential speed-up over conventional models.

questions in quantum theory, such as the possibility to protect large quantum systems against decoherence, the exact precision to which a quantum system can be controlled, the exact accuracy to which quantum-theoretic predictions hold (see, for instance, [19] and [1]). Even if a large-scale quantum computer turned out to be impossible, one could still formulate a particularly strong counterfactual argument in favor of the empirical interpretation: *if any, or several of the aforementioned facts had been different from what they actually are, then we would have violated the ECTT.* The particular strength of this argument would come from the necessity to refute the possibility of efficient quantum computing of gathering new information about the physical world, as opposed to just using already well-established physical principles, or exerting skepticism towards already criticized models.

The "either way I win" argument has of course been criticized (see for instance [18], section 2). In this paper, I do not intend to discuss this argument with all desirable precision and I will just make the following, heuristic remark. The "either way I win" argument is the proper place to discuss the empirical interpretation of complexity because it states that even if quantum computing fails to violate the ECTT, it will fail for interesting, empirical reasons.

# 5 Philosophical implications

## 5.1 CCT as a mathematical theory of computational problems: a critical appraisal

In the current state of affairs, there are significant, if not compelling, arguments in favor of the empirical interpretation of complexity. Leaving to further investigation the strengthening or correction of these arguments, the philosopher can wonder how this empirical interpretation, if it turned out to be vindicated, would change some of our fundamental intuitions about the subject matter of CCT, and the very nature of computation.

If one admits the validity of counterfactual arguments, the empirical interpretation of complexity can be considered as independent from the truth of the ECTT. Even if the ECTT is vindicated from all the different trials to which it has been subjected by unconventional computational models, it will have been established that CCT rests on a fundamental empirical postulate, and is therefore an empirical science.

This last remark calls into question our initial vision of CCT as a mathematical study of complexity properties of problems. If the empirical interpretation is vindicated, it would no longer be possible to demonstrate that a given problem, construed as an abstract mathematical entity, has certain computational complexity properties. As we have seen above, the ECTT, along with the existence of optimal algorithms, is necessary to attribute complexity properties to problems, and not just to algorithms. If the ECTT depends on a true empirical postulate, namely the empirical ECTT, then the attribution of complexity properties to a problem is not a mathematically provable result, demonstrated with purely a priori means, but an empirical result, which might be refuted by new empirical findings.

However, if the ECTT is false, then it is no longer possible to consider that the polynomial-superpolynomial time-complexity of a given problem as an intrinsic property of that problem. This is particularly obvious in the case of quantum computing. If a large-scale quantum computer is empirically realizable, then the same problem, FACTORING, will be polynomial on a quantum computer, and superpolynomial on a classical computer. Of course, it would still be possible to talk about the "complexity of the problem," to designate the complexity of the most efficient known algorithm solving that problem. But that property would be an extrinsic property of the problem, depending on the machine model at hand.

Complexity properties could still be attributed to algorithms, but a new vision of their relation to implementation would emerge, especially if a large-scale quantum computer is feasible. Again, this is particularly obvious in the case of quantum computation. Shor's algorithm can be simulated by a Turing machine, and thus by a classical computer. But this simulation would not preserve its complexity properties and create exponential slowdown. It is only on a true quantum computer that the desired complexity properties of this algorithm could be achieved. If complexity properties were still conceived as intrinsic properties of algorithms, this last remark would call for a reassessment of the distinction between hardware and software. An algorithm like Shor's algorithm is not "portable", in the sense that its complexity-theoretic advantages would be destroyed by classical implementations. But it is not so much dependent on the technological details of hardware than on the underlying physics of the hardware, as I will now demonstrate.

## 5.2 The unity of CCT as an empirical science

If CCT subject matter can no longer be conceived as complexity properties of problems, how can we conceive of it? As we have seen above, the ECTT guarantees that fundamental distinctions of CCT are robust. As we have previously seen, it also guarantees the pacific coexistence of two presentations of CCT: a pure mathematical discipline, and a study of the actual performances of real-world computers. If the ECTT is false, one would then be tempted to consider CCT as some branch of engineering, studying the properties of multifarious computational models, without any systematic unity. But this conclusion, I will argue, is not warranted by our previous analysis. To better understand how CCT maintains its unity as a scientific discipline, one has to take a closer look at the fundamental hypotheses of CCT.

First, I call "physics-dependence" the hypothesis that complexity-theoretic properties of a given computational model depend on underlying physical assumptions of this model. Physics-dependence should be distinguished from hardware-dependence. This distinction is more easily explained by considering quantum computers. A quantum algorithm depends on quantum properties of the system used for implementation, but it does not depend on the particular technological detail of that implementation. From a complexity-theoretic point of view, it does not really matter whether a quantum computer is made out of trapped ions or photons, to mention two actual implementation strategies. CCT is still independent from hardware details, even if it is no longer independent from the physics underlying that hardware.

Secondly, physics-dependence should also be distinguished from model-dependence. These last two decades have seen the birth of many different quantum computational models: quantum Turing machines, quantum circuits, quantum cellular automata, one-way quantum computer, topological quantum computation, quantum adiabatic computation, and several high-level models such as quantum lambda-calculi, quantum flow-charts, and categorical quantum languages. This list is of course open-ended, just as the list of "classical" computational models is open-ended. The study of the relations of efficient simulation between all these models raises exactly the same issues as before. Many of these models have been demonstrated to be polynomially equivalent (see, for instance, [31], [24]), but the question has been raised whether a given model, namely quantum adiabatic computation, could be more powerful than other quantum

models, allowing the computation of NP-complete problems in polynomial time [11]. This idea has been criticized along customary lines for not being empirically realizable [2].

Even if the hypothesis of model-independence in all generality is false, it becomes possible to formulate a milder form of the ECTT: *all realistic computational quantum models are intersimulable with at most polynomial overhead.* This new form is relativized to an underlying set of physical assumptions. What unconventional computational models suggest is not so much a dependence of complexity-theoretic properties on a particular model, but a dependence on the physical hypotheses underlying a class of models. Thus it is preferable to present the subject-matter of CCT as the study of the resources needed to solve some computational problem on a given class of computers, characterized by a common set of physical assumptions. If CCT had to be compared with any other branch of knowledge, it would be wiser to compare it with mathematical physics, studying the various computational potentials of different physical theories, than it would be to classify it as a branch of engineering. But it is not necessary to assimilate CCT to any existing discipline, and we can consider it as an empirical discipline on its own.

This view, however, remains problematic. For about two decades, physicists and computer scientists have tried to pinpoint what exact feature of quantum theory would cause the supposed exponential speed-up a quantum computer might provide and the question remains controversial (see, for instance, [8]). Consequently, it is hard to explain what "classical" means in "classical computational model." Worse still, models inspired by pen-and-paper computation, such as the Turing machine, do not make any explicit reference to classical physics. The input is not encoded in the state of a classical system, the dynamics of the system is not made explicit, and no mention is made of measurement. The model is classical only in the loose sense that it is not quantum: it does not make any use of specific quantum properties such as state superposition and entanglement. Even if one subscribes to the idea that CCT is physics-dependent, fleshing out this idea is a difficult task.

Nevertheless, if this view were vindicated, it would explain why CCT could maintain a systematic unity as a field, instead of just being a conjunction of computational models without any theoretical relation to one another. The different sets of physical assumptions made by different classes of computational models need not have an empty intersection. Even if the ECTT is false, some weaker physical hypothesis might still hold, and could be substituted to the ECTT at the foundations of CCT. This physical hypothesis would be respected by every class of computational model and thus unify CCT as a field.

To better explain this point, it is necessary to understand which part of CCT could be shattered by new computational models and which part could remain untouched. If a large-scale quantum computer is empirically realizable, the time-complexity of a given problem depends on the selected computational model. This would not imply that the distinction between polynomial and superpolynomial complexity classes should be abandoned, or even the distinction between P and NP for that matter. In the current state of art, FACTORING is conjectured to be in NPI, the classes of problems in NP that are neither in P nor NP-complete: the existence of an efficient solution to that problem does not imply that P = NP. Some of the models we have presented above make bigger promises, pretending to solve NP-complete problems in polynomial time, or even to solve efficiently all problems in a class larger than NP. As we have seen, these models are considered far less likely to succeed than the quantum computer, but the failure of an attempt is not the failure of an idea. Should we expect further developments in physics and unconventional computational models to yield such extraordinary results (see [22] for an interesting discussion of this point)? In recent years, quantum complexity theorist S. Aaronson has defended the exact opposite view [2]. Not only we should not expect such a complexity-theoretic wonder, but we could even state its impossibility as a physical principle, the "No SuperSearch Principle" (aka "NP-hardness Assumption"): *there is no physical means to solve NP-complete problems in polynomial time.*

S. Aaronson's main argument for this principle is the following: if an efficient solution of an NP-complete problem were possible, it would trivialize many problems we strongly conjecture to be hard. For any usual axiomatic system $A$, the following problem is in NP: $THEOREMS = \{(\phi, 1^n) : \phi \; has \; a \; formal \; proof \; of \; length \leq n \; in \; system \; A\}$. Even if no algorithm can decide every mathematical conjecture, there is a simple search algorithm that decides whether there exists a proof of length inferior to a given bound. If there were, thanks to some unconventional computational model, an efficient algorithm for this problem, it would become possible to "solve practically" any mathematical conjecture, by examining only proofs of reasonable length, e.g. a length inferior to the number of particles in the Universe. We would then have access to a form of automated mathematical research, despite the negative solution to the Enstcheidungsproblem. The P-NP problem can thus be seen as a bounded ressources version of the Entscheidungsproblem. Such implausible computational wonder should be explicitly excluded by a new physical principle, the No SuperSearch Principle, which would perform the function of the unifying hypothesis we mentioned above.

The previous argument might be questionable. It only justifies the proposed principle in terms of apocalyptic consequences for our understanding of mathematics and does not provide a proper physical ground for its acceptance. It has nevertheless a great heuristic value, for it demonstrates how much is at stake in the interpretation of CCT.

Without further developing S. Aaronson's arguments for that new principle, I would like to comment on the insight this proposition gives us on the new relations between physics and CCT allowed by the empirical interpretation of complexity. Once we accept this interpretation, it opens a two-way street between theoretical physics and complexity theory. We should not only expect that new insights from physics might shatter complexity-theoretic conjectures, established results and concepts, but we should also expect new insights on physics coming from CCT. Some CCT concepts, like the distinction between polynomial and superpolynomial complexity classes, might have a deep and physically robust meaning. Computation should not only be considered as a tool used by physics, but also as a subject matter of physics, whose properties are to be defined and studied from a physical point of view (for a similar position, see [15]).

## 6 Conclusion

The aim of this brief work was less to defend a thesis than to raise an issue. A comprehensive discussion of the arguments *pro* and *contra* the empirical interpretation would demand another article. I will be content if I have convinced the reader that the empirical interpretation of CCT is a legitimate problem.

I have also given indications on how further discussion of this interpretation should proceed. From this perspective, I tried to show that there is much at stake in the current scientific debate on the feasibility of quantum computing. The problem is not only relevant for physicists, computer scientists, and amateurs of high-end technol-

ogy, but for philosophers as well. Philosophers do not even need to wait for the definite resolution of this scientific controversy to show interest. If the "either way I win" argument is correct, complexity theory is physics-dependent, even if a large-scale quantum computer finally turns out to be unfeasible. This argument, which might be a topic of interest for philosophers studying counterfactual propositions, should be placed at the core of further discussion of the empirical view.

This view, if it were vindicated, would call for a radical reappraisal of the epistemological status of computational properties, which I have only been able to sketch out. These properties would no longer be the object of pure a priori knowledge, but should be seen as theoretical properties of the empirical processes that are harnessed to perform computation. This should be of interest not only for the philosophy of physics or computer science, but also for the general philosophy of knowledge, especially the debates concerning a priori knowledge.

## ACKNOWLEDGEMENTS

I would like to thank my Phd advisors, A. Grinbaum and J.B. Joinet, and the anonymous reviewers, for their comments of earlier drafts. I would also like to warmly thank Evan Spritzer, for the time he benevolently took to improve the redaction.

## References

[1] Scott Aaronson, 'Multinear Formulas and Skepticism of Quantum Computing', in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pp. 118–127, Chicago, IL, USA, (2004). Laszlo Babai.

[2] Scott Aaronson, 'NP-complete Problems and Physical Reality', *SIGACT News*, (March 2005). quant-ph/0502072.

[3] D.S. Abrams and Seth Lloyd, 'Nonlinear quantum mechanics implies polynomial-time solution for NP-complete and #P problems.', *Physical Review Letters*, **81**, 3992–3995, (1998). quantum-ph/9801041.

[4] Dorit Aharonov and Umesh Vazirani, 'Is Quantum Mechanics Falsifiable? A computational perspective on the foundations of Quantum Mechanics', *arXiv preprint arXiv:1206.3686*, (2012).

[5] Sanjeev Arora and Barak Boaz, *Computational Complexity. A Modern Approach*, Cambridge University Press, 2009.

[6] P. Arrighi and G. Dowek, 'The physical Church-Turing thesis and the principles of quantum theory', *Arxiv preprint arXiv:1102.1612*, (2011).

[7] Ethan Bernstein and Umesh Vazirani, 'Quantum complexity theory', *SIAM Journal on Computing*, **26**(5), 1411–1473, (1997).

[8] J. Bub, 'Quantum computation: Where does the speed-up come from', *Philosophy of Quantum Information and Entanglement. Cambridge University Press, Cambridge*, 231–246, (2010).

[9] Martin Davis, 'The Myth of Hypercomputation', in *Alan Turing: Life and Legacy of a Great Thinker*, Springer, (2004).

[10] J. Earman and J. D. Norton, 'Forever is a day: Supertasks in Pitowsky and Malament-Hogarth spacetimes', *Philosophy of Science*, 22–42, (1993).

[11] E. Fahri, J. Goldstone, and S. Gutmann, 'A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem', *Science*, **292**, 472–476, (2001). quant-ph/0104129.

[12] R. Gandy, 'Church's thesis and Principles for Mechanisms', *Studies in Logic and the Foundations of Mathematics*, **101**, 123–148, (1980).

[13] Oded Goldreich, *Computational Complexity*, Cambridge University Press, 2008.

[14] Lov K. Grover, 'Quantum Mechanics Helps in Searching for a Needle in a Haystack', *Physical Review Letters*, **79**(2), 325–328, (July 1997).

[15] A. Hagar and A. Korolev, 'Quantum Hypercomputation-Hype or Computation?', *Philosophy of Science*, **74**(3), 347–363, (2007).

[16] Amit Hagar, 'Quantum Algorithms: Philosophical Lessons', *Minds and Machines*, **17**(2), 233–247, (2007).

[17] Amit Hagar, *The Complexity of Noise: A Philosophical Outlook on Quantum Error Correction*, Morgan & Claypool Publishers, January 2011.

[18] Leonid Levin, 'The Tale of One-Way Functions', *Problems of Information Transmission.*, **39**(1), 92–103, (2003).

[19] Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information*, Cambridge series on information and the natural sciences, Cambridge University Press, 2000.

[20] Christos H. Papadimitriou, *Computational Complexity*, Addison Wesley Longman, 1995.

[21] Gualtiero Piccinini, 'The Physical Church-Turing Thesis: Modest or Bold?', *The British Journal for the Philosophy of Science*, **62**(4), 733–769, (2011).

[22] Itamar Pitowsky, 'The physical Church thesis and physical computational complexity', *Iyyun*, **39**, 81–99, (1990).

[23] Itamar Pitowsky, 'Quantum Speed-Up of Computations', *Proceedings of the Philosophy of Science Association*, **2002**(3), 168–177, (2002).

[24] Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel, 'Measurement-based quantum computation on cluster states', *Physical Review A*, **68**(2), 022312, (August 2003).

[25] Peter W. Shor, 'Algorithms for Quantum Computation: Discrete Logarithms and Factoring', in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134. IEEE Computer Society Press, (1994).

[26] Peter W. Shor, 'Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer', *SIAM J. Computing*, 1484–1509, (1997).

[27] W. Sieg, 'Church without dogma: Axioms for computability', *New computational paradigms*, 139–152, (2008).

[28] A. Turing, 'On Computable Numbers, with an Application to the Entscheidungsproblem', *Proceedings of the London Mathematical Society*, **42**, 230, (1936).

[29] Peter van Emde Boas, 'Machines Models and Simulation', in *Handbook of theoretical computer science*, volume A. Algorithms and complexity, The MIT Press, (1990).

[30] Anastasios Vergis, Kenneth Steiglitz, and Bradley Dickinson, 'The complexity of analog computation', *Mathematics and computers in simulation*, **28**(2), 91–113, (1986).

[31] Andrew Chi Chi Yao, 'Quantum Circuit Complexity', in *Proceedings of the Thirty-Fourth IEEE Symposium on Foundations of Computer Science*, pp. 352–361, Palo Alto, California, (1993).

# The Development of Models of Computation with Advances in Technology and Natural Sciences

**Gordana Dodig-Crnkovic** [1]

**Abstract.** The development of models of computation induces the development of technology and natural sciences and vice versa. Current state of the art of technology and sciences, especially networks of concurrent processes such as Internet or biological and sociological systems, calls for new computational models. It is necessary to extend classical Turing machine model towards physical/ natural computation. Important aspects are openness and interactivity of computational systems, as well as concurrency of computational processes. The development proceeds in two directions – as a search for new mathematical structures beyond algorithms as well as a search for different modes of physical computation that are not equivalent to actions of human executing an algorithm, but appear in physical systems in which concurrent interactive information processing takes place. The article presents the framework of info-computationalism as applied on computing nature, where nature is an informational structure and its dynamics (information processing) is understood as computation. In natural computing, new developments in both understanding of natural systems and in their computational modelling are needed, and those two converge and enhance each other.

## 1 INTRODUCTION: WHAT IS COMPUTING?

*"The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer."*

Turing in [1] p.436

Turing pioneered the development of first digital computers, based on his Logical Calculating Machine (Turing's name for Turing machine) simulating a human strictly following an algorithm. But he also devised two other fundamentally different theoretical models of computation: neural networks and morphological computing. In the background for all three models we can discern his computational natural philosophy. According to Hodges [2], Turing was a natural philosopher, and nature – from patterns on the animal skin to functioning of human brains - was for him possible to understand in computational terms. Turing lived in a time when computing machinery still was in its beginnings, and there was characteristic dominance of theory over practical devices.

Today on the contrary, it appears that the existing computing machinery developed faster than the corresponding theory of computation. The consequence is that for different directions of the development of computing systems different models of computation apply ranging from classical Turing Machine theories of [3] to steps beyond in [4][5][6] to interactive computing of [7], and natural computing in different variations [8][9][10][11] to the view that computing is a natural science [12][13].

The existing diversity of ideas about computing can be confusing. However, the lack of consensus about the nature of computation is not unique and it has the parallel in the current lack of consensus about the nature of information. Those two are related questions and both have two parts:

a) What is it in the world that corresponds to information/ computation?
b) How do we model that information/ computation [once we agree upon what in the world they correspond to]?

The answer to the above is not simple, as concepts are theory-laden and we use our existing theories in order to formulate new ones, going via phenomena in the real world that we identify as information/ computation.

We can compare this situation with the development of other basic scientific concepts. Ideas about matter, energy, space and time have their history. The same is true of the idea of number in mathematics or the idea of life in biology. So, we should not be surprised to notice the development in the theory of computation that goes along with the development of mathematical methods, new computational devices and new domains of the real world that can be modelled computationally.

## 2 HYSTORY OF COMPUTATION UP TO ELECTRONIC COMPUTERS

The oldest computational devices were analog. The earliest calculating tools that humans used were fingers (Latin "digit") and pebbles (Latin "calculus") that can be considered as simple means of extended human cognition [14]. Tally stick, counting rods and abacus were the first steps towards mechanization of calculation. The ancient Greek astronomical analog calculator, Antikythera mechanism, from the second century BC, calculated the motions of stars and planets. [15] Among the first known constructors of mechanical calculators was Leonardo da Vinci. Pascal invented mechanical calculator that could add and subtract two numbers directly, and multiply and divide by repetition, improved by Leibniz who added direct multiplication and division.

Traditionally, *computation* was understood as synonymous with *calculation*. The first recorded use of the word "computer" was in 1613 to denote a *person* who carried out calculations, and the word retained the same meaning until the middle of the 20th century, when the word "computer" started to assume its current meaning, describing a *machine* that performs computations.

[1] School of Innovation, Design and Engineering, Mälardalen University, Sweden. Email: gordana.dodig-crnkovic@mdh.se

Babbage was the first to design a *programmable mechanical computer*, the general purpose Analytical Engine. The first *electronic digital computer* was built in 1939 by Atanasoff and Berry and it marks the beginning of the *era of digital computing*. In 1941 Zuse designed the first *programmable computer* Z3, also the first one based on the binary system. UNIVAC was the first computer capable of running a program from memory. The first *minicomputer* PDP was built in 1960 by DEC. Since 1960s the extremely fast growth of computer use was based on the technology of *integrated circuit/ microchi*p, which triggered the invention of the microprocessor, by Intel in 1971. [16]

The progress of computing of course depends both on the development of hardware and the corresponding development of software. This includes algorithms, programming languages, compilers and interpreters, operating systems, virtual machines, and so on. Yet a lot of software development was considered as advanced applications of Turing Machine model. Computability Theory is still based on Turing Machine.

## 3 BEYOND CONVENTIONAL COMPUTING MACHINERY: NATURAL COMPUTING

The development of computing, both machinery with programs and its models, continues. We are accustomed to rapid increase of computational power, memory and usability of computers, but the limit of miniaturization within the present-day concept of computing is approaching as we are getting close to quantum dimensions of hardware. One of the ideals of computing ever since the time of Turing is intelligent computing, which would imply machine capable of not only executing mechanical procedure, but even intelligent problem solving. Thus the goal is a computer able to simulate behaviour of human *mathematician*, able of making an intelligent insight. A development of *cognitive computing* aimed towards human-level abilities to process/organize/understand information is presented in [17].

At the same time *computational modelling of human brain* in The Human Brain Project [18] has for a goal to reveal the exact mechanisms of human brain function that will help us understand both how humans actually perform symbol processing when they follow an algorithm, and also how humans create algorithms or models. Those new developments in computational modelling of brain can be seen as a part of the research within the field of *natural computing*, where natural system performing computation is human brain.

However, natural computing has much broader scope. According to the Handbook of Natural Computing [11] natural computing is “the field of research that investigates both human-designed computing inspired by nature and computing taking place in nature.” It includes among others areas of cellular automata and neural computation, evolutionary computation, molecular computation, quantum computation, nature-inspired algorithms and alternative models of computation.

An important characteristic of the research in natural computing is that knowledge is generated bi-directionally, through the interaction between computer science and natural sciences. While natural sciences are adopting tools, methodologies and ideas of information processing, computer science is broadening the notion of computation, recognizing information processing found in nature as computation. [19][8][9][20] That led Denning [12] to argue that computing today is a natural science. Natural computation provides a basis for a unified understanding of phenomena of embodied cognition, intelligence and knowledge generation. [21][22]

The idea of computing nature has important consequences for our view of computation as information processing that generalizes the idea of algorithm. Computation found in nature is understood as a physical process, where nature computes with physical bodies as objects. Physical laws govern processes of computation, which necessarily appears on many different levels of organization of physical systems.

Natural computation can be modelled as information processing based on the exchange of information in a network of agents. An agent is defined as an entity capable of acting in the world on its own behalf.

One sort of computation is found on the quantum-mechanical level where agents are elementary particles, and messages (information carriers) are exchanged by force carriers, another type of computation is on the other levels of organization. In biology, computational processes (information processing) are going on in cells, tissues, organs, organisms, and eco-systems, with corresponding agents and message types passed. In biological computing or social computing the message carriers are complex chunks of information such as molecules, or sentences and the computational nodes (agents) can be molecules, cells, organisms or groups. [23]

## 4 COMPUTATION IN CLOSED VS. OPEN SYSTEMS

As we have seen in Section 2, computational machinery evolved historically from simplest tools of extended human cognition to mechanical computers (calculators) to electronic machines with vacuum tubes and then transistors, to integrated circuits and eventually to microprocessors. During this development of hardware technologies towards ever smaller, faster and cheaper devices, the computational principles remained similar: an isolated computing machine calculating a function, executing an algorithm that can be represented by the Turing machine model.

However, since the 1950s computational machinery has been increasingly used to exchange information and computers gradually started to connect in networks and communicate. In the 1970s computers were connected via telecommunications. The emergence of networking involved a rethinking of the nature of computation and boundaries of a computer. Computer operating systems and applications were modified to access the resources of other computers in the network. In 1991 CERN created the World Wide Web, which resulted in computer networking becoming a part of everyday life for common people. By the end of 2011 an estimated 35% of Earth's population used the Internet, according to Wikipedia article Global Internet usage.

With the development of computer networks, two characteristics of computing systems have become increasingly important: *parallelism/concurrency* and *openness* – both based on communication between computational units.

Comparing new open-system with traditional closed-system computation models, Hewitt [24] characterizes the Turing machine model as an *internal* (*individual*) framework and his own Actor model of concurrent computation as an *external* (*sociological*) model of computing.

In order to provide mathematical framework for open-system modelling, Burgin and Dodig-Crnkovic analyze methodological and philosophical implications of algorithmic aspects of unconventional/natural computation that extends the closed classical universe of computation of the Turing machine type. [25] The new model constitutes an open world of algorithmic constellations, allowing increased flexibility and expressive power, supporting constructivism and creativity in mathematical modelling and enabling richer understanding of computation. The greater power of new types of algorithms also results in the greater complexity of the algorithmic universe, transforming it into the *algorithmic multiverse*. New tools are brought forth by local mathematics, local logics and logical varieties.

## 5 COMPUTATION AS INTERACTION AND INTERACTIVE COMPUTING

As we have seen in the previous sections, interaction between computational units and processes has become one of the central issues in computing. In 1998 Wegner developed the interactive model of computation [26] which involves interaction, or communication, with the environment during computation, unlikely the traditional Turing machine model of computation which goes on in an isolated system. The interactive paradigm includes concurrent and reactive computations, agent-oriented, distributed and component-based computations, [27]. Interestingly, Bohan Broderick [28] argues based on the study of technical notions of communication and computation and finds them practically indistinguishable. "The two notions may be kept distinct if computation is limited to actions within a system and communications is an interaction between a system and its environment." – Bohan Broderick ascertains.

Goldin and Wegner [27] show, that the paradigm shift from algorithms to interactive computation follows the technology shift from mainframes to networks, and intelligent systems, from calculating to communicating, distributed and often even mobile devices. A majority of the computers today are embedded in other systems and they are continuously communicating with each other and with the environment. The communicative role has definitely prevailed over the initial role of a computer as an isolated calculating machine.

The following characteristics distinguish this new, interactive notion of computation [7]:

- Computational problem is defined as *performing a task*, [in a dynamical environment – my addition] rather than (algorithmically) producing an answer to a question.
- Dynamic input and output are modelled by *dynamic streams which are interleaved*; later values of the input stream may depend on earlier values in the output stream and vice versa.
- The *environment of the computation is a part of the model,* playing an active role in the computation by dynamically supplying the computational system with the inputs, and consuming the output values from the system.
- *Concurrency*: the computing system (agent) computes in parallel with its environment, and with other agents. (Agents can consist of agents networks, recursively.)
- *Effective non-computability*: the environment cannot be assumed to be static or effectively computable. We cannot always pre-compute input values or predict the effect of the system's output on the environment.

## 6 CONCURRENCY

Even though practical implementations of interactive computing such as Internet are decades old, a general foundational theory, and the semantics and logic of interactive computing is still missing. A theoretical foundation analogous to what Turing machines are for algorithmic computing, is under development. [26][12][29][24] One important aspect of interactive computing is concurrency. In concurrent systems multiple agents (processes) interact with each other. In biology, where systems are typically concurrent, the following models of concurrent computation are used: Petri nets, Process calculi, Interacting state machines, Boolean networks (especially for gene regulatory networks).

The advantages of concurrency theory that is used to simulate observable natural phenomena are according to [30] that:

"*it is possible to express much richer notions of time and space in the concurrent interactive framework than in a sequential one. In the case of time, for example, instead of a unique total order, we now have interplay between many partial orders of events--the local times of concurrent agents--with potential synchronizations, and the possibility to add global constraints on the set of possible scheduling. This requires a much more complex algebraic structure of representation if one wants to "situate" a given agent in time, i.e., relatively to the occurrence of events originated by herself or by other agents.*"

Theories of concurrency are partially integrating the observer into the model by allowing certain shifting of the inside-outside system boundary. According to Abramsky [29]:

*"An important quality of Petri's conception of concurrency, as compared with "linguistic" approaches such as process calculi, is that it seeks to explain fundamental concepts: causality, concurrency, process, etc. in a syntax-independent, "geometric" fashion. Another important point, which may originally have seemed merely eccentric, but now looks rather ahead of its time, is the extent to which Petri's thinking was explicitly influenced by physics (…).*

*To a large extent, and by design, Net Theory can be seen as a kind of discrete physics: lines are time-like causal flows, cuts are space-like regions, process unfoldings of a marked net are like the solution trajectories of a differential equation. This acquires new significance today, when the consequences of the idea that "Information is Physical" [17] are being explored in the rapidly developing field of quantum informatics."*

If the current programme for computation is formulated as aiming at reconstruction of the computational capabilities of human, then it seems unavoidable to further develop new models of computation, especially interactive computing and natural computing. Living systems are essentially open and in constant communication with the environment. New computational models must include interactive, embodied, concurrent computation processes in order to be applicable not only to physics but also to biological and social phenomena.

As Sloman shows, concurrent and synchronized machines are equivalent to sequential machines, but some concurrent machines are asynchronous, and thus not equivalent to Turing machines. [37] If a machine is composed of asynchronous concurrently running subsystems, and their relative frequencies vary randomly, then such a machine cannot be adequately modelled by Turing machine.

Turing machines are discrete but can in principle approximate machines with continuous changes, but cannot implement them

exactly. Continuous systems with non-linear feedback loops may be chaotic and impossible to approximate discretely, even over short time scales, see [37] and [24].

Theoretical model of concurrent (interactive) computing that would be the counterpart of Turing machine model of algorithmic computing is under development. (Abramsky, Hewitt, Wegner) From the experience with present day networked concurrent computation it becomes obvious that Turing machine model can be seen as a proper subset of a more general interactive, embodied, concurrent computation.

## 7 DIGITAL VS. ANALOG, DISCRETE VS. CONTINUOUS AND SYMBOLIC VS. SUB-SYMBOLIC COMPUTATION

Among many discussions concerning concepts of computation, a prominent place is given to the controversy about the continuous/discrete vs. analogue/digital computation. [31] Some believe in the ultimately discrete nature of physical reality and deny any true continuum. Some believe that human cognition can be understood in terms of language and symbol manipulation. Understanding of nature of symbols has relevance for understanding of human cognition and information processing going on in human body (including brain and nervous system).

Trenholme [32] describes the relationship of analog vs. symbolic simulation:

*"Symbolic simulation is thus a two-stage affair: first the mapping of inference structure of the theory onto hardware states which defines symbolic computation; second, the mapping of inference structure of the theory onto hardware states which (under appropriate conditions) qualifies the processing as a symbolic simulation.*

*Analog simulation, in contrast, is defined by a single mapping from causal relations among elements of the simulation to causal relations among elements of the simulated phenomenon."* [32]

Both symbolic and sub-symbolic simulations depend on causal/analog/physical and symbolic type of computation on some level of abstraction but in the case of symbolic computation it is the *symbolic level where information processing is observed*. Similarly, even though in the sub-symbolic model symbolic representation exists at some high level of abstraction (because language is used for its description), it is *the physical agency and its causal structure that define computation*.

Freeman characterizes accurately the relationship between physical/sub-symbolic and logical/symbolic level in the following:

*"Human brains intentionally direct the body to make symbols, and they use the symbols to represent internal states. The symbols are outside the brain. Inside the brains, the construction is effected by spatiotemporal patterns of neural activity that are operators, not symbols. The operations include formation of sequences of neural activity patterns that we observe by their electrical signs. The process is by neurodynamics, not by logical rule-driven symbol manipulation. The aim of simulating human natural computing should be to simulate the operators. In its simplest form natural computing serves for communication of meaning. Neural operators implement non-symbolic communication of internal states by all mammals, including humans, through intentional actions. (…) I propose that symbol-making operators evolved from neural mechanisms of intentional action by modification of non-symbolic operators."* [33]

Consequently, our brains use non-symbolic computing internally in order to manipulate relevant external symbols/objects.

In the words of MacLennan [34], who emphasizes the importance of continuous computation for natural systems:

*"We propose certain non-Turing models of computation, but our intent is not to advocate models that surpass the power of Turing Machines (TMs), but to defend the need for models with orthogonal notions of power. We review the nature of models and argue that they are relative to a domain of application and are ill-suited to use outside that domain. Hence we review the presuppositions and context of the TM model and show that it is unsuited to natural computation (computation occurring in or inspired by nature). Therefore we must consider an expanded definition of computation that includes alternative (especially analog) models as well as the TM."*

## 8 THE UNREASONABLE INEFFECTIVENESS OF MATHEMATICS IN BIOLOGY AND BIAS OF MATHEMATICIANS

Mathematician's contribution to the development of the idea of computing nature is central. Turing was mathematician and an early proponent of natural computing who put forward two computational models of physical processes – morphological computing and neural networks.

In the context of computing nature, living systems are particularly interesting because of their complexity of informational processing, but up to now science haven't been able to adequately model and simulate the behaviour of even the simplest living organisms. "The unreasonable effectiveness of mathematics" observed in physics by Wigner [35] is missing in biology, according to Gelfand as quoted by Chaitin, see [36].

Not many people today would claim that human cognition (information processing going on in our body, including our brains) can be adequately modelled as a result of computation of one Turing machine, however complex function it might compute. In the next attempt, one may imagine a complex architecture of Turing machines running in parallel as communicating sequential processes exchanging information. We know today that such a system of Turing machines cannot produce the most general kind of computation, as truly asynchronous concurrent information processing going on in our brains. [37]

On the other hand, one may object that IBM's Watson, the winner in man vs. machine "Jeopardy!" challenge, runs on contemporary supercomputer which is claimed to be implementation of the Turing machine. Yet, Watson is connected to the Internet, and Internet is not a Turing machine. It is not even a network of Turing machines. Information processing going on throughout the Internet includes signalling and communication based on complex concurrent physical processes that cannot be sequentialized. [24][37] As an illustration see [38] on parasitic computing that implements computation on the communication infrastructure of the Internet. Real world computation is physical.

Cooper in his article Turing's Titanic Machine? [39] diagnoses the limitations of the Turing machine model and identifies the following ways for overcoming those limitations:

- Embodiment invalidating the `machine as data' and universality paradigm.
- The organic linking of mechanics and emergent outcomes delivering a clearer model of supervenience of mentality on brain functionality, and a reconciliation of different levels of effectivity.
- A reaffirmation of experiment and evolving hardware, for both AI and extended computing generally.
- The validating of a route to creation of new information through interaction and emergence.

Related article by the same author, The Mathematician's Bias and the Return to Embodied Computation, elucidates the differences of physical computation compared to universal symbol manipulation. [40]

From all above it is clear that Turing machine model of computation is an abstraction and idealization. In general, the trend in computing can be discerned towards extension to more and more physics-inspired instead of idealized, symbol-manipulating models, which are its subset.

## 9 LOGIC OF COMPUTING AND PARA-CONSISTENCY

Besides physical embodiment, one of the important aspects of computing is logic. The underlying logic of Turing's Logical Calculating Machine is fully consistent standard logic. Hintikka proposes Logic as a Theory of Computability, still within the same classical framework. [41]

Turing machine is assumed always to be in a well defined state. [24] In contemporary computing machinery, however, we face both states that are not well defined (in the process of transition) and states that contain inconsistency:

*"Consider a computer which stores a large amount of information. While the computer stores the information, it is also used to operate on it, and, crucially, to infer from it. Now it is quite common for the computer to contain inconsistent information, because of mistakes by the data entry operators or because of multiple sourcing. This is certainly a problem for database operations with theorem-provers, and so has drawn much attention from computer scientists. Techniques for removing inconsistent information have been investigated. Yet all have limited applicability, and, in any case, are not guaranteed to produce consistency. (There is no algorithm for logical falsehood.) Hence, even if steps are taken to get rid of contradictions when they are found, an underlying paraconsistent logic is desirable if hidden contradictions are not to generate spurious answers to queries."* [42]

Open, interactive and asynchronous systems have special requirements on logic. Goldin and Wegner [27] and Hewitt [24] argue e.g. that computational logic must be able to model interactive computation, and that classical logic must be robust towards inconsistencies i.e. must be paraconsistent due to the incompleteness of interaction.

## 10 INFORMATION/ COMPUTATION AND MATTER/ENERGY

As pointed out in the introduction, not only the idea of computation is under dynamic development, but similar is true of the concept of information. Both processes can be seen as a result of current rapid development of information technology/ computing machinery and our newly acquired insights in sciences, largely based on the development of information and communication technology.

Even though we are far from having a consensus on the concept of information, the most general view is that information is a structure consisting of data. Floridi [43] has the following definition of datum: "In its simplest form, a datum can be reduced to just a lack of uniformity, that is, a binary difference." Bateson's "the difference that makes the difference" [44] is a datum in that sense. Information is both the result of observed differences (differentiation of data) and the result of synthesis of those data into a common informational structure (integration of data), as argued by Schroeder in [47]. In the process of knowledge generation an intelligent agent moves between those two processes – differentiation and integration of data. It is central to keep in mind that for something to be information *there must exist an agent* from whose perspective this structure is established. Thus information is a network of data points related from an agent's perspective.

There is a distinction between the world as it exists autonomously, independent from any agent, Kantian "ding an sich", (thing in itself, nuomenon) and the world for an agent, things as they appear through interactions (phenomena).

Informational realists (like Floridi, Sayre, Vedral) take the reality/world/universe to be information. In [23] I added by analogy "information an sich" representative of the "ding an sich" as a *potential information for an agent.*

When does this potential information become actual information for an agent?

The world in itself is (proto)information that gets actual through interactions with agents and huge parts of the universe are potential information for different kinds of agents – from elementary particles, to molecules, etc. and all the way up to humans and societies.

Living organisms as complex agents inherit bodily structures (which ultimately are informational structures) as a result of a long evolutionary development of species. Those structures are embodied memory of the evolutionary past. They present the means for agents to interact with the world, get new memories, learn new patterns of behaviour and construct knowledge. World via Hebian learning forms a human's (or an animal's) informational structures.

If we say that for something to be information there must exist an agent from whose perspective this structure is established, and we argue that the fabric of the world is informational, the question can be asked: *who/what is the agent*? An agent (an entity capable of acting on its own behalf in the world) can be seen as interacting with the points of inhomogeneities (data), establishing the connections between those data and the data that constitute the agent itself (a particle, a system). There are myriads of agents for whom information of the world makes differences (Bateson's "difference that makes the difference") – from elementary particles to molecules, cells, organisms, societies… - all of them interact and exchange

information on different levels of scale and this information dynamics is natural computation. When I interact via computer, photons from the screen reach my retina, and agents are both photons and the cells that photon hits and interacts with but also all the other parts of the system that transfer and process information from my eye to my brain and back to the motor control that controls my fingers that type on the keyboard. I can also see myself as an agent and my agency in this case is different from the agency of the cells on my retina. In short, this is an agent-based (or actor-based) view of natural computation. The change in the physical world happens through data self-organization in an agent.

Information processes are governed by laws of physics and physicists are already working on reformulating physics in terms of information. This development can be related to the Wheeler's idea "it from bit". [45] For more details on current research, see the special issue of the journal Information dedicated to matter/energy and information [46], with articles by Vedral, Goyal, Brenner, Matsuno and Salthe, Fields, Fiorillo, Yoshitake and Saruwatari, Luhn and Zenil. Furthermore, a recent special issue of the journal Entropy addresses natural/unconventional computing [47] with articles by Chiribella, D'Ariano and Perinotti, Stepney, Ehresmann, Dodig Crnkovic and Burgin, Zenil, Gershenson, Marshall and Rosenblueth. All contributions explore the space of natural computation and relationships between the physical (matter/energy), information and computation.

## 11 INFO-COMPUTATIONALISM

As a result of a synthesis of the idea of *computing nature* (naturalist computationalism/ pancomputationlism) [22][48][49] [50][51] with the *informational structural realism* [43][52] (the view that nature represents a complex informational structure for a cognizing agent), the framework of *info-computationalism* is construed [21]. Within info-computationalism the time development (dynamics) of physical states in nature is understood as information processing. Such processes include self-organization processes, self-assembly, developmental processes, gene regulation networks, gene assembly, protein-protein interaction networks, biological transport networks, and similar processes found in nature. The majority of info-computational processes are sub-symbolic and some are symbolic (in case of agents capable of symbol manipulation).

Within info-computational framework, computation on a given level of organization presents a realization/actualization of the laws that govern interactions between constituent parts. Computation comes with built-in causation. What happens in every next layer of organization of matter is that a set of rules governing the system switch to the new emergent regime. It remains yet to be revealed how this process exactly goes on in nature, how emergent properties occur. With help of natural computing we may hope to uncover those mechanisms.

In words of Rozenberg and Kari: "(O)ur task is nothing less than to discover a new, broader, notion of computation, and to understand the world around us *in terms of information processing*." [19] From the research in complex dynamical systems, biology, neuroscience, cognitive science, networks, concurrency and more, new insights essential for the info-computational universe may be expected in the years to come.

## 12 MORPHOLOGICAL COMPUTING. MEANING GENERATION FROM RAW DATA TO SEMANTIC INFORMATION

In 1952 Turing wrote a paper on morphogenesis proposing a chemical model as the explanation of the development of biological patterns such as the spots and stripes on animal skin. [53] Turing did not claim that physical system producing patterns actually performed computation. Nevertheless, from the perspective of info-computationalism we can argue that morphogenesis is a process of morphological computing. Physical process – though not computational in the traditional sense, presents natural (unconventional), morphological computation. Essential element in this process is the interplay between the informational structure and the computational process - information self-structuring and information integration, both synchronic and diachronic, going on in different time and space scales in physical bodies.

Informational structure presents a program that governs computational process [23], which in its turn changes that original informational structure obeying/implementing/realizing physical laws.

*Morphology* is the central idea in understanding of the connection between computation (morphological/ morphogenetical) and information. What is observed as materials on one level of analysis, represents morphology on the lower level, recursively. So water as material presents arrangements of [molecular [atomic [elementary particle [] ]]] structures.

Info-computational naturalism describes nature as informational structure – a succession of levels of organization of information. Morphological computing on that informational structure leads to new informational structures via processes of self-organization of information. Evolution itself is a process of morphological computation on a long-term scale. It will be instructive within the info-computational framework to study processes of self organization of information in an agent (as well as in population of agents) able to re-structure themselves through interactions with the environment as a result of morphological (morphogenetic) computation.

Cognition can be seen as a result of processes of morphological computation on informational structures of a cognitive agent in the interaction with the physical world, with processes going on at both sub-symbolic and symbolic levels. This morphological computation establishes connections between an agent's body, its nervous (control) system and its environment. Through the embodied interaction with the informational structures of the environment, via sensory-motor coordination, information structures are induced in the sensory data of a cognitive agent, thus establishing perception, categorization and learning.

Essential element in this process is the interplay between the informational structures and the computational processes - information self-structuring and information integration, both synchronic and diachronic, going on in different time and space scales. [22][44][45]

From the simplest cognizing agents such as bacteria to the complex biological organisms with nervous systems and brains, the basic informational structures undergo transformations through morphological computation. Here an explanation is in order regarding cognition which is defined in general way of

Maturana and Varela who take it to be synonymous with life. [54][55]. All living organisms possess some degree of cognition and for the simplest ones like bacteria cognition consists in metabolism and (my addition) locomotion. [21] This process of interaction with the environment causes changes in the informational structures that correspond to the body of an agent, and its control mechanisms, which define its future interactions with the world and its inner information processing. Informational structures of an agent become semantic information first in the case of highly intelligent agents.

## 13 DEVELOPMENTS AND PROSPECTS OF NATURAL COMPUTATION. COMPUTING AS NATURAL SCIENCE

When we talk about natural computation by "nature" we mean everything that physically exists – not only living organisms, animals, plants and microorganisms, geological formations, astronomical objects but also machines, humans and human societies understood as physical systems – in other words all that can be described as existing in terms of matter/energy and space/time. Info-computational framework in effect replaces matter/energy (in space/time) with more basic formulation in terms of information/computation (in space/time).

On different levels of physical organization we find different types of natural computation: on quantum level, there is quantum computation, on the molecular level there is molecular computation, higher up in hierarchy we find nano-computation, networks of proteins are computing in living organisms, DNA code governs variety of computational processes in cells, metabolic processes are at the same time information processing and they are constitutive of life. Maturana and Varela equate cognition with life. [54][55] Computations of nervous systems resemble neural network models, living organisms as wholes are regulated on variety of levels and so are ecologies.

Information processing going on in the physical world can be modelled as computation – some of it on continuous flow of signals, some on discrete signals or symbols, some within living agents without conscious control, whilst other which proceed via languages require conscious living organisms for information to be processed. Morphological computing can be considered as a basis for all those physical processes that can be studied as information self-structuring. [23][48][49]

## 14 CONCLUSIONS & FUTURE WORK

*"I invite readers not on a visit to an archaeological museum, but rather on an adventure in science in making"*

Prigogine [56] p. IX

In this article too, a new science in making is presented. Starting with the short history of computational machinery and models, presentation focuses on the current state of the art of computing machinery and complex biological and social systems/networks which all are in need of better models of computation. Present account highlights several topics of importance for the development of new understanding of computation and its role in the physical world: natural computation and the relationship between the model and physical implementation, interactivity as fundamental for computational modelling of concurrent information processing systems such as living organisms and their networks, and the new developments in mathematical modelling needed to support this generalized framework. Besides the Turing machine model as well developed and generally established model of computation, variety of new ideas, still under developments are taking shape and have good prospects to extend our understanding of computation and its relationship to physical implementations.

As Stephen Hawking aptly noticed, in spite of enormous attraction of the idea of final theory of everything (including such theory of everything computational), the progress goes on:

*"Some people will be very disappointed if there is not an ultimate theory that can be formulated as a finite number of principles. I used to belong to that camp, but I have changed my mind. I'm now glad that our search for understanding will never come to an end, and that we will always have the challenge of new discovery."* [57]

## ACKNOWLEDGMENTS

The author would like to acknowledge insightful comments of two anonymous reviewers and numerous instructive discussions with Mark Burgin on different models of computation.

## REFERENCES

[1] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, pp. 433–460, 1950.

[2] A. Hodges, *Turing. A Natural philosopher*. London: Phoenix, 1997.

[3] A. M. Turing, "On computable numbers, with an application to the Entscheidungs problem," *Proceedings of the London Mathematical Society*, vol. 42, no. 42, pp. 230–265, 1936.

[4] B. J. Copeland, "What is computation?," *Synthese*, vol. 108, no. 3, pp. 335–359, 1996.

[5] M. Burgin, *Super-Recursive Algorithms*. New York: Springer-Verlag New York Inc., 2005, pp. 1–320.

[6] M. Burgin and G. Dodig-Crnkovic, "Information and Computation – Omnipresent and Pervasive," in *Information and Computation*, New York/London/Singapore: World Scientific Pub Co Inc, 2011, pp. vii –xxxii.

[7] D. Goldin, S. Smolka, and P. Wegner, Eds., *Interactive Computation: The New Paradigm*. Berlin, Heidelberg: Springer, 2006.

[8] S. Stepney, S. L. Braunstein, J. A. Clark, A. M. Tyrrell, A. Adamatzky, R. E. Smith, T. R. Addis, C. G. Johnson, J. Timmis, P. H. Welch, R. Milner, and D. Partridge, "Journeys in Non-Classical Computation I: A Grand Challenge for Computing Research," *Int. J. Parallel Emerg. Distr. Syst.*, vol. 20, pp. 5–19, 2005.

[9] S. Stepney, S. L. Braunstein, J. A. Clark, A. M. Tyrrell, A. Adamatzky, R. E. Smith, T. R. Addis, C. G. Johnson, J. Timmis, P. H. Welch, R. Milner, and D. Partridge, "Journeys in Non-Classical Computation II: Initial Journeys and Waypoints," *Int. J. Parallel Emerg. Distr. Syst.*, vol. 21, pp. 97–125, 2006.

[10] S. B. Cooper, B. Löwe, and A. Sorbi, *New Computational Paradigms. Changing Conceptions of What is Computable. Springer Mathematics of Computing series, XIII.* Springer, 2008.

[11] G. Rozenberg, T. Bäck, and J. N. Kok, Eds., *Handbook of Natural Computing*. Berlin Heidelberg: Springer, 2012.

[12] P. Denning, "Computing is a natural science," *Communications of the ACM*, vol. 50, no. 7, pp. 13–18, 2007.

[13] P. Denning, “What is computation?: Editor’s Introduction,” *Ubiquity*, no. October, pp. 1–2, 2010.

[14] A. Clark and D. Chalmers, “The Extended Mind,” *Analysis*, vol. 58, no. 1, pp. 7 – 19, 1998.

[15] J. Marchant, “In search of lost time.,” *Nature*, vol. 444, no. 7119, pp. 534–8, 2006.

[16] “The History of Computing Project web page,” 2012. [Online]. Available: http://www.thocp.net/index.html.

[17] Y. Wang, “The Theoretical Framework of Cognitive Informatics,” *Int’l J. of Cognitive Informatics and Natural Intelligence*, vol. 1, no. 1, pp. 1–27, 2007.

[18] H. Markram, “The blue brain project,” *Nature reviews. Neuroscience*, vol. 7, no. 2, pp. 153–60, Feb. 2006.

[19] G. Rozenberg and L. Kari, “The many facets of natural computing,” *Communications of the ACM*, vol. 51, pp. 72–83, 2008.

[20] S. Stepney, “The neglected pillar of material computation,” *Physica D: Nonlinear Phenomena*, vol. 237, no. 9, pp. 1157–1164, 2008.

[21] G. Dodig-Crnkovic and V. Mueller, “A Dialogue Concerning Two World Systems: Info-Computational vs. Mechanistic,” *Information and Computation*. World Scientific Pub Co Inc, Singapore, pp. 149–84, 2009.

[22] Y. Wang, “On Abstract Intelligence: Toward a Unifying Theory of Natural, Artificial, Machinable, and Computational Intelligence,” *Int. J. of Software Science and Computational Intelligence*, vol. 1, no. 1, pp. 1–17, 2009.

[23] G. Dodig-Crnkovic, “Physical Computation as Dynamics of Form that Glues Everything Together,” *Information*, vol. 3, no. 2, pp. 204–218, 2012.

[24] C. Hewitt, “What is computation? Actor Model versus Turing’s Model,” in *A Computable Universe, Understanding Computation & Exploring Na-ture As Computation*, H. Zenil, Ed. World Scientific Publishing Company/Imperial College Press, 2012.

[25] G. Dodig-Crnkovic and M. Burgin, “Unconventional Algorithms: Complementarity of Axiomatics and Construction,” *Entropy*, vol. 14, no. 11, pp. 2066–2080, 2012.

[26] P. Wegner, “Interactive foundations of computing,” *Theoretical computer science.*, vol. 192, no. 2, 1998.

[27] D. Goldin and P. Wegner, “Paraconsistency of Interactive Computation,” in *PCL 2002 (Workshop on Paraconsistent Computational Logic*, 2002, pp. 109–118.

[28] P. Bohan Broderick, “On Communication and Computation,” *Minds and Machines*, vol. 14, no. 1, pp. 1 – 19, 2004.

[29] S. Abramsky, “Information, Processes and Games,” in *Philosophy of Information*, J. Benthem van and P. Adriaans, Eds. Amsterdam, The Netherlands: North Holland, 2008, pp. 483–549.

[30] V. Schachter, “How Does Concurrency Extend the Paradigm of Computation?,” *Monist*, vol. 82, no. 1, pp. 37–58, 1999.

[31] C. J. Maley, “Analog and digital, continuous and discrete,” *Philos. Stud*, vol. 155, pp. 117–131, 2010.

[32] R. Trenholme, “Analog Simulation,” *Philosophy of Science*, vol. 61, no. 1, pp. 115–131, 1994.

[33] W. J. Freeman, “The neurobiological infrastructure of natural computing: Intentionality,” *New Mathematics and Natural Computing, NMNC*, vol. 5, no. 1, pp. 19–29, 2009.

[34] B. MacLennan, “Natural computation and non-Turing models of computation,” *Theoretical computer science.*, vol. 317, no. 1, 2004.

[35] E. Wigner, “The Unreasonable Effectiveness of Mathematics in the Natural Sciences,” *Communications in Pure and Applied Mathematics*, vol. 13, no. 1, 1960.

[36] G. Chaitin, “Mathematics, Biology and Metabiology,” 2009. [Online]. Available: http://www.umcs.maine.edu/~chaitin/jack.html.

[37] A. Sloman, “The Irrelevance of Turing machines to AI,” in *Computationalism – New Directions (M. Scheutz, Ed.)*, Cambridge, Mass: MIT Press, 2002, pp. 87–127.

[38] A.-L. Barabasi, V. W. Freeh, H. Jeong, and J. Brockman, “Parasitic computing,” *Nature*, vol. 412, pp. 894–897, 2001.

[39] S. B. Cooper, “Turing’s Titanic Machine?,” *Communications of the ACM*, vol. 55, no. 3, pp. 74–83, 2012.

[40] H. Zenil, Ed., *A COMPUTABLE UNIVERSE. Understanding Computation & Exploring Nature As Computation*. Singapore: World Scientific Publishing Company/Imperial College Press, 2012.

[41] J. Hintikka, “Logic as a Theory of Computability,” *APA Newsletter on Philosophy and Computers*, vol. 11, no. 1, pp. 2–5, 2011.

[42] G. Priest and K. Tanaka, “Paraconsistent Logic,” *The Stanford Encyclopedia of Philosophy*. Zalta, Edward N., 2013.

[43] L. Floridi, “A defense of informational structural realism,” *Synthese*, vol. 161, no. 2, pp. 219–253, 2008.

[44] G. Bateson, *Steps to an Ecology of Mind: Collected Essays in Anthropology, Psychiatry, Evolution, and Epistemology*. University Of Chicago Press, 1972, pp. 448–466.

[45] J. A. Wheeler, “Information, physics, quantum: The search for links,” in *Complexity, Entropy, and the Physics of Information*, W. Zurek, Ed. Redwood City: Addison-Wesley, 1990.

[46] G. Dodig-Crnkovic, “Information and Energy/Matter,” *Information*, vol. 3, no. 4, pp. 751–755, 2012.

[47] G. Dodig-Crnkovic and R. Giovagnoli, “Natural/Unconventional Computing and its Philosophical Significance,” *Entropy*, vol. 14, pp. 2408–2412, 2012.

[48] G. Dodig-Crnkovic, “Info-computationalism and Morphological Computing of Informational Structure,” in *Integral Biomathics*, A. Simeonov, P., Smith, L. and Ehresmann, Ed. Berlin, Heidelberg: , 2012.

[49] G. Dodig-Crnkovic, “The Info-computational Nature of Morphological Computing,” in *Theory and Philosophy of Artificial Intelligence*, SAPERE., V. C. Müller, Ed. Berlin: Springer, 2012, p. forthcoming.

[50] G. Dodig-Crnkovic and R. Giovagnoli, *Computing Nature*. Berlin Heidelberg: Springer.

[51] G. Dodig-Crnkovic, “Significance of Models of Computation from Turing Model to Natural Computation,” *Minds and Machines,*, vol. 21, no. 2, pp. 301–322, 2011.

[52] K. M. Sayre, *Cybernetics and the Philosophy of Mind*. London: Routledge & Kegan Paul, 1976.

[53] A. M. Turing, “The Chemical Basis of Morphogenesis,” *Philosophical Transactions of the Royal Society of London*, vol. 237, no. 641, pp. 37–72, 1952.

[54] H. Maturana, *Biology of Cognition*. Ft. Belvoir: Defense Technical Information Center, 1970.

[55] H. Maturana and F. Varela, *Autopoiesis and cognition: the realization of the living*. Dordrecht Holland: D. Reidel Pub. Co., 1980.

[56] I. Prigogine, The End of Certainty: Time, Chaos and New Laws of Nature. New York: The Free Press, 1997.

[57] S. Hawking, “Gödel and the end of Physics.” [Online]. Available: http://www.damtp.cam.ac.uk/events/strings02/dirac/hawking/.

# Abstract Procedures and the Physical World

**Paul Schweizer**[1] and **Piotr Jablonski**[1]

**Abstract.** The paper examines some central issues concerning the notion of implementing abstract formal structures, including effective procedures and dynamical systems, in the realm of physical space-time. We address the view originally put forward by Putnam and Searle, that virtually any physical system can be interpreted as implementing virtually any computational formalism, and defend the general conclusion that realizing an abstract procedural structure is not an intrinsic property of physical systems, but rather is a purely observer-dependent ascription. In a parallel manner, the 'trivialization' arguments originally put forward against computationalism are extended to dynamical systems theory, an alternative abstract framework that has also been advocated as providing the theoretical foundation for mentality in the natural world. Rather than attempting to distinguish 'true' from 'false' cases of implementation, we distinguish *pragmatically useful* ascriptions from those that serve no epistemic purpose.

## 1 ENGINEERED IMPLEMENTATION

From a disembodied mathematical perspective, classical computation comprises an extremely well defined and stable phenomenon. Central to the theory of traditional computation is the intuitive notion of an effective or 'mechanical' procedure, and there are any number of different possible frameworks for filling in the details and making the idea rigorous and precise. Turing's 'automatic computing machines' [1] (TMs), supply a very intuitive and elegant rendition of the notion of an effective procedure, but there is a well known variety of alternative frameworks.

According to the widely accepted Church-Turing thesis, the class of computable functions is nonetheless captured in a mathematically absolute sense by the notion of TM computability, and every alternative formalization so far given of the broad intuitive notion of an effective procedure has been demonstrated to be equivalently powerful, and hence to specify exactly the same class of functions [2]. Thus the idealized notion of in-principle computability, where all finite bounds on input size, storage capacity and length of running time are abstracted away, seems to constitute a fundamental category, a stable and fundamental 'mathematical kind'.

A related further question is whether any sort of comparable feature carries over to computation as implemented or realized in the physical universe. Turing machines and other types of computational formalisms are *mathematical abstractions* and don't exist in real time or space. In order to perform *actual* computations, an abstract Turing machine must be realized by a suitable arrangement of matter and energy, and as Turing observed long ago [3], there is no privileged or unique way to do this. Like other abstract structures, Turing machines are *multiply realizable* - what unites different types of physical implementation of the same abstract TM is nothing that they have in common as physical systems, but rather a structural isomorphism expressed in terms of a higher level of description. Hence it's possible to implement the very same computational formalism using modern electronic circuitry, a human being executing the instructions by hand with paper and pencil, a Victorian system of gears and levers, as well as more atypical arrangements of matter and energy including beer cans serving as tokens of the symbol '1' and rolls of toilet paper serving as the tape.

Adopting the conventions introduced by Schweizer [4], let us call this 'downward' multiple realizability, wherein, for any given abstract structure or formal procedure, this *same* abstract structure can be implemented via an arbitrarily large number of *distinct* physical systems. And let us denote this type of downward multiple realizability as '↓MR'. After the essential foundations of the mathematical theory of computation were laid, the vital issue then became one of engineering – how best to utilize state of the art technology to construct rapid and powerful physical implementations of our abstract mathematical blueprints, and hence perform actual high speed computations *automatically*. This is a clear and deliberate ↓MR endeavour, involving the intentional construction of artefacts, painstakingly designed to follow the algorithms that we have created. From this top-down perspective, there is an obvious and pragmatically indispensible sense in which the hardware that we have designed and built can be said to perform genuine computations in physical space-time.

## 2 NATURAL COMPUTATION?

In addition to these comparatively recent engineering achievements, but presumably still members of a single underlying category of phenomena, various authors and disciplines propound the notion of 'Natural Computation' (NC), and invoke a host of indigenous processes and occurrences as cases in point, including neural computation, DNA computing, biological evolution, molecular and membrane computing, slime mould growth, cellular automata, ant swarm optimization, etc. According to such views, computation in the physical world is not merely artificial – it is not restricted to the devices specifically designed and constructed by human beings. Instead, computation is a seemingly ubiquitous feature of the natural order, and the artefacts that we have produced constitute only a very small subset of the overall class of computational systems that inhabit the physical universe.

The disciplinary and terminological practices surrounding NC plainly invite a more thorough and rigorous examination of the underlying assumptions involved. Salient

[1] Institute for Language, Cognition and Computation, School of Informatics, Univ. of Edinburgh, EH8 9AD, UK. Email: paul@inf.ed.ac.uk.

questions in need of scrutiny include: To what extent, if any, is computation a genuine *natural* kind – is there an intrinsic unity or core of traits systematically held in common by the myriad of purported examples of computation in the physical world? In what sense, if any, can computation be said to take place spontaneously, as a truly native, 'bottom-up' phenomenon?

The issue has pronounced conceptual importance with respect to positions on the conjectured computational nature of *mentality and cognition*. According to the widely embraced computational theory of mind (CTM), which underpins cognitive science, Strong AI and various allied positions in the philosophy of mind, computation (of one sort or another) is held to provide the scientific key to explaining and, in principle, reproducing mentality artificially. The paradigm maintains that cognitive processes are essentially computational processes, and hence that intelligence in the physical world arises when a material system implements the appropriate kind of computational formalism. So it's an immediate corollary of CTM that the human brain counts as an exemplary instance of natural computation.

Hence it is crucial to CTM's theoretical stance that there be a rigorous and precise analysis of physically grounded computation in the case of organically engendered human brains. But the issue has wider and independent significance, in an attempt to gain conceptual clarity on whether and to what extent computation can be cogently viewed as a natural occurrence. And this in turn requires a general theoretical investigation and articulation of what it means for computations and other sorts of abstractly specified formalisms and structures to be implemented in the physical realm. It is this last, overarching theme that will comprise the primary focal point of the ensuing discussion.

## 3 THREE DIFFERENT SENSES

For the sake of analytical precision, we will begin by disambiguating three possible senses in which real physical systems might be thought of as 'performing a computation', and where these distinct senses are often blurred or run together by proponents of NC.

First (1), a physical system or object may be said to *obey or satisfy* a particular equation or mathematical function. For example, a falling body in the earth's gravitational field will satisfy or obey Newton's equation for gravitational acceleration. Similarly, the planets orbiting the sun satisfy or obey Kepler's laws of planetary motion. This has lead various NC enthusiasts to claim that the planets orbiting the sun, falling bodies in the earth's gravitational field, etc., are in fact *computing the values* of the equations in question. Taken to its most extreme form, this becomes the assertion that physical processes and natural laws are themselves fundamentally *computational*, and hence that computation constitutes the foundational key to the natural order.

Second (2), the activities of a physical system or process may be precisely *modelled or simulated* by a given computational formalism or depiction. For example, it is possible to create highly accurate and explanatorily useful computer models which simulate the behaviour of various complex physical events such as earthquakes, climate change, hurricanes, particle collisions, protein folding, brain processes, etc. Again, the usefulness and accuracy of these computational models has lead proponents of NC to claim that the physical phenomena *themselves* are performing such computations or are somehow instances of such computations occurring in nature.

And third (3), a physical device or process may be said to literally *implement, realize or execute* a particular algorithm or effective procedure. Thus when I write a piece of code in some artificial programming language, say Prolog, and then run this code on my desktop computer, there is a very clear and paradigmatic sense in which the electro-mechanical hardware in question is performing or executing the algorithm explicitly encoded in Prolog.

It seems uncontroversial that (3) is the basic and indeed canonical sense of computation in the physical world, and constitutes the modern historical origin of the concept. But in the Prolog example used to illustrate the import of (3), the computation in question is not a natural occurrence ─ rather it's a direct result of human design and engineering. Human artefacts in the form of electromechanical hardware devices comprise the arrangements of matter and energy that carry out the actual computation in space and time, and the procedures being executed are specified in terms of artificial programming and machine languages. In such literal and exemplary cases, real world computation is a purely synthetic phenomenon.

However, this does not in itself rule out the possibility that there could be genuine *natural* computation in the stringent sense of (3), since it is still entirely possible that some appropriate version of CTM is true. For example, if Fodor [5] is correct, then the human brain, an organically engendered 'wetware' device, is running the Language of Thought (LOT) as an indigenous formal system of rule governed symbol manipulation, in a manner directly comparable with a computational artefact. Thus if Fodor is correct, then the human brain is a paradigmatic instance of NC in sense (3).

Accordingly, in the ensuing discussion we will treat Fodor's conjectured LOT as epitomizing what genuine computational processing in the natural world would look like in its most explicit form − a spontaneously generated, bottom-up case of formal symbol manipulation. And this is compatible with the foregoing discussion of downward multiple realization, since the relation between LOT and the brain could then be viewed in typical ↓MR terms. For example, an alternative mechanical device, physically quite unlike the brain, could presumably be constructed to implement the LOT in an artificial medium.

In order to construct such an implementation, we would need to utilize expertise in engineering and materials science, which exploited the lawlike regularities that characterize the time evolution of physical systems. This expertise would be required to design the material implementation in such a way that it could be methodically interpreted, at the appropriate level of description, as behaving in a manner isomorphic to the abstract processing structure of the human brain. Indeed, it's perfectly conceivable that if we could abstract out the relevant computational structure of the LOT physically realized in brain activity, then we could run this same abstract computational structure on some version of our existing artificial hardware. And then, in perfect accord with ↓MR, the systematic and predictable behaviour of both the brain and the artificial device, seen as systems governed by natural law, could be interpreted, *at a higher level of description*, as implementations of the same abstract computational structure.

## 4 CRITIQUE OF SENSES (1) AND (2)

As noted in sense (1) above, a physical system or object, such as a piece of electromechanical hardware, may be described as *obeying or satisfying* various equations or mathematical functions. And it is by utilizing our knowledge of these regularities that we are able to construct physical realizations of abstract computational procedures, and thereby systematically and reliably preserve the implementational mapping from abstract formalism to relevant sequences of states of the physical machine. What then is to be gained by then claiming that, in addition to performing a computation in virtue of systematically preserving this mapping, the hardware is performing yet *another*, underlying computation, simply in virtue of evolving through time in accordance with natural laws?

Such a claim seems to be founded on a conflation between two of Marr's [6] classic distinctions in levels of analysis. The salient difference between what is going on in (1) as opposed to (3) is precisely the difference between the level of bare mathematical function and the level of computational algorithm. For any given function there are many different algorithms for computing its input/output values, and a mathematical function or general equation on its own does not specify any corresponding formal method or single out any one of the many different possible corresponding effective procedures as privileged. Hence merely *satisfying an equation*, as in the case of a hardware device obeying a lawlike physical regularity, is too weak to underwrite an assertion of distinctively *computational* processing, because it leaves the vital procedural details completely unspecified. Exactly *which* particular algorithm for computing the values of Kepler's laws of planetary motion is the earth currently implementing? And articulated in precisely *which* abstract computational framework?

Thus sense (1) seems to constitute an unmotivated and theoretically unilluminating inversion of perspective. Human scientists have devised mathematical abstractions in the form of general equations in order to characterize observed regularities in physical behaviour. In turn, we utilize these rigorously characterized regularities to construct artefacts that can be systematically interpreted, at a higher level of description, as implementations of selected computational formalisms. In this respect, we have a well defined implementational foundation in brute physical behaviour. The ontological and causal status of real-time computations is thus grounded in a stable, non-computational medium. But to then assert that the implementational medium *itself*, simply in virtue of evolving in accord with certain abstractly characterized patterns, is thereby performing lower level computations, seems to threaten a causal/ontological regress. And unless the particular algorithms and formalisms purportedly being executed are explicitly specified and substantiated, the assertion seems to make no additional contribution. However, the general equations as such are central to our scientific theorizing and abstract representation of the natural world. In section 7 below we shall further investigate sense (1) and the status of such formal specifications in the particular guise of dynamical systems.

In the case of sense (2), the algorithmic details missing from sense (1) are provided, but they are located in the wrong place. When complex physical events and processes are accurately simulated via computational models, it is the *artificial* computational structures which compute the values of the laws, equations and regularities governing the physical phenomena being simulated. And indeed, this is why the *models* are accurate and useful. But what motivates the further claim that the complex physical phenomena are *themselves* somehow implementations of the computations performed by the artificial models? Again, the same equations and regularities could be computed by another computational model using different underlying algorithms, programming languages, etc. to calculate the relevant values. Which of the many distinct computational possibilities is privileged or singled out by nature? In agreement with Piccinini [7], we would advocate a sharp distinction between mere computational *modelling* and genuine computational *processing* in nature.

So in the ensuing discussion we will treat (3) as the literal and canonical sense of computation in the space-time arena. We diagnose sense (1) as derived from the mathematical characterization of fundamental regularities in nature, but where the additional attribution of computational activity is due to a conflation between Marr's distinct levels of bare mathematical function versus specific algorithm for computing the values of the function. Finally, sense (2) is a case of artificial *simulation* of natural events and processes, where the values of the regularities salient to sense (1) are explicitly computed, but where this computation is merely a tool of human heuristics and is not supported by nature. In this respect sense (2) is a hybrid of the more basic content involved in (1) and (3), and will not receive any further investigation. The ensuing discussion will focus primarily on (3) as the paradigmatic sense of computation in the physical world, and will also provide an allied investigation of sense (1), since both (1) and (3) are cases of applying explicit renditions of abstract, formal procedures directly to physical events and processes.

Of course, to some extent the issue could be seen as purely terminological. One *could* choose to brand computation in sense (3) as 'classical' or 'Turing' computation, and then label senses (1) and (2) as 'computation', but of a different, broader sort. But for this broadening of the scope of application of the term to count as useful and well motivated, it would need an accompanying story to explain (i) what essential characteristics are had in common to unify all three apparently quite disparate senses of the term, and then (ii) why the category of phenomena so unified should be called 'computation' and not something else.

We don't wish to dwell on mere terminological or taxonomical disputes, and hence maintain that, whatever use of terminology one may adopt, sense (3) has clear and paradigmatic import, and it is this interpretation of the word that we wish to emphasize and investigate. Furthermore, whatever may be going on in most cases of (1) and (2), be it felicitously categorized as 'natural computation' or as something else, it is still quite distinct from what is captured by sense (3).

## 5 COMPUTATION IS NON-INTRINSIC

We will now articulate and begin to defend one of the main theses of the paper, a thesis stemming from arguments originally put forward by Putnam [8] and Searle [9, 10], that even in the quite restricted and canonical sense of (3), computation is not an inherent or intrinsic characteristic of any physical system. Instead, it's a purely *observer dependent ascription*, projected onto a physical system via an act of human interpretation. Furthermore, the extent to which a physical device can be interpreted as realizing any sufficiently rich computational formalism, such as an abstract Turing machine, is not absolute,

but instead is always a matter of degree of approximation. And the choice to interpret a physical device as implementing a particular abstract formalism is always relative to our particular purposes and potential epistemic gains.

Our normal practice of interpreting specialized artefacts as performing computations is clearly of very high pragmatic value. Nonetheless, such interpretations are ultimately dependent on human conventions and are not intrinsic to the hardware itself. Thus computation in the physical world is not sustained or underwritten by the innate structure of the systems interpreted as realizers, and computation as such is not a natural kind. We will begin our defence of this view by examining some well known arguments concerning the theoretical possibility of multifarious 'deviant' interpretations.

## 6 TRIVIALIZATION ARGUMENTS

Various critics of CTM have put forward a family of 'trivialization arguments', directly relevant to sense (3) above. The arguments are based on the contention that the notion of a physical system implementing a computational formalism is overly liberal to the point of vacuity. As a case in point, Putnam [8] offers a proof of the thesis that *every* open physical system can be interpreted as the realization of *every* finite state automaton. Putnam's argument will be explored in more detail in section 7, in the context of Dynamical Systems theory.

In the current section of the paper we will consider the closely related position advanced by Searle [9], who argues that virtually any physical system can be interpreted as following virtually any program. Thus hurricanes, our digestive system, the motion of the planets, even an apparently inert lecture stand, all possess a level of description at which they instantiate any number of different abstract formal procedures. The stomach has inputs, internal processing states and outputs, and if one wanted to, one could interpret the inputs and outputs as code for any number of different symbolic processes. And in [10] Searle attempts to illustrate the extreme conceptual looseness of the notion of implementing an abstract formalism by famously claiming that the molecules in his wall could be interpreted as running the WordStar program.

Again adopting conventions introduced by Schweizer [4], let us label multiple realizability in this direction, wherein any given *physical system* can be interpreted as implementing an arbitrarily large number of different *computational formalisms* 'upward MR' and denote it as '↑MR'. The basic import of ↑MR is the *non-uniqueness* of computational ascriptions to particular physical systems. In the extreme versions suggested by Putnam, Searle, and more recently Bishop [11], there are apparently no significant constraints whatever – it is possible in principle to interpret every open physical system as realizing every computational procedure. Let us call this extreme version 'universal upward MR' and denote it as '↑MR*'. Mere ↑MR is weaker than ↑MR*, since the former does not assert that there are no salient constraints, and hence ↑MR would be consistent with the denial that, e.g., the molecules in Searle's wall can in fact be interpreted as implementing the WordStar program, although every physical system is still interpretable as implementing some very large set of distinct computations.

In the present discussion we will not argue for or against ↑MR* but instead confine our considerations to the more modest ↑MR. In view of ↑MR, it's still never the case that any given computational interpretation of a physical system is privileged or unique, and this is far more difficult to deny than the powerful and broad sweeping ↑MR*. In turn, the non-intrinsic status of computation would seem to follow as a direct consequence of mere ↑MR alone. As long as there are at least two distinct interpretations, there is no objective fact of the matter regarding *which* computation is 'actually' being performed, nor which of the alternatives is the 'correct' or 'real' account. And this is because the computation itself is not an intrinsic property of the physical device, and is instead dependent on a human observer to supply the various alternative interpretations.

This is not to say that it's purely a matter of caprice, and that there are no objective constraints that the interpretation must satisfy. Instead, the situation is perhaps comparable to the distinction between natural kinds, such as water, and conventional kinds, such as being a table. Even though membership in either kind might be based on criteria whose satisfaction (or not) is a matter of objective truth, still the criteria for conventional kinds are not intrinsic, and there is nothing about the particular arrangement of matter now holding up my desk top computer which makes it intrinsically a table. The salient criteria stem purely from human practices and stipulations rather than from, e.g., fundamental microstructure or natural law.

The original trivialization arguments are intended to undermine CTM, by showing that attributions of computational processing are overly liberal to the point of vacuity, and hence cannot serve as a criterion for mentality in the natural world. But the potential scope of application is clearly much wider, and they also serve to trivialize the idea of 'Natural Computation' in general. According to ↑MR*, anything computes everything, and hence computational processing in the natural world turns out to be far more rampant and ubiquitous than proponents of NC ever suspected.

## 7 SYNTAX, SEMANTICS, PHYSICS

At the abstract, formal level, computation is an essentially syntactic phenomenon, and how we choose to interpret arrangements of matter and energy as constituting, say, tokens of an abstract syntactic type, and thus specifying an implementation of the basic computational vocabulary, is entirely independent of physical composition. For example, in the downward ↓MR direction there is a more or less limitless diversity in the ways in which material patterns and arrangements can be viewed as implementing the binary notation of '0' and '1', from ink marks on a piece of paper, stones placed in wooden boxes, patterns on old-fashioned punch cards, electric voltages, beer cans positioned on rolls of toilet paper, … And this applies in the reverse ↑MR direction as well, wherein the same stones placed in wooden boxes can be interpreted as implementing any number of distinct computational formalisms.

Classical computation is rule-governed syntax manipulation, and it is no more intrinsic to physical configurations than is syntax itself. It is also worth observing that discrete states are themselves idealizations, since the physical processes that we interpret as performing computations are in fact continuous, and we must abstract away from the continuity of the underlying substrate and impose a scheme of conventional demarcations to attain discrete values. Hence even

this elemental building block of digital procedures must be projected on to the natural order from the beginning. The irresistible conclusion to be drawn is that there is a fundamental gap separating 'concrete' physical reality from the human-based ascriptions of abstract syntactic features.

In turn, there is yet *another* fundamental gap separating abstract syntactic features from their semantic interpretation. Just as syntax is not intrinsic to physics, so too semantics is not intrinsic to syntax. Just as being an instance of the spoken English sentence 'The cat is on the mat' is not an inherent property of the sound waves constituting any particular utterance token, so too, the associated proposition comprising the *interpretation* of the utterance is not intrinsic to the abstract syntactic structure. Instead, the associated meaning is determined via arbitrary human convention, and the same syntactic item could just as well have had the interpretation currently expressed in English by 'The rat is on the table' or 'The dog is on the hearth'.

In the context of classical computation, one of the key constraints in the notion of an effective procedure is that the rules can be followed 'mindlessly', i.e. without knowing what the manipulated symbols are supposed to *mean*. As a consequence, there is no unique meaning determined by the procedure as such, and a multitude of distinct and *incompatible* interpretations are always possible. As a simple example, a Turing machine intended to compute the values of a particular truth function, say inclusive disjunction, can be easily reinterpreted as computing conjunction instead, simply by flipping our interpretation of the symbols '0' and '1', so that '0' is construed as denoting true while '1' denotes false. And the same procedure interpreted as computing conjunction could instead be construed as computing the values of the arithmetical function of multiplication. restricted to the numerical inputs 0 and 1.

Similarly, formal systems in general are such that the transformations on symbols are not specified with reference to their intended interpretation. Many classical *negative* results in mathematical logic stem from this separability between formal syntax and meaning. The various upward and downward Löwenheim-Skolem theorems show that formal systems cannot capture intended meaning with respect to infinite cardinalities. As another eminent example, Gödel's incompleteness results involve taking a formal system designed to be 'about' the natural numbers, and systematically reinterpreting it in terms of its own syntax and proof structure. As a consequence of this 'unintended' interpretation, Gödel is able to prove that arithmetical truth, an exemplary *semantical* notion, cannot, in principle, be captured by finitary proof-theoretic means.

In summary, there are *two* fundamental gaps separating formal procedures, standardly interpreted as computing the values of given functions, from the physical processes that we construe as implementing such procedures. First there is the gulf dividing the intended semantic interpretation from the bare syntactic formalism, and second there is the chasm between abstract syntactic formalism and physical reality. In *both* cases the gaps can only be bridged by an act of purely conventional human interpretation. And it is in this sense that computation in the physical world is inherently observer dependent.

## 8 PUTNAM AND DYNAMICAL SYSTEMS

We will now take a closer look at the foregoing sense (1) sometimes used to support the view of computation in nature. As was noted in section 3, the term 'computation' is occasionally meant as a description of the fact that some physical processes *satisfy* or *obey* a mathematical equation. Although, as we noted, there is little reason to believe that planets orbiting the sun compute an algorithmic approximation of the Newtonian laws of motion. So instead of viewing such cases in explicit NC terms, we will analyze the underlying notion of projecting the bare, abstract procedural descriptions furnished by dynamical systems onto the time evolution of physical phenomena. Thus, instead of a mapping between the physical states of the system and the computational states of an algorithm, there is a mapping between physical states and the variables of the appropriate dynamical system (DS). Following Jablonski [12], we argue that DS, when viewed in this manner, exhibit striking similarities to the inputless Finite-State-Automata (iFSA) analyzed by Putnam [8], and that the endeavour succumbs to very similar difficulties as originally proposed by Putnam in the context of strong ↑MR* arguments against computationalism.

Since an iFSA is defined by the set of monadic computational states and the rules of transitions from any given state to the next, any execution of an algorithm will take the form of the sequence of states. Putnam has noted that the evolution of any non-cyclic physical phenomena can be divided into a sequence of periods in such a way that we can map the physical states of the system onto the computational states of an iFSA. Say, we want to show that Searle's wall realises a computation performed by the iFSA defined by the states A, B and C and rules of transition A>B, B>C and C>B. If initialised in the state A the automaton will transit through states ABCBCB and will continue to oscillate between states C and B. We can claim that the wall performed 6 steps of the computation within any period of time e.g. from 12:00 to 12:06. We simply label all physical states of the wall within the first minute as computational state A, within the second minute as state B, third – C, forth – B, fifth – C and sixth – B.

Since the complexity of the thermal movements in the wall, openness of the system and, possibly, some non-reversible physical phenomena (e.g. radioactive decay) insure that every physical state of the wall will manifest itself only once, the labelling will be functional i.e. for every physical state only one computational state is given (however, a single computational state can be realised by many different physical states).

In applying this same basic strategy using a Dynamical Systems framework rather than inputless Finite-State-Automata, we first note that a system is defined by the set of its variables and rules governing the evolution of the variables over time. Unlike iFSA that have a finite number of possible states, the states of a DS are given by the vector of real number variables. Thus DS have an infinite number of possible states. Usually, the dynamical models are defined as deterministic, continuous systems so the rules governing the dynamics are given by a set of differential equations. Equations are the continuous equivalents of the rules of transitions for the iFSA. Finally, the phase-space trajectory of the system, the evolution of its variables over time, can be compared to the sequence of computational states of the iFSA.

The analogies between the two formalisms can be summarised as follows:

| iFSA | DS |
|---|---|
| Finite number of states | Infinite number of states |
| Initial state | Initial conditions |
| Table of transitions | Differential equations |
| Computational steps | Time |
| Sequence of states | Trajectory through the phase space |

Using these analogies, we can see that it is possible to map any finite trajectory of any DS onto any non-cyclic physical process in a manner similar to Putnam's original strategy. We map the first point of the trajectory onto the first physical state of the system and all consecutive points onto the corresponding physical states.

First, we need to map a real, physical time of the process onto the abstract time of the DS. Since we are interested in the finite period, we may define a mapping function *M* as follows:

$$\tau = M(t) = \left[\frac{t - t_0}{t_e - t_0}\right] * [\tau_e - \tau_0] + \tau_0$$

where $\tau$ is an abstract time of the DS, $t$ – real, physical time, $\tau_0$ – the beginning of the dynamical process, $\tau_e$ – the end of the dynamical process, $t_0$ and $t_e$ – the beginning and the end of the physical process in real time.

We need to include Putnam's condition of non-cyclic behaviour of the physical system in order to guarantee, that every moment of the physical time $t$ indicates just one physical state $s_t$ of the system. Given that DS is defined by its equations of motion

$$\frac{dx}{d\tau} = g(x)$$

where $x$ is a multidimensional vector of variables, and a single trajectory of the DS is determined by the initial conditions $x_0$ and the time interval, we can take an integral of the equations of motion that will have the form of the function of the state of the DS over time $x = h(\tau)$. This integral defines the trajectory of the system in phase space and is the equivalent to the sequence of states in the case of iFSA. However, since the trajectory is composed of the infinite number of points it has to be expressed as a function of time and cannot be presented in a form of a table of values. Now we can form a labelling function *f* that assigns abstract states of the DS to the physical states $s_t$ of the system.

$$f(s_t) = h(M(t))$$

Thus for every physical state $s_t$ we know the time *t* when it appeared (since the behaviour of the system is non-cyclic). Knowing the time *t* and the time-mapping function *M(t)* we can determine the corresponding time $\tau = M(t)$ of the abstract dynamics. Eventually, since we know the states of the DS as a function *h* of time $\tau$ we can determine the formal state $x$ of the DS, and what follows, the values of its variables. In other words, for every non-cyclic physical system and finite period of time we are able to map its states onto states of any DS.

A main difference between DS and iFSA lies in the fact that the latter have a finite number of states and steps while the former have an infinite number. However, since physical systems are continuous in nature the mapping still can be carried out. In order to perform Putnam's version of trivialization we need to know *in advance* the sequence of the states of the iFSA. In the case of DS we ought to know the trajectory of the system. That was easy for iFSAs since computation of the sequence required only a finite number of steps. However for the DS we need to integrate the equations of the DS. In many cases, nonlinear differential equations do not have analytic solutions so we are unable to obtain the trajectory of the system in the form of a function of variables over time.

This limitation is however, only epistemic in nature. Every DS has a trajectory defined for every initial condition even if we are unable to discover its analytic form. We still may conclude that, in principle, there is a mapping between any DS and physical system, although in many cases we will unable to provide the specific details.

## 9 CONSTRAINTS ON IMPLEMENATION

In response to ↑MR* and the trivialization arguments, various authors, including Chrisley [13], Chalmers [14], Copeland [15], and Block [16] have proposed a number of constraints on computational interpretations in an attempt to distinguish 'true' cases of implementation from the myriad of purportedly 'false' cases utilized by Putnam and others. Two of the most intuitively compelling restrictions are supplied by (i) causal and (ii) counterfactual considerations. The first constraint holds that the pattern of abstract state transitions constituting a particular run of the computational procedure on a particular input must map to an appropriate transition of physical states of the machine, where the relation between succeeding states in this physical sequence is governed by proper causal regularities. The second constraint holds that a necessary condition for being a 'genuine' implementation is the ability of the mapping to support counterfactual sequences of transitions on inputs not actually given. This constraint is prompted by the fact that various ↑MR* mappings from formalism to physical system, given by Putnam and others, are defined only for a single run and say nothing about what would have happened if a different input had been given (see Bishop [11] for an exception).

Although both (i) and (ii) are intuitively plausible suggestions, we view both as ultimately unsuccessful in blocking the trivialization results. See Schweizer [4] for arguments to the effect that, within the context of computation in canonical sense (3), constraint (i) provides a sufficient but not a necessary condition, while (ii) is unsatisfiable in principle (for sufficiently rich frameworks, such as those invoked in the Church-Turing thesis), and can serve only as a measure of degree of approximation.

In the context of 'computation' in sense (1), and pertinent to the above extension of the ↑MR* arguments to Dynamical Systems, we will now briefly examine two of the main points raised by Chalmers [14] and address them in the context of DS trivialization. In line with (ii) above, it has been objected that Putnam's labelling does not map counterfactual computational states onto any physical states. If a given iFSA is defined by states A, B, C, D and transitions A>B, B>C, C>B and D>B it could also perform the sequences ABCBCB, BCBCBC, CBCBCB or DBCBCB. Because Putnam's method of construction of the labelling function only works for the single chain of physical events, we would be unable to map state D onto any physical state of the wall because D did not appear in the sequence used for labelling.

Since an iFSA can perform only a finite number of state sequences, we need to form a labelling function that maps all those possible sequences onto some states of the physical

system and show that such a labelling can be constructed for an arbitrary physical device or phenomenon. In the context of classical computation, this stronger version of trivialization is known as the "*clock and dial*" reply to Chalmers objection. The argument states that every physical system can contain not only non-reversible physical phenomenon (a *clock*) but also some physical magnitude that can be set into a number of distinct states and will remain in the same state for the given period of time (a *dial*). Thus the complete state of the system is defined by the pair [$\boldsymbol{d}$, $S_t$ ], where $\boldsymbol{d}$ is the state of the dial and $S_t$ – the state of the clock at time *t*. Since our iFSA can perform four different sequences, $\boldsymbol{d}$ will take four values. The state of the dial will determine which sequence is mapped onto the states of the *clock*. Thus if the dial is set to $d_1$ we will map the first sequence onto physical states of the system during this time period. If the dial is set to $d_2$ we will map the second sequence and so on. One can argue that not every physical system contains *clock* and *dial* components, however there is certainly a large class of such systems thus the '↑MR' is conserved.

A parallel strategy can be applied in the context of DS trivialization. The infinite number of possible trajectories of the DS forces us to modify the "*clock and dial*" response to the argument. The *clock* will have to transit continuously through an infinite number of physical states within a finite amount of time (which seems to be uncontroversial since physical time is continuous) while the *dial* will have to be substituted by devices that can be set in an infinite number of states. We may picture the devices as set of continuous *sliders*, one for every variable in the equations of the DS. Every initial state of the DS can be encoded by the appropriate setting of the *sliders* just as every initial state of the iFSA can be encoded as a position of the *dial*. After that, we map the integrated trajectory of the DS onto the run of the *clock*.

A second requirement is reliability, which is closely akin to causal regularities − a proper labelling function should interpret not only one but every evolution of the physical system from the same initial state as a realisation of the same algorithm. If Searle is right, then he should be able to reliably and repeatedly reset his wall to a given initial condition and demonstrate that its physical evolution is identical to the one used for the initial labelling. Since the wall is an open system and (as required for the trivialisation argument) exhibits non-cyclic evolution it certainly will not repeat its states.

However, the "*clock and dial*" version of the argument seems to be immune to this objection. The dial can be reliably set into any of its states and the *clock* will reliably pass through the same sequence of moments. And it appears that there will be no significant difference in this regard between a continuous "*clock an dial*" and the sequential counterpart used for iFSA as long as we are able to demonstrate that the *clock* can reliably pass through its sequence of states and the *sliders* can stay in the unchanged position through the period of observation.

## 10 COMPUTATION AND PRAGMATICS

We would now like to propose a different perspective on the issue. Rather than distinguishing 'true' from 'false' cases of implementation, what these and other proposed constraints do instead is to go some distance in distinguishing interesting, conceptually rich and *pragmatically useful* implementations from the many uninteresting, trivial and useless cases that abound in the space of possibility. It's certainly true that there is no pragmatic value in most interpretive exercises compatible with ↑MR and ↑MR*. Ascribing computational activity to physical systems is *useful* to us only insofar as it supplies *informative outputs*, which in most cases will come down to new information acquired as a result of the implemented calculation.

So, interesting and useful observer dependent computation takes place when we can directly read-off something that *follows from* the implemented formalism, but which we didn't already know in advance and explicitly incorporate into the mapping from the start. That's the incredible value of our computational artefacts, and it's the only *practical* motivation for playing the interpretation game in the first place. Hence a crucial difference between our computational artefacts and the attributions of formal structure to naturally occurring open systems, as employed by ↑MR* exercises, is that the mapping in the latter case is entirely *ex post facto* and thus supplies us with no epistemic gains. The abstract procedural 'trajectory' is already known and used as the basis for interpreting various state transitions in the open system and hence characterizing it as an implementation. In sharp contrast, we can use the intended interpretation of our artefacts both to *predict* their future behavior, as well as *discover* previously unknown output values automatically.

And this is obviously why an engineered correlation obtains between fine-grained causal structure and abstract formal structure in the case of our artefacts – we want them to be informative and reliable! We also want them to be highly versatile, and this is where counterfactual considerations come to the fore in practice: over time we can do runs on a huge number of different inputs, and in principle the future outputs follow as direct consequences of the intended interpretation. So a physical system is *useful to us* as a computer only when its salient states are distinguishable by us with our measuring devices, and when we can put the system into a selected initial state to compute the output of our chosen algorithm on a very wide range of specific input values.

These *pragmatic* considerations supply clear and well motivated criteria for differentiating useful from useless cases of physical implementation. And we would advocate this type of pragmatic taxonomy in lieu of attempts to give overarching theoretical constraints purporting to distinguish 'true' from 'false' cases. Some basic desiderata for pragmatically valuable implementations include (a) fully automatic, (b) reliable, (c) versatile in the sense of computing values for a wide range of different inputs (d) non *ex post facto* (e) yielding increased predictive power with regard to future physical states of the implementing mechanisms, (f) possessing technologically manipulable initial configurations and output configurations detectable by our measuring devices and (g) physical rather than purely abstract constraints on the input and output characterizations.

Similarly, ascriptions of computation in the sense (1) to the physical systems are motivated by pragmatic reasons. Useful interpretations ought to yield simple formalisms whose equations we are able to integrate or at least investigate their properties with our mathematical resources. Variables should be mapped onto reliably observable physical magnitudes that figure in many scientific theories and are not proposed ad hoc.

Valuable interpretations of physical phenomena in terms of DS give us simplified formal description, epistemically useful and, hopefully, manifesting some predictive powers. We cannot however, claim that the physical process itself realises the mathematical equations in any other sense than that its behaviour can be fruitfully described using such equations.

## 11 CONCLUSION

Computation is an extrinsic, observer dependent interpretation that we project onto physical systems according to our purposes and potential epistemic gains. As such, it does not support a stable or independent natural kind. Diverse types of natural events and processes can be modelled or simulated using computational techniques, as in sense (2) above, but this is to be distinguished from canonical sense (3), in which the system *itself* is viewed as instantiating and executing an explicit formal procedure. However, various physical systems *do* spontaneously 'obey' clear regularities in their evolution through time, and many such regularities have been mathematically characterized in terms of Dynamical Systems theory. Although this sense (1) reading does not comprise a case of genuine *computation*, in the strict connotation of executing a well defined formal procedure, it does provide a fundamental form of mathematical representation of the natural world.

Various opponents of the Computational Theory of Mind have provided trivialization arguments to the effect that, even in canonical sense (3), the notion of implementing a computational formalism is overly liberal to the point of vacuity. Such results serve to undermine not only CTM in particular, but the more encompassing notion of 'Natural Computation' in general. In a parallel manner, we extend this strategy to sense (1) 'computation' in the guise of Dynamical Systems theory, to argue that realizing such abstract formal structures is again a matter of observer dependent ascription. As with the original trivialization strategies aimed against CTM, this extended result has deep implications for the science of mind, since Dynamical Systems have been advocated (by, e.g. van Gelder [17]) as providing an alternative theoretical foundation for mentality in the natural world.

Advocates of CTM have proposed a number of constraints on the notion of 'genuine' implementation, in an attempt to block the trivialization results and uphold a robust notion computation in the physical world. However we argue that such constraints derive purely from human interest as opposed to underlying and independent matters of fact. Rather than serving to distinguish true' from 'false' cases of implementation, what the proposed constraints do instead is to help distinguish conceptually rich and *pragmatically useful* implementations from the many uninteresting, trivial and useless cases that abound in the space of theoretical possibility.

Although practical considerations clearly guide the design and construction of our computational artefacts, such pragmatic motivations do not justify any deep or ontologically grounded distinction between genuine versus trivial interpretations. Hence we support an anti-realist view of computation in nature, and implementing or realizing abstract formal structures in general is not an intrinsic property of physical systems. In particular we have viewed the notion of implementation in the context of senses (1) and (3), but the view generalizes to all the prospective forms of non-Turing 'computation' inspired by considering natural events and processes. These are abstract, observer dependent ascriptions projected onto a more basic physical substrate.

## REFERENCES


[1] Turing, A., 'On Computable Numbers, with an Application to the Entscheidungsproblem', *Proceeding of the London Mathematical Society*, (series 2), 42, 230-265, (1936).

[2] Boolos, G., Burgess, J.P. and Jeffrey, R.C., *Computability and Logic*, 5th edition, Cambridge University Press, (2007).

[3] Turing, A., 'Computing Machinery and Intelligence', *Mind*, 59: 433-460 (1950).

[4] Schweizer, P., 'Physical Instantiation and the Propositional Attitudes', *Cognitive Computation*, 4: 226-235 (2012).

[5] Fodor, J., *The Language of Thought*, Harvard University Press, (1975).

[6] Marr, D., *Vision*, CA: W. H. Freeman, (1982).

[7] Piccinini, C., 'Computational Modelling vs. Computational Explanation', *The Australasian Journal of Philosophy*, 85(1), 93-115, (2007).

[8] Putnam, H., *Representation and Reality*, MIT Press, (1988).

[9] Searle, J., 'Minds, Brains and Programs', *Behavioral and Brain Sciences* 3: 417-424, (1980).

[10] Searle, J., 'Is the Brain a Digital Computer?', *Proceedings of the American Philosophical Association*, 64, 21-37, (1990).

[11] Bishop, J. M., 'Why Computers Can't Feel Pain', *Minds and Machines*, 19, 507-516, (2009).

[12] Jablonski, P., *Trivialisation Arguments Against Dynamical Hypotheses*. MSc dissertation, University of Edinburgh (2012).

[13] Chrisley, R. L., 'Why Everything Doesn't Realize Every Computation', *Minds and Machines*, 4, 403-420, (1994).

[14] Chalmers, D. J., 'Does a Rock Implement Every Finite-State Automaton?', *Synthese*, 108, 309-333, (1996).

[15] Copeland, J., 'What is Computation?', *Synthese*, 108:335-359, (1996)

[16] Block, N., 'Searle's Arguments against Cognitive Science'. In J. Preston and J. M. Bishop *Views into the Chinese Room*, Oxford University Press, (2002).

[17] van Gelder, T. J., 'What Might Cognition Be If Not Computation?', *Journal of Philosophy*, 91: 345-381, (1995)

# The scandal of the computational universe

## First part: the qualitative concepts

Michael Nicolaidis
TIMA lab (CNRS, INPG, UJF)

**Abstract:** This work presents a comprehensive computational-universe vision. To start, we present the computational model adopted and we show the issues that have to be resolved in order to make credible the computational universe vision. To resolve these issues we introduce an ultimate limit of knowledge: observers that are part of a system/universe have no access on information concerning the fundamental nature of the elementary entities/particles composing the system/universe but only on information concerning their behaviour. Then, we use this limit to develop a computational universe model in which the behaviour of particles is the result of a computation-like process performed by meta-objects, and in which space and time are also engendered by this computation. The qualitative results obtained are used in a companion paper to propose a computational universe vision of special relativity and quantum mechanics.
**Keywords:** Pan-computationalism, computational universe.

### 1. Preliminaries

#### 1.1. Computational model

The computational universe idea introduced by Konrad Zuse [1][2], and further developed by Jurgen Schmidhuber [3], considers that the universe can be engendered by a computation. However, several issues should be addressed for making this kind of hypothesis credible. Before discussing these issues (section 1.2), let us clarify the sense in which the term "computation" is used hereafter.

In conventional computations the state of the computer is evolving according to certain rules coded by the software. Taking a more generic perspective, not restricted to the Turing paradigm, a computation-like process can be viewed as a process where the states of a system evolve according to certain rules. Then, a natural process can be viewed as a kind of computation, where its state evolves according to certain rules (the laws of physics). This could be extended to the process of evolution of the whole universe, and we could consider this evolution as a computation where the state of the universe evolves according to rules described by the laws of physics. In particular, these laws (quantum mechanics and relativity) will describe the rules determining the evolution of the states of objects (e.g. elementary particles) evolving in a veritable space with the flow of a veritable time. However, such an interpretation would be a simple carbon copy of the current models of physics, and will not bring new light in our vision of the universe. Thus, in this work we consider a more restricted concept of computation. In particular, a computer manipulates numerical variables. Hence, it cannot determine the position of a particle in a veritable space. Thus, the system has to be fully described by a set of numerical variables. This excludes the existence of a veritable space in which evolve the particles. Instead, space has to be engendered by a computation process operating over numerical variables. Furthermore the computation rules will not be copies of the current interpretations of the theories of physics. Instead, they exclude aspects in these interpretations that lead in non-computational rules. However, the computational rules *will have to produce the same observable behaviour as these interpretations*. For instance, in computers we use a time reference (implemented by a clock signal in digital computers). This creates a fundamental time, which is not compatible with relativity where there are as many time references as inertial frames (i.e. an infinite number). Also, in the current interpretation of special relativity all inertial frames have equal stance. Thus, computing the state of the objects (elementary particles) composing the universe in just one inertial frame is not sufficient for engendering a universe complying this interpretation. We then need to compute it for all inertial frames (infinite number). This will need using an infinite set of state variables (one per inertial frame), and will require infinite computing power or infinite time. Quantum superposition also introduces infinite set of states, since observables of continuum spectrum, like the position and momentum (but also observables of discrete spectrum like the energy), have infinite number of eigenvalues, resulting in the superposition of infinite number of values. We consider rules integrating these aspects, as well as quantum non-locality, as non-computational. Then, our computation model will obey the following constraints.

i. The state of each elementary object/particle is described by a finite number of variables.
ii. The computations of all object/particle variables are rated by a unique time variable (synchronous process).
iii. The computation of the state variables of distant objects/particles does not require instantaneous communication.
iv. The rules used to compute the states of the particle are described by analytical expressions.
v. The complexity of the computations of these expressions is not relevant as far as they can be done in finite time.

The above constraints correspond to a model that is computable in the Turing sense, except the fact that such computation will provide discrete results. However, analogue computers could be used to implement the analytical expressions of the rules in point iv. Note also that, there is an open discussing on whether time, space, and physical states are ultimately continuous or discrete. However, resolving this controversy is not in the scope of this article. Instead we will provide computational universe models for both view points.

Note also that, by succeeding to develop a computational universe model, which uses simple computational rules but produces the same observable behaviour as complex theories of physics, may provide new means for resolving several controversies of modern physics. Examples of such controversies, and their treatment in the context of the proposed vision, are presented in a paper companion to this article [4], which addresses relativity and quantum mechanics.

#### 1.2. Critical issues

In the previous section we have argued that adopting a model, which is carbon copy of the theories of physics, is lucking

motivation, as it will not bring new light on our vision of the universe. Thus, we adopted a simple computational model described above in points i through to v. However, such a simple model raises several critical issues. A first set of issues is related with the question of realism. A second set of is related with the requirement that the computational model should be able to engender a universe in which the observable behaviour of its processes should be the same as the one described by the theories of physics.

The first set of issues (related to the question of realism) is:

**R1-** A computation engenders virtual objects, but in our everyday life we perceive real objects.

**R2-** In our everyday life we perceive objects evolving over time in a real space. Thus, since several millennia we consider that our universe is composed of objects immersed in a veritable space and evolving with the flow of a veritable time. On the other hand computations can engender virtual objects immersed in a virtual space and evolving with the flow of time. This time is a primary entity (i.e. it is not engendered by the computation but it is rating it). For instance, the evolution of a virtual reality engendered by a computer is rated by the computer clock.

So, a first question is how can we resolve contradictions R1 and R2.

The second set of contradictions (P1 ÷ P5) is related with the theories of physics:

**P1:** In our model, the computation of the state variables of the elementary particles is rated by a time that is external to the computation, while space is engendered by the computation. This raises the following question. How could a virtual space engendered by a computation be merged with an external time rating the computation to give raise to the 4D space-time structure described by the theory of relativity?

**P2:** In our model, the evolution of the states of the computational universe is rated by a fundamental time. How can we conciliate this fundamental time with special relativity, where there is not fundamental time but as many times as inertial frames?

**P3:** In special relativity there is no privileged inertial frame, instead all inertial frames (i.e. an infinite number) have equal stance. In our model, we use a single set of state variables for each elementary particle. Thus, the computation of these states can be done in a single reference frame, which is then becoming privileged. How can we conciliate this privileged frame with relativity?

**P4:** Observables of continuum spectrum, like the position and momentum, but also observables of discrete spectrum like the energy, have infinite number of eigenvalues. Thus, quantum superposition for these observables results in infinite number of superposition values. This looks incompatible with our model, in which the state of each elementary particle is described by a finite number of variables.

**P5:** Our computational model excludes instantaneous communication. Thus, a last question is how it can engender a universe in which quantum non-locality implies that a particle can correlate instantaneously its state with the result of a measurement performed over its entangled counterparts?

While the above contradictions seem non reconcilable, we succeed resolving them by giving a very central role to the observers that are part of a universe, that is, observers composed of the same elementary entities (particles) as those composing any structure of the universe. For such observers we derive a fundamental limit of knowledge according to which they can access information concerning the behaviour of these entities but by no means information concerning their veritable nature. Due to this limit, such observers could not distinguish a universe in which particles occupy positions in a veritable space from a computational universe in which the positions of particles are determined by state variables. This way we resolve the contradictions R1 and R2 related with the question of reality. Then we treat the contradictions P1 through to P5 by developing a computational universe model in which:

- Lorentz transformations are not reflecting the veritable structure of a veritable space-time but become a consequence of the interaction laws used to compute the evolution of the states of particles and of the measurement means that dispose the internal observers of the computational universe. This allows resolving the contradictions P1, P2, and P3.
- The interpretation of quantum mechanics based on the quantum superposition concept is replaced by a computational model consisting in computing certain deterministic functions acting on stochastic signals. This way we succeed resolving contradiction P4.
- The instantaneous "communication" between the states of distant entangled particles is treated by exploiting the distinction of the time pacing the computation from the time experienced by the internal observers of the system. This way we resolve contradiction P5.

The following sections address the qualitative analysis of our approach. Then, in a companion paper [4], we use these qualitative results to develop the quantitative analysis of our approach treating special relativity and quantum mechanics.

**2. Internal Observers, Objects and Behaviours**

In this section we introduce the concept of the internal observer of a system, and use it to establish an ultimate limit of knowledge. This limit states that we can access information concerning the behaviour of the elementary entities (i.e. particles) composing the universe, but not information concerning the veritable nature of the entities produced this behaviour. This limit is implied by the fact that the nature of information that we can access is conditioned by our position as *internal observers of the universe (in the sense that we are constituted by the same elementary entities – particles - that compose any structure of the universe)*. As a consequence, the information collected by our sensorial systems as well as by the systems we built to observe nature, comes from the interactions of the particles composing these systems with other particles (like photons, electrons,….). The latter having previously interacted with the objects that we want to observe, their states are modulated by these interactions. Thus, they bring to our sensorial systems information that is the outcome of these interactions. But *the interactions represent the behaviour of the particles*.

To better understand the foundations of this limit let us consider a system made of several entities represented by light blue circles in figure 1. Each of these entities is an object, which comprises a state and exhibits a certain behaviour produced in the following manner. Each entity interacts with other entities (interactions illustrated schematically by gray lines) and receives

information concerning their state. It determines its next state as a function of its present state and of the present and/or past states of the entities with which it interacts. Entities described in such generic terms could correspond to various interacting objects, including the cells of a cellular automata, the nodes of a parallel processor, software agents, the elementary particles of our universe, … We can call the rules that the entities use to determine their next state (while interacting with other entities), laws of interactions. Let us consider in this system a set A of entities (for example those surrounded by a circle in gray dashed line in figure 1) and the set B of entities of the system that are external to A. We observe that the information the entities of set A receive from the entities of set B concerns the states of the later and the way in which these states evolve. Thus, for determining their next state, the only information that the entities of set A could receive concerning the entities of set B are related to the states of the latter and the way these states evolve. But the state of an entity and the way this state evolves represent its behaviour. Consequently, the states of the entities of set A can contain information concerning the behaviour of the entities of set B, but in no way they can contain information concerning the veritable nature (or structure) of the objects which produce this behaviour.

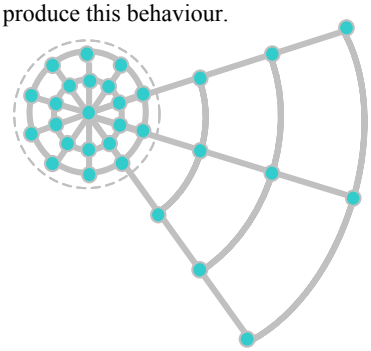


Figure 1. A system composed of a set of interacting entities

Let us now consider that the entities of set A are the elementary particles which compose a sensory organ (for example the eye) of an observer, as illustrated in figure 2.

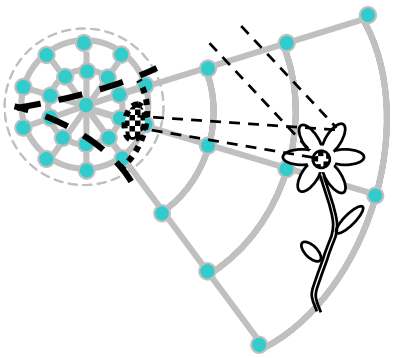


Figure 2. Objects and observer being part of a system

The particles of this sensory system interact with other particles and via these interactions they collect and transfer information to the particles that compose the mental structures of this observer. Thanks to this kind of information these structures form images of all kinds of objects of the system (e.g. universe) in which belongs the observer. For the reasons we have just explained, the states of the particles of these sensory and mental structures can contain only information concerning the behaviour of elementary particles. Thus, this discussion highlights *an ultimate limit of our knowledge*: as observers composed of the same kind of elementary particles as those composing any object of our universe (observers that are part of the universe or internal observers), we can have access to information concerning the behaviour of the elementary particles but by no means to information concerning the nature of the veritable objects which produce this behaviour. Therefore, these objects are *meta-objects* for the internal observers. We will refer to this limit as the ultimate limit of knowledge for internal observers (ULKIO) of any system.

**2.1 System closure versus internal observers**

In this section we consider a system structure more suitable for proposing a computational model of quantum systems, as the one we describe in a paper companion to this article [4]. Let us consider that the system in figures 1 and 2 is closed (in the sense that the entities composing the system could not interact with any entity external to it), and that the state of each entity of this system is determined from its own state and the states of the entities with which it interacts, following certain rules (interaction laws). The internal observer could observe (through interactions) a large number of times the state of any given entity and of the entities with which this entity interacts. Through interactions, he/she may also force the states of these entities at particular configurations and observe the outcome. Then, after a sufficient number of observations and with a certain amount of intelligence she/he could extrapolate the interaction laws. Afterwards, if these laws ere deterministic and he knows the state of a closed set of entities (i.e. not interacting with any other entities), he can use these laws to predict the evolution of this state.

Let us now consider the system of figure 3. In this system each entity is composed of two parts, the first is represented by a light blue circle and the second by a gray circle. In this system certain laws determine the state of the first (blue) part of each entity from the state of this part; the state of the gray part of the entity; and the states of the blue parts of the entities with which it interacts. Consider also that the gray part of its entity is autonomous. That is, its future state does not depend on the state of any other part of any entity but only on its own state. For instance, the gray part could be an autonomous pseudorandom generator or a random generator. We observe that for an internal observer of this system:

- Similarly to the states of the entities of figures 1 and 2, in figure 3 the states of the blue parts of the entities are observable and could also be forced through interactions to take particular values.
- An internal observer (i.e. an observer composed of elementary entities of the system), has no means for performing experiments in which he/she forces the states of the gray parts at selected values, nor observe these states.
- Due to the previous constraints, it may be impossible or it may require an intractable number of observations for the internal observer to discover the law governing the state evolution of a gray part by simply observing the states of the blue parts,
- With adequate pseudorandom generators, the behaviour of the blue parts as well as the behaviour of the whole system and the properties emerging in it may be indistinguishable with respect to the case of truly random generators.
- Similarly, it may be impossible to extrapolate the state of the

gray parts by observing the states of the blue parts.

- Knowing the state of the blue part of an entity and the states of the blue parts of the entities with which it interacts may not allow determining the next state of this blue part (since it also depends on the unobservable and unpredictable state of the gray part). However, it may be possible to restrict the set of possible values of this state (potential values). The unknown and unpredictable value of the state of the corresponding gray part will determine which particular value among these potential values will become the actual state of the blue part (actualization). Thus, an internal observer disposing a sufficient number of observations will not be able to extrapolate laws allowing her/him to predict the exact value provided by the blue part of an entity (or of a set of entities), but only laws allowing him to predict the set of possible values.

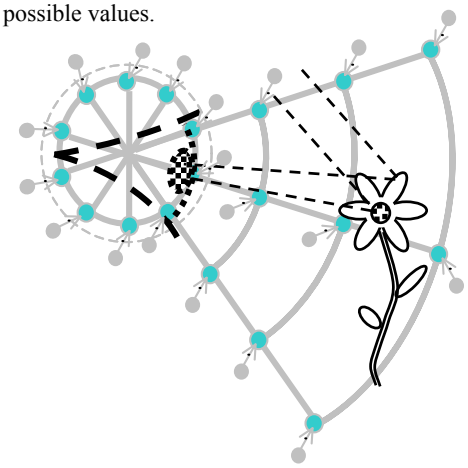


Figure 3. System closure versus internal observers

We observe that, though the system of figure 3 can be closed and possibly deterministic (e.g. when the gray parts are deterministic autonomous generators such as pseudorandom generators), for its internal observers the system will be seen as non closed and non deterministic, since some information not accessible to them (the values coming from the gray parts) is required to predict the evolution of the state of the system. For these observers, this information is a meta-information.

**3. Computational universe and emergence of space and time**

According to the ULKIO (ultimate limit of knowledge), as internal observers of the universe (i.e. composed of the same elementary particles as any other object of the universe), we can access information concerning the behaviour of elementary particles (i.e. their states and the way these states evomve) but not information concerning their veritable nature. As a consequence, we are not able to know if they are veritable particles immersed in a veritable space or if their behaviour is the result of a computation-like process such that:

- The evolution of the state of the particles is determined by "computation" rules described by analytical expressions identical to the ones describing the interaction laws of elementary particles.
- The determination of this evolution includes the evolution of the position of each particle. But this position does not correspond to a veritable position in a veritable space. Instead it is determined by the values that this process attributes to a position variable.

To illustrate this vision, let us consider that the meta-system making the computations, which engender the computational universe, is a cellular network (as shown in figure 4), composed of a set of cells (the meta-objects of the previous section) that compute the evolution of the states of elementary particles. Each cell uses its computational means to determine the next values of the state variables of a "particle" (including its position variable), as a function of the current values of these variables and of the current and/or past values of the state variables of the "particles" with which it interacts. The computation is performed following certain rules referred as interaction laws.

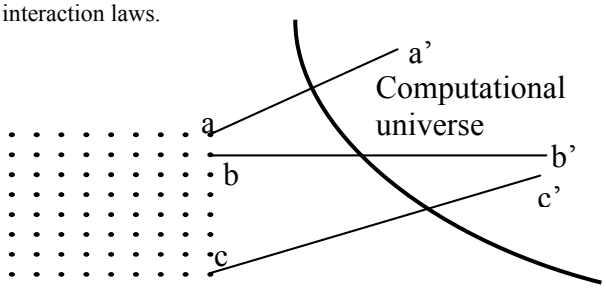



Figure 4. Cellular network and computational universe

It is worth noting that the distance between "particles" does not correspond to the distance of the corresponding cells in the cellular network, but to the numerical distance determined by the values of the position variables of the "particles". Thus, in figure 4, two cells a and b are close in the network, but their position variables have very different values. In this case, in the universe engendered by the cellular network, the corresponding "particles" a' and b' will have very distant positions. On the other hand, the cells b and c are very distant in the network but their position variables have very close values. As particles corresponding to distant cells in the network may interact, the network will dispose communication means to exchange information between distant cells as required by the interactions.

**3.1. Emergence of space**

In the illustration example given previously, the cells of the cellular network determine at each computation step the values of the position variables of the particles. If we consider that the values of the position variables represent positions in a multi-dimensional system of Cartesian co-ordinates (e.g. in a four-dimensional system) corresponding to a virtual multi-dimensional space with Euclidian structure, then, the values of the position variables determined by all the cells will lie in a subspace of this virtual space. This subspace will have a certain form (e.g. the form of a curve of three dimensions in a four-dimensional space, illustrated in figure 5 by the surface of a sphere of two dimensions). In this sense, the values of the position variables computed by the cells of the cellular network determine the form of the space engendered by this computation.

However, *this space is virtual for any observer external to this universe*, and she/he will be able to perceive its computational nature. On the other hand, in the computational universe, all structures (including the sensory and mental structures of its internal observers) are shaped by the states of the particles composing them and by their evolution. These states also determine the representations of the world emerging in these mental structures. If the computation rules used in a

computational universe are identical to the laws of interactions of a non-computational universe (in the sense that particles are immersed in a veritable space), then, the states of similar sets of particles will evolve similarly, creating similar structures and similar state configurations in both universes. Thus, *in the mental structures of the internal observers of the computational universe, the states representing the form of an object, the relative positions of a configuration of objects, etc., will be similar to the states representing the form of a similar object, the relative positions of a similar configuration of objects, etc ... in the mental structures of internal observers of the non-computational universe*.

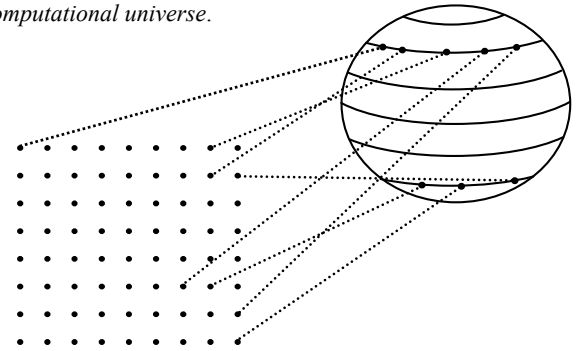


Figure 5. Emergence of space in a computational universe

Therefore, the representations of the "computational" objects and of the "computational" space emerging in states of the mental structures of the observers of the "computational" universe will be similar to the representation of the "veritable" objects and of the "veritable" space emerging in states of in the mental structures of the observers of the non-computational universe. Thus, *the perception of space and of objects emerging in the mental structures of the observers of the "computational" universe will be identical to the perception of space and of objects emerging in the mental structures of the observers of the non-computational universe. Therefore, nothing could differentiate these perceptions*. Thus*, the observers of the "computational" universe could believe that they live in a world composed of veritable objects immersed in a veritable space and nothing could prove to them that this is not true*. For the similar reason, we could not prove that we do not live in a world composed of "computational" objects immersed in a "computational" space rather than in a world composed of veritable objects immersed in a veritable space. This resolves the reality related questions R1 and R2 stated in section 1.2. The subsequent sections address certain *qualitative* questions concerning the time perceived by the internal observers of a system, which will be used to treat relativity and quantum mechanics in the *quantitative* part of this work (companion paper [4]).

### 3.2. Emergence of internal time

In this section we address a fundamental question concerning the nature of time:

- Is time an autonomous entity which has an existence per se and which paces the changes in the Universe? or
- Time does not exist a priori but it is a by-product of the evolution of the structures and processes of the universe (in the sense that this evolution creates the notion of time in our minds)?

We are interested about the internal time of a universe and more generally of a system, that is the time experienced by its internal observers. Such observers are agents, which anticipate the evolution of a process taking place in the system, by comparing it with other processes taking place in the system and selected as time references (clocks).

#### 3.2.1 A necessary and sufficient (qualitative) condition

Let us first discuss a sufficient condition that enables the emergence of internal time in any system (including a universe). We can find that the notion of the internal time (hereafter simply referred as time) is closely related to a well-known invariance, which seems to govern the evolution of our universe (we will see later that this invariance is related with the stability of the laws of physics). To describe this invariance let us consider a process H (for example the evolution of a clock) and k successive states $h_1$, $h_2$, $h_3$, …, $h_k$ of this process. Let us consider a second process G (for example the fall of a water drop) and k successive states $g_1$, $g_2$, $g_3$ … $g_k$ of G, which are respectively synchronous to the states $h_1$, $h_2$, $h_3$ … $h_k$ of H. Suppose that the processes H and G *take again place under exactly the same conditions* as before and that the state $g_1$ of G is synchronous to the state $h_1$ of H. Then, according to the above-mentioned invariance, the states $g_2$, $g_3$ … $g_k$ of G will be synchronous to the states $h_2$, $h_3$ … $h_k$ of H. In fact, this synchronism may not be perfect due to the quantum indeterminism, but will be verified with a high degree of accuracy if H and G describe the evolution of macroscopic systems. Based to this invariance, if we observe once two processes G and H, then, we will be able whenever these processes occur again to anticipate the evolution of process G by observing the evolution of H.

The above discussion is not limited to our universe but determines a *sufficient condition* for the emergence of time for observers that are part of any system or "universe", including a computational universe. Indeed, let us imagine a world in which there is always the same relationship between the paces of evolution of two processes, whenever and wherever these processes take place. In such a world we can speak about time, because:

i. We can choose a process as time reference, and
ii. After having observed once the correspondences between the different events of this process and the events of another process we can:
   - Use the reference process to predict the instant (event of the reference process) in which each event of the second process occurs.
   - Measure the duration of a process, by observing the events of the reference process in which the process under measurement starts and finishes.

As concerning the *necessary condition* for the emergence of (internal) time, let us imagine a world in which:

- Certain times the zebras are incomparably faster than the lion and certain times happens the opposite.
- A car being at a distance of several kilometres covers suddenly this distance in a fraction of a second and crushes us.
- The earth carries out hundreds of evolutions around the sun without your biological age being advanced, while several generations of people already passed, and suddenly you age of a hundred years in a fraction of a second.
- ……,

Let us imagine a world (or system) in which processes evolve arbitrarily the one with respect to the other, and thus, there is no invariant correlation between the paces of evolution of the different processes. The observation of the changes in such a world would not lead an intelligent observer to form the notion of time. Moreover, it would be improbable that such a universe will engender the intelligence. In fact, an intelligent being could not act by anticipation to protect itself from a natural phenomenon, because the speed of evolution of the phenomenon would be completely unpredictable; a herbivore could not escape a carnivore thanks to its speed, nor the carnivore catch the herbivore thanks to its speed, endurance, strategy, and power, because the relative speed of these animals would change in unpredictable way. For the same reason, the intelligence could not emerge, and in any case its existence would not have any sense: what would be the utility of intelligence if it could not anticipate any event?

In a system whose state evolves according to certain rules (laws), it is easy to check that the invariance of these laws is the necessary and sufficient condition that implies the invariance of the ratios of the paces of evolution of the processes that take place in this system. Thus, *the invariance of the laws governing the evolution of the system is the necessary and sufficient condition for the emergence of (internal) time*. In the case of a universe constituted of particles taking positions in a space (veritable or computational), this condition can be stated as: the laws that govern the evolution of the states of the particles are invariant (i.e. they are independent of the values of the position variables of the particles, and remain unchanged throughout the evolution of this universe). Indeed, in this case, the correspondence between the events of two processes will remain the same whenever and wherever these processes take place.

However, it is worth noting that this condition could be somehow relaxed without preventing the emergence of time in the mental structures of an internal observer. Indeed, the variation of the laws, which govern the evolution of a universe, will not prevent this emergence as long as this variation remains sufficiently weak or sufficiently slow to allow the prediction of the events with a sufficiently small margin of error.

*We can conclude that the internal time of a system (or a universe) is not an autonomous category but can exist if and only if the laws governing the evolution of the objects of the system are invariant.*

**3.2.2 Quantitative principle governing the internal time**

We will notice that the above invariance is not dependent on the very particular form of the laws governing the evolution of a system/universe, but on a generic principle of invariance of these laws. This invariance is the qualitative principle that underlies the emergence of time (the necessary and sufficient condition for its emergence). On the other hand, the particular correspondence between the states of two processes H and G is determined by the particular expressions describing the laws that govern the system/universe. For instance if process H is governed by an electromagnetic interaction having a given expression, and process G is governed by a gravitational interaction having another expression, the particular expressions of these laws will determine the relation of the pace of evolution of process G with respect to the pace of evolution of process H. Therefore, the particular form of the laws governing a universe will determine the quantitative manifestation of time (quantitative principle). It is also this form that determines the structure of time and consequently of space-time (e.g. Galileo or Lorentz transformations), as we show in a companion paper [4].

**3.2.3 Time versus meta-time**

In the previous sections we have seen that the invariance of the laws implies the emergence of internal time in a system/universe. *Nevertheless, the emergence of time presupposes that the state of this system evolves.* This assumption means that there is a cause that makes the system changing its states. We can call this cause engine of change or external time or meta-time. Whatever is the term we use the fact is that we tried to understand the nature of internal time by implicitly introducing a "meta-time". Since this "meta-time" is a metaphysical category (thus of inscrutable nature), we could conclude that this attempt has no sense since it replaces the question of the nature of time by the question of the inscrutable nature of a meta-time. This conclusion will be correct if time is a simple translation of meta-time. But this cannot be the case, since we have seen that time is determined both qualitatively and quantitatively by the laws governing the evolution of the states of the system. We can use the computational universe vision to make this fact more clear. This vision is very useful for illustrating the related ideas, as it allows us to give a simple and clear example of an external time of a system/universe and use it for illustrating the independence of the internal time from the external time. Let us consider that the universe is engendered by a (meta)computing system which is paced by a temporal dimension that we call meta-time. Let us suppose that this system is synchronous and its computations are paced by a meta-clock whose period corresponds to a duration T of meta-time. That is, T corresponds to the meta-time duration that the meta-system disposes for carrying out one step of computation (e.g. for exchanging information between meta-cells and computing the new states of the meta-cells). This step of computation will carry the minimal changes that can occur in the engendered universe. Therefore it will correspond to a minimal duration of time $t_h$ in this universe. Let us now consider that the clock period T is variable.

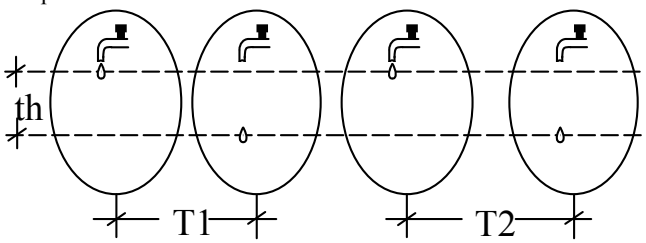



Figure 6. Independence between time and meta-time

This is illustrated in figure 6, where the period T of the meta-clock takes two different values $T_1$ and $T_2$ in two different cycles of computation. In this figure, the old and new states of each cycle are represented by the high and the low position of a water drop. Because at each clock cycle the meta-system carries out one cycle of computation, corresponding to the minimal time duration $t_h$ of the universe, then, the same time duration of the universe ($t_h$) will correspond to two different durations $T_1$ and $T_2$ of meta-time. Thus, stopping, decelerating, or accelerating the meta-clock will not have any influence on the time experienced by the observers that are part of the universe.

We can arrive in the similar conclusion if we consider continuous meta-time. Let P(T) be the function describing the evolution of an arbitrary process with the flow of a continuous meta-time T pacing the computational universe, and C(T) be the evolution of a process selected as clock in the computational universe. For convenience, process C(T) can be periodic, that is, C(T+D) = C(T), where D is a constant meta-time duration D. As CT is used as a clock in the computational universe, for the internal observers of this universe its period d will correspond to an internal-time unit. Figure 7 provides a simple illustration of a process P(T) (top of the figure), together with a periodic process selected as clock C(T) (bottom of the figure). In this figure, the clock period corresponds to a constant meta-time D as well as a constant internal-time d.

To imitate the acceleration, deceleration, and freezing of meta-time we can replace T by aT in all processes engendering the computational universe. For instance, making this substitution in the above two processes gives the functions P(aT) and C(aT). Then, for a = 1 all processes evolve as before (normal flow of meta-time). For a>1 all processes progress faster (meta-time acceleration). For a <1 all processes progress slower (meta-time deceleration). For, a = 0 all processes are frozen (meta-time freezing).

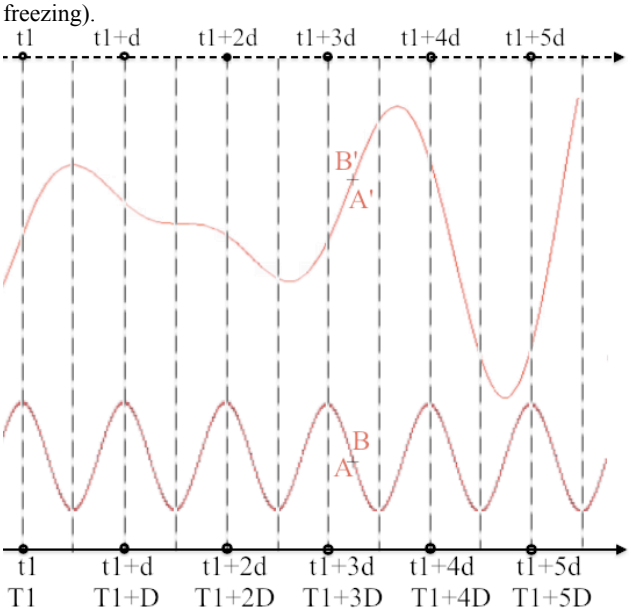



Figure 7. Process evolution in continuous meta-time

Figure 8 illustrates the effect of parameter *a* on the evolution of functions P(aT) and C(aT). In this illustration, on the left of point A we have a=2; between points A and B we have a = 0; on the right of point B we have a = 4/5. We remark that, figure 7 presents the points of view of both the external observers and the internal observers (equal distances on the horizontal axis represent equal durations for both the meta-time and the internal time. On the other hand figure 8 presents the point of view of the external observers only. This is because equal distances on the horizontal axis represent equal meta-time durations but not equal internal time durations. Indeed, equal internal-time durations must correspond to the same amount of clock periods. But in figure 8:

- The non-nil horizontal distance AB correspond to nil internal-time (nil evolution of the clock process);
- The same horizontal distances before point A and after point B represent different numbers of periods of the clock process: 2 clock periods before point A versus 4/5 clock periods after point B.

Then, to represent the view point of the internal observers we have to: joint points A and B for the process P(T) and A' and B' for the process C(T) (as no-internal time elapses between these points; expend horizontally (x2) the part of the figure before point A; and contract horizontally (by 4/5) the part of the figure after point B. Making these modifications, will result in exactly the curves in figure 7. This means that, with respect to the internal-time any arbitrary process will evolve identically as in figure 7. Thus, accelerating, deceleration or freezing the meta-time has no effect on the internal time.

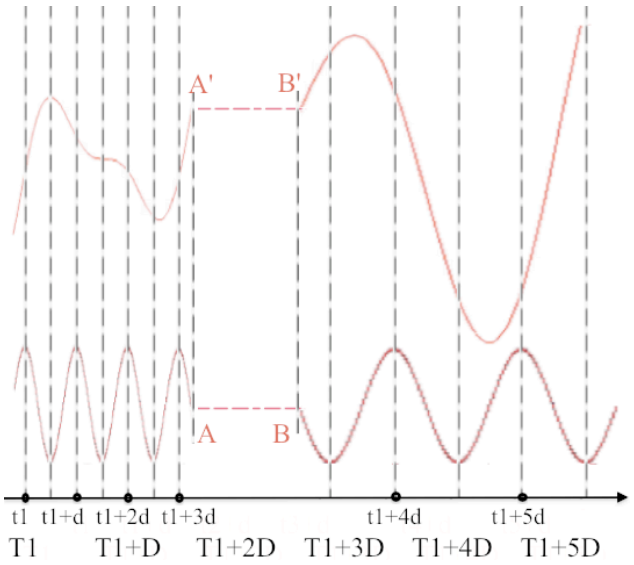



Figure 8. Effects of meta-time disturbances

The above results are generic as they consider any kind of function. A more specific analysis should consider that the next state of a physical system is generated from its present state P(T) and the present state I(T) of the physical systems with which it interacts. Then, the next state relationship can be written as P(T + DT) = F(P(T), I(T)). That is the state P of the system at instant T+DT is determined by its state P at instant T and by the state E at instant T of the systems with which it interacts. Then we raise the question about the value that we can take DT: can it be 0 or infinitesimal? Setting DT=0 gives P(T) = F(P(T), I(T)) giving always the present state P(T). Thus, DT>0. By setting T1 = T+DT we obtain P(T1) = F(P(T1- DT), I(T1-DT)). The state P of the system at an instant T1 could not be generated by the state P of the system at multiple instances. Thus, DT has to take a unique value. So, DT cannot be infinitesimal (an infinitesimal value is smaller than any given value - including the unique value of DT). Thus, similarly to any system in our universe, the computing meta-system can be represented as in figure 9, where the block implementing the next state function F(P(T), I(T)) has non-zero finite delay DT (please ignore for the moment the block ΔT drawn with dashed line).

Note that a system should start from a defined initial state. If the initial state is not defined the complete evolution of the system state may be undefined. Caution has to be taken when

considering the initialization of a system having non-zero delay. Indeed, to produce defined states, initially the state of the system has to be defined during a duration equal to this delay. Figure 10, shows a snapshot the evolution of the states P(T) and C(T) of two processes, where C(T) is selected as an internal clock in the computational universe. If the states of these processes are defined during an initial meta-time interval [T0, T0+DT), where DT is the delay of the next state function of these processes, then, theses functions will determine the states of the two processes in the interval [T0+DT, T0+2DT). Similarly from this interval will be generated the states in interval [T0+DT, T0+2DT), and so on.

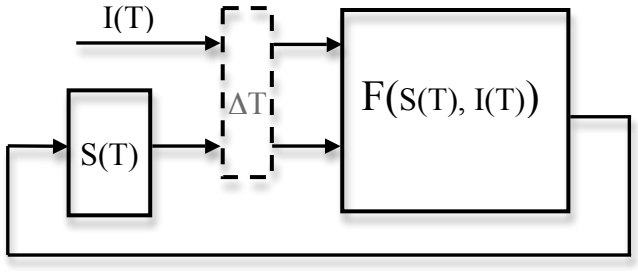



Figure 9. Next state function representation

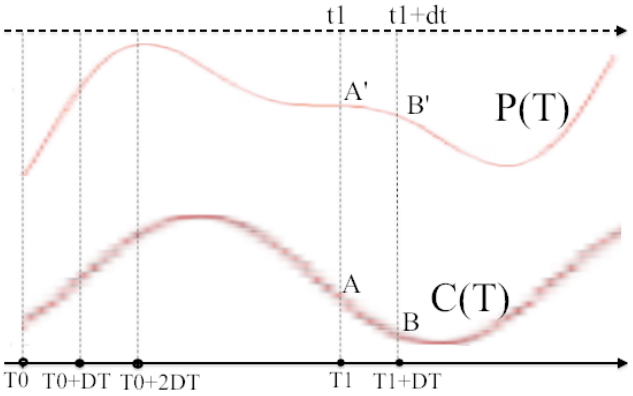



Figure 10. Effects of delay in next state computationn

Let us now consider the processes P(T) and C(T) shown in figure 10, in order to analyse what happens if we introduce on the inputs of function F(S(T), I(T)) a delay ΔT drawn in figure 9 with dashed lines. In this case, the delay required for computing the next state of the computational universe will become equal to DT+ΔT. Thus, the system will evolve by means of the next state function S(T+DT+ΔT) = F(S(T), I(T)) instead of S(T + DT) = F(S(T), I(T)), resulting in the curves shown in figure 11. The consequence is that if two states A and B of process C(T) and A' and B' of process P(T) are separated by a meta-time duration DT in figure 10, the same states will be separated by a meta-time duration DT+ΔT in figure 11. Thus, from the meta-time perspective, introducing the delay ΔT decelerates all processes of the computational universe by a factor equal to DT/(DT+ΔT). Thus, figure 11 can be obtained by expending horizontally figure 10 by a factor (DT+ΔT)/DT. However, the time experienced by the internal observers of the computational universe is determined by the relations of the paces of evolution of the processes taking place in the computational universe (e.g. the pace of evolution of process P(T) in comparison with the pace of evolution of process C(T) selected as internal clock). Similarly to figures 7 and 8, to take into account the deceleration induced on the clock process C(T) by the meta-time delay ΔT, we have to contract horizontally figure 11 by the factor DT/(DT+ΔT). Making this contraction we obtain exactly the curves of figure 10. Thus, adding the delay ΔT does not affect the time experienced by the internal observers of the computational universe. *This, simple observation is essential in order to show in the companion paper [4] that a computational model excluding instantaneous communication can engender a computational universe in which internal observers experience quantum non-locality.*

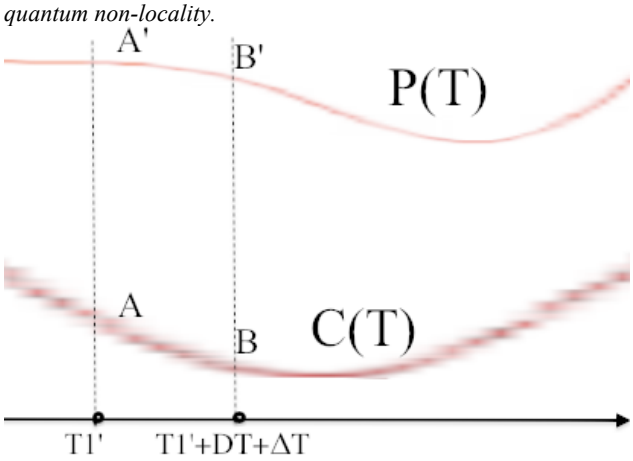



Figure 11. Adding extra delay in next state computation

**Conclusion**

We have shown that for any observer that is part of a system/universe), there is an ultimate limit of knowledge (referred as ULKIO). This limit allows the internal observers of a system/universe to receive information concerning the behaviour of the elementary entities/particles composing the system/universe, but by no means information concerning their veritable nature. The consequence is that, as observers that are part of the Universe we have no means allowing us to distinguish whether our perceptions correspond to a universe composed of veritable particles immersed in a veritable space or to a universe engendered by a computation-like process. We further discuss the emergence of space and time, and we show that their structure is determined by the rules used to compute the states of the elementary particles as well as the actions of measurements performed by the internal observers of the system/universe. These qualitative results are used in a companion paper [4] in order to develop the quantitative part of this work, treating the computational Universe vision of special relativity and of quantum mechanics.

**References**

[1] Konrad Zuse, Rechnender Raum. Braunschweig: Friedrich Vieweg & Sohn. 70 pp, 1969.

[2] Konrad Zuse, Calculating Space, MIT Technical Translation AZT-70-164-GEMIT, (MIT (Project MAC), Cambridge, Mass. 02139, 1970.

[3] Schmidhuber,A Computer Scientist's View of Life, the Universe, and Everything. LNCS 201-288, Springer, 1997

[4] Nicolaidis Michael "The scandal of the computational universe; Second part: relativity and quantum mechanics", 6th AISB Symposium on Computing and Philosophy

# The scandal of the computational universe

## Second part: relativity and quantum mechanics

Miclael Nicolaidis
TIMA Lab (CNRS, Grenoble INP, UJF)

**Abstract:** This work discusses a computational universe vision. In a companion paper, we describe a simple computation model adopted in this vision, as well as the contradictions raised when we use such a simple model for reproducing the observable behaviour of the universe as described by the theories of physics. Then, in order to resolve these contradictions, we introduce the concept of the internal observer of a system and we discuss the emergence of space and time for such observers. This discussion is qualitative and creates a framework over which we can construct our computational universe vision at the quantitative level, in order to integrate the theories of physics in our computational universe vision. The goal of the present paper is to develop the computational universe versions of special relativity and quantum mechanics.

**Keywords:** pan-computationalism, computational universe, special relativity, quantum mechanics.

### 1. Introduction

The goal of the present paper is to complete a computational universe vision proposed in a companion paper [1], by developing the computational universe versions of special relativity and quantum mechanics. In the companion paper we argued that adopting a "computation" model that is a carbon copy of these theories has no interest, as it will not bring new light in our vision of the universe. Thus, we adopted a very simple computation model. However, using such a simple model for reproducing the behaviour of physical processes, as described by the theories of physics raises several contradictions, such as:

- The computations in our computational universe model are rated by a unique time variable (e.g. a meta-clock). This is in contradiction with special relativity, where there is no fundamental time but as many times as inertial frames.
- In special relativity all inertial frames have equal stance, and the 4D reality of special relativity encompasses the state of natural systems in all inertial frames. But our model uses a single set of state variables, which computes the states of a system in only one inertial frame.
- Numerous observables (position, momentum, energy, …) have infinite number of eigenvalues, resulting in infinite number of quantum superposition values. This looks incompatible with our model, where the state of each elementary particle has to be described by a finite number of variables.
- Our computation model excludes instantaneous communication. This seems inconsistent with quantum non-locality.

While the computation model we adopted seems incompatible with the theories of physics, we show in this paper that this incompatibility does not concern the observable behaviour of physical systems described by these theories but their interpretation. To proceed, in the companion paper [1] we introduce the concept of observers that are part of the system/universe (internal observers). Such observers are composed of the same elementary entities (e.g. particles) as any other object of the system, and use measurement means also composed of such entities. Then, we have shown that the time perceived by such observers (the internal time of the system), is not related with the time pacing the evolution of the states of the system (e.g. the meta-clock in the case of synchronous digital computing), but is exclusively determined by the rules (interaction laws) computing the evolution of the states of the elementary entities/particles composing the system. In addition, we show that these rules also determine the structure of space perceived by the internal observers of the system.

Based on these qualitative results [1], the goal of the present article is to determine computation rules (representing the laws of interactions), which engender internal space and time having the 4D space-time structure of special relativity. Also, based in the distinction of the internal from the external/meta time, our second goal is to show that quantum non-locality can be reproduced for the internal observers, even if the computation model does not support instantaneous communication. Finally, using a computational system illustrated in the companion paper (figure 3 [1]), are third goal is to propose a computational model of quantum systems, which engenders the same observable behaviour as the one described by quantum mechanics, but without using infinite superposition states, as would be required by the current interpretation of quantum mechanics for observables having infinite number of eigenvalues.

### 2. Emergence of Relativistic Space-Time

#### 2.1 Preliminaries

According to the special relativity, space and time are imbricated in an inherent 4D structure described by he Lorentz transformations. However, in the computational universe, where space and time do not pre-exist but are engendered by a computation-like process, their structure cannot be an inherent property but has to emerge in the computation engendering this universe. As discussed in the companion paper [1], this space-time structure will be determined by the form of the laws of interactions, and will emerge in the measurements of the internal observers of the computational universe. The characteristic of these measurements is that they use measurement means composed of the same particles as all other objects of the computational universe. This means that the evolution of the states of these particles is determined by the same computation rules (interaction laws) as those determining the evolution of particles engendering the computational universe. Then, to develop a computational universe conforming special relativity, we have to:

i. Determine the form of computation rules (interaction laws) that could engender space-time conforming

Lorentz transformations.

ii. Determine the measurement means (length units, time units and synchronized clocks) of the internal observers.

iii. Show that, under the laws of point i and the measurement means of point ii, the measurements performed in any inertial frames obey Lorentz transformations.

As discussed in the companion paper [1], the computations are paced by a unique time reference (to be referred hereafter as fundamental time $T_0$), and they determine a single set of states for the particles, which then can correspond to a single inertial frame (to be referred hereafter as fundamental inertial frame $S_0$). So, the computation rules (interaction laws) determining the evolution of these states should be expressed in the inertial frame $S_0$. Thus, in the following we determine the family of computation rules (macroscopic [1] interaction laws) engendering the 4D space-time structure of special relativity (which is described by Lorentz transformations). In order to be generic (i.e. to cover all possible computation rules (macroscopic interaction laws) able to engender this space-time structure, we will not restrict our analysis in a specific computation rule (macroscopic interaction law). Instead we will provide a constraint that the computation rules (macroscopic interaction laws) have to obey. This constraint will determine the largest class of laws engendering the Lorentz transformations. This class of laws will be determined in the following manner:

a. Each specific macroscopic interaction law will be defined by the analytical expression of the force (or equivalently of the acceleration) that a source particle at rest in the fundamental frame $S_0$ induces to a destination particle (to be referred hereafter as the *basic form of the interaction law*). For convenience we are using the expression of the acceleration instead of the force. At this level, we do not introduce any constraint concerning the form of the interaction. That is, its analytical expression can be proportional to the inverse square of the distance separating the interacting particles (as it is the case with our familiar macroscopic interactions), but can also be more exotic ones (like polynomial, exponential, having intensities increasing or decreasing with distance, or whatever else).

b. From point a, the form of each interaction is freely determined for source particles at rest in $S_0$. This form partially determines the interaction. Then, to completely determine it, we need to determine its form for source particles having any speed in $S_0$. In order to engender a relativistic-space time, this part cannot be determined freely; instead it will have to obey a condition that we will describe next. We will refer to this condition as RCA [2] (relativistic constraint of accelerations). We show [2] that RCA represents the necessary and sufficient condition for generating the 4D space-time structure of special relativity. Thus, the relativistic constraint of accelerations (RCA) described bellow determines the largest family of computation rules (interaction laws) able to engender this structure.

**2.2 Relativistic Constraint of Accelerations (RCA)**

Modern theories of physics consider that distant objects do not interact instantaneously but the interaction is propagated in space with a finite speed equal to that of light. Thus, an object 2 being on position $\overrightarrow{P2}$ at an instant $t_2$ is not subject to the effect of the interaction of an object 1 originated from the position of this object at the present instance $t_2$, but from its position $\overrightarrow{P1}$ at a past instant $t_1$ such that $c(t_2 - t_1) = r$ (where r is the distance between $\overrightarrow{P1}$ and $\overrightarrow{P2}$). For easier reference, we will call the instant $t_1$ and the position $\overrightarrow{P1}$, satisfying this relationship with an instant-position pair ($t_2$, $\overrightarrow{P2}$), the delayed instant and delayed position with respect to the pair ($t_2$, $\overrightarrow{P2}$). The analytical expression of RCA, described below, will also incorporate this principle.

Let us consider a velocity vector $\bar{v}$ and select 3 axes X, Y, Z, which are at rest in the inertial frame $S_0$ and perpendicular to each other, and such that X is parallel to the direction of $\bar{v}$. Then, these axes constitute a Cartesian referential in the inertial frame $S_0$. Let us consider two particles 1 and 2 such that:

- The position and velocity of particle 2 at an instant $t_2$ are respectively $\overrightarrow{P2}$ and $\overline{u2} = (u_{2x}, u_{2y}, u_{2z})$,
- The position and velocity of particle 1 at an instant $t_1$, with $\Delta t = t_2 - t_1 > 0$, are respectively $\overrightarrow{P1}$ and $\overline{u1} = (u_{1x}, u_{1y}, u_{1z})$,
- The norm r of the vector $\vec{r} = (r_x, r_y, r_z)$ which connects the positions $\overrightarrow{P1}$ and $\overrightarrow{P2}$ (i.e. the distance between

[1] Note that in the context of the theory of relativity we don't deal with quantum interaction laws but with the macroscopic expressions of the interaction laws.

[2] Special relativity 4D space-time structure is determined by the transformations (Lorentz) of time and space coordinates when we change inertial frames. Then, the idea beyond RCA is to modify the expression of interactions for source particles in a manner that the dimensions of objects, the distances between objects and the pace of evolution of processes change in a manner that all spatial and temporal measurements are compatible with Lorentz transformations, thus creating a computational universe which has the same observable behaviour as the one described by special relativity.

positions $\overrightarrow{P1}$ and $\overrightarrow{P2}$ ) satisfy the relation[3] $r = c\Delta t$.

Let $\overline{a_2} = (a_{2x}, a_{2y}, a_{2z})$ be the acceleration induced at instant $t_2$ on particle 2 due to its interaction with particle 1. If we have two particles 1' and 2' identical to the particles 1 and 2 and such that:

- At an instant $t_2$' the position and velocity of particle 2' are respectively $\overrightarrow{P2'}$ and $\overline{u_{2'}} = (\frac{u_{2x} + v}{1 + v \cdot u_{2x}/c^2}, \frac{u_{2y}/\gamma_v}{1 + v \cdot u_{2x}/c^2}, \frac{u_{2z}/\gamma_v}{1 + v \cdot u_{2x}/c^2})$
  At an instant $t_1' = t_2' - \frac{r'}{c}$ with $r' = \sqrt{\left[(\gamma_v r_x + \gamma_v v \Delta t)^2 + r_y^2 + r_z^2\right]}$, the particle 1' is at a position $\overrightarrow{P1'}$ such that the vector connecting positions $\overrightarrow{P1'}$ and $\overrightarrow{P2'}$ is equal to $\vec{r'} = (\gamma_v r_x + \gamma_v v \frac{r}{c}, r_y, r_z)$ (or, in equivalent manner, $\vec{r'} = (\gamma_v r_x + \gamma_v v \Delta t, r_y, r_z)$. The velocity of particle 1' at instant $t_1$' is $\overline{u_{1'}} = \left(\frac{u_{1x} + v}{1 + v \cdot u_{1x}/c^2}, \frac{u_{1y}/\gamma_v}{1 + v \cdot u_{1x}/c^2}, \frac{u_{1z}/\gamma_v}{1 + v \cdot u_{1x}/c^2}\right)$, then, we will say that the laws of interactions satisfy the Relativistic Constraint of Acceleration (RCA) if and only if the components of the acceleration induced on particle 2' at instant $t_2$' due of its interaction with particle 1' are:

$$a_{2x'} = \frac{a_{2x}}{\gamma_v^3 (1 + v \cdot u_{2x}/c^2)^3}$$

$$a_{2y'} = \frac{a_{2y}}{\gamma_v^2 (1 + v \cdot u_{2x}/c^2)^2} - \frac{(v \cdot u_{2y}/c^2) a_{2x}}{\gamma_v^2 (1 + v \cdot u_{2x}/c^2)^3},$$

$$a_{2z'} = \frac{a_{2z}}{\gamma_v^2 (1 + v \cdot u_{2x}/c^2)^2} - \frac{(v \cdot u_{2z}/c^2) a_{2x}}{\gamma_v^2 (1 + v \cdot u_{2x}/c^2)^3}.$$

The RCA, described above for two objects is easily generalized to any number of objects [2].

As said in section 2.1 point a, the acceleration induced on a particle by a source particle that is at rest in $S_0$ can be described by any analytical expression if the source particle is at rest in $S_0$ (*basic form of the interaction law*). Then, the acceleration induced when the source particle is not at rest in $S_0$, is obtained by applying RCA on the *basic form of the interaction law*. We observe that *the conditions describing RCA do not concern the structure of space-time, but only the rules (interaction laws) used to compute the evolution of the particles* composing the computational universe.

### 2.3. Internal observers and measurement means

Let us now consider *the conditions that have to be verified by the measurements* in order to determine the structure of space and time in a coherent way.

***1st condition:*** The structure of space and time of the computational universe will be determined by the results of the measurements of the lengths of objects; of the distances between objects; and of the durations of the processes taking place in this universe. Thus, *the measurements concern objects composed of particles obeying RCA, and processes engendered by the evolution of the states of sets of particles obeying RCA*.

***2nd condition:*** As discussed in the introduction (see also the companion paper [1]), the above measurements are performed by the internal observers of this universe, which will use *measurement means constituted by the same elementary particles as the ones composing all the structures of the computational universe (i.e. particles obeying RCA)*.

***3rd condition:*** All *measurements have to use the same or identical objects as length units and the same or identical objects as clocks.*

***4th condition:*** In addition to unit lengths and clocks, performing time and length measurements requires a principle for synchronizing any two distant clocks C1 and C2 at rest in any inertial frame S. This synchronization can be done by launching towards C1 and C2, and from the middle of the distance separating them, two objects H1 and H2 having equal speeds (synchronization objects). But to determine the equality of these speeds we need to make measurements and thus to dispose in advance synchronized clocks placed in distant positions. Special relativity overcomes this deadlock by using light beans, thanks to the postulate of the invariability of light speed in all inertial frames and directions. However, in the computational universe, all properties (including the invariability of the speed of light) have to emerge from the computation of the evolution of the states of the particles. That is, no postulates exist except that the computation rules obey RCA. Then, as stated in section 2.2 (footnote 3), RCA introduces the value c of the light speed only in the fundamental inertial frame $S_0$. Hence, as any other inertial frame S moves with respect to $S_0$, an invariability of the speed of the light cannot be taken as granted. Thus, we cannot use light beans to synchronize distant clocks in any inertial frame S. To cope with this issue, we have

[3] The relations $Dt = t_2 - t_1 > 0$, and $r = cDt$, introduce in RCA the fact that the interactions are propagated in space with the speed c of the light. These relations do not introduce from the "back door" the postulate of of light speed invariance in all inertial frames, which is an inherent property of the space-time structure of special relativity. Indeed, the RCA is a property defined in the fundamental inertial frame $S_0$. Thus, the relations $Dt = t_2 - t_1 > 0$, and $r = cDt$ introduce the speed c only in $S_0$. This is also valid for the use of c in the other expressions used to describe RCA.

introduced an *original* principle for synchronizing distant clocks [2]. This principle *uses as synchronization objects two identical clocks H1 and H2. These clocks are placed in the middle M of the distance separating C1 and C2 and are reset to 0. Then, H1 and H2 are launched from M towards C1 and C2. C1 is set to 0 when H1 cross its position and the time indication of H1 is registered. Similarly C2 is set to 0 when H2 cross its position and the time indication of H2 is registered. If the two registered times are identical, C1 and C2 are considered to be correctly synchronized.*

Conditions 1, 2, 3 and 4 provide all we need for performing measurements and analyzing them in order to determine the structure of the internal space and time engendered by the computational universe. *We show that measurements performed in any inertial frames obey Lorentz transformations. The formal proofs are not presented here for space reasons, but can be found in [2]. Consequently, a computational universe employing computation rules obeying RCA engenders a space-time structure compliant with special relativity.*

**2.4 Conclusions on special relativity**

In a companion paper [1] we introduce a computational model in which the history of the universe is engendered by a computation that determines the evolution of the states of the elementary particles. In such a universe, the space and time perceived by an internal observer cannot pre-exist but has to be engendered by the computation. In particular, the perceived space emerges as a by-product of the values allocated by the computation to the position variables of the particles, and the perceived time emerges as a by-product of the paces of evolution of the processes taking place in the computational universe. As the position variables and the paces of evolution of processes are both determined by the form of the computation rules (interaction laws), these laws will determine the structure of space-time. This seems incompatible with special relativity, since in this theory space and time are imbricated in an *inherent* 4D structure described by he Lorentz transformations. However, we have shown that Lorentz transformations can emerge in the computational universe as a consequence of the computation rules (interaction laws). Thus, for its internal observers, the proposed computational universe model engenders the same observable behaviour as the one described by the theory of special relativity. As a consequence, the above-mentioned incompatibility does not concern the observable behaviour of physical systems described by this theory, but only its interpretation attributing the Lorentz transformations to the geometry of space-time.

**3. Quantum Mechanics and Stochastic Computations**

**3.1 Preliminaries**

This section treats the question of a computational model able to reproduce the non-deterministic behaviour of quantum systems. Let us start with a quick reminder of the basic concepts of quantum mechanics:

i. In quantum mechanics the state of a quantum system is described by a function $\psi$ (the wave function). This function is determined by solving Schrödinger's equation or one of its relativistic counterparts (Klein-Gordon or Dirac). Then, based to this function a plurality of values $\alpha_1, \alpha_2, \ldots$ is determined for each physical observable; such as position, translational momentum, orbital angular momentum, spin, total angular momentum, energy, etc., together with associated probabilities $P_1, P_2, \ldots$, following the rules:
ii. At each observable A is associated a Hermitian operator $\hat{A}$ (whose form has a certain relation with the expression of the observable in non-quantum physics, i.e. Newtonian or relativistic).
iii. The values $\alpha_1, \alpha_2, \ldots$ related to an observable A are the eigenvalues of the operator $\hat{A}$ of this observable.
iv. The associated probabilities are $P_1 = |c_1|^2$, $P_2 = |c_2|^2$, …, with $c_i = \langle\psi_i|\psi\rangle \quad \forall\, i \in \{1, 2, \ldots\}$, the inner product of eigenfunction $\psi_i$ of $\hat{A}$ and the wave function $\psi$.
v. If the observable A is measured, then, the result will be one of the eigenvalues $\alpha_1, \alpha_2, \ldots$ with a probability equal respectively to $P_1 = |c_1|^2$, $P_2 = |c_2|^2$, ….
vi. After the measurement $\psi$ collapses to the eigenfunction $\psi_i$ corresponding to the eigenvalue $\alpha_i$ obtained as result of the measurement. Since $\psi$ becomes equal to the eigenfunction $\psi_i$ a subsequent measurement of the observable A will give as result the corresponding eigenvalue $\alpha_i$ with probability equal to 1 (since, in this case $c_i = \langle\psi_i|\psi\rangle = \langle\psi_i|\psi_i\rangle = 1$)

Before the measurement the system is said to be in coherence. The measurement destroys this state and after measurement the system is said to be in decoherence.

The above rules concern observables having discrete spectrum. Similar rules are used in the case of observables having continuous spectrum, such as position or momentum. But in this case the statistical distribution is described by a probability density function P(a) (e.g. $|\psi|^2$ gives the probability density function of position).

*These are the strictly necessary concepts and mathematical formalism required to describe the observable behaviour of quantum systems*. But the state of coherence is "strange", since we cannot allocate to the observable a unique value as in the macroscopic world. To give it a sense, this state was ***interpreted*** by considering that the observable is in superposition on a plurality of values. Figure 1 illustrates the concept of superposition by

showing the observable A to be simultaneously on several values $\alpha_1, \alpha_2, \ldots, \alpha_n, \ldots$, to which correspond certain probabilities $p_1, p_2, \ldots, p_n, \ldots$

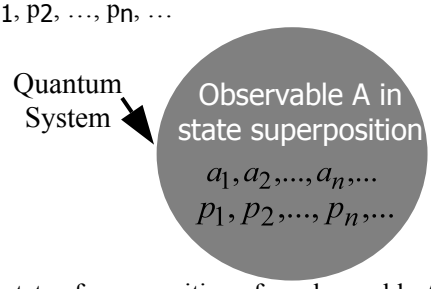



Figure 1. The state of superposition of an observable A.

In the context of a computational universe, representing this states requires as many numerical variables as the number of eigenvalues $\alpha_1, \alpha_2, \ldots, \alpha_n$. However, observables of continuum spectrum, like position and momentum (but also certain observables of discrete spectrum like energy), have infinite number of eigenvalues. This will imply using infinite number of numerical variables (corresponding to infinite memory, and infinite computing power for treating them). This is incompatible with our goal to develop simple computation model using finite number of variables. As a matter of fact, we are looking for a computational model, such that:

- It encompasses the *strictly necessary concepts and mathematical formalism required for producing the observable behaviour of a quantum system (described earlier in points i, ii, iii, iv, and v).*
- It does not make use of the superposition concept.

To further motivate the elimination of quantum superposition from our model, we remark that:

- As noted earlier, the state of superposition is not among the strictly necessary concepts for describing the observable behaviour of quantum systems.
- The concept of superposition describes a metaphysical state, since it is not possible to effectively observe the realization of this state (i.e. to observe that the system is indeed simultaneously in several states). In fact, any attempt for doing so (a measurement) leads to decoherence and provides as result a single value.
- Physics consider that our universe is composed of objects immersed in a veritable space and evolving with the flow of a veritable time (merged in space-time). In this vision, objects exist only within space. Hence, a physical object must be at any time within space. Thus, during the quantum state of coherence, where an object has not a precise position in space, this vision still requires to place it within space. This leads to the "strange" interpretation of the state of coherence, which claims that the object is in superposition over a plurality of space positions. However, in the computational universe context, there is no veritable space, instead, space is engendered as a by-product of the values that the computation allocates to the position variables of the elementary entities/particles. Thus, during the period where an object is in coherence and does not have an observable position, the computation does not need to generate value(s) for the position variable. We can apply this idea to all physical observables and eliminate the requirement to use as many numerical variables as the number of eigenvalues of the physical observables.

A last remark useful for creating our computational model for quantum mechanics is that, from the mathematical point of view, the interpretation of quantum mechanics based on the superposition concept is complete as it provides all the rules needed to determine the statistical distributions of the observables. However, in nature, each measurement of an observable produces a particular value among the ones allowed by its statistical distribution. That is, during each particular measurement there is something that selects a particular value among the possible ones and with the required probability. The interpretation based on the superposition concept does not provide means performing this selection. Thus, from the physical point of view the superposition interpretation is incomplete. This incompleteness is not allowed in the context of the computational universe, because the computation should not just predict possible values but effectively produce exact values to the variable coding a physical quantity[4] each time this quantity has to provide an observable value.

The above discussions leads to the stochastic computational model described next.

### 3.2. A Computational Model for Quantum Systems

In this section we propose a stochastic computational model of quantum systems, which reproduces their observable behaviour as described by quantum mechanics (rules i though to vi of section 3.1). Such a model considers that the behaviour of a quantum system is the result of a computation-like process, which produces the observable states of the system (the results of measurements). Except the results of measurements, any internal state involved in this computation is not observable[5]. Thus, we can consider that the computation is taking place in a meta-object: nothing concerning this object is observable except the states it returns when a measurement is performed. We call such an object a computing meta-object (CMO). In the stochastic computational model illustrated in figure 3, the CMO

[4] In this sentence, to avoid using side by side the word "observable" with two different meanings, we employ the term « physical quantity » instead of the equivalent term « physical observable » (or simply "observable") used in the rest of the paper.

[5] This is also the case for the other interpretations of quantum mechanics, where only the results of measurements are observables (e.g. the wave function and the states in superposition are not observable).

produces the observable behaviour of quantum systems by means of a computation-like process, which uses deterministic functions to transform a stochastic signal $w_a$ into a signal $a$ which during each measurement of an observable provides the result of the measurement.[6] As the stochastic signal $w_a$ is not observable, it has to be considered as a meta-signal. Note that this signal cannot be considered as a hidden variable. This is because, as $w_a$ is stochastic, there is no cause-effect relation between: on the one hand the values it will take in the future, and on the other hand its present and past values and the present and past states of the system and its environment. Thus, the outcomes of future measurements cannot be determined by the present and past states of signal $w_a$, of the quantum system and of its environment (whereas this would be the case for a hidden variable).

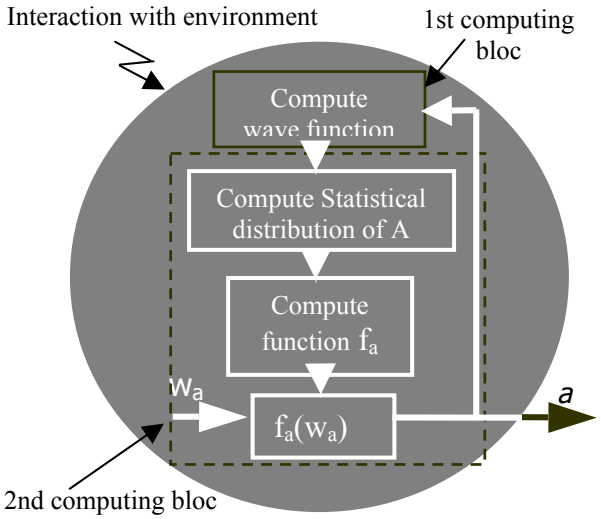



Figure 3. Computing meta-object (CMO)

The above discussions leads to the stochastic computational model described next. As shown in Fig. 3, a CMO comprises 2 computing blocks, aimed at implementing the strictly necessary concepts and mathematical formalism reported in section 3.2, points i through to vi, plus the mechanism producing effective values in case of measurement.

The first bloc computes the wave function by resolving a differential equation (Schrödinger, Klein-Gordon, Dirac) for a function of potential corresponding to the CMO environment (ultimately determined by the states of the other CMOs with which it interacts). The wave function represents at any time the state of the CMO.

The second bloc performs computation only during "measurements": each time an observable A is measured this block computes a deterministic function $f_a$ and then uses this function to transform the stochastic meta-signal $w_a$ into a stochastic signal $a$ which provides the result of the measurement of observable A. As the function $f_a$ transforms a unique value (the current value of $w_a$) to produce the current value of $a$, signal $a$ will bring a unique value (during measurements), or no value (outside measurements).

The function $f_a$ is computed in a manner that the statistical distribution (values $\alpha_1$, $\alpha_2$, … and the corresponding probabilities $p_1$, $p_2$, …) of signal $a$ produced by the transformation $a = f_a(w_a)$ is identical to the statistical distribution of the corresponding observable A determined by the rules of quantum mechanics. This computation will be possible if, for any statistical distribution $p(a)$ of an observable A, *it exists a deterministic function* $f_a$ *which transforms the stochastic signal* $w_a$ *to a stochastic signal* $a$ *that has the statistical distribution* $p(a)$. In a research report [3] we show that for a signal $w_a$ having any arbitrary but given statistical distribution of continuous spectrum $s(w_a)$ and for any statistical distribution $p(a)$, it always exists a function $f_a$ which produces a signal $a = f_a(w_a)$ whose statistical distribution is equal to $p(a)$. *The analytical description of function* $f_a$ *is also derived* [3]. This result is valid for statistical distributions $p(a)$ of discrete spectrum as well as of continuous spectrum. This result is easily generalised to the case of vectorial observables (like for instance) the position $\bar{r}$ or the momentum $\bar{p}$).

The measurement induces decoherence: the observable A takes a precise value equal to the value of signal $a$ (the result of the measurement). Accordingly, the first computing bloc computes a new wave function which is compatible with this particular state of observable A (that is the wave function becomes equal to the eigenfunction associated to the eigenvalue obtained on signal $a$ during the measurement). This influence of the value of signal $a$ on the computation of the wave function is represented in figure 3 by an arrow, which brings the value of signal $a$ to the input of the first computing bloc.

Thanks to the results in [3] guarantying the existence of function $f_a$ and providing its analytical description, *all steps related to the above description are computable by means of analytical functions*. Also, from the above discussion, these computations reproduce the mathematical formalism reported in section 3.2, points i through to vi. Thus, the proposed computation model produces the observable behaviour of quantum systems described by quantum mechanics. One of the consequences of these results is that quantum systems cannot exhibit the computing power of a truly parallel computer performing as many parallel computations as the

[6] This model of quantum objects is compliant to the system model presented in figure 3 of the companion paper [1]. The CMO corresponds to the light blue circles of this figure, and the signal $w_a$, corresponds to the gray circles.

number of the states of n q-bits states (i.e. $2^n$). This explains with very generic arguments valid for all quantum algorithms, why these algorithms do not exhibit the computing power of truly parallel computers [4].

### 3.3 Quantum Entanglement

As our computing model excludes instantaneous communication between various computing modules [1], it seems that this model could not support quantum entanglement (where a particle can correlate instantaneously its state with the result of a measurement performed over its entangled counterpart). Indeed, for a computational universe comprising huge amounts of elementary entities/particles, to minimize the time required for computing their states, parallel computation will be preferable (e.g. a cellular network where each cell continuously produces the state of a particle). However, as the cellular network does not support instantaneous communication between cells, a particle could not correlate instantaneously its state with the result of a measurement performed over its entangled counterpart.

To overcome this apparent contradiction, we can use the distinction between the internal time emerging in the computational universe and the external time (meta-time) pacing the computation [1]. For *the shake of illustration*, let us consider an example using means available in our universe. In this example, cells exchange information by means of Hertzian communication, which employs: an identification value (ID) unique to each cell a particle; an entanglement variable EV internal to each cell; and a mechanism of Hertzian emission/reception. This mechanism uses during emission a modulation frequency equal to the identification value ID of the cell and during reception a demodulation frequency equal to the value of the entanglement variable EV of the cell, as illustrated in figure 3. In this figure, particles pi and pj are entangled. At the instant of entanglement, particle pi assigns the identification value $ID_j$ of particle pj to its entanglement variable $EV_i$ ($EV_i = ID_j$). Similarly, particle pj assigns the identification value $ID_i$ of particle pi to its entanglement variable $EV_j$ ($EV_j = ID_i$). Then, if observable A of particle pi is measured, pi emits information reporting that a measurement happened as well as the result of the measurement. Particle pj uses as demodulation frequency the value of its entanglement variable ($EV_j = ID_i$). Thus, it immediately receives this information and adapts its state to the measurement. In a similar manner, particle pi can immediately adapt its state to a measurement performed on particle pj. Therefore, this computation reproduces the behaviour of entangled particles and combined with the computation illustrated in figure 3, provides a model (illustrated in figure 4) that produces a behaviour identical to the one described by quantum mechanics.

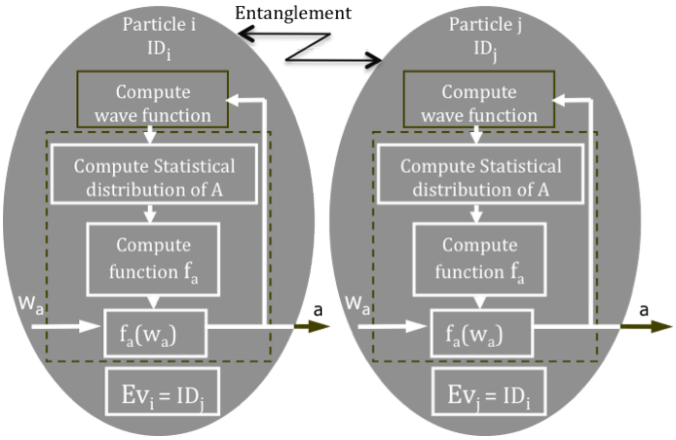



Figure 4. Computational model for entangled particles

However, as the communication is not instantaneous, the above mechanism requires some meta-time duration to transmit "information" from the cell computing the particle that undergoes measurement to the cell computing the entangled counterpart of the first particle. In the discrete time model presented in [1] and using a meta-clock of period T, integrating this duration in the meta-time period T will increase the meta-time duration of each computation step. However, as shown in [1], stopping, decelerating, or accelerating the meta-clock does not have any influence on the time experienced by the observers that are part of the universe (internal time). Therefore, the meta-time duration required for this transmission does not affect the internal time of the computational universe. Hence, quantum entanglement can emerge in the computational universe, even if the computation does not support instantaneous communication.

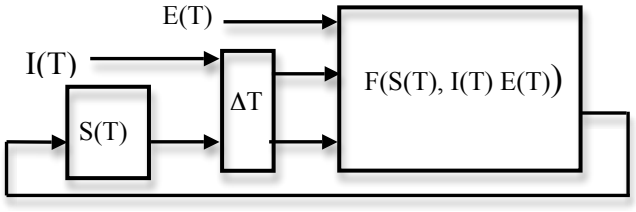



Figure 5. Next state computation for quantum non-locality

Concerning the model of continuous time, we have found [1] that inserting a delay ΔT between a) the present state S(T) of a particle pi and the states I(T) of the particles with which pi interacts, and b) the module computing the next state of particle pi, does not affect the time and the processes experienced by the internal observers of the computational universe. The related figure is shown above. In this figure we have added an input E(T) coming from the entangled counterpart of pi (particle pj). By selecting ΔT to be equal to the meta-time interval required for the communication of the computation modules (cells) producing the states of pi and pj, we observe that the computation of the next state of pi

uses the states of pi and pj corresponding to the same meta-time instance T. This fact, in combination with the fact that adding a delay ΔT does not affect the time experienced by the internal observers of the universe, implies that for this observers a measurement on pj has instantaneous impact on the state of pi.

**Conclusions and further discussion**

This paper presents the second part of a computational universe vision. The first part of this vision (described in a companion paper also presented at the 6th AISB Symposium on Computing and Philosophy) proposes a simple model for the computational universe, and also develops a qualitative framework for supporting it. To start, it shows a fundamental limit of knowledge for observers that are part of the observed system (in the sense that they are constituted by the same elementary entities/particles as the ones composing any other structure of the system). It results that the sensorial systems of such internal observers can receive information concerning the behaviour of these elementary entities but by no means information concerning their veritable nature. Then, in a universes-like system, its internal observers have no means allowing distinguishing whether their perceptions correspond to a universe composed of "veritable" particles immersed in a "veritable" space or to a universe engendered by a computation-like process. Thus, computational universe models able to produce the observable behaviour of the physical systems represent credible visions of the universe.

The companion paper also elaborates a qualitative framework dealing with the emergence of space and time perceived by the internal observers of a system/universe (internal space and internal time). In the present paper (second part of the proposed computational-universe vision), we employ this qualitative framework, in order to elaborate the computational-universe versions of special relativity and of quantum mechanics. Adopting such a framework leads to the abandon of existing interpretations of these theories (which were developed on the basis of a "physical" universe vision composed of veritable particles immersed in a veritable space and evolving with the flow of a time). The implications are two fold. On the one hand, as we are free from the conceptual constraints of a "physical" universe, certain contradictions in the existing interpretations of the theories of physics could be resolved. On the other hand, as our theories will need to conform constrains of computational models, the capability of a theory to conform such constraint could be used as criterion of its credibility. For instance, the interpretation of quantum mechanics stating that during coherence the state of a physical observable A consists in the superposition of all the eigenvalues of its operator Â, is not compatible with a computational model since the operators of certain physical observables have infinite number of eigenvalues.

Applying these ideas to the special relativity leads to a computational model where the internal space and time emerging in the computation comply with this theory (i.e. all measurements performed by internal observers obey Lorentz transformations). But at the same time it exists a fundamental time (the meta-time that paces the computation), according to which events can be ordered in a unique manner into past, present, and future. This allows re-establishing an objective time-flow missed in special relativity and can resolve a question raised by numerous philosophers [5][6], which is summarized in the foundation text of the International Conference on the Nature and Ontology of Spacetime [7]:

"A 3D world requires not only a relativization of existence, but also a pre-relativistic division of events into past, present, and future. Therefore, it appears that such a world view may not be consistent with relativity. However, the alternative view – reality is a 4D world with time entirely given as the fourth dimension – implies that there is (i) no objective time flow (since all events of spacetime are equally existent), (ii) absolute determinism (at the macro scale), and (iii) no free will. It is precisely these consequences of the 4D world view that make most physicists and philosophers agree that a world view leading to such implications must be undoubtedly wrong. But so far, after so many years of debate, no one has succeeded in formulating a view that avoids the above dilemma and is compatible with relativity."

Concerning quantum mechanics, we developed a computational model, which reproduces the behaviour of quantum systems by means of deterministic computations performed over stochastic signals, and at the same time eliminates the need of infinite number of variables, as would be required for certain physical observables if we had integrated in our model the state of superposition. Last but not least, we have shown that a computation model that does not support instantaneous communication can engender a computational universe compatible with quantum non-locality.

**References**

[1] Nicolaidis Michael, "The scandal of the computational universe; First part: the qualitative concepts", 6th AISB Symposium on Computing and Philosophy, April 2013, UK

[2] Nicolaidis Michael, "Lorentz Transformations: structure of space-time or implication of interaction laws and measurement means?",http://tima.imag.fr/publications/files/rr/2013/lts_178.pdf

[3] Nicolaidis Michael, "On the State of Superposition and the Parallel or not Parallel nature of Quantum Computing", http://tima.imag.fr/publications/files/rr/2013/qcc_179.pdf

[4] Nicolaidis Michael, "On the State of Superposition and the Parallel or not Parallel Nature of Quantum Computing: a controversy raising point of view", Proceedings AISB'11 Convention, April 4-7, 2011, York, UK

[5] Lusanna, L., Pauri, M.. (2006). Dynamical Emergence of Instantaneous 3-Spaces in a Class of Models of General Relativity. arXiv:gr-qc/0611045 v1.

[6] Tegmark, M. (2007). Shut up and calculate, arXiv:0709.4024v1 [physics.pop-ph]

[7] www.spacetimesociety.org/conferences/2004/minkowski.html